Improving the software development processes between Development and Operations departments at a telecommunications operator

Elias Söderström

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology

Espoo 03.10.2016

Thesis supervisor:

Doc. Kalevi Kilkki

Thesis instructor:

M.Ec. Arto Kiljunen

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Elias Söderström

Title: Improving the software development processes between Development and Operations departments at a telecommunications operator

Date: 03.10.2016 Language: English Number of pages: 52+6

Department of Communications and Networking

Professorship: Data Networks

Supervisor: Doc. Kalevi Kilkki

Instructor: M.Ec. Arto Kiljunen

This Master's Thesis studies the improvement of software development processes of a telecommunications company by utilizing agile development methods. Specifically, this thesis focuses on improving the processes of a development organization within the telecommunications company that has been split into two distinct development departments. In order to apply agile development methods effectively, analysis of the situation had to be performed. The primary analysis method utilized during this thesis was a set of interviews conducted together with several employees of the company. Additionally, a quantitative data analysis was performed using Six Sigma process development methods in order to complement the interview results. The results from the analyses were that the telecommunications company needs to apply certain aspects of agile development, such as preferring local development teams, and improving cross-team collaboration and knowledge transfer, while avoiding some potentially ineffective aspects of agile development that would not be applicable in the specific situation of the company. Additionally, the company needs to focus on improving the processes related to development of newer, younger telecommunications systems, as this kind of system development particularly benefits from agile development.

Keywords: Agile Development, Process Development, Process Improvement, Software Development, Six Sigma

AALTO-YLIOPISTO
SÄHKÖTEKNIIKAN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Elias Söderström

Työn nimi: Kehitys- ja ylläpito-osastojen välisten ohjelmistokehitysprosessien parantaminen teleoperaattoriyrityksessä

Päivämäärä: 03.10.2016 Kieli: Englanti Sivumäärä: 52+6

Tietoliikenne- ja tietoverkkotekniikan laitos

Professuuri: Tietoverkot

Työn valvoja: Dos. Kalevi Kilkki

Työn ohjaaja: KTM Arto Kiljunen

Diplomityö tutkii tietoliikenneyrityksen ohjelmistokehityksen prosessien parantamista ketterän ohjelmistokehityksen menetelmien avulla. Erityisesti, diplomityö pyrkii kehittämään yrityksen ohjelmistokehitysyksiköiden välisiä prosesseja, sillä yksiköt ovat yrityksessä jaettu kahteen erilliseen kehitys- ja ylläpito-osastoon. Jotta ketterää kehitystä pystytään soveltamaan tehokkaasti, yrityksen nykyisestä tilanteesta piti suorittaa kattava analyysi. Pääasiallinen analyysimenetelmä diplomityössä oli haastattelu, joka toteutettiin yhdessä yrityksen työntekijöiden kanssa. Haastattelutulosten täydennykseksi työssä toteutettiin kvantitatiivinen data-analyysi käyttäen Six Sigma -prosessikehitysmenetelmiä. Diplomityön tulosten perusteella yrityksen on sovellettava eräitä ketterän kehityksen menetelmiä, kuten paikallisten kehitystyöryhmien perustamista sekä ryhmien välisen yhteistyön ja tiedonvälityksen parantamista, mutta samalla yrityksen tulee välttää tiettyjä sille sopimattomia ketterän kehityksen menetelmiä, jotka voisivat olla haitallisia yrityksen erityisessä tilanteessa. Lisäksi yrityksen tulee keskittyä iältään nuorten tietoliikennejärjestelmien kehitystyön prosessien parantamiseen, sillä tämän kaltaisien järjestelmien kehitys hyötyy eniten ketterän kehityksen menetelmien käyttöönotosta.

Avainsanat: Ketterä kehitys, Prosessikehitys, Prosessien parantaminen, Ohjelmistokehitys, Six Sigma

Preface

As this Master's Thesis was requested by the Finnish telecommunications Company, I would like to thank all the employees of the Company from participating in the study phase of this thesis and providing me with invaluable experience and support for such an enormous task. Specifically, I would like to thank Oskari Kurki for giving me the opportunity to work on the subject of this thesis and Arto Kiljunen for guiding and supporting me in the completion of this thesis during and after the study face performed while working for the Company.

I would also like to thank the staff of Aalto University School of Electrical Engineering for supporting me in writing this thesis. My appreciation goes to Professor Raimo Kantola who began the thesis project with me, to Professor Jörg Ott who took responsibility for the thesis intermediately, and in particular Docent Kalevi Kilkki who ultimately took the supervisory role for this thesis and supported me during my hardest moments with the project.

Finally, I would like to express my gratitude to all my friends and family who have supported me during the whole lifetime of this thesis project. Without them graduation would have been but wishful thinking.

Helsinki 3.10.2016

Elias Söderström

Contents

Abbreviations

ASD	Adaptive Software Development
AM	Agile Modeling
AUP	Agile Unified Process
BSS	Business Support Systems
CEO	Chief Executive Officer
DMAIC	Define, Measure, Analyse, Improve, Control
DSDM	Dynamic Systems Development Method
Evo	Evolutionary Project Management
FDD	Feature Driven Development
FTE	Full-Time Equivalent
IT	Information Technology
ITIL	Information Technology Infrastructure Library
LSD	Lean Software Development
MIT	Massachusetts Institute of Technology
OSS	Operational Support Systems
PD	Product Development
SIPOC	Supplier, Input, Process, Output, Customer
SO	Systems Operations
TPS	Toyota Production System
TQM	Total Quality Management
XP	Extreme Programming

1. Introduction

As software has rapidly evolved over the past few decades, software development, its processes, and its end products have become increasingly more complex, specific, and demanding [BPC07]. In the software development industry there has been a push towards a more fast paced, gradual, and flexible direction in terms of working methods. This is the same direction chosen by a major Finnish telecommunications company (henceforth referred to as the Company) which will be the main focus of this thesis. The goal of the company is to attain this direction by updating and improving its processes and working methods of its software development department.

The main reason for this improvement operation is that processes within the company have become increasingly convoluted and cumbersome. Different working units have become too isolated from each other, projects and processes are inflexible, and there are noticeable issues with communication and co-operation. The high-level solution to this problem is an attempt to apply agile development to processes and working methods within the Company. This is a natural direction to be chosen because agile development is already used in certain parts of the Company.

Agile development (often called agile software development) is an umbrella term for a selection of methods for improving software development. These methods encompass themes such as developing software incrementally, improving quality of development, importance of co-operation, efficient use of resources, and prioritization of development. Most of these themes will be included in this thesis, although more emphasis will be given to certain themes than others.

This thesis aims to incorporate agile development into the working processes of the software development of the Company and increase the efficiency, proficiency, and co-operation of different working units. The research problem for the thesis consists of the following questions:

1. How can software development processes between the development and support and maintenance departments of the Company be improved.
2. How can the handover of software and transfer of knowledge related to that software be improved between these departments
3. How can collaboration between software development and other relevant departments and stakeholders within the Company be improved?

The hypothesis, fabricated by analysing the research problem questions above, is that by applying agile development into software development processes, departments within the Company are able to align and co-operate more effectively with each other and consequently produce better quality software in less amount of time and with fewer resources. Other process development methods, aside from agile development specifically, will also be utilized in attempting to improve the work processes of the Company.

Certain intermediate problems have to be solved between choosing agile development as the solution and implementing that solution. The current state of the Company in terms of its “agility”, i.e., how well it already conforms to the ideology of agile development, needs to be determined. The main problem areas where agile development will be applied also have to be defined within the Company, after which the main areas of focus need to be

defined even further. For example, the main areas of interest within the processes and departments responsible for software development will be defined more specifically. Additionally, how to actually apply agile development needs to be determined as well.

The scope of this thesis will be within the development, support, and maintenance of software in the Company, and within the collaboration of departments responsible for those tasks. More specifically, the main focus will be on change management and the handover process of software development, both of which heavily involve the departments mentioned above. Additionally, this thesis will focus more on the development of newer systems within the Company, as opposed to older legacy systems. Other important aspects of software development that will be touched upon include topics like resource management and relations between development and business but essentially these are topics that require additional studies that focus specifically on them. Primarily, the scope and focus of this thesis was chosen iteratively by examining the issues in and solutions to the underlining research problem and hypothesis. In short, the main reason for such a narrow scope and focus is the fact that a broader inspection of all the facets of software development processes requires multiple studies on each individual facet, but other reasons will be elaborated upon in later sections.

The rest of this thesis is divided into five separate sections. First, theory and basics concerning agile development and other process development methods will be presented. Next, the study methods used during this thesis will be described in detail. After this, the analysis section will determine the initial state of the focus of the study and describe results obtained from utilizing the study methods. At the end of the analysis section, recommendations for improving processes and applying agile development methods are presented, and finally the conclusion section will conclude the study by reflecting on the results and the study itself.

2. Process Development

2.1. Agile Development

During the 1990s, new and different methodologies for software development began to appear and challenge the old and cumbersome nature of the traditional waterfall-orientated methods. These methodologies included and highlighted several topics for software development, such as collaboration and communication, iterative working methods, frequency and value of software deliveries, team organization, and scheduling. [Agi15]

Eventually, these new methodologies would lead to the creation of the Agile Manifesto. In February 2001, seventeen experts of different methodologies agreed on the Agile Manifesto at a summit [Agi15b]. The manifesto, in its entirety, reads as follows [Bec01]:

> *We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:*
>
> *Individuals and interactions over processes and tools*
>
> *Working software over comprehensive documentation*
>
> *Customer collaboration over contract negotiation*
>
> *Responding to change over following a plan*
>
> *That is, while there is value in the items on the right, we value the items on the left more.*

The manifesto is supplemented by the twelve principles of agile software that further elaborate on the idea behind agile development [Bec01b]:

> *Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.*
>
> *Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.*
>
> *Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.*
>
> *Business people and developers must work together daily throughout the project.*
>
> *Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.*

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

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity--the art of maximizing the amount of work not done--is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Essentially, agile development attempts to alter the ways software was – and still is – traditionally developed. Traditional software development generally means development similar to the waterfall process model which can be described as methodical, plan-orientated, and thus very rigid.

As the manifesto and list of principles suggest, agile development shifts the focus from cumbersome processes and specified tools to individual software developers and interaction. While processes and tools are not dismissed completely, interaction and communication of personnel is deemed much more important. This not only includes communication between developers themselves but also between them and end users, business personnel, and management. The importance of face-to-face communication is heavily emphasized in agile development, especially between developers. In order to achieve any kind of agility in software development, developers must be provided with sufficient support and trust to do their job. This means that developers must have the power and responsibility to do autonomous decisions regarding their development work. Frequent self-reflection and feedback are also important for constantly improving the development team.

One of the key concepts of agile development is the idea that software that is delivered to customers should have value. This not only means that the software should be working but it should also be functional, preferably by itself, meaning that customers do not have to wait further deliveries in order to get a working product. Simultaneously, the development of software in agile development is divided into intervals and delivered in smaller parts and more frequently than in more traditional software development methods. Amidst all of this, the development team always focuses on the deliveries themselves instead of producing excessive documentation for the software.

The customer – be it end users or some larger entity that has ordered the software – is a central focus point in agile development. Feedback from customers is important but collaboration with them should be a priority throughout the development cycle. Customers should preferably be integrated into the development process from the very beginning in order to provide guidance, ideas, and feedback for the development team. An open and free relationship with customers is preferred to restrictive and formal contracts.

In agile development, change and attitude towards it are extremely important. Instead of avoiding making changes in the development cycle, they are accepted and often even embraced. Typically to agile development, changes – even major ones – can be implemented also in later stages of development if required. This means that instead of following a waterfall-orientated rigid plan where much needed changes to software might be ignored, changes can be easily implemented due to the iterative nature of frequent deliveries in agile development. The favourable view on change is one of the reasons why agile development is suitable for business environments where changes can happen frequently and unexpectedly and must be reacted to.

While all of the aforementioned characteristics are essential to agile development, the more traditional aspects of development practices are not completely ignored. For example, documentation, contracts, and planning are acknowledged to be important in development but they must not be prioritised over the core features of agile development. In other words, contracts and development plans should be flexible and applicable to changes, and producing documents should never be the main purpose of development.

Agile development does not require implementing all of its principles and characteristics into development processes in order to make development agile. Typically, implementing every single aspect of agile development is not feasible depending, e.g., on the type of development teams, individuals, corporate structure, or culture. Table 1 displays some project characteristics that are either agile or non-agile, and it is a simple and useful aid for determining whether agile development is suitable for a particular development project. It should be noted, however, that this table greatly simplifies agile development, other development methods, and the nature of development projects. For example, certain non-agile development methods can still have agile characteristics while not belonging under the agile development umbrella, hence non-agile methods can still be considered for utilization even if project characteristics might appear agile. Nevertheless, the table is an easy and simple resource for people who are not yet familiar with agile development.

Table 1: Project Characteristics - Agile versus Non-Agile [Sch04, p. 8]

Project Environment		**Project Characteristic**	
Category	**Variable**	**Agile**	**Non-Agile**
The Development Team	Communication Style	Regular Collaboration	Only When Necessary
	Location	Collocated	Distributed
	Size	Up to 50 People	More than 50 People
	Continuous Learning	Embraced	Discouraged
Project Management	Management Culture	Responsive	Command and Control
	Team Participation	Mandatory	Unwelcome
	Planning	Continuous	Up Front
	Feedback Mechanisms	Several	Not Available
The Customer	Involvement	Troughout the Project	During Analysis Phase
	Availability	Easily Accessible	Hard to Reach
Processes and Tools	Team Input	Team Has the Last Word	Team is Told What to Use
	Amount	Just Enough	More Than Enough
	Adaptability	May Be Changed	May Not Be Changed
The Contract	Requirements and Dates	Flexible	Fixed
	Cost	Time and Materials	Fixed

Agile Methodologies

As mentioned before, agile development encompasses several different agile methodologies instead of being a complete methodology by itself. These methodologies usually include only some of the principles and aspects of agile development, or they focus on certain aspects more than others. A methodology that would contain every single aspect of agile development does not exist and is not viable. As was alluded to earlier, a methodology is not required to contain every aspect of agile development in order to be agile.

Of all the agile methodologies, Extreme Programming (XP) is the most common, whereas Scrum comes closest to describing current agile development in the Company. The next list includes some of the more common agile methodologies, of which Scrum will be more closely discussed afterwards.

Incomplete list of agile development methodologies:

- Adaptive Software Development (ASD)
- Agile Modeling (AM)
- Agile Unified Process (AUP)
- Crystal Methodologies
- Dynamic Systems Development Method (DSDM)
- Evolutionary Project Management (Evo)
- Extreme Programming (XP)

- Feature Driven Development (FDD)
- Lean Software Development (LSD)
- Scrum

In order to understand the differences between these agile methodologies, it is useful to classify them according to some criteria. A typical criterion, in the context of agile development, is the flexibility (or inflexibility) of the methodology in terms of the amount of documentation and other formalities required in the methodology. Another typical criterion would be the number and length of development iterations in development projects. For example, a waterfall-type methodology would have no iterations and would require extensive documentation and formalities. Figure 1 demonstrates how certain agile methodologies can be classified and portrayed according to these criteria. [Lar03, p. 60]

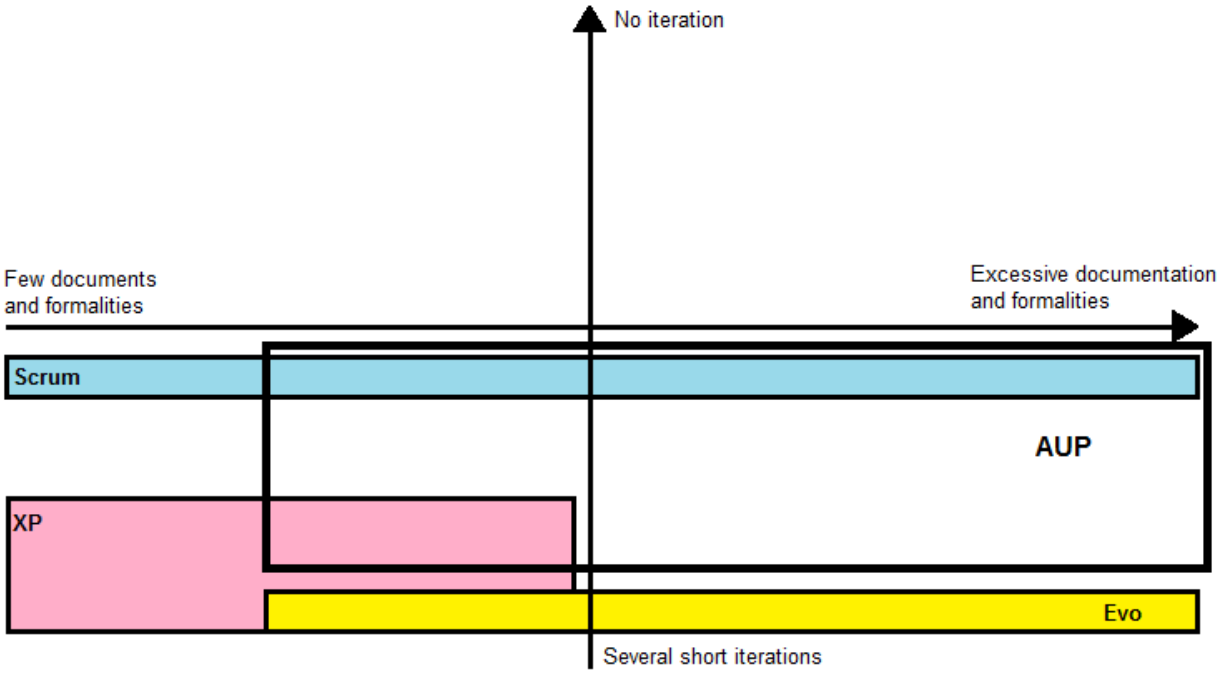


Figure 1: Classification of agile methodologies [Lar03, p. 60]

As the figure suggests, agile methodologies can vary significantly from each other in the context of these criteria. Even though most of the methodologies presented in the figure tend to be situated near the iterative end of the spectrum, the level of iteration can vary, and especially the level of documentation and other formalities is not defined specifically in any of these methodologies. When comparing the methodologies, there are some apparent differences however. For example, the amount and length of iterations in Scrum is quite strictly defined but the amount of documentation and other formalities required is left to the discretion of the implementer of the methodology, whereas in XP the amount of documentation and formalities is more strictly defined but the level of iteration required is more flexible.

2.1.1. Scrum

Scrum is one of the most commonly used methodologies under the agile development umbrella. It was conceived in 1993 for utilization in software development but due to its simplicity it is applicable in other fields as well. The name of the methodology derives from a similarly named formation used in rugby, the idea being that software development teams using Scrum function in similar ways to that of rugby players in a scrum formation during a rugby game. [Scr16]

Scrum is a relatively simple agile methodology, and therefore can be fairly easily applied to projects that require adaptation to change and include stakeholders who have a close interest in the project. Due to its simplicity, it does not explicitly detail how the project team should function. [Sch04, p. 23]

In Scrum, a project is divided into several parts, i.e., sprints (the amount of sprints depends on the size of the project and timescale of the sprints) and each of them will lead to delivering a functioning portion of the project to the customer. The requirements for a sprint are acquired from the product backlog which consists of all the prioritized features of the product that are designated by the customer. Changes to the project are implemented by adding features to the backlog, however changes do not affect ongoing sprints and are only reacted to between sprints. A product manager is assigned to modify feature priorities by examining the backlog. [Pre10, p. 82-84]

During the sprint (usually 30 days), short, approximately 15 minute scrum meetings are held daily. In these meetings, each project team member informs what he/she did since the last scrum meeting, reports any current obstacle or problems, and announces what he/she will work on before the next meeting. A team leader is assigned as the scrum master to conduct these meetings. At the end of a sprint, a demo of the product is delivered to the customer. This demo does not have to include all features of the project, but only the features selected for that particular sprint. However, the demo has to be functional in terms of the features included in the sprint. This allows the customer to assess the features implemented during the sprint. Figure 2 illustrates the chronological order of a Scrum process. [Pre10, p. 84]

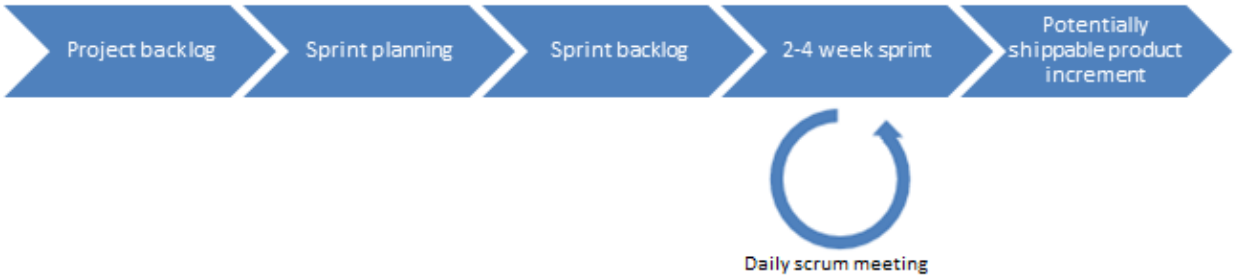


Figure 2: Scrum process chronologically [Scr16]

2.2 Process Improvement

In the context of this thesis, process development is used as a colloquial term for simply developing processes, i.e., making processes better regardless of any particular topic, field, or methodology, and as such the term process development encompasses every kind of tool, strategy, principle, and framework related to developing processes. Process improvement, on the other hand, is an umbrella term similar to agile development but focuses on quality improvement in processes and emphasizes utilization of statistical methods.

The idea for this type of process improvement materialized as early as the 1970s. By the 1980s, a specific process improvement framework, called Total Quality Management (TQM), was developed. TQM is one of the most commonly known process improvement strategies, and it encompasses several topics and themes similar to agile development, such as participatory teams, focus on the customer, and inclusion of suppliers and other stakeholders. [Mon08]

TQM was far from an unambiguous success, however. In several recorded cases, TQM was applied too rigorously into all aspects of processes in order to find all possible issues and eradicate them regardless of the impact it would cause to customers [Pro09]. Other common reasons for the failure of TQM include the lack of support from top management and business, incompetent utilization of statistical methods provided by TQM, and general lack of focus and enthusiasm for implementing TQM properly. Other similar process improvement strategies have emerged since, some even before TQM was conceived, but they have had similarly poor results on actual process improvement. Six Sigma, developed in the late 1980s, saw greater success compared to TQM by achieving to incorporate top management and business leaders in the improvement of processes, and subsequently gained a massive following amongst the process improvement crowd.

The list below consists of some well-known Process Improvement methodologies, of which lean and Six Sigma will be focused on in the next section.

Incomplete list of process improvement methodologies:

- Lean
- Quality control
- Six Sigma
- Total Quality Management
- Value Engineering
- Zero Defects

2.2.1 Lean

Lean (also known as lean thinking or lean manufacturing) is a process improvement method that originally derives from a quality control method called the Toyota Production System (TPS) that was introduced by Toyota, as the name suggests, in the 1950s. TPS itself was devised by applying earlier quality control methods within the Toyota company. In the late 1980s, the term lean was first coined when TPS was summarized and modified by researchers in the Massachusetts Institute of Technology (MIT) for application in other fields as well. [Ber16]

The primary idea behind lean is to minimize waste in processes and maximizing value to the customer which in turn will lead to increasing value while using fewer resources. Lean concentrates in reducing waste horizontally across entire product and service development processes, focusing on the flow of products and services through value driven processes – or streams – instead of trying to optimize individual aspects of those processes. The result of this approach should be processes that require fewer of any kind of resource and that produce better quality products and services. Processes will also be more flexible in terms of environmental changes and more efficient in terms of information management. As per the original idea in its conception, lean is applicable in most fields and industries, not just manufacturing. [Lea16]

Lean includes five simple principles for implementation that are as follows:

1. Identify value
2. Map the value stream
3. Create flow
4. Establish pull
5. Seek perfection

In short, the idea behind these principles is to identify what produces value to the customer in a product or service, specify all steps in the value stream of the product or service and remove steps that do not produce value if possible, and optimize the occurrence of the steps that do produce value in order to ensure an efficient product or service flow to the customer. After this flow is created, it can be altered to respond to customer demand, i.e., pull. After all of the above, the application of the principles can begin again until the value of the product or service is perfect and no waste occurs. [Lea16b]

As the description above suggests, lean shares some key aspects with agile development, such as focus on the customer, optimization in terms of resources, and information management. It is not then surprising that after lean was applied to software development in the 1990s, creating the sub-methodology of lean software development, it was later recognised as an agile approach to software development after agile development had been conceived. The principles of lean were even used to measure and justify the usefulness of agile development in its early days by demonstrating how agile development methods could reduce waste and increase development efficiency. [And12]

While lean software development mostly retains the original lean principles, lean cannot be simply applied to software development as is since software development is a very different field compared to, e.g., manufacturing, which was the original field of lean [Pop13, p. 15]. Lean or lean software development are not highly specified processes or methods anyway, and as such, utilizing lean software development should result in a unique software development process within a product pipeline or company or whatever the context by adopting the core principles and values of lean software development [And12].

As alluded to earlier, the principles and values of lean are still intact in lean software development but additionally it emphasizes aspects that are essential in modern software development and also in agile development, such as focus on people, embracing complexity and uncertainty, and transparency [And12]. Regarding aspects such as quality, iteration, and uncertainty, it is paramount to make the distinction between software development and more traditional fields, for instance manufacturing, in the context of lean. In manufacturing, uncertainty and variation in quality are not desirable and iteration generally equals wastage,

whereas in software development iteration and uncertainty are welcomed and quality is defined by the needs of the customer [Pop13, p. 16].

2.2.2 Six Sigma

While Six Sigma can be classified as a process improvement methodology, it is far from an unambiguous term. The history of Six Sigma starts at the 1980s when Motorola began developing production strategies that focused on reducing defects on its products, as they realised that products with no defects had a vastly lower failure rate in normal use. Since then, several other companies have adopted and contributed in researching Six Sigma, the most notable example being General Electric under its then Chief Executive Officer (CEO) Jack Welch. [Bra02, p. 5-6]

The term Six Sigma can have several different definitions, and as such there is debate about what it actually means. Since sigma represents standard deviation in statistics, six sigma can be construed as a deviation of 0.002 per million, and in the context of process improvement this usually means deviation in quality, e.g., 0.002 defects per million [Dhi14, p. 5-6]. In a more practical sense, Six Sigma can be considered as a set of tools for improving processes [Mun08, p. 5]. This set of tools consists of several qualitative and quantitative process improvement techniques, some of which will be discussed in detail in later sections. Six Sigma can also be seen as a philosophy where processes can be defined, measured, analysed, improved, and controlled. And finally, Six Sigma is often regarded as a methodology that defines five steps for improving processes, those steps being mentioned in the philosophy definition of Six Sigma and abbreviated as DMAIC. Even though there are other Six Sigma methodologies, DMAIC is the most commonly used [Mun08, p. 5].

Six Sigma and DMAIC share some common aspects with agile development in their implementation. In DMAIC, employees need to be given proper time and resources in order for them to complete each step. Six Sigma is an all-encompassing methodology in terms of the workforce of a company, meaning that everyone within the company should be involved in implementing Six Sigma. Most notably, both the Information Technology (IT) department and the business department should be in central roles in implementing Six Sigma, specifically regarding the transfer of knowledge and information. [Mun08, p. 14]

As mentioned previously, the DMAIC methodology consists of five different steps. The methodology starts by defining the issues faced by customers. The next step requires measuring and collecting data from the processes affiliated with the issues. In the analysis step, the collected data and the process itself are analysed in order to pinpoint problems in the process. Next, the process should be improved with the help of the analysis. And finally, the control step requires monitoring the process in order to ensure the improvements are being upheld. For each step, there are a significant number of different tools and techniques to help perform the steps efficiently. Some of these tools will be discussed in later sections and utilized in the context of this thesis. [Mun08, p.14]

Even though Six Sigma shares some similarities with lean, the end goals of these methodologies are different. Six Sigma focuses on minimizing variability in processes and subsequently defects in quality control, whereas lean concentrates on reducing waste and optimizing resource usage, which is often time. By combining these two methodologies into what has been dubbed as Lean Six Sigma, the methodologies provide two sets of tools that complement each other and help improve processes in terms of both eliminating defects and saving resources. [Ber16, p. 15]

When assessing agile development as a whole, it is easy to see why the Company wants to implement it further into its software development processes. Agile development offers a flexible, swift, and collaborative way of developing software, all of which is extremely important in the rapidly shifting business environment of the Company. Even in its suboptimal situation, where employees of the Company are heavily distributed and fixed schedules are demanded, agile development is still a lucrative methodology for utilization since it itself is flexible and can be applied in circumstances where not every single aspect of agile development is desirable. The process improvement methods discussed in this section will complement agile development well. They are easily compatible with agile development, and while any particular process improvement methodology has not been systematically implemented in the processes of the Company, they are to some extent already utilized within the Company in a more general sense. Six Sigma specifically will be considered as more of a set of tools rather than an all-encompassing methodology in the context of this thesis, as will be evident from the following sections.

3. Study Methods

Initially, several different study methods were considered during the early stages of this thesis and the study that was concluded within the Company. The original intent was to choose efficient and accurate study methods that would be able to highlight problems in the software development processes of the Company and to which agile development could then be applied.

The study methods that were eventually selected for the thesis were determined organically, i.e., they were sequentially chosen as the study progressed. In the case of interviews, it was clear from the very start that information, opinions, and suggestions regarding agile development and its current state within the Company were needed, hence interviewing relevant personnel from the Company was an obvious method for the study. The interviews are considered to be the main qualitative study method in this thesis.

Later on during the study phase, it became evident that a more quantitative approach for the thesis was desired as well. Ultimately, two different analysis methods from the Six Sigma family – Fishbone and SIPOC, both of which will be presented in detail in later sections – were chosen in order to help in identifying problem areas in the software development processes of the Company. Six Sigma was a natural choice for a quantitative methodology since as a process development method it comfortably complements agile development. Additionally, the instructor of this thesis was already familiar in Six Sigma related topics, making consultation in terms of the methodology convenient. The Six Sigma analysis itself concludes with a simple statistical data analysis concentrating on the problem areas found in the Six Sigma analysis.

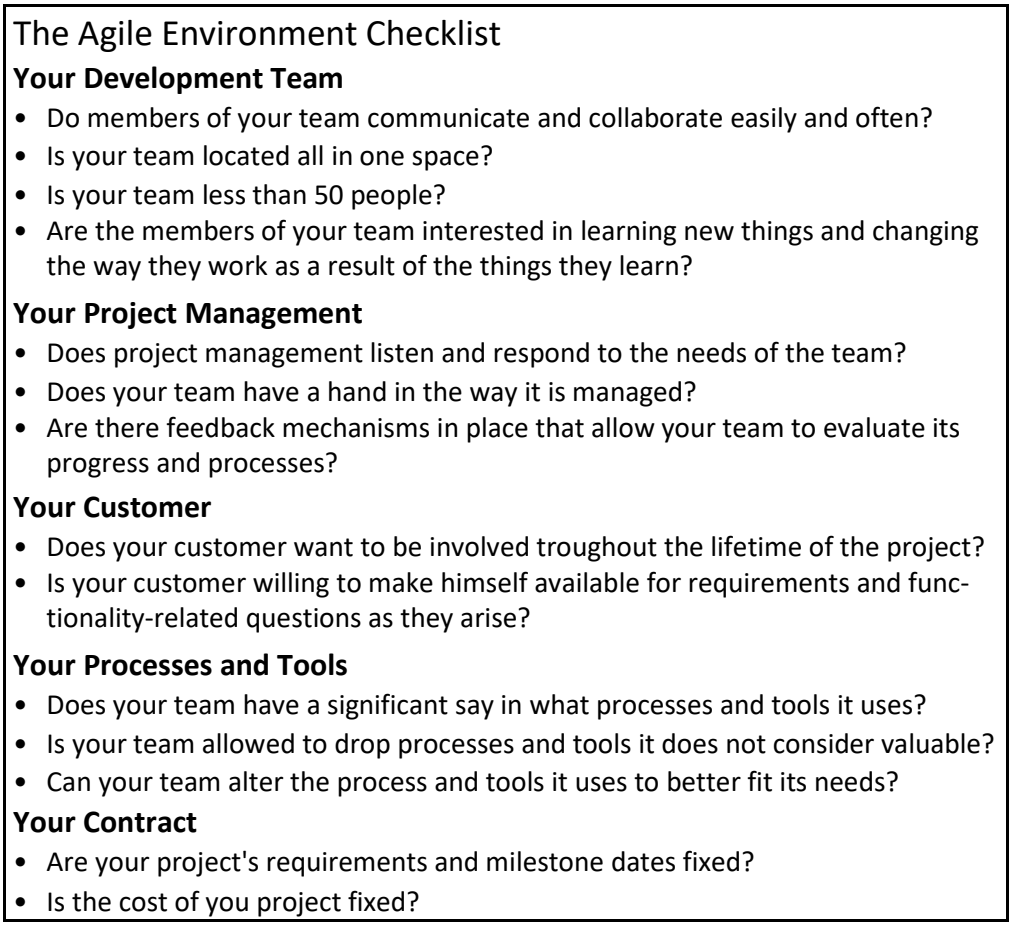
3.1. Interviews

The interviews consisted of two separate interview rounds. Questions in the first round focused primarily on the basic principles of agile development, working practices and methods, and the state of agile development in the Company at that time. The objective of the second round of interviews was to refine the results obtained from the first round and to figure out how to further apply agile development within the Company. Interviewees for the interview rounds were selected primarily from members of middle management in relevant working units by recommendation of the instructor of this thesis.

The actual questions for the first round of interviews were compiled with the help of an Agile Environment Checklist, seen in Figure 3, and applying that checklist to the specific situation of the Company. Questions for the second round of interviews were selected by examining the results from the first round of interviews and, after that, attempting to find solutions to the issues found during the first round in regards to applying agile development into the working methods of the Company. No actual scientific method for organising the interviews or selecting the interview questions was used – a fact that will be reflected upon later when examining the success of the study and particularly the interview section of it. The actual questions for both interview rounds can be found in appendixes A and B.

Both rounds of interviews were primarily conducted by phone call as most of the interviewees were located abroad, although a handful of interviews were able to be performed face-to-face in cases where the interviewee was located in Finland. All interview sessions were recorded for the purpose of writing an accurate transcription of the interviews. The

transcriptions were further analysed and scrutinised, after which a more refined analysis of the situation of agile development within the Company, as well as issues with and recommendations on applying agile development, were concluded. The conclusions from the interviews were later used in conjunction with findings from the data analysis in order to find correlating topics related to key issues in the software development processes of the Company.



The Agile Environment Checklist

Your Development Team

- Do members of your team communicate and collaborate easily and often?
- Is your team located all in one space?
- Is your team less than 50 people?
- Are the members of your team interested in learning new things and changing the way they work as a result of the things they learn?

Your Project Management

- Does project management listen and respond to the needs of the team?
- Does your team have a hand in the way it is managed?
- Are there feedback mechanisms in place that allow your team to evaluate its progress and processes?

Your Customer

- Does your customer want to be involved troughout the lifetime of the project?
- Is your customer willing to make himself available for requirements and functionality-related questions as they arise?

Your Processes and Tools

- Does your team have a significant say in what processes and tools it uses?
- Is your team allowed to drop processes and tools it does not consider valuable?
- Can your team alter the process and tools it uses to better fit its needs?

Your Contract

- Are your project's requirements and milestone dates fixed?
- Is the cost of you project fixed?

Figure 3: Agile Checklist for Project Environments [Sch04, p. 15]

3.2. Data Analysis

In addition to the qualitative-natured interview method, there was demand for a more quantitative approach for the thesis as well. A simple data analysis was desired to complement the interviews, however, the focus of such an approach was not immediately apparent. To help with this problem, out of several alternatives, Six Sigma was chosen as the more quantitative method. Six Sigma is a particularly suitable choice since, as a process improvement method, it is easily applicable with agile development concepts. Additionally, Six Sigma had been utilised within the Company before, hence it was already familiar to some of the personnel there.

As Six Sigma encompasses a substantial amount of different analysis methods, only two of those methods were actually used in this thesis. With the help of these methods, the focus and specifics of the data analysis could be determined much more comfortably. The

final analysis would simply consist of collecting raw data from relevant software development related systems and conducting a simple statistical analysis. Comparison between the quantitative data analysis and the qualitative interviews would also be performed later.

3.2.1. Six Sigma

The two Six Sigma methods utilized in this thesis are the Fishbone diagram and SIPOC. The Fishbone diagram is a simple method for testing causality in a given situation and also recognizing and curing causes that are problematic in that situation. Fishbone is useful for determining and visually presenting all of the reasons to a problematic situation in detail. The name Fishbone diagram derives from the end result of the method which is a diagram that visually resembles the backbone of a fish. [Bra02, p. 49]

The construction of the Fishbone diagram has several steps. First, causes for a specific problem or problematic situation are determined and these causes are then categorized into "major" causes that are affecting the situation. The determination of the causes can be done by brainstorming or by examining relevant data related to the situation. After that, the Fishbone diagram is created by placing a description of the problematic situation in a box on the right-hand side of the diagram and then situating the "major" causes along a line drawn from the description on the right. Finally, the causes that were previously brainstormed or otherwise determined can be placed under the "major" causes as per their categorization. Some causes can be placed under several "major" causes although ideally they should fit only under one category. [Bra02, p. 49-52]

The "major" causes can be chosen according to the situation at hand but there are also more standardized categories for certain types of situations. For example, typical categories in production processes would be machines, methods, materials, and people. After the categorization of the causes, the reasons for all of the causes can be determined by analysing why the cause happens or why it could happen. If there are similar answers to multiple different causes those answers can be interpreted as root causes. Other methods can also be used for finding root causes, e.g., a data analysis of the frequencies of the causes on the diagram. Figure 4 illustrates an example of a Fishbone diagram. [Bra02, p. 51-54]

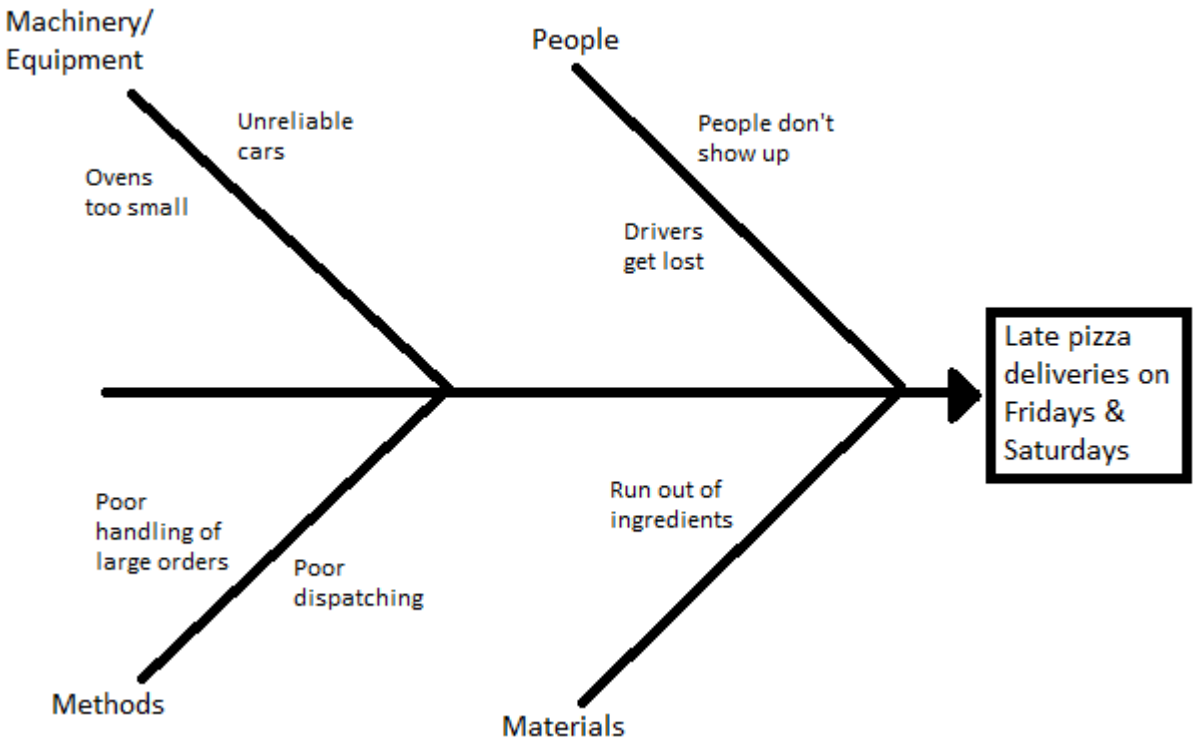


Figure 4: Example of a Fishbone Diagram [Bra02, p. 52]

SIPOC is a Six Sigma tool for examining the inputs and outputs of processes. Utilizing SIPOC in the order of the abbreviated terms, i.e., starting from suppliers and ending in customers, can be challenging, hence simple steps for the easy use of SIPOC have been determined. First, the starting and ending points of the process that is analysed have to be determined. Then the rest of the steps between the starting and ending points of the Process need to be defined, usually limiting the steps from five to seven. The process part of the SIPOC diagram should read like a simplified, linear, top-level flowchart. Next, the outputs from the process should be determined, after which customers for those outputs have to be identified. Similarly, the inputs for processes need to be established, and then suppliers for those inputs need to be determined. Inputs and outputs are often material goods especially in production industries but they can also be non-tangible, e.g., information or labour. It should be noted that both suppliers and processes can have multiple inputs and outputs respectively. Once the SIPOC diagram is finished it can be reviewed and modified accordingly. Figure 5 illustrates an example of a finished SIPOC diagram. [Bra02, p. 236-237]

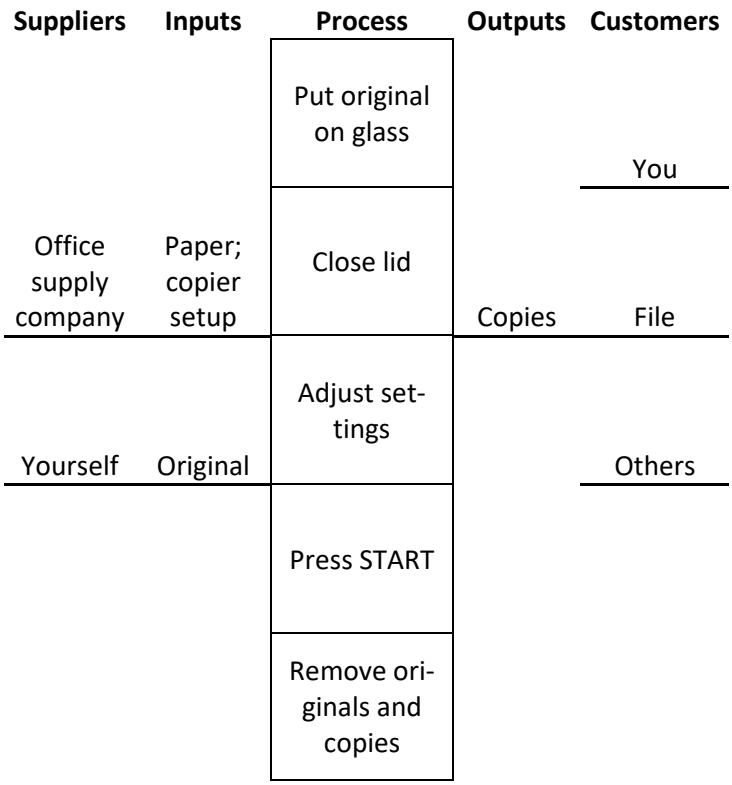


Figure 5: An example of a SIPOC diagram (Process flows downwards, the rest flow from left to right) [Bra02, p. 238]

As mentioned in previous sections, the focus of this thesis is purposefully narrow and focused because of the complexity of process development analysis in the context of the processes of the Company, and the same principle applies to the Six Sigma analysis part as well. In order for the Six Sigma analysis to be conducted properly, focus on a specific aspect of the work processes of the Company had to be chosen. Ultimately, the change management aspect of software development within the Company was chosen as the focus of the Six Sigma analysis. Reasons for selecting change management as the focus were plentiful. First, it is an essential part of software development and one that typically can cause a lot of problems especially in development work that is not yet agile. Second, change management, and change in the context of software in general, was a central and recurring topic during the interviews, as can be found in the interview section later. Third, change management was an aspect of the software development of the Company that had data easily available from it, and as Six Sigma usually requires quantitative analysis to complement it, it was extremely important to have that data available. Several other aspects of software development in the Company, e.g., the planning or testing phases of development projects, either had only small amounts of data available from them, or alternatively data from them was not easily retrievable. Since change management had this data available in relatively large amounts and from a single database, it was a convenient choice for the focus of the Six Sigma analysis.

Data for the analysis was chosen for collection from all systems related to certain telecommunications services that had had any significant amount of changes implemented in them within the past year. The data was collected by importing the data from a change management database system in spreadsheet form. This data, in its rawest form, consisted of all the changes implemented in each system, which there were ten, incidents that had occurred in those systems within the past year, and all relevant information related to those changes and incidents, such as identification numbers, priority, dates, and affected systems.

In order to effectively complete the data analysis, the raw data had to be refined. A simple spreadsheet tool was used to organize the data in such a way that further analysis was feasible. A simple statistical analysis was conducted with the help of internal statistics experts in the Company. Finally, a correlation analysis was performed in order to study correlation between systems, system age and system type, and the number of incidents occurring after changes in each system.

4. Analysis

4.1. Current Situation in the Company

Before proper analysis can be performed by using the study methods described in the previous section, it is vital to understand the specificities of the Company and its structure. In the following subsections, Company specific terminology, as well as different models and methodologies related to the operation of the Company, are explained in order to clarify the actual analysis procedure.

4.1.1. Definitions of Company Terminology

In the following sections, there will be several mentions of two entities within the Company called Development and Operations. These entities and their collaboration and communication with each other will be the focus of the analysis part of this thesis. Development, as the name suggests, is responsible of the development part of any piece of software that is being produced either within the Company or by external subcontractors, also described as vendors. The tasks of Operations consist primarily of support and maintenance of software which is provided to them by Development.

Two other entities within the Company that are often referred to in the analysis part are Business Support Systems (BSS) and Operational Support Systems (OSS). Again as the names suggest, BSS is responsible of the business operations of the Company, namely regarding customers of the Company, and OSS ensures that technical assets of the Company, e.g., its networks, are operational. Development and Operations exist within both BSS and OSS and are further divided into several working units or teams. Regarding, e.g., the interview part of the analysis, employees from both BSS and OSS, and Development and Operations, were interviewed. Additionally to the entities mentioned above, the business unit of the Company is a regularly mentioned entity during the interviews and is often simply referred to as business.

While not Company specific terms, certain concepts relevant to software development specifically need to be explained in order to understand the analysis part of this thesis properly. Change is a needed or specifically requested modification to software or to the functionality of software. Changes can occur because of planned development work or because of incidents or customer requests. Incident is an unexpected discontinuation of normal operation of a system or function. The occurrence of incidents often leads to changes in order to fix those incidents or alternatively to fix the root cause of incidents which are called problems. A delivery is a modification to existing software or the creation of new software, hence deliveries can be made to implement changes to software. A single delivery can consist of several smaller deliveries that are implemented simultaneously, and occasionally multiple simultaneous deliveries are called releases. A feature is a tangible functionality in software. A feature can include a single or several deliveries, hence the terms delivery and feature are sometimes used interchangeably in the context of this thesis.

Another term that is not Company specific but is nevertheless important to define is Information Technology Infrastructure Library (ITIL). ITIL is a set of standards for managing IT services and a framework of best practices for the production and delivery of quality IT services for customers [Ahm13, p. 553-554]. ITIL consists of five core publications – ITIL

Service Strategy, ITIL Service design, ITIL Service Transition, ITIL Service Operation, and ITIL Continual Service Improvement – that together form the ITIL Service Lifecycle [Axe16].

ITIL is commonly implemented because an organization wishes to focus on improving its customer service or to increase the effectiveness and transparency of its IT governance. Benefits of ITIL include cost savings, risk management, and IT operation streamlining, but it has some notable disadvantages as well. ITIL does not accurately specify how and which processes should be implemented or improved. While this allows ITIL to be applied into drastically different kinds of situations and organizations, implementation of ITIL requires a considerable amount of studying and training if the organization wants to benefit from applying it. The amount of effort required is also the reason why organizations often face resistance from their employees when trying to implement ITIL. Therefore, effective change management is often required when applying ITIL into work processes. [Ahm13, p. 554-555]

4.1.2. State of Agile Development in the Company

Currently, agile development is rarely utilised in the Company. In the case of project work for instance, most projects are still conducted using the more traditional waterfall method. There have been a handful of instances where agile development has been used, however. Certain projects or parts of projects have already been performed utilising agile development methods. Overall, the attitude towards agile development and applying it within the Company has been mostly positive and supportive, as is evident from the interviews. Additionally, there are plans to develop the Company's own proprietary project model by including agile development as an integral part of it.

4.1.3. Company Project Model

The Company has been trying to incorporate agile methodologies into its working methods even before the requisition of this thesis. In order to do this, the Company has created its own proprietary project model whose purpose is to integrate agile working methods into project work within the Company. The goal is to utilize the general idea of the iterative nature of agile development instead of any specific agile methodology, although the type of agile development presented in the project model clearly resembles Scrum the most. [CPM, p. 1]

In the Company Project Model, a project is divided into roughly seven stages. The project model by itself describes a traditional type of framework for development projects but the Company has already begun to revamp the project model into a more agile direction. The first stage in the project model is preparation for the project which includes a concept and documentation of the idea for a project. As per agile development guidelines, current and future stakeholders should be heavily informed and involved at this stage already even if the preparation stage might not eventually lead into an actual project. If the concept for the project is approved the pre-study stage will commence. During the pre-study, business benefits of the project are determined and clarified, and a plan to achieve those benefits is determined. A high-level backlog for the project is included in this plan. [CPM, p. 9-10]

After approving the pre-study, the planning stage can begin. Project plans can include items such as budget, business case, and risk analysis. In regular projects, these items might

be rigid and non-negotiable but with agile development they do not have to be detailed since the project plan can evolve and change during the lifetime of the project. After the project plan has been determined and approved, the design stage can start. At the beginning of this stage, the decision to use or not to use agile development during the rest of the project has to be made. The design stage defines the product backlog, after which the design is approved and the project flows naturally into the iteration stage. The design can still evolve in the iteration stage because of the iterative nature of agile development, but the main goal of the iteration stage is to begin producing features that are ready to be implemented and delivered to the end users. [CPM, p. 10-14]

When everything in the product backlog has been delivered, or alternatively when the budget for the project has run out, and all of the deliveries have been approved, the handover stage can be commenced. In the handover stage, the responsibility of the deliveries that were produced during the project is switched from the team that produced the deliveries to the team that will maintain the deliveries. The last stage is the closing stage where the project is closed and post evaluation of the project is performed. [CPM, p. 15]

Even though the project model obviously already contains some aspects of agile development, one of the goals of this thesis is to further increase the level of agility of the working methods in this project model. A particular focus will be put on the handover stage which both currently lacks agility and also has been a noticeably problematic stage for many employees of the Company, as is evident from the interview section below. The end result will not necessarily be a reworked version of the current project model, but instead the current project model will be used as a frame of reference for indicating which working methods might be in need of more agility. The project model described above is a generalized and summarized version of the actual project model of the Company because of confidentiality issues.

4.2. Findings from the Interviews

What follows are synopses of both of the interview rounds. These synopses consist of opinions and observations of the interviewees, after which conclusions of both interview rounds are presented at the end of each respective subsection. The exact interview questions used in both rounds of interviews can be found in appendixes A and B.

4.2.1. Initial Interviews

Level of Agility within the Company

As per the central theme in agile development, several interviewees agreed that by applying the incremental delivery method in software development, adapting to changes during projects and other development work becomes easier, thus ensuring that the effect of unexpected changes on project schedules can be diminished.

Almost all interviewees acknowledged that there is a discrepancy in the level of agility between Development and Operations. This discrepancy must be taken into serious consideration since it is not feasible to apply agile methodologies only in either Development or Operations. Instead, agile development needs to be applied equally in both Development

and Operations, and overall the two entities have to be more aligned with each other in terms of working methods in order to increase working efficiency.

Even though the size of working units and teams was deemed to be small enough in order for agile development to work, especially considering that some of these teams are often further divided into even smaller sub-units, there was a concern among the interviewees regarding the physical location of team members. Teams are often distributed between at least two different countries, usually more. In terms of agile development, this is problematic since agile thrives on the everyday interactions of team members which is primarily and most efficiently achieved by working in the same physical space. Because of this, arranging teams locally should be encouraged, although interviewees agreed that one exception to this could be specialist teams working on distributed systems. Team members with specific knowledge of a system can be located in another country if the system is also located in that country.

There was a consensus that collaboration both within team members and with management was deemed to be sufficient. In addition to more formal meetings, informal communication seemed to occur naturally between local team members, which again highlights the importance of locally situated team members.

Involvement in project planning varied greatly between interviewees, regardless of whether BSS or OSS was considered. One common problem was that Operations is often involved only in the later stages in deliveries even when Operations itself has a desire to be involved with Development regarding deliveries. Development, on the other hand, were concerned with the rigidness of current working methods and models, as well as the immaturity of business units and other higher entities of the organization towards agile development. Additionally, both Operations and Development were concerned of situations where external developers or vendors are used. Issues in these situations include the level of agility of external developers, contracts made with them, and communication with them.

Customer Collaboration

Some interviewees felt that feedback from customers – business units in the majority of cases – is highly dependent of the involvement of customers in projects. Since customer involvement varies greatly it should be encouraged in most cases, particularly if the customer has sufficient knowledge of the project. Especially in cases where the business unit is the customer, the presence of a business representative in projects was highly valued. Involvement of customers also often affects changes in projects and how they are reacted to. Currently, change requests from customers are needed for implementing major changes, although smaller changes can be made without a formal request.

Conclusions

When assessing the answers of the initial interviews, some fairly simple topics, upon which most interviewees agreed upon, surfaced. Most interviewees agreed that the incremental delivery method central to agile development would be beneficial if implemented. The incremental method for deliveries will make implementing changes much easier and more flexible and also less time consuming. It was also clear that all interviewees wanted Development and Operations to be aligned not only in terms of the level of agility but also mutual working methods and participation.

A persistent topic among all interviewees was the distribution of team members. While agile development highly encourages local teams that have constant face-to-face interaction, the reality is that in such a global market, in which the Company operates, having local teams as a high priority can easily become unprofitable. Therefore, instead of having localization of teams as a central topic in this thesis, the lack of it is compensated by a slight focus on tools that can reduce the disadvantage of having team members distributed globally.

It became clear that improving knowledge transfer is an obvious solution in order to improve the transparency between Development and Operations, especially considering the issues Operations had regarding its participation in earlier stages in development. Having sufficient knowledge transfer is vital overall in agile development, but in this case improving it is an efficient way of addressing a specific problem. Since collaboration within teams and with management was already deemed to be on a decent level, the foundation for solid knowledge transfer exists already. The main focus needs to be in improving knowledge transfer between different teams and entities.

Flexibility of projects and other work was a frequent subject among interviewees. However, in the context of this study, time is considered to be a critical and inflexible variable since there are usually so many stakeholders in projects that the effect of delays can potentially be massive. Budget is also usually strict by default, thus the only flexible aspect in projects is often quality. Since the actual quality of incremental deliveries in agile development is basically expected to be high, flexibility in terms of quality often means that the scope of incremental deliveries can vary, meaning that, e.g., the scope of a difficult delivery must be fairly narrow in order to ensure the quality of it. Nevertheless, even the flexibility of scope can vary depending on the type of project, which means that within the context of this particular Company, even agile projects might be rigid in all its aspects.

The problem with external developers can potentially be a major one, especially since vendors are utilized within the Company on a significant level. However, studying the collaboration of vendors and internal developers is such a significant undertaking that it falls off the scope of this thesis and is therefore a fitting topic for a separate study altogether.

Even though the attitude towards agile development and applying agile more in everyday working methods was generally met with positivity, it is clear that some level of encouragement and training in agile development is required if the Company wants to succeed in applying it. This is the case specifically with OSS as a whole but especially with Operations since they had the most reservations against agile development.

Most issues raised by the interviewees were related either directly or indirectly to the organizational split between Development and Operations. Some frequent concerns were the long term responsibility of deliveries, which usually falls to Operations without any support from Development, and also the disparity of the level of agility between units. Therefore the mutual alignment, or lack thereof, of the working methods of Development and Operations will be regarded as a central topic of this thesis.

Other less frequent but still major issues among interviewees were current working methods and models, organizational structure, trust and confidence towards agile and agile teams, and the general mindset of the support functions of the Company, especially business units. Working methods and models will be considered when suggesting how agile development could be applied but organizational structure and the mindset of business units, even if important, are such major topics that they would need much more comprehensive analysis and preferably a separate study, thus falling off the scope of this thesis.

4.2.2. Follow-up Interviews

After the more general approach to agile development and the state of the Company in the first round of interviews, questions for the second round were devised in terms of trying to find solutions to both known problems and problems found in the first round of interviews and also trying to find potential future issues in applying agile development. Questions were divided between Development and Operations in order to achieve more specific questions and answers. Additionally, questions were also divided into three major categories according to their theme, although certain questions and answers overlapped on some level.

Incremental Deliveries

Many interviewees agreed that in order to make incremental deliveries work efficiently, processes in the Development-Operations interface need to be aligned properly (for example, aligning Scrum and ITIL). This has to be recognized and supported by both top management as well as the customer – usually the business unit – by aligning their working methods according to agile development principles, e.g., by adopting the incremental approach to development and dividing their own larger tasks into smaller ones. Agile projects are not typically planned thoroughly in advance but are instead prone to potentially major changes and errors during the whole timeline of the project. Development personnel especially emphasized that this is something top management, business, and also Operations have to understand and embrace.

A common agreement was that agile development is best used in situations where the customer has a vague idea of the desired product but cannot describe it in great detail. The iterative nature of agile is suitable in this regard. For example, when developing entirely new systems, agile is an excellent option, even though the more traditional waterfall method can be sufficient as well depending on the characteristics of the project.

The majority of interviewees were concerned that the handover process can produce problems when incorporating agile since handover includes tasks, such as implementation, documentation, and testing, which can be demanding regardless of the size of individual deliveries. The amount of work that goes into handover tasks with a single small delivery is the same as it is with a single large delivery. Therefore, having a larger number of smaller deliveries can increase the amount of handover tasks significantly since every task of the handover process has to be conducted with each delivery regardless of the size of the delivery. One solution, that was suggested to solve this problem, is not to put every single delivery into production separately. Instead, aggregating several smaller deliveries into fewer larger releases reduces the overhead of multiple handover tasks significantly. Simultaneously, it is easier to implement actual functionality within a single release if the release is larger.

One desirable feature of agile development is that developing a system can be modular in the sense that a single delivery can include only a part of functionality of a larger system, and that part can function by itself without the implementation of the rest of the system. However, some interviewees were quick to point out that, considering the nature of the specific systems within the Company, this might not be possible in reality, and in many cases all the parts of a system within the Company may be required to be functional simultaneously.

Improving Quality

When applying agile development into a real life situation, some interviewees felt it is important to assess whether it actually has an effect on working methods in reality. People can often claim they are incorporating agile into their work while still adhering to non-agile work methods in reality. This is particularly evident in scheduling, meaning that deadlines for deliveries are not being kept even when they should be, as per agile requirements. Most interviewees agreed that this concern should be addressed throughout the organization since top management and business are often the parties responsible for overloading the Development department.

Many interviewees expressed their concern over resources when trying to apply agile development. Resources, or the lack there of more specifically, is a concern directly affecting the potential benefit of agile development. Regarding incremental deliveries particularly, the issue is typically time used in projects compared to the time used in general line work. Simply put, more frequent deliveries require more resources allocated to them which is a problem that business and top management need to solve.

Operations personnel felt that, after the handover process, Development should preferably retain responsibility of deliveries for a few weeks, or alternatively until the next delivery, and verify that each delivery is working properly. This would substantially benefit Operations in the sense that handover of the responsibility of deliveries would be allotted to a larger time interval during which Operations would be able to consult Development in case of potential issues with each delivery.

Handover is not a formal but instead a common process and should be transparent in the sense that Operations felt they need to have visibility into the Development pipeline in order to efficiently see what changes are made, why they are made, and how they are implemented. When new deliveries are produced, Development should be prepared in order to quickly fix bugs and other issues on short notice instead of leaving Operations to deal with incomplete deliveries on their own. Documentation related to deliveries should be sufficient, and Operations should not have to accept deliveries with missing or subpar documentation, even though documentation can often be a secondary concern because of high demand for fast implementation of deliveries. Operations especially emphasized that missing documents related to, e.g., functionality or testing can severely impede production.

Both Development and Operations agreed that Operations themselves should participate in the testing phase of deliveries, after which implementing a proper handover process becomes more convenient for both parties. During a proper handover process, Operations learns what deliveries Development is working on, what the business case of the deliveries are, and what is actually being implemented. The result of a proper handover process is that afterwards Operations should be capable of conducting incident and problem solving by themselves while still being able to consult Development if needed.

Currently, Operations feels that there is a discrepancy between the views of Operations and top management concerning the service Operations should be providing. There is a conflict between what is expected from Operations, resources that are provided to them, and the amount of work that is allocated to them. Similarly, views between Operations and Development are also divided on the subject, as Development seems to expect more contribution from Operations than what they are able to provide.

It was generally agreed that, in terms of quality assurance, Operations should be more active in the testing phase in order to be more knowledgeable of changes and fixes that are being implemented. However, this is often unfeasible because of limited resources. The situation is slightly better with older legacy systems compared to newer common systems since Operations have not yet familiarized themselves with the newer common systems, neither from a technical nor from a business perspective. Legacy systems are primarily only being maintained and thus are more familiar to Operations, whereas common systems are constantly being developed with new functionality.

Concerning legacy systems themselves, many interviewees agreed that applying agile development to the development processes of legacy systems would be possible depending on the actual systems but there are certain issues such as requiring rigorous testing of deliveries and, most notably, dependencies between other systems. Because of this, even a small change to a legacy system can require severe testing and changes in other related systems. Because there is less room for errors, rigorous control mechanisms are needed when implementing changes into live production. In general, legacy systems should not even be developed anymore as opposed to, e.g., newer common systems. Legacy systems are typically in a fairly stable state, development consists only of support and maintenance, and only necessary changes are made. Using agile development methods in developing legacy systems might not even be any more beneficial compared to, e.g., waterfall, especially if the development of legacy systems is strictly defined and planned, which is often the case.

Even though backlog prioritization is generally done by the product owner, many interviewees pointed out that several different aspects can affect prioritization, such as competition on the market, seasonal events, budget decisions, employee brainstorming, or user stories. The product owner communicates with stakeholders in order to present the reasoning behind the prioritization decisions and with developers in order to discuss the characteristics of the tasks that are being prioritized. There exists a balance between the complexity of developing new features and the monetary or some other benefit from developing those features. Occasionally, a feature does not provide much monetary benefit but it can be easily implemented, and sometimes a feature can be complex but the monetary benefits are significant enough to warrant the cost of development.

While agile requires good quality and testing with less emphasis on extensive documentation, ITIL requires constant updates on documentation and training on work routines. Interviewees generally agreed that agile development and ITIL need to be adapted to and aligned with each other in order for them to function efficiently together. Development needs to take ITIL processes into consideration in their work, and conversely Operations has to figure out how they can become more agile while still adhering to ITIL standards.

Developing smaller and more frequent deliveries requires more light weight processes than what ITIL currently provides. Some interviewees suggested that certain aspects of Agile can help in this regard, such as the use of modern tools, or incorporating more automation into work processes, especially into testing. Making certain disciplines of ITIL less rigid would be beneficial, not just in terms of agile development, but also in general. Since ITIL requires rigorous documentation and testing, among other things, the overhead of these processes increases significantly when deliveries are produced more frequently. Therefore, it is once again beneficial to consider aggregating small deliveries into larger releases, thus saving time and effort on excess processes and bureaucracy. Individual deliveries should have a distinct business benefit if they are to be put into production by themselves.

Some interviewees argued that if smaller and more frequent deliveries are utilized, many processes, such as handover, have to be performed more frequently as well and, consequently, those processes must be made lighter in order to mitigate the extra workload. ITIL processes should be streamlined and made more efficient in such a way that the workload required for those processes is decreased. For example, relevant documents for deliveries should be easily located, and documents should be able to be updated with data relating to deliveries instead of producing new documents from scratch during every delivery. Overall, most interviewees agreed that applying agile development with ITIL is possible, and the two could potentially complement each other if both of their characteristics are taken into consideration. Agile would benefit the planning of deliveries, keeping deadlines and scopes, and it would also help Operations to prepare for the workload caused by deliveries. On the other hand, in order for ITIL to work, items such as definitions, specifications, schedule, and documentation, among other things, need to be well-defined and planned.

Interviewees noted that, in reality, the quality of releases is typically not perfect, and deliveries can have some form of errors or bugs in them. It is the decision of the product owner and Development whether the delivery should be released with small errors in it or not. Critical errors have to be caught in testing and fixed before release but often there can be minor issues with releases that do not necessarily have to be fixed immediately. It is important though that Operations are informed and aware of such occasions.

Some interviewees mentioned that different customers, usually some part of the business unit within the company, have different expectations on the end result of development. For example, customer support and sales expect deliveries to be user friendly, good quality, and that they do not increase system complexity, whereas product and offering personnel are more concerned about the time-to-market and flexibility of deliveries. Cost of deliveries is naturally a concern throughout the company. Even though the expectations of business can occasionally be unreasonable, especially in terms of schedule and requirements for development, they above all else expect honesty and communication from the delivery organization. If issues – in terms of budget, schedule, functionality, or otherwise – during the development process occur, business expects those issues to be communicated accurately to them, which is why the relationship between business and the delivery organization has to be healthy and functional.

A few interviewees noted that business is often under pressure to get deliveries completed which means that they can easily dismiss quality concerns for deliveries. New features or functionality is requested constantly and quality is expected to be high but business is usually unaware of issues in development and assumes that quality comes free of charge. Furthermore, it is often Operations who takes the credit when possible issues on deliveries are fixed, all of which indicates that there is a clear lack of transparency in the development process. Simultaneously, Operations should be much more honest and strict towards business and Development in terms of what they are actually able to accomplish. For example, Operations sometimes allows Development to cut corners which obviously affects the quality of deliveries. Operations should be stricter in setting quality demands and deadlines, but they should also attempt to secure quality by participating in the development process with Development and business from the very beginning of development.

Regarding securing quality itself, interviewees from Operations expressed desire to have regular meetings with Development where details such as items being developed, deliveries approaching handover stage, and the involvement of Operations in development tasks would be discussed. Meetings of this nature should be increasingly encouraged since not all

system areas currently exercise them. In the worst cases, business has pushed deliveries so rigorously that Development has simply begun developing new features without informing Operations.

Collaboration

Participation of Operations in developing and testing would be beneficial for several aspects of development, such as handover and knowledge transfer. Operations can secure quality by trying to increase their influence on Development. As many interviewees mentioned before, Operations can attempt at being stricter towards Development; challenging them with differing viewpoints, or imposing stricter qualities on handover and testing. All of this requires sufficient knowledge on what is currently on the pipeline, thus participation on the development process becomes even more important. Still, primary responsibility of development prioritization falls to Development or business since currently Operations does not have sufficient collaboration with either of them.

Testing is a process in which Operations personnel felt they can have influence on quality since it is one of the most important aspects of handover. Deliveries will not go into production before testing has been approved. Therefore, participation in testing should be the top priority of Operations when it comes to securing quality. If participation in testing is not possible, ensuring that proper documentation about the testing process is created and provided to Operations should be a priority.

Good knowledge transfer between Development and Operations can be achieved through cooperation. Since a substantial portion of documentation is currently located in wikis, which are a good environment for organizing lots of technical information, most interviewees agreed that maintaining this kind of documentation in cooperation between Development and Operations is highly beneficial. For example, when handover for a delivery occurs, documentation can simply be updated into the wiki instead of sending documents back and forth between Development and Operations. Both parties can also see changes made to the documentation with the help of version control mechanisms. Additionally, wikis aggregate documentation into a single location where the latest version of the document, the change history of the document, and information on who has made changes to the document is easy to find. Wikis can provide access to additional files as attachments, and searching for information could be made easy by the use of a simple search function.

In addition to change history, version history for wiki documents would be beneficial as well. If external vendors are used in development they also should have access to relevant wikis. Some interviewees suggested the implementation of a discussion forum for tools and systems which would complement wikis effectively. It would allow discussion and presentation of ideas between users and developers, and consequently Development would be knowledgeable of issues and ideas for changes before actual formal change requests are made.

Interviewees from both parties felt that Operations should participate more in development, also in the early stages of development, which would be beneficial in terms of knowledge transfer as well. Instead of simply receiving documentation about deliveries, a more hands-on approach on development would induce implicit knowledge transfer. At the very least, Operations should prepare and participate in testing in order to familiarize themselves with existing functionality and have better prior knowledge of the delivery before it

is handed over to Operations. Participation in the early stages of development will also allow Operations to give their own views and comments on the development project, thus transferring knowledge from Operations to others as well.

Some interviewees commented that the utilization of documents can often be poor, hence the actual importance, priority, and quantity of documentation should be questioned. The majority of documentation is used only once, a small percentage is used more than twice, and documentation related to projects is typically used only within those projects. Therefore, it would be sensible to reassess the validity of excess documentation in development. In handover, documentation should preferably be provided slightly ahead of the deployment of deliveries even if the documentation is incomplete. Updates and corrections can be applied later.

Most interviewees agreed that a separate representative from Operations in the Development team during the development process could be a beneficial arrangement, although many also noted some concerns about the arrangement. A common consensus was that this kind of representative work should not take more than 10-15% of the time of the representative because he or she has to be able to concentrate on their own operational work. Another issue is the representative as a resource. Operations typically has fairly limited resources, and committing a resource to development tasks can hinder the effectiveness of Operations, especially if that commitment is long-term, e.g., in the case of a large project. Operations also lack the resources to participate in every development team individually.

Managing teams and projects is easier if all personnel are located in the same location, but if the personnel is located even relatively close to each other, e.g., in neighboring countries, then the issue becomes more about the tools used for communication. Most interviewees thought that cross-border working is not a major problem if personal relations between employees are in order and people understand each other and their cultural differences sufficiently. Issues often arise simply from working in a large company in general. Some geographical differences can exist even within countries however, hence good communication is crucial, both in a qualitative and in a technical sense. Some interviewees noted that communication becomes easier if personnel are distributed evenly between locations of countries instead of, e.g., having the majority of people in one country and a small subset in another. When communicating with different nationalities or cultures, attention should be put on how messages will be perceived and understood. This applies to both written and spoken communication.

A few interviewees argued that a critical mass of personnel working in the same location could potentially be beneficial. The issue is not synchronizing work between several different countries but rather the difficulty and complexity for managers trying to manage multiple different countries.

Many interviewees thought the internal processes of the company support agile poorly. Utilization of agile is not prevented but, because of the heavy bureaucracy within the company, large scale projects are favoured, whereas starting smaller and faster projects is often not viable or desirable. The issue of decision making is related to the high level of bureaucracy. Control over decisions has been taken away from lower level employees and given to people on higher hierarchy levels. Decisions have to be brought up the hierarchy ladder, which is problematic since people on higher hierarchy levels might not be knowledgeable of the work they are approving, decisions take longer to be approved, and employees feel they are not being trusted. Consequently, time-to-value of products becomes longer. A high level of hierarchy also typically prevents the implementation of major changes late in a

project timeline. Even if these kind of changes are risky it is usually more beneficial to accept late changes in a project instead of finishing a project with a suboptimal outcome and starting a new project to fix those outcomes, which is something that occasionally occurs within the company.

Some interviewees were adamant the division of Development and Operations into separate entities made in the past has been a mistake. According to them it would be beneficial to combine those entities again and make operational tasks an integral part of development.

Conclusions

Since the goal of the second round of interviews was to find solutions and potential problems in applying agile development, the answers and conclusions were fairly straightforward. A frequent theme during the interview round was the importance of aligning agile development, other processes, and Development and Operations. It was often noted that business and top management must provide support for applying agile development in order for it to work and, in fact, agile development needs to be implemented and enforced throughout the Company in order for it to be effective.

Even though smaller and more frequent deliveries were seen as a step in the right direction in terms of development, there were several concerns about them as well. The most common one was extra overhead and workload caused by the repetition of certain processes such as handover tasks that have to be conducted for each delivery. A popular solution for this issue was to aggregate deliveries into larger releases, which would also benefit development in producing concrete functionality for separate releases. In reality, achieving true modularity of functionality in system development might be difficult, and in many cases all system functions have to be operational simultaneously. Another disadvantage with this solution is that larger releases can mitigate the benefit of agile development since small and frequent releases are an essential part of it.

Handover and issues with it were some of the major topics in during the interview round. Operations had clear concerns about the responsibility of deliveries and wanted Development to retain responsibility of them longer in order to provide Operations with more support and consultation. Operations also highlighted the importance of transparency in the development process which would significantly alleviate the issues of the handover process for them. Common topics of agreement for both entities were proper documentation and participation of Operations during the whole development stage of a project and especially in testing. It should be noted, however, that participation of Operations should usually be less regular at the beginning of development and gradually increase over time. Also, as per agile development principles, rigorous documentation should be a secondary concern but, in reality, it is still an important part of handover, especially because of distributed teams.

Lack of resources was a common concern during the interviews. A unanimous agreement was that both more frequent deliveries and cross-team collaboration require more resources – usually in the form of either time, personnel, or tools – than what are currently available. The conundrum of resources is an issue for business and top management to solve, and at the very least it requires significant scrutiny in other separate studies that are focused on the issue specifically.

Lots of comments about the relationship between different entities within the Company surfaced during the interviews. Operations felt a clear discrepancy between the expectations of their work and Development and business. Ultimately, this is an issue that has to be

fixed on a higher level and is primarily a strategy issue. Additionally, Operations themselves felt that they have to be stricter and demanding towards Development in terms the workload and the quality of deliveries they receive from Development. Simultaneously, Operations should indicate their work capacity and towards business and Development such that they do not become overextended in terms of their resources. There was a common desire for more frequent joint meetings concerning general development issues. Also, the idea of having a representative from Operations be involved in the everyday work of Development was mostly met with positive remarks. This method would benefit Operations in securing quality and increasing influence over Development but the major issue is again the lack of resources, most notably time. A lack of personnel resources also means Operations are unable to participate in every individual Development team, thus some form of aggregation between Operations and Development interaction would be needed. Consequently, the level of participation has to be fairly low. Operations should participate in the everyday work of Development, even on a daily basis, but to assume this could be done in high detail with every development team is unrealistic because of resource limitations. Participation on a higher level would be beneficial as well but practical involvement in development is a higher priority.

During the interviews, it became clear that Operations are much more familiar with older legacy systems than newer common systems. While utilizing agile development with common systems would be feasible, there are some caveats with applying agile in the development of legacy systems. Most notably, because of the large amount of dependencies in legacy systems, even small changes to the systems can require rigorous testing and changes in other systems as well. When considering development of legacy systems, utilizing, e.g., the waterfall method might be more feasible.

ITIL was another major concern during interviews since some principles of agile development can contradict with those of ITIL, such as the focus on documentation and strict definitions of work routines. A common consensus was, however, that ITIL and agile development can coexist but they have to be aligned properly, and both Development and Operations have to take the qualities of ITIL and agile development into consideration. In general, ITIL can cause extra workload when utilized with smaller and more frequent deliveries but applying agile development principles might alleviate some of that workload. Therefore, making ITIL processes more efficient and less rigid would benefit both applying agile development and producing more frequent deliveries.

Knowledge transfer is a key aspect of agile development, and some practical solutions for improving it emerged during the interviews. By far the most popular suggestions was the use of wikis for storing information and documents. Wikis are already used in some extent within the Company, and maintaining wikis in cooperation between Development and Operations would be an efficient way to disseminate documentation. Wikis can be updated easily, change and version history are readily available, and they are a convenient single location for the latest information on development projects and tasks. If development projects utilize external vendors wikis can be used to improve collaboration with them as well. Discussion forums for all stakeholders of development – Development, Operations, business, and customers – could be an effective way to complement wikis in terms of knowledge transfer and communication. Early participation of stakeholders in development projects would improve implicit knowledge transfer of all stakeholders.

Even if knowledge transfer is vitally important, producing highly detailed documents should not be a priority since documents are rarely utilized more than once in their lifetime.

Focus should instead be on providing sufficient documentation in time, especially for use in handover.

Additionally, some less common and more minor topics surfaced during the interviews. Some remarks were made about the realities of quality assurance. As opposed to agile development principles, deliveries can often be imperfect and have bugs and errors. Critical errors obviously need to be found and fixed before release but minor ones might have to be left in depending on the state of resources. Product owner and Development are ultimately responsible for releasing deliveries with known errors.

Differences of customers were occasionally noted during the interviews, as different customers can varying expectations of deliveries. Business is a particularly important customer and partner for both Development and Operations, and a working relationship between Business and the delivery organization is vital. Relationships between the delivery organization and other supporting entities, most notably the business unit, is another issue that warrants further scrutiny and separate studies focusing specifically on the topic.

Some comments were mentioned of the location of teams and personnel. While local teams are always preferable, proper communication tools can effectively diminish the issues with distributed teams if teams are located at least relatively close to each other. Healthy relations between employees is usually much more important and also makes cross-border working easier. In order to maintain healthy relations, potential cultural differences have to be considered in everyday communication. The most difficult issue with distributed teams is managing multiple different global teams simultaneously.

Finally, there were some common concerns about the state of the company in terms the feasibility of agile development. Both Development and Operations generally felt that agile development is not properly supported in the Company. For example, it is often unfeasible to initiate smaller projects because of the high level of bureaucracy within the Company, and for the same reason large scale projects are usually highly favoured. Also, because of a high level of hierarchy decision making power is rarely delegated to lower levels, which in turn prevents actions typical to agile development, such as implementing changes in later stages of projects, to be performed. One drastic solution that was suggested would be to overturn the past decision of dividing the delivery organization into Development and Operations and fuse them together as they once were.

4.3. Six Sigma

Six Sigma analysis represents the more quantitative portion of the analysis part of the thesis. As mentioned in the previous chapter, the two Six Sigma methods used in this thesis are the Fishbone diagram and SIPOC. Both are utilized in the context of change management of the software development of the Company.

4.3.1. Fishbone

The Fishbone diagram, seen in Figure 6, was constructed by choosing the focus of the Six Sigma analysis, i.e., change management, as the problem and selecting standardized production process categories as the "major" causes of the problem, as explained in the Six Sigma section of the previous chapter. The causes under the "major" causes were then selected by brainstorming and categorized accordingly. Finally, causes that were deemed relevant to the analysis were marked as controllable or non-controllable.

During the Fishbone analysis, all the causes that were marked as either controllable or non-controllable were regarded as important and desirable for further analysis. Those causes that were marked as non-controllable were considered either not feasibly analysable or beyond the scope of the study. Later, it would be discovered that neither full-time equivalent (FTE), i.e., personnel resources, nor handover documentation were feasible for analysis because of lack of data. Controllable causes that were left, i.e., system name, number of changes, and number of incidents, would become the key components of the data analysis presented in a later section.

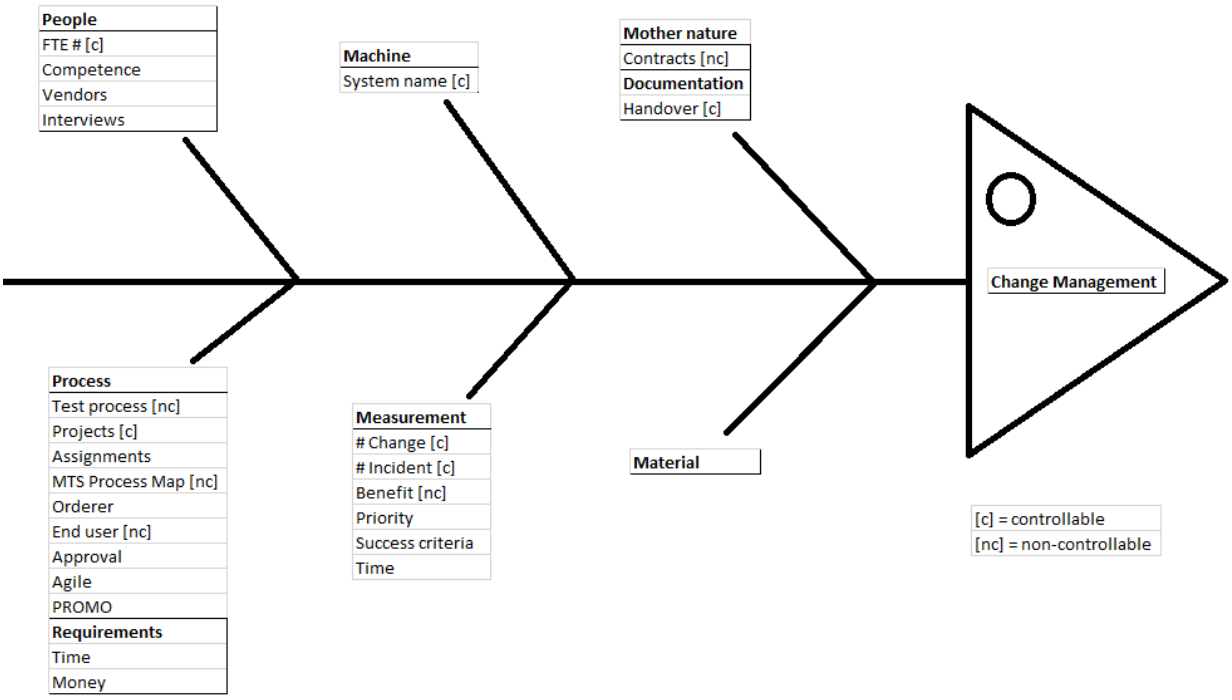


Figure 6: Fishbone diagram of Change Management

4.3.2. SIPOC

The SIPOC analysis was conducted in order to further narrow the focus of the overall analysis of the thesis within the change management process. As can be seen from figure 7, three high-level processes within change management were identified: demand, implementation, and handover. Inputs, outputs, and stakeholders of the processes were identified as per the SIPOC method, described in the previous chapter. The abbreviated stakeholders in the analysis are Product Development (PD) and Systems Operations (SO).

While the SIPOC analysis shows that all three processes have attributes that are important in the context of agile development, the handover process clearly includes attributes that are most relevant in terms of agile development. In addition to stakeholders, the handover process incorporates several different types of documents, and both the actual change and the final release. Therefore, handover was an obvious choice for increased focus in the thesis.

While the SIPOC analysis primarily supplements the qualitative part of the overall analysis section by indicating handover as a lucrative process to focus on, it also provided further insight into choosing the attributes for the more quantitative data analysis. Changes (or the number of changes in the context of the data analysis) and documentation can easily be identified as points of interest for the data analysis, but unfortunately, as mentioned in the Fishbone analysis, analysing documentation would not be feasible because of lack of data.

S	I	P	O	C
Suppliers	**Input**	**Process**	**Output**	**Customers**
Providers of the process	Inputs into the process	Top-level process desc-ription	Outputs of the process	Receivers of the process outputs
User	Change request (RFC), require-ments, priority, etc.	Demand (Evalu-ation, approval, prioritization)	Prioritized RFC	SO
Customer				
Business				
Tester				
SO	Prioritized RFC	Implementation (documentation, testing, etc.)	Change in code	SO
			Release notes	
			Change docu-mentation	
			Test documenta-tion	
PD	Change	Handover	New version	Business
	Release notes		Handover do-cumentation	User
	Documentation			Customer

Figure 7: SIPOC diagram of the Change Management Process

4.3.3. Statistical Data Analysis

In order to further focus on actual project and system development within the Company, a simple statistical analysis was required. Data for the analysis was gathered from a single change management database system that provided data of several different systems developed under the development organization. The raw data itself consisted of simple information such as the number of changes and incidents in each systems, priority, schedule, and associated systems of each change.

First, all relevant data (system name, type, and age, number of changes and incidents) was compiled into a single table, after which the data was further used to calculate the proportions of incidents and changes per system, as well as the reoccurrence of changes and incidents and the average delay of those reoccurences (see tables 2 and 3). Table 2 also

displays the correlation value for the number of changes and number of closed changes compared to the number of incidents, as well as the mean for both types of changes. By utilizing this data, a graph displaying the number of changes and incidents, as well as the number of incidents occurring after changes, was created. The graph can be seen in figure 8, in which the circumference of each data point represents the proportional amount of incidents that have occurred after changes in each system. All systems are colour-coded according to their age. Since the ending time for each change was not always available in the raw data, it was decided that only closed changes were taken into account in producing the final graph.

Table 2: Change Management Attributes

Change management attributes							
System	System type	System age	# of Changes	# fo Incidents	# of closed Changes	Incidents / Closed Changes	Incidents / Changes
Fokus	Packet	> 10y	1395	7	94	0,07	0,01
NoBill	Packet	6-10y	285	5	119	0,04	0,02
NEO	Custom	1-5y	197	38	93	0,41	0,19
Tango	Custom	< 1y	32	183	11	16,64	5,72
Webshop	Packet	1-5y	29	63	3	21,00	2,17
JOICE	Custom	6-10y	24	2	21	0,10	0,08
Copa	Custom	< 1y	17	130	9	14,44	7,65
Self Service	Custom	1-5y	16	3	14	0,21	0,19
TWAT	Custom	6-10y	8	20	7	2,86	2,50
DEMSY	Custom	1-5y	1	13	1	13,00	13,00
		Mean	200	46	37	1,25	29,82
		Correlation	-0,2658013		-0,3468117		

Table 3: Number and Average Delay of next events (change or incident) after changes and incidents

Number of ”next” events					**Avg delays of "next" events**				
Change -> Change	Change -> Incident	Incident -> Incident	Incident -> Change	All	Change -> Change	Change -> Incident	Incident -> Incident	Incident -> Change	All (Avg)
88	5	2	5	100	3,8	5,2	0,5	5,0	3,9
113	5	0	5	123	3,2	3,9		4,8	3,3
67	25	13	25	130	3,7	2,6	1,8	2,9	3,2
5	6	177	5	193	2,4	6,8	1,9	2,6	2,1
1	2	60	2	65	19,2	40,5	4,2	26,5	6,2
19	2	0	1	22	5,9	33,6		19,5	17,4
4	5	125	4	138	28,1	13,5	1,4	4,9	2,7
11	3	0	2	16	13,9	43,5		37,3	22,3
4	3	17	2	26	12,1	66,4	3,1	49,0	15,3
0	1	11	1	13		160,2	6,9	124,9	27,8

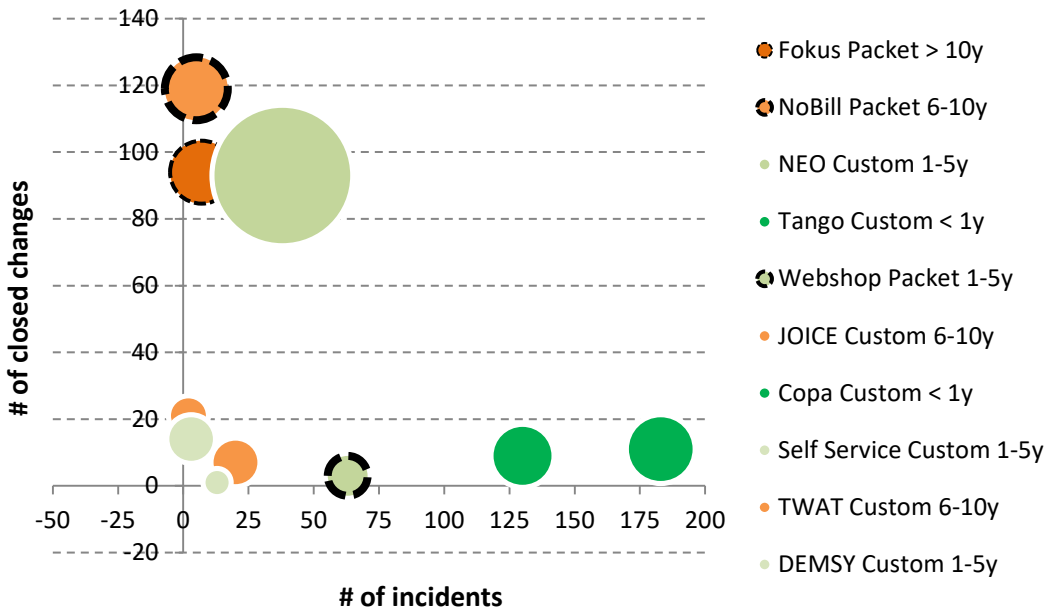


Figure 8: Graph of Changes and Incidents within Systems

The graph in figure 8 provides a good visual representation of the proportions of changes, incidents, and subsequent changes and incidents per system, but a more numerical approach was still desired. In order to provide this, several simple correlation analyses were performed by utilizing the refined data. The analyses were categorized by system age and type, and the goal was to find any correlation between the number of changes and incidents in systems of certain age or type. Packet type systems are systems that have been provided for the Company as a complete packet at delivery, whereas custom systems are systems that have been gradually developed within the Company from the start. Finally, a similar analysis studying the correlation between the priority of changes – where Priority 1 is the most critical and Priority 4 is the least critical change – and occurred incidents was performed in the context of all systems. Tables 4 to 10 illustrate these correlation analyses.

Table 4: Correlation Analysis for Systems Younger than Five Years

< 5y Systems	# of Changes (closed)	# of incidents
NEO	93	38
Tango	11	183
Webshop	3	63
Copa	9	130
Self Service	14	3
DEMSY	1	13

	# of Changes (closed)	*# of incidents*
# of Changes (closed	1	
# of incidents	-0,194383961	1

Table 5: Correlation Analysis for Systems between Ages of One and Five Years

1-5y Systems	# of Changes (closed)	# of incidents
NEO	93	38
Webshop	3	63
Self Service	14	3
DEMSY	1	13

	# of Changes (closed))	*# of incidents*
# of Changes (closed	1	
# of incidents	0,150116214	1

Table 6: Correlation Analysis for Systems Younger than One Year

< 1y Systems	# of Changes (closed)	# of incidents
Tango	11	183
Copa	9	130

	# of Changes (closed)	*# of incidents*
# of Changes (closed	1	
# of incidents	1	1

Table 7: Correlation Analysis for Systems Older than Five Years

5y < Systems	# of Changes (closed)	# of incidents
Fokus	94	7
NoBill	119	5
JOICE	21	2
TWAT	7	20

	# of Changes (closed)	*# of incidents*
# of Changes (closed	1	
# of incidents	-0,471266028	1

Table 8: Correlation Analysis for Packet Systems

Packet systems	# of Changes (closed)	# of incidents
Fokus	94	7
NoBill	119	5
Webshop	3	63

	# of Changes (closed)	*# of incidents*
# of Changes (closed	1	
# of incidents	-0,984581014	1

Table 9: Correlation Analysis for Custom Systems

Custom systems	# of Changes (closed)	# of incidents
NEO	93	38
Tango	11	183
JOICE	21	2
Copa	9	130
Self Service	14	3
TWAT	7	20
DEMSY	1	13

	# of Changes (closed)	*# of incidents*
# of Changes (closed	1	
# of incidents	-0,125531743	1

Table 10: Correlation Analysis between Change Priority and Number of Incidents

Priority of Changes	Priority 1	Priority 2	Priority 3	Priority 4	# of incidents
Copa	5	2	2	0	130
DEMSY	0	0	1	0	13
Fokus	12	66	16	0	7
JOICE	1	12	8	0	2
NEO	19	47	26	1	38
NoBill	24	45	50	0	5
Self Service	11	2	1	0	3
Tango	9	2	0	0	183
TWAT	1	1	4	1	20
Webshop	0	1	2	0	63

	Priority 1	*Priority 2*	*Priority 3*	*Priority 4*	*# of incidents*
Priority 1	1				
Priority 2	0,7255	1			
Priority 3	0,837631	0,745616	1		
Priority 4	0,112712	0,131238	0,131533	1	
# of incidents	-0,11163	-0,36154	-0,36233	-0,14721	1

4.4 Analysis Results

4.4.1 Data Analysis

As can be seen from the correlation analyses above, the general results from the data analysis are largely inconclusive. Most of the analyses showed no correlation, or in some cases even heavy negative correlation. The most conclusive result is gained from examining the changes and incidents of under one year old systems but the problem there is the small sample size and especially the fact that there are only two systems to examine. Another fascinating result would be the examination of packet systems if the Webshop system was not such a clear outlier. Because of this outlier the examination of packet systems leads to high negative correlation.

Regarding the rest of the analyses, on first glance it seems that the age of a system does not heavily affect the relationship between changes and incidents. On the other hand, examining the effect of the type of system – packet or custom – in terms of changes and incidents resulted in either strong negative correlation in the case of packet systems or not much correlation at all in the case of custom systems, the former occurring because of the clear outlier of the Webshop system. Finally, the level of priority of changes shows little correlation as well when comparing the different priorities to the number of incidents.

If certain outliers can be eliminated the correlation analyses can provide some interesting observations. First, the two youngest systems experienced a large amount of incidents

with only few changes into them, hence it seems that incidents occur more often in more immature systems. Second, if the outlier of the TWAT system is eliminated it appears that more mature systems are more stable and experience fewer incidents. Third, packet systems are potentially less prone to incidents if the outlier of the Webshop system can be ignored.

One issue with the correlation analysis is that it does not account for incidents that occur specifically as a result of changes. The graph presented in figure 8 can alleviate this issue since it represents the proportional amount of incidents that have happened after changes as the circumference of each data point which in turn represent each individual system. As the graph shows, systems that have a low number of both changes and incidents do not experience a significant amount of incidents after changes, whereas in systems with a high number of either incidents or changes, the number of incidents following changes is higher. It is interesting to note that in the two youngest systems, where the number of incidents compared to changes is high, the occurrence of incidents after changes is also relatively high, but the same is true for some of the more mature systems even if the number of actual incidents is low. One hypothesis from this could be that because more mature legacy systems have more dependencies with other systems, changes in legacy systems can easily cause incidents even with a low number of incidents overall. This would explain the anomaly that is the NEO system since, according to many employees, this particular system is central to several different functions and has a large number of dependencies with other systems.

The combined conclusion from examining the graph and the correlation analyses can be summed up in three concise points. First, the primary conclusion is that younger systems seem to experience more incidents overall and a fairly large amount of incidents following changes to them. Second, incidents often occur after implementing changes into more mature legacy systems with large amounts of dependencies with other systems even if the overall amount of incidents in these systems is lower. And third, packet type systems can potentially be more stable in terms of amount of incidents compared to custom, internally developed systems. The first two points specifically are worthy of further scrutiny since some interview results heavily correlate with these two notions, as can be seen in the next section.

4.4.2 Data Analysis and Interview Conclusions

Since one of the primary conclusions from the data analysis was that changes to younger systems induce a relatively high number of incidents, the fact that several interviewees agreed that they were more familiar with the older legacy systems rather than newer systems can be considered as one of the main conclusions of the interviews. Similarly, since interviewees stated that they are more familiar with older systems, this might be one reason why older systems experience fewer incidents when making changes to them according to the data analysis. Both of these observations correlate within the context of both the interview and the data analysis conclusions. Therefore, the susceptibility of younger systems to incidents occurring in the development of those systems is a major conclusion from the overall analysis section.

Another aspect that correlated in both the interviews and the data analysis was the concern of dependencies in older legacy systems. The data analysis shows that lots of incidents occur directly from changes to legacy systems, potentially because of the high level of dependencies in those systems. Similarly, the exact same concern was highlighted during the interviews by several interviewees. Therefore, another clear conclusion from the combina-

tion of these analyses is that agile development – in which changes are implemented frequently – should be utilized with caution in the context of older legacy systems, or alternatively some other software development method might have to be considered altogether.

While the data analysis results were fairly limited, conclusions from the interviews provide some complementary insight into the work processes of the Company. An important notion was that agile development and process improvements related to it need to be implemented throughout the organization – not just the development departments – in order to provide the best support for software development organization-wide. Also, the potential issues with more frequent changes and deliveries, especially in terms of handover, need to be considered in implementing agile development. This is specifically pertinent because of the vast amount of incidents occurring after changes in certain systems.

The Company specific issue of responsibility over deliveries needs to be solved. Operations has indicated a desire for more responsibility and support from Development over deliveries in the handover process. This issue is directly related to the collaboration and participation issues of both entities, which in turn is related to knowledge transfer issues in development. One commonly agreed solution to these issues was an Operations representative in the Development team but some issues with this approach were seen as well, most notably limited resources. Wikis and discussion forums were suggested as solutions for knowledge transfer issues specifically, although it should be noted that rigorous documentation most likely is not an efficient form of knowledge transfer. Solutions to all of these issues will help in either reducing the amount of incidents or solving incidents that have already occurred.

Finally, having local development teams was highly desired by interviewees. Failing this, communication tools for cross-border development work has to be provided, and global teams and team members have to be distributed in a way that does not impede everyday work processes. Additionally, giving decision power to lower hierarchy level employees needs to be encourages. All of these aspects can have an effect on both the occurrence and ability to react to incidents.

4.5 Personal Experiences Working for the Company

This subchapter attempts at recording the experiences of the author of this thesis working for the Company during the study phase of the thesis, in the context of agile development, software development, and working in general, in order to provide some empirical insight on the state of and problems found in the Company.

First of all, there seemed to be a lack of direction and control in everyday working in the Company, especially in terms of the study phase of this thesis. Regarding agile development, this might even be a positive aspect but in the context of the study it made determining the direction and scope of the study challenging. It is unclear whether other employees at the Company experienced the same lack of direction as reporting and monitoring methods used within the Company were not studied.

A clear oversight in the Company was the occasional absence of necessary tools, particularly in the beginning of either employment or some specific work related task. These tools were often related to access to specific systems or services but in some occasions there was a lack of actual physical tools needed for general work tasks as well. This is a

massively important issue to be fixed as working in general, let alone working efficiently, is immediately hindered when accessing proper tools is inhibited.

An equally important issue was confusion in terms of where to find relevant information related to work tasks. For example, finding relevant information from systems that were under development within the Company proved difficult even with the help of the instructor of this thesis. This is a topic where the importance of proper knowledge transfer and information distribution is easily observable and is critical in the development of complex systems and services.

Some employees of the Company seemed discontent with the current situation of the Company in terms of certain aspects of development work. The distinct division of development teams, such as the division of development into Development and Operations, seemed to cause some employees problems in terms of their ability to work efficiently. Distribution of employees between different countries also affected the everyday tasks of employees negatively. Additionally, certain employees were so busy with their line work that they sometimes did not have time to effectively focus on subsidiary or complementary tasks. In some cases, certain employees were so busy that they were not even able to respond to interview requests. Employee satisfaction and happiness and providing enough resources for employees seem issues that need to be resolved at least on some level.

4.6 Recommendations

The conclusions from the combination of the two analyses show that particular attention needs to be given to the development of newer systems, often called common systems within the Company. Several aspects of agile development can potentially be beneficial in improving development of newer systems. It should be noted, however, that utilization of agile development should be reserved primarily to newer systems since it has become clear that agile does not scale well with the development of older legacy systems. Because of the eccentricities of developing legacy systems, such as a high level of dependencies to other systems, agile development makes developing these systems cumbersome, hence other development methods, such as the more traditional waterfall method, should be preferred instead.

Implementing the incremental deliveries aspect of agile development will benefit development of newer systems in several different ways. It immediately improves the ability to react to changes during the development process, even in later stages. Simultaneously, collaboration with the customer improves automatically since the customer can observe the results of development more frequently. It is crucial to establish a feedback loop between the development team and customer in order to obtain the most benefits possible from implementing the incremental delivery method.

Even though the incremental delivery method is vital in improving development of new systems, some limitations to the method need to be introduced. First, because of the issues with the method in the handover process deliveries should not be implemented as frequently as suggested by standard agile development methods if frequent deliveries increase the overhead of the process tasks. Instead, deliveries should be aggregated into larger releases which reduces the overhead caused by individual deliveries. This is especially pertinent since the Company has had frequent scheduling issues in the past with software development. Second, deliveries should not be required to always include functionality. Although

this is a key concept of agile development, in reality it is unfeasible to force the development team to provide the customer with some concrete feature every time a delivery is made. However, aggregating deliveries into larger releases alleviates this problem as well since larger releases have a higher chance of including functionality. Third, the requirement for flawlessly functioning releases should be relieved. Software development utilizing agile development – or any other kind of method – should always aim for the highest quality software possible but the reality is that deliveries will always contain some amount of flaws and bugs on release. Instead of assuming the development department is able to produce flawless deliveries, other aspects of agile development need to be supported organization-wide in order for developers to be able to respond more efficiently to the inevitable incidents that occur from imperfect deliveries.

Aside from improving the development of newer systems in particular, the general working methods of the Company require improving as well. The incremental delivery method utilized in the development of newer systems can be generalized and applied to the working methods of employees, however, the same restrictions of delivery aggregation, functionality, and quality still apply. In addition to this, the importance of establishing local development teams became more apparent than initially anticipated during the analysis phase. Judging by the interview results, the dysfunctionality of the development department of the Company, and the emphasis agile development puts on local teams, it is paramount that the Company begins to establish more locally situated development organizations, even at the cost of some resources, as local teams bring a vast amount of benefits to software development. Having local teams instantly improves the collaboration issues Development and Operations are experiencing. Operations will be able to participate more easily in development work from the beginning of development projects, and on the other hand Development is able to take responsibility from deliveries for longer periods of time and also provide assistance to Operations during and after the handover process. Furthermore, by utilizing local teams, Operations are able to assign a representative to Development in order to further improve collaboration between them, something that was suggested as a more concrete solution for the collaboration issue. Local teams would allow the representative to work with Development more efficiently and with fewer resources.

Knowledge transfer, particularly between Development and Operations, needs to be improved drastically. Development teams must continue to update already established wikis and other databases with relevant information about systems and development tasks, after which aggregation of such databases needs to be performed in order to store the information in such a way that it is easily found when needed. Instead of producing overly rigorous documentation that rarely is utilized more than once, emphasis should be put on easy and fast dissemination of information that is concise and useful for development work. Establishing a forum for all stakeholders of development projects – developers, business, and customers – in addition to technical wikis will improve the dissemination of more general information about development among stakeholders.

Finally, improvements to working methods should also be considered in the context of the Company Project Model. The model is a decent platform upon which the level of agility of software development can be improved further. Aspects in the model that are determined in detail, such as planning and devising the backlog of the project, have already been designed to incorporate agile development principles. Other aspects, particularly the handover stage and, related to that, responsibility of deliveries, can be improved by utilizing the rec-

ommendations discussed in this section. The model can be further updated by adding the aspects of collaboration and knowledge transfer, among others.

5. Conclusions

Ultimately, this thesis provides the Company the means to continue its chosen direction towards making its software development processes more flexible by incorporating general agile development methods. In a more general sense, the recommendations of this thesis can be utilized as a high-level examination and guide on how to apply agile development into a specific situation where issues arise in the interface of software developer and maintenance teams and personnel in the context of software deliveries. As such, the results of this thesis are proprietary in nature and specifically applicable in the situation the Company currently is.

Since the examined situation of the Company consisted of two distinctly divided development entities, Development and Operations, the results of this thesis should only be applied to similarly distinct situations. Additionally, even though the division of the development organization into two distinct entities was criticized during the study, a solution for their issues must not be the fusion of these entities back into a single entity, as it has to be assumed that there is a valid reason for the division in the first place. Such a drastic solution must be studied further before it can be considered.

One aspect of the recommendations presented in this thesis that was not addressed properly is the issue of resources. Even though the initial goal and hypothesis was to make software development processes more efficient, implementing agile development into non-agile processes initially requires additional resources in the form of either time, employees, finances, or a combination of the three. In general, implementing agile development requires support from the whole organization, not just the development departments. If proper resources are not provided for implementation, benefits from agile development will be significantly diminished. Ultimately, it is the decision of top management of the organization whether they want to commit to agile development and benefit from utilizing it.

On reflection, there are aspects in this thesis that should have been improved. First, the primary study method utilized during the thesis, i.e., the interviews, lacked a specific scientific method. The initial interview questions were devised by utilizing a single agile development source book, and the follow-up questions were conceived organically by referencing the results from the first interview round. The interview section would have benefited from a more standardized and scientific method of creating the interview questions.

Second, the Six Sigma analysis was overly simplified. Selecting only two Six Sigma methods seems satisfactory considering the scope of the thesis but both of them lacked specificity. In the Fishbone method, the selection of standard production process categories as the “major” causes inhibited the discovery of enough useful causes for the problem. Instead, a more specified set of “major” causes would have benefited the results of the method. In the SIPOC method, the process column consisted of too few process steps which lead to each process step having too many sub-processes within them. This method would have benefited from dividing the process steps into several smaller ones, thus specifying the flow of each individual process.

Third, the data analysis should have been more comprehensive. Because of the difficulties of finding sufficient data from different systems in development within the Company, there were not enough data points to conduct a decisive data analysis, which is evident from the inconclusive correlation analyses. In addition to that, the data analysis assumes that incidents follow changes, whereas potentially the situation with incidents and changes

could have been contrary as well. Also, as is evident from the conclusions from the data analysis, certain data and anomalies had to be dismissed in order to produce results which in general is not a particularly scientific approach to this type of analysis.

Some inconclusive aspects of this thesis require further studying in order to thoroughly examine the effects and requirements of applying agile development into software development processes. Most importantly, resource management and requirements for utilizing agile development is a crucial subject for a separate study. The variation in resource requirements when applying agile development is potentially so significant that its effects have to be studied before actual implementation of agile development is performed. Another important topic is relations between developers and other stakeholders in a development project. Relations between developers and customers is discussed on some level in this thesis already but relations between developers, business and top management in particular is a subject that can potentially benefit agile development utilization immensely if studied separately. Similarly, utilization of external vendors in the context of software development is also a topic that requires an independent study in order to solve collaboration issues between vendors and other development departments.

References

[Bec01] Beck, K. et al. Manifesto for Agile Software Development. Online Document. Updated 2001. Cited 28.2.2015. Available: http://www.agilemanifesto.org/

[Bec01b] Beck, K. et al. Principles behind the Agile Manifesto. Online Document. Updated 2001. Cited 28.2.2015 Available: http://www.agilemanifesto.org/principles.html

[Agi15] Agile Alliance. What is Agile Software Development? Online Document. Updated 2015. Cited 3.3.2015. Available http://www.agilealliance.org/the-alliance/what-is-agile/

[Agi15b] Agile Alliance. Manifesto for Agile Software Development. Online Document. Updated 2015. Cited 3.3.2015 Available: http://www.agilealliance.org/the-alliance/the-agile-manifesto/

[Ahm13] Norita Ahmad, Noha Tarek Amer, Faten Qutaifan, and Azza Alhilali. Technology adoption model and a road map to successful implementation of ITIL. *Journal of Enterprise Information Management*, 2013, Vol. 26, Issue 5, p. 553 - 576

[And12] David J. Anderson. Lean Software Development. Online Document. Updated November 2012. Cited 6.8.2016. Available: https://msdn.microsoft.com/en-us/library/hh533841(v=vs.120).aspx

[Axe16] Axelos Globas Best Practices. What is ITIL Best Practice? Online Document. Cited 2.10.2016. Available: https://www.axelos.com/best-practice-solutions/itil/what-is-itil

[Ber16] Bernardo, N. *Lean and Digitize: An Integrated Approach to Process Improvement*. New York, Rouledge, 2016.

[BPC07] Best Price Computers. Application Development. Online Document. Updated 13.8.2007. Cited: 10.7.2016. Available: http://www.bestpricecomputers.co.uk/glossary/application-development.htm

[Bra02] Brassard, M., Finn, L., Ginn, D., Ritter, D. *The Six Sigma Memory Jogger II.* First edition. Salem, GOAL/QPC, 2002.

[CPM] Company Project Model. *Instruction for Agile*. Internal Document, not available publicly. Updated 20.12.2011. Cited 2.7.2016.

[Dhi14] Dhirendra, K. *Six Sigma Best Practices: A Guide to Business Process Excellence for Diverse Industries.* Fort Lauderdale, J. Ross Publishing, 2006.

[Lar03] Larman, C. *Agile and Iterative Development: A Manager's Guide*. Boston, Addison-Wesley, 2003.

[Lea16] Lean Enterprise Institute. What is Lean? Online Document. Updated 2016. Cited 2.8.2016. Available: http://www.lean.org/WhatsLean/

[Lea16b] Lean Enterprise Institute. Principles of Lean. Online Document. Updated 2016. Cited 2.8.2016 Available: http://www.lean.org/WhatsLean/Principles.cfm

[Mon08] Montgomery D. and Woodall W. An Overview of Six Sigma. *International Statistical Review,* 2008, Vol 76, Issue 3, p. 329–346

[Mun08] Munro, R., Maio, M., Nawaz, M., Ramu, G., and Zrymiak, D. *Certified Six Sigma Green Belt Handbook*. Milwaukee, ASQ Quality Press, 2008.

[Pop13] Poppendieck, M. and Poppendieck, T. *Lean Software Development: An Agile Toolkit.* Crawfordswille, Addison-Wesley, 2013.

[Pre10] Pressman, R. *Software Engineering: A Practitioner's Approach.* 7th edition. New York, McGraw-Hill, 2010.

[Pro09] Probst J. and Case G. 2009. Integrating Six Sigma and ITIL® for Continual Service Improvement. *Pink Elephant.* Online White Paper. 2009. Cited 26.6.2016. Available: https://www.axelos.com/CMSPages/GetFile.aspx?guid=d898a583-dff0-4c99-b082-2354edd725f7

[Sch04] Schuh, P. *Integrating agile development in the Real World.* First Edition. Massachusetts, Charles River Media, Inc., 2004.

[Scr16] Scrum Alliance. Learn About Scrum. Online Document. Updated 2016. Cited 18.4.2016. Available: https://www.scrumalliance.org/why-scrum/

Appendix A: Initial interview questions

BSS | *OSS*

How are development timelines divided? Short or long time intervals? How often is software/products delivered to the customer? | *Would it be possible to cut the timeline to shorter intervals at least in some projects?*

How are teams arranged? Are they located within the same office or further apart? How many people are in a team? | *Would it be possible to divide teams into smaller parts and/or relocate team members more locally?*

How frequent is collaboration between developers themselves? How about between developers and superiors/management? How often are meetings held? | *Do you see problems in more frequent collaboration, e.g., in holding shorter meetings more regularly? Would it be possible to collaborate more with management?*

How are meetings held? Face-to-face or online/phone? What tools are used in meetings?
How are projects planned? Do developers have any say in planning? Are the plans strict or flexible? | *Can you think of ways for the development team to be more active in planning? Are there any obstacles in the development team participating in planning?*

How is feedback gathered from the customer? Both during development and after? | *Are there potential problems in gathering regular feedback?*

What is the customer's role in projects? | *Are there any reasons why customers couldn't participate in projects?*

How are changing requirements and environments reacted to? | *Why is change generally avoided? What are the obstacles for adapting to changing requirements or environments?*

What do you see as the biggest obstacles in applying agile development to work/development processes?

Appendix B: Follow-up interview questions

Development:

Incremental deliveries:

What problems do you see in incremental deliveries in general?
How do you see support and maintenance for deliveries? Where do you think your responsibility with deliveries ends?
Do you see problems in developing also legacy systems in an agile way?
How are requirements defined for prioritizing the backlog?
Do you see problems in applying agile development with the ITIL framework?

Quality of deliveries:

Are the incremental deliveries functional themselves?
What are the testing methods for determining the quality of deliveries?
What are the testing criteria for determining the quality of deliveries?
How could knowledge transfer between development and Operations be implemented most efficiently?

Collaboration:

Would you welcome a representative from Operations and/or business to your team?
What problems do you see in cross border working? How about any solutions to those problems?
Would it be useful to encourage assembling more local teams? Any problems with that?

Operations:

Incremental deliveries:

What problems do you see in receiving frequent incremental deliveries in general?
How do you see support and maintenance for deliveries? Where do you think your responsibility with deliveries starts and the development's responsibility ends?
Do you see problems in developing also legacy systems in an agile way?
Do you think it's possible to apply agile development with the ITIL framework?

Quality of deliveries:

What does your customer, e.g., the business unit, expect from deliveries?
How much influence do you have with development in order to secure quality?
Are you a part of testing with development?

How could knowledge transfer between development and operations be implemented most efficiently?

Collaboration:

Do you think that an Operations representative in the development team would be useful?
What problems do you see in cross border working? How about any solutions to those problems?
Would it be useful to encourage assembling more local teams? Any problems with that?