Agile Control and Collaboration in Large Organizations

JACOB SIEVERS
Agile Control and Collaboration in Large Organizations

by

Jacob Sievers
Agil styrning och samarbete i stora organisationer

Jacob Sievers
Abstract
Over the past decades, a group of so called “Agile” software development methodologies have been gaining in popularity due to their ability to deliver higher value to customers at a lower cost and in a shorter amount of time. While these methodologies were developed in settings with smaller projects and teams, these are still gaining traction even among large organizations, although at a slower rate.

A key impediment towards agile adoption in larger organizations is that there are many unique challenges associated with it. These range from the lower access which teams have to customers, to the decreased ability to have collocated teams, and therefore gain the productivity benefits which it brings. To study these challenges and many more, a study was performed at the Stockholm and Munich offices of a large, multinational technology corporation.

The results in this study suggest that there are many significant challenges prevalent in large organizations using agile software development, which were grouped into three main areas: “Communication and collaboration”, “Planning and organizing” and “Tools”. To deal with these challenges, these organizations can take a number of measures. This study suggests that the role of the Scrum Master is weakened, and therefore needs to be strengthened in order to facilitate the used processes, and therefore gain the most benefit from them. Furthermore, face-to-face communication is underutilized, curbing productivity and increasing misunderstanding. Encouraging this will likely bring with it significant benefits. Finally, there is an over-emphasis on looking at non-qualitative metrics in monitoring and communication. Therefore, large organizations using agile methodologies should to a greater degree strive to evaluate, discuss and learn from the customer, and the outcome of projects, rather than the direct output of employees.

Key-words
Agile Software Development, Scaled Agile, Agile Collaboration
Sammanfattning
Över de senaste decennierna har en grupp av s.k. “Agila” mjukvaruutvecklingsmetodiker blivit mer och mer populära på grund av dess förmåga att leverera högre värde till kunder till en lägre kostnad och på kortare tid. Även om dessa metodiker utvecklades i miljöer med mindre projekt och team så har även många stora organisationer anammat dem.

Ett hinder för att stora organisationer ska använda dessa metodiker är de många unika utmaningar som finns för stora organisationer som vill det. Dessa inkluderar allt från lägre tillgång till kunder för team till en minskad förmåga att ha medarbetare på samma plats i världen, och därigenom få de produktivitetsökningar som det för med sig. För att studera dessa utmaningar och fler gjordes en fallstudie på Stockholm- och Münchenkontoren för ett multinationellt teknikföretag.

Resultaten från studien visare att det finns många stora utmaningar i stora organisationer som använder sig av agil mjukvaruutveckling. Dessa grupperades in i tre kategorier ”Kommunikation och samarbete”, ”Planering och organisering” och ”Verktyg”. För att ta itu med dessa utmaningar kan dessa organisationer göra ett antal saker. Denna studie visar att Scrum Masterns roll är försvagad, och måste därför stärkas för att facilitera och få ut så mycket som möjligt av den använda metodiken. Fortsättningsvis används ansikte mot ansikte -kommunikation i för liten utsträckning, vilket minskar produktivet och för med sig ökad risk för missförstånd. Att uppmuntra till mer av denna typ av kommunikation skulle därför föra med sig stora fördelar. Slutligen ligger en för stor betoning på användning av icke-kvalitativa mått. Stora organisationer borde därför i större utsträckning sträva efter att utvärdera, diskutera och lära sig av kunden och dess nödhet med resultatet av projekt, istället för bara den direkta produkten av medarbetares arbete.

Nyckelord
Agil mjukvaruutveckling, Storskalig agil projektmetodik, Agilt samarbete
# Table of Contents

Table of figures ...................................................................................................................... iv  
Table of tables ....................................................................................................................... v  
Table of abbreviations .......................................................................................................... vi  
Acknowledgements ................................................................................................................ vii  

## 1. Introduction ....................................................................................................................... 1  
### 1.1 Background ..................................................................................................................... 1  
#### 1.1.1 Problematization ...................................................................................................... 2  
### 1.2 Purpose and Research Questions .................................................................................. 2  
### 1.3 Delimitations .................................................................................................................. 2  
### 1.4 Outline of the Thesis ..................................................................................................... 3  

## 2. Methodology ....................................................................................................................... 4  
### 2.1 Research Design .......................................................................................................... 4  
#### 2.1.1 Phase 1 ..................................................................................................................... 5  
#### 2.1.2 Phase 2 ..................................................................................................................... 5  
### 2.2 Literature Study ........................................................................................................... 6  
### 2.3 Empirical Data & Analysis ........................................................................................... 6  
#### 2.3.1 The Case Study ....................................................................................................... 6  
#### 2.3.2 Interviews ............................................................................................................... 7  
#### 2.3.3 Data Records .......................................................................................................... 8  
#### 2.3.4 Documentation ........................................................................................................ 8  
#### 2.3.5 Observations .......................................................................................................... 9  
#### 2.3.6 Analysis .................................................................................................................. 9  
### 2.4 Quality of the Study ..................................................................................................... 10  
#### 2.4.1 Validity and Reliability ........................................................................................... 10  
#### 2.4.2 Generalizability ...................................................................................................... 10  
#### 2.4.3 Ethics ...................................................................................................................... 11  

## 3. Literature and Theory ....................................................................................................... 12  
### 3.1 Software Development ................................................................................................. 12  
### 3.2 Agile Software Development ....................................................................................... 12  
### 3.3 Collaboration and Communication in Software Development .................................... 13  
#### 3.3.1 Agile Collaboration and Communication ............................................................... 13  
#### 3.3.2 Customer Collaboration and Feedback ................................................................... 14  
### 3.4 Continuous Integration (CI) ....................................................................................... 15
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.5 Scrum</td>
<td>16</td>
</tr>
<tr>
<td>3.5.1 Principles, Roles and Artifacts of Scrum</td>
<td>16</td>
</tr>
<tr>
<td>3.5.2 Artifacts</td>
<td>16</td>
</tr>
<tr>
<td>3.5.3 Roles</td>
<td>17</td>
</tr>
<tr>
<td>3.6 Scaled Agile</td>
<td>18</td>
</tr>
<tr>
<td>3.6.1 Communication and Collaboration</td>
<td>18</td>
</tr>
<tr>
<td>3.6.2 Other Challenges</td>
<td>19</td>
</tr>
<tr>
<td>3.6.3 Practices</td>
<td>19</td>
</tr>
<tr>
<td>3.7 Agile &amp; Stage Gate Hybrids</td>
<td>20</td>
</tr>
<tr>
<td>3.8 The Iron Triangle</td>
<td>20</td>
</tr>
<tr>
<td>3.9 Software Metrics</td>
<td>21</td>
</tr>
<tr>
<td>3.9.1 Velocity and Effort</td>
<td>22</td>
</tr>
<tr>
<td>3.9.2 Outcome vs. Output</td>
<td>22</td>
</tr>
<tr>
<td>4. The Case Company</td>
<td>23</td>
</tr>
<tr>
<td>4.1 Corporate Profile Summary</td>
<td>23</td>
</tr>
<tr>
<td>4.1.1 Acquisition of the Stockholm Office</td>
<td>23</td>
</tr>
<tr>
<td>4.2 Organization</td>
<td>23</td>
</tr>
<tr>
<td>4.2.1 Business Areas</td>
<td>23</td>
</tr>
<tr>
<td>4.2.2 Development Centers</td>
<td>24</td>
</tr>
<tr>
<td>4.2.3 Functions and Products at CCN</td>
<td>24</td>
</tr>
<tr>
<td>4.3 Projects at NDC</td>
<td>25</td>
</tr>
<tr>
<td>4.3.1 R&amp;D Processes</td>
<td>25</td>
</tr>
<tr>
<td>4.3.2 Product Lifecycle Management (PLM)</td>
<td>26</td>
</tr>
<tr>
<td>5. Empirical Findings</td>
<td>28</td>
</tr>
<tr>
<td>5.1 First Order Findings</td>
<td>28</td>
</tr>
<tr>
<td>5.1.1 Collaboration &amp; Communication</td>
<td>28</td>
</tr>
<tr>
<td>5.1.2 Planning &amp; Organizing</td>
<td>30</td>
</tr>
<tr>
<td>5.1.3 Important Tools for the Development Process</td>
<td>32</td>
</tr>
<tr>
<td>6. Discussion</td>
<td>34</td>
</tr>
<tr>
<td>6.1 Common Themes in Empirical Findings</td>
<td>34</td>
</tr>
<tr>
<td>6.1.1 Conflict Between Standardization and Autonomy</td>
<td>34</td>
</tr>
<tr>
<td>6.1.2 Conflict Between Stability and Agility</td>
<td>35</td>
</tr>
<tr>
<td>6.1.3 Organizational Distance</td>
<td>37</td>
</tr>
<tr>
<td>6.2 Collaboration &amp; Communication</td>
<td>38</td>
</tr>
</tbody>
</table>
6.2.1 Collaboration and Communication in Large Organizations ........................................... 38
6.2.2 Customer Feedback ................................................................................................. 39
6.3 Planning & Organizing ............................................................................................... 41
   6.3.1 Traditional vs. Agile Methodologies ................................................................. 41
   6.3.2 Scaled Agile ........................................................................................................ 41
6.4 Metrics ......................................................................................................................... 43
   6.4.1 Velocity ............................................................................................................. 43
   6.4.2 Utilize Measurement of Outcome ...................................................................... 43
7. Conclusions .................................................................................................................... 44
   7.1 Summary of Issues ................................................................................................. 44
   7.2 Research Questions .............................................................................................. 44
      7.2.1 Encourage more frequent face-to-face communication ................................ 45
      7.2.2 Strengthen the SM’s role .............................................................................. 45
      7.2.3 Evaluate, discuss, and learn from project outcomes ...................................... 45
   7.3 Sustainability ........................................................................................................... 46
8. Bibliography ................................................................................................................... 47
9. Appendix A: List of interview subjects ...................................................................... 51
   9.1 Phase 1 .................................................................................................................. 51
   9.2 Phase 2 .................................................................................................................. 51
10. Appendix B: The Product Lifecycle Management model ........................................... 52
Table of figures

Figure 1: The research process.................................................................5
Figure 2: Visualization of the Scrum process..........................................16
Figure 3: The Iron Triangle....................................................................21
Figure 4: Business areas of the case company.......................................24
Figure 5: Different project organizations at CCN.................................26
Table of tables

Table 1: Agile communication research gaps (reproduced from Hummel et al. 2013) .............. 14
Table 2: Collaboration challenges ........................................................................................................... 19
Table 3: Agile practices that scale, according to Leffingwell (2007) ...................................................... 20
Table 4: PLM phases (main R&D phases italicized) .................................................................................. 26
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eSIM</td>
<td>Embedded Subscriber Identification Module</td>
</tr>
<tr>
<td>CCN</td>
<td>Case Company Nordic</td>
</tr>
<tr>
<td>CCMU</td>
<td>Case Company Munich</td>
</tr>
<tr>
<td>IID</td>
<td>Iterative and Incremental Development</td>
</tr>
<tr>
<td>KPI</td>
<td>Key Performance Indicator</td>
</tr>
<tr>
<td>KLOC</td>
<td>Kilo-Lines Of Code</td>
</tr>
<tr>
<td>(S)LOC</td>
<td>(Source) Lines Of Code</td>
</tr>
<tr>
<td>NDC</td>
<td>Nordic Development Center</td>
</tr>
<tr>
<td>PO</td>
<td>Product Owner</td>
</tr>
<tr>
<td>R&amp;D</td>
<td>Research and Development</td>
</tr>
<tr>
<td>SAFe</td>
<td>Scaled Agile Framework</td>
</tr>
<tr>
<td>SM</td>
<td>Scrum Master</td>
</tr>
<tr>
<td>TPO</td>
<td>Technical product owner</td>
</tr>
</tbody>
</table>
Acknowledgements

As the final part of my studies at KTH Royal Institute of Technology, this thesis not only represents the challenging, but rewarding work done during the thesis itself, but also what I have learned during the years leading up to it. This would not have been possible without a number of people whom I would like to thank.

Firstly, I would like to thank the company investigated in this study and its employees who have been nothing but supportive in my work during the spring. I would like to give an especially large thanks to everyone who agreed to participate in interviews, as well as my supervisor at the company and the Head of R&D who were especially helpful during my work. Each and every one of you helped shape this thesis.

I would also like to thank my supervisor, Anna Jerbrant, for her always valuable, plentiful and honest feedback. I could not have hoped for a better supervisor! Additional thanks also go to my seminar group and its group leader, Mats Engwall, for great feedback and discussions.

Finally, I would like to thank my family and friends who stood by me through the many ups and downs through the years which led me to this thesis. I could not have done it without you!

Thank you all!

Jacob Sievers

Stockholm, June 2016
1. Introduction

This chapter introduces you to the background of the issues being studied in this thesis. In the first section, the background and the case being studied is described, while the second section describes the core issues being faced as well as the purpose and research questions in this study.

1.1 Background

During the last half century, there has been a large shift in the kinds of project models used in different industries. While older, more mature industries have continued to rely heavily on models related to the so-called Waterfall model, IT and software development has moved towards a new breed of models known as “Agile” project models. These models are different in many ways from more traditional development methodologies. They are less rigid in their structure and rely more heavily on iteratively working on modular activities. One example of an agile methodology is Scrum, first described by Takeuchi & Nonaka (1986). Though Scrum was not the first methodology to agile principles, it described many of the practices common to the loose group of development processes known as agile methods (Beck, et al., 2001) (Fowler & Highsmith, 2001). Agile practices include self-organizing teams and more subtle control (Takeuchi & Nonaka, 1986). While these methods in many cases are a good way of improving performance, they also come with their own set of challenges (Gandomani, Zulzalil, Ghani, Sultan, & Nafchi, 2013).

Many challenges faced by organizations using agile practices are unique or more prevalent in larger organizations and projects (Leffingwell, 2007). This is in part due to the fact that the core tools, including Scrum, are designed for smaller teams and projects, which in scrum’s case means 5-9 people (Schwaber, 2004). Being designed first and foremost for small-scale settings results in many challenges for larger organizations using agile development. A significant part of the productivity increase offered by agile practices comes from frequent face-to-face communication, both in formal settings such as daily meetings and retrospective meetings, as well as in more informal settings (Leffingwell, 2007; Pikkarainen, Haikara, Salo, Abrahamsson, & Still, 2008). Additionally, there is a large reliance on frequent feedback from customers and/or users (Evbota, Knauss, & Sandberg, 2016). However, both face-to-face communication and frequent feedback is made more difficult as the number of people involved in a given project increases (Leffingwell, 2007; Barlow, et al., 2011). Furthermore, regardless of the work processes used, large organizations face challenges in communication, collaboration and control as well. These include delayed decision making, depressed motivation/morale, conflict and distortion of information (Child, 2015). To study these challenges and more, a case company was chosen for this study.

The company being investigated in this study, hereinafter referred to as “the case company”, is a multinational technology corporation with their headquarters in Munich, Germany. They have several business areas, where a growing one is the development of the “eSIM”, a type of SIM-card which, in layman’s terms, provides the same service as a SIM-card without the need for a physical card tied to a particular carrier and plan. With a device supporting eSIM, you can switch carrier or type of subscription plan on demand, without switching the physical card.
While development within the Stockholm office of the case company, which is the focal point of this study, is agile and uses the Scrum methodology, there is an overlying, traditional rigid project model. Among other things, this means the product releases follow a strict versioning and time schedule, and that requirements are immutable as development starts. To gain a complete picture of the R&D processes, many places in the case company were investigated, including, but not limited to developers, product management and internal stakeholders. Interviews were held both in the Stockholm office, belonging to the Nordic division of the case company (hereinafter referred to as CCN), and in the Munich head office for comparison and context (hereinafter referred to as CCMU).

1.1.1 Problematization
In modern R&D organizations performing software development, there are several possible strategies to use. Many organizations, large and small, choose to move more towards an agile, flexible approach, with self-managing, cross-functional teams, iterative development and changing requirements. While this approach has many strengths, there are challenges associated with it as well. Tools need significant adaption in order to facilitate the agile processes, customers can be uncooperative, making it harder to deliver valuable software, and estimation of time and cost becomes notoriously difficult. Furthermore, large organizations choosing an agile approach face additional challenges which make successful agile adoption difficult. This is in part due to the fact that agile methodologies generally were developed in small organizations, for smaller projects. Factors causing challenges for large organizations includes a larger geographical distribution, making efficient communication difficult. It also includes more conflict and a lower degree of trust.

1.2 Purpose and Research Questions
The purpose of this study is to investigate and find areas of improvement in communication, collaboration and control in large organizations using agile software development methodologies.

To help fulfill the objective of this study, the following research questions have been formulated:

RQ 1: What areas need support in large organizations in order for agile R&D processes to succeed?
RQ 2: What can be done to provide support where it is lacking?

1.3 Delimitations
This study is based primarily on a case study, hence, the data gathered and analyzed is limited to interviews and observations performed at and documentation gathered at the case company. Within the company, I have delimited myself to looking at primarily the R&D department and its interaction with product and project management. In addition, due to the time constraints placed on this study as a master’s thesis, as well as on me as a sole researcher, this study does not dive deeply in the individual identified issues, but rather discusses them together in order to answer the research questions.
1.4 Outline of the Thesis

Chapter 2
Methodology
The research methodology behind all significant parts of the study is presented.

Chapter 3
Literature and Theory
The literature and previous work which is used as basis for the discussion and conclusions at the end of the thesis is presented.

Chapter 4
The Case Company
In this chapter, the company where this thesis’ case study was performed, is presented.

Chapter 5
Empirical Findings
The main findings of this study are presented in this chapter. Most of the findings were compiled during the performed case study.

Chapter 6
Discussion
Based on the theory presented in chapter 3 and the empirical data from chapter 4 and 5, this chapter discusses issues relating to the findings and makes recommendations.

Chapter 7
Conclusions
The main takeaways and recommendations from the discussion, and final thoughts from the study, are presented.
2. Methodology

In order to produce a satisfying answer to the research questions presented above, certain choices regarding methodology have been made. These are presented below. The main part of this study will be a qualitative case study performed at the case company’s Stockholm and Munich offices.

2.1 Research Design

Since the purpose of this study is to investigate how to work with feedback and communication in a R&D organization using both agile and more traditional methodologies, a choice could be made between different research methodologies. The first main choice was whether to adopt a qualitative or quantitative research strategy. According to Skärvad & Lundahl (2016), the qualitative studies are appropriate when describing a sequence of events or processes (in a broad sense). This seemed to fit well with the purpose of this study, which led to adopting a qualitative research strategy throughout the study.

Performing a qualitative study does not, despite what the name suggest, mean that only qualitative data can be used (Skärvad & Lundahl, 2016). At an early point, both qualitative interviews and quantitative data gathering in the form of a survey were considered as the main data gathering method in this study. However, mainly due to time and personnel constraints as a lone researcher, the amount of empirical data gathering had to be limited in order to have sufficient time devoted to analysis at the end of the study. According to David & Sutton (2016), quantitative data lacks the ability to explain causal relationships in social science research. Additionally, according to Denscombe (2007), interviews are particularly good at giving in-depth, detailed information, and valuable insights. This led to qualitative interviews being chosen as the primary source of empirical data in this study.

According to Skärvad & Lundahl (2016), an exploratory study is appropriate when attempting to specify the problematization as well as assisting in developing a more precise research plan. Due to the wide initial problematization and high complexity of the environment being studied (projects and R&D at the case company), an exploratory first phase of the study was deemed appropriate to perform for this study. Accordingly, the study was divided into two parts, one exploratory first phase, and a second phase where some additional data gathering, and most of the data analysis was performed. Also, most of the results and conclusions were developed in this phase. On an abstract level, the purpose of the first phase was to identify key challenges in the R&D processes at CCN, and find common denominators, while the second phase’s purpose was to delve deeper into these challenges.

Initially, there was concern that any conclusions using purely theory would not be applicable in a real scenario. According to Denscombe (2007) a case study is appropriate when an issue needs to be studied in depth, and the conclusions need to be applicable in complex, real situations. Therefore, a case study methodology was employed, which is described further in section 2.2. According to Blomkvist & Hallin (2015), this method is appropriate when a phenomenon has already been demonstrated, but where research is limited, which is another reason that this approach was chosen.
2.1.1 Phase 1

Early in the project, it was established that the initial problem formulation was wide, somewhat ill-defined and may not even address the core issues that CCN are facing. As mentioned, an exploratory study method was used for this reason. Within this exploratory study, the main form of data gathering used was open-ended, or unstructured interviews. The procedure used in this study is described further in section 2.3.2.

All of the interview subjects during phase 1 were so called “direct stakeholders” to the study, and the answers reliability should therefore continuously be questioned (Skärvad & Lundahl, 2016). An effort was made to do just this. Though interviewees were direct stakeholders, an effort was made to interview as many stakeholder groups as possible. This was done in order to be able to establish which facts were unambiguous and uncontroversial during the analysis, as well as find out which were more unique to one role (Skärvad & Lundahl, 2016).

2.1.2 Phase 2

In phase 2 of the study, semi-structured interviews were performed alongside study of theory in order to further develop the generated empirical evidence with the help of previous theory in the subject.

The first part of phase 2 was a short period of theory study, where an overview of literature in the relevant areas was produced. This was done mainly to point the following interviews in the right direction, and ask better questions.

In the data gathering part of phase 2, several of the interviews were conducted in CCN’s development center in Stockholm, as with phase 1 due to the great accessibility. However, there were also interviews conducted at the case company’s Munich office. The main reason for choosing this office for additional interviews is Munich’s position as the main headquarters of the case company. The reason that these interviews were done in phase 2 was that they could benefit from exploratory data gathering during phase 1. Since these trips were to an office which generally does not use agile methodologies, and has a different organizational structure (CCN5) they were focused on gathering data on how CCN fits into the case company in general.
2.2 Literature Study
A key component in this study was the literature which was studied intermittently during the beginning of the study, and more intensely during the latter part, due to the abductive nature of this study.

Literature in the context of this study refers to the existing body of knowledge in the subjects studied, and can include any kind of secondary data relevant to the study (Collis & Hussey, 2014). According to Collis & Hussey (2014), the literature study can start as soon as a potential topic has been conceived, with the first step being defining the scope. After this, they recommend determining keywords and beginning the database search. The main keywords used in the literature search in this study are described at the end of this section. While Collis & Hussey (2014) regard textbooks as insufficient at a graduate level, Blomkvist & Hallin (2015) argue that the literature study can start with studying textbooks on the subject in order to gain a broad picture of a subject. In this study, Blomkvist & Hallin’s (2015) recommendation was followed, and more sources with original research, such as journal articles, were taken in as the study progressed. Furthermore, Blomkvist & Hallin (2015) recommend using the reference list from the book to find more sources. This was done for both textbooks and other literature in this study.

In this study, the search for literature began before the work with the thesis formally started, mostly with the search for literature regarding agile development as a broad subject. The literature study continued throughout the entire study, with a larger emphasis on it towards the end. The focus for the literature shifted during the study, starting with studying agile development and project management, and shifting towards studying the challenges which were identified in phase 1 in the latter parts of the study. The main tools used to search for literature were KTHB’s main search engine, Primo (which is linked to multiple databases), and Google Scholar. Besides those sources, multiple articles and other resources were acquired through recommendations by experts at the case company and scholars in adjacent fields at KTH.

Main keywords used in different combinations to search for literature during the study:

Agile, Software development, Scaled agile, Collaboration, Communication, Control, Software metrics, Scrum

2.3 Empirical Data & Analysis
In this section, the data gathering methodology and the analysis is described and discussed, especially regarding the appropriateness of the methodology.

2.3.1 The Case Study
The main research methodology used in phase 1 and 2 was the case study. Due to the fact that the presented problematization was adapted from a problem CCN is facing, there was a need to study the context in which CCN is active, and the study was designed with this in mind. According to Blomkvist & Hallin (2015), a case study is often used in social-studies research due to the high amount of complexity from real world problems that can be captured. This can be compared to e.g. a survey, where you on one hand gain highly reliable data, but on the other hand you fail to gain the rich amount of data that a case study provides (Blomkvist &
Hallin, 2015). For these reasons, the case study was employed as a research method in both phase 1 and 2 of the study.

According to Yin (2003), the case study can be used for many different purposes, whether it is exploratory, as in phase 1, or more descriptive, as in the phase 2. Furthermore, he argues that a case study has a distinct advantage against other research strategies when “a how, or why question is being asked about a contemporary set of events, over which the investigator has little or no control”. This corresponds well with this study, where the main aim is to understand the context of a contemporary technology corporation.

2.3.2 Interviews
According to Yin (2003), interviews are one of the most important sources for case studies. In the phase 1, interviews were conducted in CCN’s development center in Stockholm due to the high degree of accessibility, while phase 2 had interview subjects in Stockholm and Munich. For further details regarding the interview subjects, refer to “Appendix A: List of interview subjects”. The interviews during phase 1 were open-ended, while in the phase 2, they were instead semi-structured. In total, nine interviews were conducted during phase 1, and seven during the phase 2. The subjects that were chosen during phase 1 had several different roles, ranging from Developer to Head of Service Operations. During phase two, the roles of the interviewees were just as varied, and also included interviews conducted at the Munich office of the case company.

2.3.2.1 Open-ended interviews
Open-ended interviews can be used to explore a subject field, as recommended by Blomkvist & Hallin (2015). Instead of following pre-defined questions, open-ended interviews can follow an overarching theme instead, more similar to a regular conversation than structured or semi-structured interviews (Blomkvist & Hallin, 2015). According to Skärvad & Lundahl (2016), unstructured interviews are suitable for studies with an explorative purpose, which is why this type of interview was chosen for phase 1.

The interviews performed during phase 1 were of a semi-standardized, unstructured nature. What this means is that while the interviews had several pre-defined talking points in common, while their order and whether they were included or not varied greatly between interviews (David & Sutton, 2016). Also, when an interesting subject was found, additional, improvised questions were asked.

2.3.2.2 Semi-structured interviews
Semi-structured interviews follow a more rigid regimen than open-ended ones. The structure is more rigid, which does not allow as much flexibility as open-ended interviews, and they are therefore not as suitable for exploring a subject. Semi-structured interviews offer a compromise between structured and unstructured interviews, and can therefore be used in order to allow interviewees to expand upon subjects of interest, while increasing the reliability compared to open-ended interviews (Denscombe, 2007) (Skärvad & Lundahl, 2016).

The interviews performed during the phase 2 of the study were of a semi-standardized, semi-structured nature. Unlike with the open-ended interviews, most questions were written beforehand, and were shared between interviewees, but since the interviewees had different
roles and experiences to share, the inclusion of questions and which subjects were discussed more in depth were still flexible.

2.3.2.3 Question types
According to Skärvad & Lundahl (2016), one can use background (introductory) questions to gain rapport with interview subjects before getting into the true subject of the interview. This was used during all interviews, and especially when I had not met the interviewee before. Since these interviews were conducted early during the study, and were during an exploratory study, many of the questions were so called “process questions”, meaning openly phrased questions which intend to give the interviewees room to bring up that which is truly important to them (Skärvad & Lundahl, 2016). Many questions were also of a probing nature, used to dig deeper into already mentioned subjects, or clarify previous remarks (Blomkvist & Hallin, 2015).

Besides the data itself, interviews were used for another extremely important purpose as well: referrals, which is recommended by Skärvad & Lundahl (2016). At the end of interviews, the subjects were asked if they had any recommendations on who else to interview, which many times put me in contact with people who I would not know to interview or could otherwise not reach. Also, when an interview subject was particularly knowledgeable in a subject, they were asked for literature and web referrals as well, which resulted in several sources which were used later during the study.

2.3.3 Data Records
To maintain an accurate representation of what was said in the interviews, all interviews in phase 2 and six in phase 1 were recorded. For the rest of the interviews, field notes were taken during the interview.

Audio recording was used to keep a record of the interviews for a number of reasons. The main reason is that audio recordings provide a fairly complete, permanent record of the speech occurring during the interviews (Denscombe, 2007). While video recordings could also have been used, according to Denscombe (2007), audio recordings provide enough data and reliability for practical purposes in research. Also, in many cases, video recordings cause extra disruption to the interview climate, which outweighs the advantages it has compared to audio recordings (Denscombe, 2007). Another significant advantage for audio recordings is that they lend themselves well to being checked by other researchers, as well as interviewees, in case statements made come into question (Denscombe, 2007).

As mentioned, a few interviews at the beginning of phase 1 were not recorded and instead used field notes. While this reduces the reliability of the recollection of the interview (Denscombe, 2007), this does not have a significant impact on this study as none of those interviews were used as the sole basis for any of the significant findings in this study. These interviews, and the field notes, were primarily used for guidance at the onset of this study.

2.3.4 Documentation
According to Denscombe (2007), documents as a source of data can be treated as a full alternative to questionnaires, interviews and observations. He argues that its strengths include the easy access to the data, the cost-effectiveness and the permanence of the data. The main
weaknesses are mostly surrounding reliability, which is discussed more in-depth in section 2.4.1.

The documentation used in this study mainly includes internal documents describing processes, methodologies and roles in different parts of the organization. The internal documents contained in this thesis are included with the permission of the case company.

2.3.5 Observations
Observations are a completely distinct form of data collection, relying on witnessing phenomena first hand, instead of, for example, getting a second-hand recount of an event through an interview (Denscombe, 2007). According to Yin (2003), observations are useful in providing additional information about the phenomenon being studied, adding a new dimension of understanding for the phenomenon under study, or the context surrounding it.

In this study, I spent the majority of my time at the case company, doing what Descombe (2007) calls “participant observation”. According to him, since my role as a researcher was known to everyone at the company, my role would be “participant as observer”, which has the advantage against other options that informed consent could be gained from subjects of observations. The major downside of this is methodology is that of bias being introduced, affecting the reliability of the date.

2.3.6 Analysis
Since both phases of the study used a qualitative case study as its main approach, with interviews as the main source of empirical data, similar approaches to analyzing the data could be taken.

To analyze qualitative data in the form of interviews, the data needs to be interpreted. According to Denscombe (2007), the process of interpreting data involves four main tasks:

• **Code the data**
  Attach tags or labels to the raw data, in this case interview transcripts and notes

• **Categorize these codes**
  Identify the tags and labels which can be categorized, and categorize them into different categories

• **Identify themes and relationships**
  Find different themes and relationships among the codes and categories which were provided in the previous steps

• **Develop concepts, provide generalized statements**
  Based on the identified codes, categories, themes and relationships, arrive at some generalized conclusions. These can come in the form of concepts or hypotheses, or even theories, should the data substantiate it.

All of these steps were performed in the analysis part of both phases of the study.
2.4 Quality of the Study

In this section, the impact of the methodology on the quality of the study is discussed. Also, the generalizability and ethical concerns made are discussed.

2.4.1 Validity and Reliability

When assessing the quality of scientific work, you often speak of validity and reliability (Blomkvist & Hallin, 2015). According to Skärvad & Lundahl (2016), validity is the absence of systematic measurement errors, and can be divided into inner and outer validity. Inner validity is when a source of data measures what it intends to measure, and outer validity is whether the source of data (the indicator) actually corresponds to the phenomenon being studied. According to Blomkvist & Hallin (2015), while validity is studying the right thing, reliability entails studying it in the right way. Skärvad & Lundahl (2016) describes it as "the absence of random error". Qualitative studies using an interpretivist paradigm, such as this one, tend to have a low reliability, instead, a higher emphasis is placed on whether the research is authentic and can be understood (Collis & Hussey, 2014).

The primary source of empirical data in this study, interviews, is known to have many aspects which increase the validity (Denscombe, 2007), therefore, the empirical data in this study is deemed to have a high validity. Documentary sources on the other hand are different. According to Denscombe (2007), these can never be accepted at face value. Since the documents in this study were directly from the case company, the risk for bias is high. For that reason, any documents used in this study were usually corroborated by another source and method or only used for understanding the context.

Observations are known to have high “ecological validity”, that is it has the potential to have high sensitivity to the context (Denscombe, 2007). On the other hand, according to Denscombe (2007) the reliability may be low, due to the high reliance on the ‘self’ in participatory observation, as well as the reliance on field notes.

One of the main reasons for using multiple methods (in this case interviews, documentation and observations) to gather data is that it enables so called methodological triangulation (Denscombe, 2007). According to Denscombe (2007), there are multiple reasons for using methodological triangulation, the main ones are:

1. It enables corroboration or questioning of findings by comparing data from different methods
2. Other methods can help complement existing findings with new information

In addition to methodological triangulation, this study made extensive use of informant triangulation, which involves checking statements with multiple sources (Denscombe, 2007). According to Denscombe (2007), this increases the validity of the gathered data. Since validity of a study pre-requires reliability (Blomkvist & Hallin, 2015), this also raises the validity of the study.

2.4.2 Generalizability

It is often questioned how generalizable the findings of case studies are (Denscombe, 2007). Since this study, as with many other case studies, deals with a complex environment and many
of the findings deal with that specifically, the generalizability of this study in general is deemed to be low. However, many of the individual findings are not as fully dependent on context, and even those that are can often be compared to other case studies, increasing the generalizability of those individual findings (Denscombe, 2007).

2.4.3 Ethics
During the whole study, great care was taken to operate ethically. In order to do this, three principles proposed by Denscombe (2007) were followed.

1. The interests of participants should be protected

Certain measures were taken in order to protect the interests of participants. All of the interviews were done in private, and were anonymized before being presented in any form. Interviewees were also informed about this beforehand in order to create a safe environment during the interview where the interviewee felt secure enough to speak freely.

2. Researchers should avoid deception or misrepresentation

Before the interviews, all participants were informed about the premise of the interview and how the data will be used in the study. Also, in order to avoid misrepresentation, recordings and transcripts or notes were kept for each interview in order to maintain an accurate representation of what was said in interviews.

3. Participants should give informed consent

All of the participants in the study participated voluntarily, and consent was obtained for all data used in this study. Should a participant wish to withdraw a certain part of the information which was given after consent, that request was granted as well. The slight exception to this rule is when participation of an individual was through an observation, which means that consent was difficult to gather from each individual participant. However, according to Denscombe (2007), this is an exception to the principle of informed consent.
3. Literature and Theory
This chapter describes the theoretical background for issues being studied in the empirical part of the study, chapters 4 and 5. These are later brought up and discussed in relation to each other in chapter 6.

3.1 Software Development
Computers, and machines in general, can be said to be made up of two broad parts: hardware and software. Hardware is the tangible, physical part of a machine (O’Reagan, 2012), which is what most people visualize when they think of a computer. Software is the intangibles of a machine, which are made by one or more programmers (O’Reagan, 2012).

According to Buxmann (2013), the software industry can be said to be fundamentally different from other industries in several ways. For instance, the products themselves have the property that they can be reproduced at a cost close to zero without any loss in quality. The practice of software development also has some defining characteristics. With the help of internet, teams do not have to be co-located, and can work anywhere, while distributing their software to next to any customer in the world (Buxmann, 2013).

Buxmann (2013) argues that software development can follow several different strategies. In the beginning, it resembled a craft and followed strictly sequential stages, which was known as the “Stagewise model”. As software became larger, and more costly, this became unmanageable. Hence, a stricter plan based approach, usually referred to as the “Waterfall model” was developed. The waterfall model was the dominant software development approach for most of the 20th century, but gained more criticism as software development evolved. According to Buxmann (2013), these criticisms include:

- Plans become obsolete quickly
- There is a large overhead cost not directly related to software development
- The human factor is not paid enough attention to
- Testing the software happens too late in the process
- Mistakes are not treated as opportunities to learn for future projects

As a response to these criticisms, a set of different methodologies known as “Agile” methodologies were developed (Buxmann, 2013).

3.2 Agile Software Development
Today, agile software development is a multi-faceted discipline with many flavors of the methodology available. While it is tough to define exactly what unifies every agile software development methodology, they roughly follow the guidelines proposed by Beck et al. (2001) in “Manifesto for Agile Software Development”. The core of these practices could be said to revolve around creating and adapting to change while delivering products of high quality through simple work processes. At the same time, agile methods must increase, and not decrease the perceived economy, quality or simplicity of the product (Dingsøyr, Dyba, & Moe, 2010). In general, they also succeed quite well at this, with a median improvement of 88% in productivity and 26% in cost (Rico, 2008). However, as with any large paradigm shift, changes
in development practices take up a great deal of time and resources (Moe, Dingsøyr, & Dybå, 2010). Additionally, Moe et al. conclude that the way in which agile practices are taken up is highly dependent on previous development practices at the company, and that specialization (which is often present in large organizations) has a detrimental effect on teamwork in general, but especially in agile software development projects.

While the agile manifesto was released in 2001 by K. Beck et al., it draws from several decades of work by the authors and many others. Iterative and incremental development (IID) can be seen as the roots of agile development, and grew from quality improvement processes used at Bell Labs in the 1930's (Larman & Basili, 2003). The core idea was to follow a series of "plan-do-study-act" (PDSA) cycles in order to improve quality. PDSA was also later applied to software projects by, among others, T. Gilb, as well as Richard Zultner (Larman & Basili, 2003).

3.3 Collaboration and Communication in Software Development

Collaboration and communication in organizations has changed greatly over the last decades (Hersleb, 2007). Compared to previously, many organizations nowadays collaborate over huge distances, both organizationally and physically, with software facilitating their work (Olson & Olson, 2000). While it is common to work in this way for companies in all industries, it is extremely prevalent in software development, which according to Buxmann (2013) is more international than any other sector today.

Even with the rise of virtual collaboration tools, the gold standard when it comes to communication was, and is collocated, synchronous (face-to-face) communication (Nardi & Whittaker, 2002). Olson (2000) argues that there are many characteristic aspects of face-to-face communication which even future technology will not be able to help with. Agile software development agrees with this, proposing that “The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.” (Beck, et al., 2001). Schwaber (2007) also raises face-to-face communication as the best form of communication, and goes as far as raising increased collocation and more frequent face-to-face communication as effective alternatives to offshore development due to their productivity increasing effects. Korkala et al. (2006) and Mishraa et al. (2012) concur with this, raising face-to-face communication as the most effective communication method since it keeps the distortion and filtering to a minimum. Besides this, informal face-to-face communication is frequently raised as an efficient method of lowering the cost of communication, which grows quickly as the organization size increases.

3.3.1 Agile Collaboration and Communication

Though face-to-face communication is important in agile practices, and software development in general, codified (written-down) knowledge is extremely important as well. One of the key areas where this can be seen is in software requirements, or in the case of scrum, user stories and the backlog. Pikkarainen, Haikara, Salo, Abrahamsson, & Still (2008) observed in one case study that constant updates to the backlog, and especially adding features/stories without enough of a technical review had a negative impact on the project. They observed that the point when the backlog became unwieldy in the studied project was when the number of
features grew from dozens to hundreds. Additionally, developers in this project noted that the
time devoted to sprint planning in this case was too short, and the backlog not properly
organized, which likely increased the communication problems further.

Despite the abundance of research on communication in general, some of which is described
in short above, Hummel, Rosenkranz and Holten (2013) argue that there are many research
gaps regarding communication’s role in agile software development. This is due to the fact that
extremely few studies open up the communication process. The identified gaps are in three
categories which they refer to as “Input”, “Process” and “Output” (shown in Table 1). Furthermore,
they argue that there is evidence to suggest that communication is a critical
success factor of agile software projects, and that the higher the communication frequency, the
more productive the project.

Table 1: Agile communication research gaps (reproduced from Hummel et al. 2013)

<table>
<thead>
<tr>
<th>Category</th>
<th>Research Gaps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>What conditions surround and what factors on the environmental, organizational, group, and individual level impact on successful communication in agile software development?</td>
</tr>
<tr>
<td></td>
<td>What is the impact of the project domain and software development context (factors on the environmental, organizational, group, and individual level) on the use and effect of communication mechanisms within agile software development teams?</td>
</tr>
<tr>
<td>Process</td>
<td>What is the impact of the Scrum practices on communication informality, frequency, and quality compared to traditional, non-agile software development teams? (Does the use of Scrum practices lead to more informal and better communication?)</td>
</tr>
<tr>
<td></td>
<td>What is the impact of XP practices on communication informality, frequency, and quality compared to traditional, non-agile software development teams? (Does the use of XP practices lead to more informal and better communication?)</td>
</tr>
<tr>
<td>Output</td>
<td>What are the implications of the changing communication paradigm of agile software development for the development outcome?</td>
</tr>
<tr>
<td></td>
<td>Are agile software development teams more successful than traditional, non-agile software development teams due to improved communication? (Does more and better communication lead to better outcomes?)</td>
</tr>
</tbody>
</table>

A counterpoint to the above stated positive effects of frequent, close communication is that for
certain individuals, too much informal communication is distracting, which can cause
challenges for them in an agile project (Adolph, Kruchten, & Hall, 2012). This can be especially
prevalent with team members in possession of weak communication skills (Conboy, Coyle,
Wang, & Pikkarainen, 2011).

3.3.2 Customer Collaboration and Feedback
In agile software development, the role of the customer is critical, which can be seen in The
Agile Manifesto (Beck, et al., 2001) as well as empirical studies, such as the one by Hoda, Noble,
& Marshall (2011). Furthermore, this is not only implied by literature, or a few instances of
anecdotal evidence, but by quantitative studies. Kupiainen et al. (2015) claim in their literature study that “[agile software development] projects which were projects that were said to be definitely successful measured customer satisfaction often or always. Also, the more often customer satisfaction was measured, the more likely it was that the project would have good code quality, and the project would succeed.”.

Pikkarainen, Haikara, Salo, Abrahamsson, and Still (2008) argue that agile practices have several effects on external communication, especially with customers. The sprint review was raised as a great tool for gaining immediate feedback, not just regarding requirements “but also what was on the stakeholders’ minds and what they were thinking about in terms of needs for the service itself”. However, there were negative aspects as well. In one of the case studies described in the article, though there was significant time devoted to the customer during meetings, there was not enough time devoted to technical requirements analysis, and especially discussing it with the customer during the latter stages of the project when it had grown more complex.

Hoda, Noble, and Marshall argue in their aptly titled article “The impact of inadequate customer collaboration on self-organizing Agile team”, that lack of customer involvement in agile development leads to many challenges. These include a “Pressure to overcommit”, “Problems in Gathering and Clarifying Requirements”, “Problems in Prioritizing Requirements”, “Problems in Securing Feedback”, “Loss of Productivity”, and sometimes “Business Loss”. Furthermore, they argue that there are many possible levels of customer involvement in agile software development, and that knowing where on the spectrum ones company is can help in addressing the common problems associated with that level of involvement. Additionally, they argue that the customer has a responsibility in projects with an agile development component, to ensure the success of a project.

### 3.4 Continuous Integration (CI)

A set of tools and processes which are repeatedly brought up in relation to agile methodologies is “Continuous Integration” (CI) (Leffingwell, 2007; Cohn, 2010; Leffingwell, 2011). Cohn (2010) describes CI as “integrating new or changed code into an application as soon as possible and then testing the application to make sure that nothing has been broken”. Holck & Jørgensen (2003/2004) instead defines it in terms of two fundamental rules. Firstly, developers have access to add contributions to a development version of the software at any time and secondly, developers are obligated to integrate their own contributions to that development version properly. In practice, CI is achieved with the help of different tools and software scripts which are used to perform the tasks required in order to integrate code quickly: build the software from code, run tests on that code, and notify the developer on whether the code is properly integrated (Cohn, 2010).

While there are undeniable strengths to using CI, such as reduced integration risks, higher motivation (from seeing working software) and a better ability to isolate possible errors (Holck & Jørgensen, 2003/2004), there are some risks as well. According to Holck & Jørgensen (2003/2004), one of the more prominent risks is a possible degeneration of software architecture. Furthermore, Cohn (2010) raises a few common objections towards investing in CI. Firstly, “maintaining a build server and all those tests takes time away from other work”,

15
where Cohn (2010) argues that there really is not a lot of more work required, since an automated testing environment is required anyway, which means that the only other additional overhead is the one required to maintain a build server. Secondly, “our system is too complex; it takes hours to run a full integration test- we can’t build continuously.”, where Cohn (2010) argues that you actually do not have to run all tests for a given component, and can instead partition the tests in order to run a more manageable number at once.

3.5 Scrum
First described by Takeuchi & Nonaka (1986), Scrum is a methodology which originated as a product development methodology, but has since become tightly associated with being used as an agile development methodology. Like its forerunner IID, one of the core tenets of scrum as used in software development, is to work iteratively on a product in short bursts of work, which in scrum are known as sprints (this can be compared to the marathon of traditional methodologies). Also, similar to the “plan-do-study-act” tenet of IID, one of the core mantras of Scrum is “inspect and adapt”, which is given great prominence through the heavy emphasis which is placed on review and reflection. This cycle of work and feedback allows a team to continuously improve not just the work process, but the product, ideally delivering a usable iteration of the software after each sprint (Schwaber, 2004).

[Diagram of Scrum process]

Figure 2: Visualization of the Scrum process.

3.5.1 Principles, Roles and Artifacts of Scrum
In scrum, there are a few core principles, roles and artifacts which most companies adopting this process invariably use.

3.5.2 Artifacts
There are several artifacts which are rudimentary to understanding Scrum as a methodology, some of which are shared with other agile processes, and some which are more unique.
3.5.2.1 The Cross Functional Team
The cross functional team is the fractal unit of a company using scrum (Leffingwell, 2007). It consists of members which can perform all the core tasks in development, which usually are development, building and testing (Leffingwell, 2011). While team members may have competence specialties within these, they may work in multiple areas over the course of development (Schwaber, 2004).

3.5.2.2 Sprint
The main unit of time in Scrum is the sprint, which should be a strict timebox in which work can occur (Cohn, 2010). While Schwaber (2004) suggests that the sprint should be 30 days, the length can vary, but should be kept constant (Cohn, 2010). The sprint starts with a sprint planning meeting, and completes with a

3.5.2.3 Product Backlog
The product backlog is used for requirements management, which in scrum means user stories. A user story is simply a requirement phrased into the way a user would describe it, for example: “I should be able to add a phone number to a customer” (Gustavsson, 2011). It is the responsibility of the product owner to keep the product backlog up to date (Cohn, 2010). The general rule is that there should only be one all-encompassing backlog for a product (Cohn, 2010), but it can also be broken into several team backlogs if there are multiple agile teams working on the same product through feature or component teams (see section 3.6.3) (Leffingwell, 2011).

3.5.2.4 Sprint Backlog
Before each sprint during a sprint planning, the team commits to completing a certain number of stories from the product backlog (Schwaber, 2004). These are added to the sprint backlog, and completed one by one as the sprint progresses.

3.5.3 Roles
Scrum generally recognizes two roles which are new to organizations not using scrum (Cohn, 2010). These are “Product Owner” (PO) and “Scrum Master” (SM).

3.5.3.1 The Product Owner
Schwaber (2004) defines the PO as “responsible for representing the interests of everyone with a stake in the project and its resulting system”. Cohn (2010) describes the PO’s main objective as pointing the team at the right target, while Schwaber (2004) describes it as maximizing the value of the product, and therefore the return on investment (ROI). While it would be difficult to provide a comprehensive list of all the PO’s responsibilities, Cohn (2010) describes two core objectives for the PO: providing vision, and providing boundaries. Providing vision entails having a clear vision in mind for the product, and communicating it to the team. Mainly through producing and prioritizing a product backlog.

3.5.3.2 The Scrum Master
While the PO points the team in the right direction, the SM facilitates the team in reaching that target as efficiently as possible (Cohn, 2010). The overlying responsibility for the SM is to make sure that the team follows the scrum process (Cohn, 2010). A core part of this is removing obstacles impeding the progress of the team (Schwaber, 2004). While the SM is in
charge of the process, they do not have any authority over team members. For example, according to Cohn (2010), they may tell team members “We should work with two week sprints from now on”, but not “You’re fired!”.

3.6 Scaled Agile
While far from all agile practitioners come from a small organization, agile principles typically call for the use of small teams. According to Schwaber (2004), synergy effects make Scrum teams exponentially more productive up to 7 people, give or take two, therefore this is the ideal team size. While large teams may include more diverse skills and experiences, there are many more advantages to using smaller teams. According to Cohn (2010), there is more constructive interaction, less coordination, more satisfaction to team members, less harmful specialization, and higher productivity in small teams. This does not mean that only small companies with teams of under 10 people can use scrum however, as multiple teams can work on projects together. However, there is doubt as to how well scrum performs as the number of teams increases (Laanti, 2014), and it is clear that scaling reduces the general productivity when using agile methodologies, such as Scrum (Schwaber, 2007).

When only a few teams are involved in development, the need for control functions is low, and teams can generally operate without the overhead of heavy weight project and team coordination, but as organizations grow, they tend to put in more control measures (Leffingwell, 2007). To add even further complexity to the matter, many organizations (including the case company) have a current organizational structure, culture and processes that need to be taken into account when building the organization around agile teams (CCN5) (Leffingwell, 2007). To address these problems, several models for so called “Scaled Agile” have been developed by practitioners (Laanti, 2014). One of the first models to establish itself among these is “SAFe” (Scaled Agile Framework) which has gained a following since its introduction in the Agile 2013 conference in August 2013 (Laanti, 2014).

The foundation for SAFe comes from Leffingwell’s earlier work, such as “Scaling Software Agility” (2007). The core thesis of the book is that most of the core methods used by common agile methodologies, such as Scrum and XP (eXtreme Programming), are not only usable at scale, but bring substantial value in the form of increases productivity and customer satisfaction, among other things. However, while this may be the case, Barlow et al. (2011) found "no complete success stories of agile life cycles in large organizations”.

3.6.1 Communication and Collaboration
An important part of projects, whether large or small, is communication. This is especially so for agile software development, where face to face communication is strongly preferred compared to other types of communication (Beck, et al., 2001). This creates challenges in large organizations using agile methodologies, since the number of communication interfaces grow linearly with organization size (i.e., the number of people you can interact with in an organization grows at the same rate as the number of people), while the number of potential relationships grows exponentially (Barlow, et al., 2011). A way to combat this is through practices which lessen the cost of communication, such as collocation of people who need to communicate, and the availability of communication media (Barlow, et al., 2011). Barlow et al. (2011) also emphasize that not having clear and predictable roles and channels of
communication increases the number of potential channels a given person may have to communicate through.

Evbota et al. (2016) identified several key challenges in collaboration in the context of large-scale agile practitioners. The identified themes were “the ability to estimate, prioritize, and plan, the context of planning in terms of team build-up, work environment, and team spirit, and finally the ceremony agreement”. These are described in Table 2.

Table 2: Collaboration challenges

<table>
<thead>
<tr>
<th>Theme</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimation</td>
<td>Making long term estimations is extremely difficult</td>
</tr>
<tr>
<td>Prioritization</td>
<td>Lack of a shared vision among stakeholders can lead to disagreements on priorities</td>
</tr>
<tr>
<td>Planning</td>
<td>Includes lack of clarity regarding requirements, the unclear role of the operational product manager and how teams should be involved in planning, long and short term</td>
</tr>
<tr>
<td>Team build-up</td>
<td>The capabilities and special knowledge on a team that is built up over time is a critical resource which needs to be nurtured</td>
</tr>
<tr>
<td>Work environment</td>
<td>Different kinds of disturbances in the work environment are more prevalent in large-scale agile settings</td>
</tr>
<tr>
<td>Team spirit</td>
<td>Team spirit is built up over time, and moving members or adjusting teams may decrease this</td>
</tr>
<tr>
<td>Ceremony agreement</td>
<td>Aligning product owners’ planning abilities with the context surrounding the teams is extremely important (e.g. via coordination meetings)</td>
</tr>
</tbody>
</table>

3.6.2 Other Challenges

While large scale agile brings many benefits to the table, there are also many challenges associated with different aspects of adopting it in a large enterprise. Leffingwell (2007) lists several challenges, many of which stem from agile’s origin in small-team settings, including “small team size, close customer involvement, collocation, emerging architecture, lack of requirements analysis and documented specifications and physical involvement”.

A large challenge brought up by Turetken, Stojanov, & Trienekens (2016) is that the adoption approach of SAFe is complex and risky. To deal with this, they propose the usage of an adoption model titled the “SAFe maturity model” in order to guide the adoption, as well as gauge the level of SAFe adoption.

3.6.3 Practices

According to Leffingwell (2007), there are seven agile practices that natively scale, these are: The define/build/test component team, Two-level planning, Mastering the iteration, Smaller and more frequent releases, Concurrent testing, Continuous integration, Regular reflection and adaption.
Table 3: Agile practices that scale, according to Leffingwell (2007)

<table>
<thead>
<tr>
<th>Practice</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>The define/build/test component team</strong></td>
<td>This corresponds to the cross-functional team.</td>
</tr>
<tr>
<td><strong>Two-level planning</strong></td>
<td>Planning both releases with “broad brush strokes”, or epics, and iterations in greater detail.</td>
</tr>
<tr>
<td><strong>Mastering the iteration</strong></td>
<td>Having new, potentially shippable, working code at the end of each iteration.</td>
</tr>
<tr>
<td><strong>Smaller and more frequent releases</strong></td>
<td>Release more frequently in order to get feedback faster.</td>
</tr>
<tr>
<td><strong>Concurrent testing</strong></td>
<td>Testing is integral to the development process, all code is tested code, and there is never “theoretically working code”.</td>
</tr>
<tr>
<td><strong>Continuous integration (CI)</strong></td>
<td>The ability to integrate, build and test continuously during the development process. (See section 3.4)</td>
</tr>
<tr>
<td><strong>Regular reflection and adaption</strong></td>
<td>Reflecting regularly on how to become more effective, and adjust accordingly.</td>
</tr>
</tbody>
</table>

Many of the practices that Leffingwell (2007) claims scale are also brought up by other literature. Kruchten (2013) argues that agile practices fit different contexts to differing extents. While iterations, retrospectives, pair programming and backlogs fit well up to a large scale, daily standups and monthly releases need adaption, but scale, and the practice of using metaphors to help plan is “not useful, or requires considerable adaption”.

3.7 Agile & Stage Gate Hybrids
A fairly recent development within project management concerns models which take aspects of both agile and more conventional stage gate models. While many companies use some type of agile-traditional hybrid, and this type of project methodology has been researched since the advent of agile development (Karlström & Runeson, 2005), further empirical research is needed to test theoretical recommendations (Barlow, et al., 2011). However, Karlström & Runeson (2005) argue that this approach towards project governance is not only feasible, but carries advantages such as better cost control and on-time delivery compared to pure agile methods.

An interesting point raised by (Barlow, et al., 2011) is that large, complex organizations tend to choose simpler, less costly types of coordination such as planning and standardization in order to facilitate project work, rather than the more decentralized types of coordination proposed by agile methodologies. They argue that this is due to the fact that large, complex organizations require a significantly higher amount of coordination, which naturally leads to using these types of coordination.

3.8 The Iron Triangle
In traditional project management, the iron triangle is a way to illustrate project success and different tradeoffs which can be made in the project (Maylor, 2010). According to Maylor (2010), the triangle has three corners, which generally are titled Time, Quality and Cost. However,
while this model has been well used over the past half-century, there are many problems with it (Atkinson, 1999).

![Figure 3: The Iron Triangle](image)

According to Maylor (2010), some authors claim that modern management practices have made the consideration redundant, or at least no longer necessary, however, the evidence is lacking for this. Atkinson (1999), argues that time and cost are at best only guesses made at a time where the least is known about the project.

### 3.9 Software Metrics

Different kinds of metrics have long been a part of software development in general, and agile software development is no exception. Hartmann and Dymond (2006) argue that using metrics is inherent to agile methodologies’ approach to software development, though it can be both formal and informal. However, they also argue that measurement drives behavior, and that metrics therefore easily can be counterproductive. They argue that it’s important to make a clear distinction between measuring with the purpose of delivering value, and measuring just to document or justify the agile method. T. Gilb published *Software Metrics* in 1977, coining the term “Software Metric”. It also advocates the use of one of the forerunners to agile development, IID (Larman & Basili, 2003). However, though software metrics have been used quite far back, the needs have changed with the methodologies used.

Throughout the years, supervision of software development has relied on different metrics, ranging from lines of code (LOC, SLOC or KLOC) or function points (Cheung, Willis, & Milne, 1999) to more subjective metrics, such as number of completed “story points” per sprint, or other unit of time (Javdani, Zulzalil, Ghani, Sultan, & Parizi, 2012). However, these metrics all have different issues, and there is no silver bullet-solution to suit all purposes (Javdani, Zulzalil, Ghani, Sultan, & Parizi, 2012). In addition to this, the rise of Agile development has placed new demands on the measures used to facilitate control, making sure that they cover the right things for the new kinds of processes, products and projects which use these methodologies (Javdani, Zulzalil, Ghani, Sultan, & Parizi, 2012).
3.9.1 Velocity and Effort
One metric which is commonly used to help with control, and specifically to measure progress in agile teams is “velocity”. This is a loose term used in agile circles, often in place of productivity (even though their definitions differ). A common definition of velocity within agile software development is the number of user stories, or tasks, completed in a certain number of “sprints” which commonly consist of 1-4 weeks each (Javdani, Zulzalil, Ghani, Sultan, & Parizi, 2012). However, there are issues with measuring and using velocity as well. Javdani et al. (2012) argue that the metric is mainly useful for stable teams working full time. Furthermore, they argue that it is important to use sustainable velocity, and not short term velocity, that is, velocity should be measured over multiple sprints. Hartmann and Dymond (2006) agree with this, and also argue that it is important for agile metrics to look at trends and not just single numbers, that is velocity or any other metric should not be taken as a snapshot, and instead be viewed over a longer period of time to be truly useful. It is also important to point out that velocity only holds relevance to the specific team being measured, which means that they cannot be used to benchmark teams against each other in any way (Hartmann & Dymond, 2006). Together with velocity, effort (e.g. story points) is the most popular metric suggested by agile literature (Kupiainen, Mäntylä, & Itkonen, 2015). Together, these indicate the importance of tracking progress in agile software development, something which many large companies struggle with.

3.9.2 Outcome vs. Output
While software metrics used in the past (and in many companies, still) include output-centric metrics such as lines of codes produced or function points, agile development advocates outcome over output. That is, delivering customer value should be the focus, and is what should be measured. In the words of the agile manifesto, “Simplicity – the art of maximizing the amount of work not done – is essential” (Beck, et al., 2001), in other words: get the best possible outcome for the minimal amount of work. Hartmann and Dymond (2006) agree with this, arguing that “the most spectacular outcome might be achieved by reducing planned output while maximizing delivered value”. A way to do this is to measure customer satisfaction, which is advocated by Kupiainen et al. (2015), who even go as far as saying that successful agile projects measured customer satisfaction often or always.
4. The Case Company

This chapter objectively describes different aspects of the company which was the subject of the case study performed in this study. The focus is on aspects relevant to the rest of the study.

4.1 Corporate Profile Summary

The case company is a large multinational technology corporation headquartered in Munich, Germany. At the end of 2015, they had approximately 11,400 employees, and were represented in 31 countries across every continent. That same year, they generated sales amounting to about 2 billion euros.

According to intranet documents for external use, research and development has always been a top priority for the case company. As of 2015, they employ 1200 developers, and spend a total of 105 million euros on R&D yearly. Additionally, they hold around 7200 patents.

4.1.1 Acquisition of the Stockholm Office

While the case company was founded in Germany at the middle of the 19th century, the Stockholm office is a relatively new acquisition dating to 2009 (The Case Company, 2009). The Stockholm office was acquired as part of a firm which developed mobile device management (MDM) and sim management software (The Case Company, 2009). According to an interviewee, this firm at acquisition had a few hundred employees (CCN3). This acquisition resulted in major changes for the Stockholm office, as many functions moved, and others were integrated into a more distributed, global company (CCN6).

4.2 Organization

The case company is at its core, a matrix organization, where all employees have multiple reporting lines. It is described in further detail below.

4.2.1 Business Areas

The case company is a large multinational corporation with many different business areas. Their main areas are split into business units. These are “Currency Technology”, “Mobile Security” and “Government Solutions” (shown in Figure 4). In addition to these, they have multiple subsidiaries with other business areas.

Within each business area, there are several divisions. In Mobile Security, they are “Telecommunication Industries”, “Financial Institutions”, “Enterprise Security / OEM”, “Global Operations”, “Global R&D” and “Financial Management”. Telecommunications Industries, where CCN mainly work, is in itself a large division with many different products, for example, software for subscription management and SIM cards.

The business units and divisions of the case company are shown in Figure 4.
4.2.2 Development Centers
The R&D department of the case company is made up of multiple development centers, of which the Nordic Development Center (NDC) is one part. In addition to NDC there are development centers in many countries, including Germany, China, India, Spain and Finland. Though there are many development centers, only a few currently work on server/client software, which is a rather new field for the case company. Historically, mobile security’s main expertise has instead been within embedded systems.

NDC consists of an office in Helsinki and one in Stockholm. The Stockholm office almost exclusively works with the Telecommunication Industries division of the Mobile Security business unit. Their main focus is on developing servers used to administer mobile subscriptions (the basis for the eSIM). Other development centers focus on other technology, including different kinds of currency technology (historically, an important business area for the case company) and traditional, physical SIM-cards as well as other types of smart cards.

4.2.3 Functions and Products at CCN
Within CCN, multiple functions from the organization are represented, these range from R&D, to Product Management, to sales and marketing.

The R&D department at CCN consists of several scrum teams led by 4 team leaders and a Head of R&D. They work on a range of products, and parallel versions of those same products. Historically, their cash cow has been an over-the-air (OTA) platform which performs SIM and
mobile device management (CCN3). However, during the last years, they have been developing a replacement with some additional functionality using newer technology.

4.3 Projects at NDC

4.3.1 R&D Processes
All teams at NDC use the Scrum methodology, and have a long history of doing so, dating to before the Stockholm office was acquired by the case company (CCN10). While the organization and processes have changed significantly since then, the core methodology used in the teams remains intact. All teams in Sweden operate on 2 week sprints, after using 3 week sprints a few years back (CCN4). According to a team leader at CCN, this change was made primarily due to the faster feedback cycles which it enables, and resulted in less overhead in planning work, due to more frequent communication.

While CCN use Scrum, certain parts of the methodology are changed from how they usually are in a typical Scrum setting. One of the largest differences is that the Scrum role of product owner is split between a Product Manager (PM) and a Technical Product Owner (TPO) (CCN1). While the TPO sits inside the team and participates in development work as well as much of the planning, which is typical for the PO-role, the PM takes a share of the planning workload as well, and most of the communication with account managers, stakeholders, and occasionally clients, depending on the project (CCN7, CCN8, CCN10).

In theory, this is how the organization is supposed to look, however, many other kinds of constellations exist as well. In some cases, there is only a PM and no TPO, while sometimes there are multiple teams sharing a TPO and a PM. In other cases, a TPO answers to multiple PM’s, as well as multiple teams. These different constellations are shown in Figure 5.
4.3.2 Product Lifecycle Management (PLM)
The main project model used by the case company in general is called “Product Lifecycle Management” (PLM). In their own words:

“Product Lifecycle Management (PLM) covers all phases in the lifecycle of a product. This includes product portfolio management as well as product definition, realization, industrial engineering, utilization, and phase-out.”

The part of PLM which is of the highest relevance to R&D is the project management segment (pictured in Appendix B: The Product Lifecycle Management model). This describes 8 main steps numbered PLM 000-700, with 3 sub-steps as part of the “transition phase” leading up to the PLM 500 milestone. They are described in the following table (the full diagram can be found in “Appendix B: The Product Lifecycle Management model”):

Table 4: PLM phases (main R&D phases italicized)

<table>
<thead>
<tr>
<th>Milestone</th>
<th>Preceding phase</th>
<th>R&amp;D Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLM 000</td>
<td>PPM Initiation</td>
<td></td>
</tr>
<tr>
<td>PLM 100</td>
<td>Clarification</td>
<td></td>
</tr>
<tr>
<td>PLM 200</td>
<td>Definition</td>
<td>(Prototyping)</td>
</tr>
<tr>
<td>PLM 300</td>
<td>Concept</td>
<td>Define product and test concept</td>
</tr>
<tr>
<td>PLM 400</td>
<td>Realization</td>
<td>Implement and test product</td>
</tr>
<tr>
<td>PLM 450</td>
<td>Staging Deployment</td>
<td>Support Ramp-up</td>
</tr>
<tr>
<td>PLM 470</td>
<td>Production Deployment</td>
<td>Finalize Product Development</td>
</tr>
<tr>
<td>PLM 500</td>
<td>Production Stabilization</td>
<td></td>
</tr>
</tbody>
</table>
The main phases of R&D are between the 200 and 470 milestones, i.e. from concept to production deployment, while they play a supporting role in the definition, utilization and phase out phases. While this whole project model is shared with the production/hardware-focused part of the organization, the transition phase has two alternate interpretations for each phase, for example, the phase leading up to PLM 450 is “Staging deployment” for service focused products (such as the eSIM), while it is “Operational Staging” for production focused industries.
5. Empirical Findings

Using the methodology described in the previous chapter, a case study was performed at the case company. This chapter describes the main findings from interviews and observations at the Stockholm and Munich offices of the case company.

5.1 First Order Findings

The findings below were based directly on the empirical data gathered during this study. While they have all to some been confirmed to be real challenges faced by the case company, many of them should rather be seen as symptoms for larger underlying issues, which are described in section 6.1.

5.1.1 Collaboration & Communication

One of the most frequently discussed themes during interviews was communication. Even though the interviews were open ended, the starting point for this study was to look at metrics and KPIs, which naturally led to this theme in many of the interviews.

5.1.1.1 Progress

It quickly became clear that one of the primary reasons that the case company was interested in using more metrics was to monitor progress in a better way, which many in management found very difficult (CCN4, CCN5, CCN8). According to several interviewees, a key difficulty regarding progress is it to not only understand and communicate it inside the teams, but communicate it to the organization surrounding the teams, most notably product and project management (CCN4, CCN8). For this, metrics used for planning inside teams, such as velocity cannot be used since it only holds relevance for the specific team measuring it, according to team leaders at CCN (CCN4). The only use velocity can have outside teams is as an abstract efficiency improvement metric, since the relative trend can be used to show the rate of improvement, according to several interviewees (CCN4). However, this is dependent on velocity being used over longer periods of time on stable teams doing similar kinds of development (CCN4).

Interestingly, while several interviewees mentioned measuring progress as a challenge at CCN, an interviewee at CCMU held that they did not have many issues with gauging progress of projects (CCMU1). However, they also said that it varied greatly, and that it was down to the individuals monitoring progress since there is no agreed upon methodology for checking the progress (CCMU1). However, it is important to note that while the interviewee who made this remark has project management experience, they are not currently working directly with project management today.

5.1.1.2 Customer Feedback

In order to measure project success in better ways, several interview participants held that outcome rather than output should be monitored, that is, rather than measuring the direct output of projects (most often code), the aim should be to look at the satisfaction of customers (CCN1, CCN3, CCN4, CCN5). A difficulty with this is that the measurement of customer satisfaction today is quite blunt, only resulting in a single KPI for CCN using NPS (Net Promoter Score) (CCN4).
While it is difficult to gain valuable insights from a single KPI done on external customers, there may be other ways to gain valuable feedback. According to one interviewee, one possible way to check customer satisfaction is to gauge the satisfaction of internal customers instead of external customers (CCN5). This would also improve the feedback which they get from internal customers, which is lacking in many respects. According to one interviewee, the feedback they receive internally consists mainly of “When things are silent, you can assume they are going well. However, when mails are flying around and people are screaming, usually something is missing.” (CCN4). While the R&D department at CCN does have surveys for internal customers, this information is not forwarded downward into the organization, and can therefore not be used for continuous improvement in development teams (CCN4).

The only area where there is direct interaction between the team members and external customers is during support work, which constitutes a varying amount of time for different teams (CCN6, CCN7). This work is usually split unevenly between team members, and feedback from the customer usually does not go beyond the individual working on the task, which limits the information exchange, and therefore the possibility to use it as feedback for the teams (CCN6).

Product managers in general have little contact with customers too, and instead usually get customer feedback from account managers or others who handle more of the day-to-day interaction with customers (CCN8, CCN12). Project managers on the other hand had very differing amounts of contact with clients, with one interviewee at CCN saying that he had weekly contact with customers (CCN12), while a project manager in Munich had next to no contact with customers, and instead relied on product management for this (CCMU2). The project manager in Sweden even raised customer feedback and external communication as one of the single greatest challenges in his job, saying that “we have an internal project space, but not one for communication with the customer where we can share documents. Instead we use a lot of emails. To keep track of where we are and in which document, that is a huge risk and a weakness.”.

5.1.1.3 Information Flow and Decision Making
According to several of the interviewees, the slow and sometimes non-existent flow of information is another key issue facing the organization (like many other large companies) (CCN4, CCN5, CCN6, CCN7, CCN8). This can create other issues than the difficulties described above. One interviewee who had been at CCN for many years described the situation as “It’s not like we don’t get any information, but we get significantly less than we used to. Previously we got the information that was available, while now everything disappears into numbers, which are split up and spread out [over the organization]”. This slow flow of information has other implications as well, including slow decision making. One interviewee noted this, saying that “All approvals have to go so far up […] it can take several weeks or months [to make a decision]” (CCN10).

5.1.1.4 Reliance on Formal Communication
What became clear during many interviews at CCN is that outside of the R&D department at CCN, there is a large dependence on using explicit knowledge to communicate in general. As an example, the Head of R&D at CCN is obligated to each other week compile a project report
for all projects at CCN (CCN5). This is later consolidated with project reports from other development centers at CCN and discussed in a product portfolio board (CCN5). Another area where there is a high reliance on explicit knowledge in the form of documentation is quality improvement. Regularly, product managers manually fill out quality reports for their products, listing things such as planned requirements, tested requirements and completed requirements (CCN8). According to a product manager at CCN this work has given results, but there is significant overhead to the manual completion and often adjustment to different reports (CCN8). Furthermore, it is often unclear to what extent the reports are used in the end, saying “We have been bad at following things up. If you write a report, you want people to notice it. Not for my sake, but for others.” (CCN12). At CCN there is also a will to make communication more efficient by consolidating channels, and, for example, having a single point of contact for each team (CCN12). This point of contact is however typically decided on an ad-hoc basis. While a project manager liked having a single point of contact, this can create issues. He was quoted as saying that “I can get a delivery date from this person. I know that I can’t get this from R&D or the developers, so I get it from the TPO or the product manager, who is my contact.”. It is not inconceivable that getting a delivery date from a person who is one or more stages removed from the person doing the work can result in unfulfillable promises being made.

5.1.2 Planning & Organizing

Another major area which came to light during the study was different issues relating to control, and especially planning & organizing therein, within projects and R&D at CCN.

5.1.2.1 Rigid Work Processes

According to several within management at R&D at CCN, most of the organization outside Sweden use more rigid work processes to a large degree and have more centralized control than Sweden (CCN1, CCN4, CCN5). In addition to this, the case company has many different business areas, of which only a clear minority are around software (CCN1, CCN5). Of these, the software business areas tend to have less centralized control with looser organization and less rigid methodologies compared to the other business areas (CCN5). As mentioned in section 4.1, most of the organization uses the PLM project model, which does not dictate the particular work processes of developers in it. However, according to many interviewees at R&D, the project model is more suited towards a waterfall-way of working, with rigid requirements, a strong emphasis on sequential development, and lower focus on customer interaction during the project (CCN1, CCN3, CCN4, CCN5). According to several interviewees, this creates unnecessary overhead, and as one participant describes it: “If we add this [increased project planning and control] to something which is already technologically complex, we create a large complexity in the organization which slows us down and makes us unnecessarily expensive” (CCN5). Furthermore, it can be difficult to explain to customers or stakeholders that are used to a non-agile way of working that features which were planned early, can be changed, due to the working in an agile way within PLM (CCN4).

Since most of the organization is used to a more rigid way of working, some stakeholders from other parts of the organization view the methodology used in Sweden unfavorably (CCN5). An interviewee outside of R&D also expressed that the methodology has a low impact on the work
produced (CCN9). This is reflected in one of the interviews from inside R&D, which noted that they would like a greater degree of enthusiasm from other parts of the organization for the work process they use in Sweden (CCN5). As things stand now, R&D in Sweden mostly chooses its own way of working in most cases, without influence from outside the projects, but they still have to deliver according to an overlying project model which is waterfall-like (CCN5).

5.1.2.2 Decision Support and Trust

Another key issue which stems from the same conflicts as above is the different approaches to decision support which exist in the organization. While R&D in general prefers a bottom-up approach, where decisions need to be supported from the bottom of the organization, other parts of the organization have many times enforced a top-down approach to decision making (CCN3, CCN5, CCN6, CCN8). An example of this is the different approaches towards quality enforcement in code (CCN3, CCN6). According to several interviewees, with support from within the teams, many of the teams in Sweden have spent significant time implementing CI measures to increase the quality of code though things such as unit testing, code reviews and tools such as SonarQube (CCN3, CCN6, CCN7).

While significant time has been spent implementing CI processes and tools, this has by their own admission increased the quality of the code and the delivery time (CCN6, CCN7). This is also corroborated by research and literature, which raise CI as essential to the agile method (see section 3.4). In contrast to this, previously, external consultants have been hired without support from within the teams to examine the code quality of all code in use at CCN (CCN3, CCN6, CCN7). This was later compiled into a report, which was delivered to management, and by them, to the developers, who according to an interviewed developer were expected to fix the issues described on top of their regular duties (CCN6). According to an interviewee, additional resources were later devoted towards fixing the issues while still moving forward with the projects (CCN6). However, this proved difficult to do, and progress still slowed significantly during this period of time, at the same time as morale was lowered among the developers (CCN3, CCN6). Morale was lowered not only due to the top-down approach, which to them was unusual, but also since the findings of the report in their mind could not be discussed or questioned in the same way as they are used to being able to discuss other matters relating to their work (CCN6).

In general, the sentiment among developers seems to be that management does not trust them. Regarding the aforementioned code report, one developer said that “They don’t trust the Swedish teams with investigating themselves. We saw what they did as them pointing from up there and saying ‘Your code sucks, you have to correct this, here is a bag of money!’”. In my time at the company admin rights for developers were also removed, which also had a negative effect on trust and morale. Regarding this, one developer said “It was a bit weird that they were that harsh. [It’s like they said] ‘We can’t trust anyone!’”.

5.1.2.3 Planning

The usual fulfillment rate for the number of features and story points accumulated during a sprint is 50-60% of what is planned, but that number fluctuates heavily according to a product manager (CCN8). In addition to this, a team leader at CCN said that a key metric used by many organizations to facilitate planning for teams, velocity, is not used by almost any of the teams,
making planning significantly harder inside teams (CCN4). According to a product manager, these difficulties are mirrored in product and project management as well, but in this case, the difficulties often stem from being notified quite late regarding delays, which reduces the ability to mitigate the delay through, for example shifting resources where needed (CCN8). Instead, the most common solution is adjusting the scope (CCN8). This ties together with difficulties described above, regarding communicating progress, since planning is highly dependent on the current progress during the sprint.

5.1.2.4 Team Stability
One underlying theme to many of the interviews was that the stability of teams varied greatly. According to one developer, many times, the Technical Product Owner of a product can be exchanged quite hastily, which according to him results in loss of non-codified knowledge regarding the product, which slows down development and results in a worse product being delivered in the end (CCN7). One product manager went as far as saying that “We are not stable at all. Total chaos. I have had teams in India where people have: disappeared, resigned, moved, new people coming in, consultants. All the time. I have had more than one year with no additional functionality in the product whatsoever.” (CCN10). It is important to point out that the employee turnover differs greatly between locations, with India having the highest turnover, resulting in very unstable teams (CCN10).

5.1.3 Important Tools for the Development Process
A theme which was brought up by many interviewees in different roles was the high importance of tools, especially in connection to the ability to automate or make tasks more efficient (CCN1-8). This theme often overlaps with the other presented themes, which is why subjects are brought up here as well as above.

5.1.3.1 Reporting and Documentation
As mentioned, different kinds of manual reports are used to communicate to a large extent in the organization. According to a product manager, significant time is spent on quality improvement reporting (CCN8). According to multiple interviewees, this manual reporting and subsequent processing of the reports relies on data which is often readily available in software tools, such as JIRA, used by several teams at the company (CCN4, CCN8). According to a product manager, there is an effort to utilize this data to automate certain reporting, but since it is dependent on people both using the tools, and using them in a proper way, this has been found difficult in many cases (CCN8).

Another process which could be simplified by more automation and better tools is documentation. According to a product manager, significant time is spent on manually creating and updating release notes. Not only does this have an impact on productivity, but on quality of the notes as well, since errors are easily made and the reports need to be proof-read and manually fixed (CCN8).

5.1.3.2 Continuous Integration (CI)
Teams at several sites at the case company, Stockholm included, make a conscious effort to implement better practices regarding continuous integration (CI). These include automatic
testing, building and deployment of software. Even though this work is mostly central to developers, which was made clear by the fact that all interviewed developers brought up the subject, product managers can also see the impact that it is having. According to one product manager, this impacts his role in several ways, one is that it raises the quality of the releases, due to the frequent, automatic testing (CCN8). Another is that it lowers the price per release since deployment is frequent and simple compared to when using older tools. This lowers the time and work needed to take software from completed code to actual release (CCN8).

CI is one of the prime areas where the benefits of international cooperation can be seen. One of the R&D departments of the case company where this work has come the furthest is in Munich, Germany. The R&D department there was highly involved in improving the practices of multiple teams in Sweden (CCN6).

Aspects of CI were also discussed by a developer in relation to the previously mentioned external code quality report (CCN6). According to him, the way they work in Sweden, this kind of quality work needs to come from below, which CI assures it does. It does this in two ways: it provides instant feedback, laying the code-quality responsibility directly on the developer, and, just as importantly, it is viewed favorably by developers (CCN6, CCN7).

While there is a great internal effort to implement CI measures, as of today, teams follow a rigid release schedule with significant planning preceding each release (CCN3). Additionally, there are great technical hurdles to being able to release frequently, which is a current focus area within CI at CCN (CCN6). Also, the level to which teams have implemented CI practices varies greatly between teams, with some teams being quite far behind in this regard. There are several reasons for this, such as internal resistance, but the single greatest reason is the amount of legacy code which is present in several older products (CCN6).
6. Discussion

In chapter 5, ten themes in three main categories were identified when analyzing the empirical data gathered in this study. These were “Collaboration and communication”, “Planning & Organizing” and “Tools”. When discussing these primarily in relation to each other, a couple of common, underlying themes were found. In order to deal with both the first order findings in chapter 5, and the common themes presented in chapter 0, a couple of main areas of theory were studied and brought in from chapter 3. While the key issues facing the case company could arguably be said to be within “Collaboration and communication”, and “Planning & Organizing”, metrics were also brought in as an area due to its ability to support processes in both. In communication, it can increase the clarity and decrease the chance for misinterpretation, and in Planning & Organizing it can help in key areas such as building decision support and estimation.

6.1 Common Themes in Empirical Findings

Many of the themes and subjects mentioned in section 5.1 can be seen as symptoms for larger, underlying issues. These were identified during the analysis of the empirical data and are described below.

6.1.1 Conflict Between Standardization and Autonomy

**Based on: Reliance on Formal Communication; Rigid Work Processes; Important Tools for the Development Process**

Many of the challenges raised by interviewees show that there is a conflict between the desire to standardize different aspects of work processes, at the same time as there is an opposing force valuing decentralization, flexibility and adapting to context.

The high, and in some cases increasing reliance on standardized reports (CCN8) shows that there is a desire to standardize information exchange internally. The way the case company is organized today, some of this reporting is extremely important, such as the bi-weekly project reports representing each development center (see section 5.1.1.4). These are important for the project portfolio board to do their work (CCN5, CCMU1). On the other hand, as mentioned in section 5.1.1.4 and 5.1.3.1, this kind of work also causes some unnecessary overhead, which becomes all the more obvious when the reports are not used, or when something which can be done automatically needs to be reported manually. This can be compared to how Barlow et al. (2011) emphasize the need for standardized channels for communication in order for the number of communication interfaces to stay low, and therefore decrease the complexity of communication.

According to interviewees, tools used by the development teams are important to standardize in order to facilitate different aspects of work and management (CCN4, CCN6). As described in section 5.1.3, many downsides of increased decentralization can be seen here, as many teams today resist adopting even highly proven tools, such as different CI tools, which are proven to work well by other teams (CCN6). Additionally, there is resistance from developers towards increasing adoption of tools mainly facilitating management, such as JIRA (CCN1, CCN3). This is a major reason that sprint burndown charts are not used in the teams (CCN3).
Another area where the conflict between standardization and autonomy is evident is in the different work processes used at CCN. While the PLM model (described in section 4.3) a rigid, mostly unidirectional project model, is used all over the case company, all of the individual teams in the Stockholm office use Scrum within their teams today. This standardization could make it easier to transfer workforce between offices, and is supported by many in management (CCN5). However, according to interviewees at the case company, PLM is more adapted to a rigid, waterfall-manner of working within projects (see section 5.1.2.1). This in many cases conflicts with the majority of teams which follow an agile methodology within projects. In Sweden, all teams have adopted Scrum as a methodology, while only a minority of teams in other locations use different agile methodologies, while the rest use methodologies similar to waterfall or other rigid methodologies (CCN1, CCN5, CCMU1-4). As described in section 5.1.2.1, there exists a desire within the company to conform to the demands placed by the rigid project model, while at the same time adhering to the current practices, and harnessing the strengths of them (CCN5). Issues faced by CCN which are reinforced by this includes difficulties in working with requirements (CCN6, CCN8), difficulties in tracking progress and control in general (CCN8) and general frustrations from senior management with the lack of conformity (CCN5).

In addition to the diversity in the different methodologies used in different development centers, projects are also organized differently besides the methodology used, which has direct effects on communication (CCN6). Some of the different configurations are described in section 4.3 and shown in Figure 5. According to interviewees at the case company, there is not a general answer to which of these configurations works best, as this depends heavily on the demands of the project (CCN1). What can be deduced however is that none of the different configurations are typical for Scrum, where the product owner typically is a proper team member (Schwaber, 2004). At CCN, the product owner is not a team member, and typically only attends sprint planning meetings and demos, i.e., not daily scrum meetings. The role at the case company which mostly corresponds to the typical product owner role in scrum is instead the technical product owner, who usually in addition to taking up some duties of the product owner, works as a developer the majority of their time.

6.1.2 Conflict Between Stability and Agility
Based on: Rigid Work Processes; Decision Support and Trust; Planning; Team Stability

A theme which to a degree ties together with the conflict between standardization and autonomy is the one between stability and agility.

As mentioned, the Stockholm office is used to working with agile processes with an overlying, more rigid project model (PLM). There are conflicting opinions as to which is more favorable to the development they do at CCN today (CCN4). One of the main reasons brought up for working with more rigid requirements is that it is more predictable, however, there is disagreement as to whether that is true (CCN4, CCMU3). While the PLM model favors a way of working where the scope (requirements) is rigid, and time and resources are more fluid, Scrum, (and consequently how they work in Sweden) favors adjusting the scope, while resources and time are rigid (CCN4). This can be compared to the iron triangle described in section 3.8. As mentioned in section 5.1.2.1, it can be difficult to explain to customers or stakeholders that are
used to a non-agile way of working that suddenly a certain, agreed upon feature will not be developed as the scope changes (CCN4). In practice, the process used in software development by CCN is a type of stage gate hybrid, as described in section 3.7. As mentioned in section 5.1.2.1 and 6.1.2, the satisfaction with this model varies greatly between interviewees and departments, with product management generally favoring a stage gate heavy approach (PLM) and R&D favoring a more agile approach (Scrum). According to Karlström & Runeson (2005), this is quite typical, with management generally fearing agile methods, and developers favoring them. However they also argue that there are significant benefits to using stage gate hybrids as a software development approach (see section 3.7). Many of these benefits are also prevalent at CCN, which is shown in interviews with product managers.

There are also conflicting schools at the case company as to who needs to support decisions. While there is evidence of a top-down approach in several cases (CCN4, CCN8), the Stockholm office favors bottom-up support (CCN8). While a bottom-up approach can be favored for many reasons, it is often turbulent, and quite slow, and relies on high motivation and participation (Burnes, 2009), which the slow adoption of tools in many teams shows (CCN8). The opposite approach can also be favored for many reasons, including it generally favors more stable environments (Burnes, 2009).

Ideally, every project wants to deliver on time. As described in section 5.1.1.1 and 5.1.2.4, in some part, this relies on team stability and some manner of predictable stability in how fast any given feature can be developed, which often does not exist, and is not generally kept track of at CCN today (CCN4, CCN8). When things do not go according to plan in software development, several things can be done, which in some manner rely on being flexible. While PLM and customers of the case company rely on requirements being kept stable, being agile requires requirements to be updated after time as delays happen, and as knowledge is gained (Schwaber, 2004) (CCN4). While this sometimes happens, requirements are more rigid than may be typical in an agile environment (CCN4). Additionally, customer participation in development is low, and change of requirements is therefore difficult, often resulting in internal negotiations between product management and developers (CCN8), instead of involving the external customer. This interaction with customers is discussed further in section 6.2.

In agile software development, a central tenet is that teams should be as stable as is possible, and research has shown that hyper-productive teams have to be stable (Sutherland, Harrison, & Riddle, 2014). As described in section 5.1.2.4, though some teams in long projects can be considered stable at the case company, many are not. However, the upside to not having stable teams is that resources can be allocated more easily. While team stability can be an issue in Sweden, it is significantly worse in India, where the employee turnover is extremely high (see section 5.1.2.4). Both theory and interviewees from this study confirm that this lowers the development velocity by a large amount, which means that this is a significant argument against moving projects from a location with more stable teams, to India or other locations with a higher turnover.
6.1.3 Organizational Distance

Based on: Progress; Customer Feedback; Information Flow; Reliance on Formal Communication; Planning; Reporting and Documentation

In several interviews, organizational distance was brought up, and one interview directly connected it to several of the challenges which CCN face today (CCN4, CCN5, CCN10).

The first area where this likely has an impact is in tracking progress (described in section 5.1.1.1). According to a product manager, they find it extremely hard to track progress of development, and often get warnings of impending delays extremely late during the development cycle. As described in section 4.3, at CCN, the role of product manager (PM) together with the technical product owner (TPO) corresponds to the Scrum role of product owner. One of the main differences in roles is that the PM is not a team member, and does not attend daily meetings, or work alongside the team. The only team meetings they usually attend are the sprint planning and the demos (CCN8, CCN10). While this usually works well according to them, challenges occur due to insufficient (or unclear) communication, which being more closely knit would likely remedy. Another possible impacting factor is that the relationship between the PM and the team is a client-supplier relationship, rather than a team relationship as is typical in Scrum. This likely has an impact on the perceived organizational distance, further reducing the amount of communication. This is indicated by one interviewee requesting “more of a partner relationship”, rather than the client-supplier relationship which exists today (CCN5).

The second area where organizational distance has a large impact is customer feedback, described in section 5.1.1.2. While the case company is a customer focused company, as you need to be when you only have several hundred customers per year, the interaction between customers and R&D/product management is low (CCN4, CCN5, CCN8, CCN10), which is atypical for an organization working with scrum, or even agile practices in general (Cohn, 2010). Instead of the product manager or TPO working closely with the customer during development, that role is typically handled by account managers, which limits the technical and product related knowledge present in that interaction, leading to a tendency to overcommit (CCN10). As a product manager said, “...Then they [account managers] say ‘Yes, we will deliver this’, and then we may have three different client projects on our hands at once, but we always have the same pool of resources, which leads to delays, which leads to us taking in consultants, which takes a long time. Everything takes a long time.”. It also reduces the information flow to the teams doing the development (CCN8). Typically, the customer has an extremely central role in Scrum, often participating in demos and some day-to-day interactions (Schwaber, 2004). That role is instead handled by product management at CCN, who typically take the role of the customer in this interaction. These factors result in many challenges, but one of the most critical ones is that often account managers make promises which R&D cannot deliver on, among other things, due to the lack of knowledge about current capabilities (CCN10).

As described in section 5.1.1.3, the challenges with information flow are not only present in customer feedback and interactions, but in many facets of daily work at the company (CCN5, CCN10). While this does not seem to be a limiting factor on day-to-day work, it can complicate
matters extensively when making critical decisions regarding, for example, resource planning in projects (CCN10). The opposite regarding decision making can also be a challenge, namely that decisions can be made without key stakeholders participating, or in some cases even being informed (CCN10). One such case was when an interviewee from product management described four people being moved from a project without them getting any notice, which they found out only during the following sprint planning meeting. The missing resources for the project would subsequently not be filled for two months, which slowed development down significantly, while they were still expected to deliver according to the same schedule (CCN10).

Reliance on formal communication, described in section 5.1.4, is likely also influenced by the organizational distance, due to the lower ability to communicate face-to-face. Instead of speaking directly to people you need information from, processes for reporting can be created. While this has the advantage of codifying knowledge, which enables easy distribution (Tidd & Bessant, 2013), tacit knowledge is easily lost in this kind of exchange. There are many examples of this loss of tacit knowledge at CCN today (and as one interviewee says, disregard for it), including how development of a product can be moved to an entirely different team in a different country, only keeping the product manager (CCN10). According to an interview with an employee with a background in product management, this can result in not only no feature progress being made in a very long time, sometimes more than a year, but also in a reduced development velocity when new features actually start being produced (CCN10).

Another area which is likely influenced by the organizational distance is planning (see section 5.1.2.3). This can be seen in an interview with a product manager, where it was brought up that often things go extremely well until preconditions or requirements change slightly, resulting in large delays. Observations confirmed this as well, for example, in one instance, a requirement and a general blueprint for a software solution had been agreed upon by stakeholders. This solution was developed over two months, only to be rejected by stakeholders when it was deemed that the (agreed upon) solution would not be sufficient. This is likely not only the product of organizational distance however. More frequent communication and demos with stakeholders may have brought this to light earlier. This is discussed further in section 6.2.1.

6.2 Collaboration & Communication
One of the primary themes brought forward by interviewees in this study was challenges in collaboration. These findings are presented in section 5.1.1 and are discussed in relation to theory, primarily from section 3.3, here.

6.2.1 Collaboration and Communication in Large Organizations
A key difficulty which can be seen both in theory and the data gathered in this study is that communication and collaboration are difficult in large organizations (see section 5.1.1) (Evbota, Knauss, & Sandberg, 2016). One reason for this at the case company may be the conflict which exists between Scrum’s emphasis on cross-functional, self-organizing teams and the need for clear roles and channels for communication which exist in large organizations (Barlow, et al., 2011). CCN solves this in part by having a single point of contact for teams, which is usually the TPO or SM in a team, but decided ad hoc (see section 5.1.1.4). Since there is no formal agreement regarding this, it does not increase the organizational clarity (Barlow, et al., 2011),
and carries the additional downside of encouraging less communication directly between the concerned parties, likely increasing distortion and misunderstanding (see section 3.3) (Korkala, Abrahamsson, & Kyllönen, 2006).

In a smaller organization, lack of formalized contact points would likely not be an issue, however empirical data from this study shows that it can be an issue in a larger organization, primarily that information is easily lost (see sections 5.1.1.3, 5.1.1.4 and 6.1.3). Barlow et al. (2011) confirm this, and argue that the difficulties from lacking formalized communication interfaces only grow as the size of the organization increases, especially when using agile methodologies (see section 3.3.1). For example, one interviewee stated that people are sometimes not informed of large decisions which impact their work to a great extent, with no rationale given (CCN10). Two possibilities exist for alleviating this. Firstly, a higher degree of transparency in combination with more frequent, informal communication between departments will give more chances for these matters to come to light. It will also raise the degree of trust between departments, which sometimes can be lacking, as can be seen in a number of interviews (CCN5, CCMU1, CCMU2). This is also confirmed by previous research (Hettonen & Blomqvist, 2005). Additionally, Schwaber (2007) and Hummel et al. (2013) argue that raising the frequency of communication can increase overall productivity substantially. One way in which it might do this is through a lower cost for communication (see section 3.3.1).

The second option is having more formalized routines for these matters. This may help ensure that the right stakeholders are informed of important decisions. However, this would likely be difficult to realize, as routines are not always followed as it is today (CCMU1), and having more routines to follow would by itself not make it easier, but rather harder to follow them all. Another aspect which is likely costing CCN in productivity is the lack of face-to-face communication, high reliance on reports and other types of written communication, as well as the many intermediaries which are often prevalent in some types communication (see section 5.1.1 and 6.1.3). Previous studies and the agile manifesto itself repeatedly raise (frequent) face-to-face communication as cornerstones of agile software development projects which performed well (Beck, et al., 2001; Schwaber, 2007; Korkala, Abrahamsson, & Kyllönen, 2006; Mishraa, Mishrab, & Ostrovskae, 2012). Addressing this issue fully likely requires many actions on the part of CCN. For instance, product managers should be encouraged to communicate much more frequently, and more informally with the development teams than they do today. Additionally, there needs to be a larger awareness regarding organizational distance which is prevalent in many cases (see section 6.1.3). Information being lost when traveling through intermediaries is a common problem, so when possible, communication should be informal, and directly between the concerned parties, no matter how many degrees of separation these usually have between them. This reduces the distortion of information and increases ability to receive immediate feedback (see section 3.3).

6.2.2 Customer Feedback

As previously described, a key component of all agile methodologies is the crucial role that the customer plays. At CCN, the role of the customer is most often played by an internal client, which is most often product management, as well as other internal stakeholders. External stakeholders only give feedback to developers in the case of support work, and this is never
used for the continuous improvement processes (CCN6, CCN7). While this lowers the dependence on external communication, it creates several issues. Firstly, the Agile Manifesto’s two first principles are that “our highest priority is to satisfy the customer through early and continuous delivery of valuable software” and that you should “welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage” (Beck, et al., 2001). Not involving the customer regularly in order to deliver higher value as more is learned is a clear violation of those principles, which likely results in a lower level of customer satisfaction. In addition to this, not getting feedback from users is a motivational issue, which can be seen in interviews. As one interviewee stated “When things are silent, you can assume they are going well. However, when mails are flying around and people are screaming, usually something is missing.” (CCN4). This goes to show the importance of having regular communication with customers, where there is an opportunity to give not only negative feedback, but also feedback on what is going well. Another interviewee also confirmed the importance of high motivation and morale in R&D organizations, which shows that there is an awareness regarding this, but that a lot more can be done (CCMU1). Another reason to increase the amount of customer feedback is that a literature review by Kupiainen et al. (2015) established that agile projects which are successful often, or always measured customer satisfaction, and in fact performed better the more often that customer satisfaction was measured. While they do not comment much on the reason for this, a possible (and not too far-fetched) interpretation is that having an immediate awareness regarding the opinion of the customer influences the project positively on many levels, both in the project control, as well as on the developer levels with increased morale and less misunderstandings regarding what the customer wants, which is something that CCN has had trouble with in the past (CCN6).

When comparing the challenges faced by the case company to the challenges described by Hoda, Noble, & Marshall (2011), nearly all of the challenges that they identify can be corroborated by data from this study. Of the main themes “Pressure to overcommit”, “Problems in Gathering and Clarifying Requirements”, “Problems in Prioritizing Requirements”, “Problems in Securing Feedback”, “Loss of Productivity” and “Business loss”, only “Business loss” cannot be identified in the empirical data. “Pressure to overcommit” can be seen, for example, when looking at how account managers typically handle a significant amount of client communication, leading to over-promising, and delays (see section 6.1.3). “Problems in Gathering and Clarifying Requirements” and “Problems in Prioritizing Requirements” can quite literally be seen in several interviews, and in the lack of flexibility present in requirements in general at CCN (see section 6.1.1). “Problems in Securing Feedback” can be seen in that there generally is no, or next to no customer feedback during development. As Hoda et al. (2011) argue, “In absence of customer feedback, teams were unable to assess how well the features met the requirements”, which undoubtedly is something that developers from CCN recognize. For example, in section 6.1.3, a case is described in which a solution was developed over two months, only to be completely rejected, arguably due to infrequent customer feedback. While “Loss of Productivity” cannot be directly seen in the empirical data, effects leading to loss in productivity, such as lacking morale, delays and trouble in planning can be directly seen (see sections 5.1.1.2, 5.1.2.3 and 6.1.3).
An aspect of the case company’s work process which is quite irregular compared to most companies following Scrum is that the roles corresponding to the product owner in Scrum (TPO and PM) have very little contact with customers (CCN8, CCN12). According to Schwaber (2004), the aim should be for customers to directly drive development in Scrum, and while this is undoubtedly hard to achieve in a large-scale setting, like at the case company, the aim should be the same. Anything else will deliver suboptimal value to the customer. Since this issue touches on many different departments of the organization, it can be extremely hard to address. One immediate way of addressing it is encouraging PMs and developers to interact more with customers in general, but to address the issue in a sustainable way, processes in other departments than product management and R&D need to change as well. Customers and users should ideally be involved at frequent demos. This should be possible if it is set as an expectation during the initial stages of a project. It is important to stress that the one of primary reason to work more closely with customers is that the customers get tremendous value from the ability to influence the product during development. Some customers are more used to this way of working, or see value in it easily, while others will likely need to be convinced (CCMU3).

6.3 Planning & Organizing
Many of the challenges that the case company are facing originate from challenges in Planning & Organizing, as can be seen in section 5.1.2. These findings are discussed here.

6.3.1 Traditional vs. Agile Methodologies
Some challenges may be traced to the traditional view on project management that is prevalent in many areas of the organization (CCN4), which can be illustrated with the iron triangle described in section 3.8. While there is significant interplay between the three corners (time, cost, quality) in traditional project management (Maylor, 2010), CCN4 argues that this is not the case in Scrum. In Scrum, you attempt to keep time and cost constant, but you can vary the scope (features and quality), by adding or removing features, and changing requirements as the project moves forward (CCN4). Explaining this to someone used to working with traditional project management, or to a customer who expects a planned set of features at an exact point of time is very difficult (CCN4, CCMU3). However, according to CCMU3, while varying the scope may be difficult to justify to customers, not doing it only hides impacts on the other corners (just like in traditional project management).

6.3.2 Scaled Agile
As mentioned in section 3.6, researchers have argued that while there are many challenges associated with scaling agile practices, there are those that do scale well. According to Leffingwell (2007), these are The define/build/test component team, Two-level planning, Mastering the iteration, Smaller and more frequent releases, Concurrent testing, Continuous integration, Regular reflection and adaption. From the data gathered during this study, it can be seen that CCN fulfill, or aspire to fulfill, several of these in a satisfactory way. These are “The define/build/test component team”, “Two-level planning”, “Mastering the iteration” and “Regular reflection and adaption”. While all of these undeniably have potential for improvement, there are firm processes in place to deal with all of these, and with regular reflection and adaption,
these are set to improve further. More significant problems for individual teams can be found in “Smaller and more frequent releases”, “Concurrent testing”, and “Continuous integration”. While there are teams that fulfill each of these, several teams are lagging behind, and thus cannot draw the benefits from having these practices in place (see section 5.1.3.2).

6.3.2.1 Key Improvement Areas

As mentioned, many of the practices mentioned by Leffingwell (2007) have standardized practices in place throughout CCN, however they are not always dealt with in a satisfactory manner. Areas where there is significant room for improvement are described below.

While “two-level planning” is used extensively (with PLM on one level, and Scrum on another, see section 4.3 and 5.1.2.1), this division with one “waterfall like” methodology on one level, and one agile on another creates many issues which are described in section 5. Having this level more light-weight, encouraging more customer participation, and making requirements more flexible would resolve this conflict. As many interviewees have mentioned, there is great value PLM and this kind of division, but it can also be seen that there is significant room for improvement.

“Smaller and more frequent releases” is a difficult area in CCN’s position. Releases at CCN usually follow significant planning (with the exception of so-called special releases) (CCN3), and cannot be deployed without some effort (see section 5.1.3.2). This makes it challenging to release frequently to customers. However, as described in section 5.1.3.2, there is a large push towards using more CI tools all over the case company, which would facilitate this significantly. An aspect to keep in mind is that the criticality of an application impacts the appropriateness of releasing frequently, and if an application can be regarded as critical, it may be useful to avoid doing this (Kruchten, 2013) It is also important to note that everything concerning releases is dependent to some extent on the ones performing the application deployment, which is not usually R&D (CCN3). This limits the amount of control R&D can have regarding this.

The primary benefit of smaller and more frequent releases is that it enables one to gather feedback faster from users (Leffingwell, 2007). However, user feedback is not frequently used today at CCN, or the case company in general (see section 5.1), which limits the benefit that they may see from using this practice to a larger extent. This highlights the importance of increasing the amount of user feedback as well, which is discussed further in section 6.2.2.

“Concurrent testing” and “Continuous integration” are areas where CCN and the case company have gained significant ground in recent years (see section 5.1.3.2). Since these are tightly related, they will be discussed together. While there are teams as CCN who use both of these practices extensively, there are those who find it difficult. The primary reason that these teams are lagging behind according to interviewees is legacy code. Since all of CCN’s older products were made without concurrent testing and continuous integration in mind, significant resources need to be spent in order to bring these up to the level of newer products. This is similar to the objections towards CI raised in section 3.4. However, while CI would undeniably help with maintenance, the investment may not be motivated since these products are set to be replaced by newer systems which do have these tools and practices in place already, rendering the counterarguments towards the objections moot.
6.4 Metrics
One thing which teams within CCN could do to improve, is utilize metrics better to help with estimation, planning and communication. As things stand today, the KPIs and metrics they use in connection with R&D are underutilized, and key metrics are even missing.

6.4.1 Velocity
While research shows that velocity is used extensively by agile teams and has a high impact on the work done (Javdani, Zulzalil, Ghani, Sultan, & Parizi, 2012; Kupiainen, Mäntylä, & Itkonen, 2015), it is not tracked by almost any of the teams at CCN (CCN4). According to one interviewee, tracking velocity is critical for being able to plan sprints appropriately (CCN4). The impact of not utilizing this for sprint planning can be seen in that all stories committed to a sprint are almost never completed, and often only around 50-60% of them are (CCN8). If more teams at CCN were to track velocity, they would gain knowledge of their capacity, which could help with planning, leading to more accurate estimates on higher planning levels (e.g. PLM, see section 4.3.2).

6.4.2 Utilize Measurement of Outcome
While there are several options for tracking some kind of output in software development, not all of them are appropriate in agile software development, as discussed in chapter 3.9. Since one of the principles of the agile manifesto is “maximizing the amount of work not done”, while reaching a satisfactory outcome, only tracking output with a quantity-centric metric would be counter-productive. This view is not only shown in theory, but shared in many parts of the organization (CCN1, CCN3, CCN4, CCN5). It was even shown to some extent in non-agile parts of the organization as a worthy goal (CCMU1, CCMU3). However, at the same time as there is an awareness regarding this, the only way that outcome is measured is via net promoter survey, gauging their satisfaction with CCN as a whole, which naturally results in a single digit representing NDC as a whole. For internal customers, there is a more complete customer survey, but it seems like these are not used as feedback for the development teams. Consequently, neither of these work as a basis for outcome-based measurement. This means that new ways to gauge customer satisfaction are needed.

A proper evaluation of outcome should be able attributable to a specific team in order for them to be able to learn from it. In order to do this, the teams will likely have to employ vastly different solutions due to the different constellations used for projects, and the different accessibility to customers which they have. Teams which work alone on a product can use a survey or similar to gauge the satisfaction, while teams which are not alone on a product may have to use product or project management as a connector to the customer to a larger extent. Projects which are not client projects, and which do not have users already will instead have to rely on product management and other stakeholders instead of external customers. However, in this case it is important to evaluate the satisfaction regarding the outcome, and not how well the initial plan was followed as is typical in traditional project management (see section 3.8). Instead the agile principle of “Responding to change over following a plan” should be followed, and the evaluation be done accordingly (Beck, et al., 2001).
7. Conclusions

This chapter summarizes some of the issues presented in chapter 5, restates the research questions and summarizes relevant parts of chapter 5 and 6 in order to answer them. Furthermore, this chapter presents several tangible suggestions for different actions the case company could take to combat the challenges presented in previous chapters. They are based on theory in industrial management, as well as empirical evidence gathered during the case study.

7.1 Summary of Issues

There are many challenges associated with the R&D processes used. According to many stakeholders at CCN, the time schedule is rarely followed though, with many delays occurring, often without forewarning. For example, according to one interviewee, many times the goal fulfillment-rate of sprints varies greatly, with an average of around 50-60% completed goals, sprint after sprint. There is a slow, and occasionally non-existent flow of information, and a lack of customer involvement, which is atypical for an R&D-organization using agile principles. Furthermore, important decisions are not always supported on levels on which they have a large impact, which impacts trust between people in the organization. While some believe this occurs independently of the methodology used, processes in place and local culture, others think that R&D at CCN can be tougher to manage and monitor due to them having a weaker structure, and using other methodologies than most of the organization. This, among other things, results in a desire from the rest of the case company for the R&D department of CCN to conform to a larger degree than they do at the present time. Others, still, believe that making the methodology more rigid would only hide these issues under an illusion of having a more rigid plan and structure, which would subsequently have to be updated when the illusion is shattered.

7.2 Research Questions

**RQ 1:** What areas need support in large organizations in order for agile R&D processes to succeed?

Three main areas were identified in the empirical findings. These were "Collaboration and Communication", “Planning & Organizing” and “Tools”. Within collaboration and communication, many symptoms for the large organizational distance can be seen, such as a low access to the customer and a slow information flow. Issues within planning and organizing include the internal conflict between standardization and autonomy, as well as between stability and agility. Within tools, two main areas were identified as of high importance in the findings, reporting and documentation and continuous integration.

While all of these areas are of high importance, the ones which CCN would gain the most from looking at are “Collaboration and Communication” and “Planning & Organizing”. Largely due to the fact that “Tools” is a focus area for many within CCN already, where there is significant progress being made with wide support on all levels of the company.

**RQ 2:** What can be done to provide support where it is lacking?

While there are many possible suggestions given in chapter 6, such as an increased adoption of scaled agile practices, increased collaboration with customers and increasing the
implementation of metrics in the company, some are more critical and/or will give a higher reward than others. These are described here.

7.2.1 Encourage more frequent face-to-face communication
As described in the discussion chapter, particularly section 6.2.1, frequent face-to-face communication both between departments and with the client has many benefits and can alleviate many of the issues facing large organizations using agile practices. In particular, frequent, informal face-to-face communication lowers distortion of information, raises trust and increases the ability to give and receive immediate feedback, an extremely important part of agile software development. Additionally, frequent face-to-face communication and more communication with clients can increase productivity substantially for agile teams (see sections 3.3 and 6.2). Failure to achieve this may result in further deterioration in trust, more delays, and more issues with overcommitment. Effectively, product managers and developers need more frequent, informal communication, that is, there needs to be more communication outside formal meetings, and as one interviewee put it “more of a partner relationship”. Furthermore, clients need to participate more in the development process. While many clients will see the benefits of this easily, some may not and will need convincing. However, if this is raised early in the negotiation, or development process, it will raise the chances of success.

7.2.2 Strengthen the SM’s role
The most important role in strengthening the work process and ensuring an agile team works efficiently is the SM. While CCN have a SM in every team, they typically split their time between development and SM-duties. Implementing some of the suggestions presented here likely requires considerable effort in order to set up changes as well as sustaining them. This will require significant devotion from SM’s, who are responsible for their teams following the scrum process. Therefore, SM’s should be allowed additional time for their SM duties, and should not be expected to contribute in the same way towards a team’s regular tasks. In addition to this, I propose implementing a forum for SM’s to coordinate and discuss activities between teams. There are multiple purposes for this, one is to learn from each other’s successes and failures. Another is to bring teams to the same level in important areas such as CI-adoption. Besides the benefits described above, devoting more resources to the SM role and facilitating collaboration between teams will strengthen the processes that are already in place, which will increase the performance of teams in the long run.

Another reason to strengthen the SM’s role is to help with the adoption and standardization of scalable agile practices. That not all agile practices scale well is something which managers are generally aware of, however, there is no consensus in the organization as to which aspects of scrum that are generally scalable. This exists in literature however. Adopting a common picture of what is possible, what is desirable, and where the organization is at present would help push CCN forward in this regard. A first step would be to gain a complete picture of where on the road to scaled agile adoption everyone is. The second would be to see what individual teams do well, and then standardize those aspects across all teams at CCN.

7.2.3 Evaluate, discuss, and learn from project outcomes
Both the theoretical data and the empirical data presented in this study conclude that the best way to evaluate the success of a product is to look at the outcome rather than the output of
projects. The aim should be to give agile teams feedback on how much value they have delivered to the customer or measure the satisfaction of customers, rather than quantitatively measuring the code produced or the fulfilment of the initial goals of the project. This cannot be accomplished with the practices set in place today for several reasons. One of the primary reasons is that customer satisfaction is not measured on a level which is granular enough. To evaluate outcome and ensure it is usable, any measurement should be attributable to a single team. To do this, different types of projects will need different solutions due to many factors, such as the differing organizations and the differing access which projects can have to customers and users. While the case company strives to provide cost efficient solutions to many types of customers, they also need to stay at the forefront of technology and deliver the maximal possible amount of value to users in order to stay competitive. Doing this without fully utilizing customers and users is not possible.

7.3 Sustainability
From the start, the purpose of this study has been tied to improving the R&D processes used in large organizations. This has involved looking at different aspects which influence this directly, such as information flow and planning. However, there is an aspect which may not touch directly on this, but which has long term implications both inside and outside of the company’s walls, and that is sustainability. Sustainability can be divided into three main pillars: social, economy and environment, and one less often discussed fourth pillar: human (Munn, 2002). According to Penzenstadler et al. (2012), there is currently little research on this topic within software development. However, that does not stop us from reflecting on the role that the choice of work processes used may play in these different aspects of sustainability.

Large organizations using agile software development can be seen to be more sustainable due to a few different factors. Cross functional teams are important to many different agile methodologies. These bring together people from several disciplines, and educational backgrounds and can therefore be said to promote diversity compared to organizations which typically group specialists from the same field and functions together. Furthermore, research has shown that employees in companies using agile development are generally more satisfied with their jobs, as well as more engaged (Cohn, 2010). This is not only good from a social sustainability perspective, but also from an economic perspective, since job satisfaction and engagement drive many business outcomes, such as customer satisfaction, productivity, number of accidents, employee turnover, and even profit (Harter & Schmidt, 2002). According to Harter & Schmidt (2002), the greatest difference between low and high employee engagement can be seen in its effect on employee turnover, which as discussed in section 6.1.2 has a great effect on team stability, which in turn affects development velocity. This should result in companies using agile methodologies gaining additional productivity enhancing benefits due to having a lower employee turnover. In the case company, this may contribute to India, where agile methodologies are not used to a large extent, having a larger turnover than, for example, the Stockholm office.
8. Bibliography


9. Appendix A: List of interview subjects

9.1 Phase 1

<table>
<thead>
<tr>
<th>Date</th>
<th>Reference</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017-01-19</td>
<td>CCN1</td>
<td>Team Leader</td>
</tr>
<tr>
<td>2017-01-20</td>
<td>CCN2</td>
<td>Team Leader</td>
</tr>
<tr>
<td>2017-01-24</td>
<td>CCN3</td>
<td>Team Leader</td>
</tr>
<tr>
<td>2017-02-03</td>
<td>CCN4</td>
<td>Team Leader</td>
</tr>
<tr>
<td>2017-02-03</td>
<td>CCN5</td>
<td>Head of R&amp;D</td>
</tr>
<tr>
<td>2017-02-07</td>
<td>CCN6</td>
<td>Developer</td>
</tr>
<tr>
<td>2017-02-13</td>
<td>CCN7</td>
<td>Scrum Master / Developer</td>
</tr>
<tr>
<td>2017-02-13</td>
<td>CCN8</td>
<td>Product Manager</td>
</tr>
<tr>
<td>2017-02-21</td>
<td>CCN9</td>
<td>Head of Service Operations</td>
</tr>
</tbody>
</table>

9.2 Phase 2

<table>
<thead>
<tr>
<th>Date</th>
<th>Reference</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017-03-20</td>
<td>CCN10</td>
<td>Director Product Management</td>
</tr>
<tr>
<td>2017-03-28</td>
<td>CCN11</td>
<td>Developer</td>
</tr>
<tr>
<td>2017-03-29</td>
<td>CCN12</td>
<td>Project Manager</td>
</tr>
<tr>
<td>2017-03-30</td>
<td>CCMU1</td>
<td>Head of Global PMO</td>
</tr>
<tr>
<td>2017-03-30</td>
<td>CCMU2</td>
<td>Project Manager</td>
</tr>
<tr>
<td>2017-03-30</td>
<td>CCMU3</td>
<td>Head of Department</td>
</tr>
<tr>
<td>2017-03-31</td>
<td>CCMU4</td>
<td>Head of Product Management</td>
</tr>
</tbody>
</table>
10. Appendix B: The Product Lifecycle Management model