Aalto University
School of Science
Degree Programme in Computer Science and Engineering

Oskar Ehnström

Using lean software development principles to develop digital services

Master's Thesis
Espoo, April 26, 2016

Supervisor: Professor Marjo Kauppinen, Aalto University
Advisor: Suvi Uski, D.Soc.Sc
Sari Kujala

Aalto University
School of Science

Aalto University
School of Science
Degree Programme in Computer Science and Engineering

ABSTRACT OF
MASTER'S THESIS

Author:	Oskar Ehnström		
Title:	Using lean software development principles to develop digital services		
Date:	April 26, 2016	**Pages:**	vi + 62
Major:	Software Engineering and Business	**Code:**	T-76
Supervisor:	Professor Marjo Kauppinen		
Advisor:	Suvi Uski, D.Soc.Sc Sari Kujala		

Lean software development focuses on cutting waste and enabling companies to focus on creating value. However, traditional businesses may not be able to utilize lean software development on their own. The goal of this thesis was to investigate how a software vendor can utilize lean principles to develop digital services for their customers. The research method used was based on grounded theory and adapted to allow for the existing literature on lean and lean software development.

The current literature on lean software development includes a refined list of lean software development principles. These principles have been studied in a number of case studies about lean and lean software development. There is not enough research to definitively label any list of principles as established. There are some indications that eliminating waste, constant improvement and delivering fast are more known or studied principles. The focus on eliminating waste and constant learning fit withing the model of a software vendor where the domain may be unfamiliar but speed is of the essence and value has to be delivered constantly for the work to continue. Eliminating waste is the principle most associate with lean, but constant improvement and empowerment are important principles at the studied vendor. The benefits of a lean software development project by a vendor are domain knowledge and the ability to pinpoint problems. Domain knowledge can be transferred between domains by the vendor. The vendor can help pinpoint problems in the organization and optimize the whole. Challenges of lean software development projects include the clash between lean and traditional business functions that are not able to move at the same speed. Lean software development projects also require effort form the customer and may result in unpleasant, but useful, results as the result of validating assumptions.

Lean software development is meant to help companies reduce waste and create value. Further study is needed, but lean software development seems to work well for a software vendor who can help the customer optimize the whole and transfer knowledge between domains.

Keywords:	lean, agile, lean software, lean software development, service creation, lean service creation, LSC
Language:	English

Aalto-universitetet
Högskolan för teknikvetenskaper
Examensprogram för datateknik

SAMMANDRAG AV
DIPLOMARBETET

Utfört av: Oskar Ehnström

Arbetets namn:
Att utnyttja lean principer i programvaruutveckling för att skapa digitala tjänster

Datum:	Den 26 April 2016	**Sidantal:**	vi + 62
Huvudämne:	Programvaruproduktion och affärsverksamhet	**Kod:**	T-76
Övervakare:	Professor Marjo Kauppinen		
Handledare:	Suvi Uski, D.Soc.Sc Sari Kujala		

Lean programvaruutveckling fokuserar på att eliminera avfall och göra det möjligt för företag att fokusera på att skapa värde. Traditionella företag kan dock ha svårt att utnyttja lean programvaruutveckling på egen hand. Målet för detta diplomarbete var att undersöka hur ett mjukvaruföretag kan utnyttja lean principer för att skapa digitala tjänster för sina kunder. Forskningsmetoden som användes var baserad på grounded theory och tillämpad för att tillåta existerande literatur om lean och lean mjukvaruutveckling.

Den nuvarande litteraturen om lean programvaruutveckling innehåller en lista över lean-programvaruutvecklingens principer. Dessa principer har undersökts i ett antal fallstudier om lean och lean programvaruutveckling. Det finns inte tillräckligt med forskning för att definitivt klassa några principer som etablerade. Att fokusera på att eliminera avfall och att konstant lära sig passar in i modellen för en mjukvaruleverantör där domänen kan vara obekant, snabb takt är viktigt och man måste leverera fort för att fortsätta. Att eliminera avfall är den princip som mest associeras med lean, men konstant förbättring och bemyndigande är principer som är viktiga hos mjukvaruleverantören som var del av denna studie. Fördelarna med ett lean mjukvaruprojekt utfört av en leverantör är domänkunnande och förmågan att precisera problem och optimera helheten. Leverantören kan förflytta information mellan domäner. Till utmaningarna med lean programvaruutvecklingsprojekt hör konflikter mellan lean och traditionella business-funktioner som inte klarar av samma tempo som lean. Lean programvaruprojekt kräver också insatser från kundens sida och kan resultera i obekväma, men nyttiga, resultat då man validerar antaganden.

Lean programvaruutveckling skall hjälpa företag eliminera avfall och skapa värde. Det krävs fortsatt forskning, men lean programvaruutveckling ser ut att fungerar för en mjukvaruleverantör som kan hjälpa kunden optimiera helheten och dela information mellan domäner.

Nyckelord: lean, agile, agil systemutveckling, tjänsteutveckling

Språk: Engelska

Acknowledgements

I wish to thank Professor Marjo Kauppinen, my thesis supervisor, for her time and effort spent on guiding me through this thesis. I also wish to thank my instructors Suvi Uski and Sari Kujala, as well as Eva Raita, my other thesis camp instructor, for their help.

Thank you also to Futurice and their customers for the opportunity to study and write about the company. And finally, thank you to the other thesis campers for all your support.

Espoo, April 26, 2016

Oskar Ehnström

Contents

Chapter 1

Introduction

1.1 Motivation

The adjective lean is defined in the Oxford Dictionary of English (3 ed.) as "(Of an industry or company) efficient and with no wastage" (Stevenson, 2010). However, the example given in the definition goes on to say: "staff were pruned, ostensibly to produce a leaner and fitter organization", which gives lean an unfair focus on personnel reduction. Lean is an ideology focused on cutting waste, but what constitutes waste may come as a surprise to many. Lean thinking permeates the whole organization and is much more complex than simply reducing staff.

The lean ideology was developed at Toyota to meet the demands on the Japanese car market (Holweg, 2007). The Japanese market for cars was much smaller than the American market and could as such not handle the large batch sizes that were the norm for American factories. Lean advocated a value chain based thinking to produce the needed items just in time, minimizing inventory and other costly activities. This meant Toyota could fulfill the market with minimal resources.

Traditionally technological advancement has had the biggest impact on manufacturing. However, a new service based model is quickly gaining popularity(Vargo and Lusch, 2004). This changes the traditional value chain, where the value was created by the item itself. The service based model states that value is created through the use of the item. This means that technological advancement in services are becoming immensely important. The term for the model where value is created through goods is called goods dominant logic and the model where value is created through the rendering of services is called service dominant logic (Vargo and Lusch, 2008). This idea is by no means new, and was discussed as "servitization" in 1988 (Van-

dermerwe and Rada, 1988). Models of co-creation have also been proposed and studied from the perspective of the social sciences (Edvardsson and Olsson, 1996). A new field of study called service science has been proposed to study the poorly understood service innovation that is taking place (Maglio and Spohrer, 2008).

Some companies have gone as far as providing, what they call, experience centric services (Zomerdijk and Voss, 2010). According to the article these services aim to have several touchpoints with their customers, where customers come into contact with the provider organization in some way. This builds loyalty and a relationship between the customer and service provider. These touchpoints form a customer journey.

> *Uber, the world's largest taxi company, owns no vehicles. Facebook, the world's most popular media owner, creates no content. Alibaba, the most valuable retailer, has no inventory. And Airbnb, the world's largest accommodation provider, owns no real estate. Something interesting is happening.* - Tom Goodwin (Goodwin, 2015)

It could be argued that what is happening is the digitalization of traditional businesses. Companies are under extreme pressure to adapt or disappear. Cloud based infrastructure provides small companies with the ability to scale their services on a level never seen before while big companies struggle with large IT departments and heavy processes.

Lean is a way for companies to adapt a value chain based thinking in order to optimize their whole business, cut waste, and enable faster reacting to upcoming competitors. Manufacturing is, however, not like digital service creation. In the digital space there is no concept of manufactured items. What lean needs to enable is service creation.

1.2 Research problem and questions

Lean principles are becoming more common in the software industry. This can be seen as companies developing products are using practices borrowed from lean manufacturing. The products are not as tangible as in manufacturing, so the focus of lean has to shift from the production line to the software development processes.

The companies that have implemented lean software development and who have been studied are software development organizations building their own products. This is quite natural, as lean needs to spread through the

Question	Literature review	Empirical study
RQ1	x	
RQ2	x	x
RQ3		x

Table 1.1: Research questions and their respective sections

organization in order to be effective. If one controls the organization and product this should be easier to achieve.

Organizations that are not focused on software development often buy software development services from software vendors. It is possible that lean software development could bring just as much value, if not more, to the organization, but this has not been studied.

The research problem is thus defined as follows:

How can a software vendor utilize lean principles to develop digital services for their customers?

To investigate this problem three research questions have been set up in table 1.1.

What are the established lean software development principles? (RQ1)

How can a software vendor utilize lean software development principles in their work? (RQ2)

What are the benefits and challenges of lean software development projects? (RQ3)

Both existing literature and an empirical study is used to answer these questions. The answer to RQ1 is based on the literature review chapter. The answer to RQ2 is based on the literature review chapter as well as the empirical study. The answer to RQ3 is based on the empirical study. The research questions and their corresponding chapters are summarized in Table 1.1.

1.3 Scope of the thesis

This section presents the scope of the thesis.

The scope of the literature review is existing literature on lean software development principles and lean software development projects. The review compares these to find similarities and differences. Articles and books were chosen based on relevance and prominence in search results. Although the material is not guaranteed to represent all of the available knowledge, the coverage can be regarded as sufficient for this analysis.

The scope of the empirical study is one software vendor using lean software development as well as parts of one of their customer organizations where lean projects were executed. The focus on the customer organization is on the projects done together with the lean software vendor and the whole of the organization is not in the scope of this research.

1.4 Structure of the thesis

This section presents the structure of the thesis.

Chapter 1 is the introduction of the thesis. The first section of the chapter presents the motivation for the thesis. The second section section defines the research problem and research questions. The third section presents the scope of the thesis. The fourth and final section goes through the structure of the thesis.

Chapter 2 is the literature review. The first section presents lean principles found in the current literature. These are summarized in Section 2.3. The second section presents four case studies of lean software development done in different companies. The third section is a summary of the chapter.

Chapter 3 describes the methods used in the thesis. The first sections presents the method used for the literature review. The second chapter presents the methodological solution and motivation for this method. The third section presents the data collection methods. The fourth section goes through the participants of the study. The fifth section describes the process of analysis. The sixth and final section evaluates the method that was used.

Chapter 4 presents the results of the empirical study. The first section describes how the software vendor utilizes lean software development methods. In this section the focus is on the software vendor and their ways of working. The second section presents the benefits and challenges of lean software development projects as described by the employees of the vendor and two customer representatives.

Chapter 5 discusses the results and their wider implications. The most important results are reiterated and discussed. The results are also compared to the previous case studies found in literature and criticized where

appropriate.

Chapter 6 presents the conclusions of this thesis and suggestions for further research.

Chapter 2

Literature review

This chapter presents some of the existing literature on lean software development. The chapter also presents some existing literature on past lean software development projects. The case studies are compared to the general principles of lean software development and to each other. Existing principles present in literature are summarized and analyzed in Section 2.1.9. Existing case studies are summarized and compared in Section 2.2.5.

2.1 Lean software development principles

One book in particular has been the foundation of many lean initiatives of various kinds. This is the work by Poppendieck & Poppendieck in "Lean Software Development: An Agile Toolkit" (Poppendieck and Poppendieck, 2003). In this book the authors motivate why lean works in software development and translate some of the common aspects of lean manufacturing into the language of software development. They also present seven principles of lean software development. These are adaptations and expansions of the principles developed for lean manufacturing.

2.1.1 Eliminate waste

Eliminate waste is a fundamental principle of lean. Waste in this context is understood as anything that does not produce value to the customer. This may seem like a clear definition, but once one starts measuring what is actually producing value and what is not the results may be surprising. In manufacturing inventory is considered to be waste. Ideally products should move from one stage to the next immediately. In software engineering one equivalent of inventory is unfinished features (Poppendieck and Poppendieck,

Table 2.1: The seven wastes of software development as defined by Poppendieck & Poppendieck

Waste	Description
Partially done work	Unfinished features and other code not in production
Extra processes	Processes that do not serve to add value
Extra features	Features that end users do not need or want
Task switching	Overhead that comes from context switching between tasks
Waiting	Waiting for others
Motion	Moving something from where it is produced to where it is needed.
Defect	Defects cause work to be repeated

2003). Knowledge about how to deliver value more efficiently is, however, not waste (Poppendieck and Cusumano, 2012). Activities that actually lead to more knowledge about the process need to be identified and separated from those that do not add knowledge. Eliminate waste is also the only principle that was not later modified by Poppendieck & Cusumano (Poppendieck and Cusumano, 2012).

One of the main reasons why inventory is considered waste is that inventory may contain defects that have not yet been discovered (Poppendieck and Poppendieck, 2003). These defects, the book points out, become much more expensive to fix later in the process. Defects are discovered much faster in the next steps of the process when batch sizes are small and inventory is minimal. In software engineering inventory often manifests as features waiting for testing or other forms of approval. Inventory in the form of features stuck at some crucial point in the system may also be blocking other, more important features.

Learning to see waste can be challenging when unfamiliar with the concept. For this reason Poppendieck & Poppendieck have gathered together a list of the seven wastes of software development. This list is based on the seven wastes of manufacturing by Shigeo Shingo (Poppendieck and Poppendieck, 2003). This list is presented in Table 2.1.

Eliminate waste is one principle that is present in all of the case studies presented in this paper (Middleton, 2001)(Middleton and Joyce, 2012)(Middleton et al., 2005)(Rodriguez et al., 2014). This seems to indicate that this is indeed considered to be the most important principle of lean.

2.1.2 Decide as late as possible

Deciding as late as possible is about keeping several options available until the last responsible moment to make a decision (Poppendieck and Poppendieck,

2003). The reason, according to Poppendieck & Poppendieck, is that there will be changes, so the right thing to do is prepare to be able to handle change. This reflects the same thinking as agile: "Responding to change over following a plan" (Fowler and Highsmith, 2001).

Lean principles recognize that it has hard to understand a problem perfectly the first time and plan accordingly (Poppendieck and Poppendieck, 2003). Keeping several options available while learning about the problem and trying different approaches increases the chance of having a working solution when a decision has to be made writes Poppendieck & Poppendieck.

The book also points out that deciding as late as possible is not about avoiding decisions until it is too late (Poppendieck and Poppendieck, 2003). It states that a decision should be made at the last responsible moment. Of course, it may not be easy to know exactly when that moment is, but it is still better than not deciding or locking oneself to one solution that turns out to be the wrong one.

This chapter of the book also mentions several practices that are generally considered good development practices (Poppendieck and Poppendieck, 2003). Separation of concerns, interfaces, and modules are mentioned as some examples of practices that enable deciding as late as possible. All of the examples mentioned in the book are ones that could be considered universal good practices for software development. It follows, that lean does not conflict with agile, or indeed any other development methodology, but rather encourages good praxis.

Empowerment is also mentioned as a way of deciding as late as possible (Poppendieck and Poppendieck, 2003). In this context empowerment is about teaching the people working on a project how to make decisions. The idea is that the people working on the project are the ones who know the needs of the project and can make the best decisions about how to proceed. By allowing the team to make the decision it is possible to avoid overhead and get the best possible information available. There is no need for management to get involved, which frees up management resources for other tasks.

2.1.3 Amplify learning

Amplify learning is the second principle presented in the book, and focuses on quality and learning (Poppendieck and Poppendieck, 2003). This principle was later redefined by Poppendieck as "learn constanstly" (Poppendieck, 2010). The principle of learning is crucial in lean, as the whole process is based on the aforementioned concept of removing waste. For this to be successful waste needs to be identified and this skill has to be learned. The book presents several tools to help facilitate learning.

Many of the tools presented are familiar from the world of agile software development. Feedback is a natural and effective part of learning. For this to be an effective tool in practice there needs to be a feedback cycle and effort to gather feedback. The book also mentions that software projects are often ill structured problems where a mentality of trying something first then fixing it works better than a predetermined plan (Poppendieck and Poppendieck, 2003). This ties closely to the build-measure-learn cycle proposed by Ries (Ries, 2011). Another way of looking at it is seeing the software context as a chaotic system where a clear solution can not be elicited through requirements engineering, but instead has to be found through trial and error.

The authors also mention iterations, a concept central to agile development. (Poppendieck and Poppendieck, 2003) Iterations are cycles of development where after a specified time the status is reviewed and tasks and objectives are either refined or considered done. Iterations enable learning as knowledge about the process and project is refined each iteration.

A couple of tools mentioned that are probably not as well known to those familiar with agile methodologies are synchronization and set-based development (Poppendieck and Poppendieck, 2003). The authors write that set-based development explores multiple options simultaneously before committing to one solution. Synchronization is closely related to continuous integration, where software is continuously released with only minor changes instead of in large batched of changes.

The concept of deciding as late as possible that is mentioned in the original book by Poppendieck & Poppendieck (Poppendieck and Poppendieck, 2003) is not present in the later revision of the principles presented by Poppendieck & Cosumano (Poppendieck and Cusumano, 2012). The principle is, however, included in the "learn constantly" principle. Deciding as late as possible is done to learn as much as possible before making a decision, so including it in the learning principle makes sense.

2.1.4 Deliver as fast as possible

Delivering as fast as possible is a lean practice as well as an agile one. In lean the idea is that when you deliver fast you get feedback fast and can react to the feedback (Poppendieck and Poppendieck, 2003). The principle is mentioned in a later article as "Deliver fast" (Poppendieck and Cusumano, 2012).

To deliver fast Poppendieck & Poppendieck suggest a few tools to use (Poppendieck and Poppendieck, 2003). The first, and perhaps most fundamental, tool to deliver fast in lean is to limit work in progress (WIP). By settings hard boundaries on the amount of work in the system at any given

point the system is encouraged to seek out bottlenecks and resolve them. Reducing WIP also leads to concrete time savings in the form of reduced switching times and the drop in utilization that these cause. On a practical level having a WIP limit shows as a limited amount of tasks per person with a "Doing" status.

Delivering fast means a software project can not be equated with an item to be manufactured (Poppendieck and Cusumano, 2012). Instead, the small, incremental pieces delivered should be the items and the focus should be set on optimizing the flow to deliver these small updates.

The pull system works in such a way that work is not pushed to the next stage, but instead each stage pulls work from the previous stage (Poppendieck and Poppendieck, 2003). This system is used to make sure no part of the process becomes overworked. It also helps people feel empowered, as they know their tasks and can work independently by choosing their tasks from the list of work available from the previous stage.

Queuing theory is another factor in flow thinking and it fits with the idea of lean and delivering fast (Poppendieck and Poppendieck, 2003). The theory is concerned with cycle time, the time it takes from an entity to entering the process to the time it exits. This time needs to be short in order to be able to deliver fast, and for it to be short there can be little queues and variability in the system, as the authors point out.

Poppendieck & Poppendieck also point out that sometimes it is enough to calculate how much time a new tool will save compared to the cost of acquiring the tool (Poppendieck and Poppendieck, 2003). This is because delivering fast might have an impact on how valuable the delivered product is. For example, being first to market might allow you to ask for a premium price that would make the tool worth the cost despite initially unfavorable conditions.

2.1.5 Empower the team

Empowering the team is about motivating the team to do great work. If the team feels they have a purpose and are able to work towards it they can be more productive compared to being told what to do. Access to the customer is mentioned as a positive factor for both motivating and empowering the team.(Poppendieck and Poppendieck, 2003) This concept of empowering the team is closely related to the concept of self organizing teams that agile promotes. The self organized team moves towards a common goal and organizes themselves in the best way to achieve that goal (Cockburn and Highsmith, 2001).

The lean approach promoted by Poppendieck & Poppendieck mentions

the "master developer" as a tool to be used to empower the team (Poppendieck and Poppendieck, 2003). They define the concept as that of an experienced developer who keeps the overall picture in mind. This approach stems from the concept of the master engineer at Toyota (Poppendieck and Poppendieck, 2003), but is not commonly used as a concept in agile. It may be that the concept of self organizing teams and scrum masters have eclipsed the master developer.

As businesses become more dependent on IT and it becomes a part of their core business the concept of IT-departments should also change (Poppendieck and Cusumano, 2012). The authors agree that software development can no longer be seen as a separate entity that can be isolated in some IT-department. To engage everyone and optimize the whole software development will most likely involve other functions besides software development.

2.1.6 Build integrity in

The concept of integrity is in this context understood as quality. The principle was later expressed as "build quality in" (Poppendieck and Cusumano, 2012). Software quality seems like the preferred phrasing these days and if integrity is considered the same as quality a lot of similarities with the lean and agile approaches emerge.

The authors mention model driven design as a tool to achieve integrity (Poppendieck and Poppendieck, 2003). They describe it as a method where a model is construction on which all development is based. This model is understandable by both the customer and the development team. This concept is very close to the agile approach of acceptance testing. Where the customer is asked to test the product in order to validate that a feature was implemented as intended.

Another tactic to achieve integrity presented by Poppendieck & Poppendieck is to use proven components (Poppendieck and Poppendieck, 2003). One possible way to achieve this in daily development would be to use open source components that have been used elsewhere and are battle tested, so to speak.

Refactoring and testing are also mentioned as tools to build in quality (Poppendieck and Poppendieck, 2003). Both are widely accepted good practices and should be a rule rather than an exception in any software project.

The main takeaway from the lean concept of building integrity in, as presented by Poppendieck & Poppendieck, is that they tools they present and advocate for are not that different from the tools used today in software development to achieve good quality. Lean software development advocates for sound engineering principles.

2.1.7 See the whole

See the whole emphasizes the lean concept of flow and optimizing the whole process rather than individual steps (Poppendieck and Poppendieck, 2003). In a later article, Mary Poppendieck talks about "Optimize the whole", a slight modification to the principle (Poppendieck and Cusumano, 2012) the idea however, remains the same.

Local optimizations are mentioned as one specific problem that may seem like added value, but are in fact adding waste to the overall process (Poppendieck and Poppendieck, 2003). One solution also presented by the authors is to use aggregate data. By using data that is not tied to a specific measurable entity, the focus stays on the overall picture.

The authors also point out that it is often hard to see the real cause of a problem. To overcome this they suggest the method of the five whys (Poppendieck and Poppendieck, 2003). Ask "Why?" five times, and you are more likely to find the root cause of a problem and not only the symptoms.

The whole may also be extended outside ones own organization. Including vendors and suppliers in the overall pictures results in possibilities to extend the lean methods further than ones own organization and claim the benefits (Poppendieck and Poppendieck, 2003). This may of course be difficult, and the book goes through several forms of contracts with their pros and cons. The foundational idea in this relationship is trust. When suppliers, vendors and customers can trust each other waste is minimized and value is maximized.

2.1.8 Keep getting better

A new principle, not presented in the book by Poppendieck & Poppendieck, but presented in a later article is "Keep Getting Better" (Poppendieck and Cusumano, 2012). Even when every systems seems to be running well the system is very likely completed and could still be improved. The idea behind this principle is to always look for opportunities to be better. Every systems should be inspected using the scientific method and improved whenever possible, which the authors point out.

Getting better is more than just keeping weekly meetings to brainstorm about what could be done next (Rother, 2009). It should ingrain the whole organization on a deeper level. This may be why Toyota has allowed it's methods to be studied in such details. It is not the practices that matter, but the deeper principles behind them.

Table 2.2: Lean software development principles

2003 (Poppendieck and Poppendieck, 2003)	**2012** (Poppendieck and Cusumano, 2012)
Eliminate waste	Eliminate waste
Amplify learning	Learn constantly
Deliver as fast as possible	Deliver fast
Build integrity in	Build quality in
See the whole	Optimize the whole
Empower the team	
Decide as late as possible	
	Engage everyone
	Keep getting better

2.1.9 Summary of lean principles in literature

The most established lean principles found in this sample of sources are presented in-depth in section 2.1 and listed in Table 2.1.9. The principles have been proposed by Poppendieck & Poppendieck in 2003 (Poppendieck and Poppendieck, 2003), but the idea of using lean for software predates these explicit principles (Raman, 1998). This study is not comprehensive enough to make the claim that these principles are established in the industry overall, but they do provide a good starting point.

The principles have evolved since their conception. Some of the principles have not changed at all or only slightly, while others seem to have completely morphed into new principles. Figure 2.1.9 shows the original principles from 2003 and a subjective comparison with the principles from 2012. The following paragraphs will analyze the principles and how they have changed between 2003 and 2012.

Eliminating waste is still as valid in 2012 as it was in 2003. It seems that the principle had not undergone any major changes or refinements, which would suggest the the principle is valid.

Learning plays a vital role in lean. Formulated in 2003 as “Amplify learning” and 2012 as “Learn constantly” learning has remained a central part of lean software development. The principle is all about generating knowledge to improve the product by trying different variations and having short feedback cycles. This idea has remained unchanged and speaks for the validity of this principle.

Delivering fast is still valid. The name has changed only nominally from

"Deliver as fast as possible" to "Deliver fast". The excuse of "as possible" from 2003 has disappeared as the industry has shown that the tools for delivering fast are available and in use.

Integrity is the same as quality. In the "Build integrity in" principle, the only thing that has changed is the word integrity. The industry has moved to using the word quality, but the idea is still the same. In 2012, the article summarizing the principles focused on continuous integration, while the book from 2003 presents a number of tools for building in integrity.

See the whole and optimize it. "Optimize the whole" as the principle is called in the 2012 article reflects the content of the chapter "See the whole" of the 2003 book. This indicates that the principle has changed very little, if at all. The name has simply been updated to better reflect the idea.

Empowering the team does not have a clear match. Empowering the team is one of two principles presented in 2003 that do not have clear equivalents in the 2012 list. The reason for this is not clear. Some of the other principles cover areas that were included in "Empower the team", which indicates that there is no need for a separate category. Expertise, as presented under this principle, could just as well be covered by "Keep getting better" and "Motivation" fits under the "Engage everyone" principle.

Decide as late as possible is also without equivalent. The second principle from 2003 to not have a clear equivalent in the list from 2012 is the "Decide as late as possible" principle. This principle focuses on being open to change as long as possible. There is no clear way of nesting this under one of the other categories. The principles of "Optimizing the whole" or "Build quality in" could incorporate the ideas from this principle, but are not a perfect match.

Engage everyone is a new principle. The principle of "Engage everyone" was not included in the 2003 book, but is present in the list from 2012. The principle is closely related to optimizing the whole, but focuses more on the people. This focus also makes it related to empowering the team.

Keep getting better is the other new principle. This principle matches the lean principle of constant improvement. It is a difficult part of lean, but an essential pillar of it. It is surprising that it was not more prominently featured in the original seven principles and quite natural that it is on the 2012 list. It ties together with "Learn constantly", but has a more abstract meaning and applies to the ways of working, rather than the product being worked on.

All of the principles presented in this section are established in the current literature to some degree. In the next section case studies and their focus will be presented, which will give some indication of which of these principles are the ones most studied and used.

Short cycles are both lean and agile. Agile uses short iterations, or cycles (Highsmith, 2002) (Williams and Cockburn, 2003) in order to constantly refine what the team is working on. Feedback is gathered early and often to elicit the right requirements (Paetsch et al., 2003). This is the same mentality as lean. There needs to be constant feedback in order to learn what works and what does not.

A multidisciplinary team can also be lean. Agile promotes using self-organizing, multidisciplinary teams in order to achieve a good user experience (Sohaib and Khan, 2010). While this practice has been made known by agile, it matches the lean principles as well. Poppendieck & Cusumano especially point out that as the value stream spans the whole organization, people with different backgrounds will inevitably be involved and will all contribute to the value of the product (Poppendieck and Cusumano, 2012).

Working software provides feedback. The idea of "working software over comprehensive documentation" is know from the agile manifesto (Fowler and Highsmith, 2001), but this does in fact reflect the lean ideology as well. Lean emphasizes reducing waste and constant learning. Comprehensive documentation is waste if the product is not useful for the end users. Proving the idea with working software creates value.

The term "leagile" in software development stems from 1999 and was before that used in supply chain strategy (Wang et al., 2012). The meta study of lean and agile practices found that the terms are sometimes used interchangeably, and at the very least, have many of the same core concepts.

2.2 Past case studies

This section presents four case studies of companies adopting lean software development principles. The first company, Timberline Inc. is a software development company and their lean transformation was focused on the tempo of development. The second organization is BBC World, where the internal software development unit adopted lean principles. The third company is Elektrobit a provider of wireless embedded systems based in Finland. And the fourth organization is a large software company where a controlled experiment was set up to evaluate lean software development.

2.2.1 Timberline Inc.

The case study of Timberline Inc. appears to be one of the first case studies done on lean software development methods (Middleton et al., 2005). In this

study the authors study how a traditional software development company used lean principles in order to streamline work and **cut waste**. The authors were also interested in how companies involved in both manufacturing and software could match the already implemented lean processes of manufacturing to software development. At this time software development was not generally thought of as lean or iterative, but it was clear that software was becoming a larger part of manufactured products.

The focus of this lean transformation was the concept of **takt**, the tempo of development (Middleton et al., 2005). The development team was forced to switch between tasks as too much work was being pushed into the system, causing the amount of work in progress to increase. Incomplete code as well as time spend on switching tasks are sources of waste in lean software development. This switching back and forth, also called thrashing, was mitigated by splitting projects into smaller pieces called 'kits'. Smaller units of work and a WIP limit effectively causes the system to flow more smoothly according to the study. This concept of smaller batch sizes is essential for any iterative process as well as the fast feedback cycle used in lean.

Timberline also used the concept of a daily stand-up meeting familiar from agile as well as moved team members around to form co-located teams (Middleton et al., 2005). This was mentioned to cause some dissatisfaction in the well established organization, but the challenges were overcome. This concept of co-located teams is a core concept of agile for the same reasons it is important in lean software development. Having a short as possible feedback cycle is crucial to catch defects as soon as possible, before they become costly.

The existing processes in the organization caused problems for the movement of people to those areas of the production flow that needed more resources (Middleton et al., 2005). The reason for this was stated to be simple legacy and history. Once these outdated processes were identified they could be improved easily. These habits and old processes may often persist only because there is no radical change in the environment. Adopting lean processes with an open mind will, at the very least, allow the organization to take a critical look at the existing processes.

The study also points out the importance of creating a **learning organization** by gathering information and using it to improve the process (Middleton et al., 2005). The study does not go into detail as to how this was accomplished. The test used for how well information was displayed was that a person unfamiliar with the work could walk into the project area and see the current state.

Self-managing teams were also a result of gathering data and using it for decision making (Middleton et al., 2005). The positive results mentioned

in the study were reduced management costs but another aspect of a self-managing team is likely to be empowerment. The motivation of the team may very well rise considerably when they feel empowered to make their own decisions. Empowerment is also one of the seven principles presented by Poppendieck & Poppendieck (Poppendieck and Poppendieck, 2003).

The clear conclusion of the study was that lean techniques do transfer into software development (Middleton et al., 2005). Extreme examples of improved processes included one in which only 1.4% of steps added value before the move to lean. After implementing lean the time to fix defects during the development cycle dropped 65%-80%, among other improvements. Customer satisfaction was also observed to be "overwhelmingly positive" and was speculated to be because of iterative development.

This study points out that the move to lean has to be a company wide effort (Middleton et al., 2005). This is tightly related to the concept of **optimizing the whole**. The value added by it can only be limited to part of the organization at best if the whole organization is not on board with the lean transformation.

2.2.2 BBC Worldwide

This study followed the introduction of lean practices in a nine person team working for the BBC Worldwide organization (Middleton and Joyce, 2012). The team worked on various tasks for customers inside the BBC and included management, development and testing roles.

The study focused a fair bit on lead time(Middleton and Joyce, 2012). That is, the time from a customer request to the time the requested entity is delivered. Variance in lead time was something that was to be reduced, as low variance makes for a more stable and predictable process. By reducing variance and decreasing the size of units of work, the team was able to reduce cycle times and achieve a greater number of small deliverables. This type of increase in number and decrease in size can enable faster iterations and allows a team to build the right thing more quickly. This concept is tightly related to the principle of **optimizing the whole**, where you optimize the whole value chain in order to maximize throughput.

Building in short iterations also reduces risk, as there is less unfinished code waiting (Middleton and Joyce, 2012). This reduces both the risk of building the wrong thing as well as the risk of bugs in the code. The study noted a practical example of this by stating that customers were able to evaluate a "...tangible product rather than just progress reports". The data also indicated that there were fewer bugs in the code and these bugs were fixed faster. Fewer bugs is of course a good example of the **build quality**

in principle.

A familiar practice from agile, the daily stand-up, was used by the team (Middleton and Joyce, 2012). The daily stand-up is a meeting where the team goes through what they have done, what they will do and if there is anything blocking them, all while standing up. The meeting is meant to be a quick and efficient way of getting the team on the same page. However, the study points out that the stand-up focuses on the tasks and data and not as much on the people which, according to the study, is the custom in agile. This could be a misunderstanding of the ideas of agile by the authors of the study. Both the agile and lean ways of conducting stand-ups focus on knowledge sharing in order to facilitate a smoother development process. Badly implemented ways of doing this may of course affect either implementation and begin to focus more on assigning blame and measuring people, which is usually seen as unhealthy.

The study also pointed out that the stand-ups were not meant for team members to give reports of their work. The most efficient way to conduct the stand-ups was for each team-member to highlight problems they had encountered so these could be addressed and the flow would not be interrupted. This is also the idea behind the agile stand-ups, but it can be misunderstood.

Performing these daily stand-up meetings in front of the Kanban boards used by the team served a special purpose (Middleton and Joyce, 2012). By analyzing the placements of the tasks on the board it was easy to visually see bottlenecks which interrupted the flow of development. Eliminating these bottlenecks would then be one way for blocked team members to do meaningful work while not breaking their WIP limit, as specified by the lean principles. This indicates an awareness of the **optimize the whole** principle.

The type of work the team performed did not change, but the way they handled the requested pieces did. The team focused on splitting requests into small pieces with, what they called, Minimal Marketable Features (Middleton and Joyce, 2012). This idea is closely related to the Minimum Viable Product idea presented by Eric Ries (Ries, 2011). Both are in essence the same approach of delivering maximal value to the customer with minimum effort in order to then learn and adapt.

The study also pointed out that lean practices are not an alternative or replacement of professional software engineering practices (Middleton and Joyce, 2012). This supports the idea that lean is a set of principles, rather than practices. Challenges related to the new, lean, process was, according to the study, often due to constraints with the existing organization and their ways of working compared to the team using lean. The study listed this as a reason why lean is more about a cultural shift than it is about implementing

certain practices.

2.2.3 Elektrobit

Elektrobit is a Finnish provider of wireless embedded systems (Rodriguez et al., 2014). The case study was conducted in 2010 and the organization had used agile practices since 2007. The case study followed how some key performance indicators (KPI) changed as the organization started using lean practices.

The study was particularly focused on how lean and agile practices can be combined in software development (Rodriguez et al., 2014). The authors focused on elements that characterize the combination of lean and agile. They were also interested in what challenges the combination presents, as well as which elements of the combination were easy to implement. As agile has only appeared to increase in popularity in the software development business this study is a good reference on how lean and agile can be combined in order to produce software. The hardware related business also presents some unique challenges compared to purely software based models of lean and agile.

The study found that the move to lean had indeed influenced practices of the company (Rodriguez et al., 2014). Discussions with employees revealed that the lean principles had expanded concepts like reducing waste to be considered throughout the organization. In comparison, previous practices had left these responsibilities largely on the product owner alone. However, the change from agile to lean was described as "an incremental improvement in which Agile is not abandoned when Lean is adopted.". This is quite natural, as agile practices focus more on the software development work whereas lean takes a more holistic approach to the whole value chain.

When the subject was speed and flexibility the discussions with the subjects of the study focused on the importance of short lead-times and the ability to cope with change (Rodriguez et al., 2014). These are important questions business-wise, as maneuvering fast and responding to change can be the deciding factor in a fast paced environment like software development. Responding to change is one of the things both agile and lean aim to enable in order to cope with chaotic systems.

Eliminating waste was seen as an aspect specifically related to lean principles that had not been incorporated in the earlier, agile, practices (Rodriguez et al., 2014). Tightly related to that, seeing the whole also brought a more holistic approach to the company compared to the agile mindset before the change. When discussing specific practices, minimizing work-in-progress (WIP) was found to be easy to motivate and understand as a way to reduce waste. This becomes clear when one thinks of unfinished code as being

the software equivalent of inventory with possible bugs and unnecessary or unwanted features.

Elektrobit's ways of working also enabled them to have short feedback cycles (Rodriguez et al., 2014). This is a critical component for the ability to adapt and respond to change quickly. They also handled uncertainty by continuous learning. Estimations were small but accurate, which is the typical way for agile estimations to work. Another aspect that related to both agile and lean was that participants in the study pointed out that delaying decision making should not affect the release of the software. This borrows from agile as it does not allow the schedule to be delayed, but recognizes that lean principles call for delaying a decision until it has to be made in order to keep all options open.

Lean principles emphasize "perfection" or "learn constantly", at Elektrobit they found that Kanban provided an added value as a process because of its ability to visualize queues and enable finding the root cause of problems (Rodriguez et al., 2014). Agile also focuses on continuous improvement, so the change towards lean was likely a natural evolution towards organizational learning from the more team focused approach of agile.

Learning was enabled by organizational transparency (Rodriguez et al., 2014). This was the most stressed element of the discussions with the participants of the study. Transparency enabled knowledge sharing and enabled visibility on all organizational layers and in all directions.

Finally, participants chose to mention the people factor of software development (Rodriguez et al., 2014). This is very much related to agile, but has its place in lean principles as well in the form of "engage everyone" as presented by (Poppendieck and Poppendieck, 2003).

The study presents some challenges that Elektrobit encountered regarding lean principles (Rodriguez et al., 2014). Flexibility of the whole value stream was a challenge. Teams also perceived that they were unable remove waste even though they had identified it due to complex project set-ups. Long feedback loops were still an issue due to challenges in involving management and third parties into the development process. These challenges are familiar from traditional lean manufacturing and agile software development, which could mean that lean software development was not a silver bullet in this case, even if it did improve the overall situation.

2.2.4 Information Systems Department

In "Lean Software Development: Two Case Studies" author Peter Middleton sets out to study whether lean principles can be applied to software development (Middleton, 2001). This study appears to be one of the first studies that

tackles this question and predates the work of Poppendieck & Poppendieck which was published in 2003.

The study presents the foundation of lean principles and how they might be applied to software development (Middleton, 2001). The most important result, however, is the experiment conducted by Middleton to study the effects of applying lean methods on a traditional software process. In this experiment, two small teams in a large organization were selected to try lean practices in their daily work.

The traditional process was first streamlined somewhat to enable the introduction of lean principles (Middleton, 2001). Once the new process was stable, lean principles were introduced. The implementation focused most on aiming to reduce waste by stopping the process once a defect was found. This initially slowed down both teams, as team members were not allowed to work on other tasks while the issue was resolved. This was done to limit their WIP. Once the teams learned to see defects and address them more quickly overall progress improved compared to the traditional process.

Although the study is very limited, focusing on two small teams for a short period of time, it is able to highlight some of the most prominent organizational challenges. In this particular organization the hierarchical structure of the teams introduced friction in reporting issues, which is an essential part of lean (Middleton, 2001). The hierarchy also fostered a culture where people needed to take jobs they had little aptitude for in order to advance in their careers. One manager also felt that they were securing their job by not sharing information. This could have been a result of the organizations less than ideal policies on learning, which were mentioned as a challenge for the application of lean. Finally, some aspects of the lean process were hindered by third parties inside the organization who were unable to deliver the required quality.

An overall conclusion of the problems uncovered in the study was that problems were often a result of organizational challenges, and the issues with quality were, in fact, the symptom of deeper problems. Finally, the study also concluded that "no inherent reason has been found to suggest that lean techniques cannot be used in software process." (Middleton, 2001). This might have influenced others to try to replicate these results in larger studies.

2.2.5 Summary of case studies

The case studies in Section 2.2 all have their own areas of focus, but some areas are clearly the focus of most studies. Table 2.3 shows a summary of lean principles and the amount of focus they receive in each case study. The table shows that learning has a strong focus in all the case studies. Quality

	Timberline Inc. (2.2.1)	BBC Worldwide (2.2.2)	Elektrobit (2.2.3)	Two Case Studies (2.2.4)
Eliminate waste	Strong	None	Strong	Strong
Optimize the whole	Strong	Strong	Strong	None
Constant learning	Strong	Strong	Strong	Strong
Empowerment	Strong	None	Strong	Strong
Deliver fast	Some	Strong	Strong	None
Build quality in	Some	Some	Some	Strong

Table 2.3: Focus areas of the case studies

Company	**Software area**
Timberline Inc.	Construction industry
BBC Worldwide	Media
Elektrobit	Wireless embedded systems
"Two Case Studies"	Unknown

Table 2.4: Types of software by the case study companies

has at least some focus in all of the studies. Empowerment, optimizing the whole and eliminating waste have a strong focus in three out of four case studies, but no focus in one. Interestingly, there is no single case study that ignores these three principles, the focus is quite spread out among the case studies.

The concept of optimizing the whole was also mentioned as Flow. This corresponds to the original lean model more than the lean software development principle. In this case they can be grouped together under the same general topic.

One aspect of the case studies that should be mentioned is that the "Two Case Studies" (2.2.4) study is very small compared to the other case studies. The focus on this study is therefore not as spread out as some of the others. Some of the principals would also have been hard to measure in this small scale study.

Organizational resistance and challenges related to the existing organization was mentioned in all of the studies as having a negative impact on lean principles.

Finally, the companies studied have one aspect in common: all the companies are software companies focusing on their own product. The technology and business context should, as such, be familiar to the teams working on the products.

2.3 Literature summary

Lean software development principles from various sources have been studied and presented. Some authors have presented more academic frameworks (Poppendieck and Cusumano, 2012; Poppendieck and Poppendieck, 2003) while others have focused on case studies (Middleton, 2001; Middleton and Joyce, 2012; Middleton et al., 2005; Rodriguez et al., 2014). There is, as such, no definitive list of lean software development principles. Based on the principles presented in Section 2.1 and Section 2.2 a summarised list of the principles found in literature has been composed in Table 2.3.

Table 2.3 is not a complete list of all software development principles that could be considered lean, but encompasses the most predominant ones. In Chapter 4 the principles used at a lean software development consultancy company will be presented. The next chapter will present the methods used to gather those results.

Table 2.5: Lean software development principles from literature

Principle	**Description**
Eliminate waste	The foundation of lean. All other principles are based on this notion of eliminating anything that does not produce value.
Learn constantly	In order to eliminate waste there is a need to constantly learn and adapt. Learning applies both to the product and ways of working.
Deliver fast	Delivering fast is essential for learning as the feedback loop needs to be small enough to make a difference in the process.
Build quality in	By reducing quality errors you reduce waste as there is less need to fix issues. A broken product is waste.
Optimize the whole	By optimizing the whole you reduce wait times and inventory, both of which is considered to be waste.
Empower the team	Empowering the team allows those who are familiar with the product make the decisions to avoid unnecessary overhead and thus waste. It also motivates the team.
Decide as late as possible	Deciding as late as possible gives time to learn and make decisions based on new information. This reduces the change of producing unusable work.
Engage everyone	Engaging everyone is closely related to both optimizing the whole and empowering the team. It enables everyone to work for a common goal.
Keep getting better	This is very closely related to the learning constantly principle. Apply what you learn in order to always keep getting better.

Chapter 3

Research methods

This chapter presents the methods used to gather and analyze the data of this study. First, the method for gathering literature is described. Second, the research problem is reiterated and the motivation for the used methods are presented as well as described in short. Third, the way data was collected through interviews is presented. Fourth, the participants and their backgrounds are disclosed. Fifth, the process of analysis using grounded theory is explained. Finally, the methods are critically analyzed and their validity and credibility are evaluated.

3.1 Literature review

This section presents the methods used to gather material for the literature review.

Google Scholar was the primary source for material on lean and lean software development. Searching the numerous databases in Google Scholar with the keywords mentioned in Figure 3.1. Once some suitable primary articles had been found these could be used to find related articles. By utilizing the related articles feature on Google Scholar more related articles could be found. The sources of known articles also led to other articles on the subject and especially to heavily cited articles that provide the foundation for many other articles. Searching for articles written by the authors of known articles also led to other articles that dealt with the subject. Conferences and journals that included some known articles also proved to include other similar articles and were a good source of material.

Search term
lean software development
lean software management
lean project management
digital service creation
lean

Figure 3.1: Keywords used to find sources

3.2 Methodological solution

The research problem defined in Section 1.2 aims to understand the lean software development process. Choosing a method that focuses of understanding a phenomenon is preferable, as there is limited data about the research area. The most straightforward way of understanding the underlying culture of lean in this context is by interviews. A proven method is needed to analyze the data from the interviews. Grouping by themes found in literature, such as the principles presented in Chapter 2, is one option. Another option is to analyze the data as such, without preconceived notions of the results. Grounded theory (GT) is well suited for this purpose. Furthermore, grounded theory can be done iteratively using both top to bottom and bottom up approaches.

Grounded theory was first presented by Glaser and Strauss in 1967 (Glaser and Strauss, 1967). Their field was sociology and the aim of grounded theory was to enable qualitative researchers to formulate theories based on scientific methodology which would be relevant in that field. The method was aimed at sociologists, but the authors recognized that the theory could be useful for any domain looking to understand social phenomena using qualitative data.

As the name suggests, grounded theory is grounded in the data (Glaser and Strauss, 1967). This means that when using grounded theory one starts by gathering data and analyzing it. This is different from the traditional approach in science where you first form a hypothesis and then verify or disprove the hypothesis by empirical study. This roundabout way of forming a theory has led some to calling the hypothesis formed by grounded theory as "reverse engineered" (Lazar et al., 2010).

Various forms of empirical data can serve as the starting point for grounded theory research (Lazar et al., 2010). In this research the source material is interviews, but GT can also be used with ethnography, observation, and case studies among others.

The GT method is performed in four steps (Lazar et al., 2010). The

first step is coding. This is where the researchers identify phenomenon in the source material and gives them names or codes. In GT coding is done as open coding. This refers to the fact that the coding is done with an open mind, letting the concepts emerge from the source material and not from outside influences. The second step is developing concepts. This is where codes that describe similar phenomenon are grouped together to form concepts. In the third step, categories are grouped together to form categories. Finally, a theory is formed based on the causal connections between concepts and categories.

Qualitative research differs from quantitative research in non-trivial ways (Golafshani, 2003). It seeks to understand a phenomenon in a certain context and does not place as much emphasis on repeatability as quantitative research. This does not make qualitative research less valid, but it does make it different.

3.3 Data collection

The empirical study was conducted as a series of interviews. Five interviews were conducted, each lasting between 50 and 65 minutes.

The interviews were conducted as semi-structured interviews. Two sets of questions were prepared, one set for Futurice employees and one for customer employees. The Futurice employee questions were modified slightly to better suit the expertise of the participant. The questions may be found in Appendix A. During the interviews participants were encouraged to answer with their own interpretations. Follow up questions were also asked based on the direction of the answers the participants gave.

The questions were prepared based on the lean principles presented in Chapter 2. Subjects were asked about previous experiences with software projects and with lean methods. The questions explored project based work as well as organizational culture. Questions were also left intentionally up for interpretation in order to map the subject's own interpretations and avoid unnecessary bias by the interviewer.

3.4 Participants

The participants were from Futurice (3 participants) and Futurice's customer organizations (2 participants).

Participants were asked to sign a letter of informed consent before the interview. They were also told that they may end the interview at any

time and choose not to respond to any and all questions. Furthermore, the participants had the right ask for any disclosed information to be removed from the final interview notes.

The interviews were recorded in audio format and detailed written notes were done based on the recordings. The interviewer agreed to archive the recordings responsibly and not share the recordings with any third party. The letter of informed consent can be studied in Appendix B.

After each interview the subjects were asked for feedback regarding the interview. This feedback allowed the interviewer to refine the questions between interviews. There were also some good statements about the subject that were mentioned during this free-form part of the interview and were included in the final transcription of the interview.

3.5 Process of analysis

The interviews were analyzed in two rounds. The first round was a bottom up approach, following the traditional grounded theory ideology as closely as possible. The second round was a top down approach where the research questions were used as a basis for analysis.

Coding of the interviews was done using the Atlas.ti software. In the first round the transcribed notes from the interviews were imported into the software and codified manually. The process consisted of reading through the notes and highlighting interesting concepts and quotes while assigning these their own codes. The end result was 607 different codes, although many of these were almost identical and many were different ways of expressing the same idea. The challenge was to remember the exact codes used for a specific phenomenon from one article to the other. Fortunately the software suggested codes based on the typed characters which made the coding more consistent throughout.

Categorization can also be done with the ATLAS.ti software. The codes were grouped together based on similarity. Several codes could easily be grouped together under the same concept, but there were unique codes as well. The categorization naturally focused on the codes that appeared most often in the interviews. Figure 3.2 shows the ATLAS.ti view of the different codes (on the right) as well as groups of codes (on the left). These were used as basis for categorization.

In the end post-it notes was the way to go. Once the codes had been grouped by similarity and theme it the underlying concepts started to emerge. The ATLAS.ti software has a feature that allows users to make networks of concepts. The initial idea was to use this feature to build the concepts and

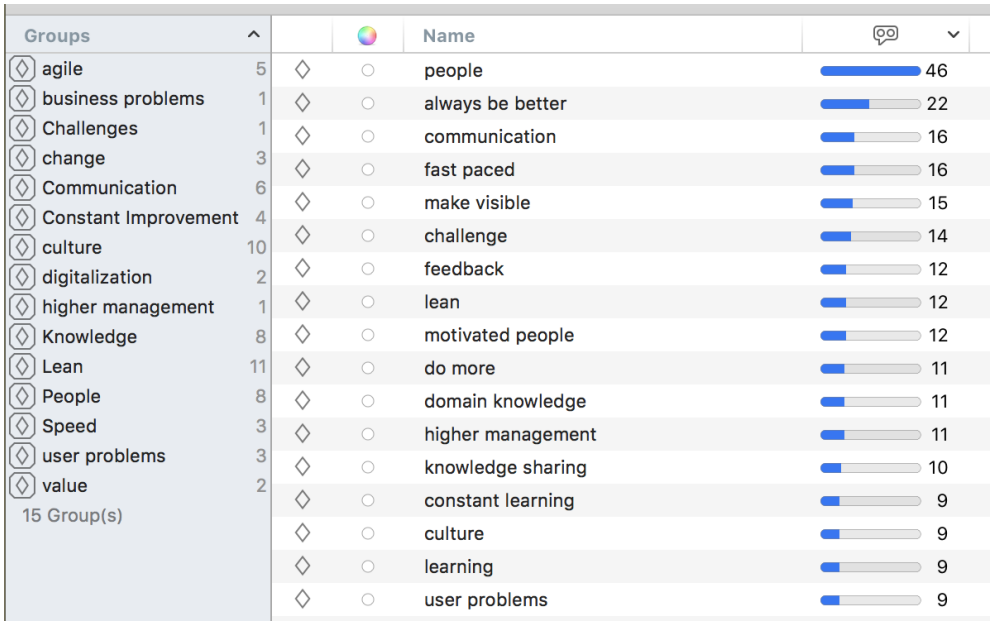

Figure 3.2: The ATLAS.ti software can sort codes based on their frequency

connections between them, but the software was, in the end, not as natural as post-it notes on a board. Groups of codes were written down and grouped together, then concepts around these were written down on post-it notes. These were then grouped according to their respective research questions. Figure 3.3 shows the process of categorization on the boards using post-it notes.

In the end some clear concepts did emerge from the first round on analysis. The brainstorming and physical activity of grouping the post-its proved to be a good way of looking at the concepts from a different angle.

The second round of analysis was a top down approach. In this round the literature and research questions were used to guide the analysis. The challenge with this was to keep the observations grounded in the data and not allow the prior knowledge taint the results of the analysis while still searching for links between the literature and the empirical results.

In the second round of analysis the transcripts of the interviews were again used as basis for the analysis together with the codes from the first round. During the second round the codes were linked with concepts from the relevant literature where appropriate instead of grouping together all codes into separate categories.

This iterative way of analysis proved to be a good way of gaining a more thorough and multidimensional view of the data. Although the dataset was

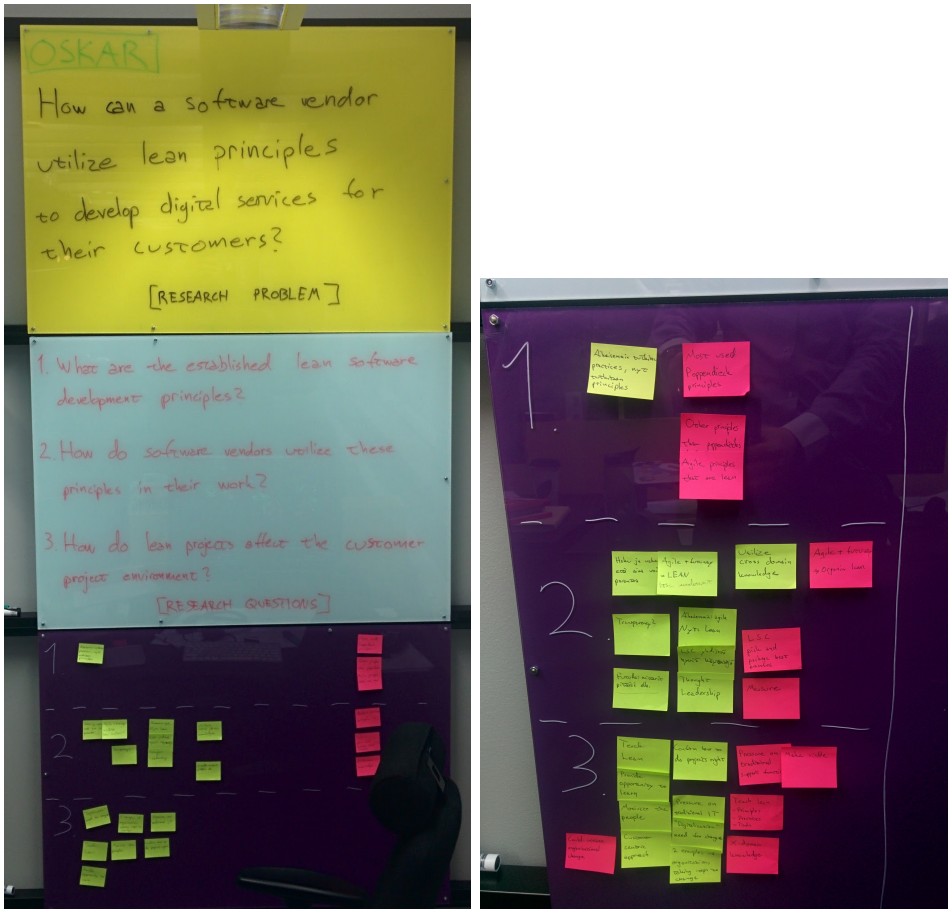

Figure 3.3: Post-it notes on a "whiteboard" in the process of being categorized.

quite small some insight could be distinguished from the data.

3.6 Method evaluation

The preconceptions and background of the author were recognized and taken into account during the analysis process and special care was taken to analyze the data sensitively. However, it can still be assumed that the analysis is not entirely objective, as the researcher always affects the process.

Subjects were asked for clarifying questions and viewpoints when they seemed to veer too far off course. Clarifications about statements were asked to confirm the assumed meaning of answers which could have been interpreted by the subjects as "the right answer" and thus confirmed. The subjects were, however, informed repeatedly that there were no wrong answers and as such they could feel safe answering any way they wished.

The credibility of this study is sound but the low number of subjects and organizations studied makes the findings of the study specific and not generalizable. However, the results of the study serve as a good baseline for further research. The analysis of this study is valid for a specific set and can be used as a base for further study.

The interviews were conducted in one round. Ideally GT is done iteratively so that once one round of interviews has been analyzed the theory can be confirmed or refined by a second round of interviews. This study lacked the resources to conduct several iterations of interviews, which has to be taken into account when assessing the validity of the data.

Themes from the existing literature were clearly present in the answers of the subject. Some of the questions were specifically formulated to seek out answers related to the existing literature, but even those that were not supported the principals present in the existing literature. Existing literature supports the findings of the study.

Chapter 4

Results

This chapter presents the findings of the empirical study. An analysis of the interviews is compared to the existing lean software development principles found in literature.

4.1 Utilizing lean principles

This section presents how a software vendor can utilize lean software development principles. The main focus of the section is on the empirical study conducted at a specific software vendor (Futurice Oy). The lean software development principles in use at the vendor are presented, analyzed and compared to the principles present in existing literature.

4.1.1 Lean principles at Futurice

Eliminating waste is associated with lean. All of the subjects at the software vendor and at the customer mentioned eliminating waste when asked what they associated with lean. This indicates that this principle is among the more widely known aspects of lean software development. One subject at the vendor mentioned that their daily work varied a lot, but they were always evaluating whether what they were doing was producing value.

The concept of seven wastes by Poppendieck & Poppendieck presented in Chapter 2 can be used to categorize the wastes recognized at Futurice. Out of the seven wastes extra processes and extra features were spoken of in the interviews, as well as waiting for others. These seems to be forms of waste that were found and recognized as waste. Defects were also mentioned in the interviews, but partially done work, motion and task switching were not explicitly mentioned as sources of waste. No specific processes were

mentioned that would be examples of cutting waste, but instead individuals and their impact were emphasized.

Feedback and validation eliminates waste. Asking for feedback and validating what is being built were aspects of eliminating waste that were mentioned in the interviews. Practical examples of avoiding waste was expressed by one subject in the form of avoiding unnecessary meetings and documentation by being close to the customer.

There needs to be some freedom for innovations. Subject F2 said that a part of their daily work was less efficient and optimized time in order to spur innovation. This is one aspect where lean is applied not by optimizing a certain task, but by trying to optimize the whole in new and innovative ways.

> "[Lean] is doing things that have value and doing them with minimal movement."

Engineering laziness is a practical example of lean. The concept of "engineering laziness" was proposed by F2. The example they gave was that engineers are inherently lazy and want to eliminate any unnecessary work. This would make engineers good at adopting the lean principles and explain why lean can be adopted and appreciated in engineering domains.

Recruitment is crucial to eliminate waste at the vendor. Both F1 and F2 agree that people at Futurice are the main driving force behind eliminating waste. The recruitment process seeks out people who pro-actively seek out and eliminate waste. This makes the recruitment process extremely important, but it also makes the process of eliminating waste somewhat hidden. There is no clear company wide program or process for eliminating waste, it is just part of everyday work life.

> F2: "Constant learning is probably required for our existence."

Constant improvement is ingrained in the culture. Subjects F1, F2 and F3 all expressed that they felt that more could be done when discussing good practices at Futurice. This opinion was expressed several times during the interviews by all subjects at Futurice. One practical example of what could be done better came from F1, who said that more could be done to show where and how waste has been eliminated. Another example of practices that enable constant improvement was expressed by F2 as regular retrospectives and reviews in daily activities. The opinion that there is always room for improvement shows a value shared by all subjects and which corresponds with the lean principles.

Measuring is needed for improvement. When asked what more could be done to eliminate waste F3 responded with simply: "measure". This reflects the opinions on measuring of F1 and F2 as well. In order to improve there has to be a way of knowing whether any improvement is happening. Measuring and especially validation were discussed by F1 as essential parts of lean work. Furthermore, F2 pointed out the need to validate ideas fast in order to minimize wasted effort put into work based on invalid assumptions.

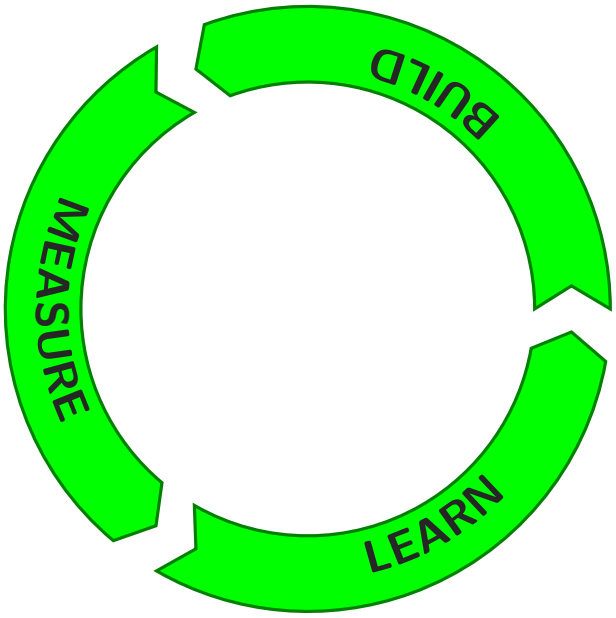

Figure 4.1: The Build-Measure-Learn cycle (Ries, 2011)

The build-measure-learn cycle shown in Figure 4.1.1 is presented in the book "The Lean Startup" (Ries, 2011). The author, Eric Ries, advocates for a way of validating a startup using lean principles. The ideas from this book has been combined with agile and design thinking to form the basis for Futurice's "Lean Service Creation". It is easy to see why measuring plays such a big part in the interviews with Futurice employees. The build-measure-lean cycle presents it as a way of validating whether your product solves a problem for your users.

At Futurice Build-Measure-Learn is used to develop new services. Subject F2 saw the build-measure-learn cycle as something that has been around, in some form, for a very long time and nothing new in that sense. The BMS-cycle is an iterative process, meaning it promotes splitting the work into iterations and refining the work as the process goes along. In F2's opinion

Futurice has borrowed from this and other good practices and created a lean way of developing digital services. Subject F1 also points out that in the center of the build-measure-learn cycle is the user's and customer's needs.

Delivering fast is seen as an agile practice. The lean principle of delivering fast was not discussed separately in any interview at the vendor. Rather, it is seen as a tool for learning and validation. Subject F3 says that they try to get feedback as early as possible, and for that they need something to deliver.

> F3: "Usually after one week we have something to show the customer"

One problem mentioned by F2 and F3 regarding delivering fast was that the customer might not be available to give feedback on the deliverable. In that case delivering fast does not produce value since there is no learning to be gained from the customer.

Delivering fast can be seen as being linked to the constant improvement principle. To be able to constantly improve there is a need for constant and feedback, which in turn requires the pace of delivery to be fast.

Building quality in is important for the vendor, but is is not seen as a part of lean, but rather as a more general good practice. One important aspect mentioned by F2 is that the people take ownership of their projects and thus produce quality. In fact, one challenge mentioned by F2 is that the concept of quality might be different for the customer and the vendor. This could be, for example, from political reasons inside the organization according to F2.

F3 mentions practices like continuous integration and refactoring as "Obvious things to do" when discussing learning during the project. These are practices that directly relate to the quality of the project. F3 also says that these initiatives always comes from the vendor team in their experience.

Building quality in also ties together with the *Defect*-waste presented in Table 2.1. This shows that the Eliminate waste principle ties together with building quality in.

Optimize the whole is a more unknown principle. The idea of optimizing the whole organization is quite important in the Lean Service Creation concept developed at Futurice, but it is less known on its own.

The LSC model works on the new service creation level, and as F1 points out: "it should affect the whole customer organization.". However, there is no structured way for project teams to optimize the whole. The project teams do try to affect the customer, but it appears that the challenge is reaching wide enough inside the customer organization. F2 points out that

the worst improvement suggestion is the one that is never made, and trying to understand the customer's processes is always important.

Empowerment is important at the vendor. The central rule at Futurice is called "3x2" and is stated by subject F2 as being a central part of their daily work: "...always thinking 3x2, is there value?". The rule, pictured in Figure 4.2) is a guidline for making decisions and states that people, customer and numbers now and in the future should be considered. Subjects F1, F2, and F3 all state that autonomy is an important part of the type of work performed at Futurice. The management system at Futurice enables this kind of empowerment, as stated by F1. The 3x2 rule seems to be an important part of employees having freedom and responsibility while still playing by the same rules.

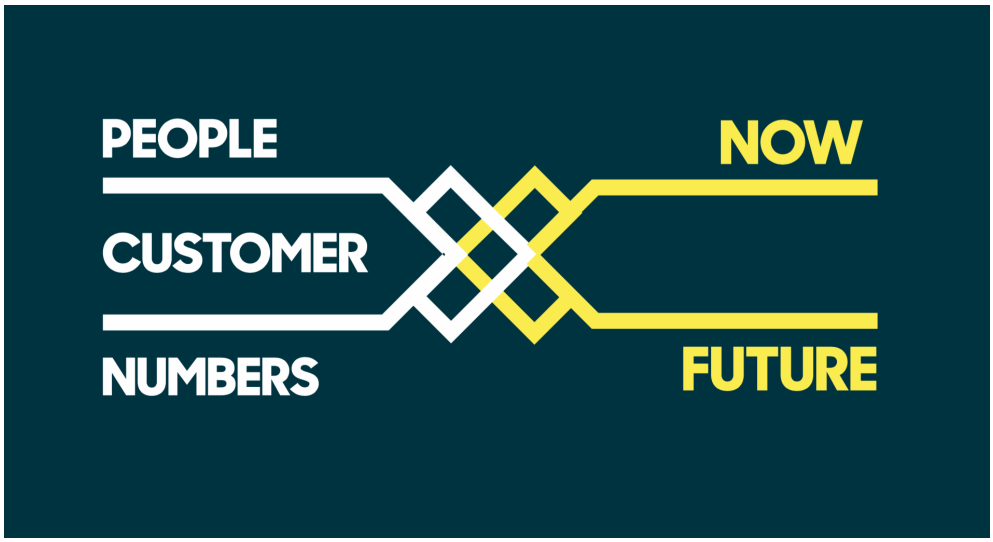

Figure 4.2: Futurice model of 3x2 (Used with permission)

Decide as late as possible is a quite unknown principle. The principle was not discussed in interviews at the vendor or customer. Iterations was much more in the focus of the work at the vendor and their customer.

It may be that this principle is baked in to the concept of constant validation, learning and iterations. Further study is needed to investigate whether this is the case or if the principle simply is not as applicable to software development as it is to manufacturing.

Engaging everyone is an interesting principle for a software vendor. According to F1 the LSC model of solving a common problem should steer the project towards engaging everyone. F3 mentions internal discussion platforms as tools for engaging, but says more could be done on this front.

Principle	Description
Eliminate waste	Most known and applied principle in use at Futurice. The principle is applied in daily activities by using it to guide decisions in a pragmatic way.
Constant improvement	The principle seems to be of the culture and a driver behind the mentality that there is always improvements to be done.
Deliver fast	This is at the core of the business, but is seen more as an agile practice, although it definitely is lean.
Build quality in	Quality is seen as a good practice and is highly emphasized, though not specifically as a lean practice.
Optimize the whole	This principle is seen less in the daily development, but is highly emphasized in the digital strategy consulting.
Empower the team	Empowerment and trust are core values at the company, but are not commonly associated with lean.
Decide as late as possible	This principle is not used as a lean principle, although some of the aspects of it are part of development.
Engage everyone	Engagement is not a core part of lean principles at the company, but is seen as an important aspect of the work.

Table 4.1: Summary of lean principles used at Futurice

There is potential for improvement in this regard. Subject C1 from the vendor's customer pointed out the benefits of working with motivated people, so this aspect is successful in some regards and could be utilized more.

4.1.2 Summary

In summary, the most known and explicitly used principles at Futurice are eliminate waste and constant learning. These principles are also the ones most closely associated with lean software development. Empowerment is a big part of the culture at Futurice, but it is not seen as directly related to lean software development.

The activities described by the subjects at Futurice are not always clearly a part of just one lean principle. This could be indicates that the principles are related and some principles may be linked to each other. It could also be a result of the less structured way Futurice has adopted lean. The principles have not been applied with a certain list of lean software development principles. As F1 said in their interview: "First we build Futurice, then we found out about this thing called lean."

Futurice clearly implements some of the lean software development principles presented in Chapter 2. A summary of the principles and their usage is presented in Table 4.1.2. In the next section the benefits and challenges of implementing these lean principles will be presented.

4.2 Benefits and challenges of lean software development projects

This section presents the benefits and challenges of lean software development projects. These findings are the results of interviews with Futurice employees and Futurice customers.

4.2.1 Benefits

Lean software development can be used to **pinpoint problems in existing processes**. Subject C1 was asked how lean has affected projects they responded by saying that it has put a lot of pressure on existing IT-support functions. This indicates that the current IT-support model is not well suited for the lean model. This is valuable information for the company since the first step of eliminating waste is to identify it.

Lean software development promotes **optimizing the whole** organization. Subject F1 points out that if a new service is created for their business it should affect the whole organization: "...it should affect absolutely everything in that client organization". F1 says that, in the end, digital services end up integrating into all activities. This allows allows the organization to focus on creating value.

Cross domain knowledge, which is a valuable resource, is promoted by lean software development. Respondents from both Futurice and their customers feel that Futurice can provide value by utilizing knowledge gained from one domain in another domain. F1 said they felt Futurice may in-fact provide more value compared to a customer's internal project team because of this cross domain knowledge, which was not possible before. By constantly learning and reflecting on their own work, Futurice can build on previous knowledge from various domains and combine seemingly separate pieces of knowledge.

The lean approach to constant learning and improvement **supports the vendor employees in learning** about a domain that is not their core expertise. Both F2 and C2 see domain knowledge as valuable and something that could be utilized more. C2 saw domain knowledge as a way for Futurice

to bring a stronger package to the customer and way to utilize practices across customers. F2 said the Futurice has a lot of domain knowledge and has a genuine interest in learning about the customer's organization.

Domain knowledge is gathered in various ways. F2 said that domain knowledge is gathered in various ways, among them being close to the business environment and constantly asking questions. F1 also states that asking "why?" repeatedly is important. Furthermore, F1 lists *finding people who do know*, *following money flows*, *having an open dialogue* and *showing that you don't know* as ways of accumulating domain knowledge. The answers given by the subjects indicate that a genuine interest in the customer domain is essential. There are several practical ways of gathering the information once you have the right mindset.

Lean software development projects **teach customers about lean principles**. C1 expressed that they had learned considerably about lean from Futurice. C2 also said that the lean project they were a part of had confirmed their opinion of how projects should be done, and spoke of concepts such as constant feedback and short cycles. Although these are not terms used as much in lean literature, the concepts match the lean ideology. Furthermore, C2 said they had learned how to get to the working solution.

Lean software development can **motivate people to learn**. C1 said that they had learned about practical tools used in lean projects as well as the lean philosophy. Furthermore, C1 said that learning is something that motivates them in their work. This matches very well with the opinion expressed by F1, who pointed out that the customers are people and want to enjoy their work and work with motivated people.

Lean projects can be an example. When asked how big organisations can be more lean F2 responded by saying: "Big organisations change through small example projects". They expanded on that by saying it is hard for big companies to change, so change usually happens as a result of successful small example projects. They also said there are plenty of examples of this. F1 echoed this and listed two examples of organizations where Futurice first implemented small example projects, but went on to initiate a organizational change. In one case the ways of working have already, according to F1, caused a change on an organizational level.

4.2.2 Challenges

Lean software development projects may be demanding on traditional support functions like IT-support, as pointed out by subject C1. This can make these projects quite challenging to implement successfully. Where before tasks were planned well in advance and IT support was aware of future

needs, lean requires that IT functions are able to respond to rapid change and new needs for integrations across functions. Subject C2 also mentioned that IT has historically been seen as a support function and that once data is seen as a strategic element, outsourcing IT is no longer possible.

Integrating any new system into other existing systems, both IT-systems and existing business processes, may be challenging. Lean software development promotes optimizing the whole, which could easily lead to many new integrations. C1 says that the amount of problems caused by integrations into other systems was made clear by the software project they were a part of. The fact that the problems were clear in a lean project indicates that lean does not automatically solve these kinds of technical problems. C1 states that in order to move a project forward pressure is needed in the right places and this was actively done in a lean project.

Lean projects by a vendor require effort from the customer. C2 points out that utilizing an outside vendor for projects, including lean projects, is expensive and requires the buyer to be able to utilize the vendor. According to C2, the internal processes of the customer organization should not be in the way of the vendor. This could present some challenges. Furthermore, C2 mentions that a challenge of buying a project from a vendor is that knowledge is not increased inside the company.

Another challenge faced by the vendor is that the quality required by the customer may not be what the vendor expects. Since a lean vendor is focused on building quality in and avoiding waste like bugs in their code it may be hard to accept that the customer needs to priorities other factors over quality of the deliverable. This was pointed out in the interview by subject F2. The challenge here seems to be the opposite of the customer not getting good enough quality, but rather the vendor focusing too much on quality. The solution for this, as proposed by F2, is clear communication.

> F2: “Appreciating a negative result is not natural, but people in this business will learn to appreciate it.”

Validating assumptions about the user can be hard. One challenge of lean and LSC, as pointed out by F1, is that validating assumptions requires some bravery. There is the risk of learning that a solution is not the right one. F1 says that it is easier to think that the solution is the important part, not the problem. The importance of this is mirrored by F2, who says that one may go years building something blindly, without knowing if it actually works, which makes you think it does work. F2 points out that it is just as important to learn if something does not work as it is to learn if it does. Invalidating a solution or assumption can be seen as a negative result by the

Table 4.2: Summary of benefits in lean software development projects

Benefit	**Description**
Pinpoint bottlenecks	Lean requires optimizing the whole and focusing efforts on eliminating wasteful activities. It can be used to pinpoint processes that do not scale and become bottle necks for other processes.
Cross domain knowledge	Lean has a strong focus on constant learning and improvement. This translates to domain knowledge for the vendor that can be utilized in other projects and may be transferable to other domains.
Teach lean	Lean projects teach the people involved about lean and how it works in practice.
Be an example	Lean projects may also function as examples of the lean ideology and show the rest of the organization how and why lean works.
Learning motivates	Learning new ways of solving problems motivates the people involved in lean software development projects.

customer, but it is just as important as a positive result. That is why F2 says that customers will learn to appreciate it.

Finally, one challenge for the vendor is penetrating the customer organization. According to F1 new digital services should affect the whole organization. This, according to F1, makes it crucial to be able to affect the customer organization. At the same F1 admits that this does sometimes fall short due to other constraints.

4.2.3 Summary

There are many benefits to lean software development projects (Table 4.2), but there are some challenges as well (Table 4.3). Lean can present problems for integrating into the surrounding non-lean organization, as was the case in some existing case studies (Middleton and Joyce, 2012). Lean can also pinpoint bottleneck by existing processes and teach the lean ideology.

The biggest difference found in the interviews with subjects at the software vendor and their customer compared to the existing case studies was the concept of transferable domain knowledge. This sets the software vendor and customer relationship apart from the case studies where software was developed in-house. By constantly learning in all projects the vendor is able to amass domain knowledge from various domains and can utilize this

Table 4.3: Summary of challenges in lean software development projects

Challenge	**Description**
Demanding on support functions	Lean can be very demanding on support functions, like IT-departments, that are not used to the pace and requirements of lean projects.
Integration to existing systems	Integrating into existing systems is challenging and is often required by lean projects that strive to optimize the whole.
Effort from the buyer	Lean projects require effort from the buyer to make sure there are no roadblocks that stop the lean process.
Validation of assumptions	Validating assumptions may be challenging to do and may result in apparent negative results which are often incorrectly seen as a failure of the process which may cast doubt on lean software development principles

knowledge in other domains. This could enable the customer of the vendor to gain a competitive advantage and advance their own field.

The challenges presented by lean software development projects may not be that different from the problems faced by non-lean projects. The difference, it could be argued, is that lean amplifies all these challenges by being a faster way of working and more demanding of existing functions.

Chapter 5

Discussion

This chapter discusses the results presented in Chapter 4. Results are placed in a wider context and their implications are reviewed.

5.1 RQ1: Established lean principles in literature

Lean principles have been studied in literature in parallel with agile. The two concepts are closely related, but not identical. Where agile frameworks focus more on the practical aspects of software development, lean can be seen as a set of higher principles comparable with the core of the agile manifesto. Eliminating waste seems to be the most known and most fundamental of these principles.

The lean concept of inventory as waste translates well into software development, since waste is anything that does not produce value for the end user. This is a very strong stance and it is not easy to tell what exactly does and does not produce value. Practical examples of waste include unfinished code or code which is not useful for the end user. In software code can also end up as waste if it is blocked from reaching the end user by a bottleneck in the release process.

Even when the code reaches the end-user it may still be waste if it is not useful for the user. Feature bloat is when features are added but provide no additional value. This can happen when there is a strong focus on adding features and not on solving the users' needs. Lean could be a way for companies developing software to embrace a way of thinking that does not encourage feature bloat, but focuses on the essentials of the software and solving the user need as efficiently as possible.

Constant learning is crucial when developing digital services. On a production line focused on producing a large quantity of standardized products the average worker did not need to continuously learn new methods. In fact, requiring workers to constantly learn new methods could conceivably have slowed them down. Lean was developed in an environment where resources were scarce and every bit of work had to go into producing value for the end user. This is very much like the competitive market of digital services today. Lean is an excellent fit into this market because it focuses on cutting all activities that do not produce value for the end user, and as such, it allows a lean company to move faster than their competitors.

Delivering fast is a core lean software development principle. The short iterations required for the lean process to gather feedback mean that a lean software development project needs to be able to deliver fast and often. If an implemented feature turns out to not produce value for the end user it needs to be cut before any more effort is put into it. The effort can then be directed towards another feature that might produce value.

In summary, eliminating waste, constant learning and delivering fast, could be argued to be the core lean principles. However, this study alone is not sufficiently broad to make that statement. Finally, there are indications that the other principles could be seen as ways of eliminating waste instead of being on the same level, but more data and analysis is required to make such claims.

5.2 RQ2: Utilizing lean principles

Eliminating waste is clearly the principle most closely associated with lean. This becomes clear in the interviews with Futurice employees as well as their customers. This principle is easy to grasp on a superficial level and is easy to agree with. The challenge is to understand the principle on a deeper level and accept the definition of waste the lean promotes.

Constant improvement is important to all of the subjects. In every interview at both Futurice and their customers the need for constant improvement was emphasized. This seems to be a lean principle that is built into the people who are part of lean projects. There were no mention of external motivation for improvement, but the subjects still emphasized constant improvement.

In order to improve you need to know your progress. Measuring was mentioned several times in the interviews with Futurice employees. The concept of measuring may seem cold and impersonal, but for engineers, the data-driven approach of measuring in order to improve may feel natural. It is also important to note that measuring does not imply measuring personal

performance, but rather the overall system. This is something that is easily misunderstood and has a negative connotation in some circles.

It is by no means certain that every individual at the organization has the same understanding of lean. The vendor promotes a mindset where individuals and teams function with high autonomy and are thus expected to analyze their own behavior. To aid in this analysis, Futurice uses a rule called 3x2 where the consequences to people, customer and numbers are considered now and in the future for the outcome of your decision. This rule is an example of lean software development in practice.

> We trust our people to make the best decisions themselves. Nobody else can know better what they should be doing.
> - *Hanno Nevanlinna, Culture Director*

There could be some risks associated with this high level of autonomy if the organization is unable to successfully recruit or train their employees to follow the lean mindset. However, trust is one of the company's core values, and it has been successful thus far.

From the interviews conducted with Futurice employees, it is clear that lean is seen as just one formalization of good principles to follow when creating digital services. The principles that are mentioned match the lean principles, but it is clear that these principles have been accumulated through independent experiences and knowledge sharing. Not through forced doctrine.

Lean is more than a process or tool that can simply be taken off the shelf and implemented into any organization. Lean has to be adapted to the organization. This is perhaps the most important lesson to learn regarding lean. There is no amount of certificates or practices that can make a truly lean organization. There is a lean work culture present at Futurice, but there are few forced rules or processes. The people do not talk about lean, but instead of the practical forms it manifests as. For example, optimizing the whole manifests as communicating with the customer and trying to affect their ways of working to better support lean projects.

Lean can be used with both agile and waterfall. There seems to be no conflict between lean and agile or lean and waterfall according to the subjects. Lean is seen as a "higher principle" and to include "ways of working, purpose, and values". Lean is specifically said to not be an alternative to agile or waterfall by one of the subjects.

Agile is a way of being lean. What becomes clear quite quickly in the interviews with people at Futurice is that Agile is a way of implementing lean principles. The focus of many agile processes may be in the tangible

steps and phases of the process, but the idea underlying the whole of agile is lean. By constantly refining what the team does and trying small changes instead of creating large specifications, the teams reduces waste and creates value for the end-user.

Compared to the other case studies (Middleton, 2001; Middleton and Joyce, 2012; Middleton et al., 2005; Rodriguez et al., 2014) it seems clear that Futurice has a less structured approach towards lean. A lot of emphasis is places on eliminating waste and constant improvement but less on documented practices that should be followed. Another notable exception at Futurice compared to the other case studies is that empowerment and responsibility is heavily emphasized by the culture.

5.3 RQ3: Benefits and challenges of lean software development projects

The benefits and challenges of lean software development projects done by an outside vendor are largely the same as with other lean software development projects. These benefits and challenges are often similar to the problems faced also by non-lean projects, but they may be amplified by the fast pace of lean projects.

The biggest difference between in-house lean projects and lean software development projects implemented by a software vendor like Futurice is the domain knowledge gained by the vendor. This flow of information is pictured in Figure 5.1. The domain knowledge increases the ability of the vendor to deliver value to that domain, but also enables them to share and adapt domain knowledge from other domains to their current projects.

Lean software projects are a good way to pinpoint bottlenecks by the existing processes. Because the lean principles focus on eliminating wasteful activities and optimizing the whole they reveal problems with existing processes and functions. An example of this is traditional IT support functions that are unable to keep up with the pace of development in lean software development projects.

Teaching lean principles is also a big part of lean software development projects. The people involved in lean projects learn how to utilize the benefits of lean principles in their work. Learning about lean principles also serves to motivate the people through learning new skills that are useful to solving their problems. Furthermore, lean projects also function as an example of the lean methodology for the rest of the organization and teach lean in practice on an organizational level.

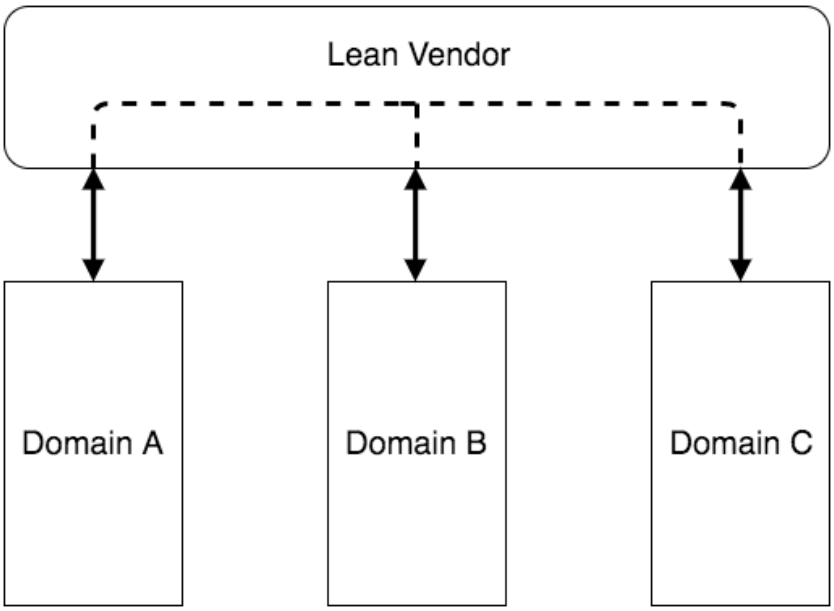

Figure 5.1: The flow of information between the vendor and different business domains

The challenges of lean software development projects are mostly around how the project fits in to the existing organizational structure and the existing software systems. While putting pressure on existing processes and functions may be a good thing in the long run, it still presents a challenge for the project that needs to be overcome in order to succeed. The existing support functions may simply not be able to work at the speed the lean project requires.

Making sure the vendor is not producing waste by waiting on others requires effort from the customer. This is another challenge of lean projects. The customer needs to be able to allocate resources to making sure the lean project is able to deliver value.

Finally, one challenge of lean software development projects is that validating assumptions may be hard and may lead to findings that are discouraging. Since the lean process promotes validating assumptions in order to find what produces value, it may also lead to invalidating assumptions. This can feel negative results, but they are valid results and should be seen as a positive outcome. The process works as intended and by invalidating an assumption less waste is produced and the organization may move on to the next assumption instead of wasting resources on the invalid assumption.

The benefits and challenges presented here should be seen as a representation of one case study. Although these results are from interviews with subjects at a software vendor and their customer the sample size limits the

scope of the results. To be able to verify these findings a larger sample size would be needed from a wider range of both vendors and customer organizations.

Chapter 6

Conclusions

Lean principles translate well into software development. Lean principles should allow software vendors to gain an advantage in development speed and quality according to the existing literature. Software development companies can move towards building the right thing faster by focusing on creating value for the end user. They can also build the thing right, by focusing on quality and allowing the people who are knowledgeable about the product make the best decisions. Furthermore, they can keep doing this by constantly learning and adjusting their ways of working, never stagnating and always wanting more.

Lean software development principles have been studied in current academic literature, and there are a number of principles that can guide lean software development. These principles are based on the proven lean methodology as well as on sound software engineering practices.

Lean and agile are definitely different, but this does not mean that they are incompatible. Lean can be seen as a higher principle, whereas agile gets down to a more practical level. Eliminating waste, focusing on creating value for the user and constantly learning are lean software development principles that are core to the agile software development processes through various practices under various names.

At Futurice, lean is utilized widely though not through directly applying principles found in current literature. Eliminating waste is the most known principle and the one associated with lean. Constant learning and improvement is also a lean principle that is widely used and understood at the company. Although Futurice is lean on many levels, there is still room for improvement. As the participants from Futurice all pointed out: "We could do more."

Lean software development principles can help a software vendor pinpoint problems in the customer organization. This happens as lean focuses on

optimizing the whole and meets resistance in the organization surrounding it.This can help the customer focus on problem areas in order to optimize their whole organization.

A lean software vendor can transfer knowledge between different business domains as they serve different customers. This means practices and principles that work in one domain may be utilized in another domain that would perhaps not have come up with this on their own. One interesting area for further study would be to see if this can be utilized more than before due to the digitalization of businesses.

This study should be considered as a starting point for a more in-depth follow up. The findings provide an interesting starting point and need to be validated with a more thorough study involving several companies. Other software vendors should be studied to see if the findings presented in this study can be applied to them. A more in-depth literature review of other industries besides the software industry could also provide valuable insight of transferable principles and practices which could be applied to software engineering.

Lean applied to a wider range of competences in software development should be studied. The participants of this study, although from different backgrounds, represent only a part of the different competences involved in digital service creation. How lean affects the work of designers, service designers, analysts and business consultants, among other competences, should be studies in more detail.

I have no doubt that lean principles will continue to be applied in other industries. Manufacturing provided the starting point of modern lean principles, but many of the underlying ideas have already been proven to be naturally transferable to the software development industry and the digital service creation industry. We might very well see it adopted in more traditional service industries.

As long as there is competition between companies lean will succeed. Even though adopting lean can be challenging, it can be done, and companies who can manage this will eventually weed out those who can not. Our resources are limited and there is simply no room for waste.

Bibliography

Alistair Cockburn and Jim Highsmith. Agile software development:the people factor. *Computer*, 34(11):131–133, 2001. ISSN 00189162. doi: 10.1109/2.963450.

Bo Edvardsson and Jan Olsson. Key concepts for new service development, 1996. ISSN 0264-2069.

Martin Fowler and Jim Highsmith. The agile manifesto. *Software Development*, 9(August):28–35, 2001. ISSN 10708588. doi: 10.1177/004057368303900411.

Barney Glaser and Anselm Strauss. The discovery of grounded theory: strategies for qualitative inquiry. *Aldin, Chicago*, 1967.

Nahid Golafshani. Understanding reliability and validity in qualitative research. *The Qualitative Report*, 8(4):597–607, 2003. ISSN 14754762. doi: 10.3367/UFNr.0180.201012c.1305. URL `http://www.nova.edu/ssss/QR/QR8-4/golafshani.pdf`.

Tom Goodwin. The battle is for the customer interface, May 2015. URL `http://techcrunch.com/2015/03/03/in-the-age-of-disintermediation-the-battle-is-all-for-the-customer-interface/`.

J. Highsmith. What is agile software development? *The Journal of Defense Software Engineering*, 15(10):4–9, 2002. ISSN 00189162. doi: 10.1109/2.947100.

Matthias Holweg. The genealogy of lean production. *Journal of Operations Management*, 25(2):420–437, 3 2007. ISSN 02726963. doi: 10.1016/j.jom.2006.04.001.

Jonathan Lazar, Jinjuan Heidi Feng, and Harry Hochheiser. *Research methods in human-computer interaction*. John Wiley & Sons, 2010.

Paul P. Maglio and Jim Spohrer. Fundamentals of service science. *Journal of the Academy of Marketing Science*, 36(1):18–20, 2008. ISSN 00920703. doi: 10.1007/s11747-007-0058-9.

Peter Middleton. Lean software development: Two case studies. *Software Quality Journal*, 9(4):241–252, 2001. ISSN 09639314. doi: 10.1023/A:1013754402981. URL `http://link.springer.com/10.1023/A:1013754402981`.

Peter Middleton and David Joyce. Lean software management: Bbc worldwide case study. *IEEE Transactions on Engineering Management*, 59(1): 20–32, 2012. ISSN 00189391. doi: 10.1109/TEM.2010.2081675.

Peter Middleton, Amy Flaxel, and Ammon Cookson. Lean software management case study: Timberline inc. *Extreme Programming and Agile Processes in Software Engineering*, pages 1–9, 2005. ISSN 03029743. doi: 10.1007/11499053_1. URL `http://dx.doi.org/10.1007/11499053_1`.

F. Paetsch, A. Eberlein, and F. Maurer. Requirements engineering and agile software development. *WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003.*, pages 308–313, 2003. ISSN 1080-1383. doi: 10.1109/ENABL.2003.1231428. URL `http://www.computer.org/csdl/proceedings/wetice/2003/1963/00/19630308-abs.html`.

Mary Poppendieck. Lean software development principles. *October*, pages 1–2, 2010. doi: 10.1201/EBK1439817568-12. URL `http://www.poppendieck.com/`.

Mary Poppendieck and Michael a. Cusumano. Lean software development: A tutorial. *IEEE Software*, 29(5):26–32, 2012. ISSN 07407459. doi: 10.1109/MS.2012.107.

Mary Poppendieck and Tom Poppendieck. *Lean software development: an agile toolkit*. Addison-Wesley Professional, 2003.

Sowmyan Raman. Lean software development: is it feasible? *Digital Avionics Systems Conference, 1998. Proceedings., 17th DASC. The AIAA/IEEE/SAE*, 1:C13/1–C13/8 vol.1, 1998. doi: 10.1109/DASC.1998.741480.

Eric Ries. *The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses*. Random House LLC, 2011.

Pilar Rodriguez, Jari Partanen, Pasi Kuvaja, and Markku Oivo. Combining lean thinking and agile methods for software development: A case study of a finnish provider of wireless embedded systems detailed. *2014 47th Hawaii International Conference on System Sciences*, pages 4770–4779, 2014. ISSN 15301605. doi: 10.1109/HICSS.2014.586. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6759188`.

Mike Rother. *Toyota Kata: Managing people for improvement, adaptiveness and superior results*. McGraw-Hill Professional, 2009.

Osama Sohaib and Khalid Khan. Integrating usability engineering and agile software development: A literature review. *2010 International Conference on Computer Design and Applications, ICCDA 2010*, 2(Iccda):V2–32–V2–38, 2010. doi: 10.1109/ICCDA.2010.5540916. URL `http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5540916`.

Angus Stevenson. *Oxford Dictionary of English (3 ed.)*. Oxford University Press, 2010. ISBN 9780199571123.

Sandra Vandermerwe and Juan Rada. Servitization of business: Adding value by adding services. *European Management Journal*, 6(4):314–324, 1988. ISSN 02632373. doi: 10.1016/0263-2373(88)90033-3.

Stephen L. Vargo and Robert F. Lusch. Evolving to a new dominant logic for marketing. *Journal of Marketing*, 68(1):1–17, 2004. ISSN 0022-2429. doi: 10.1509/jmkg.68.1.1.24036.

Stephen L. Vargo and Robert F. Lusch. From goods to service(s): Divergences and convergences of logics. *Industrial Marketing Management*, 37 (3):254–259, 2008. ISSN 00198501. doi: 10.1016/j.indmarman.2007.07.004.

Xiaofeng Wang, Kieran Conboy, and Oisin Cawley. Leagile" software development: An experience report analysis of the application of lean approaches in agile software development. *Journal of Systems and Software*, 85(6): 1287–1299, 2012. ISSN 01641212. doi: 10.1016/j.jss.2012.01.061. URL `http://dx.doi.org/10.1016/j.jss.2012.01.061`.

Laurie Williams and Alistair Cockburn. Agile software development: It's about feedback and change. *Computer*, 36(6):39–43, 2003. ISSN 00189162. doi: 10.1109/MC.2003.1204373.

L. G. Zomerdijk and C. a. Voss. Service design for experience-centric services. *Journal of Service Research*, 13(1):67–82, 2010. ISSN 1094-6705. doi: 10.1177/1094670509351960.

Appendix A

Interview questions

This appendix includes the questions used to guide the semi-structured interviews.

Interview questions

Futurice 1

Time: *60-90min*
Subjects: *Futurice employees involved in projects*

1. Basic information
    a. Age
    b. Title
    c. Experience in software
2. What is your role at Futurice?
3. How does a typical day, if there is such a thing, look like for you?
4. How fast paced would you say the work here is?
    a. Why? compared to what?
5. How long does it take for a project to start? (calendar time from go-ahead to first hour billed)
6. What are the factors that affect project startup time? (calendar time)
7. At what stage of the process do you try to get feedback from the customer? (Prototype, sketch, idea)
8. Do you feel that the customer is able to provide feedback as often as you'd like?
    a. If not, how do you deal with that? What does it affect?
9. Do you make an effort to learn how to do things better during the project?
    a. How, or why not?
10. How do you feel about these fast paced projects?
11. Are you familiar with lean software development?
12. What is lean in your opinion?
13. What is different with lean? compared to waterfall and agile
14. How does lean show in your daily work?
15. Are customers interested in lean?
    a. Why do you think that is?
    b. What about agile?
16. Are customers moving to lean methods?
    a. Why?
    b. What about agile?
17. What is lean service creation?
18. Do customers want to learn about LSC?
    a. Why?
19. Do you think lean service creation brings benefits to your work?
    a. How about the customer's?
20. What are the biggest challenges for customers right now?

    a. Does lean/LSC/agile help them deal with those?
    b. How?
21. What are the biggest challenges for Futurice in customer projects?
22. How does Futurice deal with those challenges?
    a. using lean?
    b. other methods?
    c. examples?
23. As Futurice is not the expert in the customer's domain, how do projects handle the unknowns of the customer domain?
24. Do you try to influence your clients to modify their processes?
    a. Why?
    b. How?
25. Why do you think companies buy projects from Futurice?
26. What would you say is the reason companies like Futurice can be lean and agile?
27. If a part of an organization wants to be lean, but the organization is not friendly to change, how do you think lean projects could be set up anyway?
    a. Could Futurice help with this?
    b. Could Futurice do this to show it can be done?
28. Does futurice facilitate building quality in?
    a. How?
    b. What more?
29. Does Futurice facilitate constant learning?
    a. How?
    b. What more could Futurice do?
30. Does Futurice eliminate waste?
    a. How?
    b. What more could they do?
31. Does Futurice empower teams?
    a. How?
    b. What more could they do?
32. Does Futurice engage everyone?
    a. How?
    b. What more could they do?
33. Does Futurice see the whole?
    a. How?
    b. What more could they do?
34. What do you think I should have asked, but didn’t?
35. Do you have any feedback for me?

Futurice 2

Time: *60-90min*
Subjects: *Futurice employees with focus on culture*

1. Basic information
    a. Age
    b. Title
    c. Experience in software
2. What is your role at Futurice?
3. How does a typical day, if there is such a thing, look like for you?
4. How fast paced would you say the work here is?
    a. Why?
    b. Compared to what?
5. What is lean?
6. What is different with lean?
    a. Compared to waterfall and agile
7. How does lean show in daily work?
8. Are customers interested in lean?
    a. Why do you think that is?
9. Are customers moving to lean methods?
    a. Why?
10. What is lean service creation?
11. Do customers want to learn about LSC?
    a. Why?
12. What are your goals when educating customers about lean/LSC?
    a. For them?
    b. For Futurice?
13. What are the biggest challenges for customers right now?
    a. Does lean/LSC help them deal with those?
    b. How?
14. What are the biggest challenges for Futurice in customer projects?
15. How does Futurice deal with those challenges?
    a. using lean?
    b. other methods?
    c. examples?
16. As Futurice is not the expert in the customer's' domain, how do projects handle the unknowns of the customer domain?
17. Do you try to influence your clients to modify their processes?
    a. Why?

    b. How?
18. Why do you think companies buy projects from Futurice?
19. What would you say is the reason companies like Futurice can be lean and agile?
20. If a part of an organization wants to be lean, but the organization is not friendly to change, how do you think lean projects could be set up anyway?
    a. Could Futurice help with this?
    b. Could Futurice do this to show it can be done?
21. Does Futurice facilitate constant learning?
    a. How?
    b. What more could Futurice do?
22. Does Futurice eliminate waste?
    a. How?
    b. What more could they do?
23. Does Futurice enable teams?
    a. How?
    b. What more could they do?
24. Does Futurice engage everyone?
    a. How?
    b. What more could they do?
25. Does Futurice see the whole?
    a. How?
    b. What more could they do?
26. What do you think I should have asked, but didn't?
27. Do you have any feedback for me?

Customer

Time: ~60min
Subjects: Product owners, stakeholders in software projects

1. Basic information
    a. Age
    b. Title
    c. Experience in the business
2. Mikä on roolisi teillä?
1. Mitä siihen sisältyy?
2. Miltä tyypillinen päivä näyttää sinulle?
3. Monessako softaprojektissa olet ollut mukana täällä?
4. Oletko ollut mukana softaprojekteissa aikaisemmin?
5. Eroaako Futuricen kanssa tehdyt projektit muista kokemuksistasi?
    a. Miten eroaa?
6. Mikä on Futuricen projekteissa erityisen hyvää?
    a. Entä huonoa tai vaikeaa?
7. Onko Futuricem kanssa tehdyt projektit muuttanut käsitystäsi softaprojekteista?
    a. Miten?
8. Miksi ostetaan projekteja ulkopuoliselta toimijalta?
9. Mikä on sinä hyvää
    a. ja mikä huonoa?
10. Mitkä ovat futun vahvudet
    a. ja mitkä ovar teidän yhtiön vahvuudet?
11. Mitä opit Futuricelta?
12. Mitä luulet että Futurice oppi teiltä?
13. Onko Futuricen tapa tehdä projekteja vaikuttanut teidän prosesseihin ja kulttuuriin?
14. Miksi näin on käynyt? Miksi ei?
15. Miten Futuricen ja teidän yhtiön yhteistyö voisi olla vielä parempaa?
16. Onko teidän yhtiön toiminta on muuttunut yleisen digitalisaation myötä?
    a. miten?
    b. mahdollisuuksia vai uhkia?
    c. miten siitä saisi mahdollisuuden?
17. Onko lean käsitteenä tuttu?
18. Osaatko kuvata leania omin sanoin?
19. Entä Lean Service Creation?
20. Voitko kuvata sitä omin sanoin?
21. Oletko osallistunut LSC kurssiin?
    a. Oliko se hyödyllinen
    b. Mitä opit siellä?

22. Olisitko kiinnostunut osallistumaan LSC kurssiin?
23. Miten sanoisit että lean ja lean service creation on vaikuttanut projektien tekemiseen täällä?
24. Miten teidän yhtiössä löydetään ja päätetään uusista mahdollisuuksista?
    a. Kilpailijoilta, sisäisiä kilpailuja?
25. Pyrkiikö teidän yhtiö parantamaan omia prosessejaan sisäisesti ja karsimaan turhaa tekemistä?
26. Miten?
27. Onnistutaanko tässä?
28. Mitä voisi tehdä enemmän tämän suhteen?
29. Panostetaanko teidän yhtiössä kokonaisuuksien näkemiseen? Esimerkiksi projektissa, syntyykö pullonkauloja?
30. Miten pullonkauloja vältetään tai ratkaistaan?
    a. Voidaanko niitä ratkaista projektissa itsenäisesti?
31. Pyrkiikö teidän yhtiö tukemaan jatkuvaa oppimista?
32. Miten?
33. Mitä voisi tehdä enemmän?
34. Miten laatuun suhtaudutaan teidän yhtiössä? Asiakaslähtöisesti/Sisäisesti…
35. Onko ihmisillä teillä yleensä monta asiaa hoidettavana kerrallaan?
    a. Miksi? Miksi ei?
    b. Mitä voisi tehdä jos näin on?
36. Siirtyvätkö projektit osastolta toiselle teillä?
    a. Miten se onnistuu?
37. Mikä on sellainen kysymys jonka minun olisi pitänyt kysyä mutta en kysynyt?
38. Mitä palautetta sinulla on minulle?

Appendix B

Letter of informed consent

This appendix includes the letter of informed consent signed by the participants in the study.

SUOSTUMUS TUTKIMUKSEEN

Diplomityö
Aalto Yliopisto / Futurice Oy

Oskar Ehnström
Tietotekniikan kandidaatti
oskar.ehnstrom@aalto.fi
Puhelinnumero 050 30 45 038
Perustieteiden korkeakoulu
Aalto Yliopisto

Vastuullinen professori Marjo Kauppinen
Department of Computer Science, Aalto University School of Science
marjo.kauppinen@aalto.fi

Allekirjoittamalla tämän suostumuksen suostun haastateltavaksi tietotekniikan opiskelija Oskar Ehnströmin diplomityö -tutkielmaa varten. Tutkimusjulkaisuihin valittavissa haastatteluotteissa ei esitetä haastateltavien tai muiden haastatteluissa mainittujen henkilöiden nimiä ellei siitä pyydetä erikseen lupaa. Haastattelutallenteista kirjatut tekstitiedostot ja haastattelujen äänitallenteet arkistoidaan mahdollista jatkotutkimusta varten. Ymmärrän, että tiedot käsitellään luottamuksellisesti.

Helsingissä __/__ 2015

_____	_____
Haastateltavan allekirjoitus	Tutkijan allekirjoitus
_____	_____
Nimenselvennys	Nimenselvennys