Managing Dependencies in Large-Scale Agile

Henrik Vedal[1], Viktoria Stray[1,2(✉)], Marthe Berntzen[1], and Nils Brede Moe[2]

[1] Department of Informatics, University of Oslo, Oslo, Norway
{henrikav,stray,marthenb}@ifi.uio.no
[2] SINTEF Digital, Trondheim, Norway
{viktoria.stray,nils.b.moe}@sintef.no

Abstract. Delivering results iteratively and frequently in large-scale agile requires efficient management of dependencies. We conducted semi-structured interviews and virtual observations in a large-scale project during the Covid-19 pandemic to better understand large-scale dependency management. All employees in the case were working from home. During our data collection and analysis, we identified 22 coordination mechanisms. These mechanisms could be categorized as synchronization activities, boundary-spanning activities and artifacts, and coordinator roles. By using a dependency taxonomy, we analyzed how the mechanisms managed five different types of dependencies. We discuss three essential mechanisms for coordination in our case. First, setting Objectives and Key Results (OKRs) in regular workshops increased transparency and predictability across teams. Second, ad-hoc communication, mainly happening on Slack because of the distributed setting, was essential in managing dependencies. Third, the Product Owner was a coordinator role that managed both inter-team and intra-team dependencies.

Keywords: Product Owner · OKR · Slack · Distributed teamwork

1 Introduction

In large-scale agile software development, teams are surrounded by a larger development context that is often characterized by a high number of dependencies [1]. Teams, therefore, need to understand dependencies within their team as well as to other teams, and to understand how to efficiently manage, or coordinate, these dependencies [2]. A dependency occurs when the progress of one activity, such as a development task, is dependent on the output of a previous activity [3,4]. The more dependencies, the greater the coordination effort is required. Additionally, agile development is the emergence of tasks and work structures during the project [5], which implies that new dependencies will emerge during the development process. Therefore, large-scale agile is characterized by a high number of dependencies that is difficult to plan for up-front, and coordination has been identified as a top challenge to successful large-scale agile [6,7].

P. Gregory and P. Kruchten (Eds.): XP 2021 Workshops, LNBIP 426, pp. 52–61, 2021.
https://doi.org/10.1007/978-3-030-88583-0_6

Table 1. Table of coordination strategy components [8]

Distinct component	Component	Definition
Synchronization	Synchronization activity	Activities performed by all team members simultaneously that promote a common understanding of the task, process, and or expertise of other team members
	Synchronization artefact	An artefact generated during synchronization activities. The nature of the artefact may be visible to the whole team at a glance or largely invisible but available. An artefact can be physical or virtual, temporary or permanent
Structure	Proximity	This is the physical closeness of individual team members. Adjacent desks provide the highest level of proximity
	Availability	Team members are continually present and able to respond to requests for assistance or information
	Substitutability	Team members are able to perform the work of another to maintain time schedules
Boundary spanning	Boundary spanning activity	Activities (team or individual) performed to elicit assistance or information from some unit or organization external to the project
	Boundary spanning artefact	An artefact produced to enable coordination beyond the team and project boundaries. The nature of the artefact may be visible to the whole team at a glance or largely invisible but available. An artefact can be physical or virtual, temporary or permanent
	Coordinator role	A role taken by a project team member specifically to support interaction with people who are not part of the project team but who provide resources or information to the project

Research within large-scale has addressed coordination at the organizational, project, inter-team and team level in both distributed and co-located settings [1,8–10]. While research-based knowledge on coordination in large-scale agile is expanding, there are still unresolved questions, such as which coordination mechanisms used in large-scale agile are more effective. Guided by the need for more knowledge on coordination in large-scale agile, we address the following research question: *How are dependencies managed in large-scale agile projects?* To address this question, we report on a case study conducted in a distributed development team in a large-scale organization.

1.1 A Framework for Coordination in Agile Teams

In this study, we rely on a theory of coordination for agile teams developed by Strode and colleagues [8]. The theory is relevant in large-scale contexts because it takes into account that organizational structure, project complexity, and

uncertainty influence coordination [8]. We chose this theory as a lens for investigating large-scale coordination because it captures both explicit coordination (e.g., an insight report) and implicit forms of coordination (such as knowledge sharing) [1,8].

Coordination mechanisms are specific activities and artifacts aimed at addressing dependencies of three types [3]: 1) Knowledge dependencies is when a form of information is needed for a project to progress and consist of four sub-categories: expertise, requirement, task allocation, and historical. 2) Process dependencies are defined through two categories, activity and business process, which entails when a task must be completed before another task can be initiated. 3) Resource dependencies are composed of entity and technical, which is when an object is needed for a project to progress. For example, entity dependency is when a person is not available and this affects project progress [3].

The theoretical framework proposes three categories of coordination mechanisms to manage these dependencies (see Table 1): Synchronization mechanism such as daily stand-up meetings and product backlog; structure mechanisms referring to the proximity, availability, and substitutability of team members; and boundary spanning mechanisms that connect interdependent teams [3,8].

The theory further proposes that agile coordination mechanisms lead to coordination effectiveness, where agile team members have a shared understanding of their goals and priorities as well as how each team member's tasks fit together, as well as the necessary tools and artefacts available at the right time and place, thereby being able to sufficiently manage their dependencies [8]. The original framework included typical agile coordination mechanisms, such as daily stand-up meetings and the product backlog [3,8]. However, in large-scale settings, it is common to also use other project management mechanisms [1], including goal-setting frameworks [10]. One such framework that we will focus on in this study is the Objectives and Key results-framework, described next.

1.2 Objectives and Key Results

Objectives and key results (OKR) is a goal-setting framework to define a certain set of objectives in an organization and measure its progress. Instead of spending months setting long-term goals, OKR is designed to help organizations achieve their business goals much quicker in a structured manner. It is defined as "a critical thinking framework and ongoing discipline that seeks to ensure employees work together, focusing their efforts to make measurable contributions that drive the company forward" [11, p. 6]. An objective describes in short terms what the team wants to achieve, while key results allow the team to measure their progress and show when the objective has been reached.

The OKR-framework focuses on balancing business value and measureability [11]. This may explain why large-scale agile companies choose to adopt it. Another attractive feature of OKR that is compatible with agile is the emphasis on broad participation and collaboration across organizational levels [11], right down to the development teams.

2 Research Method

We chose to conduct a case study because it is an empirical research method for investigating contemporary phenomena and because it is particularly fit when the boundaries between context and phenomenons are ambiguous or not apparent [12]. Our case is an agency within a sizeable Norwegian municipality with approximately 50,000 employees distributed among 50 organizational units. Established as a project in 2017, but later organized as an agency in early 2020, the case comprises six departments and seven product areas, consisting of 11 permanent and three temporary teams structured with people from multiple departments (Fig. 1). These cross-functional agile teams deliver solutions such as web solutions, mobile solutions, document-handling solutions, and business systems. The work entails connecting existing systems in the municipality to create a shared service platform for agencies and businesses. Considering the amount of data the municipality holds, the objective is to facilitate the creation of high-quality, valuable services for all citizens of the municipality.

Our primary data collection was from Team Alpha. Their goal was to create a platform to facilitate easy access and sharing of data within agencies in the municipality and ensure it is being put to use. This included making intuitive and secure solutions, good documentation, and ultimately enabling the citizens to have access to better solutions. The team had ten permanent members and one part-time member (a UX designer). All permanent team members were interviewed in December 2020. See Table 2 for a description of the different roles. We interviewed one team lead, one tech lead, one UX designer, four back-end developers, two front-end developers, and one data scientist. The video conferencing tool Zoom was utilized, allowing easy access to record the interviews with

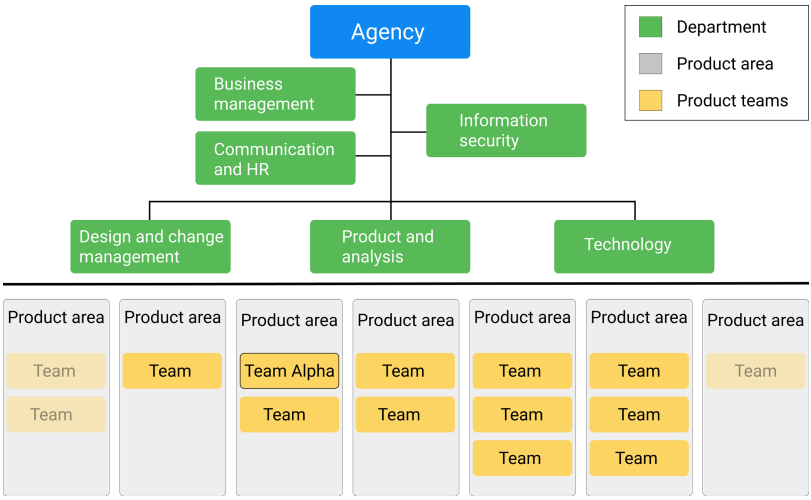

Fig. 1. The large-scale set-up.

permission from the interviewee. The average interview length was 48 min, and all interviews were transcribed.

We analysed the interview transcripts by systematically coding them in Nvivo 12. One analytical strategy proposed by Yin [12] is the reliance on a theoretical proposition, and we chose to guide the data analysis by using the coordination framework outlined above [8]. Following this framework, the analysis was organized and helped us point out relevant contextual conditions [12].

Table 2. Roles in team Alpha

Role	Members	Description
Team lead	1	Ensures that the team is moving in the right direction, communicating company goals, and facilitating a good flow of information
Tech lead	1	Also referred to as the architect, performs development work, coordination, and provides technical guidance for the team members
UX designer	2	Works with illustration, design, and collection of data and insight from other teams and agencies
Back-end	4	Management and operation of the existing systems, as well as the continuous implementation of new server-side features
Front-end	2	Development of new client-side functionality, creating a user interface and coordination with designers
Data scientist	1	Acquires requirements, use cases, and proof of concept for new features to be built

3 Results

We used the dependency framework [8] to categorize coordination mechanisms and how they managed the different dependencies. Figure 2 gives an overview of the identified coordination mechanisms and how they address relevant dependencies. In addition to the components shown in the figure, we also identified 11 synchronization artifacts, such as Kanban boards showing the status of the different tasks, and Github pages for documentation. In the following, we report on three coordination mechanisms that addressed the most dependencies (five or more) since these indicate to be the most important. These were OKR workshop, ad-hoc communication and Product Owner (PO).

3.1 OKR Workshop

Working towards the same goals was identified as a critical success factor. To achieve this, the project relied on the OKR framework. Every quarter the teams

Strategy components	Coordination mechanisms	Knowledge				Process		Resource	
		Expertise	Requirement	Task allocation	Historical	Activity	Business process	Entity	Technical
Synchronization activities	Daily standup	✓		✓		✓		✓	
	Pre-sprint planning	✓	✓				✓		
	Sprint planning	✓	✓	✓			✓		
	Sprint		✓			✓	✓	✓	
	Retrospective	✓		✓	✓				
	OKR workshop	✓	✓		✓		✓	✓	
	One-on-one meetings	✓	✓			✓	✓		
	Ad hoc communication	✓	✓	✓		✓		✓	
Boundary spanning activity	OKR training	✓	✓					✓	
	Inter-team meetings	✓						✓	
	Team lead meetings				✓		✓	✓	
	Ad hoc communication	✓	✓	✓		✓		✓	
Boundary spanning artefact	OKR tracker		✓	✓			✓		✓
	KPI						✓		✓
	Insight reports		✓		✓	✓	✓		
	Status reports from management							✓	
	Support tickets	✓			✓	✓			
	Chat logs		✓		✓				
Coordinator role	Tech lead	✓	✓					✓	
	Team lead	✓	✓	✓				✓	
	Product owner	✓	✓	✓			✓	✓	
	Data scientist	✓	✓					✓	
	UX designer	✓	✓					✓	

Fig. 2. Coordination mechanisms and dependencies identified.

arranged OKR workshops to set the direction and linking to the overall objectives of the project to the teams. Each team was encouraged to look at other teams' OKRs to understand the dependencies between teams. Team Alpha's OKR meeting handled internal team dependencies such as expertise, requirement, historical, business process, and entity.

Further, the OKR workshop managed expertise and requirement dependencies because they relied on very specific knowledge to create optimal OKRs. The workshop also managed entity dependencies as many individuals on the team provided valuable knowledge to ensure the quality of the OKRs. The team members stated how OKRs provided increased transparency, predictability, shared goals and an increased sense of ownership to what was produced in the project. The agency also utilized an OKR tracker, a tool which allowed any team to view the progress of other teams. One of the hardest parts about using OKR that was stated by several team members, was quantifying objectives through key results and the corresponding choice of words. A member of team Alpha stated: "*I think OKR is difficult. It is useful to maintain focus, but it is hard to create good, measurable key results which make sense.*" This further emphasizes the need for this workshop to manage the dependencies, as they are reliant on it to improve their collective understanding and set better OKRs.

3.2 Ad Hoc Communication

Besides regular inter-team and intra-teams meetings, there was a substantial amount of communication performed ad hoc, mainly using Slack. The ad-hoc communication on Slack managed expertise (team members reached out to other experts in the project), requirements (a team member located specific product related domain knowledge), task allocation (discussions led to emerging tasks) and activity dependencies (a team member needed information to continue work).

The Slack infrastructure provided public slack channels, dedicated team channels and private messages. Private messages was used both internally in the team and externally to communicate with others in the large-scale project. While much information was sent as private messages, the project members were encouraged to ask questions in open channels. When team members were unsure about details of the domain or technology, they found it easy to reach out on Slack to locate this information. The tech lead explained the following: *"Our team have a lot of experience with cloud technology and other project members often ask for assistance. We have similarly reached out to other teams that have specific knowledge which might be relevant for us".* This spontaneous communication often led to unscheduled meetings and positively benefitted the participants. The communication also involved agreeing on pair programming and discussing debugging. Because all project members were distributed, this coordination mechanism was probably more evident as the majority of communication during work hours was digital.

3.3 Product Owner

The PO was a coordinator role that managed a total of five dependencies. The role managed dependencies of types expertise, requirement, task-allocation, business process, and entity, as shown in Fig. 2. The PO communicated stakeholder interests, checked status, and pointed teams in the right direction. The majority of the work performed by the PO was coordination related to both intra- and inter-team coordination. Working tightly with the team lead and tech lead of Team Alpha, the PO assisted in the discussion of which key results (related to OKR) to prioritize throughout the quarter. This is what ultimately decides many of the tasks which the team works with during any point in time, which in turn manages requirement and task allocation dependency. It was not always evident to the product teams what to prioritize. The PO managed much of the inter-team coordination with the appropriate teams. This allowed the teams to be aligned and focus on their product, enabling autonomy and avoiding potential bottlenecks and misunderstandings.

4 Discussion

Understanding coordination in a distributed environment is critical to project success [13]. We studied how dependencies were managed in a large-scale agile

project that was forced to work from home due to Covid-19. We now discuss our research question: *How are dependencies managed in large-scale agile projects?*.

The use of OKRs served as an essential mechanism for managing dependencies because the approach was a foundation for setting direction for all teams. We identified how the OKR workshop managed dependencies such as expertise, requirement, historical, business process, and entity. The key to a successful workshop was having an overview of past decisions and specific knowledge about the product and project goals. In the workshop, team lead and tech lead managed requirement dependencies, which enabled effective prioritization, which is vital in high uncertainty complex projects [8].

The second most important mechanism to dependency management was ad hoc communication, which addressed expertise, requirement, task allocation, activity, and entity. For example, Team alpha often agreed to do pair programming ad hoc, which was positive, since one of the main challenges of remote pair programming is the initiation of pairing [14].

All scheduled and unscheduled communication was performed digitally. However, a high number of Slack channels resulted in challenges of getting an overview of what was going on in the project. Our findings are in accordance with [15], which found that coordination challenges are evident in distributed teams. Our results further showed that it was hard to balance ad hoc communication on Slack and scheduled meetings. We found that some discussions could go on for too long on Slack which created misunderstandings, instead of organizing a meeting to discuss the dependency and resolve the blocking of progress. Our findings are consistent with the findings of [13], which showed that a lack of guidelines when using Slack resulted in coordination being confusing and frustrating for team members. Another challenge with the use of Slack was that it was expected that the project members answered reasonably quickly, which some interviewees said led to increased interruptions and context switching in the distributed teams.

The third most important mechanism for managing dependencies was the PO role. The PO managed knowledge, process, and resource dependencies. Our results confirm previous research, that the PO role is important in large-scale agile for managing dependencies between and within the team [16–18]. In our case, the PO worked in close collaboration with the team leader and tech lead, thus managing process-related and technical dependencies. In accordance with the findings of Bass [17], we found that the PO performed a wide set of different functions. Our findings are also in line with Remta et al. [18]. In their study of POs in a company that implemented the Scaled Agile Framework, they found that the PO role entails responsibilities such as being a gatekeeper, motivating, communicating and prioritising [18], addressing five types of dependencies.

5 Conclusion and Future Work

In conclusion, our study suggests that by discussing OKRs, the teams manage dependencies both within the teams and also inter-team dependencies. We found

that ad-hoc communication mostly happened on Slack and that this communication made team members locate expertise in the large-scale project as well as discussing requirements, task-allocation and activity dependencies. The PO played an important role because it managed five different types of dependencies. In this large-scale project, there was no dedicated role focusing on OKRs, so discussions about them in the teams took time during meetings and required much facilitation. Our findings indicate that large-scale projects would benefit from having a dedicated "OKR master" to facilitate and follow up the OKR process. Future work should further explore how OKRs can be used to align teams in a large-scale distributed set-up.

References

1. Dingsøyr, T., Moe, N.B., Seim, E.A.: Coordinating knowledge work in multiteam programs: findings from a large-scale agile development program. Proj. Manag. J. **49**(6), 64–77 (2018)
2. Malone, T.W., Crowston, K.: The interdisciplinary study of coordination. ACM Comput. Surv. (CSUR) **26**(1), 87–119 (1994)
3. Strode, D.E.: A dependency taxonomy for agile software development projects. Inf. Syst. Front. **18**(1), 23–46 (2016).
4. Crowston, K., Osborn, C.S.: A coordination theory approach to process description and redesign (1998)
5. Boehm, B., Turner, R.: Management challenges to implementing agile processes in traditional development organizations. IEEE Softw. **22**(5), 30–39 (2005)
6. Bass, J.M.: Future trends in agile at scale: a summary of the 7th international workshop on large-scale agile development. In: Hoda, R. (ed.) XP 2019. LNBIP, vol. 364, pp. 75–80. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30126-2_9
7. Bass, J.M., Salameh, A.: Agile at scale: a summary of the 8th international workshop on large-scale agile development. In: Agile Processes in Software Engineering and Extreme Programming-Workshops, p. 68 (2020)
8. Strode, D.E., Huff, S.L., Hope, B., Link, S.: Coordination in co-located agile software development projects. J. Syst. Softw. **85**(6), 1222–1238 (2012)
9. Stray, V., Moe, N.B., Mikalsen, M., Hagen, E.: An empirical investigation of pull requests in partially distributed BizDevOps teams. In: The 16th ACM/IEEE International Conference on Global Software Engineering (ICGSE), pp. 110–119 (2021)
10. Berntzen, M., Stray, V., Moe, N.B.: Coordination strategies: managing inter-team coordination challenges in large-scale agile. In: Gregory, P., Lassenius, C., Wang, X., Kruchten, P. (eds.) XP 2021. LNBIP, vol. 419, pp. 140–156. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-78098-2_9
11. Niven, P.R., Lamorte, B.: Objectives and Key Results: Driving Focus, Alignment, and Engagement with OKRs. Wiley, Hoboken (2016)
12. Yin, R.: Case Study Research and Applications: Design and Methods, 6 edn. SAGE Publications, Upper Saddle River (2017)
13. Stray, V., Moe, N.B.: Understanding coordination in global software engineering: a mixed-methods study on the use of meetings and slack. J. Syst. Softw. **170**, 110717 (2020)

14. Smite, D., Mikalsen, M., Moe, N.B., Stray, V., Klotins, E.: From collaboration to solitude and back: remote pair programming during Covid-19. In: Gregory, P., Lassenius, C., Wang, X., Kruchten, P. (eds.) XP 2021. LNBIP, vol. 419, pp. 3–18. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-78098-2_1
15. Espinosa, J.A., Slaughter, S.A., Kraut, R.E., Herbsleb, J.D.: Team knowledge and coordination in geographically distributed software development. J. Manag. Inf. Syst. **24**(1), 135–169 (2007)
16. Berntzen, M., Moe, N.B., Stray, V.: The product owner in large-scale agile: an empirical study through the lens of relational coordination theory. In: Kruchten, P., Fraser, S., Coallier, F. (eds.) XP 2019. LNBIP, vol. 355, pp. 121–136. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-19034-7_8
17. Bass, J.M.: How product owner teams scale agile methods to large distributed enterprises. Empir. Softw. Eng. **20**(6), 1525–1557 (2015).
18. Remta, D., Doležel, M., Buchalcevová, A.: Exploring the product owner role within safe implementation in a multinational enterprise. In: Paasivaara, M., Kruchten, P. (eds.) XP 2020. LNBIP, vol. 396, pp. 92–100. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58858-8_10