Managing dependencies in large-scale agile

A case study of coordination in distributed teams

Henrik Aspenes Vedal

Thesis submitted for the degree of
Master in Programming and System Architecture
60 credits

Institute for Informatics
The faculty of Mathematics and Natural Sciences

UNIVERSITY OF OSLO

Spring 2021

Managing dependencies in large-scale agile

A case study of coordination in distributed teams

Henrik Aspenes Vedal

Managing dependencies in large-scale agile

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Abstract

Background: Coordination is very important in large-scale agile software development. When software projects grow in size or work distributed, additional difficulties are often introduced. Dependency management is therefore important for coordination in agile teams. By understanding how coordination practices manage different dependencies in distributed software development, organizations can introduce and adjust the practices which best fit their coordination needs.
Objective: The objective of this study is to investigate how distributed teams in large-scale agile manage dependencies to achieve effective coordination. This is done by identifying practices that act as coordination mechanisms and examining what dependencies they manage.
Method: In this qualitative multiple-case study, two distributed teams and their contexts were examined. The two separate teams were referred to as team Alpha and team Bravo. A total of 21 meetings were observed, and ten semi-structured interviews were held with team Alpha. In team Bravo, a total of 7 semi-structured interviews were held. Chat logs and documents from both teams were also a part of the collected data material.
Results: The findings of this study revealed 37 coordination mechanisms with 123 dependencies in team Alpha and 34 coordination mechanisms with 108 dependencies in team Bravo. These coordination mechanisms included agile and non-agile practices, and managed knowledge, process, and resource dependencies. The most important coordination mechanisms were product owner, OKR workshop, and ad hoc communication, which each managed a total of five dependencies. Working remotely further introduced difficulties, which complicated meetings, tools, and ad hoc communication.
Conclusion: Using a dependency taxonomy proved very useful to identify coordination mechanisms and dependencies in distributed teams in large-scale agile. The coordination mechanisms are presented with their best-matched dependencies. Practices such as ad hoc communication, collaboration tools, daily standup, written slackup, retrospective actions, sprint, and OKR workshop are presented and discussed. Coordinator roles such as product owner, team lead, and data scientist are also discussed. All these mechanisms are central for team Alpha and Bravo to coordinate effectively while working distributed. Further, complications introduced with OKR, and the distributed situation, are discussed with regards to dependency management. The findings of this study support previously known challenges with distributed teams and contribute to a better understanding of how to coordinate in distributed teams in large-scale agile more efficiently.

Acknowledgements

Writing this thesis has been a challenging yet very rewarding experience. It has been incredibly motivating to have the opportunity to study subjects that I find interesting, despite being performed remotely because of the pandemic. Completing this thesis would not have been possible if it was not for a number of people. I would like to sincerely thank my supervisor Viktoria Stray, and doctoral research fellow Marthe Berntzen for invaluable guidance. With excellent advice, discussions, and directions, I have been able to finish a thesis that I am proud of. I am also very grateful for the help of Jan Henrik Gundelsby, who helped provide an excellent case.

Furthermore, I am very grateful for the participants in this multiple-case study. Their welcomeness made me look forward to observations and interviews and really helped the outcome of this thesis. Also, many thanks to the team leads for such quick responses and friendly attitudes. It made the data collection process as efficient and easy as I could hope for.

Lastly, I would like to share my gratitude to fellow students and my roommate. They provided inspiration and joy throughout the entire process. A special thanks to my family and friends for their support, and my sincerest gratitude to my girlfriend for being by my side this entire time. Without your support, I would not be where I am today.

Henrik Aspenes Vedal
Oslo, May 2021

Contents

List of Figures

List of Tables

Introduction

Cross-functional autonomous teams are to an increasing degree being used in IT projects. Teams like these can be very much effective in organizations within markets that are recognized with rapid changes. These companies must adapt to complex environments, where the focus may change very fast to meet the customer's needs. The benefits of cross-functional autonomous teams lie in utilizing the competence throughout the team, combined with more decentralized decision-making. This results in a flatter governance structure, with distributed responsibility and more effective decision-making. Meanwhile, there are still challenges that need to be addressed.

To this date, researchers within software engineering have addressed questions related to leadership, coordination, organizational context, design of teams, and team processes (Stray, Moe and Hoda, 2018). In large-scale agile software development, teams are surrounded by a more significant development context that is often characterized by a high number of dependencies (Dingsøyr, Moe and Seim, 2018). These dependencies are challenging to plan for upfront, and coordination has been identified as a top challenge to successful large-scale agile (Bass, 2019; Bass and Salameh, 2020). Therefore, teams need to understand dependencies within their team and other teams and understand how to efficiently manage or coordinate these dependencies (Malone and Crowston, 1994). To further advance knowledge on these topics, this thesis will, in collaboration with SINTEF, study how large-scale agile teams manage their dependencies.

1.1 Motivation

With experience from military service, followed by my years as a student, it is evident for me that coordination and communication are important factors. As a part of a team in the service, the ability to communicate commands, strategies, and your own opinion is the foundation for its effectiveness and success. This is also something I learned as a student, constantly in a situation where collaboration, tasks, and different dependencies are to be managed at any point in time. Reflecting on these experiences, I truly acknowledge the importance of good communication and coordination, over time also developing an interest in the subject. These factors are also contributors of motivation when working in a development team and relevant for the software industry, which also makes it relevant as a research matter. Therefore I have chosen to study coordination in autonomous teams for this master thesis.

1.2 The research project

As one of Europe's largest research facilities, SINTEF performs thousands of projects each year for customers of all sizes. Dating back to 1950, their research has laid the foundation for innovation in nationalities across the globe, which has resulted in them being one of today's leading institutes of research. With expertise within fields from the deep oceans to deep space, SINTEF contributes to value creation and increased competitiveness within public and private sectors (SINTEF, 2020).

For autonomous cross-functional teams to be effective in today's and future value creation processes, characterized by complex processes and high uncertainty, new knowledge is needed on how such teams work. It is necessary to find out how autonomous cross-functional teams can work in different industries and across industries and test new models for team organization where the expertise and benefits of the Norwegian working culture are utilized in the best possible way. Funded by Forskningsrådets BIA-program, SINTEF initiated in 2017 a research project in cooperation with four business partners: Kantega, Knowit, Skandiabanken and Storebrand. The project's focus was to address three core themes concerning team autonomy: leadership, coordination, and knowledge management (SINTEF, 2020).

1.3 Research question

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. The research area for this study is coordination in two distributed agile teams in a large-scale agile agency. Both teams utilize the framework Objectives and Key Results (OKR), which influences how the teams prioritize their work. The process of both teams will be individually studied by examining meetings, tools, communication, and other agile and non-agile practices used in everyday work. These practices act as coordination mechanisms and are continually addressing dependencies in both teams. The focus of this thesis will be to study these practices and how they manage dependencies, which resulted in the following research question:

"How are dependencies managed in distributed teams in large-scale agile?"

1.4 Approach

Prior to this thesis, I have had the lucky opportunity to work as an intern in a large software consultancy agency as a software developer. With an already established network through my internship, I chose this to be the company to conduct my research. A large municipality in Norway is one of the clients of the consultancy agency and later

agreed to participate. Two distributed agile teams within the agency are investigated in separate contexts, involving all members of the teams. As the year 2020 is marked by a global pandemic, the situation requires the teams to work full-time from home. The data was accordingly collected remotely through observations, documents, and interviews. They are further used and evaluated at a team level, with the focus of answering the research question.

In this thesis, I will conduct a multiple-case study. Combined with a theoretical model proposed by Strode, Huff, Hope and Link (2012), I aim to analyze the collected data and uncover dependencies and related coordination mechanisms. Following this theory resulted in a well-structured and planned data collection, serving as my research design. Further, the data analysis was initiated with the coding of the material in a software application called Nvivo. The raw datasets were converted into more deliberate data by organizing the data into themes and textual codes. Principles and strategies from Yin (2017) are followed to establish and maintain reliability and validity throughout the collection and analysis of the data. This is further explained in Chapter 3.

1.5 Structure

Chapter 2: Background: This section will provide a theoretical foundation and investigate the most essential parts of the field. The expected result is to get a proper understanding of agile development and how coordination in autonomous teams fit into this.

Chapter 3: Method: This part will be elaborating on my choice of method. This includes the data collection to be conducted, associated with analysis and how the data should be validated.

Chapter 4: Research context: Context about the teams being researched is presented.

Chapter 5: Results: A presentation of the results from collected data will be held.

Chapter 6: Discussion: The results presented in the last section will be discussed.

Chapter 7: Conclusion and future work: Conclude the findings and propose possible directions for future research.

Background

This chapter contains a presentation of the theoretical context on software development, including the emergence of agile and the value of coordination in software development. The dependency taxonomy used in this study is also presented and context on autonomous and distributed teams. Furthermore, background on Objectives and Key Results (OKR) - the goal-tracking framework utilized in the case teams is introduced.

2.1 Software development

Software development is the process of planning, designing, creating, and maintaining applications or software components and belongs to the engineering discipline (Wohlin, Šmite and Moe, 2015). The term "software development" was coined in a conference in 1968, aimed at discussing the need for a software development discipline (Kirk and MacDonell, 2015). Prior to the deriving of software development processes, systems were often created without much of a plan, where design and architecture often were determined through short-term decisions. Although working well with small systems, it proved difficult as the workload increased and the systems scaled. Methodologies were later introduced, imposing a more structured approach to software development and defining disciplined processes to focus on more predictable and efficient work (Awad, 2005). Accordingly, the traditional way of developing software emerged, focusing on uncovering design flaws before the programming is initiated (Munassar and Govardhan, 2010).

2.1.1 Plan-driven development

A plan-driven process model is a formal procedure for developing a product, also referred to as traditional software development. The functionality to be implemented is well planned and documented before the work starts, often set up in a linear order where one task must be completed before the next can start. Characterized by individual role assignment, the plan-driven approach attempts to anticipate the product's features before the initiation of development (Moe, Dingsøyr and Dybå, 2008). Royce (1970) presented a paper on his recommendations for process in system development, defining a stage-wise model of the software process later known as the Waterfall Model. Comprising five stages, it emphasizes the importance of stability and is regarded as the first software process model.

Plan-driven process models have since derived from this, and their sequential work

patterns represent a new approach of planning and management in software projects (Li, Moe and Dybå, 2010). Individual phases consist of work assigned to teams of highly independent professionals within each field (Moe, Dingsøyr and Dybå, 2008). In the instance where something must be changed in an earlier phase, the entire process must be adjusted backward. Promoting stability and long-term planning, plan-driven processes focus on good documentation and high-quality requirements and are best fit to predictable and well-defined projects (Awad, 2005). Its rigid structure poses many dependencies between teams, and its lack of flexibility can be time-consuming and potentially very expensive. Although highly context-dependent, studies show that agile perform better regarding software quality and time-to-market than waterfall processes (Li, Moe and Dybå, 2010).

2.1.2 Agile development

Agile processes were introduced as an alternative to the traditional processes methodologies, one of which being the waterfall model. In 2001, seventeen representatives from different methods, such as Scrum and Extreme programming (XP), arranged a meeting to discuss future trends in software development. It was evident that their individual methods had many similarities, resulting in the emergence of the "Agile Alliance" and a manifesto for agile software development (Awad, 2005). This manifest promotes a set of values that embrace factors such as people, interactions, working software, customer collaboration, and change, rather than processes, tools, contracts, and plans.

Considering agile in the context of software engineering, Larman and Vodde (2008) emphasizes the importance of defining what it actually stands for. *"Agile is not a practice. It is a quality of the organization and its people to be adaptive, responsive, continually learning and evolving"* (Larman and Vodde, 2008, p.137)

Often being perceived as a formal practice that one can do, agile actually represents the underlying values that agile methods and practices represent. One can do concrete methods or practices that encourage agility, but according to Larman and Vodde (2008), either you are agile, or you are not. Fink and Neumann (2007) define agility as "*the ability to respond operationally and strategically to changes in the external environment. The response has to be quick and effective for the organization to be considered agile*". Agile methods often involve delivering working software in iterations, resulting in short-term decision-making in contrast to the traditional approach. Agile teams are set up to be self-managing teams and are encouraged to become involved in project management decisions (Moe, Aurum and Dybå, 2012).

2.1.3 Large-scale agile

Initially designed for small single-team projects, agile methods have shown potential benefits such as high adaptability to change, reduced risk, and continuous customer involvement, making them attractive for larger projects and companies (Dikert, Paasivaara

and Lassenius, 2016). Compared to smaller projects, large-scale agile is characterized by the need for additional coordination and is more challenging to implement (Dyba and Dingsoyr, 2009). This difficulty is associated with the higher organization inertia, which comes with the increasing size, slowing down organization change (Livermore, 2008).

There is much variation on how to define large-scale agile throughout studies. Dingsøyr, Fægri and Itkonen (2014) present a taxonomy for defining the size of teams and how coordination should be approached in each scenario (see figure 2.1). Although the taxonomy is based on a theoretical model of a project, they emphasize how this taxonomy can help characterize research and its related scale in agile software development. Further, Dikert, Paasivaara and Lassenius (2016) identified a few studies which regarded the number of people involved as the deciding factor, ultimately defining large-scale agile as software development organizations with at least 50 people or a minimum of six teams. Considering this definition, the team size would be around 6 or 7 people. The taxonomy from Dingsøyr, Fægri and Itkonen (2014) will be used as the definition in this thesis.

Level	*Number of teams*	*Coordination approaches*
Small-scale	1	Coordinating the team can be done using agile practices such as daily meetings, common planning, review and retrospective meetings.
Large-scale	2-9	Coordination of teams can be achieved in a new forum such as a Scrum of Scrums forum.
Very large-scale	10+	Several forums are needed for coordination, such as multiple Scrum of Scrums.

Figure 2.1: A taxonomy of scale of agile software development projects

Projects of large size often introduce high complexity with regards to organizational and technical context and dependencies among tasks and teams. This further increases the need for formal documentation, reducing the agility as a result (Dingsøyr, Moe and Seim, 2018). In addition to inter-team coordination, development teams are sometimes required to interact with other organizational units, which often are non-agile in nature. Large projects are dependent on the ability to manage this complexity, considering its negative effect on performance (Dingsøyr, Moe and Seim, 2018; Floricel, Michela and Piperca, 2016). In a study about large-scale agile transformations, Dikert, Paasivaara and Lassenius (2016) uncovered challenges such as change resistance, insufficient customization of agile, lack of training, and reverting to old ways of working. As organizations and teams transition into large-scale agile and new challenges emerge, it is imperative to adapt. Further, in large-scale agile, many development teams are dependent on an efficient coordination process to manage the increased complexity. Accordingly, it is important to study coordination practices in large-scale agile development (Dingsøyr, Moe and Seim, 2018).

2.2 Coordination

By focusing on collaborative teamwork and collaboration with customers, agile methods aim to achieve early delivery of quality software. Like other project teams, agile development teams collaborate effectively using mechanisms to coordinate their interdependent work. Unfortunately this is quite difficult, and coordination breakdowns are a major problem in software development (Strode, 2016; Herbsleb, 2007; Kraut and Streeter, 1995; Curtis, Krasner and Iscoe, 1988).

To be able to maintain productivity across teams in large-scale agile projects, coordination is essential. A large amount of dependencies, uncertainty, and the complexity that comes with it requires a continuous emergence of coordination mechanisms (Berntzen, Moe and Stray, 2019). In organization studies, coordination is defined differently across organization-, project- and team level in software development (Strode, Huff, Hope and Link, 2012). As organizations grow and progress into large-scale, coordination across teams is equally important as inside the different individual teams. Coordination across teams is defined as inter-team coordination and is important in situations where particular tasks must be performed under certain constraints caused by inter-dependencies. As a result, derived the definition of coordination as managing dependencies (Malone and Crowston, 1994).

Dependency management is by definition central to coordination. This is achieved by using coordination mechanisms that address dependencies in any given situation (Strode, 2016; Malone and Crowston, 1994). Dependencies are described by Crowston and Osborn (1998) as *"when the progress of one action relies upon the timely output of a previous action or on the presence of a specific thing, where a thing can be an artifact, a person, or a piece of information"*. Strode (2016) further states, *"When dependencies occur in a development project, they can be managed well, poorly, or not at all"* (Strode, 2016, p. 24).

2.2.1 Coordination theories

This section will present the most relevant theories concerning coordination. The theories and definitions of coordination emerge from various fields and are necessary to address to establish context.

Deriving from sociology, Van de Ven, Delbecq and Koenig Jr (1976) define coordination as *"integrating or linking together different parts of an organization to accomplish a collective set of tasks"*(Van de Ven, Delbecq and Koenig Jr, 1976, p. 322). Additionally, three categories of coordination mechanisms are identified; impersonal, personal, and group (Van de Ven, Delbecq and Koenig Jr, 1976, p. 69). Originating from organizational theory, Mintzberg (1980) further defines six coordinating mechanisms: mutual adjustment, direct supervision, standardization of work processes, standardization of outputs, standardization of skills, and standardization of norms.

In order to study coordination mechanisms, both Mintzberg (1980), and Van de Ven,

Delbecq and Koenig Jr (1976), provide interesting approaches. Mintzberg (1980) explains coordination with regards to tasks, while Van de Ven, Delbecq and Koenig Jr (1976) consider coordination in the respective parts of an entire organization. Strode (2016) further defines coordination in a dependency taxonomy. She explains that taxonomies are useful when little is known about a topic, and further research could assess the applicability of her dependency taxonomy in contexts such as large-scale or distributed agile software development.

Nyrud and Stray (2017) performed a case study in large-scale agile on inter-team coordination, identifying eleven different coordination mechanisms. These mechanisms were mapped into five categories and drawn on the framework proposed by Van de Ven, Delbecq and Koenig Jr (1976). The framework proved to be an important tool in this case study, providing better awareness and understanding of coordination as a concept. Furthermore, Stray, Moe and Aasheim (2019) conducted a qualitative case study of coordination in autonomous DevOps teams. Using the framework designed by Strode (2016), the results revealed 34 coordination mechanisms and 77 pairs of dependencies. The coordination mechanisms act as a defined group of components called coordination strategy components. These components - synchronization, structure, and boundary spanning, are used to manage dependencies (Strode, Huff, Hope and Link, 2012). The strategy components are detailed in table 2.1

Stray, Moe and Aasheim (2019) also recommended using the same taxonomy on different datasets to compare the outcome. Including the recommendation by Strode (2016), the taxonomy is suitable for the approach of my multiple-case study with regards to distributed teams in large-scale agile. I have therefore chosen to follow the dependency taxonomy of Strode (2016) to uncover the different coordination mechanisms and dependencies in the collected data.

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.

Table 2.1: Table of coordination strategy components (Strode, Huff, Hope and Link, 2012)

2.2.2 Dependency taxonomy

Strode, Huff, Hope and Link (2012) present a theoretical model of coordination mechanisms and dependencies through an empirical case study of three co-located cases of agile software development. This theoretical model proposes an agile coordination strategy to increase coordination effectiveness. Strode (2016) further highlights the importance of managing dependencies:

> "When dependencies are well managed this suggests that appropriate coordination mechanisms are in place to support the smooth flow of inter-dependent work. When dependencies are poorly managed then coordination mechanisms might be inappropriate, inadequate, or absent" (Strode, 2016, p. 24).

Poorly managed dependencies may lead to a lack of progress or delays, as people have to wait for resources to be available. Being potential bottlenecks in the development

process, they need to be identified and handled to ensure workflow.

This taxonomy describes three different categories of dependencies: knowledge, process, and resource. Knowledge dependency 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. 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. Resource dependencies are composed of entity and technical, which is when an object is needed for a project to progress. Further details about the different dependencies are described in table 2.2.

Dependency		**Description**
Knowledge dependency	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.
	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
	Task allocation	Who is doing what, and when, is not known and this affects, or has the potential to affect, project progress.
	Historical	Knowledge about past decisions is needed and this affects, or has the potential to affect, project progress.
Process dependency	Activity	An activity cannot proceed until another activity is complete and this affects, or has the potential to affect, project progress.
	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.
Resource dependency	Entity	A resource (person, place, or thing) is not available and this affects, or has the potential to affect, project progress.
	Technical	A technical aspect of development affects progress, such as when one software component must interact with another software component and its presence or absence affects, or has the potential to affect, project progress.

Table 2.2: Table of dependencies (Strode, 2016)

2.3 Agile teams

Compared to traditional software development with an individual role assignment, agile relies on teamwork which often is stimulated through cross-functional autonomous teams, illustrated in the figure below. With more involvement and participation from the team, an increased commitment, motivation, and a desire for responsibility can be expected (Lundene and Mohagheghi, 2018).

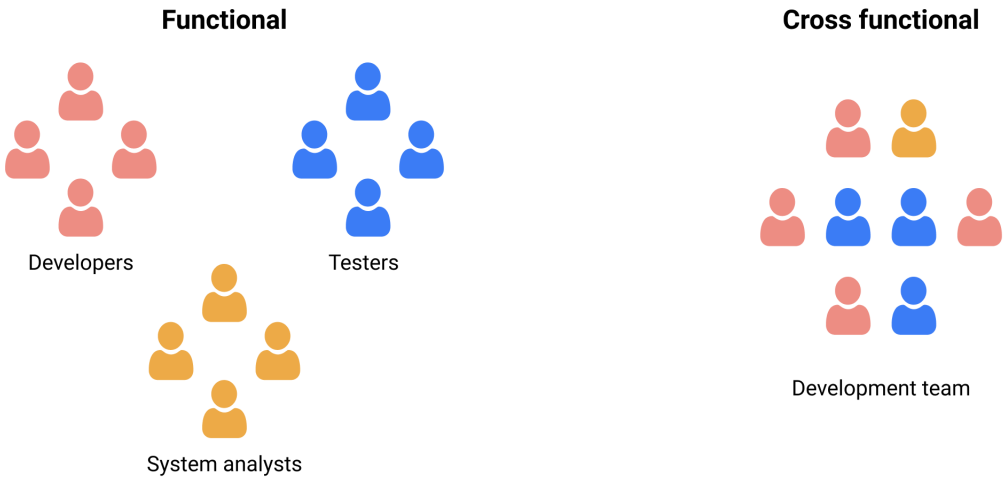

Figure 2.2: Example of cross-functional teams

Wheelan and Hochberger (1996) define a team as *"a workgroup that has shared goals and effective methods to achieve them"*. A workgroup is further defined as a group of members that want to create a shared view of group goals and develop a structure to achieve these goals. This implies that workgroups in many organizations might not be teams (Gren, Torkar and Feldt, 2017). Moreover, to which degree a working group or team is mature has also shown to increase effectiveness, where maturity is dependent on the relationships between individuals (Gren, Torkar and Feldt, 2017; Wheelan, Murphy, Tsumura and Kline, 1998).

2.3.1 Autonomous teams

Software development companies experience an increasing amount of complexity that demands cross-functional autonomous teams. The structure of the team has to support rather than impede, which requires clear boundaries. This structuring is done by putting all needed skills within the team, significantly increasing team size. As a result, it introduces challenges regarding shared leadership and shared decision-making (Stray, Moe and Hoda, 2018). An example of a cross-functional autonomous team structure is illustrated in figure 2.2.

Dybå and Dingsøyr (2015) provided a briefing on agile project management, based

on an extensive amount of large-scale industrial studies. They elaborate on the transition from traditional management and direct supervision to the agile approach using autonomous teams and decentralized decision making. With authority at the operations level, decisions are made faster, and problem-solving is more efficient.

Including these self-managing teams, the briefing also explained how a redundancy of knowledge within the team also is an important factor. Lundene and Mohagheghi (2018) emphasized this as one of the pre-conditions of autonomous teams, enabling better capability to adapt to changing situations and knowledge sharing. As the autonomy increases, the team is expected to perform better when task interdependence is high. Meanwhile, if the interdependence is low, it can negatively affect the team's efficiency (Stray, Moe and Hoda, 2018).

2.3.2 Distributed teams

Working with large-scale agile requires a substantial amount of coordination, considering the number of teams and individuals working simultaneously in a software development organization. Factors such as dependencies among people, tasks, synchronization, and schedules are continuously emerging challenges to be managed. These are compounded with teams being geographically distributed, resulting in increased development time and coordination challenges being more difficult to manage compared to co-located teams (Espinosa, Slaughter, Kraut and Herbsleb, 2007). Berntzen and Wong (2021) presents challenges of distributed teams such as reduced communication and coordination quality, resulting in reduced performance and work engagement. Team members who are not co-located communicate less frequently and may lack knowledge about each other's situations, potentially reducing trust, commitment, and knowledge sharing (Berntzen and Wong, 2021).

In a large number of project failures where coordination has been an issue, it is usually in a large-scale agile situation and with distributed teams. Because members of a distributed team seldom meet physically, one could argue that its maturity is negatively affected. A study interviewing a set of coaches and managers of agile processes identified a lack of camaraderie and team identity as challenges within distributed teams (Gren, Torkar and Feldt, 2017). Accordingly, it is important to study how the dispersion of agile teams affects their performance and ability to coordinate (Espinosa, Slaughter, Kraut and Herbsleb, 2007).

2.4 OKR

Objectives and key results (OKR) is a goal-setting framework to define a specific set of objectives in an organization and measure its progress. Instead of spending months setting long-term goals, OKR is designed to help organizations 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" (Niven and Lamorte, 2016, p. 6). OKR provides benefits such as focus, the frequent establishment of priorities, and transparency. Introducing OKR enables transparency on an inter-team level, which allows teams to cross-functionally align, provide feedback, and collaborate (Niven and Lamorte, 2016).

In order to measure progress, the framework consists of a set of objectives and key results. The objectives are defined for a certain period, and key results are defined to progress towards these objectives continually through tasks. This is illustrated in figure 2.3.

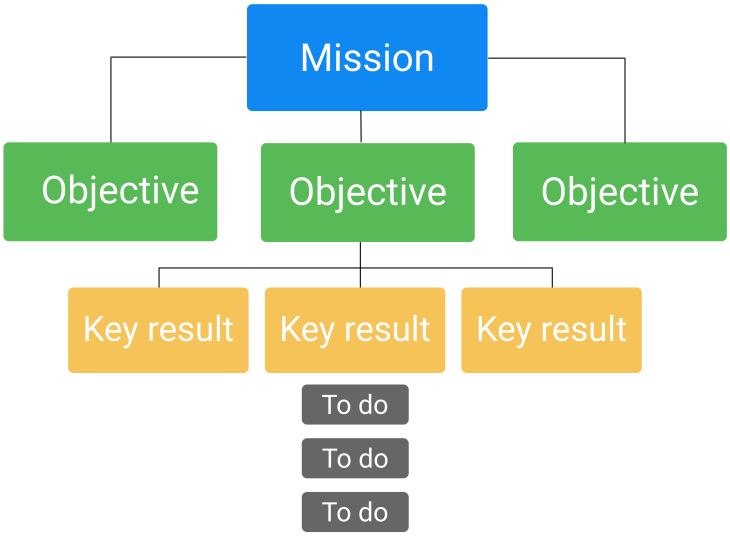

Figure 2.3: Figure displaying OKR through objectives and key results

2.4.1 Objectives

An objective describes in short terms what the team wants to do. A well-described objective should be obtainable within a quarter and represent a shared imagination of the team (Niven and Lamorte, 2016). It is defined as *"a concise statement outlining a broad qualitative goal designed to propel the organization forward in a desired direction."* (Niven and Lamorte, 2016, p. 8).

An example of an objective from Niven and Lamorte (2016) is *"Design a compelling website that attracts people to OKRs"*. They explain the importance of setting a qualitative and short objective, using plain language and which can be obtained within the time frame. The objectives should preferably start with a verb to provide direction and action and positive language to make it compelling and motivating.

2.4.2 Key results

Key results allow the team to measure their progress and are designed to show when the objective has been reached. It is defined as *"a quantitative statement that measures the achievement of a given objective"* (Niven and Lamorte, 2016, p. 8). Objectives can be vague or unclear, making it challenging to quantify high-quality key results.

Referring to the objective above (see 2.4.1), it is important to define how "compelling" translates into numbers, emphasizing the importance of context. Finding a good compromise between value and measurability is essential for the quality of a key result. Niven and Lamorte (2016) also recommends the implementation of scores for key results, describing to what extent the organization is likely to achieve it. The scores can communicate expectations and provide clarity and context into what progress looks like for a particular key result.

Method

The purpose of this chapter will be to assess the methodical approach for this thesis. The first section describes qualitative research and the methods used, consisting of interviews, observations, and how to conduct them. The following section will comprise the analysis and categorization of the findings that emerged from the previous section, including its reliability and validity. These findings will be further discussed in forthcoming sections in the thesis.

3.1 Qualitative research

O'Leary, 2017 describes qualitative data as represented through words, pictures, or icons, whereas quantitative data is represented through numbers and statistical analysis. He mentions that qualitative research often focuses on a small number of in-depth cases. This allows multiple realities instead of the singular truth that quantitative research usually relies on (O'Leary, 2017).

Traditionally, qualitative research involves three methods: case study, ethnographic study, and grounded theory (Gerring, 2006). The past year with the entire world confined in a global pandemic, there has been a heavy increase in the work performed from home and the distribution of teams. These rapid changes are having a serious effect on how coordination is performed and are important to highlight. Case studies seek to explain certain circumstances, such as "how" or "why" some phenomenon works (Yin, 2017, p. 33). This thesis seeks to address this, resulting in a multiple-case study being applied, with the focus of uncovering how different coordination mechanisms are used and why within two autonomous teams.

3.1.1 Multiple-case study

In order to facilitate the relevance of the findings, a logical plan connecting the empirical data to the research questions, and eventually to its conclusion, is of absolute interest (Yin, 2017, p.60). As mentioned earlier, this study aims to answer how different coordination mechanisms are used in two autonomous teams and why. A decision to follow a dependency taxonomy from Strode, Huff, Hope and Link, 2012 was taken before the initiation of the data collection. This allowed me to assemble a research plan following the taxonomy, facilitating the emergence of an observation protocol and interview guide.

A case study is an empirical research method for investigating contemporary phenomenons, or cases, in-depth within its real-world context. This is mainly when

the boundaries between context and phenomenons are ambiguous or not apparent (Yin, 2017, p. 45). To put it another way, a case study is relevant when you want to understand a real-world case and its contextual conditions. Meanwhile, both single- and multi-case studies are considered by Yin as case study designs within the same methodological framework (see figure 3.1). With this presumption proceeded the choice of a multiple-case study, designed for and revolving two different autonomous teams later referred to as Team Alpha and Team Bravo.

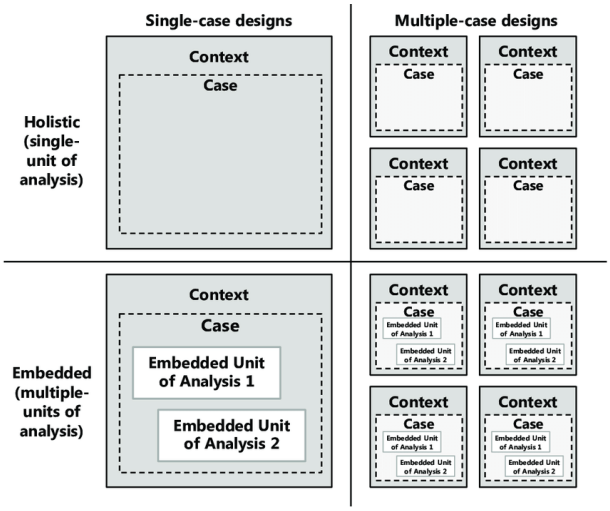

Figure 3.1: Different case study types (COSMOS Corporation)

3.1.2 Research site

As mentioned in Chapter 1, a software consultancy agency was chosen as the research site. This Swedish consultancy agency has a Norwegian department, which is where this study was conducted. As I had recently worked in a summer internship there, a dialogue was initiated 13th of October 2020. Using Slack, a digital communication tool for channel-based instant messaging, a message was sent to one of the department managers about potential projects for the study. There was an early interest in the possibilities of studying one or more teams in an agency within a large Norwegian municipality.

On the 21st of October 2020, one agency agreed to this and the possibility of examining a large and interesting team, being Team Alpha. A dialogue was started with the product owner the following day, further referring me to the team lead of Team Alpha. A meeting with the team lead was held 6th of November 2020, discussing and planning the data collection process and sending a request to access Slack and other necessities for the observations to begin. This was quickly arranged, and as a result, the observations started remotely on the 9th of November 2020. An overview of this is illustrated in figure 3.2.

Figure 3.2: Timeline of research period

3.2 Data collection

Two common methods in qualitative research are interviews and observations. They provide the necessary depth and information to respond to the research question and corresponding subjects, primarily through observing daily agile processes and investigating lived experiences of the team. Bell (2010) emphasizes the importance of "informed consent", requiring all researchers to carefully prepare, explain and consult before any data-collecting. She further explains:

> "Research ethics is about being clear about the nature of the agreement you have entered into with your research subjects or contacts." (Bell, 2010, p. 45-46)

Accordingly, an application was sent to the "Norwegian Centre for Research Data"(NSD) before initiating any research or data collection. This application defined the foundation for my research, requesting permission to perform observations and interviews concerning my research agenda. The application was sent 17th of August and approved 25th of August 2020. Leading into the data collection, the participants were also presented with a consent form that explained the scope and purpose of my research, ensuring they were comfortable and approved with what this entailed.

To increase the potential benefits of the selected data collection methods, Yin presents four principles. If followed correctly, they are to aid in the process of establishing construct validity and reliability (Yin, 2017, p. 170). My approach to these principles can be viewed in table 3.1.

Principle	Approach
Use multiple sources of evidence	The data was collected through interviews, observations, and additional documents and resources.
Create a case study database	All of the raw collected data was stored safely on Dropbox, ready for analysis. The anonymized data was further studied and processed, is stored on Google drive.
Maintain the chain of evidence	All data were stored in folders with intuitive names, granting a clear overview and easy access to data from different stages of the study.
Exercise care when using data from electronic sources	Combined with the data collected through observations and interviews, chat logs, and the additional resources obtained are cross-checked to ensure their validity

Table 3.1: Approach on the basic principles of data collection

3.2.1 Observation

The initial phase of the data collection consisted of the observation of the regular meetings in Team Alpha. With this insight, I learned about their process and work, significantly increasing my understanding of their product and organization. What the participants say they do, compared to what they actually do, can differ to a very large extent. The observation tends to take place in the real world, as opposed to a constructed research world. Observation allows a sense of reality and works through the complexities of social interactions. It is defined as a method of data collection that relies on a researcher's ability to gather data through his or her senses (O'Leary, 2017).

The observations lasted from the 9th of November to the 7th of December 2020, consisting of 21 meetings throughout the time period (see table 3.2). During this period, the team worked remotely because of the global situation of the pandemic, and as a result, I participated in their digital meetings. A structured approach to the observations was chosen, following an observation protocol. This protocol included the number of participants, time stamps, content, and any abnormalities if any. It can be challenging to convert the observed situation of the participants into a research tool. The complexities of the social interactions proved to be harder to grasp through the observations of the digital meetings. However, they were still a good experience and valuable addition to my data collection. Observation is narrowed by understanding and what we manage to take in through our senses. If done correctly, it can provide the same rich, in-depth, verbal, and non-verbal qualitative data as interviews (O'Leary, 2017).

Meeting	Amount
Daily standup	15
Sprint planning	5
Retrospective	1
Total	**21**

Table 3.2: An overview of the meetings observed in Team Alpha

Moreover, the observation has the potential to provide other valuable data and behavior that the participants are not aware of themselves, making it a good addition to the data collection. This made me well equipped to handle the next phase of the data collection, allowing me to create a comprehensive interview guide covering a variety of relevant subjects. These subjects included their individual roles, coordination, communication, meetings, and their process. This facilitated good insight in their work, ultimately granting a solid research foundation in combination with the observation material.

3.2.2 Semi-structured interviews

The primary source for data collection in this study is interviews. As an in-depth data collection method, interviews provide rich, non-verbal as well as verbal data. It is a flexible method that enables the exploration of tangents and, in a structured manner, provides the generation of standardized, quantifiable data. Interviews are a method of data collection that allows the researcher to seek open-ended answers related to questions, subjects, or themes (O'Leary, 2017). Moreover, Bell, 2010 explains the values of an interview response:

> "The way in which a response is made (the tone of voice, facial expression, hesitation, etc.) can provide information that a written response would conceal." (Bell, 2010, p. 157)

Further, semi-structured interviews are described as interviews with a more flexible approach. In combination with the interview guide, the semi-structured strategy gives the researcher the freedom to shift the interview direction to follow the natural flow of the conversation. With the flexibility from the semi-structured method, a strategic exploration can be conducted on exciting topics that might surface (O'Leary, 2017). A carefully constructed interview plan provides the intended data and structure to the interview and can be found in the appendix. The team members from both rounds of interviews were contacted either through Slack or email, requesting their participation and further agreeing on an appropriate time to schedule the interview.

The first round of interviews was held with Team Alpha from the 4th until the 15th of December 2020. All interviews were arranged remotely and carried out without any problems of any sort. The interviews are presented in table 3.3.

Role	Time at team	Date	Duration
Tech lead	2 years	04.12.20	55:45
Team lead	1.5 years	09.12.20	53:07
Data scientist	2 years	09.12.20	49:26
UX designer	1.5 years	10.12.20	58:15
Front-end developer	3 months	11.12.20	47:48
Front-end developer	1 year 2 months	14.12.20	51:40
Back-end developer	1.5 years	07.12.20	28:50
Back-end developer	8 months	14.12.20	58:19
Back-end developer	2 years	14.12.20	38:19
Back-end developer	2 years	15.12.20	42:44
Average	**1 year 6 months**		**48 minutes**

Table 3.3: Overview of interviews held in Team Alpha

The second and last round of interviews were held with Team Bravo, from 1st until 16th of February 2021 (see table 3.4). All interviews were successfully held remotely, except for one, which unfortunately was canceled midway because of a connection issue. The second half of the interview was answered through email.

Role	Time at team	Date	Duration
Developer	1 year 4 months	01.02.21	37:44
Developer	2 years	04.02.21	1:11:04
Developer	4 months	04.02.21	34:45
Developer	3 months	04.02.21	-
Tech lead	4 months	09.02.21	39:32
UX designer	9 months	10.02.21	48:54
Team lead	3 years	16.02.21	46:01
Average	**1 year 2 months**		**45 minutes**

Table 3.4: Overview of interviews held in Team Bravo

Both rounds of interviews were held in Norwegian and with a predefined meeting length of 1 hour, the interviews ranged in length from 28 minutes to 71 minutes long. A video conferencing program called Zoom was utilized, allowing easy access to record the interviews with permission from the interviewee. The average interview length in Team Alpha was 48 minutes and 45 minutes for Team Bravo.

3.3 Data analysis

One of the underlying challenges after collecting data is to best preserve the richness and quality of it. Therefore, the analytical part is a delicate process, where we are

moving from a qualitative data set to a concrete understanding and interpretation of the participants and their situation. O'Leary (2017) mentions the main steps in the process of reflective analysis. Initially, it starts with identifying biases and noting overall impressions. This is followed by reducing, organizing, and coding the data. Then, a systematic search for patterns and interconnections. Further, the themes must be mapped and built. Finally, one must build and verify theories, in addition to concluding. This is all while keeping the research question, aims and objectives, methodological constraints, and theory clearly in mind (O'Leary, 2017).

The analytical process was started by uploading the transcribed material into the systematic coding tool Nvivo, a qualitative data analysis software. One analytical strategy proposed by Yin is the reliance on a theoretical proposition. As the presented theory of Strode shapes my data collection in Chapter 2, it has yielded analytical priorities. Following this theoretical proposition, the analysis was organized and helped me point out relevant contextual conditions to be described (Yin, 2017, p. 216). The findings are presented in Chapter 5.

3.4 Reliability and validity

Once the process of collection and analysis is complete, the dependability and quality of the data must be sufficient. Four tests are regularly used to determine the quality of most empirical research and its design, being reliability, construct-, internal- and external validity. This section will present these factors and how the collected material reflects on these criteria.

3.4.1 Reliability

Consistency, often referred to as reliability, is to what degree the research can procreate results given constant circumstances. Bell (2010) comes with the following definition: *"Reliability is the extent to which a test or procedure produces similar results under constant conditions on all occasions"*(Bell, 2010, p. 117). The circumstances in which these procedures were held might change. Similarly, the data collection methods, in this case, will differ between interviews and observations. Crescentini and Mainardi (2009) further explains:

> "If the researcher has made the process clear and transparent, the question of reproducibility can better be treated. For example, if the population's characteristics are clearly described, it is easier to imagine how the results can be extended or reproduced in a similar population. However, it is obvious that a replication of a process that involves the same researchers should lead to same results, but in qualitative studies what is done, seen, and heard (and also what will happen) will not be precisely the same as the original information." (Crescentini and Mainardi, 2009, p. 437)

Yin defines reliability as the ability to demonstrate the repetition of the operations in the study while yielding the same results (Yin, 2017, p. 78). Case study protocols, developing a case study database, and maintaining a chain of evidence are factors to increase reliability and decrease errors and biases. Based on these principles, the data collection is performed and documented with regard to and concern for its reliability.

3.4.2 Validity

While reliability is about consistency, it is critical to aim this within the correct field of research. Validity is a more complex concept, and often the definitions differ to some extent. Essentially, it reassures that the correct results are obtained, which is usually how definitions describe it. O'Leary (2017) defines it as *"When we have validity, we know we are measuring what we intended to measure and that we have eliminated any other possible causal relationships"*(O'Leary, 2017, p.64).

Construct validity is referred to as *"identifying correct operational measures for the concepts being studied"* (Yin, 2017, p. 78). According to Yin (2017), having multiple sources of evidence and having key informants reviewing the draft report of the case study are enablers of this.

Internal validity refers to *"seeking to establish a causal relationship, whereby certain conditions are believed to lead to other conditions, as distinguished from spurious relationships"* (Yin, 2017, p. 78). Considering this is only relevant for explanatory or causal case studies, assessing this is not relevant.

External validity is *"showing whether and how a case study's findings can be generalized"* (Yin, 2017, p. 78). The strategy for ensuring this is in regards to the research design. In single-case studies, this entails applying a theory while making use of replication logic in multi-case studies. The general idea is augmenting the study design with "why" and "how" questions, ultimately making it easier to arrive at an analytical generalization.

Research context

This chapter will describe the two teams of the multiple-case study, the different roles and meetings within and the organization, and their overall goal.

4.1 Organization and project case

The project case is an agency within a large municipality in Norway. Comprising a set of product teams, they are aimed to be a driving force in a digital transformation and contribute to the municipality being able to realize its vision for digitization. It was established as a project in 2017 but was further organized as an agency from 01.01.2020. This commitment emphasizes that developing good joint solutions for the inhabitants across sectors in the municipality is a lasting investment.

The case comprises six departments and seven product areas, consisting of 11 permanent and three temporary teams structured with people from multiple departments. These product teams are cross-functionally structured to 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.

The two teams studied in this multiple case study will be referred to as "Team Alpha" and "Team Bravo" and are product teams within the agency, contributing to the vision of digitized and sustainable solutions. This is illustrated in figure 4.1. Both teams follow Objective Key Results (OKR), a framework that the entire agency also utilizes. Accordingly, they have to adapt their work routines to fit the principles of OKR, which comprises aligning their tasks under key results. This allows for measuring progress against objectives, set within each team once every quarter to fit under the higher levels of objectives on the organizational levels.

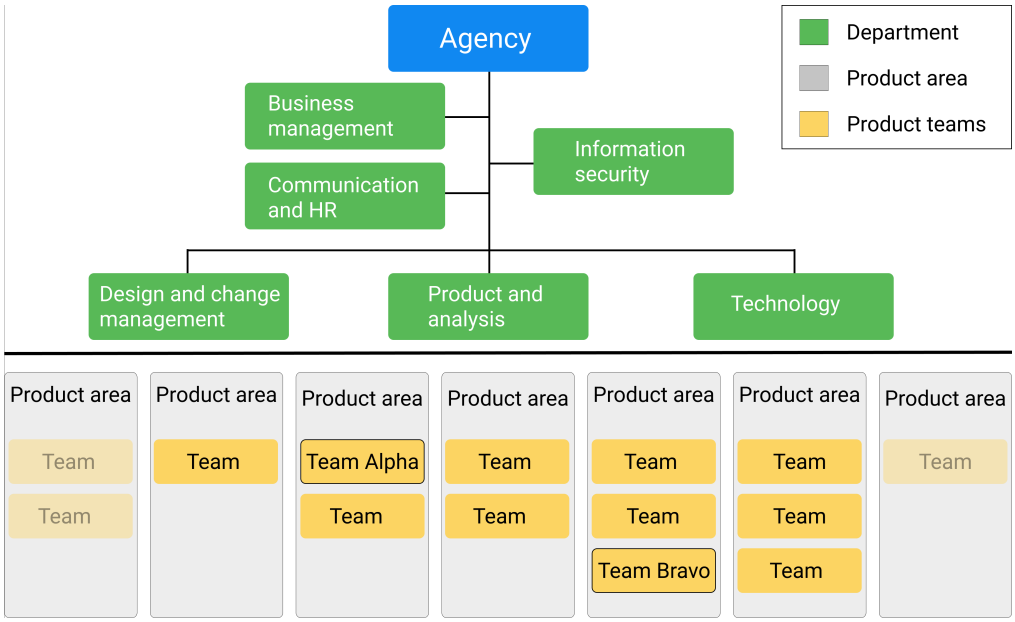

Figure 4.1: Structure of the agency

4.2 Team Alpha

Team Alpha was the first team studied for the thesis and the larger one as well. Their goal is 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 includes making intuitive and secure solutions, good documentation, and ensuring that the different agencies know this, ultimately enabling the citizens to have access to better solutions. Team Alpha has been operational since August 2018 and has since then been redefined from a single product team to a product area because of the emerging size. This entails a pretty substantial scope, resulting in a large team and different roles.

4.2.1 Roles

The product area is, as mentioned, quite large and, as a result, in need of a variety of roles. How they are utilized, and function in different large-scale settings might differ from context to context. This is why it is important to outline and is detailed in table 4.1.

4.2.2 Meetings

Team Alpha has a few regular meetings, at present being held digitally because of the distribution of the team. Each meeting, the team uses a face cam and mic, allowing better flow and a better understanding of body language. The team lead acts as chair of the meeting, ensuring structure following the agenda for the relevant meeting. Team Alpha works rigidly and performs their work in one-week sprints. An overview of the meetings is presented in table 4.2

Role	Amount	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.

Table 4.1: Roles in team Alpha

Meeting	When	Description
Sprint planning	Mondays	Discuss status and decide which key results to be prioritized in the following sprint, often with a few predefined suggestions from the product owner to align the team with the objectives of the agency.
Daily standup	Tuesday - Friday	Each of the team members gives a small update on the work that has been done, what is in progress and any potential problems.
Retrospective	Once a month	Discuss and vote for product and process related practices to start, stop, and continue doing. Outcome of meeting is a set of actions to follow up on.
OKR workshop	Once a quarter	Discuss status and set new objectives for the upcoming quarter, as well as creating corresponding key results.

Table 4.2: Meetings in team Alpha

4.3 Team Bravo

The second team investigated in this study, Team Bravo, was started in January 2019. Being a total of seven team members currently, and five at the minimum, their size has over time mostly remained the same. As a part of the agency's vision to provide valuable services to the citizens of the municipality, their product entails offering information

to citizens as they need it. They are ultimately ensuring that the correct individuals receive relevant information when they need it. The product and technology of Team Bravo have recently been through a modernization process aimed at improving systems and introducing serverless solutions. As a result, the current process of creating and integrating new functionality and standards is faster and less complicated.

4.3.1 Roles

Team Bravo is confined to a more limited set of roles, compared with team Alpha. Their product has not experienced the need for a proper user interface until recently. As a result, they have no developers dedicated to a singular role as a front- or back-end developer. Further, Team Bravo does not have a product owner to report to at the current time. Accordingly, the team lead has most of this responsibility, resulting in more decisions being made on team-level, thus making Team Bravo very autonomous. A description of the roles in team Bravo is presented in table 4.3.

Role	**Amount**	**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	1	Works with illustration, design, and collection of data, and insight from other teams and agencies.
Developer	4	Work consists of building and maintaining the product, functioning as full-stack developers with high redundancy of skills. This includes front-end, back-end, and cloud services, among other things.

Table 4.3: Roles in team Bravo

4.3.2 Meetings

Team Bravo is not confined to sprints as opposed to Team Alpha, and work can continue over multiple weeks if deemed necessary and per the objectives. They have also introduced a chairman, which is a person responsible for ensuring that everyone is heard during a meeting. When someone raises their hand, the chairman is responsible for this and facilitates discussions in an orderly manner, reducing interruptions between the team members. Their meetings are detailed in table 4.4

Meeting	When	Description
Backlog meeting	Mondays	Discuss status and decide which key results to be prioritized, often with a few predefined suggestions from the team lead and tech lead to align the team with the objectives of the agency.
Written slackup	Tuesday - Friday	The team members share a written update with the rest of the team similar to a daily standup.
Open forum	Wednesday	With a predefined agenda, the team can show demos, discuss. or do clarifications on pressing matters. The meeting is canceled if nothing is submitted to the agenda beforehand.
Retrospective	Once every two weeks	Every second week, the backlog meeting is combined with a retrospective meeting. The meeting differs every other time between a standard retrospective meeting and a health-oriented retrospective meeting.
OKR workshop	Once a quarter	Discuss status and set new objectives for the upcoming quarter, as well as creating corresponding key results.

Table 4.4: Meetings in team Bravo

4.4 Communication and tools

Due to the global situation caused by the pandemic, Team Alpha and Bravo find themselves working full-time distributed. While working remotely, both teams utilize communication tools such as Slack, Google meets, and Whereby, allowing them to perform their meetings and workshops as intended. Tools such as Trello and Jira are used to visualize their workloads in kanban boards and the status of the different tasks, including Confluence and Github pages for various documentation. Furthermore, Miro and Google slides are used in conjunction with workshops and meetings, functioning as open workspaces that allow the teams to discuss and collaborate digitally.

Results

Chapter 2 describes the coordination theory from Strode, Huff, Hope and Link (2012), where coordination mechanisms are categorized through eight different coordination strategy components; synchronization activity, synchronization artefact, boundary spanning activity, boundary spanning artefact, availability, proximity, substitutability, and coordinator role. It is further explained how Strode (2016) divides dependencies into three categories; knowledge, process, and resource.

Chapter 4 describes Team Alpha and Bravo. Details about the agency and team roles, tools, and regular meetings are explained in separate contexts to illuminate their characteristics.

The management of dependencies is central in coordination, and it is important to understand how coordination mechanisms are used to address this. This chapter will present the emerged coordination mechanisms and dependencies using the proposed taxonomy from Strode (2016). Based on these results, I aim to answer the research question:

"How are dependencies managed in distributed teams in large-scale agile?"

An overview of Team Alpha and Bravo's dependencies and coordination mechanisms can be viewed in figure 5.1 and 5.3. Deriving from the data material described in Chapter 3, 37 coordination mechanisms and 123 dependencies were identified in Team Alpha, and 34 coordination mechanisms and 108 dependencies in Team Bravo.

5.1 Dependencies and coordination mechanisms

In this section, the identified coordination mechanisms and dependencies are presented for team Alpha and Bravo. When categorizing the emerging coordination mechanisms, the boundary spanning activities and artefacts have been regarded as activities and artefacts outside the team. This is different from Strode, Huff, Hope and Link (2012), where boundary spanning is defined as external to the project. The following sections are grouped by the main dependencies; knowledge, process, and resource. Many coordination mechanisms are multi-purpose, meaning they manage more than one dependency. To present the results in an organized manner, each section will present the coordination practices with its best-matched dependency. Both teams have many of the same coordination mechanisms but will be presented only once. Any differences with regards to coordination mechanisms between the two teams will be detailed.

5.1.1 Team Alpha

Strategy components	Coordination mechanisms(37)	Knowledge				Process		Resource		Total
		Expertise	Require-ment	Task allocation	Histo-rical	Activity	Business process	Entity	Tech-nical	
Synchronization activities	Daily standup	✓		✓		✓		✓		4
	Pre-sprint planning	✓	✓				✓			3
	Sprint planning	✓	✓	✓			✓			4
	Sprint		✓			✓	✓	✓		4
	Retrospective	✓		✓	✓					3
	OKR workshop	✓	✓		✓		✓	✓		5
	One-on-one meetings	✓	✓			✓	✓			4
	Ad hoc communication	✓	✓	✓		✓		✓		5
Synchronization artefacts	JIRA	✓		✓		✓				3
	Objectives	✓	✓	✓			✓			4
	Key results	✓	✓	✓		✓				4
	Tasks	✓		✓		✓			✓	4
	Retrospective actions	✓		✓			✓		✓	4
	Github pages		✓		✓					2
	Backlog		✓	✓						2
	Pull request	✓	✓					✓	✓	4
	Collaboration tools	✓		✓				✓	✓	4
	Whiteboard	✓	✓							2
	Chat logs		✓		✓					2
Boundary spanning activity	OKR training	✓	✓					✓		3
	Inter-team meetings	✓						✓		2
	Team lead meetings				✓		✓	✓		3
	Ad hoc communication	✓	✓	✓		✓		✓		5
Boundary spanning artefact	OKR tracker		✓	✓			✓		✓	4
	KPI						✓		✓	2
	Insight reports		✓		✓	✓	✓			4
	Status reports from management							✓		1
	Support tickets	✓			✓	✓				3
	Chat logs		✓		✓					2
Availability	Full-time team	✓								1
	Constantly online	✓	✓					✓	✓	4
Proximity	Distributed teams						✓	✓		2
Substituability	Rotating support role	✓			✓					2
Coordinator role	Tech lead	✓	✓					✓		3
	Team lead	✓	✓	✓				✓		4
	Product owner	✓	✓	✓			✓	✓		5
	Data scientist	✓	✓					✓		3
	UX designer	✓	✓					✓		3
		27	24	15	9	10	13	18	7	
	Total	75				23		25		123

Figure 5.1: Dependency taxonomy of team Alpha

As mentioned before, a total of 37 coordination mechanisms and 123 dependencies were identified in Team Alpha. The knowledge dependency constitutes 61% of the total dependencies, equivalent to a total of 75 dependencies. In comparison, the resource and process dependency constitutes 25 and 23 dependencies individually. This is illustrated in the figure below, and a total overview for Team Alpha can be seen in figure 5.1.

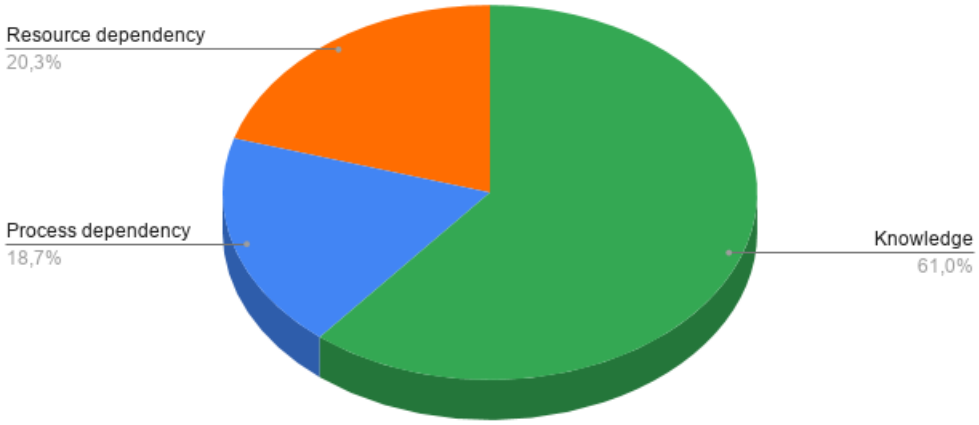

Figure 5.2: Distribution of dependencies in team Alpha

5.1.2 Team Bravo

Strategy components	Coordination mechanisms(34)	Knowledge				Process		Resource		Total
		Expertise	Require-ment	Task allocation	Histo-rical	Activity	Business process	Entity	Tech-nical	
Synchronization activities	Written slackup	✓		✓		✓		✓		4
	Backlog meeting	✓	✓	✓			✓			4
	Backlog grooming	✓	✓				✓			3
	Retrospective	✓		✓		✓				3
	Open forum	✓	✓	✓						3
	OKR workshop	✓	✓		✓		✓	✓		5
	One-on-one meetings	✓	✓			✓	✓			4
	Ad hoc communication	✓	✓	✓		✓		✓		5
Synchronization artefacts	Trello	✓		✓		✓				3
	Confluence		✓		✓					2
	Objectives	✓	✓	✓			✓			4
	Key results	✓	✓	✓		✓				4
	Tasks	✓		✓		✓			✓	4
	Retrospective actions	✓		✓			✓		✓	4
	Collaboration tools	✓		✓				✓	✓	4
	Backlog		✓	✓						2
	Pull request	✓	✓					✓	✓	4
	Whiteboard	✓	✓							2
	Chat logs		✓		✓					2
Boundary spanning activity	OKR training	✓	✓					✓		3
	Inter-team meetings	✓						✓		2
	Team lead meetings				✓		✓	✓		3
	Ad hoc communication	✓	✓	✓		✓		✓		5
Boundary spanning artefact	OKR tracker		✓	✓			✓		✓	4
	Insight reports		✓		✓	✓	✓			4
	Status reports from management							✓		1
	Chat logs		✓		✓					2
	Full-time team	✓								1
Availability	Constantly online	✓	✓					✓	✓	4
	Distributed teams						✓	✓		2
Proximity	Fullstack developers	✓								1
Substituability	Tech lead	✓	✓					✓		3
Coordinator role	Team lead	✓	✓	✓				✓		4
	UX designer	✓	✓					✓		3
		25	22	15	6	9	10	15	6	
	Total	68				19		21		108

Figure 5.3: Dependency taxonomy of team Bravo

Similar to Team Alpha, the knowledge dependency for Team Bravo makes up the majority of the dependencies identified. Totaling 68, knowledge dependencies constitute 63% of all dependencies. Further, the process dependency has registered 19 cases and the resource dependency a total of 21 cases. The frequencies are illustrated in the figure below.

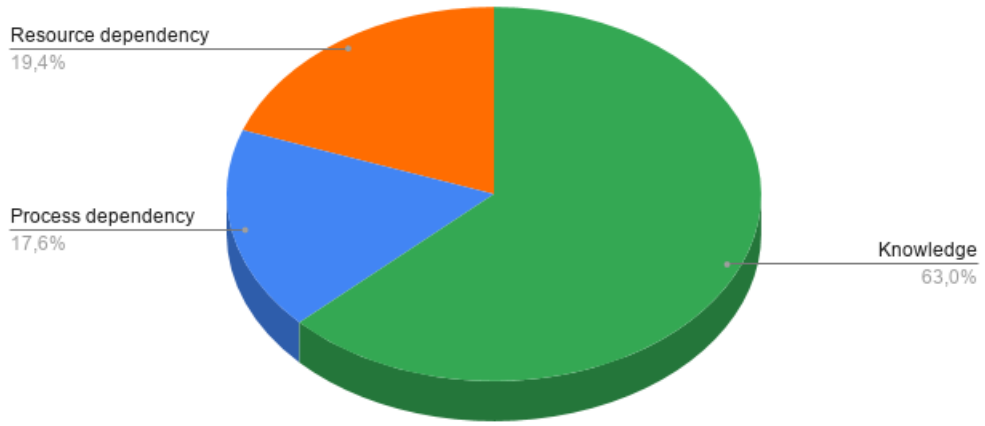

Figure 5.4: Distribution of dependencies in team Bravo

5.2 Knowledge dependency

The knowledge dependencies were most evident in both teams and are defined as *"When a form of information is required for a project to progress"*. Totaling over 60% of dependencies in both teams (see figures 5.2 and 5.4), most coordination mechanisms managed multiple dependencies. Table 5.1 shows an overview of the best-matched mechanisms covering multiple knowledge dependencies and will be explained individually. The additional coordination mechanisms with fewer knowledge dependencies will be presented in the following sections relevant for them.

Strategy component	**Coordination mechanism**	**Total dependencies**	**Team**
Synchronization activity, Boundary spanning activity	Ad hoc communication	5	Both
Synchronization activity	OKR workshop	5	Both
Synchronization artefact	Objectives	4	Both
	Key results	4	Both
	Retrospective actions	4	Both
Coordinator role	Product owner	5	Alpha

Table 5.1: Best matched coordination mechanisms for knowledge dependencies

Ad hoc communication

Although both teams held regular inter-team and intra-teams meetings, there was still a substantial amount of communication performed ad hoc, mainly using Slack. The ad hoc communication coordination mechanism comprises both within the individual teams but also in inter-team contexts. This is why it is listed as both synchronization and boundary spanning activity. This communication might be about the implementation of new technology and locating domain knowledge, pair programming and debugging, and anything else work-related, which requires a person to locate knowledge outside of regular meetings. The team lead in Bravo emphasized the following: *"You can sit all day just chatting. There is so much in all possible channels that you can end up doing nothing but just reading updates and questions and answering and communicating with people. So we spend a lot more time on digital communication than before"*.

When teams work distributed, ad hoc communication constitutes most communication during work hours next to scheduled meetings. It addressed knowledge dependencies such as expertise, requirement, and task allocation. This is done by asking or locating someone with the particular knowledge needed (expertise and domain) and can be a time-consuming process if who is doing what is unknown (task allocation). Ad hoc communication also managed activity dependency (process dependency) as the person could be stuck on a task which is why the communication was initiated, and entity dependency (resource dependency) because specific people or resources might not be available.

OKR workshop

The OKR workshop was arranged once a quarter by both teams. It was typically held co-located, but with regards to the pandemic, it was done digitally. OKR served as the foundation of prioritization of workload in team Alpha and Bravo, and the OKR workshop managed knowledge dependencies such as expertise, requirement, and historical. It further managed dependencies such as business process dependency (process dependency) and entity dependency (resource dependency).

The team members managed expertise and historical dependencies, as it required a perspective of past decisions and specific knowledge about the product to create optimal OKRs. The requirement dependency was managed by the team lead and tech lead, as they had discussed the current status and possible priorities prior to the meeting. Parts of the agenda of the OKR workshop entailed that the team members discuss this proposition, which also managed the entity dependency.

Aimed at setting a direction for the teams in the upcoming quarter, the OKR workshop addressed five dependencies. It was identified as one of the coordination mechanisms managing the most dependencies and is essential for the teams and their work in the following period.

According to the OKR framework, the team is to set these objectives and quantify them through the key results. One of the hardest parts about using OKR was stated by multiple team members from both teams, being the quantifying of objectives through key

results and the corresponding choice of words. It was the deciding factor of what tasks would be prioritized in the upcoming quarter, which further emphasized the need for high quality and intuitive objectives and key results.

Someone from team Alpha stated: *"I think OKR is difficult. It is useful for maintaining focus, but it is hard to create good, measurable key results which make sense"*. Another member from Alpha further explained: *"This has been a problem for us. Typically, we cannot progress a key result in a single week, but suddenly after three weeks, when tasks are finished, it is updated from 0 to 5"*. The difficulties experienced with OKR were similar in both teams. The OKR workshop was a challenging and time-consuming meeting from a complex framework, but both teams were successful with OKR and stated that they were continually improving.

Objectives

Objectives are what both teams worked towards to progress, which teams set for each quarter through the OKR workshop explained above. The optimal objective is written in plain and qualitative language, short, and should be obtainable within a quarter. They managed four dependencies; expertise, requirement, task allocation (knowledge dependencies), and business process (process dependency). The objectives are dependent on different objectives from management to align teams in the direction envisioned by the agency and specific knowledge about the domain and tasks to be created optimally.

Key results

In order to work towards these objectives, the teams progressed key results continually throughout the quarter as they completed relevant tasks. The optimal key result is a quantitative statement that concisely represents how to achieve an objective. Objectives are qualitative by nature and can, as a result, be vague. Although the teams defined their objectives, it was challenging to create good and meaningful key results. This is mentioned above and was observed in both teams. Key results managed four dependencies; expertise, requirement, task allocation (knowledge dependencies), and activity dependency (process dependency). This is because they required specific domain and task information to create the key results and who was working with what. Key results were also dependent on defined objectives before they could be created.

Retrospective actions

Retrospective actions also emerged as a coordination mechanism that managed four different dependencies; expertise, task allocation (knowledge dependencies), business process (process dependency), and technical (resource dependency). It was regarded as a synchronization artefact mainly because it was generated through a retrospective meeting, serving as an important part of its function as a meeting. The retrospective meeting held in Team Alpha and Bravo was not utilized in the same rigid manner as intended in Scrum, where the meeting is only to focus on process-specific issues. In this

context, the retrospective actions could cover a broader scope of issues than what they usually would in standard agile processes.

When actions were defined following each retrospective, the responsibility for following up on the actions was distributed with regards to the technical aspect, domain knowledge, and possible expertise on the subject. Meanwhile, it was evident that to what extent the actions were pursued could vary. One team member from Team Alpha stated;

"They are written down but I have the feeling that they occasionally are forgotten. It may not be until the very next retrospective before we take a look at the list of actions and discuss what has been done since last time"

This was further mentioned in a couple of more instances from Team Alpha, where despite delegating responsibility, it did not hide the fact that the team had to perform work towards their product as well. This is in line with the business process dependency, as both teams had to prioritize their work regarding OKR. For example, moving Jira to the cloud had been defined as an action in a few instances in Team Alpha. Further, a member of Bravo explained: *"It's not always possible to boil it down to a specific task. It might as well be actions with regards to changing the structure of meetings"*. This was true for team Alpha as well, where they recently restructured their weekly planning meeting to make it more relevant for participants.

Considering the potential size or ambiguity of a retrospective action, it may be challenging to balance attention between following up on it and performing actual work. Difficulties aside, the retrospective actions were valuable assets for both teams.

Product owner

The product owner was the third coordination mechanism which had a total of five dependencies, being expertise, requirement, task allocation (knowledge dependencies), business process (process dependency), and entity (resource dependency). Only used in team Alpha, it was categorized as a coordinator role through the taxonomy and existed to communicate stakeholder interests, check status, and point teams in the right direction. Working tightly with the team lead and tech lead of Team Alpha, the product owner assisted in discussions of which key results to prioritize throughout the quarter. This is what ultimately decided many of the team's tasks during any point in time, which in turn managed requirement and task allocation dependency.

On the other hand, Team Bravo did not have a product owner. The product area for Bravo did not have a product owner at the moment, making them significantly more autonomous compared to Team Alpha. Decisions were more or less exclusively taken at team level, providing flexibility for them as a team. As a result, the team lead of team Bravo inherited a lot of this responsibility and functioned as a product owner for their team.

5.2.1 Expertise dependency

The coordination mechanisms that best matched the expertise dependency are listed table 5.2, mainly mechanisms that required specific knowledge or experience. The expertise dependency was managed by 27 coordination mechanisms in team Alpha and 25 mechanisms in team Bravo and is defined as *"Technical or task information is known only by a particular person or group and this affects, or has the potential to affect, project progress"*.

Strategy component	Coordination mechanism	Total dependencies	Team
Synchronization activity	Daily standup	4	Alpha
	Written slackup	4	Bravo
	Open forum	3	Bravo
Boundary spanning activity	Inter-team meetings	2	Both
Substitutability	Rotating support role	2	Alpha
	Fullstack developers	1	Bravo

Table 5.2: Best matched coordination mechanisms for expertise dependency

Daily standup

Tuesday to Friday, team Alpha arranged their daily standup meeting just before lunch. Following the predefined agenda, the team members shared what they had done, were doing, and possible hindrances. It was held remotely, and the team lead functioned as a meeting facilitator. The tech lead explained the following: *"The standups are ok, but it quickly becomes an update of what has been done since last, which is not as important. It should rather be about what you will be doing and potential problems"*. The meeting participants were encouraged to use a webcam, and the meeting was also the only daily scheduled interaction in team Alpha except for the planning meeting on Mondays. The daily standup managed expertise, task allocation (knowledge dependencies), activity dependency (process dependency), and entity dependency (resource dependency).

Written slackup

In contrast to the regular daily standup on video utilized by team Alpha, Bravo had recently changed to a written standup from Tuesday to Friday. They have named it a "slackup" and was essentially a regular standup but in a written format, managing the same dependencies as the standup. This meeting format allowed the team to avoid potential context switching during complex work. As a result, they could maintain their focus and share their status report when it was convenient.

Open forum

Open forum was a meeting reserved for occasions deemed necessary in team Bravo and managed expertise, requirement, and task allocation (knowledge dependencies). If an agenda was defined prior to the meeting, the meeting was arranged accordingly. If there

were no agenda or content for the meeting, it would be canceled. This flexible approach provided a platform for extra discussion on matters, which was a good way to save time and avoid digressions in other regular meetings.

Inter-team meetings

This was a meeting available for all teams in the agency and was one of the main contributors to information flow between teams. Every second inter-team meeting, the team lead had to share status from their team, which managed the expertise (knowledge dependency) and entity (resource dependency). Team members from any team in the agency could participate to get a status update of work, technology, and other relevant information. For instance, this facilitated ad hoc communication across teams if a team recently implemented new technology or framework. If this were of interest to other teams and their product, it would be a great platform to initiate knowledge sharing and problem-solving.

Rotating support role

The scope of team Alpha included both development of new functionality and the maintenance of the already existing system, which managed expertise and historical dependencies (knowledge dependencies). To handle this efficiently and promote cross-functionality, a rotating support role was introduced. Rotating every week, a member from team Alpha would be responsible for managing any potential support tickets. There were not a significant number of users yet, which allowed team Alpha to communicate and perform the support through Slack.

Fullstack developers

The developers of Team Bravo had a more comprehensive range of tasks than in Team Alpha. This was because the team had not reached the point where they required all-out front-end and back-end developers. The developers mainly worked with building and maintaining the product, and occasionally front-end, and as a result functioning as full-stack developers. Most development work consisted of setting up the infrastructure serverless in Amazon Web Services (AWS). As a result, there was a good amount of knowledge redundancy which managed expertise dependency, and promoted substitutability.

5.2.2 Requirement dependency

Table 5.3 presents the two coordination mechanisms which will be detailed in this section. They managed the requirement dependency because of the cases where they are utilized required domain-specific knowledge. As displayed in figure 5.1 and 5.3, the requirement dependency was managed by 24 coordination mechanisms in team Alpha and 22 coordination mechanisms in team Bravo. A requirement dependency is defined as *"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".

Strategy component	Coordination mechanism	Total dependencies	Team
Synchronization artefact	Backlog	2	Both
	Whiteboard	2	Both

Table 5.3: Best matched coordination mechanisms for requirement dependency

Backlog

Both teams utilized two different backlogs each; one priority backlog and one backlog for additional tasks. The prioritized backlog consisted of tasks that were to be prioritized in the following period regarding the objectives (OKR), and the standard backlog consisted of tasks that would be done eventually. The content of the priority backlog would be discussed and processed in the weekly sprint planning meeting for team Alpha and the backlog meeting for team Bravo regarding their respective OKRs. Including requirement dependency, the backlog managed task allocation (knowledge dependencies).

Whiteboard

With regards to the distributed work situation, most work and discussions were held remotely. During the summer of 2020, the agency opened for limited use of the offices. As a result, whiteboard sessions were sometimes held physically at the office. The whiteboard provided a simple solution to discuss and visualize without much overhead, which in addition to managing requirement dependency, also managed expertise dependency (knowledge dependency). It was a valuable alternative compared to digital discussions, where it would be much harder to read body language and contribute to the discussion without drowning out other voices.

5.2.3 Task allocation dependency

The coordination mechanisms most relevant to present in this section are presented in table 5.4. The mechanisms managed task allocation dependency and were all meetings where knowledge of current work is relevant. The task allocation dependency was managed by a total of 15 coordination mechanisms in both team Alpha and team Bravo (see figures 5.1 and 5.3), and is defined as *"Who is doing what, and when, is not known and this affects, or has the potential to affect, project progress"*.

Strategy component	Coordination mechanism	Total dependencies	Team
Synchronization activity	Backlog meeting	4	Bravo
	Sprint planning	4	Alpha
	Retrospective	3	Both

Table 5.4: Best matched coordination mechanisms for task allocation dependency

Backlog meeting

Every Monday, a status and backlog meeting was held in team Bravo. The team lead and tech lead prepared suggestions for which tasks to prioritize in advance, and status was given from each team member regarding their current work and potential difficulties, similar to a daily standup. This was followed by adjusting tasks in Trello, discussing the current workload and the potential necessity of moving additional tasks from the product backlog into the priority backlog regarding the OKRs. The backlog meeting managed expertise, requirement, task allocation (knowledge dependencies), and business process (process dependency), which in combination with the written slackup constituted most of the weekly coordination in team Bravo.

Sprint planning

Like team Bravo, team Alpha had a weekly meeting where they would have a plenary discussion of key results, the progression of various tasks, and which new tasks to initiate. A pre-planning meeting would be held in advance with the product owner, team lead, and tech lead to discuss what to be prioritized. The sprint planning meeting was divided into two parts, an initial general discussion of key results, followed by a more detailed and technical part. A problematic aspect observed from this meeting was discussions. Body language and possible bandwidth implications were factors that complicated digital discussions, making the propositions from the pre-planning unfortunately leading. Similar to the backlog meeting, the sprint planning managed expertise, requirement, task allocation (knowledge dependencies), and business process (process dependency), which in combination with daily standup also constituted most of the weekly coordination in team Alpha.

Retrospective

Both teams also used retrospective meetings to reflect on their current product and process, with a common goal of identifying practices to start, stop, and continue doing. The meeting was arranged once a month for both teams and managed expertise, task allocation (knowledge dependencies), and activity dependency (process dependency).

While team Alpha arranged individual retrospective meetings, team Bravo combined their weekly backlog meeting with the retrospective when it was due. Team Bravo also held a health retrospective every two weeks between the regular retrospectives. The health retrospective remained focused on social aspects, well-being, and the overall work process. A questionnaire would be given to each team member prior to the meeting, and the answers are discussed with regards to the potential room for improvement.

5.2.4 Historical dependency

Table 5.5 shows coordination mechanisms that managed the historical dependency. Documentation in the form of chat logs, Confluence or Github pages, provided an option

to view past decisions and discussions. The historical dependency was managed by a total of 9 coordination mechanisms in team Alpha and six mechanisms in team Bravo (see figures 5.1 and 5.3), and is defined as *"Knowledge about past decisions is needed and this affects, or has the potential to affect, project progress"*.

Strategy component	Coordination mechanism	Total dependencies	Team
Synchronization artefact, Boundary spanning artefact	Chat logs	2	Both
Synchronization artefact	Github pages	2	Alpha
	Confluence	2	Bravo

Table 5.5: Best matched coordination mechanisms for historical dependency

Chat logs

Chat logs were both synchronization and boundary spanning artefacts, deriving from ad hoc communication and meetings. They could contain written discussions that would be easy to find in Slack or perhaps the status from a written slackup in team Bravo. It was important to address this as a coordination mechanism because it covered a significant amount of communication within and across teams. The chat logs provided a historical perspective that might prove helpful in many situations and managed requirement dependency by saving specific domain knowledge available for later.

Github pages

Similar to chat logs, documentation was important to gain knowledge and granted easy access to past decisions and solutions. Github pages were used by team Alpha for their documentation. They managed historical dependency and requirement dependency (knowledge dependencies), explaining domain-specific information and being available to view for future decisions.

Confluence

As detailed in table 5.5, team Bravo used Confluence. This tool was used for documentation and past decisions, similar to Github pages in team Alpha. Confluence managed historical and requirement dependency (knowledge dependencies), where specific domain knowledge was required for sufficient documentation and available for future decisions.

5.3 Process dependency

The process dependencies which emerged in team Alpha and Bravo made up for 18,7% and 17,6% respectively (see figures 5.2 and 5.4). This dependency addresses process-related practices and how they might affect project progress, which is defined as *"When a task must be completed before another task can proceed and this affects project progress"*. The

three coordination mechanisms that managed both process dependencies are listed in table 5.6 and will be further explained. The following sections elaborate on the remaining dependencies and a presentation of the coordination mechanisms they were managed by.

Strategy component	Coordination mechanism	Total dependencies	Team
Synchronization activity	Sprint	4	Alpha
	One-on-one meetings	4	Both
Boundary spanning artefact	Insight reports	4	Both

Table 5.6: Best matched coordination mechanisms for process dependencies

Sprint

Team Alpha used a more rigid process method, where they iteratively worked in one-week sprints similar to Scrum. The direction and prioritized workload were determined in the weekly planning meeting on Mondays, where predefined propositions of what to focus on were prepared in advance from pre-sprint plannings. Including the management of activity and business process dependencies (process dependencies), the sprint managed both requirement (knowledge dependency) and entity (resource dependency). Team Alpha depended on the predefined propositions from the pre-sprint plannings, which provided domain knowledge prepared by the product owner, team lead, and tech lead.

One-on-one meetings

This coordination mechanism is a practice used by both teams, which functioned as a guidance meeting between the tech lead or team lead and the team's members. As a result, the teams had regular conversations where they could share and discuss matters in confidentiality. This way, possible topics outside the agenda of meetings could be addressed and solved efficiently. The meeting managed expertise and requirement dependency (knowledge dependencies) because the meetings were mostly about clarifications and guidance regarding requirements and specific task- and domain-related knowledge. It also addressed both process dependencies because team members might need this meeting to progress their task, which is also affected by existing processes (OKR).

Insight reports

As a boundary spanning artefact, insight reports were produced by the UX designer. They were helpful for developing the product in both teams and provided insight from relevant persons. It was a valuable mechanism that managed the requirement dependency by retrieving relevant knowledge about the domain, and historical dependency as it can be viewed at a later time as a form of documentation (knowledge dependencies). It further managed both process dependencies. The insight was required for the product to be designed and implemented in alignment with users, customers, and management to continue corresponding work.

5.3.1 Activity dependency

Table 5.7 contains the different coordination mechanisms which most accurately addressed the activity dependency. The teams utilized Jira and Trello to visualize the progress and responsibility of various tasks, which provided an organized way of viewing the workload. Ten coordination mechanisms managed the activity dependency in team Alpha and 9 in team Bravo (see figures 5.1 and 5.3), and is defined as *"An activity cannot proceed until another activity is complete and this affects, or has the potential to affect, project progress"*.

Strategy component	Coordination mechanism	Total dependencies	Team
Synchronization artefact	Jira	3	Alpha
	Trello	3	Bravo
	Tasks	4	Both

Table 5.7: Best matched coordination mechanisms for activity dependency

Jira

For project management, Jira was used by team Alpha to organize tasks and backlog. Functioning as a Kanban board, it provided a visualization of the workload and managed three dependencies. Expertise and task allocation dependencies (knowledge dependencies) were handled by the need for specific knowledge by the team members and distribution of responsibility and tasks. Activity dependency is addressed because many tasks must be completed before another can be initiated, which is visualized and managed with Jira.

Trello

Team Bravo similarly used a project management tool called Trello. It managed dependencies such as expertise and task allocation (knowledge dependencies) through team members' knowledge and the distribution of responsibility and tasks. Activity dependency is addressed because many tasks must be completed before another can be initiated, which is visualized and managed with Trello.

Tasks

Managing expertise, task allocation (knowledge dependencies), and technical dependencies (resource dependency), tasks were distributed based on specific knowledge regarding expertise, domains, and OKRs. In many cases, tasks had to be completed before further work could continue and therefore managed the activity dependency. The technical dependency was also strictly related to tasks and was therefore managed because components often are dependent on each other to interact or function.

5.3.2 Business process dependency

The business process dependency was managed by 13 mechanisms in team Alpha and ten in team Bravo (see figures 5.1 and 5.3). The remaining mechanisms most relevant to explain in this section are presented table 5.8. They managed business process dependencies because the meetings were affected by OKR, existing processes, and additional practices introduced when teams were distributed. A business process dependency is defined as *"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"*.

Strategy component	**Coordination mechanism**	**Total dependencies**	**Team**
Synchronization activity	Pre-sprint planning	3	Alpha
	Backlog grooming	3	Bravo
Proximity	Distributed teams	2	Both

Table 5.8: Best matched coordination mechanisms for business process dependency

Pre-sprint planning

The pre-sprint planning involved the product owner, the team lead, and the tech lead. Arranged every Friday, the purpose of the meeting was to tentatively plan what to prioritize in the upcoming planning meeting in team Alpha. Further, the pre-sprint planning managed expertise and requirements dependencies(knowledge dependencies) and business process dependency. The expertise and requirement dependency were managed by the specific knowledge of the team lead and tech lead and the advice from the product owner. The team lead explained the following:

> "When we are in the pre-planning, we simply go through what we most likely have to work with the next week on a general level. Perhaps we discuss which tasks to include, but the idea is to not make it decisive for what is chosen in the planning. But it is clear when the discussion isn't flowing very well, what we have talked about prior to the meeting can be very conclusive."

The outcome of the meeting was strictly a tentative plan for the upcoming planning meeting. However, it could be somewhat leading because the discussions in the planning meeting often were insufficient or did not flow as well, compared to when held co-located.

Backlog grooming

The backlog grooming served the same function for team Bravo as the pre-sprint planning did for team Alpha. The team members were encouraged to provide their opinions for the backlog grooming if they had any specific thoughts on what key results should be of priority in the upcoming week. The team lead and tech lead would discuss what they should prioritize and create a tentative plan. Following this, the tentative plan

would be discussed in the backlog meeting and the workload determined in plenary.

Distributed teams

Because of the pandemic, the teams worked remotely for the better part of 2020. Working as a distributed team managed business process dependency and entity dependency (resource dependency) because their work process relied on specific digital resources to coordinate properly. The lack of co-location had enabled the teams to reduce unnecessary meetings, and was stated by members from both teams. The team lead of team Bravo explained: *"You can be much more productive during the day because there aren't as many meetings. The threshold to initiate a meeting is probably a little bit higher than physically asking someone in the office"*.

The threshold to initiate meetings had increased, but the amount of digital communication had also increased. Both teams explained how they would reach out on Slack if there were any implications, which differed from when they could physically walk up to someone and ask. Further, members of both teams stated that the threshold to approach others was increased. As a challenge introduced by the distribution of teams, this could result from limited body language and physical interaction, which further influenced meetings relying on discussions.

5.4 Resource dependency

As one of the less frequently observed dependencies, the resource dependency covered 20,3% of dependencies identified in team Alpha and 19,4% in team Bravo (see figures 5.2 and 5.4). The coordination mechanisms which covered both resource dependencies will be explained initially and are listed in table 5.9. In the following sections, the dependencies and the remaining relevant coordination mechanisms will be detailed. Resource dependency is defined as *"When an object is required for a project to progress"*.

Strategy component	Coordination mechanism	Total dependencies	Team
Synchronization artefact	Pull request	4	Both
	Collaboration tools	4	Both
Availability	Constantly online	4	Both

Table 5.9: Best matched coordination mechanisms for resource dependencies

Constantly online

Team Alpha and Bravo both worked distributed, which entailed a substantial amount of communication being performed digitally. As a collaboration tool, Slack was utilized by both teams extensively during work hours. This included meetings, ad hoc communication, social events, and more. The individual threshold for balancing work and answering questions differed in both teams by being constantly online and managed expertise, requirement (knowledge dependencies), technical and entity dependency

(resource dependencies). By locating domain knowledge and sharing one's expertise, being constantly online depended on various tools and people. Team members from both teams explained different thresholds regarding context switching, for example, how much time they would continue working before checking the chat logs. A member of team Bravo explained: *"You can ignore Slack for two hours if you want, but the expectations for your answer is higher compared to a comment on the Kanban board"*.

It is reasonable to think that this expectation could lead to an unnecessary amount of context switching. On the other hand, as ad hoc communication could be inquiries for assistance in many cases, the expectation for an answer was probably aligned with this. As a result, team members could feel obligated to answer within a relative time frame if someone actually needed help for their work to be continued. A member of Alpha stated the following when asked about a change of threshold when asking for help: *"I think it is pretty similar, but there is a change of format which means you have to send a message or engage in a video call. You can't turn around and ask someone"*. It would almost seem as if the threshold to ask for help was experienced as unchanged for some team members, but in reality would be slightly higher. This could be a subconscious challenge for communication in teams when converting from co-location to working remote.

Collaboration tools

Team Alpha and Bravo both utilized an extensive amount of tools to facilitate coordination and collaboration. This included Miro, Slack, Github, Whereby, and more. These have been grouped into one coordination mechanism and addressed four dependencies; expertise, task allocation(knowledge dependencies), entity, and technical (resource dependencies). The various tools were used for meetings and to plan and prioritize workload, which were valuable resources for a distributed team to be productive and coordinate properly.

Pull request

In collaboration with the version-control tool Github, pull requests were used by the teams to submit their work. The pull requests consisted of implemented code according to tasks, and would be reviewed by peers before they ultimately could be applied to the latest version of the product. They covered expertise and requirement dependencies (knowledge dependencies) due to the specific knowledge required to create and review the quality of the pull request. Further, pull requests managed entity and technical dependencies (resource dependencies) as they were dependent on certain team members to be reviewed, and their presence or absence could affect the system.

5.4.1 Entity dependency

Entity dependency was managed by 18 mechanisms in team Alpha and 15 mechanisms in team Bravo (see figures 5.1 and 5.3). Table 5.10 details the various mechanisms which best fit the entity dependency and is defined as *"A resource (person, place or thing) is not*

available and this affects, or has the potential to affect, project progress". They were managed by entity dependency because the absence or presence of individuals in OKR training, team lead meetings, and coordinator roles could affect project progress.

Strategy component	Coordination mechanism	Total dependencies	Team
Boundary spanning activity	OKR training	3	Both
	Team lead meetings	3	Both
Coordinator role	Team lead	4	Both
	Tech lead	3	Both
	UX designer	3	Both
	Data scientist	3	Alpha

Table 5.10: Best matched coordination mechanisms for entity dependency

OKR training

Roughly on a one-year basis, teams across the agency were trained with regards to OKRs. To produce high-quality objectives and key results effectively, the teams participated in a course held by certified OKR facilitators. The amount of time a member had been on team Alpha and Bravo varied, which meant that some junior team members could go a long time without proper training. The OKR training managed expertise and requirement dependencies (knowledge dependencies) and entity dependency (resource dependency). It provided a platform for the teams to increase their understanding and how they could set better OKRs. This could result in improved coordination with regards to the regular meetings of team Alpha and Bravo, which already relied on a collective mindset on OKR concerning their products.

Team lead meetings

The agency had recently experienced a lot of growth, which resulted in a lack of structure for routines and processes. The distribution of responsibility was somewhat unclear, which also caused some obstacles. To manage this, all team leads held a weekly coordinating meeting. The meeting managed historical dependency (knowledge dependency), business process dependency (process dependency), and entity dependency (resource dependency), which provided the opportunity for the team leads to align. This was done by sharing their experiences with OKRs, meetings, and workshops, which provided each team lead with the knowledge to improve their individual teams and avoid problems experienced by other teams.

Team lead

The team lead's responsibility consisted primarily of administrative and coordination tasks to enable the team to perform their work in alignment with the product owner and other stakeholders. As a result, the team lead ensured that the team moved in the right direction, communicating company goals and facilitating a good flow of information. Team lead of team Bravo also functioned as a product owner, as their product area did not

have a product owner at the time. Therefore, they had more flexibility to make decisions on a team level and were more autonomous than Alpha and other teams with product owners. The team lead managed expertise, requirement, task allocation dependencies (knowledge dependencies), and entity dependency(resource dependency).

Tech lead

The tech lead performed a lot of coordination work and participated in many meetings but also aimed to be a developer. The responsibility consisted of technical guidance for others, the planning of further development of the platform, and actual development. The role as tech lead managed expertise, requirement (knowledge dependencies), and entity dependencies, by locating and acquiring valuable knowledge which ultimately could help the team improve their product.

UX designer

Including assistance in illustrations and various designs for the solutions in the platform, designers worked with the collection of data and insight from other teams and agencies. It was done through coordination and exploring other product areas and performing interviews, and writing reports of the findings. This was important for the teams, as their product is aimed towards other agencies and the facilitating of their needs for data (team Alpha), and providing services for the citizens (team Bravo). The UX designers managed expertise, requirement (knowledge dependencies), and entity dependencies, as they provided valuable information necessary for the direction of the products.

Data scientist

Team Alpha also had a data scientist, which has been defined as a coordinator role. As opposed to the developers who primarily worked with building the platform, the data scientist focused on coordinating and dealing with other agencies, which ultimately was the platform's user. Their work consisted of indicating different requirements, understanding the user cases, and creating proof of concepts that allowed the developers to create optimal solutions fitting the needs of their users. The data scientist managed dependencies such as expertise, requirement (knowledge dependencies), and entity dependencies by coordinating with other teams and agencies, providing valuable information for identifying additional needs and requirements for the product.

5.4.2 Technical dependency

Seven mechanisms managed the technical dependency in team Alpha and six mechanisms in team Bravo (see figures 5.1 and 5.3). The two mechanisms listed in table 5.11 managed the technical dependency, as they provided valuable information, and their absence could affect project progress. The technical dependency is defined as *"A technical aspect of development affects progress, such as when one software component must interact with another software component and its presence or absence affects, or has the potential to affect, pro-*

ject progress".

Strategy component	Coordination mechanism	Total dependencies	Team
Boundary spanning artefact	KPI	2	Alpha
	OKR tracker	4	Both

Table 5.11: Best matched coordination mechanisms for technical dependency

KPI

Key performance indicators (KPI) were defined on a general level in the agency and used in teams which the measure fits. The KPIs were defined to measure volume, satisfaction, and conversion rate. These values did not fit the product of all teams in the agency, including team Bravo. They were utilized in Team Alpha alongside their OKRs, allowing them to view their progress compared to other teams. Moreover, the KPIs were expected to be in use for all teams eventually but have not yet been prioritized. KPI managed business process dependencies (process dependencies) and technical dependencies, because it provided metrics that potentially could affect their work process and prioritization.

OKR tracker

The OKR tracker was a digital dashboard available for all in the agency, which provided transparency and inter-team insight. It managed requirement and task allocation dependencies (knowledge dependencies) because it required domain-specific knowledge to set up for the individual teams and contributed to an overview of different teams' workload and performance. The OKR tracker also managed the business process dependency as it showed the progression of different key results, which was also relevant for the team to further plan their workload. If the metrics shown in the tracker were incorrect, it could negatively affect the work of teams, which in turn managed the technical dependency.

Discussion

In Chapter 5, the results of this multiple-case study were presented. The particular context of team Alpha and team Bravo was assessed. The findings showed 123 dependencies managed by 37 coordination mechanisms in team Alpha and a total of 108 dependencies managed by 34 mechanisms in team Bravo. This chapter will present, discuss, and compare the findings with existing studies in order to answer the research question:

"How are dependencies managed in distributed teams in large-scale agile?"

The findings will further be discussed in relation to large-scale agile and implications for theory and practice.

6.1 Dependencies and coordination mechanisms

This section will present the most significant coordination mechanisms used to manage dependencies in the two teams. The findings related to knowledge, process, resource dependencies will be discussed and compared with other studies. As presented in Chapter 5, the coordination mechanisms will also be categorized with the best-matched dependencies in the following sections.

6.1.1 Knowledge dependency

Totaling over 60% of dependencies in the two teams, the knowledge dependency constituted 75 coordination mechanisms in team Alpha and 68 mechanisms in team Bravo (see tables 5.1 and 5.3). The large number of knowledge dependencies in both teams indicates that team Alpha and Bravo rely heavily on the availability and sharing of knowledge. Many of the coordination mechanisms found in this study manage multiple dependencies and are therefore necessary to further examine as they have the most significant influence on coordination in large-scale agile. Accordingly, this section will discuss ad hoc communication, product owner, daily standup, written slackup, OKR workshop, and retrospective actions.

Ad hoc communication

One of the prominent contributors to the management of dependencies was ad hoc communication, which addressed knowledge dependencies such as expertise, requirement, and task allocation. Further, ad hoc communication also managed activity

dependency (process dependency) and entity dependency (resource dependency). The ad hoc communication constituted the majority of coordination in the teams, next to the regular meetings. This is especially inherent in team Bravo, as they removed the daily standup meeting entirely. Further, unplanned discussions were often not prioritized in regular meetings because of the predefined agenda and limited time. Emphasized by team members from both teams, reaching out on Slack was their first choice to locate expertise and facilitate discussion.

These findings are consistent with the findings of a large-scale agile case study by Stray (2018). The study discovered how the project members spent on average 1.1 hours a day on scheduled meetings, compared to 1.6 hours a day on unscheduled meetings and ad hoc communication. Moreover, a mixed-methods study by Stray and Moe (2020) surveyed members of globally distributed teams, also uncovering how teams spent more time in unscheduled meetings compared to scheduled ones.

Product owner

The findings show that the product owner was an important coordinator role related to both knowledge, process, and resource dependencies. Previous research has found that the product owner role is important for large-scale coordination both for coordination within the team, but also between teams, in large-scale agile (Berntzen, Moe and Stray, 2019). In the case of team Alpha, the product owner worked in close collaboration with the team lead, thus managing process-related dependencies and managing technical dependencies through coordination with the tech lead. This finding is in line with Berntzen, Moe and Stray (2019), who found coordination with the team lead was an important product owner activity.

Further, Bass (2015) found that the product owner performs a wide set of different functions. Remta, Doležel and Buchalcevová (2020) studied the product owner in a company that implemented the Scaled Agile Framework and found that this role entails being a gatekeeper, motivating, communicating, and prioritizing. This is consistent with the findings of this study because the product owner role contributes to addressing five types of dependencies and performs a wide set of responsibilities.

Daily standup

The daily standup meeting was used by team Alpha to have a short and daily status update, where all team members participate and share what they have done, are doing, and possible obstacles. Stray et al. (2016) explained in their study how it is imperative to have the meeting before lunch to make it less disruptive and to be standing up to keep it short. Team Alpha performed their standups similar to the study, where it was followed by lunch. Meanwhile, the team was working remotely at the moment of observations, and therefore the team members were sitting down during the virtual meeting. Further, the tech lead of team Alpha explained how it would be optimal if there were increased focus on what will be done instead of what has already been done. This is in line with Stray et al. (2016), which mentioned how it is important for daily standups to focus on

the future.

Written slackup

Rather than having a daily standup, Bravo used a written format called slackup in the distributed setting. With the same goal as a regular standup of giving a brief update on current work, the written slackup provides additional flexibility to share the update and is less disruptive. This is also in line with Stray, Moe and Sjoberg (2018), where the findings showed how it was important to make the meeting less disruptive. The written format resulted in less context-switching, which provided team Bravo with more room to focus on complex work. Additionally, a mixed-methods study of globally distributed teams by Stray and Moe (2020) explained how there is more focus and efficient decision making in unscheduled meetings, making unplanned coordination valuable in distributed teams. Accordingly, this can be beneficial for team Bravo as additional meetings can be initiated when necessary.

Retrospective actions

The retrospective meeting held in Team Alpha and Bravo was focused on process-specific and product-specific issues, which could result in actions with a broader scope of topics. The responsibility to follow up actions after each retrospective was appropriately distributed among team members. Regardless, the actions were not always fully prioritized and followed up. Team members of both teams emphasized how these actions often could end up forgotten or partly done, much because the retrospective actions ended up being additional tasks next to the regular work. Retrospectives provided a regular reflection for the team to adjust and become more effective, which is in line with one of the core principles of agile. Moreover, there is minimal research about retrospective actions as coordination mechanisms. The findings of this study indicated that OKR perhaps required too much attention from the team members, leaving little room to follow up on retrospective actions and self-improvement for the teams.

OKR workshop

As a regular meeting arranged once a quarter, the OKR workshop was used by both teams in this study. Aimed to set a direction for the teams in the upcoming quarter, the OKR workshop addressed five dependencies. It was identified as one of the coordination mechanisms managing the most dependencies and important for the teams and their work in the following period. Several members of both teams explained what they perceived as difficult with OKR as a framework, being the creation of objectives and key results. Considering the agenda of the OKR workshop, these difficulties are highly correlated with the meeting itself.

Strode (2016) explains in her study how poorly managed dependencies could be the result of inappropriate, inadequate, or absent coordination mechanisms. Considering the number of dependencies managed by the OKR workshop and the recurrent difficulties explained by both teams, it might be appropriate to evaluate if the meeting manages

its dependencies sufficiently. As a result, other coordination mechanisms could also be negatively affected. An example of this could be retrospective actions. Despite their efforts, both teams could not find the time or priority to follow up on the actions after each retrospective meeting because the OKRs often required their attention.

6.1.2 Process dependency

The process dependencies identified in this multiple-case study make up 18,7% in team Alpha and 17,6% in Bravo (see figures 5.2 and 5.4). This dependency addresses activity and process-related practices and how they might affect project progress. An example would be dependencies between tasks or how an existing business process influences how teams work. Process dependencies are often a part of everyday work and routines, which could often be less visible or go by unnoticed. Therefore, it is important to examine further how they might have influenced team Alpha and Bravo. Sprint and distributed teams managed multiple dependencies, and detailed important team characteristics such as process rigidity, and how working remotely had influenced their coordination. Possible findings with regards to this will be discussed below.

Sprint

Team Alpha used a sprint-based process for their development work, confined to one-week iterations. A sprint planning meeting would be arranged each Monday to discuss the prioritized workload and their ambitions for the sprint. In addition, a daily standup was held each day to briefly inform about what each team member is working with. Cooper and Sommer (2018) presented findings from six different case studies from agile development. Their study showed how each sprint was initiated with a sprint planning meeting, followed by a daily standup to ensure the work is on course and ended with a demo and a retrospective. The findings from team Alpha are consistent with this, although demos and retrospectives were not arranged each week.

In another context, team Bravo did not use sprints but a more flexible approach similar to the principles of Kanban. Accordingly, their work could continue over multiple weeks, and the backlog was reviewed each Monday to discuss the work in progress. Team Bravo and their choice of converting the daily standup to a written slackup also reduced disruptions, which is in line with the principle of reducing waste, central in Kanban and lean processes.

As explained above, the development processes of team Alpha and team Bravo are different with regards to rigidity. As neither of the teams fully followed a certain process method, it is difficult to compare their approach with existing research on team-level, and if it affected their productivity. Team Alpha, with its 11 team members were quite larger than team Bravo and their total of seven team members, which also should be taken into consideration.

Distributed teams

Because of the global pandemic, team Alpha and Bravo were working full-time distributed. As a result, all scheduled and unscheduled coordination was performed digitally. The findings show that the number of meetings was reduced, and the threshold to initiate ad hoc communication had increased. Regardless of the increased threshold, the amount of digital coordination had still increased compared to when the teams were co-located. Team Alpha and Bravo have worked remotely for the past year and have primarily utilized the communication tool Slack for coordination. Members of team Bravo stated that ad hoc chatting often could lead to disorganized communication and chat logs.

This is in accordance with Espinosa, Slaughter, Kraut and Herbsleb (2007), where findings show that coordination challenges are more evident in distributed teams compared to co-located teams with regards to communication. It was also difficult to find a balance between chatting and initiating meetings for discussions, where chats and threads often ended up too long. This is consistent with the findings of Stray and Moe (2020), which showed that a lack of guidelines when using Slack resulted in coordination being confusing and frustrating for some team members.

Further, the findings showed unbalanced activity in meetings, both in team Alpha and Bravo. Members of both teams explained that meetings requiring discussion were affected after the teams started working remotely. The virtual meetings made it difficult to read body language, and network issues could often lead to participants talking simultaneously and canceling each other's voice out. Accordingly, the retrospectives, as well as the planning meeting for team Alpha, and the backlog meeting for team Bravo, were discussed less in meetings. This is in line with the findings of Stray and Moe (2020), where results showed that the retrospectives arranged co-located lasted longer because the participants were more invested in the discussions compared to the virtual meetings. The interviewees of the study further stated that the reasons behind this were that they could use whiteboards, stickers and have informal co-located discussions. The virtual meetings did not provide this in the same way.

6.1.3 Resource dependency

The resource dependency covered 20,3% of dependencies identified in team Alpha and 19,4% in team Bravo (see figures 5.2 and 5.4). This dependency addresses resource-related practices and how they might affect project progress, such as people, tools, technicalities, and other resources. As team Alpha and Bravo were working distributed, they relied more on the presence of people and collaboration tools in order to coordinate properly. Accordingly, the findings with regards to team lead, data scientist, and collaboration tools will be discussed.

Team lead

Categorized as a coordinator role, the team lead managed a total of four dependencies in both teams. The responsibility of a team lead includes coordinating with the team and the work of the individual members, communicating with the product owner, reducing noise, and aligning the team according to the agency's vision. Further, the team lead of Bravo also functioned as a product owner at the time of the interviews. As a result, team Bravo was very autonomous, and much of the decisions for their product were taken at the team-level.

For both cases, this is consistent with the findings of Moe, Dahl, Stray, Karlsen and Schjødt-Osmo (2019). Their study showed how one team member was responsible for communicating with the project and organization to reduce the challenge of having to deal with too many additional tasks. This included protecting the team from unnecessary interruptions and deciding what information was relevant to bring forward to the team. The importance of shielding the team from external dependencies was emphasized, which is true for both team Alpha and Bravo.

Data scientist

Team Alpha is a team with roles of various scope and responsibilities. One of these was the data scientist, who is in charge of indicating different requirements, understanding use cases, and creating proof of concepts. This work included much inter-team coordination and allowed the developers to create optimal solutions fitting the needs of their users.

A study by Hukkelberg and Berntzen (2019) explored the challenges of integrating data science roles in agile teams. Their findings emphasized the importance of a data-driven platform that aims to make data-driven decisions to fully utilize data scientists' competence in autonomous teams. This is in line with team Alphas's product and vision to facilitate easy access and sharing of data.

Collaboration tools

Considering the current situation where team Alpha and team Bravo were working remotely, it was evident that collaboration tools played an important role. These were tools that facilitated project management, communication, video conferences, bug-tracking, version control, and more. Factors involving communication were not considered as necessary when the teams were co-located because much of this collaboration could be performed physically. The findings of this multiple-case study show how team Alpha and team Bravo utilized a comprehensive set of tools to support and coordinate their work in a distributed setting.

Stray and Moe (2020) discovered how Slack supports agile teams, with more focus and efficient decision making in unscheduled meetings, making tools and unplanned coordination valuable in distributed teams. These findings are consistent with the findings in team Alpha and team Bravo, as they used Slack for most of their

communication. The findings of Espinosa, Slaughter, Kraut and Herbsleb (2007)) also indicated that large-scale distributed software development organizations could substantially benefit from promoting the use of tools and practices that strengthen different types of team knowledge. This is in line with the findings of ad hoc communication and the extensive amount of digital communication in team Alpha and Bravo.

6.2 Dependency management in large-scale agile

Large-scale projects are defined as projects with two to nine teams, and very large-scale projects contain ten or more. As a result, the need for new or adjusted agile practices is introduced (Dingsøyr, Moe, Fægri and Seim, 2018). Findings by Berntzen, Stray and Moe (2021, in press) showed how coordination mechanisms were useful to manage dependencies across teams in a large-scale program and how the organization under study responded to coordination problems that emerged when scaling. They emphasized how large-scale agile programs could benefit from adopting coordination mechanisms to their specific organizational contexts and needs to better cope with uncertainty, novelty, and complexity. A different study by Dingsøyr, Moe and Seim (2018) explored coordination in a large-scale program, bringing attention to the importance of scheduled and unscheduled meetings and the need to change coordination mechanisms over time.

This is in line with the findings of this study, as both team Alpha and Bravo have changed meeting formats, structure, and tool usage with regards to their specific needs. The findings showed that the knowledge dependencies constituted a substantial amount of the total dependencies uncovered in team Alpha and Bravo. The coordination mechanisms managing these dependencies were mostly characterized by strategy components such as synchronization- activities and artefacts and coordinator roles. They proved to be very central in the coordination performed in team Alpha and Bravo and were continually evaluated for potential improvements. Replacing daily standup with written slackup and Jira with Trello (team Bravo), restructuring planning, and replacing google slides with Miro (team Alpha), were among the changes done by the teams to improve their situations. The findings of Dikert, Paasivaara and Lassenius (2016) are consistent with these choices, where they discovered how choosing and customizing the agile approach was the number one contributor to success in large-scale agile transformations.

Berntzen, Stray and Moe (2021, in press) further explained how challenges with maintaining an overview of work were typically introduced with knowledge dependencies as teams grew in size and number. They recommended mechanisms to showcase who works with what in different teams and utilize communication tools to synchronize and align the teams. Team Alpha and Bravo used similar coordination mechanisms to manage process and resource dependencies through collaboration tools, the OKR tracker, various intra- and inter-team meetings, and ad hoc communication, indicating proper management of inter-team dependencies in this large-scale agile context.

On the other hand, their work and prioritization are heavily influenced by OKR, which is why it is crucial for team Alpha and Bravo to continually evaluate the need for change or new coordination mechanisms. The findings show that OKR was perceived as difficult by the teams and indicated potential complications to their coordination. One explanation could be that OKR requires the teams to tailor the framework to their individual contexts or introduce additional mechanisms to manage the dependencies. It is hard to determine what would result in the optimal outcomes for Alpha and Bravo, but continued focus on OKR and proper management of dependencies could benefit both teams.

6.3 Implications for theory

6.3.1 Dependency taxonomy

The dependency taxonomy by Strode (2016) was used in this study to identify dependencies and agile practices in two large-scale distributed teams. This multiple-case study used interviews and observation of team members from two similar contexts in an agency. The results presented by Strode (2016) are based on interviews of team members from three different project cases, varying in complexity and customer involvement. With regards to the results, the difference in type and scale is important to keep in mind. Strode (2016) further proposed the applicability of the taxonomy on large-scale or distributed contexts, which both are consistent with this study. Using the taxonomy allows for the mapping of coordination mechanisms with their best-matched dependencies. As a result, it was possible to collect the most appropriate agile and non-agile practices and identify which were inappropriate, inadequate, or absent in the case contexts.

Considering the framework by Strode was originally designed for and applied to software projects, the applicability to large-scale agile would potentially lead to a difference in the identified coordination mechanisms and dependency management. For example, her study defined the boundary spanning to be activities and artefacts beyond project level. As a large-scale and distributed context was chosen for this multiple-case study, it is important to examine the inter-team dependencies. Accordingly, I redefined boundary spanning to apply beyond the team level to better address the inter-team context. Both team Alpha and Bravo had inter-team dependencies and have been presented to some extent through boundary spanning activities and artefacts. However, any further was considered beyond the scope of this study.

Other studies have also applied the framework of Strode. For example, Berntzen, Stray and Moe (2021, in press) employed the framework in a large-scale setting which also used OKR, adapting the framework to the inter-team level in a study with 16 development teams. Their study discovered coordination strategies that reflected the complex environment that large-scale agile often is characterized by. Their findings were also consistent with this study, which uncovered both agile and non-agile coordination practices. On the other side, Strode (2016) only included agile practices as coordination

mechanisms in her study. Stray, Moe and Aasheim (2019) similarly conducted a qualitative case study of coordination in autonomous DevOps teams, using the proposed framework by Strode (2016) to map agile practices and dependencies. The findings of my study extensively showed how non-agile practices were important as well. Practices such as communities of practice, OKR, ad hoc communication, and various tools and meetings were all significant contributors to coordination in large-scale agile and managed multiple dependencies.

6.3.2 Limitations

The data collected and the results presented for this multiple-case study were confined to two teams and their contexts. Considering *"coordination mechanisms are dynamic social practices that are under continuous construction"* (Jarzabkowski, Lê and Feldman, 2012, p. 907), the results could be different if the data were collected at a different point in time. Regarding this, it could have been interesting to observe the teams over a more extended period of time. Had a different case been chosen with a different context, the results could also have been different. The observations and interviews were arranged remotely, potentially excluding important body language and other valuable interactions in meetings. Considering the teams were working distributed, this limitation would be hard to avoid. Lastly, the results of this multiple-case study are influenced by my interpretation of the dependency taxonomy and how the various dependencies were addressed by the coordination mechanisms identified. A different person might find that the coordination mechanisms addressed different dependencies, which would yield different results.

6.3.3 Validity

The taxonomy proposed by Strode (2016) was used to identify coordination mechanisms and dependencies. By using a theory, the external validity increases accordingly. The observations of team Alpha also allowed me to clarify any questions I had, which allowed me to get a deeper understanding of their work and reduce the threat of construct validity. The teams were also contacted for clarifications and questions, which further improved the validity of the collected data.

6.4 Implications for practice

The findings of this study showed several implications for practice. Most important of all, teams and organizations should make sure to prioritize coordination mechanisms that address multiple dependencies. This is in order to manage the dependencies properly and avoid potential difficulties with regards to coordination on an intra- and inter-team level. As organizations progress into large-scale and very large-scale agile, breakdowns and failures with regards to coordination are more common. Accordingly, I suggest that

organizations investigate the coordination challenges and dependencies experienced by teams to better understand the coordination mechanisms used and to uncover potential inadequacies.

Moreover, as many large-scale agile teams currently work distributed, all communication is performed digitally. The findings of this study show that transitioning from co-location to working remotely quickly could result in increased ad hoc communication, disorganized chats, and lack of structure when using Slack. With proper routines, chat logs would be better structured and information easier to locate. Further, the expectation to answer within a reasonable timeframe would often urge team members to check Slack. By using channels and tags for cases of different priority and relevance, team members could reduce their context switching and potentially only be disrupted by inquiries that need urgent attention. Such routines could also encourage more unscheduled meetings instead of long threads of discussions. According to Stray and Moe (2020), this yields increased focus and efficient decision-making, which could positively influence coordination in teams.

As observed in team Bravo, replacing the daily standup with a written slackup could help distributed teams in large-scale agile to reduce disruptions further. It is relatively easy to implement and could assist teams to focus on complex work over longer periods of time. This could be especially valuable in large teams with many developers, as their work often requires longer spans of focus to work efficiently.

Conclusion and future work

This multiple-case study aimed to investigate how coordination mechanisms facilitate large-scale agile development by examining how they were used to manage dependencies. The aim was to address the research question: *How are dependencies managed in distributed teams in large-scale agile?*

The findings show that dependencies in two teams in a large-scale setting were managed by a broad set of different agile and non-agile coordination mechanisms. The contexts of these two teams were examined, consisting of 6 departments and 11 product teams. The two teams are referred to as team Alpha and Bravo. The data collection consisted of observing 21 meetings in team Alpha and interviews with team members from both team Alpha and Bravo.

A dependency taxonomy proposed by Strode (2016) was used to map the broad scope of coordination mechanisms. This taxonomy proved to be a valuable tool to categorize what mechanisms were used by team Alpha and Bravo and which dependencies were managed by these. A total of 123 coordination mechanisms in team Alpha and 108 in team Bravo were categorized in an organized manner, visualizing the foundation for how the teams worked and coordinated within and across teams. This included coordination mechanisms such as meetings, tasks, collaborative tools, roles, and their influence on the work and process of both teams. The majority of the coordination mechanisms were multi-purpose, addressing more than a singular dependency.

Every coordination mechanism that emerged from team Alpha and team Bravo was presented in Chapter 5 with their best-matched dependency and carefully examined. This illuminated how the different coordination mechanisms managed their dependencies. The findings of this study suggest that by discussing Objectives and Key Results, the teams managed dependencies both within the teams and inter-team dependencies. Further, ad hoc communication happened mostly on Slack and allowed team members to locate and discuss expertise, requirements, task allocation, and activities. This was facilitated by the extensive use of collaboration tools, which also proved to be very valuable in a distributed context with no co-location.

Roles such as the product owner in team Alpha and the team lead of both teams were important roles that managed knowledge, process, and resource dependencies. The team lead of team Alpha and Bravo were essential coordinator roles, successfully guiding and shielding their teams from any unnecessary external noise. Their cases and differences have been detailed, where process rigidity and meetings have been discussed in Chapter 6. Despite different contexts, both teams shared many of the same coordination mechanisms and managed the corresponding dependencies similarly.

Accordingly, both teams experienced many of the same difficulties with OKR. There were indications that certain dependencies were not adequately managed, which was discussed concerning retrospective actions in section 6.1.1. Existing mechanisms might not have managed the dependencies sufficiently, or the teams might need to evaluate the need for additional coordination mechanisms.

Future work should further explore how OKRs can be used to align teams in large-scale agile projects. In this large-scale project, no dedicated roles were focusing on OKRs in the teams. As a result, discussions were very time-consuming during meetings and required much facilitation. The findings of this study indicate that large-scale projects utilizing OKR would benefit from having a dedicated "OKR master" to facilitate and follow up on the process. It would be interesting to investigate whether such an informal role could contribute to better coordination by having one person coordinating OKR-related aspects within the team and other teams, if OKR is present on an inter-team level. Further, applying the framework of Strode (2016) to an inter-team context in large-scale agile would also be interesting to get a better understanding of inter-team dependencies and coordination. This could benefit software development organizations by improving coordination and provide valuable insight to better handle the growth of agile teams and projects.

Bibliography

Awad, MA (2005). 'A comparison between agile and traditional software development methodologies'. In: *University of Western Australia* 30.

Bass, Julian M (2015). 'How product owner teams scale agile methods to large distributed enterprises'. In: *Empirical Software Engineering* 20.6, pp. 1525–1557.

— (2019). 'Future trends in agile at scale: a summary of the 7 th international workshop on large-scale agile development'. In: *International Conference on Agile Software Development*. Springer, pp. 75–80.

Bass, Julian M and Abdallah Salameh (2020). '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.

Bell, Judith (2010). *Doing Your Research Project: A guide for first-time researchers*. McGraw-Hill Education (UK).

Berntzen, Marthe, Nils Brede Moe and Viktoria Stray (2019). 'The product owner in large-scale agile: an empirical study through the lens of relational coordination theory'. In: *International Conference on Agile Software Development*. Springer, pp. 121–136.

Berntzen, Marthe, Viktoria Stray and Nils Brede Moe (2021, in press). 'Coordination strategies: managing inter-team coordination challenges in large-scale agile'. In: *International Conference on Agile Software Development*. Springer.

Berntzen, Marthe and Sut I Wong (2021). 'Autonomous but Interdependent: The Roles of Initiated and Received Task Interdependence in Distributed Team Coordination'. In: *International Journal of Electronic Commerce* 25.1, pp. 7–28.

Cooper, Robert G and Anita Friis Sommer (2018). 'Agile–Stage-Gate for Manufacturers: Changing the Way New Products Are Developed Integrating Agile project management methods into a Stage-Gate system offers both opportunities and challenges.' In: *Research-Technology Management* 61.2, pp. 17–26.

Crescentini, Alberto and Giuditta Mainardi (2009). 'Qualitative research articles: guidelines, suggestions and needs'. In: *Journal of workplace learning*.

Crowston, Kevin and Charles S Osborn (1998). 'A coordination theory approach to process description and redesign'. In:

Curtis, Bill, Herb Krasner and Neil Iscoe (1988). 'A field study of the software design process for large systems'. In: *Communications of the ACM* 31.11, pp. 1268–1287.

Dikert, Kim, Maria Paasivaara and Casper Lassenius (2016). 'Challenges and success factors for large-scale agile transformations: A systematic literature review'. In: *Journal of Systems and Software* 119, pp. 87–108.

Dingsøyr, Torgeir, Tor Erlend Fægri and Juha Itkonen (2014). 'What is large in large-scale? A taxonomy of scale for agile software development'. In: *International Conference on Product-Focused Software Process Improvement*. Springer, pp. 273–276.

Dingsøyr, Torgeir, Nils Brede Moe and Eva Amdahl Seim (2018). 'Coordinating knowledge work in multiteam programs: findings from a large-scale agile development program'. In: *Project Management Journal* 49.6, pp. 64–77.

Dingsøyr, Torgeir et al. (2018). 'Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation'. In: *Empirical Software Engineering* 23.1, pp. 490–520.

Dyba, Tore and Torgeir Dingsoyr (2009). 'What do we know about agile software development?' In: *IEEE software* 26.5, pp. 6–9.

Dybå, Tore and Torgeir Dingsøyr (2015). 'Agile project management: From self-managing teams to large-scale development'. In: *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. Vol. 2. IEEE, pp. 945–946.

Espinosa, J Alberto et al. (2007). 'Team knowledge and coordination in geographically distributed software development'. In: *Journal of management information systems* 24.1, pp. 135–169.

Fink, Lior and Seev Neumann (2007). 'Gaining agility through IT personnel capabilities: The mediating role of IT infrastructure capabilities'. In: *Journal of the Association for Information Systems* 8.8, p. 25.

Floricel, Serghei, John L Michela and Sorin Piperca (2016). 'Complexity, uncertainty-reduction strategies, and project performance'. In: *International Journal of Project Management* 34.7, pp. 1360–1383.

Gerring, John (2006). *Case study research: Principles and practices*. Cambridge university press.

Gren, Lucas, Richard Torkar and Robert Feldt (2017). 'Group development and group maturity when building agile teams: A qualitative and quantitative investigation at eight large companies'. In: *Journal of Systems and Software* 124, pp. 104–119.

Herbsleb, James D (2007). 'Global software engineering: The future of socio-technical coordination'. In: *Future of Software Engineering (FOSE'07)*. IEEE, pp. 188–198.

Hukkelberg, Ivar and Marthe Berntzen (2019). 'Exploring the challenges of integrating data science roles in agile autonomous teams'. In: *International Conference on Agile Software Development*. Springer, pp. 37–45.

Jarzabkowski, Paula A, Jane K Lê and Martha S Feldman (2012). 'Toward a theory of coordinating: Creating coordinating mechanisms in practice'. In: *Organization Science* 23.4, pp. 907–927.

Kirk, Diana and Stephen G MacDonell (2015). 'Progress report on a proposed theory for software development'. In: *2015 10th International Joint Conference on Software Technologies (ICSOFT)*. Vol. 2. IEEE, pp. 1–7.

Kraut, Robert E and Lynn A Streeter (1995). 'Coordination in software development'. In: *Communications of the ACM* 38.3, pp. 69–82.

Larman, Craig and Bas Vodde (2008). *Scaling lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum*. The Agile Software Development Series. Pearson Education Inc. ISBN: 0-321-48096-1.

Li, Jingyue, Nils B Moe and Tore Dybå (2010). 'Transition from a plan-driven process to scrum: a longitudinal case study on software quality'. In: *Proceedings of the 2010 ACM-IEEE international symposium on empirical software engineering and measurement*, pp. 1–10.

Livermore, Jeffrey A (2008). 'Factors that Significantly Impact the Implementation of an Agile Software Development Methodology.' In: *JSW* 3.4, pp. 31–36.

Lundene, Kjell and Parastoo Mohagheghi (2018). 'How autonomy emerges as agile cross-functional teams mature'. In: *Proceedings of the 19th International Conference on Agile Software Development: Companion*, pp. 1–5.

Malone, Thomas W and Kevin Crowston (1994). 'The interdisciplinary study of coordination'. In: *ACM Computing Surveys (CSUR)* 26.1, pp. 87–119.

Mintzberg, Henry (1980). 'Structure in 5's: A Synthesis of the Research on Organization Design'. In: *Management science* 26.3, pp. 322–341.

Moe, Nils Brede, Aybüke Aurum and Tore Dybå (2012). 'Challenges of shared decision-making: A multiple case study of agile software development'. In: *Information and Software Technology* 54.8, pp. 853–865.

Moe, Nils Brede, Torgeir Dingsøyr and Tore Dybå (2008). 'Understanding self-organizing teams in agile software development'. In: *19th australian conference on software engineering (aswec 2008)*. IEEE, pp. 76–85.

Moe, Nils Brede et al. (2019). 'Team autonomy in large-scale agile'. In: *Proceedings of the Annual Hawaii International Conference on System Sciences (HICSS)*. AIS Electronic Library, pp. 6997–7006.

Munassar, Nabil Mohammed Ali and A Govardhan (2010). 'A comparison between five models of software engineering'. In: *International Journal of Computer Science Issues (IJCSI)* 7.5, p. 94.

Niven, Paul R and Ben Lamorte (2016). *Objectives and key results: Driving focus, alignment, and engagement with OKRs*. John Wiley & Sons.

Nyrud, Helga and Viktoria Stray (2017). 'Inter-team coordination mechanisms in large-scale agile'. In: *Proceedings of the XP2017 Scientific Workshops*, pp. 1–6.

O'Leary, Zina (2017). *The essential guide to doing your research project*. Sage.

Remta, Daniel, Michal Doležel and Alena Buchalcevová (2020). 'Exploring the Product Owner Role Within SAFe Implementation in a Multinational Enterprise'. In: *XP2021 Companion Proceedings*. Ed. by Maria Paasivaara and Philippe Kruchten. Cham: Springer, pp. 92–100. ISBN: 978-3-030-58858-8.

Royce, Dr Winston W (1970). 'MANAGING THE DEVELOPMENT OF LARGE SOFTWARE SYSTEMS'. en. In: p. 11. URL: https://leadinganswers.typepad.com/leading_answers/files/original_waterfall_paper_winston_royce.pdf.

Stray, Viktoria (2018). 'Planned and unplanned meetings in large-scale projects'. In: *Proceedings of the 19th International Conference on Agile Software Development: Companion*, pp. 1–5.

Stray, Viktoria and Nils Brede Moe (2020). 'Understanding coordination in global software engineering: A mixed-methods study on the use of meetings and Slack'. In: *Journal of Systems and Software* 170, p. 110717.

Stray, Viktoria, Nils Brede Moe and Andreas Aasheim (2019). 'Dependency management in large-scale agile: a case study of DevOps teams'. In: *Proceeding of the 52nd Hawaii International Conference on System Sciences (HICSS 2019)*. University of Hawai'i.

Stray, Viktoria, Nils Brede Moe and Rashina Hoda (2018). 'Autonomous agile teams: challenges and future directions for research'. In: *Proceedings of the 19th International Conference on Agile Software Development: Companion*, pp. 1–5.

Stray, Viktoria, Nils Brede Moe and Dag IK Sjoberg (2018). 'Daily stand-up meetings: start breaking the rules'. In: *IEEE Software* 37.3, pp. 70–77.

Strode, Diane E (2016). 'A dependency taxonomy for agile software development projects'. In: *Information Systems Frontiers* 18.1, pp. 23–46.

Strode, Diane E et al. (2012). 'Coordination in co-located agile software development projects'. In: *Journal of Systems and Software* 85.6, pp. 1222–1238.

Van de Ven, Andrew H, Andre L Delbecq and Richard Koenig Jr (1976). 'Determinants of coordination modes within organizations'. In: *American sociological review*, pp. 322–338.

Wheelan, Susan A and Judith M Hochberger (1996). 'Validation studies of the group development questionnaire'. In: *Small group research* 27.1, pp. 143–170.

Wheelan, Susan A et al. (1998). 'Member perceptions of internal group dynamics and productivity'. In: *Small Group Research* 29.3, pp. 371–393.

Wohlin, Claes, Darja Šmite and Nils Brede Moe (2015). 'A general theory of software engineering: Balancing human, social and organizational capitals'. In: *Journal of Systems and Software* 109, pp. 229–242.

Yin, Robert (Sept. 2017). *Case Study Research and Applications: Design and Methods*. 6th ed. Thousand Oaks, California: SAGE Publications. 414 pp. ISBN: 978-1-5063-3616-9.

Interview guide

Introduction

- Present myself and the project
- Ask for consent to record

General

- What is your current position?
- How long have you been working here?
- What are your main responsibilities?

Team

- When was the project started?
- Have anything changed since you started on the team (technology, positions, framework)?
  - Have anything improved or become worse?
- How do you mostly work as a team, individually or with each other?
- What would you say is the goal of your product?

Coordination

- How do you coordinate within and across teams?
- Which platforms do you use for communication?
  - How do you think it works?
- Do you feel the coordination in the team is influenced by the lack of co-location?
- How has working remotely impacted you and your work routines?
- If you are stuck on a task, how do you proceed to solve it?
- Do you think your threshold to inquire assistance has changed after the team started working from home?

- Have you noticed a change of inquiries from others?
  - In what way?
- Do you experience any interruptions or context switching during work?
  - Do you think it affects your work in any way?
- How would you describe the information flow in the team, and in the agency?

Meetings

- What is your opinion on your current regular meetings?
- Have you noticed any effects of the retrospective actions?
- How many meetings a day do you have on average?
- Do you feel the current meetings can be improved in any way?

Process

- What is your opinion on OKR?
  - Do you have any prior experience with it?
  - Do you think it allows you to work effectively?
  - How do you think OKR as a framework fits your team and product?
  - What do you think is the purpose of using OKR?
  - What do you perceive as difficult with OKR?
- Do you get any followup or documentation on OKR from management?
  - How do you feel about that?
- How are tasks picked and prioritized?
- Where is your threshold for taking on new tasks and domains?
- Do team members end up working with similar tasks and domains over longer periods of time?
  - Do you think it affects the knowledge redundancy in the team?
- Is there anything about your current process that could be improved?

Closing

- Is there anything else you would like to add?
- Do you have any questions about this interview?