Proceedings of the 52nd Hawaii International Conference on System Sciences | 2019

Dependency Management in Large-Scale Agile: A Case Study of DevOps Teams

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

Nils Brede Moe
SINTEF
nils.b.moe@sintef.no

Andreas Aasheim
University of Oslo
aasheim.andreas@gmail.com

Abstract

Managing dependencies between teams and within teams is critical when running large-scale agile projects. In large-scale software development, work is carried out simultaneously by many developers and development teams. Results are delivered frequently and iteratively, which requires management of dependencies on both the project and team level.

This study explores coordination mechanisms in agile DevOps teams in a large-scale project and how the mechanisms address different types of dependencies.

We conducted a case study where we observed 38 scheduled meetings and interviewed members of five DevOps teams and two teams supporting the DevOps teams. By using a dependency taxonomy, we identified 20 coordination mechanisms (eleven synchronization activities and nine synchronization artifacts). Eight of these mechanisms seem essential for coordination in large-scale projects because they addressed more than four types of dependencies. The main implication is that project management needs to combine many practices handling all the dependencies in large-scale projects.

1. Introduction

Dependency management in large-scale agile software development is of great importance because the work is carried out simultaneously by many developers and development teams. In large-scale projects, defined as projects with two to nine teams [7], there is an exponential growth of interdependencies, and effective coordination is a critical element for success [2, 3, 11].

Furthermore, in large projects, interdependencies are more uncertain than in small projects; therefore, teams need to know who the experts and stakeholders are and which experts and stakeholders to coordinate work with, particularly when they are outside the team or even at a different site. Moreover, agile methods are emergent [4], which means that processes, principles, and work structures emerge during the project rather than being predetermined. As a consequence, dependencies in large-scale agile projects will emerge during a project.

Dependencies in development projects can be managed well, poorly, or not at all, but when managed well, it suggests that appropriate coordination mechanisms are present [30]. Malone and Crowstone [18] define coordination as "the managing of dependencies between activities."

Understanding how to manage dependencies in large-scale agile projects may help product leaders, managers, and developers create better agile behaviors and more successful projects by choosing the appropriate coordinative practices from the large number of agile practices that are employed. While several studies have been performed on dependencies and coordination in small projects, studies on coordination and how to manage dependencies in large-scale agile is lacking [20, 31]. Further, Dingsøyr et al. [9] suggest that there is a need to develop a further understanding regarding coordination mechanisms in large development programs to investigate how coordination mechanisms are tailored to the specific context of a project.

Motivated by the need to understand how to manage dependencies in large-scale agile, we have identified the following research question:

How are dependencies managed in large-scale agile projects?

To address this question, we conducted a case study to investigate the management of dependencies. Our work aims to answer a call for trying out a recently developed taxonomy in a large-scale context [30].

The remainder of this paper is organized as follows. Section 2 outlines the relevant background. Section 3 describes the research methods used. Section 4 reports our results. Section 5 discusses the results, implications for practice and the limitations of the study. Section 6 concludes and suggests future work.

URI: https://hdl.handle.net/10125/60137
ISBN: 978-0-9981331-2-6

Table 1. A description of the eight dependency types based on [30]:

Dependency		Description
Knowledge dependency	Expertise	Knowing who knows what is essential in large projects. Expertise dependencies are managed when people identify the roles and expertise of other team members that are needed. When technical or task information is known only by a particular person or group, this has the potential to affect project progress.
	Requirement	Requirements are a critical input in software development because they define the basic functions and qualities of the software. When domain knowledge or details of requirements are not known and must be located or identified, this has the potential to affect project progress. Prioritizations of requirements in agile projects and customer access are vital.
	Task allocation	Knowing who is doing what is useful information when it comes to managing dependencies in large-scale projects. When it is not known who is doing what or when they are doing it this has the potential to affect project progress.
	Historical	Historical dependencies are defined as the need for organizational memory and knowledge about previous decisions [13]. When knowledge about past decisions is needed, this has the potential to affect project progress.
Process dependency	Activity	When an activity cannot proceed until another activity is complete this may affect project progress. That is, progress is blocked or delayed as people wait for resources, necessary information or the activities of others to be completed.
	Business process	When existing business processes cause activities to be carried out in a certain order, this may affect project progress.
Resource dependency	Entity	Having to wait for information or people to be present is a common issue in large-scale projects. When a resource (person, place, or thing) is not available, this has the potential to affect project progress.
	Technical	Technical dependencies are involved when the presence or absence of a software component that another software component must interact with affects project progress.

2. Background

To understand dependencies in large-scale agile, it is first important to understand autonomous agile teams and coordination in large-scale projects.

2.1 Agile teams

Agile favors self-management (where teams determine the best way to handle the work), emergent processes (processes, principles, and work structures emerge during the project rather than being predetermined), and more informal coordinating mechanisms [4]. Pikkarainen et al. [23] used Malone and Crowston's theory from 1994 to study two small co-located agile projects and found Sprint planning meetings, daily meetings, and open work area to be important for enabling communication and coordination in agile teams. Moreover, Pries-Heje and Pries-Heje [24] identified coordination as one of the critical elements that explained why Scrum worked in a globally distributed agile project, especially the product backlog, scrum board, sprint backlog, and daily meetings were identified for achieving coordination. Moe et al. [21] studied a co-located Scrum project using a teamwork model and found team members "not knowing what others were doing" to be a major problem for the agile team. When team members do not know what others are doing, it is challenging to manage dependencies. The coordination suffered due to misapplication of Scrum practices partially caused by an existing organizational structure that promoted specialization of skills within individuals (ibid). Strode [30] explored dependencies in three co-located agile software development projects and found that coordination mechanisms such as cross-team talk, a product backlog, a done checklist, task breakdown sessions, and a wallboard displaying stories, tasks, and task assignments were important to manage dependencies.

2.2. A dependency taxonomy

The previously mentioned case study on agile projects by Strode [30] addressed dependencies and coordination mechanisms by a dependency taxonomy in co-located projects. The taxonomy identifies three types of dependencies: *knowledge*, *process,* and *resource*. *Knowledge dependency* occurs when team members are waiting for information about a requirement, a task, technical information, a past decision, or because they do not know what other team members are doing [30]. *Process dependencies* occur when a team has to wait for a process to complete. *Resource dependencies* occur when the workflow is blocked because the team is waiting for a resource to become available. These three types of dependencies are further divided into eight subcategories, see Table 1 for the descriptions.

2.3 Dependencies in Large-Scale Agile

In large-scale software development, coordination is an essential but challenging success factor [25, 26]. Previous studies have used frameworks to study large-scale coordination. For example, Scheerer et al. [27] used the multiteam systems perspective from organizational psychology to study inter-team coordination and found that strategies that rely on organic and cognitive mechanisms may help achieve coordination effectiveness in large-scale agile projects. Nyrud and Stray [22] applied a framework proposed by Van De Ven, et al. [32] in a case study on inter-team coordination in large-scale agile and identified eleven different coordination mechanisms (such as open work area, stand-up meeting, retrospective, backlog grooming, demo, Sprint planning, and Jira).

One of the earliest mechanisms introduced for managing dependencies across several teams in large-scale agile 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 [25, 26]. Paasivaara et al. [25] found that Scrum of Scrums in two large-scale projects with 20 teams was challenging. The results showed that the audience was too big to keep everyone interested, and the participants did not know what to report.

Because of challenges with agile practices in large-scale projects, agile consultants have created several frameworks for scaling agile, such as the Large-Scale Scrum (LeSS) [15] and Scaled Agile Framework (SAFe) [16]. SAFe describes inter-team coordination practices such as a product increment planning event, Product Owner sync meeting, a demo and Scrum of Scrum meetings.

2.4 DevOps

DevOps is a pretty new concept for developing software that extends agile principles to the entire software delivery process [14]. In most companies, development and operations exist as separate functions, therefore, the collaboration in DevOps seeks to bridge the silos of software development and operations functions, and the idea is that this will reduce the amount of overhead that is usually prevalent in organizations where there are a lot of hierarchies, middle-managers, and inter-team cooperation [17]. DevOps is about rapid, flexible development iterations through domain-crossing team compositions that break complex architecture and features sets into small chunks that can be produced and deployed independently [10]. Furthermore, a key concept in DevOps is continuous integration, defined as a process that is triggered automatically and includes inter-connected stages such as compiling code, running tests, validating code and building deployment packages [12].

To sum up, DevOps highlight principles such as: 1) Knowledge sharing by breaking down barriers between development and operations 2) Automation of build, deployment and test, 3) Embracing shared responsibility and 4) Ensuring continuous software development [14].

3. Method

Our case is a large Norwegian municipality with approximately 50,000 employees, and 50 organizational units, all varying in size, responsibilities, domain, IT-competence, and funding. The municipality has its own development program, which is responsible for integrating hundreds of internal systems and is a data hub for communicating with the population and other business partners. Examples of solutions they provide are web solutions, mobile solutions, document-handling solutions, and business systems. We conducted a case study to investigate the research question since a case study is an appropriate method to answer “how” and "why" questions [33]. We wanted to investigate how agile teams manage dependencies and why the coordination mechanisms identified in the project were used. The project had seven teams, which makes it a large-scale agile project (following the definition in [7]). Five teams were DevOps team; teams Jupiter (7 members), Pluto (5 members), Mars (8 members), Saturn (5 members), and Venus (4 members). Two of the teams supported the DevOps teams; they are called

team Earth (8 members) and team Customer (6 members).

3.1 The DevOps teams

Five of the teams under study followed DevOps principles, meaning that, for features they implemented, they had full responsibility from developing and testing to deploying to production and monitoring the feature. Each team had responsibility for a set of components, and other teams could access these components through a defined API. All the components could be deployed independently.

The team size varied from four to eight members, with an average of six members. Some teams focused on frontend-oriented tasks, others on backend-oriented tasks. Each team had a team lead (who they sometimes referred to as the Scrum Master). All the teams followed an agile approach, and they could choose either Scrum, Kanban or a combination. Independent of which agile method they followed, they all used agile practices such as daily stand-up meetings, Scrum of Scrum meetings, and product demos. Jira was the project management tool where user stories, issues, sprint plans, and priorities were collected. Each of the teams had a separate electronic Kanban board showing their tasks in the columns "to do," "in progress," "awaiting," and "done."

3.2. Data collection and analysis

We chose participant observations as the primary method for data collection. We observed 38 scheduled meetings, see Table 2 for an overview. We also observed the teams working. The meetings were observed between November 2017 and January 2018. Additionally, we supplemented with four interviews of project members in February 2018, mainly to confirm our observations and help us understand issues that were unclear after the first round of data analysis.

Our first step in data analysis was to prepare a summary and a reflection paper of each note from the meetings and other observed material. In total, we analyzed more than 60 pages from the observed meeting notes. The reflection paper included details of the organization under study, the project, the teams, the meetings, the roles, and other coordination observations.

We used both inductive and deductive coding techniques [19]. The taxonomy of dependencies and coordination mechanisms proposed by Strode [30] was used to acquire an overview of the key concepts. All the data sources were uploaded into a software program tool for analyzing qualitative data called QSR NVivo. The data items were given a descriptive name, a code, and each code was defined uniquely. The coding approach was guided by Crowston and Osborn [6] and aimed at identifying dependencies and their associated coordination mechanisms.

Table 2. Overview of the meetings observed

Observations	Total	Team Observed
Daily Stand-up	12	3 team Jupiter, 3 team Mars, 2 team Saturn, 4 team Venus
Demo meetings	6	Participants from all 7 teams
Sprint meetings	2	Participants from team Venus
Scrum of Scrum meetings	5	Participants from all 7 teams
Project meetings	7	Participants from team Earth and the customer
Workshop	3	Participants from Jupiter, Pluto, Mars, Saturn, and Venus
Team leader meetings	2	Participants from Jupiter, Pluto, Mars, Saturn, and Venus
Retrospective	1	Participants from team Pluto
Sum:	**38**	

4. Results

Table 3 provides an overview of the identified coordination mechanisms in the categories "synchronization activities" and "synchronization artifacts," and how they address relevant dependencies for this large-scale project. There are eight types of dependencies, and the coordination mechanisms can address one or more dependency.

We found 20 synchronization activities and artifacts. In this section, we report on the practices and artifacts that addressed four or more types of dependencies since these indicate to be effective and essential coordination mechanisms in large-scale projects.

4.1. Daily stand-up meetings

The daily stand-up meeting was vital for managing task allocation dependencies and expertise dependencies (knowing who knows what). In these meetings, a team member started by telling the team what he or she had done since the last daily stand-up meeting before discussing obstacles. An obstacle often caused another team member to discuss what was the

Table 3: Dependencies and coordination mechanisms identified in the large-scale program

Knowledge dependency	
Process dependency	
Resource dependency	

(Coordination mechanisms marked in bold addressed four or more dependencies)

			Dependencies							
			Knowledge				Process		Resource	
			Expertise	Requirement	Task allocation	Historical	Activity	Business process	Entity	Technical
Coordination Mechanisms	Synchronization activities	**Scrum of Scrum meetings**	■	■	■		■			
		Team leader meetings	■			■		■	■	
		Daily stand-up	■		■		■	■	■	■
		Retrospective	■			■				
		Software release								■
		Workshops	■	■						
		Sprint Planning meetings	■	■	■					
		Ad hoc conversations	■	■	■		■	■		■
		Project meetings	■		■					
		Prep. for product demo	■			■				
		Product demo to customer		■						
	Synchronization artifacts	Wiki		■						
		Task			■					
		Product backlog			■		■			
		Communication tools	■	■	■	■	■		■	
		Project management tools		■	■	■				
		Priority list	■		■					
		Kanban board		■	■		■		■	
		Whiteboard		■						
		Open work area		■	■		■		■	

best solution to the identified problem. The discussion usually ended up with coordinating tasks with a discussion of who should be involved in solving the task and the obstacles (task allocation dependencies). The team member ended his or her round by telling what would be done before the next meeting Then, the next team member started his or her status update, and the cycle began again by telling what was done since the last meeting. Often, team members raised obstacles 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) so that the team leader could be aware and manage the dependencies with other stakeholders. After the update from the last team member, team members and team the leader provided general information before summing up the meeting and ending it.

4.2 Scrum of Scrum meetings

The Scrum of Scrum meeting was a weekly scheduled meeting with a duration of 60 minutes. The participants in the meeting were project managers (from supplier and customer), team leaders, UX designers, Product Owners, architects, test lead, and security manager. Generally, 12 participants attended the meeting. This meeting was the most important meeting to manage expertise dependencies. In the meeting, the roles and expertise of others and who should coordinate with whom in the different teams were discussed. Every team leader gave a status of their tasks to the project manager customer. During the meeting, the participants discussed issues reported by the team leaders. The Product Owners and the project managers discussed the issues with the specific team leader and suggested solutions to what the DevOps teams could do to solve the problems. A project manager commented on how the meeting helped to solve dependencies:

We schedule this type of meeting every week because it is valuable to gather different important roles from the project. It allows us to prepare and discuss problems when roles, such as Product Owners and team leaders, are gathered together. This makes it worth spending one hour weekly at this type of meeting.

The Scrum of Scrum meetings were essential for managing activity dependencies (when progress is blocked or delayed as people wait for resources, necessary information, or the activities of others to be completed [30]). A team leader explained:

We have external dependencies. We had several cases where we were supposed to integrate with external parts of the system, but either they had not completed their part, or what they had done was not documented or they had not granted us access. It is frustrating when we are done with our deliverables, but we have to wait for this external part to be able to deliver it to the customer.

These kinds of issues were shared at the Scrum of Scrum meetings, as well as task allocation and requirements.

4.3 Sprint planning meeting

The Sprint planning meetings for a DevOps team lasted two hours and were divided into two phases; pre-planning and Sprint planning. First, 60 minutes with pre-planning was conducted. The purpose of the pre-planning was to plan unfinished tasks from the last Sprint and plan new tasks for the new Sprint period. The team leader focused on goals and tasks from the "to do" list from the Kanban board and discussed the tasks with the team members. Together, the team members coordinated the tasks and estimated when the task should be done. The Sprint planning meeting ensured coordination between the team members. After the pre-planning, the team ate lunch.

After the lunch break, the team met for Sprint planning. The duration was a new 60 minutes with the team members, the team leader, and the Product Owner. The goal was to agree on the tasks list suggested in the pre-planning. If the Product Owner disagreed on the priority and estimates, the Product Owner changed the priority list and the product backlog. The Sprint planning meeting was the most important meeting to manage requirement dependencies. In this meeting, the team became aware of the priorities and details of the user stories and tasks.

4.4 Team leader meetings

The team leader meeting was a weekly meeting, scheduled for one hour. The participants during this type of meeting were all the team leaders and the project manager. The topics for the meeting were challenges in the project and status updates for each team. Every team leader talked about what was done since last team leader meeting and updated the project manager about finished tasks. The project manager also gave a brief update from the management meetings held the day before. Issues and information about resources were also often a hot topic in the meeting and a concern for the project members. It was normal to make tactical decisions regarding resources in each team during the meeting. A team leader from one of the teams with a high historical competence commented on the future work:

If the plan is to continue, we need more resources in the form of back-end developers. We don't want that one of our team members shall be removed because our team has too many priorities in the project. We also do not want new resources who do not have the historical knowledge needed.

Moreover, another team leader from the project stated, *"We need more and competent resources in order to reach the goals and complete our tasks in this project."*

4.5 Ad hoc conversations

Informal ad hoc conversations occurred several times a day. Every team in the project practiced this way of managing dependencies. The informal conversations took place everywhere, especially where the teams were located. The open work area made it easy for the team members to make quick discussions, which created a fast working culture. By having other teams within a short distance, it was possible to walk from team to team. Ad hoc conversations often occurred between developers and the project manager. When developers or team leaders from the DevOps teams were unsure about details of the domain requirements, it was easy to talk to a specialist from the customer who was co-located in the room with the teams. The spontaneous communication when tasks and requirements were unclear often lead to unscheduled meetings. One of the Product Owners 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. If there are several small

issues, it is easier to handle the problem by talking with other team members instead of using time on the issues in the scheduled meetings.

4.6 Open work area with boards

The DevOps teams were seated in an open work area, see Figure 1. The open work area facilitated coordination through easy access to other team members and teams and promoted ad hoc conversations. The work area enabled frequent discussions of tasks and possible solutions to problems. Moreover, with meeting rooms available just a few meters from the seating arrangements, it was possible to implement informal and unscheduled meetings.

The project manager commented: "*By using the offices in the best possible way, and with so many participants involved in the project, we know from earlier that an open area makes the teams more autonomous and enables decisions in the project.*" The observations of the teams made it clear that the open work area and the visual boards were essential to managing task allocation dependencies because the project members could easily see who was working on what and help each other. Many of the teams used the board also during other meetings, typically in the daily stand-up meetings.

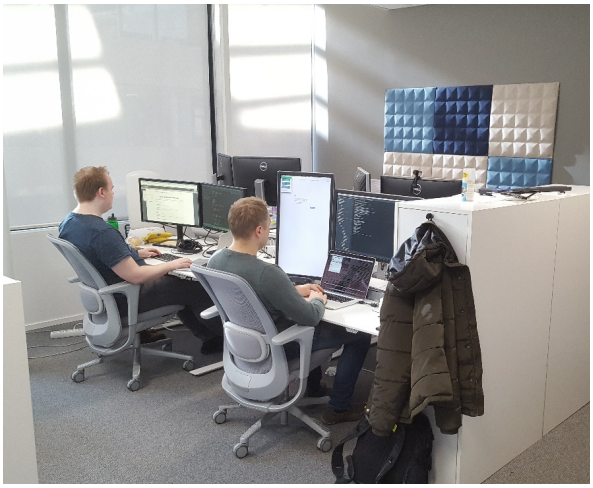

Figure 1: The open work area from Saturn

4.7 Communication tools

The teams made use of several tools for communications, such as Slack[1] and Skype.[2] Slack is a communication platform where team members can post messages to a group of people in chat rooms or directly to individuals. Team members have conversations in different chat rooms (called channels) according to a topic or a team. Through Slack, the team members informed others about issues, deliveries, and other work-related tasks. Besides, the tool was also rather social; team members invited other teams to lunch or other social small events during the workday.

Skype is a tool for video conversations and chat messaging. This tool was used when team members joined meetings, such as daily stand-up and demo meetings through video if they were located at home or on a special trip away from the office.

Both Skype and Slack helped manage entity dependencies because it made it possible to reach out to people who were not present more easily. The tools thus reduced the time having to wait for information and absent resources.

5. Discussion

In Section 4, we reported on the practices and artifacts that addressed the most types of dependencies since these are the most effective and essential coordination mechanisms in large-scale agile projects. These mechanisms lead to frequent production settings, a common understanding of what was being created, and enabled autonomous decisions.

We now discuss our research question: *How are dependencies managed in large-scale agile projects?* By using a dependency taxonomy [30], we identified two main categories of dependency management: synchronization activities and synchronization artifacts.

5.1. Synchronization activities

Two of the synchronization activities were inter-team coordination mechanisms: the Scrum of Scrum meetings and the team leader meetings. Both coordination mechanisms manage four dependencies, and together, they complement each other by managing seven dependencies.

The goal of the Scrum of Scrum meetings was to allow teams to communicate with each other to ensure that the solutions integrated well with the fundamentals of the other teams. From the literature, the Scrum of Scrum meeting is suggested to be time-boxed to last a maximum of 15 minutes [20], just like the daily stand-up meeting [29]. However, in a large-scale context with seven teams, this is not sufficient if the goal is to have it as a synchronization meeting that manages knowledge

[1] Slack is a registered trademark of Atlassian, www.slack.com
[2] Skype is a registered trademark of Skype Tech., www.skype.com

dependencies and especially expertise dependencies. Our results suggest that 60 minutes is a better time frame for a Scrum of Scrum meeting where various roles are present, and the meeting is used for managing dependencies, not just reporting status information from each of the teams. Other research also indicates that Scrum of Scrum needs to last longer than 15 minutes [5, 25].

In our study, the Scrum of Scrum meetings were successful as a synchronization activity because they allowed teams to communicate with each other and integrate information and knowledge from other teams while simultaneously giving status about tasks to the project manager. However, the Scrum of Scrum meetings were not enough and had to be supplemented with the team leader meetings. In the team leader meetings, there were usually 6 participants with similar roles, while in the Scrum of Scrum there were 12 participants with various roles. The team leader meetings, having fewer participants and roles, allowed for different discussions than the Scrum of Scrum meetings. Bick et al. [2] also found that the Scrum of Scrum meeting should be supplemented with other inter-team-level meetings in large-scale projects.

The rest of the synchronization activities in the large-scale agile project was intra-team coordination mechanism. Daily stand-up meetings helped in managing team-internal dependencies, specifically task-allocation dependencies. The daily stand-up meeting allowed the team members to share information on who was doing what, and when. The daily stand-up meeting managed six types of dependencies and, together with ad-hoc conversations, it was the most important coordination mechanism for the teams.

A recent study of a large-scale agile project found that project members spent on average 1.1 hours per day in scheduled meetings and 1.6 hours in ad-hoc conversations and unscheduled meetings [28]. In that study, all roles, including developers, testers, and managers, said they spent more time in unplanned coordination (ad-hoc conversations and unscheduled meetings) than they did in planned coordination (scheduled meetings). One reason for spending much time in unplanned coordination might be that ad hoc conversations are an efficient way of managing dependencies, and we found that it was used to manage five different types of dependencies, across all the three categories: knowledge, process, and resource.

The goal of the Sprint planning meetings was to delegate tasks to team members, estimate time on the tasks, and prioritize the tasks. Abrahamsson et al. [1] suggested that these meetings should be divided into two phases: First, users, management, the customer, and the Scrum team should decide the goals for the next sprint. Second, the Scrum Master and the Scrum team should focus on how to implement the product increment during the Sprint. In our case study, the meeting was also split into two phases, but in the opposite order. First, pre-planning with the team leader and the developers. Second, the Product Owner, team leader, and developers were gathered. This worked very well to manage task allocation dependencies effectively.

5.2. Synchronization artifacts

The teams made use of several tools for communications, such as Slack and Skype where they informed each other about deliveries and other work-related tasks. Slack managed historical dependencies because the logs could be read by all at a later time to find the reasoning of a previous decision. Slack also managed task-allocation dependencies because team members used the tool to discuss who should do what.

An open work area with boards enabled ad-hoc conversations, which is the main reason why it managed several types of dependencies. A recent study of a very large-scale project also found that the open work area contributed to efficient coordination and knowledge sharing [8]. The open work area and the boards coordinated activity dependencies by giving people necessary information of when others were completed with activities. The boards also helped manage task allocation dependencies because all project members could see who was working on what. The teams also used the boards in the meetings when they discussed requirement dependencies.

5.3. Implications for practice

Our research has several implications for practice. First, organizations should make sure that they prioritize the coordination practices and meetings that manage a high number of dependencies.

Second, agile projects change over time and so do the synchronization activities and artifacts. When teams or managers think of introducing a new coordination mechanism, they should evaluate which type of dependencies the mechanism will help manage (similar evaluations should be made when removing or changing existing practices).

Third, synchronizations tools such as Slack should be supported by the organization so that it is available and used by everyone, independent of which department they belong to. If practices are only used by part of the large-scale project, the project will experience misalignment in how dependencies are managed.

5.4. Limitations

As in any empirical study, there are some limitations that need to be discussed. Regarding the data collection, the presence of researchers may be intrusive and alter the behavior of the meeting attendees. Nevertheless, we believe this effect was small because most of the teams were observed over a longer period.

Another limitation is that the amount of observation per meeting type probably affected what dependencies we identified were managed by that particular meeting. For example, if we had observed more retrospective meetings, it might be the case that we would have identified that the practice managed more dependencies.

6. Conclusions and future research

In this case study, we used a dependency taxonomy to explore agile practices that acted as coordination mechanisms in a large-scale project. Meetings, ad-hoc conversations, communication tools, and an open work area with boards provided an essential venue for managing dependencies. The dependency taxonomy was useful for describing different dependencies and their associated agile practices that helped achieve effective project coordination.

Our study supports the finding that Scrum of Scrum meetings by themselves are not enough to manage inter-team dependencies in large projects. Other types of meetings, such as a meeting with the project manager and all the team leaders, is also necessary. The main implication of our study is, therefore, that project management needs to combine many coordination practices to be able to handle all the dependencies in large-scale agile projects.

Future research should focus on other types of agile teams (e.g. BizDev and BizDevOps) to investigate synchronization activities and artifacts and see what dependencies they manage and how they are different or similar to a project with DevOps teams. Additionally, one should investigate the two other strategy components in the framework by Strode [30]: structure (with the components proximity, availability, and substitutability) and boundary spanning (with the components spanning activity, spanning artifact, and coordinator role). Especially interesting to investigate would be what kind of dependencies are managed by the different agile roles acting as coordination mechanisms. Furthermore, one should analyze chat logs from communication tools such as Slack to see what kind of dependencies that are discussed and managed.

7. Acknowledgments

We are grateful to all those who participated in this research and to the anonymous reviewers for their valuable comments. This work was supported by the Research Council of Norway (grant 267704) and the companies Kantega, Knowit, Sbanken and Storebrand through the research project Autonomous teams.

8. References

[1] Abrahamsson, P., Salo, O., Ronkainen, J., and Warsta, J., "Agile Software Development Methods: Review and Analysis," *VTT Technical Research Centre of Finland, VTT Publications 478,* 2017.

[2] Bick, S., Spohrer, K., Hoda, R., Scheerer, A., and Heinzl, A., "Coordination challenges in large-scale software development: a case study of planning misalignment in hybrid settings," *IEEE Transactions on Software Engineering,* no. 1, pp. 1-1, 2017.

[3] Blichfeldt, B. S. and Eskerod, P., "Project portfolio management – There's more to it than what management enacts," *International Journal of Project Management,* vol. 26, no. 4, pp. 357-365, 5// 2008.

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

[5] Cohn, M., "Advice on conducting the scrum-of-scrums meeting," 2007.

[6] Crowston, K. and Osborn, C. S., "A coordination theory approach to process description and redesign," In Malone, T. W., Crowston, K. & Herman, G. (Eds.) Organizing Business Knowledge: The MIT Process Handbook (pp. 335–370). Cambridge, MA: MIT Press. 1998.

[7] Dingsøyr, T., Fægri, T. E., and Itkonen, J., "What is large in large-scale? A taxonomy of scale for agile software development," in *International Conference on Product-Focused Software Process Improvement*, 2014, pp. 273-276: Springer.

[8] Dingsøyr, T., Moe, N. B., Fægri, T. E., and Seim, E. A., "Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation," *Empirical Software Engineering,* journal article pp. 1-31, 2017.

[9] Dingsøyr, T., Moe, N. B., and Seim, E. A., "Coordinating Knowledge Work in Multi-Team Programs: Findings from a Large-Scale Agile Development Program," *arXiv preprint arXiv:1801.08764,* 2018.

[10] Ebert, C., Gallardo, G., Hernantes, J., and Serrano, N., "DevOps," *IEEE Software,* vol. 33, no. 3, pp. 94-100, 2016.

[11] Faraj, S. and Sproull, L., "Coordinating Expertise in Software Development Teams," *Management Science,* vol. 46, no. 12, pp. 1554-1568, 2000.

[12] Fitzgerald, B. and Stol, K.-J., "Continuous software engineering: A roadmap and agenda," *Journal of Systems and Software,* 2015.

[13] Grinter, R. E., "Understanding dependencies: A study of the coordination challenges in software development," University of California, Irvine, 1996.

[14] Jabbari, R., bin Ali, N., Petersen, K., and Tanveer, B., "Towards a benefits dependency network for DevOps based on a systematic literature review," *Journal of Software: Evolution and Process,* 2018.

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

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

[17] Lwakatare, L. E. *et al.*, "Towards DevOps in the Embedded Systems Domain: Why is It So Hard?," in *2016 49th Hawaii International Conference on System Sciences (HICSS)*, 2016, pp. 5437-5446.

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

[19] Miles, M. B., Huberman, A. M., Huberman, M. A., and Huberman, M., *Qualitative data analysis: An expanded sourcebook*. sage, 1994.

[20] Moe, N. B. and Dingsøyr, T., "Emerging research themes and updated research agenda for large-scale agile development," presented at the Proceedings of the XP2017 Scientific Workshops, Cologne, Germany, 2017.

[21] Moe, N. B., Dingsøyr, T., and Dybå, T., "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.

[22] Nyrud, H. and Stray, V., "Inter-team coordination mechanisms in large-scale agile," presented at the Proceedings of the XP2017 Scientific Workshops, Cologne, Germany, 2017.

[23] Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P., and Still, J., "The impact of agile practices on communication in software development," *Empirical Software Engineering,* vol. 13, no. 3, pp. 303-337, 2008.

[24] Pries-Heje, L. and Pries-Heje, J., "Why Scrum Works: A Case Study from an Agile Distributed Project in Denmark and India," in *2011 Agile Conference*, 2011, pp. 20-28.

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

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

[27] Scheerer, A., Hildenbrand, T., and Kude, T., "Coordination in large-scale agile software development: A multiteam systems perspective," in *System Sciences (HICSS), 2014 47th Hawaii International Conference on*, 2014, pp. 4780-4788: IEEE.

[28] Stray, V., "Planned and Unplanned Meetings in Large-Scale Projects," in *Proceedings of the XP2018 Scientific Workshops*, Porto, Portugal, 2018, pp. 1-5: ACM.

[29] Stray, V., Moe, N. B., and Sjøberg, D. I. K., "The Daily Stand-Up Meeting: Start Breaking the Rules," *IEEE Software,* 2018 (in press).

[30] Strode, D. E., "A dependency taxonomy for agile software development projects," *Information Systems Frontiers,* journal article vol. 18, no. 1, pp. 23-46, February 01 2016.

[31] Strode, D. E., Huff, S. L., Hope, B., and Link, S., "Coordination in co-located agile software development projects," *J. Syst. Softw.,* vol. 85, no. 6, pp. 1222-1238, 2012.

[32] Van De Ven, A. H., Delbecq, A. L., and Koenig, R., "Determinants of Coordination Modes within Organizations," *American Sociological Review,* vol. 41, no. 2, pp. 322-338, 1976.

[33] Yin, R. K., *Case study research: Design and methods*. California: SAGE Publications 2002.