University of Bergen
Master Thesis

Challenges with Scaling Scrum to Large-Scale Software Development: A Case Study

Author:
Simen Jensen

Supervisor:
Prof. Bjørnar Tessem

A thesis submitted for the degree
Master of Information Science

in the Department of Information Science and Media Studies

May 31, 2017

Abstract

Agile software development methods have become popular since the introduction of the Agile Manifesto in 2001. Agile methods, such as Scrum, are originally created for small co-located teams but have been adopted to large-scale development organizations. The accompanying challenges of using Scrum in large-scale development are not fully explored and understood.

This thesis aim to explore and identify challenges regarding large-scale agile development in a global software development organization. The research is done in form of a single case study, which empirically examine an organization's use of the Scrum framework. The results are analyzed with a congruence analysis of the case study, with basis in previous research on large-scale agile development.

The thesis results in four hypothesis which are categorized into three main problem areas regarding scaling Scrum in the organization: coordination, communication, and processes. The results form a collective basis for answering why large-scale Scrum is challenging to scale, and what Scrum characteristics makes it challenging.

Keywords: Agile, Large-scale, Scrum, Case study, Software engineering

Preface

This master thesis is written in my final year of the master program Information Science, at the Department of Information Science and Media Studied at the University of Bergen.

The thesis is written i collaboration with an anonymous large-scale development organization who have assisted the researcher, by providing an organization environment to investigate. I am very thankful for this.

I would like to thank my supervisor Professor Bjørnar Tessem for valuable guidance throughout the master course. I would also like to thank Kristoffer Aalhus and Jørund Vedøy for their fruitful academic discussions.

I thank everyone involved in the research project, who were willing to participate in interviews and/or observation sessions. I also thank my fellow master students, Ole Andreas Krumsvik, Hanne Åserød, and Eivind Flobak for their help and motivation.

Contents

List of Figures

List of Tables

List of Acronyms

CoP Communities of Practice

CPO Causal-process observations

HQ Headquarter

JIT Just-in-time

LeSS Large Scale Scrum

MTS Multiteam system

NENT National Committee for Research Ethics in Science and Technology

NSD Norsk senter for forskningsdata

PO Product Owner

RUP Rational Unified Process

SAFe Scaled Agile Framework

SM Scrum Master

SoS Scrum of Scrum

SoSoS Scrum of Scrum of Scrum

WIP Work in Progress

XP eXtreme Programming

Chapter 1

Introduction

This chapter gives a short introduction of the background for the project and research problem. Further, the chapter describes the targeted audience as well as the researchers personal motivation for the project. The research method, with its scope, and limitations is described before listing the thesis' structure

1.1 Project Background

Since the introduction of the Agile Manifesto in 2001, agile software development has gained widespread interest [1]. The most popular method, Scrum, is an agile framework that focuses on project management. Scrum provides developers with an environment that focuses on communication and collaboration between both customers and developers. Scrum is in many companies the de facto standard for developing software [1].

One of the challenges related to Scrum, that was not specified when the method was introduced, is how to scale up the method to larger projects [2]. Williams & Cockburn [3] states that agile methods are best suit collocated teams of about 50 people or fewer, which offer challenges for large-scale projects. Traditional Scrum is clearly specified and documented in most areas, but there are unexplored challenges related to large-scale Scrum, such as coordination across teams, and communication [4]. Teams in large-scale projects must coordinate their work with each other to not create impediments or other problems for the different teams.

In many cases, large-scale Scrum divides people into teams where each person is given a different role with specific responsibilities. Common for the

teams, regardless of the members' role is that they must cooperate and coordinate internally in the team. As projects grow, multiple people are required to finish the project, and the project's complexity grows, providing new challenges.

1.2 Research Problem

Since the entry of Agile Manifesto in 2001, several agile methods have appeared with basis in the manifestos tenets [5]. Agile in large teams was the dominant research question at the XP2010 conference [1]. Further, Dingsøyr & Moe, in a workshop report state that fundamental assumptions in agile development are strongly challenged when practiced in large-scale projects [1]. There is much information and research about agile in small teams [6], however there are few studies on large-scale agile, and the topic requires further research [4]. Paasivaara et al. [7] explicitly suggest that further *empirical* research is needed on how to tackle issues regarding large-scale agile projects, since agile development highly involves people and interactions.

The research in this thesis aims to contribute to an improved collective knowledge based on empirical research about large-scale agile projects. By identifying issues regarding large-scale agile development, one can further build on this knowledge within similar contexts, or apply insight from the results considering whether it is situationally relevant within e.g. a similar organization.

1.2.1 Research Questions

The following research question is investigated in the thesis:

> **RQ** - Why is scaling Scrum challenging for a large-scale development organization?

The research question aims to investigate the reasons behind why an organization performing large-scale agile development has issues with using the Scrum method. The question aims to provide answers to what characteristics make Scrum difficult to scale up.

1.3 Target Group

The thesis is aimed at people interested in large-scale agile development. It is aimed at researchers within the field, students who are looking for inspiration to their thesis or assignments, and for practitioners or organizations in the industry within the same case specific context who can learn from the studied organization's challenges.

The reader is not required to have any prior knowledge to the field of software engineering or agile methods, as the relevant topics are explained in the thesis. Any experience in the field, however, comes in handy for further reading and for their own inferences sake.

1.4 Personal Motivation

The use of agile methods in the industry is a relevant phenomenon for my future in the software industry. The Scrum framework, which is originally designed for small teams, co-located teams, promote challenges when adopted to large organizations. Since there is little knowledge about how best to adopt the framework, I find it interesting to see how an organization in the industry does this.

The reason for choosing the subject is because I knew little about it before. By exploring the subject, I hope to contribute to the research field, by identifying and classifying empirical data from a case study, which can further be investigated in the future.

1.5 Research Method

The research method used in this project is a single-case study, aiming initially to apply process tracing as a within-case analysis. The case study focuses on an organization's use of Scrum in a large-scale context, and goes in depth in the selected organization's processes and general use of the Scrum at scale. The method is used to observe the organization in its natural setting and gain a holistic view of the different phenomena.

Process tracing is attempted to be used as a qualitative analysis tool for describing and evaluating social phenomena. With use of the method, I attempt to evaluate evidence to establish causal connection between events. A central part of process tracing is to test diagnostic pieces of evidence,

by performing process tracing tests based on the evidence's sufficiency and necessity to establish causation. Process tracing requires causal mechanisms in the data. Due to lack of such, a wider approach was used.

The congruence case study method is used to evaluate the evidence in the case, without explicit mechanisms being traced. This led to weaker causal strength in the evidence. The congruence analysis still provides confirmatory or disconfirmatory claims of plausible causal relationships.

1.6 Scope and Limitations

The thesis addresses issues regarding large-scale agile development. Even though there are many agile development methods, frameworks, and strategies, the thesis focuses on Scrum. Other methods in Scrum's vicinity, such as eXtreme Programming (XP) are not included in the thesis, while others such as Kanban are, due to them being relevant for the case. Traditional methods such as Waterfall are outside the thesis' scope.

The research is done in form of a single-case study. While this allows for a deeper understanding of the specific case, it excludes a comparison towards another case with the same research method. Cross-case synthesis across multiple cases can make case-analysis easier, and strengthen the findings [8]. While there is some research regarding large-scale agile, there is not a lot compared to traditional agile research [9]. This poses a challenge because of the lack of basis for comparison.

Because of the master thesis limited time frame, I was unable to acquire more data for the case study. The research is done in collaboration with a globally distributed software development organization. This meant that preferably, the researcher should have traveled to the organization's different development sites to acquire data from first hand sources. This was omitted from the research design due to time and cost limitations.

1.7 Thesis Structure

The thesis is structured into six chapters:

Chapter 1. Introduction

Chapter 1 give a short introduction of the thesis, including project background, research problem, target group, personal motivation, research method,

scope and limitations, and research contribution

Chapter 2. Theory

Chapter 2 presents the relevant literature for the thesis. The chapter introduces terms and concepts connected to agility and agile methods. Furthermore, the chapter describes current literature and related work on some of the dominant issues regarding scaling agile methods

Chapter 3. Research Method

Chapter 3 gives an overview of the research methods used in the thesis. The chapter introduces relevant literature about the methods, as well as the case. This includes an explanation of the studied organization, along with execution of the research plan, design, and data collection.

Chapter 4. Results

Chapter 4 describes the case's context, and how the thesis' hypotheses were generated. Further, the chapter describes each hypotheses in detail, with basis in the data collection combined with relevant literature, and is thereafter summarized.

Chapter 5. Discussion

Chapter 5 resumes the research questions. The hypotheses are discussed with basis in the research questions. Finally, the study's design, process, findings, and ethics are evaluated based on the conducted case study.

Chapter 6. Conclusion

Chapter 6 presents the key results found in the research. Further, the chapter introduces possible future work based on the the research's results.

Chapter 2

Theory

This chapter will present the relevant literature for the thesis. The chapter introduces terms and concepts connected to agility and agile methods. Furthermore, the chapter describes current literature and related work on some of the dominant issues regarding scaling agile methods.

2.1 Agile Methods

Agile methods is a collective term used for the different methods that are seen as agile. Common for the different methods is that they follow an agile mindset. Agility is a term that is commonly used when describing animals that are nimble, fast, or flexible. When we adapt the term to humans in a software development context, the term keeps core meaning, that one can adapt to an environment. Adaptation is at the core of agile methods and is embedded in the values of the Agile Manifesto [10]. Agile development is not a specific process one follows, but rather a way of working, that is built on the agile philosophy [11]. The agile philosophy is way of thinking that in 2001, was described in the agile Agile Manifesto [10]. The manifesto consists of four values and twelve principles that serves as a basis for the different agile methods. The four core values in the Agile Manifesto are:

Individuals and interactions over processes and tool

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

The manifestos' authors underline that the items on the left are more heavily

emphasized than the right ones. This does not mean that agile teams e.g. disregard all documentation above working software, but rather that it is valued more and thereby avoid documenting elements that do not increase the product's value [12].

Agile methods were introduced as a reaction to traditional software development, which are often document driven and process oriented [6]. Traditional development follow a high-level structure, where a set of requirements are predetermined before the development starts, and followed and inspected throughout the process. This practice has been criticized for being impossible to follow through due to the industry and technology moving too fast [6]. One of the prominent factors to the creations of agile methods, is organizations' constant need to react to marked dynamics, technological innovations, and new customer requirements [13].

Agile methods stress the principle of people interacting and working together to produce working code [14]. By promoting interactions between individuals, sharing of information becomes an ordinary event that help people learn and become better at their work. By interacting frequently with customers, the process can be changed quickly when it needs to. This allows the project to respond to unpredicted events and therefore change over following a stipulated plan. Continuous interaction within the project also compensates for minimizing documentation by making sure the involved parts are up to date on coherent tasks [14]. One of the key principles of agile methodology is to add business value to the customer by delivering working software [5]. Documentation in itself does not add value to the customer, and is therefore given a lower priority over e.g. working software that the customer values more.

Traditional methodologies provide strict rules and regulations for handling most situations [14]. Agile methods however, offer a minimum set of actions that must be completed to handle special situations. This promotes individuals to find creative ways to handle their problems and adapting them individually for each team. This way of thinking, that is in coherence with agile principles and values, can be referred to as having an agile mindset.

2.1.1 Scrum

Ken Schwaber and Jeff Sutherland developed a guide for Scrum, with roles, events, artifacts, and rules, called The Scrum Guide [15]. Scrum is defined

as a lightweight agile process framework that is and has been used since the 1990s to manage product development.

Scrum facilitates for people to follow the three pillars of empirical process control: transparency, inspection, and adaptation [15]. Transparency refers to the process being visible to the people responsible to the outcome. Inspection is required to detect undesirable variances and avoiding them. Adaptation attributes to the ability to adjust unacceptable results by altering the process or other relevant factors. Scrum provides four events for inspection and adaptation: daily scrum, sprint planning, review, and retrospective [15].

The central values in Scrum were stated by Schwaber and Sutherland as commitment, courage, focus, openness, and respect [15]. They state that Scrum depends on people following these values in order to achieve the goals of the team.

Scrum team

A Scrum team consists of a scrum master, product owner, and a development team. By following the Scrum values and working self-organized and cross functional, the team choose self how best to carry out their work [15]. By following these principles, the team is designed to operate productively, flexible, and creatively.

Scrum Master The Scrum Master (SM) role is responsible to help and facilitate Scrum for the rest of the team. The SM ensures that the team understand the process by presenting the team with relevant Scrum theory, practice, and rules [15]. The SM also help people outside the team understand the which interactions with the team are helpful and which are not. SMs can also serve on an organizational level by leading and coaching the organization in adapting, planning, or implementation Scrum. The role of an SM is essentially facilitating and making the process go smoothly of all involved parties in the Scrum process.

Product Owner The Product Owner (PO) is the person responsible for maximizing the work and product being developed [15]. The PO's main responsibility is the product backlog, which is a list of the product's needs. The backlog needs to be clearly expressed, ordered, optimized, visible, transparent, clear, and understood. All of the tasks can be done by anyone in

the team, but are ultimately the PO's responsibility to maintain [15]. The PO can typically get help from the SM with e.g. finding techniques for managing or planning the backlog, since the roles are compatible.

Development Team The development team consists of a group of professionals who work on the product that is being developed. The team is self-organized and turn the product backlog into increments of a potentially releasable functionality [15]. The team consists of three to nine team members to achieve optimal productivity. The team must be large enough to uphold internal team-interaction, but still small enough to be able to coordinate efficiently. Scrum does not recognize other titles within the development team, other than developer. The team may have specialized skills or areas of responsibility, but the team as a whole is accountable for the development [15].

Scrum Events

Scrum (illustrated in figure 2.1) prescribes events to minimize the need for unforeseen meetings [15]. All events in Scrum are time-boxed, meaning their maximum duration is predefined. The events in a sprint may end when the purpose of the event is achieved, leading to less waste in the Scrum-process.

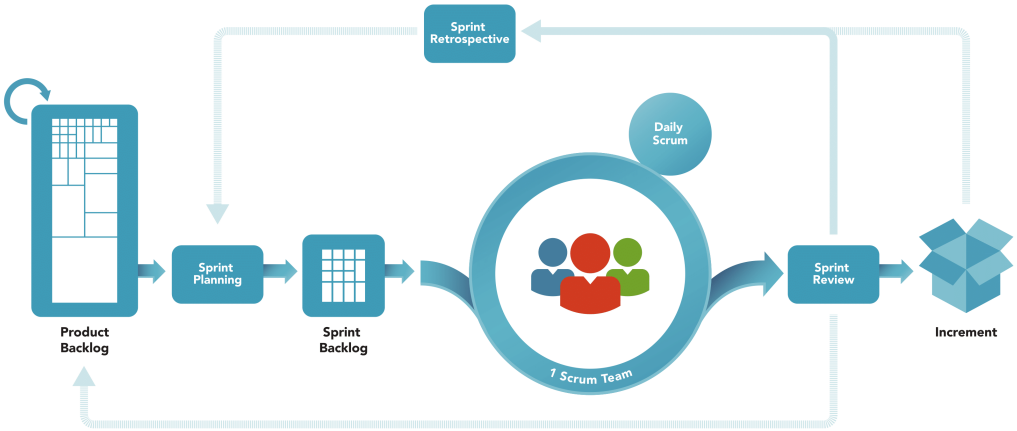

Figure 2.1: The Scrum framework [16]

Sprint The sprint is the container for all other events in Scrum, and is the core of Scrum [15]. Sprints last for one month or less, and during this time, a product increment is developed to such an extent that it can be potentially releasable. Sprints are conducted consecutively and continuously until the product is completely developed.

Sprint Planning At the start or before every sprint, there is a sprint planning event. The sprint planning is conducted by the entire Scrum team, and is a collaboration to make the best possible plan [15]. Sprint planning lasts for maximum eight hours. The sprint planning's goal is to find out what increment can be delivered in the upcoming sprint and how this work will be achieved [15].

Daily Scrum Daily scrum is a meeting dedicated for the development team to synchronize their activities until the next day. Daily scrum meetings are designed to answers three questions [15]:

- What did I do yesterday that helped the development team meet the sprint goal?
- What will I do today to help the development team meet the sprint goal?
- Do I see any impediment that prevents me or the development team from meeting the sprint goal?

The meeting is a key inspect and adapt meeting, meant to improve communication, eliminate other meetings, identify impediments, highlight and promote decision-making, and improve the teams' knowledge [15].

Sprint Review Sprint reviews meetings are held at the end of the sprint, and are meant to inspect the product increment and optimize its value by collaborating and give feedback on the choices made during the sprint. The main goal of a sprint review is to revise the product backlog to meet new challenges and opportunities [15]. Sprint reviews include the following elements among others [15]:

- Attendees include the Scrum team and key stakeholders invited by the Product Owner.
- The product owner explains what Product Backlog items have been "Done" and what has not been "Done".
- The development team discusses what went well during the sprint, what problems it ran into, and how those problems were solved.
- The entire group collaborates on what to do next, so that the sprint review provides valuable input to subsequent sprint planning.

Sprint Retrospective The sprint retrospective occurs between the sprint review, and sprint planning. The goal of a retrospective is to create a plan for the upcoming sprint for how it can be improved, based on experiences and knowledge from the previous sprint. A retrospective is time-boxed to a maximum of three hours. The key activities in a retrospective are:

- Inspect how the last sprint went with regards to people, relationships, process, and tools
- Identify and order the major items that went well and potential improvements
- Create a plan for implementing improvements to the way the Scrum team does its work [15]

The sprint retrospective provides the team with a formal opportunity to increase productivity in the following sprints, by learning from previous mistakes.

2.1.2 Kanban

The lean approach to software development has become increasingly popular in recent years [17]. Kanban is a process tool which builds on the lean mindset, and is meant to increase efficiency, by providing a framework with a minimum set of constraints and guidelines for the user to work with [18]. The lean mindset originates from the manufacturing industry at Toyota, aiming to deliver value to the customer by eliminating waste and delivering only what is needed, when it is needed, and in the amount needed, also known as just-in-time (JIT) [17].

Both Scrum and Kanban are less prescriptive than traditional methods, meaning they have fewer constraints regarding what is allowed to do and/or not allowed to do. Kanban is nevertheless a more adaptive tool compared to Scrum, which means it has fever rules to follow than Scrum. Kanban has only three constraints, while traditional methods such as RUP has over 120 constraints, as seen in figure 2.2.

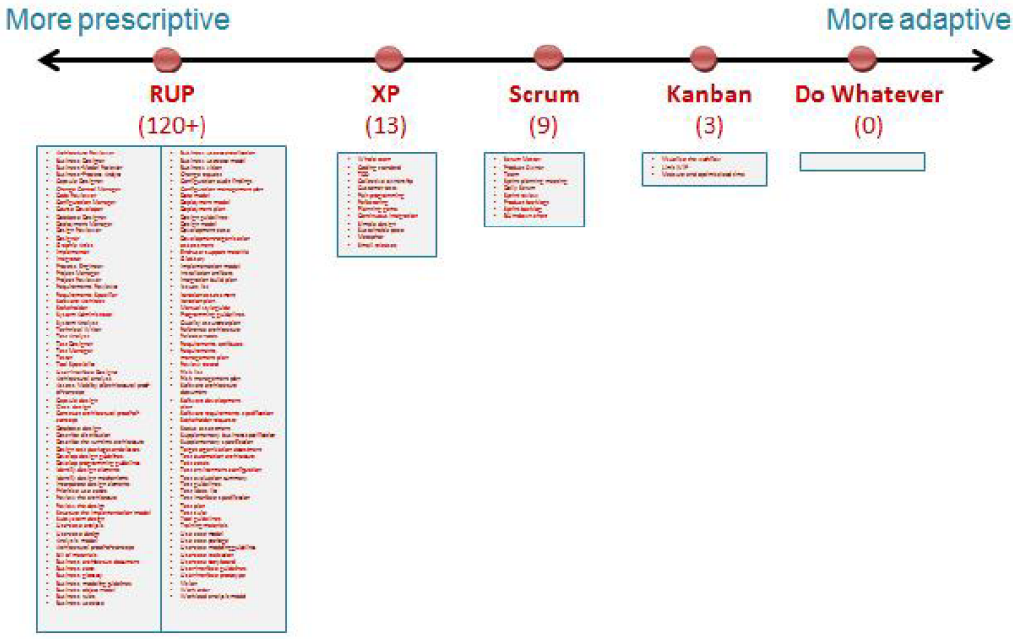

Figure 2.2: Illustration of the amount of constraints in different methods. Prescriptive vs adaptive scale [18].

One of the differences between Kanban and Scrum, is that Kanban does not prescribe any roles. This does not mean that it does not allow roles. Kanban allows you to add the roles that you need, based on the users needs. However, Kniberg [18] states that the general mindset of Kanban is "less is more", meaning that if you are uncertain, start with less. The same goes for timeboxed iterations, they are not prescribed, but one can choose to add the activities. This makes Kanban a more flexible framework compared to Scrum, and allows for modifications based on the users needs.

The three prime principles of Kanban are: **1)** Visualize the workflow, **2)** Set work in progress (WIP) limits, and **3)** Measure the flow [17].

1) Kanban visualize the workflow in one single Kanban board, as shown an example of in figure 2.3. Kanban boards does not have to look like this, and can be modified in the way the team sees fit. The purpose of a Kanban board is to create a smooth flow through the process and minimize lead time [18]. Lead time refers to the average time it takes to complete one item.

2) The WIP limit is the number items that can be worked on at the same time. For example, in figure 2.3 two tasks can be worked on at the same time in the "Selected" column, while three can be worked on at the same time in the "Develop" column. This is done to evolve a culture of trying to get items done before starting new items [18].

3) Measuring the flow is an important element in Kanban. Continuous delivery of value to the customer should be emphasized. The main goal of measuring the flow is to investigate opportunities that can increase the flow which leads to delivering better value to the customer faster.

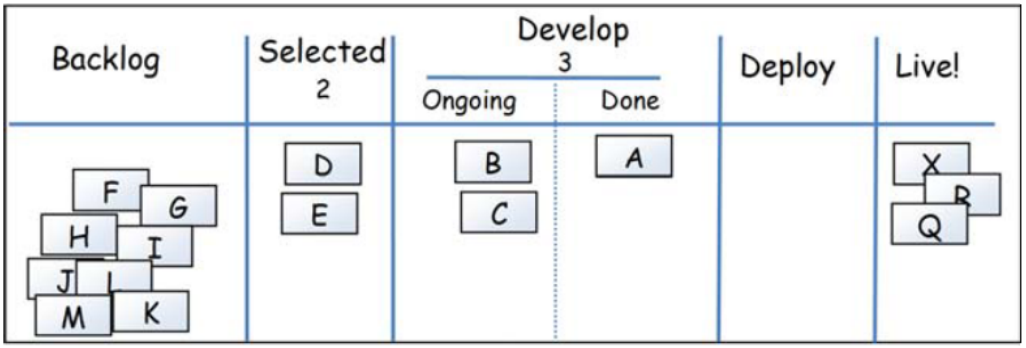

Figure 2.3: Example of a Kanban board [18]

Studies on using the Kanban approach show various benefits. The software development process and management of the software process were reported as the most prominent topics done in the studies [17]. Table 2.1 shows the benefits of using Kanban in software development, based on 37 primary studies on the Kanban approach [17].

Table 2.1: Benefits of the Kanban approach [17]

Benefits	**Percentage of Reported Benefits**
Enhancing visual control that facilitated and supported the decision-making process	45.9%
Facilitating the coordination of cross-functional teamwork and imposing self-organization	37.8%
Empirically introducing quality circles and kaizen events	29.7%
Reducing the cycle time/lead time	29.7%
Increasing customer satisfaction and realizing high value	27%
Decreasing market and technical risks of the product	24.3%
Developing continuous improvements strategies	45.9%

Increasing the predictability in the delivery of the final products with the constraint of changing customer requirements	35.1%
Ensuring skills development and cohesiveness for teams	16.2%
Driving and facilitating organizational change management and culture changes	32.4%
Enhancing quality of product, indicated by decreasing the defects rate, increasing the quality assurance pass rate, and reducing the number of bugs	16.2%

2.1.3 Scrumban

Scrumban has evolved as a framework for developing software over the years [19]. However, it still does not have a guide or definition as e.g. Scrum does. Scrumban is a mixed method consisting of a combination of Scrum and Kanban. The method adopt elements from both methods with the aim to integrate the task board workflow with Scrum concepts, and visualization from Kanban [20].

Scrumban's mixture of elements from both methods promotes Scrum structures such as, retrospectives, reviews, and daily updates, with the combination of Kanban's WIP limits and clear execution of stages [20]. Table 2.2 compare and summarize the differences and similarities between Scrum, Kanban, and Scrumban.

Table 2.2: Differences and similarities between Scrum, Kanban and Scrumban [21]

	Scrum	**Kanban**	**Scrumban**
Rules	Very descriptive	Not descriptive	Descriptive
Boards	Reset for each sprint	Used continuously	Used continuously
Roals	SM, PO, and the team	Specialized team	Specialized team
Iterations	1-4 week sprints	Need based	Need based
Planning routines	sprint planning	On-demand & release planning	On-demand & bucket planning

Task estimation	Done before each sprint	During planning (optional)	Optional
Task assignment	Assigned to the team	Taken by team members	Taken by team members
Prioritization	By refined backlog	By priority columns	By priority columns
Task limits	Limited by sprints	Limited by WIP	Limited by WIP
Meetings	Planned mandatory	On-demand and optional	On-demand and optional
Performance metrics	Burndown chart	Lead & cycle time cumulative flow	Lead & cycle time, average cycle time
New tasks in a live iteration	Not allowed	Allowed	Allowed

The Scrumban method is believed to be especially suited for maintenance projects, projects prone to programming errors, or projects with unexpected user stories [22]. In these kind of projects, timeboxed sprints are unnecessary, whereas other Scrum events may still be useful [22].

Reddy [19] states that Scrumban over the years has been used to help organizations accelerate their transition to Scrum from other development methods.

Scrumban is however not recommended for very large projects [19]. Scrumban is suggested for projects with no more than five teams, because coordination issues may becomes too big of a problem [19].

2.2 Scalability

As agile methods have become increasingly popular in the world of software development, new challenges arise. Agile methods were originally designed for small co-located teams [7], but are today used in large-scale contexts. Problems with how to scale up agile methods to a large-scale have been an issue that has not been fully considered when the methods were first created. Most studies on scalability offer a selection of agile principles, values, and industry best practices [2]. There are few empirical studies with a theoretical underpinning that focuses on the scalability of Scrum

[4]. Furthermore, the top burning research question by practitioners at the XP2010 conference, was “agile and large projects” [23]. This displays the need for research on the topic.

Within the research community, there is disagreement on what can be considered as ”large-scale” agile development [4]. Projects that are “large-scale” can e.g. refer to the project cost, number of teams, lines of code, number of requirements. Dingsøyr et al. [9] argue that these some of these factors are unreliable when defining what is large-scale. Costs vary across projects and countries. Code can be generated by tools or be modifications of an existing code. Requirements vary in implementation-time or other variabilities. However, agile methods have a strong emphasis on communication. To achieve efficient communication, the number of people involved in the development is therefore an appropriate measurement of the project's scale. Scrum allows multiple Scrum teams to work on different parts of the same project. A project that includes multiple Scrum teams can therefore be defined as ”large-scale” according to Dingsøyr et al. [9].

Roach [24] describes three dimensions of growing Scrum, scale, distribution, and saturation. Distribution refers to a number of teams being located at different geographic locations. Saturation means to which extent you have Scrum pervaded in the organization, breaking down the traditional ”silo-structure”. Scale is described as the number of coordinating teams working on large projects. This definition matches Dingsøyr et al's [9] definition on scaling Scrum, and is what this thesis addresses when referring to ”scale” or ”large-scale”.

At the workshop Agile2011, many academics meant that the themes that should be further researched were: agile across projects and across organization, the ”core” of agile, distributed agile development, and the role of architecture and knowledge management in agile development [5].

At the XP2013 workshop at the 14th International Conference on Agile Software Development, participants answered what they thought was the most important research challenges regarding large-scale agile development [4]. The answers were analyzed and grouped into topics shown in table 2.3.

Table 2.3: Suggested research agenda on large-scale agile software development [4]

Rank	Topic	Description
1	Inter-team coordination	Coordination of work between teams in large-scale agile development

2	Large project organization / portfolio management	What are effective organizational structures and collaboration models in large projects? How to handle a distributed organization?
3	Release planning and architecture	How are large projects planned? How can the scope be reduced? What is the role of architecture in large-scale agile?
4	Scaling agile practices	Which agile practices scale and which do not? Why and when do agile practices scale?
5	Customer collaboration	How do product owners and customers collaborate with developers in large-scale projects?
6	Large-scale agile transformation	How can agile practices be adopted efficiently in large projects?
7	Knowledge sharing and improvement	When is the whiteboard not enough? How can communities of practice be established? What measurements are relevant to foster improvement?
8	Agile contracts	How can contracts change the mindset of customers from upfront planning to agile principles? What legal limitations exist in contracts that reduce agility in large projects?

2.2.1 Inter-Team Coordination

An inter-team context present challenges as large-scale development require coordination. Matheieu et al. via Scheerer et al [25] state that this usually results in a hierarchical team of teams setup. Furthermore, the organizational setup is defined as a multiteam system (MTS), which is defined as:

> Two or more teams that interface directly and interdependently in response to environmental contingencies toward the accomplishment of collective goals [25] (p. 4780).

In an attempt to handle inter-team communication, the Scrum of Scrums (SoS) technique was introduced to large-scale Scrum contexts. While it is not a part of traditional Scrum, it is widely used in large-scale organizations in the industry [7]. The technique facilitates an environment where the different Scrum teams can coordinate and plan their progress together in

a single event. SoS's are time time-boxed and have the same basic format as daily scrum meetings. However, SoS meetings are commonly attended by only one representative from each team [7]. SoS meetings are usually conducted in the same format, where four questions are answered by every team [7]:

1) What did you do since the previous meeting?
2) What will your team do by the next meeting that is relevant to other teams?
3) What obstacles does your team have that affect other teams?
4) Are you about to put something in another teams's way?

The meeting can be arranged from once a day, to 3 times a week, based on the demand. Dingsøyr et al. [9] suggests coordination in large- (2-9 teams) and very large-scale (10+ teams) teams can be conducted in forum such as Scrum of Scrum meetings.

Ktata & Lévesque [26] state that it is more challenging to scale, than to simply use approaches such as SoS. Scaling is about "ensuring effective knowledge sharing and making the right decisions" [26] (p. 63).

Scrum of Scrum ceremonies have been studied, and was in some instances found to help manage inter-team coordination, especially in projects where teams are not co-located [27]. However, other results from studies on SoS, show that audience was too wide to keep everybody interested [7]. The participants in the study [7] also had trouble knowing what was valuable to report at the SoS meetings, and ended up not reporting anything. SoS have been identified as extremely challenging regarding coordinating on a inter-team level, in seven incidents [25], which show that research on these types of issues are needed.

Events similar to SoS have been suggested to handle inter-team coordination issues, such as Feature Coordination CoPs. This Community of Practice (CoP) are meetings where a few teams working on a common feature, work together with the same feature at the meeting [28]. Paasivaara et al. [28] found this technique to be successful in their study on CoP in large-scale agile development. Their research, however, show that SoS did not work in the studied organization [28].

2.2.2 Distributed Large Project Organization

Large-scale projects and the distribution of teams are often interconnected. When projects are of a large scale, organizations might have to employ teams that are not co-located in order to achieve enough developers, competency, reduce costs or similar factors [29]. A survey from 2008, show that several respondents indicated that they were successfully applying agile development with over 200 people [30]. Furthermore, they also indicated that they were applying distributed agile development. Ambler [30] points to geographical and organizational distribution as two of the main factors to consider when scaling agile methods. Geographical distribution refers to persons or teams being located at different places, while organizational distribution refers to teams or persons in different departments, divisions, companies etc.

When conducting distributed agile development in several countries, the chances of encountering different organizational cultures increase [31]. However, several studies show that differences in national culture are not the dominant influential factor regarding cultural issues, but rather cultural factors such as the role of values and norms, and attitudes towards bureaucracy and authorities [31]. Their work show that the implementation of agile methods needs to consider the role of the relevant culture, or else its implementation will likely prevent the team from performing optimally [31]. Difference in culture can affect a persons understanding of each other, their values, or normative practices [32]. Bannerman et al.'s [32] work suggest that this may lead to issues related to different perceptions of authority, inconsistent work, lack of mechanisms for creating shared understanding, and reduced cooperation. Issues regarding distribution can be divided in three categories: temporal distance (time zones, synchronizing work), geographical distance (physical presence), and socio-cultural distance (persons norms, values, and perspectives) [32].

2.2.3 Release Planning and Architecture

The way a lot of large-scale Scrum processes are organized, with self-organizing teams and SoS meetings, development teams should ideally have a low degree of inter-team dependencies. A common way of defining coordination is "management of dependencies" [33]. However, researchers found that POs are often complaining about creating value of the customers and prioritizing requirements due to coordination issues [34]. Scheerer et al. [25] recognize

incomplete and ever-changing requirements along with interdependencies in requirements as some of the main problem areas in a large-scale agile development context.

Large scale development increases the complexity of the organizational structure. Pernstal et al. [35] recognize problems with dependencies between development tasks, and artifacts, leading to communication and coordination issues across departments at Volvo Car Corporation and Volvo Truck Corporation. Pernstal et al. [35] argue that applying agile methods such as Scrum as a standalone solution to handle these issues, would not work [35]. However, lean practices and principles, building on lean product development would work according to Hibbs et al. [36] and Petersen [37] via Pernstal et al. [35]. This is because it can be applied to any scope, and its holistic view is a prerequisite for scaling agile [35]. Even though there is research on how lean principles and practices can be used in software engineering, Pernstal et al. [35] argue that empirical evidence is needed in a large-scale development context when looking at lean practices and principles.

Evbota et al.'s [38] exploratory case study research on large-scale agile organizations resulted in a model (see figure 2.4) which gives an overview of key aspects of collaborative planning in large-scale agile development.

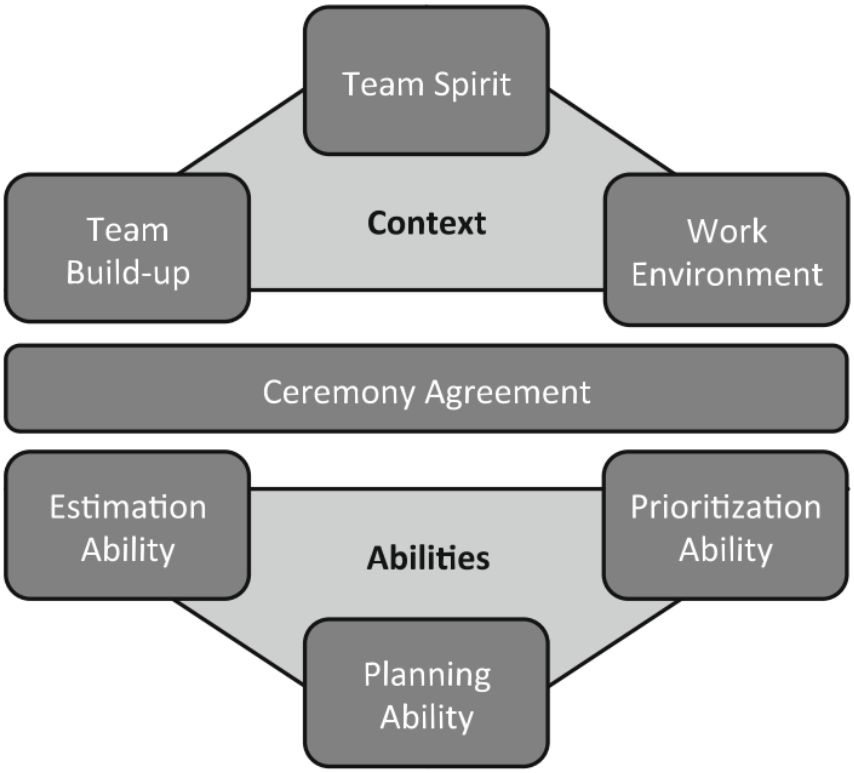

Figure 2.4: Key aspects of collaborative planning in large-scale agile organizations [38] (p.32)

Evbota et. al. [38] found that agile teams need to bring together the ability to estimate required work, combine knowledge into a good plan for the coming iteration(s), and prioritize with respect to business value as the most important technical ability challenges, regarding planning for large-scale agile organizations. Communication is the main challenge regarding the three factors, displayed in figure 2.4. Estimation ability involves challenges to make a long-term estimate because of too much content in e.g. the product backlog. Prioritization ability is the view of what is to be prioritized, which often leads to disagreements, inconsistencies between backlogs, and lack of transparency. Planning ability refers the transformation of priorities into a concrete plan.

Further, Evbota et. al. [38] found challenges tied to the context of planning, specifically team build-up, team spirit, and work environment. Team build-up refers to the product owners and program leaders view of the team's capabilities to develop the product. Team spirit is the teams' ability to work together and function successfully after working together over a period of time. Work environment refers to the general work area, how it is organized in relation to information flow between teams and individuals.

Ceremonial agreement is specified as the room between the two domains

[38]. It includes issues related to information flow (described in subsection 2.3.1), and coordination (described in subsection 2.2.1)

2.2.4 Large-Scale Agile Frameworks

Industry organizations that use Scrum in a large-scale setting have recognized issues related to the framework. This has lead to the development of new frameworks, specifically tailored to large-scale development. Some of the most well known frameworks are Large-Scale Scrum Framework (LeSS) and Scaled Agile Framework (SAFe)

Large-Scale Scrum Framework (LeSS)

The LeSS framework (shown in figure 2.5) uses Scrum in large, which allows for many teams to work together on one product [39]. It provides the users with a set of rules and principles to follow. The authors of the framework state themselves that LeSS is not a new and improved Scrum, but rather about figuring out how to apply the different elements that LeSS provides [39].

> LeSS is not really about enabling an existing big group to 'do Scrum at scale.' Rather, LeSS is about descaling the organization, and creating a design that systemically enables agility at scale, with simple elements, to be LeSS Agile [39].

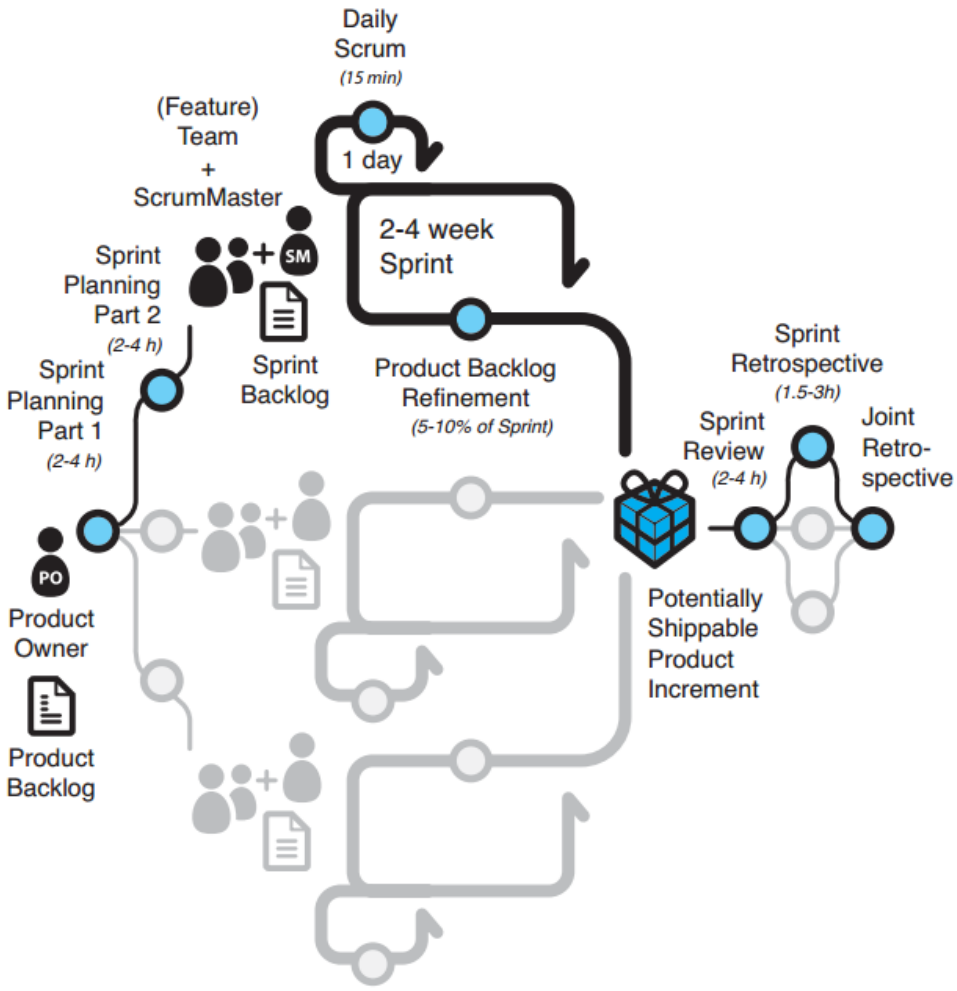

Figure 2.5: The LeSS framework illustration [39]

Compared with traditional Scrum, LeSS is a step in the other direction than for example Kanban. The LeSS framework adds more rules and guidelines. However, the framework is not very prescriptive compared to SAFe. The authors state that it is a "barely sufficient methodology", in the same way as traditional Scrum is [39].

Scaled Agile Framework (SAFe)

SAFe (shown in figure 2.6) is a framework that build on underlying agile and lean principles [40]. The framework is prescriptive in the way that it applies a lot of rules to every part of the process. The four main levels of SAFe is team, program, portfolio, and value stream [40]. These levels are explained in detail on how everything from team structure, to enterprise strategy should be executed by the organization.

Ken Schwaber, one of the authors of the Scrum Guide, is however negative

to the SAFe framework due to its rigidness.

> The boys from RUP (Rational Unified Process) are back. Building on the profound failure of RUP, they are now pushing the Scaled Agile Framework as a simple, one-size fits all approach to the agile organization. They have made their approach even more complicated by partnering with Rally, a tools vendor. Consultants are available to customize it for you, also just like RUP [41].

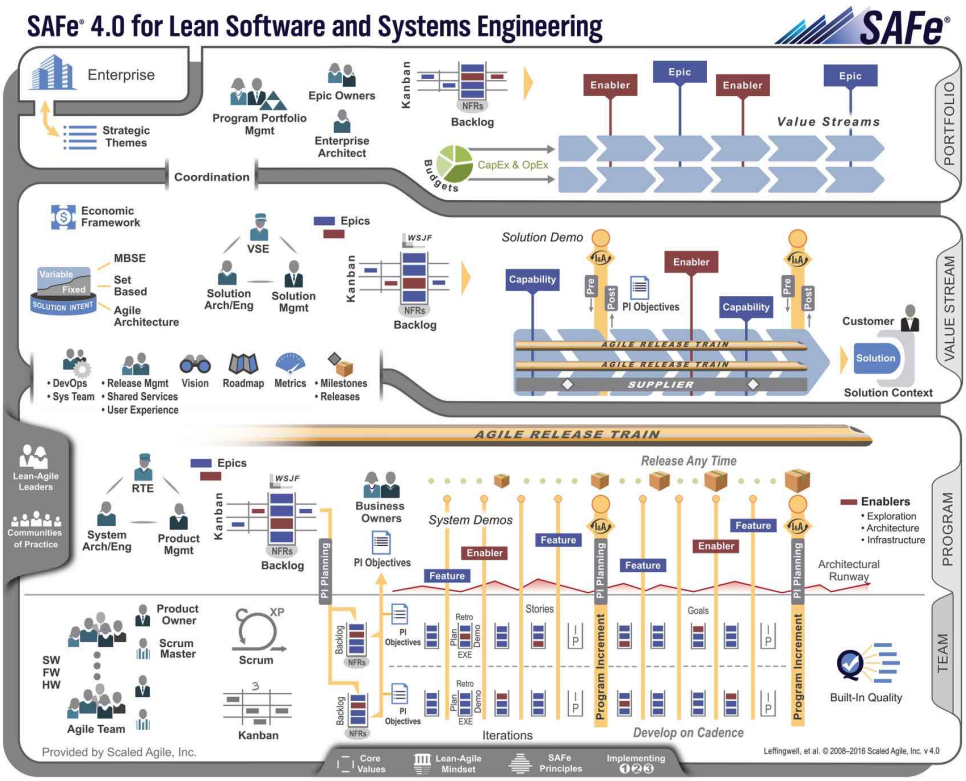

Figure 2.6: The SAFe 4.0 framework diagram [40]

Both of the above-mentioned frameworks are more prescriptive than traditional Scrum. They attempt to handle issues related to scaling Scrum such as inter-team coordination, release planning, and project organization by applying rules and guidelines to the process.

Even though there are multiple techniques and frameworks that help scale agile development processes, there is not a common "best practice". This shows that there is need for empirical research on organizations' use and experiences with scaling agile methods.

2.3 Organizational Communication

Communication is among the most important processes in any organization. Communication is defined as a process where persons or groups send or exchange information, with focus on the transmission [42]. It can further be specified to the transmission of not only information, but also ideas, attitudes, feelings in the form of verbal or non-verbal signals. A simple communication process between two actors is exemplified in figure 2.7.

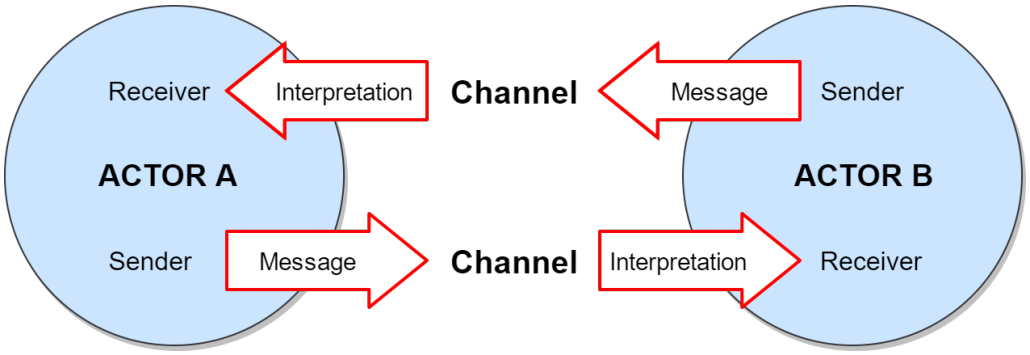

Figure 2.7: Communication process [42] (p. 281)

Communication is decisive for both internal integration and external adaptation [42], and is therefore an important to any organization. Jacobsen & Thorsvik [42] divide organizational communication into six subcategories that specifies communication's importance: Coordination, culture, decision making, learning, information retrieval, and information dissemination.

2.3.1 Communication Channels

A communication channels' ability to transmit rich information is central in communication research [42]. Four points can be defined for channel to be able to give rich information, when it:

a) can transfer many different signals simultaneously

b) gives opportunity for fast feedback

c) gives the opportunity to utilize natural or oral language, and

d) ables the sender and receiver to personalize and adapt the message to each other [42].

The goal of communicating in an organizational context is often to achieve rich, precise, and fast communication. Several organizational communica-

tion researchers have found that employees prefer, and if given the ability to choose, choose face-to-face communication, as they think this is the most effective, especially relative to time [42]. Face-to-face communication facilitates for non-verbal communication trough body language which allows the sender to receive information about interest, opinion, and status between to parties. It also allows for transmission of feelings, which is important for the receiver to interpret [42]. These points are some of the most important losses of rich information, when communicating through for example text. Figure 2.8 shows the degree of richness in the information, based how the information is transmitted.

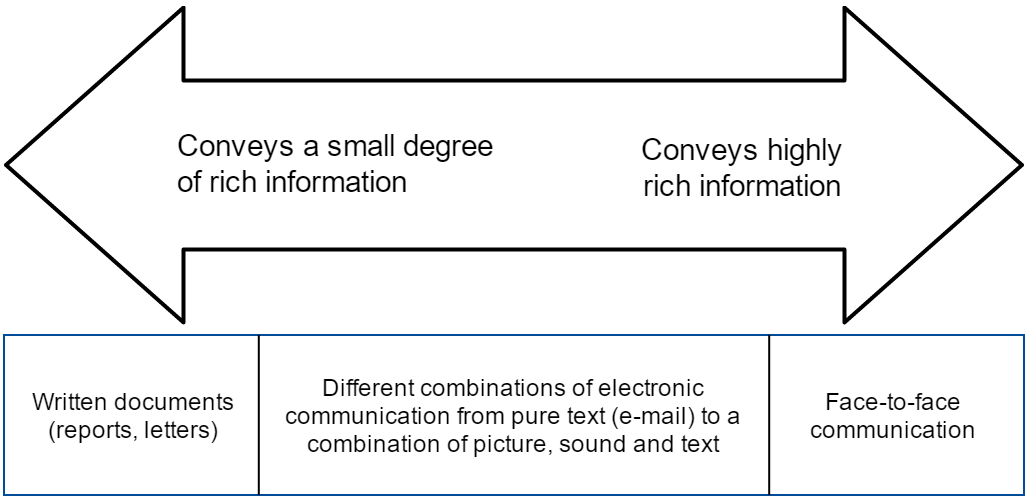

Figure 2.8: Different channels' ability to transmit rich information [42] (p. 286)

2.4 Organization Culture

Organization culture is the term used for describing cultural processes that exist within an organizational context. In most definitions of organization culture, there is an explicit focus on thoughts, experiences, and meanings that are common for multiple persons in an specific social context [42].

One of the more popular definitions of organizational culture is Edgar Schein's:

> Organization culture is a pattern of basic assumptions developed by a given to a group as it learns to master its problems with external adaptation and internal integration - as functions sufficiently well until its considered as true, and therefore is learned to new members as the right way to perceive, think, and feel in relation to these problems [42] (p. 130)

Organization culture becomes an increasingly important factor to consider when applying distributed agile development in areas with different culture, as the management complexity rises. Jacobsen & Thorsvik [42] point out that organizational culture is studied in order to find basis for whether there is a joint experience of belonging and fellowship, as this is crucial for the organization's successfulness. Organizational culture can be described with five general effects on behavior, explained in table 2.4.

Table 2.4: General effects on behavior [42]

Belonging and fellowship
Organization culture can give a basis for social fellowship and belonging. Covering these basic needs, can contribute to reduce uncertainty and insecurity. To many people, the organizational culture will to a great extent define their identity, which can cause less absence from work, better stability, and relations in the work
Motivation
The experience of social fellowship and identity can contribute to motivated employees. By allowing employees to feel that they are working for something bigger than themselves has shown to have a motivating effect.
Trust
The trust between management and employees, and employees in different work-areas, depends largely on the strength of the organization's culture. By having a strong culture, the trust increases, and there is less need for monitoring. Trust can in many cases work as a substitute for information- and administrative control.
Cooperation and coordination
All the previously mentioned effects are contributory to making the cooperation between people, groups, and hierarchical levels easier. By feeling a belonging, a fellowship, motivated and trusted helps members of the same culture develop a common language, a way of communicating that makes cooperation easier. Having a good culture impacts coordinating activities, since it happens through a joint desire to cooperate, rather than being forced to.

Management
Culture can specify what is appropriate behavior when a person is going to complete a task on behalf of the organization. If there is a lack of e.g. trust between employees and management, the organization can introduce a forced bureaucratic management culture. When an organization replace the use of forced bureaucratic management, is this often called clan-management.

Research on the relationship between agility and organizational culture mainly show that adoption of agile methods often is problematic [43]. This is often due to factors such as incompatibility between development culture and hierarchical culture orientation [43]. In large software organizations, it is also common that subcultures appear. This is admittedly not necessarily a negative thing as it can lead to a rich diversity in the organization [42].

Chapter 3

Research Method

This chapter will give an overview of the methods used in the thesis. The chapter introduces relevant literature about the methods, as well as the case. This includes an explanation of the studied organization, along with execution of the research plan, design, and data collection.

3.1 Case Study

Case studies focus on one instance of something that is to be investigated. The case is comprehensively studied, typically using data generation methods such as interviews, observation, document analysis, and questionnaires [44]. Case studies aim to gather rich and detailed information about the specific case, its processes, and relationships. By gathering rich data about the specific case, the researcher can explain how and why certain outcomes occur in given situations. This allows the researcher to retain a holistic and real-world perspective of specific case in both small group behavior and on an organizational level [8]. Oates [44] identifies four features that characterize case studies,

- Focuses on depth rather than breadth. The case is observed in detail, and the researcher therefore gains in-depth data about the phenomena.
- Natural setting. The case is explored in its natural setting and the participants continue to "work as normal". The researcher tries to disturb the setting as little as possible.
- Holistic study. Complex relationships and processes are researched to find out how they are connected to each other.

- Multiple sources and methods. Data is gathered for multiple sources, both qualitative and quantitative to gain exhaustive research data.

3.1.1 Case Study in Software Engineering

Case study is a research methodology often used in software engineering research, to study various contemporary phenomena in its natural context [45]. As empirical studies have become accepted in software engineering, the knowledge within the field is growing continuously. Runeson & Höst [45] points out, that in order to investigate complex real life issues involving human interactions with technology, we have to move beyond analytical research paradigms since the don't provide sufficient insight [45]. The plan for the data collection is characterized by the researcher being separated from the participants, meaning that the researcher will not intervene in the teams' or managements day-to-day work. The observational data is collected by participating in formal and informal meetings as a passive observer, going to lunch with the teams, listening to both formal discussions and casual chatter. Data is also collected from interviewing different team members from diverse teams, and relevant members of the company's management

Case studies often serve different purposes based on what the researcher want to achieve with the work. Runeson & Höst [45] distinguishes between four types of case studies that each serve a different purpose.

- Exploratory – Finding out what is happening, seeking new insights, and generating ideas and hypothesis for new research
- Descriptive – portraying a situation or phenomenon
- Explanatory – seeking an explanation of a situation or a problem, mostly but not necessary in the form of a causal relationship
- Improving – trying to improve a certain aspect of the studied phenomenon

The case study investigation at the organization was a mixture of exploratory and explanatory purpose, depending on the different stage of the process. The study is exploratory in the way that I continuously gained new knowledge about what is happening, and generating new hypothesis throughout the study, both when observing new phenomena and finding relevant causal connections. Furthermore, the case is explanatory in the manner that every piece of data retrieved from the qualitative and quantitative research was sought for an explanation in the form of causal relation-

ship. The design of the case study can be described as a flexible case study design. Runeson & Höst [45] describe flexible design as a process where the key parameters of the study may change during the study, and that case studies typically have a flexible design.

Case studies provide a high level of conceptual validity due to it being easy to adapt to the intended context. This gives case studies an advantage over quantitative methods when it comes to separating small pieces of relevant data contra gathering large samples [46]. Case studies have a powerful way deriving new hypothesis. The researchers qualitative and personal connection to participants in the study, allows the participants to explain events or thoughts that were unforeseen for the researcher. For example: "Were you thinking X when you did Y?". "No, I was thinking Z" [46]. This form of deriving new hypothesis is very different from quantitative research and can provide valuable hypothesis within the relevant situation.

The way case studies examine individual cases in detail, allows for a deeper exploration of causal mechanisms. A case study allows researchers to identify many variables or conditions that must be present to activate the causal mechanism [46]. Case studies qualitative nature also supports modeling and assessing complex causal relations within the specific case. Combining case studies with process tracing evidence is required to document complex interactions [46].

3.1.2 Case Study Criticism

The use of case studies in software engineering can be criticized for several reasons. Case studies have criticized for being bias by the researcher, and impossible to generalize from [45]. The results generated from case studies are often very different from analytical and controlled experiments. This type of critique can be met with rethinking the importance of statistical significance up against deeper value of qualitative data understanding [45]. Allan [47] also emphasizes that case studies cannot generalize their findings, but Robert Yin [8] defends the position that case studies lead to theory building that is applicable for the world as a whole.

The case presented in this thesis shows a company in a distinctive case. It is possible to generate broader conclusions that are relevant beyond this specific case. The main types of generalizations that can be made from case study research includes, theory, concepts, implications and rich insight, or a combination of the aforementioned [44]. This generalization can be transfered to cases in similar contexts, with comparable factors. For example

cases where the organization's agile structure is similar.

Case studies are by many researchers seen as a "soft" research method. Researchers often confront the method by asking how to define the "case", how to determine what data is relevant, or what to do with the data [8]. This makes the researcher's job of filtrating the relevant data exceedingly important. Furthermore, cases are defined as a "thing" that is to be investigated, either an organization, system, decision etc. [44]. The researcher therefore has to define the case to such an extent, that it is well defined and understandable by anyone not involved in the process (further described in subsection 3.5.4).

3.2 Process Tracing

George and Bennet identifies advantages of combining process tracing with case studies with, which provides a valuable combination for testing hypothesis and developing theory in specific environments [46]. Process tracing is a qualitative analysis tool, which can contribute to describing social phenomena [48]. In order to help describe these phenomena, Beach et. al. [48] points to the importance of careful description of the process as the foundation of process tracing. Beach et al. [48] states that "Process tracing inherently analyzes trajectories of change and causation, but the analysis fails if the phenomena observed at each step in this trajectory are not adequately described" (p. 823). Combining process tracing with a case study therefore facilitates for a successful analysis of the observed phenomena, due to the qualitative nature of case studies. Runeson & Höst [45] argues that triangulation is important to increase precision of empirical research. By triangulation several forms of data, both qualitative and quantitative, one can obtain data from previously undiscovered areas. Blatter and Haverland [49] identifies process tracing as a within-case method that concentrates on specific mechanisms or processes of interest within the selected case.

Some of the advantages of using process tracing is according to Beach et al. [48]:

(a) "Identifying social phenomena and describing them,

(b) evaluating prior explanatory hypotheses, discovering new hypothesis, and assessing new causal claims, and

(c) gaining insight to causal mechanisms" (p. 824)

The process tracing approach in the thesis, will in accordance with Beach et. al.'s [48] points;

(a) identify social phenomena within the domain of agile software development, with focus on scalability and describing them.

(b) look at previous literature regarding scalability in agile projects and evaluate their hypotheses to look at their relevance towards the specific case. Discover new hypothesis, and assess new causal claims.

(c) understand the causal mechanisms with basis in the data collection and established literature.

The general process tracing approach is centered around "process induction" which involves observing apparent causal mechanisms and heuristic rendering as potential hypotheses for future testing [50]. This inductive approach attempts to map the different causal paths leading to scalability issues in large-scale development. The project's process induction can therefore enrich the case study approach in a way that leads to a collection of data that is thoroughly described and can thereby be examined for causal evidence.

3.2.1 Causal-Process Observations & Causal Inference

The observed phenomena in process tracing is often referred to as Causal-process observations (CPOs). CPOs are the diagnostic pieces of evidence examined in within-methods such as process tracing [51]. Seawright and Collier [49] explains CPOs as "an insight or piece of data that provides information about context, process or mechanism that contributes distinctive leverage in causal inference..." (p. 10). Blatter & Haverland [49] describes it as "A cluster of empirical information that is used to e.g. determine the temporal order in which causal factors work together to produce the outcome of interest..." (p. 10) Common for both definitions of CPOs, are that they explain the importance of a distinct piece or pieces of data the that can provide interesting information about the process or mechanism that can lead to a causal explanation of a phenomenon. CPOs are the cornerstone in process tracing, and are crucial to form the basis for drawing inferences about causality. To define a causal effect, one do not need to understand all the relevant CPOs or causal mechanisms involved, but one must define the concept of causal effect to identify the causal mechanisms [50].

In order to draw causal inferences about processes, one has to have sufficient evidence to able to provide a satisfactory explanation. The distinction be-

tween correlation and causality can be hard to separate when working with qualitative data since the researcher alone weighs the importance of each piece of evidence, hypothesis, or process. There is no clear divider for when a piece of evidence crosses the threshold from prediction to explanation. Bennet and George [50] explains that it is possible to have non-predictive explanations as well as non-explanatory predictions. Dion [50] goes as far as to saying that "case study methods do not require causal relations of necessity and sufficiency [...] case study methods offer stronger inferences on the existence of such relations than on the equifinality or probabilistic causality" (p. 13).

3.2.2 Process Tracing Test-Evaluation

Evaluating data with process tracing can be done according to four types of empirical tests, which evaluate evidence in different ways (displayed in figure 3.1) [51]. There are two criteria for establishing causal connection between events; whether passing the test is "necessary" or "sufficient" for establishing causal connection. Collier [51] argues that the criteria for necessity and/or sufficiency should be seen as heuristic standards for discussing evidence for causal evidence. Strong inference on existence of relations in case study methods leads to probabilistic causality, which is adequate, and means that causal relations of necessity and sufficiency is not required [50].

Process Tracing: Types of Tests

		Sufficient to Establish Causation	
		No	**Yes**
Necessary To Establish Causation	**No**	**1. Straw in the Wind Test** ***Passing***: Affirms relevance of hypothesis, but does not confirm it ***Failing***: Suggests hypothesis not relevant, but does not eliminate it ***Implication for rival hypotheses***: None	**3. Smoking Gun Test** ***Passing***: Confirms hypothesis ***Failing***: Does not eliminate it ***Implication for rival hypotheses***: None
	Yes	**2. Hoop Test** ***Passing***: Affirms relevance of hypothesis but does not confirm it ***Failing***: Eliminates it ***Implication for rival hypotheses***: None	**4. Doubly Decisive Test** ***Passing***: Confirms hypothesis and eliminates others ***Failing***: Falls short in establishing sufficiency and/or necessity ***Implication for rival hypotheses***: passing eliminates them, as noted above

Figure 3.1: Process tracing: Types of tests [51] (p. 3)

Using process tracing can according to Collier [51] contribute to creating a more complete picture of a systematized qualitative analysis.

Straw in the Wind Test

The Straw in the wind test is the weakest of the four tests explained in figure 3.1. The test demand the least of the researcher's knowledge and assumptions due to it not providing neither necessary nor sufficient criteria for rejecting or supporting a hypothesis [51]. The test itself is not decisive, but can help increase the plausibility or raise doubts about a hypothesis. The straw in the wind test is therefore a good basis for starting the process of finding causal connections.

Hoop Test

The hoop test does not provide direct support for a hypothesis, but can eliminate the hypothesis. The necessary criteria are met in the hoop test, but it does not provide sufficiency. It is called a hoop test because "a hypothesis must 'jump through the hoop' to remain under consideration, but success in passing a hoop test does not affirm the hypothesis" [51] (p. 5-6). The test allows the researcher to come closer to a causal connection between events due to its possibility to eliminate hypothesis.

Smoking Gun Test

The smoking gun test implies catching the suspect holding a smoking gun. The smoking gun test greatly support the hypothesis, but does not reject it if it fails. The test provide sufficiency, but not necessary confirmation criteria [51].

Double-Decisive Test

The double decisive test is the only test that confirm a hypothesis and eliminates all others. Collier [51] argues that for the test to work, the researcher must identify all other hypotheses, and eliminate them.

3.3 Congruence Case Study

Congruence case study method is in many ways similar to process tracing. It is a within-case method, where the goal is to produce a form of mechanistic evidence of causal process in a case [52]. While the two within-case methods are similar in many ways, one of the most important differences is that linking causes and outcomes, also called causal mechanisms, are not explicitly theorized [52]. This means that the understanding of causal mechanisms is seen minimalistic, and the causal process linking the cause to the outcome is cut off [52]. This leads to congruence cases producing within-case evidence without any explicit mechanisms being tracing [52]. This view on congruence case study is often represented by a "black-box", since we do not explicitly know about the entities and activities, as shown in figure 3.2.

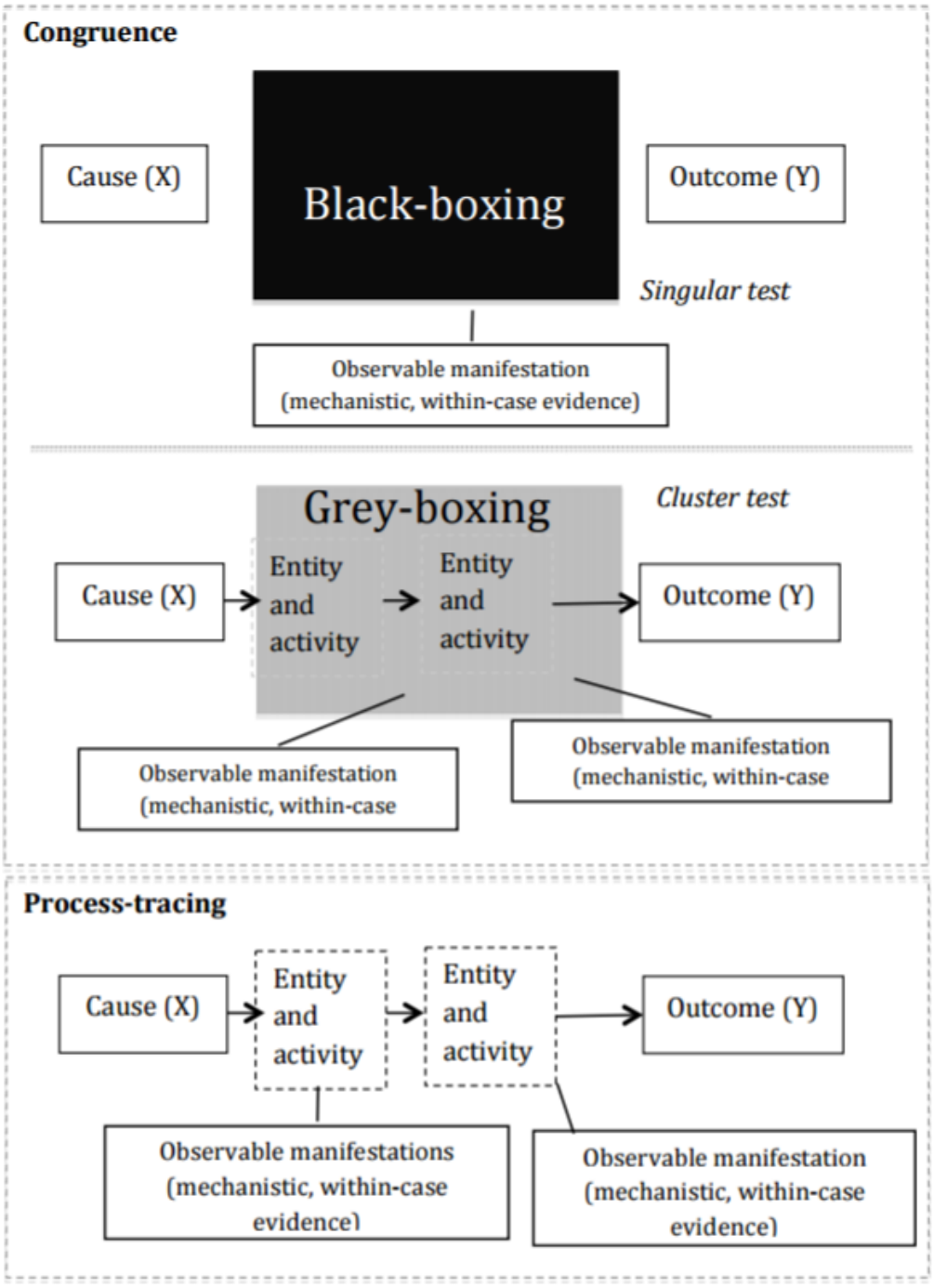

Figure 3.2: Mechanisms in congruence and process-tracing case studies [52]

George & Bennett describes congruence as a "very weak tool that only provides evidence of correlations across values of X and Y" [52] (p. 352). Further they state that:

> ...the investigator begins with a theory and then attempts to assess its ability to explain or predict the outcome in a particular case. The theory posits a relation between variance in the inde-

> pendent variable and variance in the dependent variable... The analyst first ascertains the value of the independent variable in the case at hand and then asks what prediction or expectation about the outcome of the dependent variable should follow from the theory. If the outcome of the case is consistent with the theory's prediction, the analyst can entertain the possibility that a causal relationship may exist [52] (p. 352).

This statement raises questions about if we learn anything by identifying casual relationships between X and Y. Beach & Pedersen [52] however state that knowing the value of both X and Y variables is important in order to select the appropriate case for a within-case analysis.

3.3.1 Congruence Analysis Inference

A congruence analysis approach provides empirical evidence for the relevance or strength of a theoretical approach, by analyzing the evidence in a case study [53]. Inference made using congruence analysis as a within-method produces either confirmatory or disconfirmatory claims about the existence or non-existence of a plausible causal relationship [52]. Compared to process tracing, congruence case studies provides evidence that is relatively weak. Process tracing relies on the confirmatory evidence to make strong inferences that X is causally related to Y through observable manifestations, as shown at the bottom of figure 3.2. Beach & Pedersen [52] states that congruence case studies on the other hand find predicted, and theoretically unique evidence. The mechanisms (black-box) are not traced explicitly, and therefore make a relatively weak inference regarding the causal relationship.

Evidence in congruence case studies overlaps with process tracing's definition of CPOs (see subsection 3.2.1). Causal inference in congruence case studies is separated by the above-mentioned differences. Congruence tests can be split in two, singular test, and cluster tests, as shown in figure 3.2. A singular test is a "single proposition about potential evidence is assessed multiple times during a temporal process or across space" [52] (p. 357). In the cluster test, "multiple non-overlapping propositions about evidence are assessed empirically" [52] (p. 357).

3.3.2 Explaining Outcome Congruence Studies

The term "explaining outcome" congruence aims to to account for why a particular outcome occurred [52]. The term can be used to assess causal relationships in cases without aiming to generalize the results beyond the case itself. Explaining outcome congruence cases can therefore be understood in a more holistic fashion [52].

> Cases are always too complicated to vindicate a single theory, so scholars who work in this tradition are likely to draw on a mélange of theoretical traditions in hopes of gaining greater purchase on the cases they care about [52] (p.359).

Congruence cases often have many causal conditions involved which give complex outcomes. Beach & Pedersen [52] state that scholars therefore often question the benefits of generalizing from the studied case to other cases.

3.4 Research Ethics

The company involved in this study, called Nihil, and actors involved in the study has been anonymized to preserve the employees' personal information, thoughts, and other sensitive data. Information about Nihil has also been anonymized to preserve the company's technical and business information, ideas, and other private or sensitive information. This means that some of the information in the thesis is on purpose, generalized, imprecise or otherwise undefined to preserve certain sensitive data.

The project is also reported to Norwegian Center for Research Data (NSD), and approved (shown in appendix B). NSD is the Data Protection Official for Research for all Norwegian universities, governed by the Ministry of Education and Research. Furthermore, the research was done in accordance with Oates' principles of research ethics in relation to the law and research, participants directly involved in the research, their right to; participate, withdraw, informed consent, anonymity, and confidentiality [44]. These points cover the most important aspects of the researcher's relationship regarding the participants and ethical research.

Informing participants, and giving them the ability to consent to the research, ensure them that they:

a) understand the purpose of the project, and their role in the project

b) can assess their own situation

c) can make an independent decision if they want to participate without outer pressure on the basis of information and own preferences and values

d) can freely communicate their decision [54]

Appendix A shows the information given to the participants engaged in the research.

3.5 Research Approach

The case study research has been does according to Yin's model (see figure 3.3) of how to conduct case study research, with usage of techniques described in section 3.5.4. How the results of the case study was achieved, is shown in the subsequent subsections: 3.5.1 - 3.5.6.

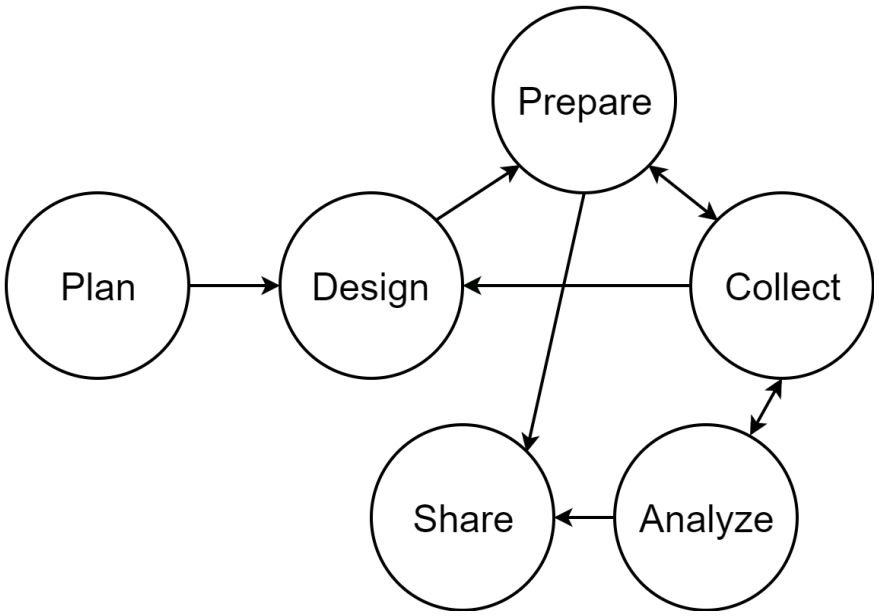

Figure 3.3: Doing case study research: A linear but iterative process [8]

3.5.1 Plan

The planning phase was spent planning how to conduct the field work, specifically, outlining a plan for how, where, and when to conduct the different data-generating events described in section 3.5.4.

Research Questions Planning how to conduct the work, is largely depended on the study's research questions. Case studies are suitable for answering research questions in form on how, and why.

The research questions aims to find out why the organization experience challenges with scaling Scrum (RQ1), and identify the characteristics which make large-scale Scrum difficult to execute (RQ2). The thesis addresses the questions from a organization theoretical viewpoint, which includes both formal and informal traits about the organization, as well as the people which make up the organization. This gives a holistic view of Scrum in large scale, as it includes several areas within the organization.

The research questions aim to evaluate alternative theories and build new ones. The goals is to examine all factors in the case, and see which pre-existing theory or models matches the findings in the case which may lead to new theories [44].

3.5.2 Design

Every type of empirical research has a research design. The research design is a plan for how to get from an initial set of questions to be answered, to some set of conclusion about the questions [8]. Every research design should include at least four problems: what questions to study, what data are relevant, what data to collect, and how to analyze the results [8].

According to Yin [8], there are five components of a research design that are especially important in case study research:

- a case study's questions;
- its propositions, if any;
- its unit(s) of analysis;
- the logic linking the data to the propositions; and
- the criteria for interpreting the findings

Study Questions To answer the study's questions, one have to chose a method that will answer the right questions. Case study research is often appropriate for answering "why" and "how" questions, such as the research questions, described in section 1.2.

Study Propositions Each proposition in the study directs attention to something that is to be examined within the scope of the study [8]. However, "why" and "how" questions may not sufficiently point to what the study is about. The thesis proposes that scaling the agile frameworks Scrum to multiple teams, is challenging. Previous literature suggests that this is a burning research question, with insufficient empirical knowledge, as explained in section 2.2.

Unit of Analysis A case study is the study of a string of processes and phenomenon. Defining the case is a fundamental problem in qualitative research, and is one of the things case study research often is criticized for [8].

The case examined in this thesis is a single-case study with Nihil as the organization to be studied. The study only examines some of the organization's departments, directly connected to their large-scale agile software development. Departments such as sales, finance, strategy etc. is not included in the study's scope. Furthermore, the study also only addresses people and teams related to the relevant product that is being developed.

Yin [8] points to the importance of bounding the case. These are other clarifications outside the general definitions explained in the aforementioned paragraph and subsection 3.5.4. Bounding the case helps determine the scope of the data collection, especially how to distinguish data about the subject from external data [8].

The context for the case study are people included in the development teams, such as developers, POs, SMs, and relevant members of management.

Linking the Data to Propositions The process of analyzing the data from the case is heavily dependent on how the data is linked to the propositions. The analysis require the researcher to combine and assemble the case data as a direct reflection of the study propositions [8]. This is done by following the process tracing method explained in section 3.2, and running temporary hypotheses through process tracing tests in an iterative process, further explained in section 4.2.

Criteria for Interpreting the Findings An important part of case studies is designing a strategy for identifying rival hypotheses to your findings [8]. The more rival hypotheses that have been addressed and rejected,

the stronger the found hypotheses are. It is important to think of this before the data collection has been completed, so that it becomes a part of the study's results, and not part of a further study [8].

In the initial phase of the case study, the design was considerably more exploratory than in the later phases. This allowed the me to explore areas that was not part of the initial research problem, but seemed interesting to pursue, either to confirm or reject.

Quality of Research Design

To establish the quality of empirical social research, four criteria (often referred to as tests) are commonly used. Yin [8] expresses that the tests (see table 3.1) deserve explicit attention due to their importance throughout the case study work. Therefore, case study research design may continue beyond the initial design phase.

Table 3.1: Case study tactics for four design tests [8]

Tests	Case Study Tactic	Phase of Research in which Tactic Occurs
Construct validity	• Use multiple sources of evidence	Data collection
	• Establish chain of evidence	Data collection
	• Have key informants review draft case study report	Composition
Internal validity	• Do pattern matching	Data analysis
	• Do explanation building	Data analysis
	• Address rival explanations	Data analysis
	• Use logic model	Data analysis
External validity	• Use theory in single-case studies	Research design
	• Use replication logic in multiple-case studies	Research design
Reliability	• Use case study protocol	Data collection
	• Develop case study database	Data collection

Construct Validity Construct validity refers to identifying the correct operational measures for the concepts being studied [8]. One of the main criticisms to case studies are according to Flyvberg and Ruddin [8], that "the researcher fails to develop a sufficient operational set of measure and that

subjective judgments - ones tending to confirm a researcher's preconceived notions - are used to collect the data" (p. 46).

The researcher needs to cover two steps, in order to meet the test of construct validity [8].

1. Define neighborhood change in terms of specific concepts (and relate them to the original objects of the study - see chapter 2)
2. Identify operational measures that match the concepts (preferably citing published studies that make the same matches - see section 2.2)

Internal Validity The Internal validity test seeks to establish a causal relationship. The investigator tries to explain how and why an event (x) leads to event (y). If the investigator fails to include all relevant factors e.g. (z), the research design have failed [8]. The process of distinguishing causal factors from false factors is only relevant in explanatory studies, since descriptive and exploratory studies are not concerned about causal relationships [8].

Furthermore, internal validity extends to making inference in case study research [8]. The research design needs to consider rival explanations, and analyze the evidence's convergence and degree of truth in order to explain the accuracy of the inference [8].

The research is designed to explore as many explanations as possible. This is done by attending and observing both formal meetings and other informal events. The research is exploratory organized for the initial phase, while transitioning over to an explanatory fashion when sufficient data is gathered about a phenomenon.

External Validity External validity tests handle the generalization of the study's findings. Yin [8] points to the importance of using theory or theoretical propositions to help generalize the lessons learned from the case study. The external validity of case studies are as mention in section 3.1.2, one of the main critiques against the method. The generalization of case studies can be split in to two categories: statistical, and analytical generalization.

In statistical generalization an inference is made about a population on the basis of the empirical data collected [8]. This method is often used in surveys, and is often less relevant when conducting case study research.

Analytical generalization may be based on either: modifying, corroborating, rejecting, or otherwise advancing theoretical concepts that is referenced in the case study design, or new concepts that arose upon the completion of your case study [8]. Either if generalizations are made from predefined conditions, or uncovered at the conclusion of the study, the generalizations will be on a conceptual level higher than the specific case [8], as shown in figure 3.4. This can help define new research focusing on similar situations in a similar context.

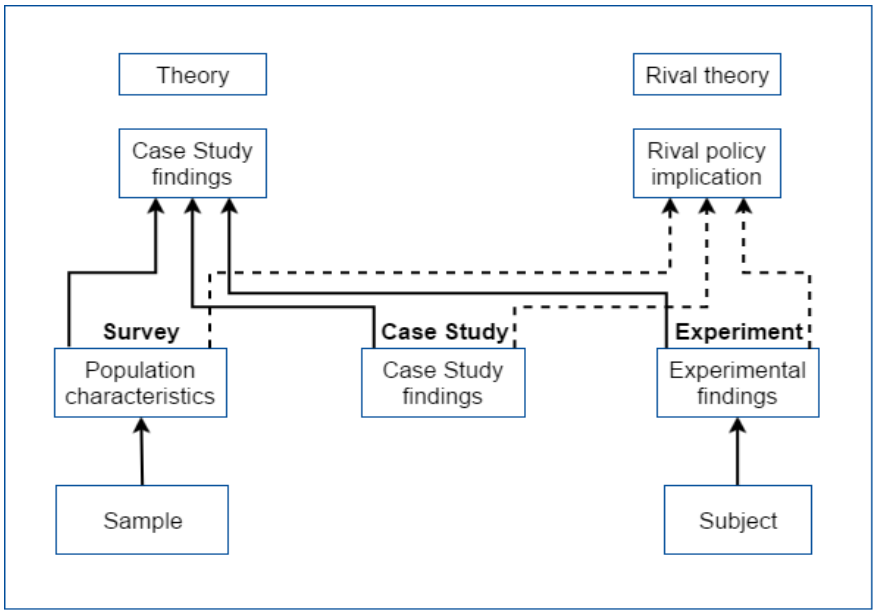

Figure 3.4: Making inference: Two levels [8] (p. 41)

The case presented in the thesis display a company within a distinctive context. The case is essentially not generalizable to areas outside the described context because of its distinctive characteristics. However, single elements can be forwarded on a higher conceptual level, and thereby be generalizable to some extent.

Reliability The goal of the reliability test is to minimize errors and biases in the study. This means that if a researcher would follow the same strategy, and conduct the same study, he or she should arrive at the same findings and conclusions [8]. Poor documentation is a factor that can weaken the reliability of the study, as is can not be reviewed. The suggested approaches to handle documentation problems are case study protocols and case study databases, as shown in table 3.1.

To minimize errors and biases in the study, observation-protocols were designed and used to document observation events with specific elements, such

as: date, time, type of activity, length of activity, location, participants and general observation notes. Interviews were written in a semi-structured fashion, and the general questions are replicable from the interview-guide, while follow-up questions and clarifications are documented in form of interview transcripts. By doing this, the study would arrive at approximately the same results if it were conducted by someone else. The most problematic feature regarding the research's reliability, is that the case would be hard to do over again, as there probably is a lot of specific features within the company which may not exist in other companies.

3.5.3 Prepare

The preparation of a case study is a complex task which takes into account challenges such as, gaining approval for the study, and show how human subjects will be protected [8]. The preparation can according to Yin [8] be split into five points:

1. Desired skills and values
2. Training for a specific case
3. Developing a protocol for the study
4. Screening candidate cases
5. Conducting a pilot case study

Desired Skills and Values Case study research is a demanding tasks that requires a large set of skills and values. Yin [8] states that the demands of a case study on the researchers intellect, ego, and emotions are far greater than any other research method. Several elements skills are involved when conducting case studies, such as ethical dilemmas, technical aspects of the data collection, and mediating continuous interaction between theoretical issues and the data collection, which require making a lot of judgment calls [8].

I am new to case study research, and the skills required to conduct the study was therefore accordingly. The most helpful process in preparing and acquiring these skills, was reading other case studies, case study protocols, and talking to others who had done it before.

Training for a Specific Case The goal of training for a specific case, is to understand 1) Why the case study is being done, 2) What evidence

is being sought, 3) What procedural variations can be anticipated, and 4) What would constitute supportive or contrary evidence for any given proposition [8].

In the training for the case, I revealed several flaws in the case study design. In the initial phase of the project, it became apparent that the research questions were too broad to figure out in a relatively small project, such as a masters thesis. Furthermore, the plan was to interview more people than anticipated. Interviewing and transcribing appeared to be a more extensive task than anticipated, and the number of interviewees was reduced from ca fifteen to five.

Developing a Protocol for the Study A case study protocol is a document describing the case which is to be studied. A case study protocol should include the following sections [8]

- **A:** An overview of the case study including objectives, issues, and relevant literature about the selected topic
- **B:** Data collection procedures for protecting human subjects, identification, probable sources of data and logistic reminders
- **C:** Data collection questions that must be kept in mind when collecting the data
- **D:** A guide for the case study report, with outline, format, presentation, etc.

The case study protocol used for this project was a document which was gradually transfered, rephrased into the thesis, as it corresponds with a master thesis regarding content and structure.

Screening Candidate Cases The process of screening candidates for the case study aims to make sure that the final candidates are identified, prior to the formal data collection [8]. This was a fairly straightforward process, because of the exploratory initial phase, with a lot of observation gave a good indication of which candidates could contribute concerning large-scale Scrum. For example, persons with inter-team responsibilities within the organization had more knowledge about recurring issues, hence would often be more suitable candidates for e.g. interviews.

Conducting a Pilot Case Study A pilot case study is a test of the case study, which helps the researcher to refine the data collection plans [8]. A

pilot case study was not conducted prior to the case study because of the limited time frame available.

3.5.4 Data Collection

To improve our collective knowledge about software engineering, we need to gather data about the field. Software engineering is an extremely people-oriented activity and in order to gain knowledge about how the people work, it is essential to conduct studies on real practitioners [55].

Table 3.2 shows an excerpt of the data collection techniques used in the thesis' field work with their goal, volume of data gathered from the technique and alternative use.

Table 3.2: Data gathering techniques [55]

Technique	Used by researchers when their goal is to understand:	Volume of data	Also used by software engineers for
Direct techniques Interviews and questionnaires	General information (including opinions) about process, product, personal knowledge etc.	Small to large	Requirements and evaluation
Shadowing and observation	Time spent of frequent of tasks (intermittent over relatively short periods), patterns of activities, some goals and rationale	Small	Advanced approaches to use case or task analysis
Indirect techniques Fly on the wall	Time spent intermittently in one location, patterns of activities (particularly collaboration)	Medium	
Independent techniques Analysis of work databases	Long-term patterns relating to software evolution, faults etc.	Large	Metrics gathering
Documentation analysis	Design and documentation practices, general understanding	Medium	Reverse engineering

Nihil Case

For this single case study, a company called Nihil was cooperating with the researcher. Nihil conducted large-scale Scrum on a daily basis and was interested in investigating it for improvement potential themselves. The cooperation was aimed at understanding how the company applied large-scale Scrum in practice, and look for case-specific peculiarities in relation to e.g. previously researched studies. By doing this, I could be able to identify causal connections to why scaling Scrum can be challenging.

Nihil is a cloud service business primarily located in western Europe. The company develops, operates, and maintains cloud services for users all over the world. The company has several million users worldwide. Nihil's management and some of the development is located in Western Europe, while other parts of the development is located in Asia, Eastern Europe, and North America. The business is therefore affected by a diverging social mixture from different organizational cultures.

The organization primarily uses Scrum as a management framework for developing software in the organization. Some of the development is based on lean development methods such as Kanban, but is affected by the organization's requirements regarding structure, and therefore have additional events to ensure traceable processes and management overview among other things.

The examined case's context in this thesis was a continuation of an existing and ongoing development project. Nihil's current project was mainly focused on developing new features and adapt the product to both new and existing users, as well as maintaining the product. The business has worked with agile methods for about ten years, and the agile methods are well incorporated in the business. Nihil can therefore be seen as an experienced agile business, and are will therefore not be afflicted by regular adaptation problems with adopting agile methods such as, organizational resistance, management apathy, and inadequate training regarding agile methods [56].

The cooperation between the researcher and Nihil started in August 2016, and ended May 2017. The data gathering process described in the subsequent sections and subsections, took place between October 2016 - March 2017. During the process, six formal interviews interviews took place (fur-

ther described in section 3.5.4), as well as 14 structured observations events and numerous unstructured observations (described in 3.5.4). Documents were also found during the full duration of the thesis work (August 2016 - May 2017). The study can be described as a short-term, contemporary study, which examines events that occur in the present [44]. The time limit is one of the boundaries for the project. Yin [8] suggests the time and geographical limits are some of the main characteristics that describe a case's degree of completion.

Observation

Observation is the process of watching and paying attention to something in order to analyze, form theories, make influence or impose meaning [44]. Passive participatory observation was used to collect data about the Nihil's agile process. Participatory observation is an approach where the researcher is involved in a social setting for a limited amount to time to observe different phenomena relevant for the study [57]. The level of interaction stretches from fully active to fully passive, and refers to the degree of involvement the researcher has in the observation. Passive observation means that the researcher has a bystander role and does not involve himself in e.g discussions. The observation-process in the the case, can be specified as overt observation, meaning that the participants know that they are being watched. The case study included both participatory observation and non-participatory observation, based on what event was ongoing. Typically, formal events such as Retrospectives, Sprint reviews and Scrum of Scrum were passively observed. Informal activities such as development, lunch breaks and "office chatter" was part of the participatory observation.

"Fly on the wall" is a technique where the researcher is an observer without being physically present [55]. The "fly on the wall"-technique was used on some remote Scrum meetings. The main advantage of this technique, is that it requires very little time from the participants. However, it requires much time to analyze the data.

One advantage of being a passive observer rather than an active is that it is easier to achieve objectivity and avoiding bias and influencing the participants in the study. By conducting the observation in a passive fashion, the observed group acted in its natural environment and could promote their meaning and attitude in a familiar environment. The observation-sessions in the case study reached from unstructured to semi-structured, based on if there was a underlying hypothesis that basis for the observation, or the

sessions were of a more exploratory and thereby unstructured fashion.

Some of the observation-sessions were in the form of shadowing. When shadowing, the researcher follows the participant around and records their activities. The main difference between shadowing and observation is that the researcher shadows one person at a time, but in observation, one can observe many simultaneously [55]. Both the shadowing and observation sessions were documented in an observation-protocol with details of where, who, when, and what was going on during the session.

Interview

As mentioned earlier, interviewing is a common way of gathering qualitative data about participants in a case study. Interviews are adaptable in many contexts due to its flexible design. Interviews can be tailored to fit the study's design and can be either structured, unstructured, or in between. The aim of qualitative interviewing is to evoke the respondent's information in form of their behavior, attitude, norms, beliefs, and values [57]. Semi-structured interviews were conducted in this study. Semi-structured interviews are designed to allow the interviewer to follow a general interview plan, but still have the opportunity to ask follow-up questions as new information is learned during the interview [57].

One of the disadvantages with interviews are that they are cost inefficient [55]. The researcher needs to schedule the meeting, and attend it. Furthermore, the data from interviews needs to be transcribed when it is recorded. Additionally the participants' reports of events may not reflect the reality [55]. This is why it is an advantage to gather information of different sources, to gain an overview of case. By doing this, the researcher can easier understand if the participant is telling the truth, even though there is no guarantee.

Documents

Documents can be an alternative source of data in addition to interviews and observation. Oates divides documents into two types: found documents and researcher-generated documents [44]. Found documents are the relevant documents that exists prior to the research. Researcher-generated documents are documents that would not have existed if it was not for the research task. The documents are organized and evaluated, and are used

by the researcher as source of data in the analysis phase. Examples of this are field notes, models and diagrams [44].

The documents used in this thesis' research are a combination of found- and researcher-generated documents from the case research at Nihil. The found documents primarily consists of documents related to the Scrum process, such as Scrum Boards, burndown charts, Scrum review notes, and retrospective notes, as well as the organization's wiki-page. Researcher-generated documents such as notes were generated from attending meetings, interviews, general observations and conversations with employees.

3.5.5 Analyze

There is no "recipe" for analyzing data derived from case studies. The analysis of case study evidence is according to Yin [8], one of the least developed aspects of doing case studies. Yin [8] suggests four general strategies for analyzing a case study.

- Relying on theoretical propositions
- Working the data from the "ground up"
- Developing a case description
- Examining plausible rival explanations

The original objectives and design of the case study was based on theoretical propositions explained in section 3.5.2. This formed the case study, and helped lay the theoretical basis for the case study analysis. Yin [8] points to that this strategy further reflects on the case's research questions, reviews of the literature, and new hypotheses or propositions.

Working with the data from the "ground up" is an inductive strategy which can occur by "playing with the data" [8]. This strategy is useful for investigating unexplored data for pieces of evidence. Yin [8] proposes that this strategy can be the start of an analytical path, leading the researcher towards possible unexplored areas of the data. This was done as an initial part of the analysis (further explained in section 4.2).

Developing a case description is an alternative research strategy, if the above-mentioned strategies are not used. The strategy aims to organize the case study according to some descriptive framework [8].

Examining plausible rival explanations is a strategy that can be combined with the three aforementioned strategies.

> The typical hypothesis in an evaluation is that the observed outcomes are the result of a planned intervention. The simple or direct rival explanation would be that observed outcomes were in fact the result of some other influence besides the planned intervention and that the investment of resources into the intervention mat not actually have been needed [8] (p. 140).

Examining plausible rival explanations is an essential part of working with process tracing. Testing rival hypotheses (explanations) is done by investigating causal connections between events and evaluation its necessity and sufficiency, explained in section 3.2.2. The more rival hypotheses that are addressed and rejected, the stronger are the researcher's findings [8].

3.5.6 Share

The case study is written in a linear-analytic structure, meaning the sequence of subtopics start with the issue being studied, followed by literature, methods, data collected, analysis and findings, ending with a conclusion [8].

Sharing the data from a case can be quite sensitive. There are a lot of factors to consider, since they might identify the case or its subjects. Runeson & Höst [45] state that the researcher must find a balance between what to share and not, and foremost how to share it, in order to avoid identification (see section 3.4)

Potential Audiences

The audience of a master thesis is fairly wide (see section 1.3). Case studies have according to Yin [8] more potential audiences than other types of research. Each audience has a different need, and no report will satisfy all audiences to the fullest extent. This leads to the thesis having several sections, with a different degree of explanations, theory, and real-world descriptions, to serve the different audiences.

Chapter 4

Results

This chapter describes the case's context, based on the relevant contextual factors regarding the results. The chapter explains how the hypotheses were generated. Further, the chapter describes each hypotheses in detail with basis in the data collection combined with relevant literature, and is thereafter summarized.

4.1 Nihil Case Context

The studied organization in the case, is called Nihil. As mentioned in subsection 3.5.4, Nihil is a commercial actor, which performs large-scale Scrum development in a distributed setting, extending over several geographical sites. This means that the organization needs to be organized accordingly.

Nihil organize teams based on the project's specific feature-areas. This means that each team work with a high-level area of development e.g. user experience, test tools, deployment etc. The teams each have one PO and SM, while each PO and SM are responsible for multiple teams, making the ratio unbalanced (further explained in subsection 4.3.1). Most teams are traditionally organized in terms of size. Alpha team is the only team the investigated section of the organization which is unconventionally organized by having eighteen developers. The teams that are located at the same sites are mainly co-located in terms of sitting next to each other, with some exceptions. This does not include POs and SMs, as they are incorporated in multiple teams.

Nihil is experienced in terms of conducting agile development, as they have conducted agile development for more than ten years. Nihil performs all the

traditional Scrum events with very few deviations from the Scrum guide (explained in subsection 2.1.1). The different sprints' duration are decided by each team based on how much work they contain, and how much time each team needs. The teams coordinate using SoS. In addition, they maintain formal communication in standard Scrum meetings and additional irregular events, subsequently explained in hypothesis 4.3.1.

4.2 Generating the Hypotheses

The within-method process tracing focuses on tracing links between possible causes and observed outcomes [46]. The researcher analyzes documents, interviews, observations and other pieces of data to see if he can establish a probable hypothesis. Thereafter, the hypothesis is evaluated using process tracing tests to determine the strength of their causal inference. It is important to underline that the tests are used heuristically, and should not be taken rigidly [48]. The process is extensive and demands high level of insight to the data-material.

The process of generating hypotheses was a exhaustive process that took basis in both relevant literature and the data gathered during the case study. The hypotheses-generating process can be classified as iterative. A hypothesis was initially explored by either being observed in the case, proposed by other authors within the field, or by interviewing or talking to Nihil employees. From the initial hypotheses was thought of, until it was fully explored, it underwent a series of tests to explore its relevance and strength. To try to achieve sufficient data to establish the hypotheses' causal inference, the hypotheses was re-explored and re-tested through observation and interviews with different people with different points of view. For example, the researcher could have had assumptions that a hypothesis was established with a strong causation related to a phenomenon, only to later on be refuted by a subsequent hypotheses. This underlines the importance of recurrently revisiting previous hypotheses in qualitative case work, thereby making the process iterative.

The iterative work with the hypotheses in this study is affected by the lack of observable causal mechanisms, in form of sequential events. This lead to a minimalistic understanding of the causal processes linking cause to outcome. This kind of "black-box" mechanisms (see figure 3.2) leads to a less certainty of causal relationships, because the possibility of lacking some variables, as explained in section 3.3. The results of this circumstance is

further discussed in chapter 5.

In the case explained in this thesis, the following hypotheses were generated.

Table 4.1: Hypotheses (Subhypotheses are indented and marked with a "•").

Hypotheses
Nihil's Scrum structure forms coordination issues
• in multiple-team organization
• in single-team organization
Communication distances in Nihil create a lack of individual team members' project understanding
Rigid processes in Nihil impair agility

4.2.1 Concept Operationalization

This subsection aims to clarify some of the concepts that are not directly measurable on their own. The goal is to make the concepts understandable in their hypothesis' context.

Coordination Issues Coordination issues refers to the problem of organizing different elements or activities to enable them to work together. Coordination issues in multiple-team organization primarily addresses issues regarding activities in an inter-team coordination context (both team-to-team, and individual-to-team). Single-team coordination issues refers to the problem of organizing activities and/or elements within a single team.

Project Understanding Project understanding involve individuals' perception of a project, or smaller parts of the project such as isolated processes. Communication is tightly connected to the involved individuals' understanding of the project.

Rigid Processes Rigid processes are opposites of the values of agile development. Rigidness refers to an organizational or development process being too inflexible to achieve the desired outcome (explained in section 2.1).

4.3 Hypotheses

4.3.1 Nihil's Scrum Structure Forms Coordination Issues

Challenges regarding Nihil's Scrum structure can be divided into two parts: Multiple-team organization and Single-team organization. Scrum structure refers to how Scrum events and team(s) are structured and carried out.

Multiple-team organization

Nihil's Scrum development is organized in a way that is customized for efficient large-scale development. The development teams are relatively traditionally organized, but Product owners (PO) and Scrum Masters (SM) are responsible for multiple teams simultaneously as shown in figure 4.1. The organization's POs are or have been responsible for two to three Scrum teams at the same time. CPOs from the study indicate that POs that are responsible for three Scrum teams, have issues handling the different projects.

People involved in the different projects, with the same PO, indicate that the PO's level of involvement on the different projects are varying due to factors such as: team size, previous PO's product understanding, the scope of the team's responsibility-area, the team's skills.

> Having a product owner responsible for just one team, is too little, three is too much, two is just right. The exception is Alpha team, where one product owner would have more than enough to do. - Nihil employee

Alpha team's size is much larger than Nihil's other Scrum teams, with over the recommended size of three to nine people, this is why the employee suggests that one PO is enough (further explained in subsection 4.3.1 - Single-team organization).

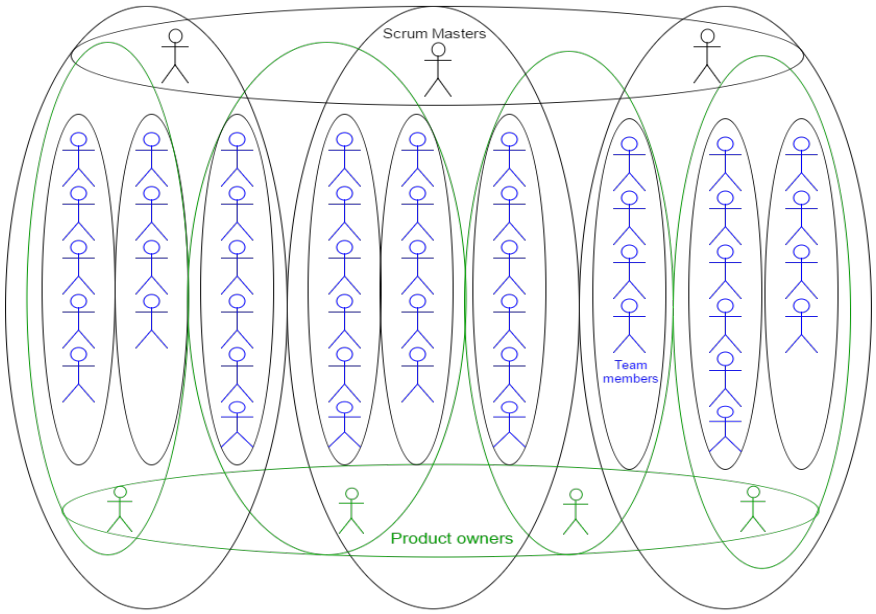

Figure 4.1: Nihil's team organization

Having a single PO responsible for multiple teams makes communication channels more complex. Nihil has implemented team-leads for every team. Team-lead is not an official Scrum role, but serves a function as a mediator between the PO and the rest of the team. With POs being responsible for multiple teams, they have less time per team, which often lead to use of alternative communication channels, providing poorer information (see hypothesis 4.3.2).

At Nihil, SMs are in the same way as POs responsible of multiple teams. The main difference is that they are generally responsible for even more teams. The different teams' SMs, and POs, have a lot of differences between responsibility-areas and assignments. Case data show that SMs deviate in the their perception of how the role is supposed to be carried out. Some SMs involve themselves in how the product is made, instead of carrying out their coach role. Some are more or less absent from their team, while others are mostly interested in documenting the process. One would think SMs had the same problems as POs regarding coordination, but this has not shown to be a prominent factor in the study.

To coordinate the POs' assignments, thoughts, ideas, and shared organizational understanding, the POs meet once a week to discuss this in joint meeting, which is not part of "official Scrum". The meeting reminds of a Scrum of Scrum (SoS) meeting where all parties gives a high-level status update of their progress, and is included in what the other teams are doing. This means that the meeting is affected by both the positive and negative effects of implementing such an extra event. The meeting helps the different

POs gain an understanding of the other teams current work, issues, future plan etc. This event weakens the hypothesis' strength, as its structure helps coordination between POs.

In order to handle coordination issues between teams, Nihil uses SoS meetings. SoS is a technique where a representative for each team gathers in form of a meeting and discusses their work. The questions answered by each team is in standard SoS manner; 1) What have you done since last sprint?, 2) Are you about to put something in another team's way?, 3) Is anything slowing your team down or getting in their way?, and 4) What will your team do before we meet again? Some teams, often remotely located teams, attend with their whole team, while team-lead for each team located at HQ attend the SoS meetings to speak on behalf of the team. SoSs are carried out with up to 26 teams, reporting their progress and future path.

The use of SoS in Nihil is useful in varying degree based which individual is attending, their role, or their team's role in the organization. Results from Paarisvaara et al. [7] work, show that audience in SoS meetings can be too wide to keep the involved parties interested. This is also reflected in the Nihil case, both in SoS, and weekly PO meetings. Meeting attendees who are invested in the development, show a lot greater interest in "lower level"-issues related to feature development, and do not pay much attention to "high level" and underlying goals. However, there are those who are invested in multiple teams or are part of the organizations management. This side is divided between wanting to know about both high-level goals, and smaller product features.

> Regarding Scrum of Scrum meetings, it is sometimes nice to know what the organization is working with. Nevertheless, most of the time, the teams work on the same feature or area, which can be totally different from the what your team is working with. This is not always so useful, and you can sometimes see people being bored at the meetings, playing cards etc. - SoS-attendee

A suggestion to solve the issue regarding a varying audience in SoS meetings, was Scrum of Scrum of Scrum meetings (SoSoS), which was suggested by Cohen [58]. The concept of SoSoS is based on having multiple SoS meetings, where each SoS sends one representative to a higher level meeting, as shown in figure 4.2. The thought is to reduce size of the SoS meetings, to keep all attendees interested.

CoPs is a another technique which can also tackle similar large-scale issues [28]. In CoPs, teams working on the same feature work together in the

meeting, and it is therefore easier to stimulate the attendees interests, as they are likely to involved in the topic.

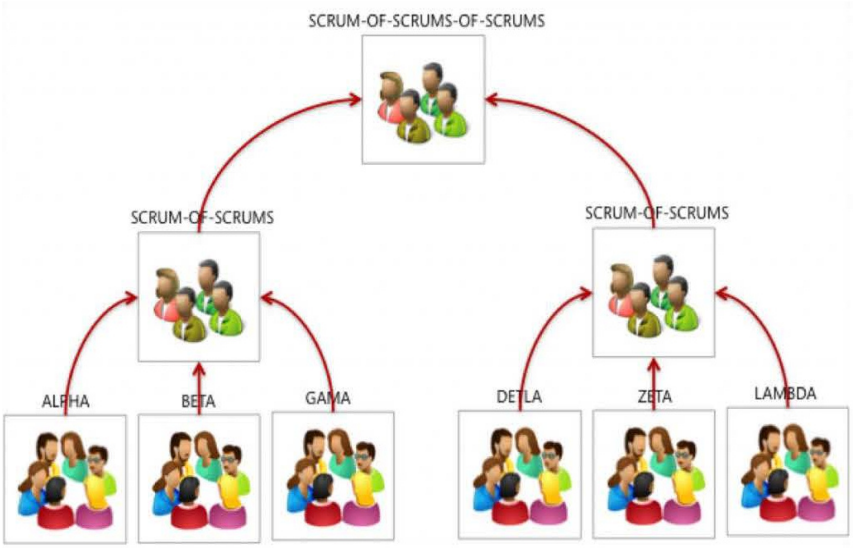

Figure 4.2: Scrum of Scrum of Scrums [59]

Single-team organization

The different teams in Nihil are organized in different ways. Some teams have only three developers, while other teams, such as Alpha team, have up to eighteen developers in the same team. Having more than nine developers in the same team is not recommended because it is a high chance of coordination issues within the team [15]. Because of Alpha team's area complexity, the team was also distributed over five locations in three continents.

CPOs from the study show that traditional Scrum events did not fit the team's development style. The team had problems delivering shippable increments within the sprints' timebox. This, among other minor factors lead to the decision that Alpha team changed development method from Scrum to Kanban. Kanban removes the traditional timeboxed events that you find in Scrum, to a continuous work-flow. Kniberg [18] encourages this kind of experimental behavior in order to customize the agile process to your environment.

> Scrum and Kanban are both empirical in the sense that you are expected to experiment with the process and customize it to your environment. In fact, you have to experiment. Neither Scrum nor Kanban provide all the answers – they just give you

> a basic set of constraints to drive your own process improvement [18] (p. 17).

The tactic of changing to Kanban seems sensible based on research, which reports that enhanced visual control that facilitates and supports the decision-making process is increased with 45,9%, as well as 29.7% of Kanban users reduce the lead time of tasks, as shown in table 2.1. David Anderson [17] stated that "Kanban is neither a software development life cycle nor a project management methodology; instead, Kanban is used to incrementally improve an existing process" (p. 1881).

The transition from Scrum to Kanban was nevertheless not fully complete. Nihil require their teams to conduct events from Scrum, such as reviews and retrospectives as well as estimation of backlog items. The development method of Alpha team can therefore be described as Scrumban (explained in subsection 2.1.3), a mixture between Scrum and Kanban. The researcher did not have time to fully follow the transition to Kanban, as the data collection process ended, and could have been explored further in detail.

> I think it's more important that you have a framework that facilitates for flexibility, where you can make changes during the process. This is much more important than using a framework, just to use it. - Alpha team member

Wang et al. [22] states that Scrumban is especially suited for projects with unexpected user stories, maintenance projects, or projects prone to programming errors. One of the main advantages of Scrumban, according to Wang et al. [22], is the ability to exclude time-boxed sprint, but still keep other Scrum events such as daily scrum, retrospectives etc.

It turns out however, that at an earlier point of time, several teams in Nihil used to perform Kanban development. Some teams teams switched over to Scrum due to Kanban being too disorganized, and Scrum was supposed to help "clean up" the backlog by having stricter frames and guidelines. Other teams switched because they simply did not manage to operate with Kanban as a development method, due to the method's lack of guidelines. However, switching to Scrum did not work for several of the teams.

Both the Scrum and Kanban methods are self-learned by most of the employees at Nihil. Employees in the organization state that they have learned the methods by reading literature, and/or participating in the organization's existing Scrum process.

Hypothesis Summary

The diagnostic pieces of evidence from the Nihil case show that their Scrum structure affects the scalability of Scrum in large-scale. The CPOs in the case study clearly show that the organization of multiple-team can be improved. POs have to allocate their time based on which team needs their help the most. This can lead to other teams getting reduced help from their PO, and can lead of lower efficiency in the team. However, a single PO per team turns out to be inefficient time commitment, as there is not enough work for a single PO.

Nihil's use of SoS also has room for improvement in their large-scale Scrum structure. The audience in the meetings proves to often be thematically uninterested in the meetings mostly due to too large variety in content. A change of SoS structure could help improve this issue, possibly in form of suggested techniques such as SoSoS, or CoP.

Some of the teams in Nihil have trouble adapting to, and making Scrum work. Scrumban was supposed to be the answer to this issue. Even though experimental behavior can help with the process, it had for other teams failed at an earlier point in time. Furthermore, it is unsure if the change of method was successful, as the data collection ended before the researcher could see the effects of Kanban in use.

4.3.2 Communication Distances in Nihil Create a Lack of Individual Team Members' Project Understanding

Communication is at the heart of agile development. The first value of the agile manifesto promotes individuals and interactions over processes and tools. However, communication is not an easy task to accomplish perfectly when scaling up Scrum.

> Communication is the most important when we are as spread out as we are. It can always be improved, and I don't think we are where we should be. - Nihil management employee

Nihil uses several different communication tools and forums to distribute information, and communicate both privately, and in community. In total, Nihil uses twelve digital communication tools, and ten forums in form of meetings, seminars, and group sessions. The different tools and forums are used and work to different degrees. Some of them are nearly never

used, while some are used every day. Multiple CPOs at Nihil shows that communication is critical for the development to go smoothly.

> In large development situations [...], a mismatch of adequate communication mechanisms can sometimes even hinder the communication [13] (p. 303).

Bannerman et al. [32] distinguishes between three types of distance: temporal, geographical, and socio-cultural. These distances can offer challenges, especially regarding communication, and need to be overcome for agile software development on a global scale to be realized [32].

Temporal Distance

> Temporal distance, a measure of the dislocation in time between two people wishing to interact, may create communication issues such as reduced hours of collaboration, difficulties in holding synchronous meetings, and response delays [32] (p. 5310).

Nihil operates in three different continents, and is therefore affected by temporal distances within the organization. Based on the field work done in the case study, I was not able to find any factors that suggested temporal distance was an evident issue. The field work did not extend beyond one location, and the data was in many cases dependent secondary sources. The few relevant CPOs from the study show that the temporal distance between individuals in the organization was easily worked around by performing recurring scheduled meetings weekly, where relevant topics were reported.

Geographical Distance

> Geographic distance, a measure of the effort required to visit another person's home site, makes communication difficult because of the reduced ability to hold face-to-face meetings [32] (p. 5310).

Distribution of IT development is not a new phenomenon. Distribution of software development refers to people developing software at different geographical location, either different cities, countries, or event continents. Greater distances between peoples often come with challenges. Bannerman et al. [32] state that lack of face-to-face meetings reduces informal communication, which can lead to lack of critical task awareness, "teamness", and reduced trust. Loss of direct contact to all levels of management also

leads to demoralizing bad practices, which is likely to occur in distributed settings [60].

A recognized problem at Nihil's management, is the lack of informal communication between employees not located at the same geographical location. One of the greatest challenges in distributed teams is according to Šteinberga & Šmite [60], creating and maintaining a relationship and a sense of belonging between team members. The culture where people casually meet in the hallway or at "the watercooler" is an important factor for casually distributing information across the organization. Employees at Nihil who are not located at the headquarter (HQ) struggle with this issue. This issue may in worst case result in absence from work, instability at the workplace, and poor work relations [42].

> People who sit at the HQ experience the casual information, where e.g. a person asks; - "Have you seen the new transparency models?" - "No, i have not, what it that?". People who sit outside the HQ experience this issue, especially those who sit alone, for example at a home office. - Nihil employee

In a response to this issue, Nihil tries to create points of contact, in form of weekly and biweekly meetings, where people can receive this kind of informal or casual information. This response helps create a culture where the employees feel a belonging and fellowship to the organization, but will not replace the benefits of face-to-face communication [42].

Geographical distribution can also lead to relational problems. Being geographically distributed involve practical issues related to meeting each other and forming a personal relationships. Having a good relation to other team members is important in agile development. The core values in Scrum are commitment, courage, focus, openness and respect [15]. Several people at Nihil state that they have not physically met everyone that they would have wanted to meet, they have only "met" via video-conferences or similar communication channels. Most employees use Skype as their main communication tool, especially when communication with remote team members. Several employees promote its instant messaging functionality as the main advantage, providing them with the ability to make decisions fast. Others feel that instant message services such as Skype, becomes a time thief and a stress factor because it allows you to see the whole team's discussions. Furthermore, employees with roles who require a holistic overview of the development process, such as POs, express that they miss meeting their team members face-to-face. Their main reason for this is so that they can more easily understand peoples opinions and values via rich interactions

such as body language and gestures. Forming an impression and building a relationship of their colleagues is valued among the employees, as it forms a relationship built on safety and respect.

Socio-cultural Distance

> Socio- cultural distance, a measure of a person's understanding of another's values and normative practices, may create issues relating to inconsistent work practices, different perceptions of authority, and lack of mechanisms for creating shared understanding and avoiding misunderstandings and reduced cooperation [32] (p. 5310).

Large scale agile organizations are often affected by socio-cultural distance due to the organizations' size and complexity. Nihil's organization culture fits into the differentiation perspective on subcultures. The cultural manifestations is interpreted differently by the different groups in the organization, which therefore creates subcultures which exists side by side in the organization [42], as show in figure 4.3. Even though different subcultures can interpret things differently, Jacobsen & Thorsvik [42] points out that they can work independently, and live live side by side without conflict.

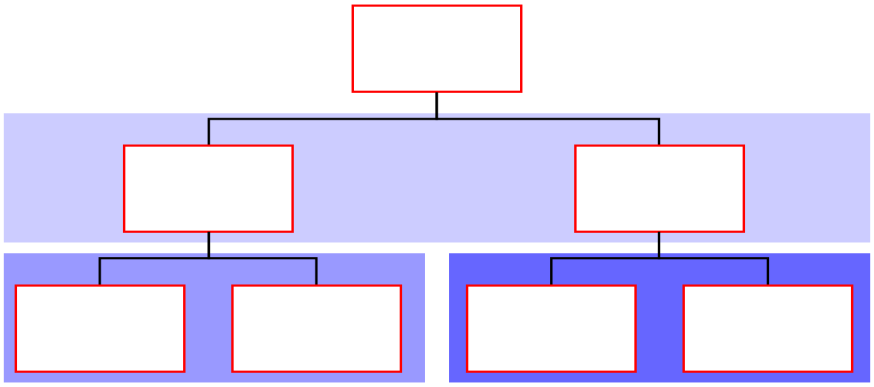

Figure 4.3: Differentiation perspective [42] (p. 143)

The different subcultures within Nihil comes to light when comparing teams across the different locations. CPOs show that teams at different locations act in various ways regarding their tasks. For example, teams located in North America require accurate information, preferably from different sources in order to work accordingly to what the central management wants. If the information is uncertain or undercommunicated, they might interpret things in their own way and pursue a path in different direction than the management wants. Central European teams on the other hand, have a tendency to cling on to information before passing it on to other. The teams

try to perfect the information, which can be viewed as a taking pride in their work. Ramesh et al. [31] points out that managers should perform fine-grained adaptation of development methods, in order to adapt it to the cultural context in which they are used to. A more gradual adaptation of Scrum to other locations, with frequent coaching could thus have helped reduce the cultural differences within the organization.

This kind of difference in behavior can be seen as different organizational subcultures. However, it is not certain, and can be affected by other factors such as the type of the work assignments, responsibility area, or other topics which has not been explored by in the research due to the thesis' scope and limitations.

Poor communication often affect the inter-team coordination. For example, when a team makes a decision, which at the time seems like a good idea, the product is lead in one direction. At the same time, another team makes a different decision, which leads the product a different direction. This makes teams suboptimize their work, which could be solved with improved communication.

Language is another factor which can inhibit effective communication. Nihils work language is English, which everyone understands, but to various degrees. The main problem with language is not that the employees do not understand the language, but that nuances are lost in the translation. These types of minor language barriers are common, especially in distributed agile projects [61].

SoS meetings are one of the events which illuminates inter-team communication issues. The events intercept some of the communication problems within the organization, but should not be the only technique, as they can often be the last formal inter-team communication step before the teams start working on the task.

> I have for example seen that different teams have a differing opinion of something, even though they have been in the same SoS meeting, with the same PO. We can never be good enough when it comes to communication - SoS attendee

Clarifying and eliminating communication errors before starting the development can save organization a lot of time by eliminating duplicate work or having to change something later in the process.

Hypothesis Summary

Communication is one of the processes that has a lot of improvement potential at Nihil. Central employees in the organization state that communication should be improved, and that communication can hinder project effectiveness.

Nihil uses a lot of different tools and forums with hopes to achieve good communication, both in distributed- and co-located settings. The tools and forums benefit the organization to various degrees.

Temporal distance is not a prominent issue at Nihil. However, geographical and socio-cultural distance present more issues. Being geographically distributed challenge the core values of Scrum regarding personal relationships, and the lacking face-to-face communication co-located employees experience. The Socio-cultural distance at Nihil contributes to differentiate the teams' understanding of processes, created by subcultures within the organization. Further, the communication in the organization is affected by minor language barriers which creates flawed communication flow.

4.3.3 Rigid Processes in Nihil Impair Agility

Nihil is a large organization with multiple teams working on the same product simultaneously. This leads to challenges with keeping everyone informed about the different pars of the product and processes. How to keep track of the process is tightly connected to hypothesis 4.3.2, as communication is at the core of informing the different parties in the development process. Another common way of informing other parties about the process is by documentation.

One of the core values in the agile manifesto is working software over comprehensive documentation [10]. Even though agile development values working software over documentation, it does not mean that one should not document anything. Hunt [62] suggests that agile projects should have "sufficient" documentation both in short, and long-term projects. Long-lived projects will therefore need more documentation that short-lived ones. "Sufficient" is a vague term which is up to each individual to interpret.

> The challenge with an organization as big as Nihil, is that you need a lot documentation, because things need to be searchable, since you cannot keep everyone informed. The application is also so big that there is no one who can manage to keep track

> of everything, so we have to document a lot - Nihil employee.

Nihil is a growing organization and as it grows, people are replaced. Management employees therefore state that they can no longer rely on the collective memory of the employees, and the processes has to be documented accordingly. Documenting the different processes is therefore one of the things that has been prioritized. After the processes are described, and possibly adapted if needed, they are presented to the relevant people.

One of the main issues with growing an organization is how to scale up without losing the agile benefits of a smaller organizations. The question of centralizing decisions is relevant at Nihil. Agile methods such as Scrum facilitates for self organizing teams which choose how best to accomplish their work [15]. However, some of the advantages of being a large organization might be lost when each employee choose their own way.

> If you leave parts of the process to each individual to interpret, the results can be good because most people will do it in a pragmatic way. However, I think we can lose quite a lot of efficiency when you work in such a large-scale as we do. - Nihil management employee

Jacobsen & Thorsvik [42] summarizes the advantages and disadvantages of centralizing and decentralizing the authority of decisions in organizations, shown in table 4.2

Table 4.2: Advantages and disadvantages connected to centralization and decentralization [42] (p. 89)

	Centralization	**Decentralization**
Advantages	• Clear management signals • Clear responsibility • Consistent practice • Predictable practice	• Local adaptation • Flexibility • Motivating • Speed
Disadvantages	• Local information gets lost in the hierarchy • Low flexibility • Demotivating • Slow	• Lacking control and suboptimization • Unclear responsibility • Differing practice • Unpredictability

CPOs from the study shows that the organization had trouble keeping pro-

cesses uniform and streamlined. Everyone had a different idea about how things should be structured and formulated, and when several hundred employees do this in their own way, it can become chaotic. As the organization grows, it seems that it moves from a decentralized structure towards a more centralized. Several processes, such as Scrum reviews, Scrum team dashboards, and sprint goals are standardized, which creates consistency in the organization, but deprives employees with the ability to make decisions. The organization is far from extremity regarding centralization, but is neither decentralized to a full extent, but somewhere in between (see figure 4.4).

Figure 4.4: Scale for degrees of centralization and decentralization in organizations [42] (p. 89)

Both the organization's requirements for documentation some processes and events, and streamlining processes can be viewed as impediments to the workflow. Power [63] states that "anything that obstructs the smooth flow of work through the system and/or interferes with the system achieving its goals" (p. 88). Impediments like these highly impacts the workflow in the organization and are exact opposite of the first and second values of the agile manifesto [10].

Hypothesis Summary

Nihil is a large organization with the need to inform the different parts of the organization of events and artifacts. This is primarily done twofold, either by performing extensive documentation or standardizing processes. The organization recognizes that as people are replaced, they need to document their knowledge. Furthermore, the tasks that teams do in various ways can be more easily inspected if standardized.

Documentation and standardization is often seen as necessary in large organizations, but they are paradoxically opposites to agile values and principles.

Chapter 5

Discussion

This chapter resumes the research question. The hypotheses are discussed with basis in the research question. Finally the study's design, process, findings, and ethics are evaluated based on the conducted case study.

5.1 Research Question

The results from the study aim to answer the initial research question listed below. The research question is based on previous literature within research on large-scale agile projects, and aim contribute to the research field's collective knowledge about empirical large-scale agile research.

> **RQ** - Why is scaling Scrum challenging for a large-scale development organization?

The discussion will revolve around the resulting hypotheses from the study, and have basis in the research question. The study will also be discussed and evaluated in regards to the research method, and the overall research process.

5.2 Hypotheses

5.2.1 Nihil's Scrum Structure Forms Coordination Issues

Multiple-team organization

Nihil's use of POs is structured to utilize resources in large-scale development by having one PO responsible for multiple teams. While this is not in conflict with Scrum "rules" regarding either teams or roles, described in The Scrum Guide [15], it presents challenges compared to traditional Scrum. Scrum builds on values and principles from the Agile Manifesto, including the following value

> The most efficient and effective method of conveying information to and within a development team is face-to-face conversation [10].

Structuring several teams with less than one PO per team, leads to reduced time spent by each PO per team. Further, communication gets conveyed over different channels. These factors causes POs to have problems gaining overview and handling the projects. In order to try to counteract these problems, Nihil uses team-leads to ease the communication process between the team and the PO. Although these measures are in place, the coordination is far from perfect. This suggests that structuring multiple teams with a joint PO is a large part of the organization's coordination issue.

The research could have more easily seen the effect POs and teams by conducing a multiple case study with an organization within a similar context, but with a one-to-one relationship between POs and teams. By performing this kind of research, the researcher could further investigate the differences between the POs areas of responsibility, and assignments, to see their effect on how the Scrum teams are structured.

Scrum master and Product Owner are the two main roles which relates to the teams' daily work. Nihil's SMs are generally responsible for more teams than POs. In addition, they are inconsistent in the way the carry out their work as explained in subsection 4.3.1. With these factors affecting the SMs work, one would think the same kind of coordination issues would be prominent in their work. However, there were few indications of these kind of occurrences in the case. SMs' role would be interesting to further investigate in the same way as POs (as mentioned above), but with more

focus on what enables the SM role to coordinate across multiple teams without issues.

Two techniques are used at Nihil to try to increase the coordination between teams, Scrum of Scrums, and PO meetings. Both types of meetings have similar structure and are conducted weekly. SoS meetings are primarily attended by team leads, while PO meetings are attended by only POs. Common for the meetings are that the audience is quite large. SoS have an audience of from ca 20 teams, with attendees varying from one to the whole team. PO meetings include only the PO from each team, and has an audience of ca 10 participants. Bass [27] states that there currently are no agile ceremonies specifically designed towards large-scale agile software development, and this is the reason why organizations experiment with ceremonies such as SoS and PO meetings.

The techniques are both afflicted by some of the participants' disinterest in the meetings. The disinterest is usually a result of too varied topics in the meetings. Based on this, several of the meeting's attendees find the meetings to be of low value to them. This phenomenon matches findings in other studies on SoS [7, 38]. SoS meetings is one of the events at Nihil which clearly can be improved. The meeting is meant to improved inter-team coordination, but results in being unproductive and demotivating because of the employees lack of interest in the content. This leads to the use of SoS to be one of the factors which makes Scrum more challenging to scale for the large-scale development organization. PO meetings suffer from the same issue regarding disinterest as SoS meetings, even though the meetings are conducted with less participants.

Both types of meetings are "extra" events which are not part of traditional Scrum. This means that the meetings are not in any way mandatory to conduct in order to follow Scrum. Nihil can learn from Kniberg & Skarin's [18] suggestion about experimenting and customizing the process to your environment. Scrum, according to the Scrum Guide [15], emphasizes on its three pillars to uphold empirical process control; transparency, inspection, and adaptation [15]. By implementing Scrum with more focus on these three pillars, the organization will be better equipped against undesired effects of e.g. SoS and PO meetings. By focusing of transparency, the organization will increase its common understanding of processes. With better inspection, it can detect undesirable variances in the process. Adaptation allows for adaptation of the process if undesirable aspects are found trough inspection.

The CPOs affirming the hypothesis originate from multiple sources, which

strengthen the claim. Most of the CPOs are also primary sources, in form of observation or interview, which contain firsthand data about the subject which further help strengthen the hypothesis. The Scrum structure is strongly validated as a reason which makes scaling Scrum a challenge for Nihil. The structure of the PO role, as well as the coordination events across teams, are regarded as the prominent characteristics which make scaling difficult in the organization. The overall strength of the (sub)hypothesis is based on these reasons, and is therefore strongly verified.

Single-team organization

The hypothesis has implications related to traditional Scrum development. Elements from traditional Scrum, such as teams' size, are built on principles known from psychology. Teams in organizations are rarely composed of over ten people, since small teams often lead to high performance concerning reaching the organization's goals [64]. Nihil's decision of equipping Alpha team with eighteen developers in the same team, therefore seems confusing, as it contradicts general team-research [64], and the Scrum Guide's suggestion [15].

The decision of changing development method to Kanban for Alpha team, is bold based on the organizations history with Kanban use. Multiple teams at Nihil has previously changed from Kanban to Scrum due to disorganization of requirements. The teams that previously moved from Kanban to Scrum, had trouble, even with the improved rules and guidelines that Scrum provides, contra Kanban. Switching to Kanban means, less guidelines and structure for the team, hence the bold decision.

Vijayasarathy & Turk [56] summarize the main reasons for organizations' choosing, and benefits of adopting agile methods. The research show that Nihil's reason for changing development method correspond with the study's respondents' reasons for adopting agile methods. The study present project turn-around time and complexity of software as the organizations main reasons for use of agile methods [56]. Furthermore, the organizations state that meeting customer needs in a better way, faster time to delivery, increased flexibility in development, and improved software quality are the most frequent experienced benefits of using agile methods.

Alpha team's problems with timeboxing and meeting deadlines, correspond with Al-Baik & Miller [17], and Vijayasarathy & Turk [56] results regarding project turn-around time (lead time) and especially realization of faster time to delivery. This indicates that Kanban could be a good choice for the team.

However, since the adoption led to "Scrumban" (explained in subsection 2.1.3), it is unsure if the transition will have had full effect. There is also the additional element of a disadvantageous amount of developers in the same team.

The decision to switch development method to Kanban raises questions about the team's probability of success due to the organizations earlier experience with the method. Even if the challenges with timeboxed events are resolved, there is no guarantee that the team will not face problems, similar to historical Kanban teams. There is however a slightly higher chance of the process being successful than before, due to the inclusion of Scrum events in Kanban (Scrumban), such as retrospectives, reviews, and estimation of backlog items.

The hypothesis is largely dependent on second hand sources, where employees report on their own, or the organization's earlier experience with a phenomenon. These kind of reconstructed reports are not as reliable as data from direct observation, due to it being hard to verify. This is one of the factors that largely weakens the hypothesis. The CPOs also originate from few separate sources, which further weakens the hypothesis' claim. Furthermore, it is hard to conclude with anything, due to the project's time limit forcing the researcher to end the case study. The hypothesis is therefore verified weakly, due to these varied factors.

5.2.2 Communication Distances in Nihil Create a Lack of Individual Team Members' Project Understanding

Good communication is essential for an organization to function well. Communication is one of many elements which gets more complex as an organization grow in size. This entails that it might impact as a challenge of scaling Scrum. Members of Nihil's management recognizes that communication is a challenge for the organization. Lack of communication between agile actors have been claimed to increase the inter-personal distance between actors involved in the process, and in some cases even lead to project failure [13]

The lack of informal communication within the distributed organization is a problem at Nihil. Several distributed employees face this challenge daily. The measurement of this effect is not investigated in the thesis, but it can in worst case lead to poor work relations, instability, and absence

from work [42]. The Scrum framework facilitates for formal events such as meetings, and relational dependencies between some roles. Formal events are admittedly important for the organization's formal communication, and are easier to adapt considering the effects of geographical distribution in a large-scale context. Scrum has no explicit events or artifacts for helping informal communication, as the framework is designed for small co-located teams [3]. This causes indirect communication to be one of the characteristics which make scaling Scrum to a distributed large-scale development difficult. Pikkarainen et al. [13] found in their study on agile practices, that informal communication works as a factor which indicate less need for documentation, and more productive software development.

Geographical distance further bring challenges regarding employees' personal work-relationship in the large-scale organization's utilization of the Scrum framework. Distributed development often causes employees to communicate using digital tools as their main form of communication. This can in many cases work fine, but can in other situations be troublesome. Communication via e.g. text, and video are less rich communication channels than face-to-face (see figure 2.8). This can lead to communication challenges, especially for employees with roles such as PO and SM, who require a comprehensive overview of the development process, in form of inter-team coordination and communication.

As a result of insufficient richness in the communication, employees can encounter misunderstandings in other teams' work, thus increase the lack of the project's understanding.

Nihil is a large organization, reaching over multiple continents. The challenges related to this is presented in form of socio-cultural differences, rather than temporal distances, as presented in section 4.3.2. The socio-cultural differences at Nihil can be identified as different organizational subcultures. The subcultures work side by side without any inherent conflict, and can even exist simultaneously without affecting one another. The different subcultures behave in different ways based on different factors, as explained in section 4.3.2. Although the teams' behavior is differentiated based on the teams' geographical location, this does not seem like the most important factor regarding their behavior. The different teams' attitude towards authorities seems to highly impact their behavioral patterns. Ramesh et al. [31] summarizes findings from studies on cultural impacts in global software development which show similar results. An example of this is North American teams' need for reaffirmation of instructions, which indicate that they consider the hierarchical structure of the organization as more horizontal

than other teams.

The socio-cultural effects in the study are hard establish with certainty because of the limitations of the study, further explained in section 5.3. The results are nevertheless valid examples of reasons why scaling Scrum to large-scale is challenging for an agile software development organization. Vijayasarathy & Turk [56] found that incompatibility with development culture as one of the main limitations of agile development. Inconsistent practices with basis in socio-cultural differences between teams, lead to difficulties for the management to communicate efficiently across cross-cultural teams. Further research on global software development in large-scale Scrum contexts could be interesting to investigate in depth, to find more evidence of socio-cultural distance in global Scrum-based development organizations.

The CPOs connected to the hypothesis are strongly verified based on the type of evidence. Informal communication combined with socio-cultural and geographical distance result in strong inference about the organization's communication issues, which directly influences the understanding of the project. The evidence is largely obtained from multiple sources, and the different CPOs show consistent data regarding the hypothesis. This contribute to making the causal inference of the hypothesis strongly verified.

5.2.3 Rigid Processes in Nihil Impair Agility

The first value of the Agile Manifesto is "Individuals and interactions over processes and tools" [10]. This means that individuals and interactions are valued over processes and tools. Processes and tools are definitely important for the development to go smoothly, but they should not compromise agility in form of valuing individuals and interactions.

Agile methods such as Scrum are developed for small co-located teams. This is done to ensure agile development with focus on efficiency. This present challenges as organizations grow, and development gets more complex. The implications of scaling Scrum in the organization are varied, but are among the aforementioned hypotheses, associated with the implementation of rigid organization processes.

Documenting processes is frequently used at Nihil to ensure the traceability of information within the organization. It is primarily done in form of writing reports or documents which are searchable for the organization. This

is in addition to the traditional Scrum artifacts that are also used, such as: user stories, sprint backlog, story points, estimates, and burn down charts. Petersen & Wohlin [65] states that the use of documentation can be reduced by physically moving people together, and increasing direct communication. This is however challenging in large-scale organizations as teams can be for example geographically distributed. It has also been suggested that new agile ceremonies are needed which could review and refine Scrum artifacts [27]. However, there is a risk that additional ceremonies and artifacts can distract developers from the producing working code [27]. Documentation's role in agile development is rated as unimportant in studies on adoption of agile methods [56], congruent with agile values, explained in section 2.1.

Together with a sizable focus of documentation, Nihil operates with centralizing processes in hopes of increasing efficiency within the organization. Although centralization has several advantages (shown in table 4.2), such as consistent and predictable practices, it is generally inconsistent with the basic agile values. This lead teams to not being able to self-organize in how to best accomplish their work. These kind of effects contribute to impair the agility of the organization in a way that in worst case can hinder technological innovations.

Nihil can be considered as an organization with "growth-pains", meaning that it is in a state where the organization is increasing in size, but do not know how to handle the accompanied changes.

The CPOs associated with the hypothesis show that the organization tries to adapt to its large-scale environment. The use of documentation is increasing in pace with the organization's size, and therefore leads to more documentation apart from what is normal in traditional Scrum. Centralization of processes prove to be an increasing factor in large-scale development, which reduce the agility in the organization. The CPOs related to the hypothesis are a mixture of primary and secondary sources of information. The hypothesis' inference is moderately verified based on the evidence, showing that several processes at Nihil are becoming increasingly rigid. The organization is far from as rigid as organizations that are using traditional methods, but it is somewhere in between. This further indicate that scaling Scrum to a large scale becomes increasingly difficult based on organizational size.

5.2.4 Summary

The study aims to find out why scaling Scrum is challenging for large-scale development organizations, and what characteristics make scaling difficult.

The hypotheses are presented with different inferences of why scaling is difficult and what characteristics makes it difficult. The inferences are verified individually, with varying strength, as seen in table 5.1.

Table 5.1: Hypotheses' strength

Hypotheses	**Strength**
Nihil's Scrum structure forms coordination issues	
• in multiple-team organization	Strongly verified
• in single-team organization	Weakly verified
Communication distances in Nihil create a lack of individual team members' project understanding	Strongly verified
Rigid processes in Nihil impair agility	Moderately verified

Sub-hypothesis 4.3.1 present reasons for why scaling scrum is challenging in the organization. Issues related to coordination, such as role distribution across multiple teams and the adaptation of extra events such as SoS and PO meetings are both indications of reasons why scaling is difficult in the organization.

Single teams such as Alpha team have an untraditional structure, with a high amount of developers in the team. The team also switched development method to Kanban (Scrumban) despite other teams having trouble with this earlier. The sub-hypothesis indicates that these factors can be some of the reasons behind the organization's scaling challenges. However, the hypothesis has multiple uncertainties tied to it, and weak evidence, which reduces its causal inference.

Communication pervades the organization in most ways. Issues regarding communication in the organization, explained in hypothesis 4.3.2, is strongly verified as a reason for why scaling becomes difficult. The evidence is unambiguous and evident, and provide notable characteristics as to why scaling Scrum is challenging in the organization.

Rigid processes at Nihil are identified as a moderately verified reason to the challenges with scaling Scrum. The rigid processes are classified and divided into documentation and centralization. The organization still performs agile development, but is affected by the need for systematic oversight, due to its size and complexity. This is one of the reasons why scaling Scrum to large-scale organizations can become challenging.

The resulting hypotheses can be categorized into three areas, based on their main challenges toward large-scale agile development.

- Hypothesis 4.3.1 - Coordination

- Hypothesis 4.3.2 - Communication
- Hypothesis 4.3.3 - Processes

The categorization is not a mutually exclusive division of the hypotheses, as the topics overlap to a great extent. The resulting categories match previously suggested research challenges in large-scale agile development [4], shown in table 2.3.

The hypotheses combined form a collective basis for why scaling Scrum to the large-scale agile development organization is challenging. The accompanied characteristics of the issue are identified as indicating factors of the hypotheses.

5.3 Evaluation of the Study

The evaluation of the study is presented to give the reader an understanding of study, and the choices made during the research process. This section is meant to give the reader a justification of the choices made about the research process.

5.3.1 Research Design

The research described in this thesis was initially thought to be a case study with process tracing used as a within-method. Process tracing as a within-method is according to Collier [48] hard to do well. Some of the most prominent problems in process tracing is missing variables, measurement error, and that probabilistic relationships are harder to address than in quantitative research. Process tracing also gives close attention to the sequence of variables [48].

The study is designed to identify and investigate of disaggregated causal mechanisms in accordance with the description of process tracing (explained in section 3.2). The analysis of the collected data in the case show after multiple iterations, that the data material lacked sequential events to analyze for causal mechanisms. This led to the evidence in the study being "mechanistic", which is often weak in terms of enabling causal inference because the causal mechanisms are not explicit [52]. The case's lack of explicitly theorized mechanisms led the within-case method to match a congruence case study (explained in section 3.3). Congruence case studies do not have to specifically identify entities and activities tied to the mechanism.

Even though the advantages of using process tracing were lost in switching the analytical method to a congruence case study, it is still valuable in several research situations. Beach & Pedersen [52] argues that although it might always seem logical to choose process tracing, congruence case studies are typically less demanding in terms of analytical resources, and the researcher only makes claims about *plausible* casual relationships within the case [52]. This way of analyzing evidence, lay the foundation for the most promising causal conjectures in the case, and can then be further empirically investigated in a more rigorous way, using e.g. process-tracing [52].

Further, Beach & Pedersen [52] state that in order to claim that one are providing evidence of a causal process, one has to explicitly state the causal processes by providing and explaining the causal mechanisms. One has to differentiate between process tracing and congruence case study methods in order to avoid confusing and/or contradictory statements about the methods [52]. "We suggest that one should not claim to be doing process-tracing when one is not actually tracing a causal process" [52] (p. 353).

The research design was according to Beach et. al.'s [48] points concerning the advantages of using process tracing:

(a) identify social phenomena within the domain of agile software development, with focus on scalability and describing them.

(b) look at previous literature regarding scalability in agile projects and evaluate their hypothesis to look at their relevance towards the specific case. Discover new hypothesis, and assess new causal claims.

(c) understand the causal mechanisms with basis in my data collection and established literature.

Points (a) and (b) are still valid for the congruence case study, while point (c) is disregarded due to the lack of causal mechanisms in the studied case. The lack of causal mechanisms is reflected in the thesis' limitations regarding its time frame. The time frame of the thesis, combined with a novice case study researcher led to the results, not being as exhaustive in terms of causal inference, as first intended in the case study's design.

Research Design Quality

Construct validity The study show a high degree of construct validity. Several of the results in this study, correlate with previous findings in similar studies, e.g. as discussed in section 5.2.1. Nihil also strive with issues which

are suggested research topics by researchers within the agile community (see table 2.3).

Internal validity The research was designed to explore as many rival explanations as possible by initially being exploratory, as explained in section 3.5.2. Due to the time limitations of the study, all causal factors related to the hypotheses were not explored. This weakens the internal validity of the study. By not ruling out all rival explanations, one can not say that the explanation is internally valid. The study have however, found the most prominent traits regarding scalability issues in the organization, which can be further explored, as further explained in subsection 5.3.3

External validity Congruence case studies can according to Beach & Pedersen [52] be used to assess causal relationships in a case without the aim to generalize the results beyond the case itself. The case presented in this thesis aim to generalize toward organizations within similar contexts. For example organizations with similar Scrum structure, organization of events, or with other resembling factors in a large-scale development context. The outcome explained in the case, can be perceived with a holistic view which in other cases may not be generalizable because of the amount of distinctive characteristics.

The thesis analyzes the case from a software engineering and organizational theoretical point of view. The organizational point of view allows readers which are unfamiliar with the technical aspects of software engineering to more easily understand if the case is generalizable towards e.g. other organizations' context, as this often can be more recognizable.

Reliability Measures were made to ensure the reliability of the study, explained in section 3.5.2. However, the study has implication concerning its reliability. Evidence from the study would have had problems demonstrating similar results under consistent conditions due to the organization being dynamic. The results from the study with low verification due to low amount of evidence, or secondary sources will have low reliability. The results with high verification (see table 5.1) have higher reliability, due to the strength of the evidence and especially their source. The mixture of the evidence in the study, result in an overall moderate reliability of the study.

5.3.2 Research Process

The research process lasted eight months, and reached from August 2016 to May 2017. This included everything from planning and designing the case, to executing it and writing the thesis. The time constraints were the limitation which proved to impact the case to the greatest extent. The case turned out to require more time to complete than first anticipated.

Process tracing as a within-case analysis method proved to require a proficiency which extends beyond the researcher's skill regarding qualitative research. The thesis describe the work conducted by one researcher over a ten month period at a master course. The work indicate that the researcher underestimated the work needed to conduct a process tracing analysis, which can further indicate that process tracing as an analysis method, is unfit for smaller projects with novice researchers.

The need for explicit, and sequential events in process tracing, lead the researcher to believe that the projects was also unfit for this type of analysis. The project lacked a defined starting and end-point, as it was an continuous development of an existing product.

5.3.3 Research Findings

The CPOs in the study have various degree of supporting evidence. Some of the CPOs are highly supported by other findings, and therefore weighs heavier than the ones without supporting ones. Findings from e.g. interviews where the interviewee speaks of a phenomenon which only they experienced, is for example hard to support with complementary evidence. This leads to hypotheses being verified to various levels based on not only the amount of evidence which points in its direction, but also the strength of each individual piece of evidence. The sum of the hypotheses' strength is summarized in table 5.1.

The causal effect of an explanatory variable is defined by Bennet & George [50] as "the change in the probability and/or value of that dependent variable that would have occurred if the explained variable had assumed a different value" (p. 1). This is a troublesome definition of causal effect, because it is highly connected to the reliability of the study. The problem is that it is not possible to re-run history and change only one variable that would allow us to observe the cause effect of that variable, in the same way that it is impossible to redo the study with the exact same results [50].

In the study, it proved to be harder to than expected to discover a sequence of variables which could be sufficiently identified as causal mechanisms. Due to time limitations in the study, some areas were not explored to a full extent. For example, the researcher could have explored new areas concerning the research question, or further investigated the aforementioned hypotheses. The limitations of a congruence case study, contra process tracing caused a lower degree of causal strength of the findings.

5.3.4 Research Ethics

The research described in this thesis is done according to the Norwegian National Research Ethics Committee's guidelines for research ethics in science and technology (NENT), explained in section 3.4. NENT state that "research that involves research subjects raises special requirements regarding respect for the individual subject" [54].

The data presented in the thesis is confidential, meaning that it is anonymized to such an extent that the persons can not be identified by the information presented in the thesis. This lead to issues regarding being able to sufficiently describing persons with e.g. specific roles, areas of responsibility, or similar features. The anonymisation process have to some extent prevented extensive description of the case. This has made the case imprecise in some areas, but is however considered more important than identifying characteristics related to the subjects in the study.

Chapter 6

Conclusion

This chapter conclude the main results found in the research. Further, the chapter introduces possible future work based on the the research's results.

6.1 Research Question

> **RQ** - Why is scaling Scrum challenging for a large-scale development organization?

The thesis aim to investigate reasons for why scaling Scrum is challenging for a large-scale development organization, and what characteristics make scaling Scrum difficult. The thesis aims to contribute to an increased collective collective knowledge regarding challenges of large-scale agile development.

As presented in the thesis there are multiple reasons for why scaling Scrum is challenging for a large-scale development organization. The thesis result in four hypothesis which include multiple characteristics which indicate why scaling Scrum is difficult.

The resulting hypotheses (shown in table 6.1) are obtained from a congruence case study conducted at a global software development organization using Scrum on a large scale.

Table 6.1: Hypotheses (Subhypotheses are indented and marked with a "•").

Hypotheses
Nihil's Scrum structure forms coordination issues
• in multiple-team organization
• in single-team organization
Communication distances in Nihil create a lack of individual team members' project understanding
Rigid processes in Nihil impair agility

The results of why scaling scrum is difficult in the organization, can be categorized into three main areas: Coordination, communication, and processes. Neither of the three categories are isolated, and the results from the study overlaps a lot in how they can be categorized. The hypotheses' categorization matches previously suggested research challenges regarding agile research, with topics such as: inter-team communication, large project organization, and scaling agile practices [4].

Issues with coordination across multiple teams is seen as a strong factor to why it is challenging for the organization to scale up Scrum. Multiple characteristics related to structure and coordinating events, indicate that Scrum is challenging to scale. Coordination in the way single teams are organized is weakly verified as a reason, because of limitations to the study.

Communication is largely related to all the work in the organization, and proves to be an important factor connected to the challenge of scaling Scrum for a large-scale organization. Distinctive characteristics related to communication, such as indirect communication affects the organizations ability to scale well, and present challenges in large-scale Scrum.

Rigid processes that get in the way of agile development suggest to be a moderately contributing factor to why scaling Scrum is challenging in the organization. Increased documentation and centralized decision making are characteristics of rigid processes which indicate that scaling Scrum becomes increasingly difficult at large scale.

The resulting hypotheses combined, create a collective basis for answering the main reasons for why scaling Scrum is challenging in the organization. The accompanied characteristics help indicate the reason behind the challenges.

6.2 Future Work

The thesis' case study gives insight to some of the challenges regarding large-scale agile development in the organization. The congruence case method gives mechanistic evidence, which is weak in terms of enabling causal inference. It would be interesting to analyze a similar case with a real process-tracing approach, with focus on events and activities throughout a project with a well-defined start and end point. By performing a sequential process tracing approach, the resulting evidences' causal inference would be stronger. A follow-up process tracing study, could base its approach on the results from this thesis' work.

The main cause for limitation to the thesis was its time constraint. Further work on the same project as described in the thesis could lead to a deeper and more exhaustive investigation of the phenomena in the case. The case could be further investigated, with multiple new hypotheses which could provide a larger basis for explaining the challenges behind scaling Scrum in the organization. Such an investigation would also presumably identify more rival hypotheses, which again would strengthen results.

Bibliography

[1] T. Dingsøyr and N. B. Moe, "Towards Principles of Large-Scale Agile Development A Summary of the Workshop at XP2014 and a Revised Research Agenda," *Agile Methods. Large-Scale Development, Refactoring, Testing, and Estimation*, vol. 199, pp. 1–8, 2014. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-14358-3_1

[2] A. Q. Gill, B. Henderson-Sellers, and M. Niazi, "Scaling for agility: A reference model for hybrid traditional-agile software development methodologies," *Information Systems Frontiers*, pp. 1–27, 2016. [Online]. Available: http://dx.doi.org/10.1007/s10796-016-9672-8

[3] L. Williams and C. Alistair, "Agile Software Development: It's about Feedback and Change," *IEEE Computer Society*, vol. 36, pp. 39–43, 2003. [Online]. Available: http://ieeexplore.ieee.org/document/1204373/

[4] T. Dingsøyr and N. B. Moe, "Research challenges in large-scale agile software development," *ACM SIGSOFT Software Engineering Notes*, vol. 38, no. 5, pp. 38–39, 2013. [Online]. Available: http://dl.acm.org/citation.cfm?id=2507288.2507322

[5] T. Dingsøyr, S. Nerur, V. Balijepally, and N. B. Moe, "A decade of agile methodologies: Towards explaining agile software development," *Journal of Systems and Software*, vol. 85, no. 6, pp. 1213–1221, jun 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121212000532

[6] D. Cohen, M. Lindvall, and P. Costa, "An Introduction to Agile Methods," in *Advances in Computers*, 2004, vol. 62, no. C, pp. 1–66. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0065245803620012

[7] M. Paasivaara, C. Lassenius, and V. T. Heikkila, "Inter-team coordination in large-scale globally distributed scrum: Do Scrum-of-Scrums really work?" *Empirical Software Engineering and Measurement (ESEM), 2012 ACM-IEEE International Symposium on*, pp. 235–238, 2012. [Online]. Available: http://doi.acm.org/10.1145/2372251.2372294

[8] R. K. Yin, *Case Study Research: Design and Methods*, 5th ed. Sage Publications, Inc, 2014.

[9] T. Dingsøyr, T. E. Fægri, and J. Itkonen, "What Is Large in Large-Scale? A Taxonomy of Scale for Agile Software Development," in *Product-Focused Software Process Improvement*, 2014, vol. 8892, no. 7465, pp. 273–276. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-319-13835-0_20

[10] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas, "The Agile Manifesto," 2001. [Online]. Available: http://agilemanifesto.org/history.html

[11] S. Shore, James., Warden, *The Art of Agile Development*. California: O'Reilly Media, Inc., 2008.

[12] R. Freedman, *The Agile Consultant*. Apress, 2016.

[13] M. Pikkarainen, J. Haikara, O. Salo, P. Abrahamsson, and J. Still, "The impact of agile practices on communication in software development," *Empirical Software Engineering*, vol. 13, no. 3, pp. 303–337, jun 2008. [Online]. Available: http://link.springer.com/10.1007/s10664-008-9065-9

[14] J. Highsmith and A. Cockburn, "Agile software development: the business of innovation," *Computer*, vol. 34, no. 9, pp. 120–127, 2001. [Online]. Available: http://ieeexplore.ieee.org/document/947100/

[15] K. Schwaber and J. Sutherland, "The Scrum Guide - The Definitive Guide to Scrum: The Rules of the Game," *Scrum.org*, vol. 2, no. October, p. 17, 2011. [Online]. Available: http://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-us.pdf

[16] Scrum.org, "What is Scrum?" 2017. [Online]. Available: https://www.scrum.org/resources/what-is-scrum

[17] O. Al-Baik and J. Miller, "The kanban approach, between agility and leanness: a systematic review," *Empirical Software Engineering*, vol. 20, no. 6, pp. 1861–1897, dec 2015. [Online]. Available: http://link.springer.com/10.1007/s10664-014-9340-x

[18] H. Kniberg and M. Skarin, *Kanban and Scrum-making the most of both*, D. Plesa, Ed. C4Media Inc., 2010, vol. 1. [Online]. Available: http://www.infoq.com/minibooks/kanban-scrum-minibook

[19] A. Reddy, *The Scrumban Revolution*, 1st ed. Addison-Wesley Professional, 2015. [Online]. Available: http://dl.acm.org/citation.cfm?id=2810092

[20] L. Tal, *Agile Software Development with HP Agile Manager*, 1st ed. Berkeley, CA: Apress, 2015. [Online]. Available: http://link.springer.com/10.1007/978-1-4842-1034-5

[21] D. Misevičiūtė, "Scrum vs. Kanban vs. Scrumban – What's the difference?" 2016. [Online]. Available: http://www.eylean.com/blog/2016/08/scrum-vs-kanban-vs-scrumban-whats-the-difference/

[22] X. Wang, K. Conboy, and O. Cawley, ""Leagile" software development: An experience report analysis of the application of lean approaches in agile software development," *Journal of Systems and Software*, vol. 85, no. 6, pp. 1287–1299, jun 2012. [Online]. Available: http://dx.doi.org/10.1016/j.jss.2012.01.061

[23] S. Freudenberg and H. Sharp, "The Top 10 Burning Research Questions from Practitioners," *IEEE Software*, vol. 27, no. 5, pp. 8–9, sep 2010. [Online]. Available: http://ieeexplore.ieee.org/document/5551011/

[24] P. Roach, "Scrum at Scale: Changing the Conversation," 2014. [Online]. Available: https://www.scruminc.com/scrum-scale-case-modularity/

[25] A. Scheerer, T. Hildenbrand, and T. Kude, "Coordination in Large-Scale Agile Software Development: A Multiteam Systems Perspective," in *2014 47th Hawaii International Conference on System Sciences*. IEEE, jan 2014, pp. 4780–4788. [Online]. Available: http://ieeexplore.ieee.org/document/6759189/

[26] O. Ktata and G. Lévesque, "Agile development," in *Proceedings of the 2009 C3S2E conference on - C3S2E '09*. New York, New York, USA: ACM Press, 2009, p. 59. [Online]. Available: http://portal.acm.org/citation.cfm?id=1557626.1557636

[27] J. M. Bass, "Artefacts and agile method tailoring in large-scale offshore software development programmes," *Information and Software Technology*, vol. 75, pp. 1–16, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.infsof.2016.03.001

[28] M. Paasivaara and C. Lassenius, "Deepening Our Understanding of Communities of Practice in Large-Scale Agile Development," in *2014 Agile Conference*. IEEE, jul 2014, pp. 37–40. [Online]. Available: http://ieeexplore.ieee.org/document/6910401/

[29] J. M. Bass, "Agile Method Tailoring in Distributed Enterprises: Product Owner Teams," in *2013 IEEE 8th International Conference on Global Software Engineering*. IEEE, aug 2013, pp. 154–163. [Online]. Available: http://ieeexplore.ieee.org/document/6613080/

[30] S. W. Ambler, "Agile Software Development at Scale," in *Balancing Agility and Formalism in Software Engineering*, 2008, pp. 1–12. [Online]. Available: http://link.springer.com/10.1007/978-3-540-85279-7_1

[31] B. Ramesh, L. Cao, J. Kim, K. Mohan, and T. L. James, "Conflicts and complements between eastern cultures and agile methods: an empirical investigation," *European Journal of Information Systems*,

no. June 2012, 2017. [Online]. Available: http://link.springer.com/article/10.1057/s41303-016-0023-0

[32] P. L. Bannerman, E. Hossain, and R. Jeffery, "Scrum Practice Mitigation of Global Software Development Coordination Challenges: A Distinctive Advantage?" in *2012 45th Hawaii International Conference on System Sciences*. IEEE, jan 2012, pp. 5309–5318. [Online]. Available: http://ieeexplore.ieee.org/document/6149537/

[33] N. B. Moe, H. H. Olsson, and T. Dingsøyr, "Trends in Large-Scale Agile Development," in *Proceedings of the Scientific Workshop Proceedings of XP2016 on - XP '16 Workshops*, no. 7465. New York, New York, USA: ACM Press, 2016, pp. 1–4. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2962695.2962696

[34] A. Scheerer, S. Bick, T. Hildenbrand, and A. Heinzl, "The Effects of Team Backlog Dependencies on Agile Multiteam Systems: A Graph Theoretical Approach," in *2015 48th Hawaii International Conference on System Sciences*, vol. 2015-March. IEEE, jan 2015, pp. 5124–5132. [Online]. Available: http://ieeexplore.ieee.org/document/7070428/

[35] J. Pernstål, R. Feldt, and T. Gorschek, "The lean gap: A review of lean approaches to large-scale software systems development," *Journal of Systems and Software*, vol. 86, no. 11, pp. 2797–2821, nov 2013. [Online]. Available: http://dx.doi.org/10.1016/j.jss.2013.06.035

[36] C. Hibbs, S. Jewett, and M. Sullivan, *The Art of Lean Software Development: A Practical and Incremental Approach*, 1st ed. O'Reilly Media inc., 2009.

[37] K. Petersen, "Is Lean Agile and Agile Lean?" in *Modern Software Engineering Concepts and Practices*. IGI Global, 2011, no. Beck 2000, pp. 19–46. [Online]. Available: http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-60960-215-4.ch002

[38] F. Evbota, E. Knauss, and A. Sandberg, "Scaling up the Planning Game: Collaboration Challenges in Large-Scale Agile Product Development," in *Agile Processes, in Software Engineering, and Extreme Programming: 17th International Conference, XP 2016*, 2016, vol. 251, pp. 28–38. [Online]. Available: http://link.springer.com/10.1007/978-3-319-33515-5_3

[39] The LeSS Company B.V., "Large Scale Scrum (LeSS)," 2017. [Online]. Available: https://less.works/less/framework/introduction.html

[40] Scaled Agile Inc., "Scaled Agile Framework," 2017. [Online]. Available: http://www.scaledagileframework.com/

[41] A. Elssamadisy, "InfoQ," 2013. [Online]. Available: https://www.infoq.com/news/2013/08/safe

[42] D. I. Jacobsen and J. Thorsvik, *Hvordan Organisasjoner Fungerer*, 4th ed. Bergen: Fagbokforlaget, 2013.

[43] J. Iivari and N. Iivari, "The relationship between organizational culture and the deployment of agile methods," *Information and Software Technology*, vol. 53, no. 5, pp. 509–520, may 2011. [Online]. Available: http://dx.doi.org/10.1016/j.infsof.2010.10.008

[44] B. J. Oates, *Researching Information Systems and Computing.* London: SAGE Publications Ltd., 2006.

[45] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, no. 2, pp. 131–164, apr 2009. [Online]. Available: http://link.springer.com/10.1007/s10664-008-9102-8

[46] J. Seawright, "Case Studies and Theory Development in the Social Sciences. By Alexander George and Andrew Bennett. (MIT Press, 2005.)," *The Journal of Politics*, vol. 70, no. 1, pp. 276–278, jan 2008. [Online]. Available: http://www.journals.uchicago.edu/doi/10.1017/S0022381607080231

[47] G. Allan, "A critique of using grounded theory as a research method," *Electronic Journal of Business Research Method*, vol. 2, no. 1, pp. 1–10, 2003. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.129.9102

[48] D. Collier, "Understanding Process Tracing," *PS: Political Science & Politics*, vol. 44, no. 04, pp. 823–830, oct 2011. [Online]. Available: http://www.journals.cambridge.org/abstract_S1049096511001429

[49] J. Blatter and M. Haverland, "Case Studies and (Causal-) Process Tracing," *SSRN Electronic Journal*, no. October, pp. 1–27, 2014. [Online]. Available: http://www.ssrn.com/abstract=2390618

[50] A. Bennet and A. L. George, "Process Tracing in Case Study Research," *MacArthur Foundation Workshop on Case Study Methods, Belfer Center for Science and International Affairs (BCSIA), Harvard University*, pp. 104–105, 1997. [Online]. Available: https://www.uzh.ch/cmsssl/suz/dam/jcr:00000000-5103-bee3-0000-000059b16b9d/05.19.bennett_george.pdf

[51] D. Collier, *Process Tracing: Introduction and Exercises.* Department of Political Science, University of California, Berkeley, 2010. [Online]. Available: http://dmeforpeace.org/sites/default/files/Collier_ProcessTracing.pdf

[52] D. Beach and R. Pedersen, *Causal Case Study Methods.* Ann Arbor, MI: University of Michigan Press, 2016. [Online]. Available: http://www.press.umich.edu/6576809

[53] J. Blatter and M. Haverland, "Congruence Analysis," in *Designing Case Studies.* London: Palgrave Macmillan UK, 2012, pp. 144–204. [Online]. Available: http://link.springer.com/10.1057/9781137016669_4

[54] The National Committee for Research Ethics in Science and Technology (NENT), *Guidelines for Research Ethics in Science and Technology*, 2nd ed., 2016. [Online]. Available: https://www.etikkom.no/en/ethical-guidelines-for-research/guidelines-for-research-ethics-in-science-and-technology/

[55] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian, "Selecting Empirical Methods for Software Engineering Research," in *Guide to Advanced Empirical Software Engineering*, F. Shull, J. Singer, and D. I. K. Sjøberg, Eds. London: Springer London, 2008, vol. 53, no. 9, pp. 285–311. [Online]. Available: http://link.springer.com/10.1007/978-1-84800-044-5_11

[56] L. R. Vijayasarathy and D. Turk, "Agile software development: A survey of early adopters," *Journal of Information Technology Management*, vol. XIX, no. 2, pp. 1–8, 2008. [Online]. Available: http://www.aom-iaom.org/jitm_pdfs/jitm_08/article3.pdf

[57] A. Bryman, *Social research methods*, 4th ed. Oxford: Oxford University Press Inc., 2012.

[58] M. Cohn, "Advice on Conducting the Scrum of Scrums Meeting," 2007. [Online]. Available: https://www.scrumalliance.org/community/articles/2007/may/advice-on-conducting-the-scrum-of-scrums-meeting

[59] L. Faria, "Scrum of Scrums: Running Agile on Large Projects," 2013. [Online]. Available: https://www.scrumalliance.org/community/articles/2013/june/scrum-of-scrums-running-agile-on-large-projects

[60] L. Šteinberga and D. Šmite, "Towards a Contemporary Understanding of Motivation in Distributed Software Projects: Solution Proposal," *Scientific Papers, Computer Science and Information Technologies, Vol. 770*, vol. 770, pp. 15 – 26, 2011. [Online]. Available: https://dspace.lu.lv/dspace/bitstream/handle/7/2266/LUR-770-Datorzinatne.pdf?sequence=1{&}isAllowed=y

[61] S. Dorairaj, J. Noble, and P. Malik, "Effective Communication in Distributed Agile Software Development Teams," in *Agile Processes in Software Engineering and Extreme Programming*, 2011, vol. 77, pp. 102–116. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-642-20677-1_8{%}5Cnhttp://link.springer.com/content/pdf/10.1007/978-3-642-20677-1_8.pdf

[62] J. Hunt, "Agile Methods and the Agile Manifesto," in *Agile Software Construction*. London: Springer London, 2006, pp. 9–30. [Online]. Available: http://dx.doi.org/10.1007/1-84628-262-4_2

[63] K. Power, "A Model for Understanding When Scaling Agile Is Appropriate in Large Organizations," in *Agile Methods. Large-Scale Development, Refactoring, Testing, and Estimation: Revised Selected Papers of the XP 2014*, 2014, pp. 83–92. [Online]. Available: http://link.springer.com/10.1007/978-3-319-14358-3_8

[64] G. Kaufmann and A. Kaufmann, *Psykologi i organisasjon og ledelse*, 4th ed. Bergen: Fagbokforlaget Vigmostad og Bjørke AS, 2009.

[65] K. Petersen and C. Wohlin, "The effect of moving from a plan-driven to an incremental software development approach with agile practices," *Empirical Software Engineering*, vol. 15, no. 6, pp. 654–693, dec 2010. [Online]. Available: http://link.springer.com/10.1007/s10664-010-9136-6

Appendix A

Informed Consent Form

Forespørsel om deltakelse i forskningsprosjektet

Skalering av smidige programvareutviklingsprosjekter: en case studie

Bakgrunn og formål
Forskningsprosjektet er en masteroppgave ved Universitetet i Bergen, institutt for informasjons- og medievitenskap. Prosjektet tar opp utfordringer med å skalere utviklingsprosjekter av typen Scrum fra et til flere teams. I prosjektet vil det bli analysert data fra flere kilder for å finne årsakssammenhenger til hvorfor skalering er problematisk i det aktuelle prosjektet. Prosjektet gjennomføres i samarbeid med itslearning.

Utvalg av deltagere til forskningsprosjektet blir gjort på grunnlag av hvilket utviklingsprosjekt den enkelte deltaker er involvert i.

Hva innebærer deltakelse i studien?
Studien vil innebære datainnsamling i form av intervju og observasjon. Observasjonsdelen av studien krever ingen aktiv deltakelse, mens intervju krever aktiv deltagelse fra personene som ønsker å delta. Intervjuspørsmålene vil i hovedsak omhandle deltakerens engasjement og deltakelse i utviklingsprosjekter. Spørsmålene vil være rettet mot tanker og synspunkter ol. angående måten utviklingen foregår på. Data fra intervju registreres i form av lydopptak og notater.

Hva skjer med informasjonen om deg?
Alle personopplysninger vil bli behandlet konfidensielt. Kun student og veileder til masteroppgaven vil ha tilgang til personopplysninger. Notater og lydopptak lagres sikkert på en digital lagringsenhet.

Prosjektet vil anonymiseres, og deltakere i prosjektet vil ikke kunne gjenkjennes i publikasjon av oppgaven.

Prosjektet skal etter planen avsluttes 01.06.2017. Datamaterialet anonymiseres etter prosjektslutt.

Frivillig deltakelse
Det er frivillig å delta i studien, og du kan når som helst trekke ditt samtykke uten å oppgi noen grunn. Dersom du trekker deg, vil alle opplysninger om deg bli anonymisert.

Dersom du har spørsmål til studien, ta kontakt med:
Simen Jensen på epost simen.jensen@uib.no / tlf. 904 78 039,
eller veileder Bjørnar Tessem på epost bjornar.tessem@uib.no / tlf. 555 84 103

Studien er meldt til Personvernombudet for forskning, NSD - Norsk senter for forskningsdata AS.

Samtykke til deltakelse i studien

Jeg har mottatt informasjon om studien, og er villig til å delta

---
(Signert av prosjektdeltaker, dato)

Appendix B

NSD Approval

Bjørnar Tessem
Institutt for informasjons- og medievitenskap Universitetet i Bergen
Fosswinckelsgate 6
5007 BERGEN

Vår dato: 31.10.2016 Vår ref: 50358 / 3 / ASF Deres dato: Deres ref:

TILBAKEMELDING PÅ MELDING OM BEHANDLING AV PERSONOPPLYSNINGER

Vi viser til melding om behandling av personopplysninger, mottatt 03.10.2016. Meldingen gjelder prosjektet:

50358	*Skalering av smidige programvareutviklingsprosjekter: en case studie*
Behandlingsansvarlig	*Universitetet i Bergen, ved institusjonens øverste leder*
Daglig ansvarlig	*Bjørnar Tessem*
Student	*Simen Jensen*

Personvernombudet har vurdert prosjektet og finner at behandlingen av personopplysninger er meldepliktig i henhold til personopplysningsloven § 31. Behandlingen tilfredsstiller kravene i personopplysningsloven.

Personvernombudets vurdering forutsetter at prosjektet gjennomføres i tråd med opplysningene gitt i meldeskjemaet, korrespondanse med ombudet, ombudets kommentarer samt personopplysningsloven og helseregisterloven med forskrifter. Behandlingen av personopplysninger kan settes i gang.

Det gjøres oppmerksom på at det skal gis ny melding dersom behandlingen endres i forhold til de opplysninger som ligger til grunn for personvernombudets vurdering. Endringsmeldinger gis via et eget skjema, http://www.nsd.uib.no/personvern/meldeplikt/skjema.html. Det skal også gis melding etter tre år dersom prosjektet fortsatt pågår. Meldinger skal skje skriftlig til ombudet.

Personvernombudet har lagt ut opplysninger om prosjektet i en offentlig database, http://pvo.nsd.no/prosjekt.

Personvernombudet vil ved prosjektets avslutning, 01.06.2017, rette en henvendelse angående status for behandlingen av personopplysninger.

Vennlig hilsen

Kjersti Haugstvedt

Amalie Statland Fantoft

Kontaktperson: Amalie Statland Fantoft tlf: 55 58 36 41
Vedlegg: Prosjektvurdering

Dokumentet er elektronisk produsert og godkjent ved NSDs rutiner for elektronisk godkjenning.

NSD – Norsk senter for forskningsdata AS Harald Hårfagres gate 29 Tel: +47-55 58 21 17 nsd@nsd.no Org.nr. 985 321 884
NSD – Norwegian Centre for Research Data NO-5007 Bergen, NORWAY Faks: +47-55 58 96 50 www.nsd.no

Personvernombudet for forskning

Prosjektvurdering - Kommentar

Prosjektnr: 50358

INFORMASJON OG SAMTYKKE
I følge meldeskjemaet skal deltakerne i studien informeres skriftlig og muntlig om prosjektet og samtykke til deltakelse. Informasjonsskrivet er godt utformet.

METODE
I meldeskjemaet har dere krysset av for at dere skal innhente personopplysninger gjennom intervjuer og observasjon. Personvernombudet forutsetter at alle som observeres informeres og samtykker til deltagelse, dersom det skal registreres personidentifiserende opplysninger fra observasjon.

INFORMASJONSSIKKERHET
Personvernombudet legger til grunn at dere behandler alle data og personopplysninger i tråd med Universitetet i Bergen sine retningslinjer for innsamling og videre behandling av forskningsdata og personopplysninger.

PROSJEKTSLUTT OG ANONYMISERING
I informasjonsskrivet har dere informert om at forventet prosjektslutt er 01.06.2017. Ifølge prosjektmeldingen skal dere da anonymisere innsamlede opplysninger. Anonymisering innebærer at dere bearbeider datamaterialet slik at ingen enkeltpersoner kan gjenkjennes. Det gjør dere ved å slette direkte personopplysninger, slette eller omskrive indirekte personopplysninger og slette digitale lydopptak