Coordination Value in Agile Software Development

A Multiple Case Study of Coordination Mechanisms Managing Dependencies

Viktoria Stray
University of Oslo
SINTEF Digital
Norway
stray@ifi.uio.no

Nils Brede Moe
SINTEF Digital
Trondheim
Norway
nilsm@sintef.no

Diane Strode
Whitireia Polytechnic
Porirua City
New Zealand
diane.strode@whitireia.ac.nz

Emilie Mæhlum
Blank
University of Oslo
Norway
em@blank.no

ABSTRACT

Background: Agile software projects involve a high degree of coordination between project members to manage complexity and frequent change. There is a need to understand what coordinating mechanisms are valuable for project and team coordination. Coordination mechanisms, such as meetings and Slack, can foster a smooth workflow but also fragment work by interrupting the focused work of developers. *Objective*: This study aimed to investigate valuable coordination mechanisms and how they can be balanced against the need for uninterrupted work periods. *Method*: We conducted 30 interviews and observed 109 meetings in five companies using agile software development methods. We used coordination-dependency mapping to identify valuable coordination mechanisms. *Results*: Valuable coordination mechanisms included instant messaging tools, daily stand-up meetings, boards, open work area, Scrum of Scrums, bug crush days, BizDev meetings, and Make it Happen meetings. *Conclusion*: We advise companies to identify valuable coordination mechanisms using coordination-dependency mapping and then to bundle, schedule, and substitute these coordination mechanisms to reduce interruptions to development work.

CCS CONCEPTS

• Software and its engineering ~Software creation and management

KEYWORDS

Meetings, Interruptions, Fragmented work, Communication, Theory, Teamwork, Large-scale agile, Start-up

ACM Reference format:

Viktoria Stray, Nils Brede Moe, Diane Strode and Emilie Mæhlum. 2022. Coordination Value in Agile Software Development. A multiple case study of coordination mechanisms managing dependencies. In *Proceedings of CHASE'22: 15th International Conference on Cooperative and Human Aspects of Software Engineering, 10 pages,*
https://doi.org/10.1145/3528579.3529182.

CHASE'22 , May 21–29, 2022, Pittsburgh, PA, USA

ACM ISBN 978-1-4503-9342-3/22/05.
https://doi.org/10.1145/3528579.3529182

1 INTRODUCTION

To handle complexity and frequent changes, agile software projects need a high level of coordination between project participants [15]. This coordination depends on the context the teams work in. For example, small co-located projects have different coordination needs than e.g., large-scale, distributed, virtual, or hybrid projects. Therefore, teams must reflect and adjust how they coordinate their work and what agile practices are effective for achieving effective coordination. Additionally, agile teams need to balance individual work activities, such as programming, with team activities that are primarily for coordinating complex team action [31].

Developing software is complex because of the additional need for teams to coordinate their work with multiple experts, stakeholders, and other teams. Further, agile software projects involve frequent changes in customer needs, business models, and technology, including system architecture. Managing the dependencies between these interconnected factors is critical [5]. However, it is not enough to put individuals together and expect them to know how to manage all dependencies; there must be some mechanisms in place to achieve coordination [3]. Coordination mechanisms in software development include meetings, communication tools, roles, documents, code repositories, and how the project and teamwork are organized [7, 30].

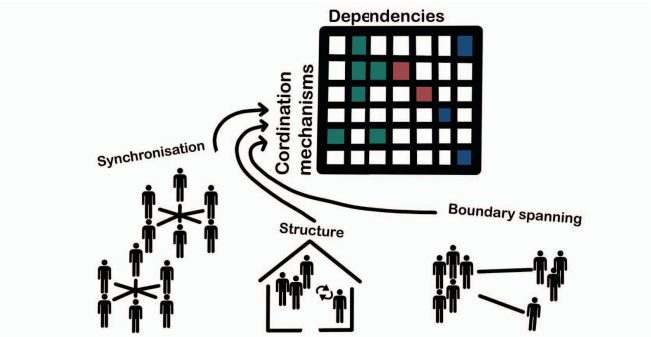
Understanding how to manage dependencies in agile projects may assist product leaders, managers, and developers develop better work habits and more successful projects by selecting the proper mechanisms from among the many mechanisms available. Strode [29] proposed that dependencies can be managed well, poorly, or not at all. Unmanaged dependencies can constrain or block progress, leading to delays as people wait for resources, features to be completed, or requirements to be clarified. When dependencies are poorly managed, coordination mechanisms are usually inappropriate, inadequate, or absent. Conversely, when dependencies are well managed, this suggests that appropriate coordination mechanisms are in place to support the smooth flow of interdependent work and that there will be fewer interruptions and reduced waiting time.

Table 1. Eight dependency types and how they can be managed [30]

	Type	Description	Example
Knowledge dependency— when a form of information is required	Expertise	Technical or task information is known only by a particular person or group, and this affects or has the potential to affect project progress.	Expertise dependencies are managed in daily stand-up meetings when people identify the expertise of other project members that is needed to solve obstacles.
	Requirement	Domain knowledge or a requirement is not known and must be located or identified, and this affects or has the potential to affect project progress.	Requirement dependencies are managed when team members become aware of details of user stories in sprint planning meetings.
	Task allocation	Who is doing what and when is not known, and this affects or has the potential to affect project progress.	Task allocation dependencies are managed when the team members in daily meetings discuss who should be involved in solving the tasks and obstacles raised.
	Historical	Knowledge about past decisions is needed, and this affects or has the potential to affect project progress.	Historical dependencies are managed when a senior developer shares important information about architectural principles and how they emerged.
Process dependency— when a task must be completed before another task can proceed	Activity	An activity cannot proceed until another activity is complete, and this affects or has the potential to affect project progress.	Process dependencies are managed when the board displaying stories and tasks acts as an indicator of potential activity timing conflicts.
	Business process	An existing business process causes activities to be carried out in a certain order, and this affects or has the potential to affect project progress.	Business process dependencies are managed when the backlog is prioritized according to the order occurring in the business process.
Resource dependency— when an object is required for project to progress	Entity	A resource (person, place, or thing) is not available, and this affects or has the potential to affect project progress.	Entity dependencies are managed when the IT support team is available and can be approached directly when needed.
	Technical	A technical aspect of development affects or has the potential to affect project progress.	Technical dependencies are managed when the continuous integration and automated unit test system ensure that changes to one part of the code base have a minimal impact on other parts (API-first design approach).

Dependencies in agile software projects are handled primarily by mutual adjustment [7, 19], with meetings as the primary mechanism. Meyer et al. [13] found that developers spend around five hours in planned meetings during a typical work week. Stray and Moe studied global agile teams and found that the project members spent approximately eight hours per week in scheduled meetings and nine hours in unscheduled meetings [26]. Despite being essential to coordination, meetings often result in interruptions and a temporary suspension of primary activities such as programming [1]. Immersion in coding tasks is vital for good-quality software, the progress of tasks, and developers' perceived productivity, and getting into "the flow" is difficult when developers experience fragmented work because of coordination activities [13]. After an interruption, developers returning to programming tasks need 15 minutes to collect their thoughts and make the first edit [21]. Because of this, companies need to identify the most valuable coordination mechanisms and organize them to reduce the interruption problem. They need to identify the most useful meetings to conduct, the best way to structure teams, the most effective tools to use, and the optimal number, type, and timing of activities for interacting with stakeholders and other teams (e.g., balancing synchronous and asynchronous coordination). Therefore, this study addresses the following research question: *RQ1: What are valuable coordination mechanisms in agile software projects?* We applied a research-based tool for evaluating coordination mechanisms in a multiple-case study to answer this question.

Evidence from the study reported in this paper is part of a broad study of coordination in agile software development. Among the 30 interviews of this study, 9 were reused from the study reported in [30], 4 were reused from the study reported in [27] and 8 were reused from the study reported in [26]. The remaining 9 interviews were new interviews for this study. This study extends our previous findings. Firstly, the current study combines results from studies in



Synchronization is achieved with activities and artefacts produced and used during those activities. Such activities are performed by project members simultaneously that promote a common understanding of a task or process.

Structure is related to how the project is set up. Availability, proximity, and substitutability are aspects of structure. For example, how available is the customer for clarifications, are team members continually present.

Boundary spanning occurs when someone within the project interacts with other organizations, business units, or stakeholders to achieve project goals. Boundary spanning activities are performed to elicit assistance or information externally.

Figure 1: Three types of coordination mechanisms

Table 2. The five cases

Project	Context – product development	About the company	Interviews and observations
Steel	Steel was a Scrum project we studied in 2010 with a team of 10 that took place in a quasi-public sector organization, which provided information critical to New Zealand and various international agencies.	A commercial service provider with 200 employees in Australia, Asia and Europe.	Interviews (5), Work environment (1), Daily Stand-up (1), Story breakdown meeting (1)
Silver	Silver was a Scrum project we studied in 2010 with a single small team of four that was critical to the country. The purpose of the project was to redevelop a legacy system using new technologies.	A commercial software development firm with 20 employees in New Zealand.	Interviews (4), Work environment (1)
Copper	Project Copper was a Scrum and DevOps project we studied in 2018. The project had seven teams, developing web, mobile and document-handing services for the citizens. The five DevOps teams had full responsibility from developing and testing to deploying to production and monitoring of a feature.	A municipality in Norway with 50,000 employees and 50 organizational units. The municipality has its own development program, which is responsible for integrating hundreds of internal systems.	Interviews (4), Daily Stand-up (12), Demo meetings (6), Sprint meetings (2), Scrum of Scrum meetings (5), Project meetings (7), Workshops (3), Team leader meetings (2), Retrospective (1)
Gold	We studied Project Gold in 2017. Gold had a group of 30 team members in four agile virtual Scrumban teams distributed equally across Norway and Poland (15 people at each site). They delivered a system for companies to improve risk performance and to manage quality, health and safety.	A large Norwegian software company that produces and sells specialized software for the engineering domain. The company also has offices in Poland, Germany, China, Malaysia, the US, and the UK.	Interviews (8), Daily Stand-up (6), Retrospective (7), Demo meetings (1), Scrum of Scrum meetings (1), Workshops (3), Task force meeting (1), Bug Triage (1)
Titanium	A start-up company developing a multisided platform for sustainable consumer behavior in Norway. We studied the teams from 2019 to 2020.	A company founded in late 2016 that has around 20 employees. It was among the top 10 promising Norway-based start-ups to watch in 2020	Interviews (9), Daily stand-up meetings (19), Make it Happen meetings (11), Company-wide meetings (2), BizDev meetings (2), Sprint planning (2), Backlog grooming (1), Tech Team meetings (6), Project meetings (5), Workshop (1)

different countries, different agile contexts, and carried out by different researchers. Using the same coordination-dependency mapping technique in all cases, and finding both the same and new coordination mechanisms, provides stronger evidence, beyond that of single studies, for the utility of coordination mapping for identifying valuable coordination mechanisms in software development. Secondly, this study presents new coordination mechanisms not previously reported. Thirdly, this study presents new ways that agile project teams can organize coordination mechanisms to reduce the problem of work interruptions.

2 BACKGROUND

Complex software projects generate a mesh of dependencies [29]. When many teams and stakeholders are spread across continents, complexity increases [4, 25]. Software projects that do not effectively handle dependencies experience coordination breakdowns [5]. Therefore, managing dependencies is a critical aspect of success in a software project [11].

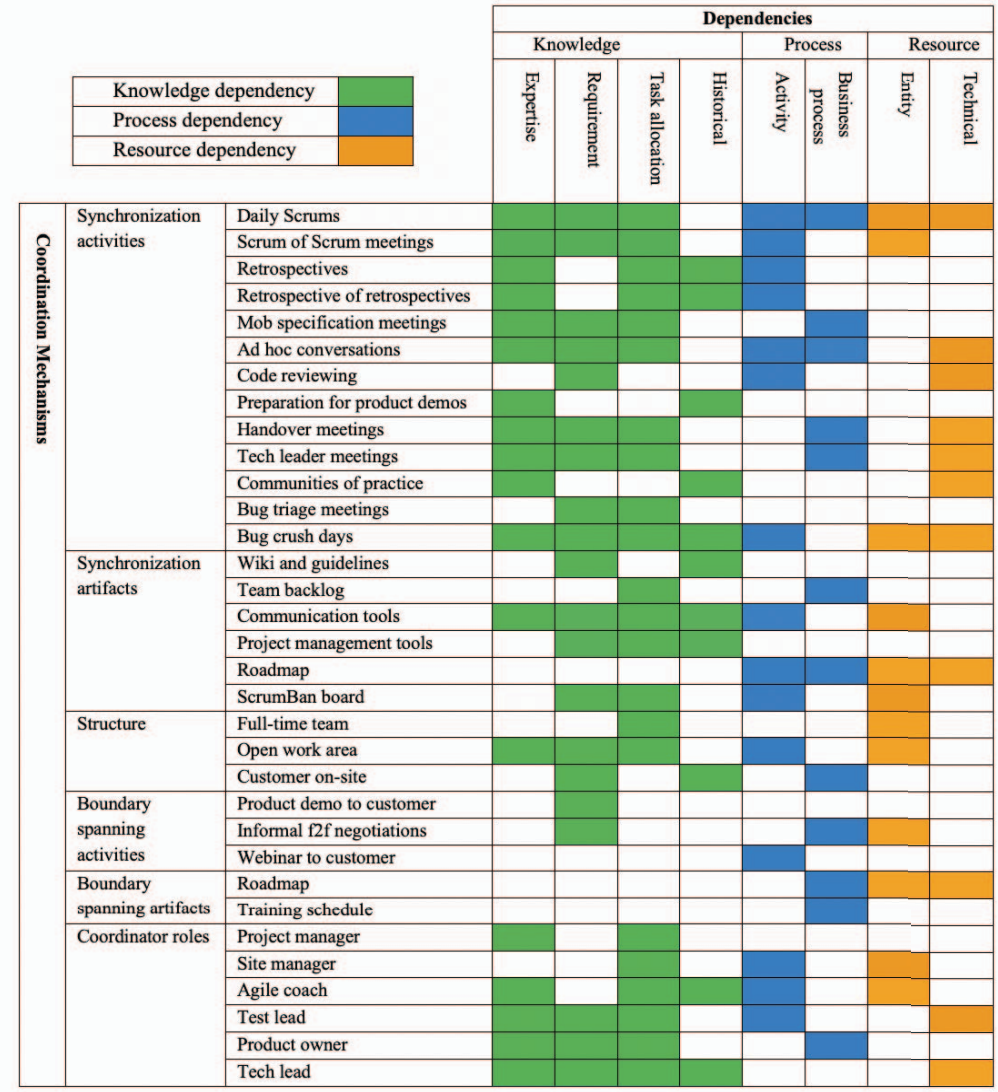
The association between coordination and dependencies is not new. In 1994, Malone and Crowston [12] defined coordination as "the managing of dependencies between activities." Their work led to the idea that agile software development projects need appropriate coordination mechanisms to manage dependencies [30]. Recently, companies are implementing large-scale frameworks, but they still experience coordination challenges [6].

Agile software development projects show three types of dependencies: knowledge dependencies, process dependencies, and resource dependencies [29]. Coordination mechanisms are used to manage these dependencies. For example, an organization can have a daily stand-up meeting where teams in 15 minutes get an overview of who is doing what and can solve problems and address technical dependencies, task dependencies, or knowledge dependencies by discussion and action. There are three types of coordination mechanisms [29]: synchronization, structure, and boundary spanning, see Figure 1.

Mapping coordination mechanisms and dependencies (coordination-dependency mapping) provide a way to analyze what coordination mechanisms address what dependencies in a situation [29]. Such a mapping shows which coordination mechanisms address multiple dependencies; these coordination mechanisms are potentially more valuable in a project because one coordinative activity (e.g. a stand-up meeting) or artefact (e.g. a task board) can address many dependencies. Table 1 defines these dependencies, their eight subtypes, and give examples.

While effective coordination mechanisms are vital for managing dependencies, coordination potentially leads to interruptions and fragmented work. When developers are faced with fragmented work as a result of coordination, such as attending a meeting, they believe their productivity is impaired [13]. A meeting represents a special type of interruption which can lower the employees' later motivation by influencing their mental state [23]. After an interruption, people have to scan and evaluate any new information they encounter and several short interruptions have a greater effect than one long interruption [34]. When employees feel that meetings constrain their time and are unproductive, they get dissatisfied at work [2]. However, meetings are mostly perceived as unproductive during development phases, while during, for example, planning phases, they are seen as more beneficial [14]. Therefore, the project phase also influences which coordination mechanism that gives the most value.

Table 3. Dependencies and coordination mechanisms identified in Gold



3 RESEARCH METHODOLOGY

We chose five cases of agile software development from different companies to study the coordination mechanisms used and the dependencies they managed. We chose these five cases because we wanted to study coordination in different contexts. Case studies were appropriate to address our research question because they provide in-depth detailed information about how work is carried out in complex ongoing situations such as software development. We followed the guidance for quality cases studies in Yin [32] and Runeson and Höst [24] in the design and execution of the study.

The cases selected were active agile projects at the time of data collection. Each case was in a different company and had different contexts. Two of the cases (Steel and Silver) were early colocated Scrum projects. Two of the cases were in large-scale settings (Gold and Copper), while Titanium was a start-up company. The companies were in New Zealand and Norway.

3.1 Data Collection

We collected the data from 2010 to 2020. The context of the cases and the details of the data collected are shown in Table 2. Our source data included interview transcripts, field notes taken during observations of meetings, project documents, photographs of work sites, and Slack logs. Interviews and observations were the primary data sources with other data used to understand the case team and project and as supporting evidence for mechanisms and dependencies in the case. In each company, we interviewed people in different roles including project leader, developer, business analyst, domain expert, or tester.

The interviews began with asking for role and experience information. Then the participant was asked to describe the main work activities they carried out in their project and their responsibilities, followed by questions regarding communication, coordination, and dependencies. Appendix A shows the interview guide.

3.2 Data analysis

The data analysis followed the guidelines of Miles and Huberman [16]. We also closely followed the procedure for identifying coordination mechanisms and dependencies in each case as described in [29, 30]. We carefully checked that the dependencies and coordination mechanisms we identified met the definitions provided in [30] and that new coordination mechanisms or dependencies were included in the results. An iterative procedure was followed to constantly check and revise the allocation of data (quotes from transcripts, observations, or other evidence) to dependencies and coordination mechanisms and that there was evidence for an association between each dependency and mechanism we identified. To analyze the data, all the interviews were transcribed, and we performed the data analysis using NVivo. The observation notes and meeting minutes were included in the analysis and treated as transcripts. First, each transcript was read to identify dependencies and how they were managed (i.e., what coordination mechanism was involved). Then the transcript was read again to identify coordination mechanisms and what dependency was being addressed. If new dependencies and coordination mechanisms were identified, then transcripts were reviewed again to look for instances of these dependencies. In this way, the transcripts were read and checked more than once. An example of the analysis is as follows. The following quote from Steel has the identified coordination mechanism and dependency underlined. "*Most of the time … that whiteboard will stay up for a few days before the next design discussion happens or the next whiteboard discussion happens, and then, in that case, people can look over their shoulder and see what they drew, and get on with actually coding to that plan…*". 'whiteboard' was coded as a coordination mechanism of type 'synchronization artefact' and named 'whiteboard'. The dependency is also underlined in the quote. The dependency was coded as a knowledge dependency of type 'requirements' because the team members could all refer to this one model to allow them to proceed with coding the required model. Thus the whiteboard was an artefact that synchronized the knowledge of the team about the design (which we considered to be the detail of a requirement), enabling them all to have a common understanding of what was needed.

In the final stage of analysis, we created a coordination dependency map for each case in spreadsheets. This spreadsheet mapped each identified coordination mechanism with its associated dependency. Some dependencies had no coordination mechanisms and some coordination mechanisms could not be matched to a dependency. These dependencies and coordination mechanisms were not included.

4 RESULTS

In the five cases, we identified more than 60 coordination mechanisms that address one or more of the eight dependencies shown in Table 1. Coordination mechanisms that address multiple dependencies are potentially more valuable than other coordination mechanisms. Table 3 shows how we used coordination-dependency mapping in the project Gold to identify valuable mechanisms. The maps for Silver and Copper are available online (doi.org/10.6084/m9.figshare.13850414).

In the next section, we first present the two ubiquitous coordination mechanisms Instant messaging tools and Daily stand-up meetings. These mechanisms managed the most dependencies across all the cases (at least six). Then we present two coordination mechanisms from each of the three different agile contexts we studied: co-located (Steel and Silver), large-scale (Copper and Gold), and start-up (Titanium). For coordination in the large-scale projects, two inter-team practices were valuable: Scrum of Scrum meetings and bug crush days. In the start-up company Titanium, two valuable coordination practices were BizDev meetings and Make it Happen (MiH) meetings. All these coordination mechanisms we considered valuable because they addressed four or more types of dependencies.

4.1 Ubiquitous coordination practices

Instant messaging tools for ad hoc conversation. Slack, and similar tools, enabled informal communication and fast feedback across teams and sites and provided transparency regarding what people were doing and who should do what (i.e. addressed task allocation dependencies). These tools reduced the waiting time for information and missing resources. Surprisingly, even in projects with open work areas, Slack was used to ask fellow team members questions when a team member did not want to approach and disturb another team member (for example, they saw that the person was concentrating when working with a headset). Slack managed entity dependencies because project members could reach out to the whole project. One developer in the project Gold stated, *"I find that Slack is much more efficient than e-mail. The team members know and respond immediately when someone posts a screenshot on Slack."*

Some teams used Slack for spontaneous conversations, which made it a mechanism that initiated and maintained other coordination mechanisms. Such ad hoc conversations were used to get information members needed when solving problems or when requirements were unclear, and they often led to unscheduled meetings. Further, because the conversations were spontaneous, they reduced waiting time. In distributed teams, we found instant messaging tools overcame the problem of not being co-located. Ad hoc conversations on Slack were initiated to get help on a problem, gain access to a service, or request specific information. For example, the following messages were sent one early morning on Slack in Gold:

Person 1: Anyone able to reach https:## at the moment? I get Runtime Error
Person 2: Same for me
Person 3: I sent an e-mail to NN

Within 30 minutes, the problem was solved. Slack logs can be searched and therefore support handling historical dependencies. A developer in Titanium stated, "*I usually communicate via Slack because I am not going around trying to find people. It is easier to just write to them if I need something, and then I have it documented somewhere.*"

The daily stand-up meeting. This meeting was vital in all projects for managing internal team dependencies, especially task allocation and expertise dependencies (i.e., knowing who knows what). The project members stated what they planned to work on and explained the obstacles they faced. Explaining an obstacle often caused another team member to share a solution. Team members often raised obstacles that were of types process dependencies (having to wait for other teams/persons to complete a module) and resource dependencies (having to wait for information or technical bugs to be solved). The meeting also enabled the team leader to be aware of and manage the dependencies with other stakeholders. The meeting often ended by coordinating tasks in a discussion about who should be involved in solving the task and the obstacles involved (task allocation dependencies). One developer in Titanium stated, "*At first I didn't really like the daily stand-up meetings, but now I see that they give you a good idea of what the team as a whole is working towards, and it's good to know in case what you are working on might affect other team members' work, so you can collaborate and come up with some solution.*" The team members also said that problem-focused communication was the most important part of the daily stand-up meetings because it helped them identify dependencies and make decisions more effectively. Most teams in Copper had their daily stand-up meeting scheduled right before lunch. This time choice reduced the number of interruptions because the team members would go to lunch together immediately following the meeting.

4.2 Co-located coordination practices

Open work area. In an open work area, when project members were unsure about the details of requirements, it was easy to talk to a specialist or to the customer representative. As one developer in Steel commented, "*We have a person on our team, who is one of the gurus, and he sits with us, and we annoy him constantly …*" In the large-scale projects, it was possible to walk between teams to solve inter-team dependencies and get help from other experts. One product owner in Copper stated, "*By sitting in the same location as the customer and a short distance from the DevOps teams, it is possible to make important decisions through fast, informal conversations.*" Moreover, with meeting rooms just a few meters from the seating arrangements, it was easy to organize spontaneous meetings to discuss project matters and come to common understandings. Open work area was a coordination mechanism that managed knowledge, process, and resource dependencies.

Boards. Readily visible boards on display in the open work area were artefacts for coordinating activity dependencies by giving information about when others had completed their tasks. The boards also managed task allocation dependencies because all project members could see who was working on what. Boards were also used in daily stand-ups, showing coordination mechanisms are interconnected. In project Silver, the board showed avatars of the project members on each of the tasks. A developer in Silver commented on how he knew who was working on what, "*So, normally, that would be the task wall you know, 'where are the avatars?' to quickly see what people are working on.*" The teams also used the boards in the meetings when discussing requirement dependencies.

4.3 Large-scale coordination practices

Agile software development relies on mutual adjustment. However, mutual adjustment in its pure form requires everyone to communicate with everyone, which is a challenge in large-scale projects. To cope with the coordination challenge in large-scale projects, many projects introduce a plethora of meetings, which can lead to an overwhelming number of meetings. The key to success is identifying the most valuable meetings and combining them with other practices. We now present Scrum of Scrums and bug crush days – coordinative practices that were combined with Slack and the daily meetings.

Scrum of Scrums. This was a weekly scheduled inter-team synchronization meeting where one representative from each team met. In Copper, the Scrum of Scrums meeting ensured that relevant information and knowledge challenges from teams and their key stakeholders were shared, to ensure solutions integrated well and both technical and process impediments were quickly solved. Eight team representatives met with the test lead, architect, security manager, and project manager for a maximum of one hour. By including relevant experts who formed a cross-functional team, the meeting managed dependencies, such as when progress was blocked as people waited for resources or necessary information or for the activities of others to be completed. A team leader explained, "*We were supposed to integrate with other parts of the system, but they had not completed their part and had not granted us access. It is frustrating when we are done but we have to wait for others to finish.*" Knowledge and expertise dependencies were also identified and solved in the meeting by the cross-functional team.

Bug crush days. These days are an example of synchronous work sessions where several project members focus on the same task to increase speed. In Gold, testers registered bugs to be fixed by the developer, who solved the bugs and passed the solution back to the testers. Because developers were then working on other issues, there was a lot of waiting, and it could take weeks to fix a bug. If the testers and developers needed to talk to each other, one of them was often not available. To solve this problem, Gold introduced a bug crush day, where everyone worked on closing bugs every other Tuesday. The bug crush day allowed testers and developers to coordinate easily by getting fast feedback, reducing the need to hold meetings or wait for responses. Because the amount of context switching was reduced, the time to solve a

specific task was shortened. A secondary effect was that the bug count was kept low and fixing new bugs could be postponed until the next bug crush day, reducing interruptions to ongoing work.

4.4 Start-up coordination practices

Software start-up companies have unique characteristics such as cross-functional units operating under highly uncertain conditions, often under extreme time pressure, which pose several coordination challenges [22]. In Titanium, the development team had meeting-free days to focus on programming and designing without interruptions. A designer in Titanium explained how he appreciated the practice: "*It's easier to think more clearly when you know which days you have to get things done, and it's easier to prioritize the tasks to work on*". It also allowed for more unplanned meetings and ad hoc conversations from Tuesday to Thursday. It also made it easier for the rest of the organization to schedule meetings with the development team since they knew which days they were available for meetings. To ensure cross-company coordination, Titanium relied on BizDev meetings and MiH meetings together with ubiquitous coordination practices to increase the speed of decision-making and make sure everyone was working towards the same goal, which will be described next.

BizDev meetings. At these scheduled meetings, representatives from the development team and from the business departments, such as marketing, operations, and finance, came together every week or other week to solve challenges, plan and prioritize for the following weeks, and align the upcoming tasks and projects. They also evaluated completed software features and tried to manage dependencies for tasks that were delayed. At Titanium, these meetings were usually held on Fridays at 11:00, lasted less than 30 minutes, and were supported by Google Meet with screen sharing, an Excel sheet with chosen tasks for the upcoming period, and Zendesk with issues from both customers and others in the organization. A designer said, "*It helps a lot with our process that we all use Zendesk and go through the issues together and plan for the next week.*"

The frequent meeting with all key people in the company reduced handover problems and enabled a continuous planning process and alignment of the business and development teams. We observed that the atmosphere in this meeting was good and that there was a lot of laughter. In the start-up, this meeting was valuable for handling dependencies between the different teams, as it dealt with a total of six dependencies. This meeting addressed an activity dependency because everyone learned that the developers had to implement the possibility for a discount code before the marketing campaign could be launched on Instagram. The meeting helped align the development team's plans with the plans of the other departments such as Marketing.

Make it Happen (MiH). These weekly scheduled meetings were held at 12:00 and involved all employees in the start-up. The goal of the meeting was to align the different departments in the company on what had been done and what was committed to for the upcoming week. The meeting was held on Google Meet as the departments were distributed and the CEO was sharing her screen with a Google presentation. The departments had filled in their information in the presentation before the meeting. They always started the meeting by mentioning the goal for 2020 and a representative from each department (tech, marketing, operations, and finance) stated what had happened last week and what they were going to do in the coming weeks. The MiH meetings usually lasted between 15 and 30 minutes and were particularly important for managing expertise and task allocation dependencies. One developer expressed how he appreciated the MiH meetings: "*We have these weekly company-wide meetings where we hear from all the different departments and know what's going on. That's really helpful because you feel more motivated when you see that everything is in place for us to work together and get to a certain goal.*" A senior developer stated, "*The overview we get from the MiH meetings is more than enough to know what others outside the teams are working on.*"

5 DISCUSSION

We set out to identify valuable coordination mechanisms in agile software development projects. We used coordination-dependency mapping to understand what mechanisms were used, what dependencies they managed, and which mechanisms were most valuable for coordinating a project. In five independent case studies we found and described eight coordination mechanisms that all addressed more than four types of dependencies and are therefore valuable in coordinating agile software projects. These coordination mechanisms were as follows:

- Ubiquitous in all situations
  - Instant messaging tools for ad hoc conversation
  - The daily stand-up
- For colocated situations
  - Open work area
  - Boards
- For large scale coordination mechanisms
  - Scrum-of-Scrums
  - Bug crush days
- For startups
  - BizDev
  - MiH

A coordination-dependency mapping gives a picture of those coordination mechanisms that are valuable because they are managing multiple dependencies. However, just prioritizing the coordination mechanisms according to how many dependencies they manage is not enough. To increase the value of coordination mechanisms, organizations and their agile teams need to consider how to arrange coordination activities to reduce their potential to interrupt development work. Our findings show that organizations can *bundle* coordination mechanisms together in time, *schedule* them at appropriate times, and *substitute* mechanisms if they both address the same dependency.

Bundling. While migrating to a strategy based on ad hoc coordination seems to be a good solution, it is still crucial to balance the need to reduce waiting time and get fast feedback against being

interrupted too often. One solution is to implement meeting-free days and days allocated for meetings. This practice reduced work fragmentation in Titanium. There had to be a very good reason to invite tech people to a meeting on other days than Mondays and Fridays. Consequently, the developers could focus on programming for whole days, and it became easier to plan how much time they had available for the development tasks. Whillans et al [31] argue for the need to investigate which team activities need to be scheduled and which can be allowed to occur on an ad hoc basis.

Scheduling. In particular, there is a need to pay attention to the scheduling of synchronization activities, usually meetings, to further reduce interruptions. For example, when the daily meeting is held in the morning, some team members wait until after the meeting to start working on tasks that need concentration to avoid work fragmentation [28]. In Copper, teams held the meeting right before lunch to reduce the number of interruptions, and in Titanium, teams sometimes conducted the daily stand-up meeting by writing to each other in Slack.

Substitution. Mutual adjustment, which is coordination by feedback between project members [17], is vital in agile software development [7, 19] but there are different ways to achieve mutual adjustment. Our findings suggest that projects and teams could rely on using instant messaging tools for mutual adjustments, both in co-located and distributed settings. Further, implementing common work sessions, such as bug crush days, reduced the need for scheduled, more formal, meetings to achieve mutual adjustment. If projects can reduce the number of meetings, the speed of development increases because project members are interrupted less often and are available for quick clarifications. If key experts are in meetings all day, the process becomes slow. Since distance makes team members communicate less [20], organizations must find ways to foster mutual adjustments in the virtual and hybrid settings that agile teams are often required to work in today.

In this study, we call mechanisms that address multiple dependencies 'valuable'. However, some agile practices that have a coordinative function also serve other useful purposes in a project or in a team (e.g. a product demonstration meeting can improve the relationship with a customer; pair programming improves the quality of code). Our claims about the value of a practice are only related to its value for coordinating. Practices with limited coordination value might have other values we have not considered.

5.1 Key takeaways

We have demonstrated the benefits of mapping coordination mechanisms (e.g. artefacts such as boards and practices such as BizDev meetings) to the different types of dependencies that occur in agile software development projects.

We argue that prioritization should be given to mechanisms that can reduce work fragmentation. Coordination has a time and effort overhead and can interrupt workflow. Therefore, if a mechanism addresses few dependencies, involves many people, requires significant time, or creates many interruptions, it is a candidate for being removed, replaced, or changed—for example, by reducing the frequency of a specific meeting.

A practice that was vital for the coordination of dependencies was the use of instant messaging tools. Tools such as Slack can replace one or more coordination mechanisms (e.g., a stand-up or ad hoc meeting), support both synchronous and asynchronous coordination, and can be adjusted to the needs of the individual. Further, Slack increased the awareness of what others were working on. Earlier studies have also found that instant messaging tools make communication more transparent and open [8]. When team members know what others are doing it is easier to initiate contact, which is especially important when project members are working distributed [9, 26]. Therefore, Slack supports both frequent feedback and monitoring, which is key for good teamwork [18]. Open communication is also crucial for start-ups because of the need to respond fast to changes, use limited resources in an optimal way by effectively handling engineering activities, reduce misunderstandings and confusion, and improve the understanding of progress, code conflicts, and competences [22]. Relying on tools that can be adapted to a context is beneficial for coordination, because people have different sensitivities to interruptions depending on the type of work they are doing, and tools may also be adaptable to accommodate changing coordination needs over time.

Start-up companies are known to rely on hypothesis-driven development and continuous experimentation [10]. They need to quickly analyze the data from experiments, organize frequent feedback loops, and make new decisions. Frequent cross-company coordination and dialogue are therefore central for such companies to develop successful products and services. Further, start-ups are typically small and must align their business and technology strategies to avoid waste of resources [33]. Open communication was facilitated in our start-up case with the practices of MiH and Bizdev meetings. These meetings were considered vital for open communication and the handling of dependencies across departments. Developers view time spent in meetings as inefficient, but still perceive most meetings to be valuable [14]. Further, the use of Slack was vital for effective coordination, especially to manage resource dependencies.

Meeting duration is a key consideration in large-scale projects. Our large-scale cases often coordinated in Scrum of Scrum meetings. These meetings needed to have a long enough duration to enable participants to discuss problems and achieve the managing of dependencies. Getting together for only 15 minutes was not adequate because this would initiate several extra meetings to resolve issues raised.

5.2 Limitations, Validity and Reliability

This empirical research has limitations. The case study findings are not necessarily applicable or generalizable to all agile software development contexts, although we chose a range of cases so our findings should have some relevance to similar contexts. During observations, the presence of researchers can be intrusive and change the behavior of the meeting participants. Nevertheless, we

consider this effect to be small since most of the teams were observed over a long period of time. Another caveat is that the number of observations per meeting type likely influenced the number of dependencies that were managed by that particular meeting. For example, if we had observed more planning meetings, we might have found that the practice manages more dependencies. Also we might have identified more or different dependencies if we interviewed more people or engaged with the organizations for longer periods.

We designed this case study to achieve construct and external validity and reliability [16, 24, 32] Internal validity is concerned with causal relationships and is not relevant in our study. To identify the constructs, which are dependencies and coordination mechanisms, we followed the same data analysis procedure in each case. Although this procedure is prone to the analysts' subjective interpretation, we mitigated this by analyzing the data material according to an existing framework that defines common types of mechanisms [30] and dependencies [29]. To strengthen construct validity, we shared draft findings for each case with the research participants so they could comment on their relevance. We used triangulation: of cases by choosing to study more than one case; of participants by including a range of people in different roles to interview and a range of meetings to observe within each case.

External validity in case study research is concerned with analytical generalizability. We carefully selected a range of cases that were typical of agile software development and also unique in some respects and found that the theory we employed was supported, that dependencies are associated with coordination mechanisms in agile software development. Our selected cases have characteristics common in agile software development situations, so our findings may be useful for organizations in similar situations. Reliability is concerned with the extent to which the data and analysis are independent of the specific researchers. By using the same unit of analysis (employing case selection criteria), and analysis procedures in each case we aimed for a degree of researcher independence and consistency in the study.

6 CONCLUSION

Our multiple-case study shows that coordination mechanisms are integrated; they support each other and are developed and co-evolved in the organizational environment. This ecosystem of agile practices, tools, meetings, roles, and artefacts must be evaluated continuously. Mapping the value of each mechanism and understanding their interactions and how they can be better organized to reduce work interruptions can help complex agile projects to succeed. We have shown that some coordination mechanisms have a higher value than others: communication tools such as Slack, Scrum of Scrums, bug crush days, BizDev meetings, and Make it Happen meetings. We highlight at least three ways that companies can organize coordination to reduce interruptions: by bundling coordination mechanisms in time (having meeting-free days), by scheduling them to suit the situation (by adjusting the timing of meetings), and by substituting one coordination mechanism for another (when two mechanisms address the same dependency). Overall, this study shows that organizations can foster spontaneous coordination and reduce disruptions to development work by identifying how they coordinate and then taking considered steps to balance coordination work and development work so that developers have significant time to focus on development.

Future work on coordination could strengthen the success of agile software development organizations. For example, conducting research to understand the dependencies that commonly occur in software development and identify all of the possible alternative coordination mechanisms that can manage those dependencies. Studies of how organizations optimize coordination, by bundling, scheduling, substituting, or other means, to reduce the coordination burden on projects and teams would be valuable. Finally, by studying how coordination needs change over the lifecycle of a software development project, we can design flexible coordination tools and ensure that coordination is minimized but effective.

ACKNOWLEDGMENTS

We acknowledge and appreciate the time and effort of our research participants in contributing to this study. We thank Andreas Aasheim for helping collect data. This work was supported by the Research Council of Norway (grant 267704 and grant 309631).

REFERENCES

[1] Addas, S. and Pinsonneault, A. 2018. Theorizing the Multilevel Effects of Interruptions and the Role of Communication Technology. Journal of the Association for Information Systems. 19, 11 (2018), 1097–1129. DOI:https://doi.org/10.17705/1jais.00521.

[2] Allen, J.A., Sands, S.J., Mueller, S.L., Frear, K.A., Mudd, M. and Rogelberg, S.G. 2012. Employees' feelings about more meetings: An overt analysis and recommendations for improving meetings. Management Research Review. (2012).

[3] Berntzen, M., Hoda, R., Moe, N.B. and Stray, V. 2022. A Taxonomy of Inter-Team Coordination Mechanisms in Large-Scale Agile. IEEE Transactions on Software Engineering. (2022). DOI: https://doi.org/10.1109/TSE.2022.3160873.

[4] Bick, S., Spohrer, K., Hoda, R., Scheerer, A. and Heinzl, A. 2017. Coordination challenges in large-scale software development: a case study of planning misalignment in hybrid settings. IEEE Transactions on Software Engineering. 1 (2017), 1–1.

[5] Cataldo, M. and Herbsleb, J.D. 2013. Coordination breakdowns and their impact on development productivity and software failures. IEEE Transactions on Software Engineering. 39, 3 (2013), 343–360.

[6] Conboy, K. and Carroll, N. 2019. Implementing large-scale agile frameworks: challenges and recommendations. IEEE Software. 36, 2 (2019), 44–50.

[7] Dingsøyr, T., Moe, N.B. and Seim, E.A. 2018. Coordinating knowledge work in multiteam programs: findings from a large-scale agile development program. Project Management Journal. 49, 6 (2018), 64–77.

[8] Giuffrida, R. and Dittrich, Y. 2013. Empirical studies on the use of social software in global software development–A systematic mapping study. Information and Software Technology. 55, 7 (2013), 1143–1164.

[9] Herbsleb, J.D. 2007. Global Software Engineering: The Future of Socio-technical Coordination. Future of Software Engineering (FOSE '07) (May 2007), 188–198.

[10] Khanna, D., Nguyen-Duc, A. and Wang, X. 2018. From MVPs to pivots: a hypothesis-driven journey of two software startups. International Conference of Software Business (2018), 172–186.

[11] Kraut, R. and Streeter, L. 1995. Coordination in software development. Communications of the ACM. 38, 3 (Mar. 1995), 69–81. DOI:https://doi.org/10.1145/203330.203345.

[12] Malone, T. and Crowston, K. 1994. The interdisciplinary study of coordination. ACM Computing Surveys. 26, 1 (Mar. 1994), 87–119. DOI:https://doi.org/10.1145/174666.174668.

[13] Meyer, A.N., Barton, L.E., Murphy, G.C., Zimmermann, T. and Fritz, T. 2017. The Work Life of Developers: Activities, Switches and Perceived Productivity. IEEE Transactions on Software Engineering. 43, 12 (2017), 1178–1193. DOI:https://doi.org/10.1109/TSE.2017.2656886.

[14] Meyer, A.N., Murphy, G.C., Zimmermann, T. and Fritz, T. 2019. Enabling Good Work Habits in Software Developers through Reflective Goal-Setting. IEEE Transactions on Software Engineering. 47, 9 (2019), 1872–1885. DOI:https://doi.org/10.1109/tse.2019.2938525.

[15] Mikalsen, M., Moe, N.B., Stray, V. and Nyrud, H. 2018. Agile digital transformation: a case study of interdependencies. International Conference on Information Systems 2018, ICIS 2018. (2018).

[16] Miles, M.B. and Huberman, A.M. 1994. Qualitative data analysis: An expanded sourcebook. Sage.

[17] Mintzberg, H. 1980. Structure in 5's: A Synthesis of the Research on Organization Design. Management Science. 26, 3 (Mar. 1980), 322–341. DOI:https://doi.org/10.1287/mnsc.26.3.322.

[18] Moe, N.B., Dingsøyr, T. and Dybå, T. 2010. A teamwork model for understanding an agile team: A case study of a Scrum project. Information and Software Technology. 52, 5 (2010), 480–491.

[19] Moe, N.B., Stray, V. and Hoda, R. 2019. Trends and updated research agenda for autonomous agile teams: a summary of the second international workshop at XP2019. (2019), 13–19.

[20] Noll, J., Beecham, S. and Richardson, I. 2010. Global software development and collaboration: barriers and solutions. ACM inroads. 1, 3 (2010), 66–78.

[21] Parnin, C. and Rugaber, S. 2011. Resumption strategies for interrupted programming tasks. Software Quality Journal. 19, 1 (2011), 5–34. DOI:https://doi.org/10.1007/s11219-010-9104-9.

[22] Paternoster, N., Giardino, C., Unterkalmsteiner, M., Gorschek, T. and Abrahamsson, P. 2014. Software development in startup companies: A systematic mapping study. Information and Software Technology. 56, 10 (2014), 1200–1218. DOI:https://doi.org/10.1016/j.infsof.2014.04.014.

[23] Rogelberg, S.G., Leach, D.J., Warr, P.B. and Burnfield, J.L. 2006. " Not another meeting!" Are meeting time demands related to employee well-being? Journal of Applied Psychology. 91, 1 (2006), 83.

[24] Runeson, P. and Höst, M. 2009. Guidelines for conducting and reporting case study research in software engineering. Empirical software engineering. 14, 2 (2009), 131–164.

[25] Sekitoleko, N., Evbota, F., Knauss, E., Sandberg, A., Chaudron, M. and Olsson, H.H. 2014. Technical dependency challenges in large-scale agile software development. International conference on agile software development (2014), 46–61.

[26] Stray, V. and Moe, N.B. 2020. Understanding coordination in global software engineering: A mixed-methods study on the use of meetings and Slack. Journal of Systems and Software. 170, (Dec. 2020), 110717. DOI:https://doi.org/10.1016/j.jss.2020.110717.

[27] Stray, V., Moe, N.B. and Aasheim, A. 2019. Dependency Management in Large-Scale Agile: A Case Study of DevOps Teams. Proceedings of the 52nd Hawaii International Conference on System Sciences (2019), 7007–7016.

[28] Stray, V., Moe, N.B. and Sjoberg, D.I.K. 2020. Daily Stand-Up Meetings: Start Breaking the Rules. IEEE Software. 37, 3 (May 2020), 70–77. DOI:https://doi.org/10.1109/MS.2018.2875988.

[29] Strode, D.E. 2016. A dependency taxonomy for agile software development projects. Information Systems Frontiers. 18, 1 (Feb. 2016), 23–46. DOI:https://doi.org/10.1007/s10796-015-9574-1.

[30] Strode, D.E., Huff, S.L., Hope, B. and Link, S. 2012. Coordination in co-located agile software development projects. Journal of Systems and Software. 85, 6 (Jun. 2012), 1222–1238. DOI:https://doi.org/10.1016/j.jss.2012.02.017.

[31] Whillans, A., Perlow, L. and Turek, A. 2021. Experimenting during the shift to virtual team work: Learnings from how teams adapted their activities during the COVID-19 pandemic. Information and Organization. 31, 1 (2021), 100343.

[32] Yin, R.K. 2018. Case study research and Applications: Design and Methods. SAGE publications.

[33] Yogendra, S. and Sengupta, S. 2002. Aligning business and technology strategies: a comparison of established and start-up business contexts. IEEE International Engineering Management Conference (2002), 2–7.

[34] Zijlstra, F.R., Roe, R.A., Leonora, A.B. and Krediet, I. 1999. Temporal factors in mental work: Effects of interrupted activities. Journal of Occupational and Organizational Psychology. 72, 2 (1999), 163–185.

APPENDIX A

Appendix A, the interview guide, is available here: https://doi.org/10.6084/m9.figshare.19338041