Tampere University

Roope Yli-Pukka

AGILE METHODS IN HARDWARE DEVELOPMENT

Master's Thesis
Faculty of Engineering and Natural Sciences
Examiner: Prof. Minna Lanz
Examiner: Dr. Niko Siltala
December 2021

ABSTRACT

Roope Yli-Pukka : Agile Methods in Hardware Development
Master's Thesis
Tampere University
Mechanical Engineering
December 2021

Manifesto for Agile Software Development is celebrating its twenty-years anniversary. Agile methods are widely acknowledged in software development but are recently seen beneficial in other industries as well. This thesis is done as a part of an internship in a large multinational leading technology company. Motivation for the thesis came from the company's need to align development practices within the research and development organization including software, hardware, and validation teams.

The objective of this thesis is to find out how suitable agile methods are in hardware development, what are the best practices and how to integrate those in the development process. This thesis aims to answer to three research questions:

1. How agile methods can be used in hardware development?
2. How agile methods are currently seen in target company?
3. How to get started with agile development?

Theoretical background of the thesis introduces agile methods in general and compares the differences of agile in software and hardware development. Agile methodologies such as Scrum, Kanban, Scrumban and LeSS are introduced among with the traditional development methods. Case study follows a research path which is built around the observation of the present development process, survey study and specialists' interviews.

The key findings of this study have revealed the current opinions and experiences towards agile. Thesis identifies practices that can be used in hardware development to make it more agile. Finally, a suitable agile method is proposed and guidance on how to get started is introduced.

Keywords: agile, Scrum, LeSS, hardware, product development

TIIVISTELMÄ

Roope Yli-Pukka : Ketterät menetelmät laitteiston kehityksessä
Diplomityö
Tampereen yliopisto
Konetekniikka
Joulukuu 2021

Ohjelmistokehityksen ketterän kehityksen manifestin julkaisemisesta on kulunut 20 vuotta. Ketteriä menetelmiä on alettu soveltamaan myös laajemmin kuin pelkästään ohjelmistokehityksessä. Motivaatio diplomityölle syntyi tarpeesta tutkia ketterien menetelmien soveltuvuutta laitteistokehityksessä samalla yhdistäen tuotekehitysprosessia eri tiimien välillä. Diplomityö on tehty osana harjoittelua johtavalle teknologia alan yritykselle.

Diplomityön tavoitteena on tutkia ja löytää sopiva ketterä menetelmä kohde yrityksen tiimille, joka vastaa laitteisto- ja mekaniikka-suunnittelusta. Tavoitteena on löytää sopivia käytäntöjä ja kuinka integroida nämä käytännöt osaksi tiimiä. Työssä vastataan kolmeen tutkimuskysymykseen, jotka ovat:

1. Kuinka ketteriä menetelmiä voidaan soveltaa laitteisto- ja mekaniikkasuunnittelussa?
2. Kuinka ketteriin menetelmiin suhtaudutaan kohde yrityksessä?
3. Kuinka päästä alkuun ketterän kehityksen menetelmillä?

Työn tausta ja kirjallisuuskatsaus esittelee ketteriä menetelmiä yleisesti. Ketteristä menetelmistä Scrum, Kanban, Scrumban ja LeSS ovat esitelty syvällisemmin. Työssä käsitellään myös lyhyesti perinteisiä tuotekehitysmenetelmiä. Työn metodologia on tapaustutkimus, joka sisältää tuotekehitysprosessin havainnointia, kyselytukimuksen sekä asiantuntijoiden haastattelut.

Työn tuloksena esitellään kokemuksia ja suhtautumista ketterän kehityksen menetelmiin kohdetiimeissä. Työssä tunnistetaan hyödyllisiä ketterän kehityksen käytäntöjä, joilla voidaan tehdä laitteisto- ja mekaniikkasuunnittelusta ketterämpää. Päätuloksena työssä ehdotetaan kohdeyrityksen laitteisto- ja mekaniikkasuunnitelutiimille ketterän kehityksen menetelmä ja ensiaskeleita menetelmän käyttöönotossa.

Avainsanat: ketterät menetelmät, ketterä kehittäminen, Scrum, LeSS, tuotekehitys

PREFACE

This thesis has taught me a lot about agile development practices. In truth, I could not have achieved my current level of success without a strong support group. First, I would like to thank the case study company and my manager for the opportunity to do the master's thesis. The journey has been challenging yet rewarding. I would also like to thank the thesis supervisor Niko Siltala for truly helpful comments and tips during the writing process. Thank you, prof. Minna Lanz, for examining the thesis together with Niko Siltala. Thank you Reetta Mäkinen for your continuous support and love during the project.

Special thanks to the gentlemen deep in the Skellari for making these past years unforgettable.

Tampere, 28th of December 2021

Roope Yli-Pukka

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

ACM	Association for Computing Machinery
Ch	Chapter
EE	Electronics Engineering
HW	Hardware
JIT	Just-In-Time
LeSS	Large-Scale Scrum
ME	Mechanical Engineering
OOPSLA	Object-Oriented Programming, Systems, Languages & Applications
R&D	Research and Development
SW	Software
WIP	Work in Progress

1. INTRODUCTION

Manifesto for Agile Software Development (Beck et al., 2001) is celebrating its twenty-years anniversary. During these twenty years of product development agile practices and principles are increasingly adopted in software development (Feldmuller, 2018; Gustavsson & Rönnlund, 2013; Klein et al., 2016). Agile methods are recently seen as an opportunity in other fields as well when a need for new development methods are generally acknowledged (Ovesen, 2012). A recent report (State of Agile, 2021) shows that other areas than software development have adopted agile principles and practices as well. While hardware development differs in many ways from the software development it is seen possible yet challenging to adapt agile methods into hardware development. This thesis will lead its readers to the world of agile methods and will look at agile in the aspect of hardware development.

1.1 Background and motivation

This thesis is done as a part of an internship in a large multinational leading technology company. The R&D unit in Finland is part of innovation business unit whose focus is expand company beyond its core business and find new ways to add significant value to the company and provide new ways to use company's technologies in the markets. There are teams in hardware design (combining electronics and mechanics design), software development and validation. The primary function of the R&D unit is to take risks and develop and test new ideas quickly.

During the pandemic the unit went from onsite to remote working and there was a need to increase transparency. The hardware team adapted from software team an agile approach to product development and started to implement some agile practices from the Scrum Framework such as Product Backlog, Sprints and Daily Scrums. Software team have been using Scrum successfully for years, so it was decided to try Scrum for hardware development. Along with the Scrum the hardware team started to use Jira Software as an agile project management tool. They had already used it for test planning and bug tracking but for example for mechanical engineers it was mostly new tool. This thesis was an opportunity to study how hardware development can benefit from agile practices.

1.2 Research scope and questions

The objective of this thesis is to find out how suitable agile methods are in hardware development, what are the best practices and how to integrate those in the development process. This thesis intends to answer the following three research questions:

1) How agile methods can be used in hardware development?

Agile development is iterative and Incremental rather than plan-driven and strictly scheduled development process. Hardware development differs from Software development a lot and introduces new challenges to iterative and Incremental design process. Findings are based on literature and state of the art.

2) How agile methods are currently seen in target company?

The hardware team has adopted some agile practices from Scrum and has been using those for over a year. The survey and interviews are used to take the stock out of the current situation with agile methodologies.

3) How to get started with agile development?

Best practices are studied from the literature and findings from observation, survey and interviews are analyzed to form best practices for the hardware team. Global pandemic restricted the on-site working, observation, and interviews. Company had strict on-site access limitations, so the observation of the current Scrum workflow was done mainly remote, and the interviews were done via Teams. Company had been using Jira as an issue tracking software. That ruled out the discussion and comparison of the different issue tracking tools.

1.3 Thesis structure

This thesis is divided into four chapters: theoretical background, methodology, results, and conclusions. Theoretical background is split into eight sub chapters: 2.1 Agile development, 2.2 Scrum, 2.3 Large Scale Scrum (LeSS), 2.4 Kanban, 2.5 Scrumban, 2.6 Traditional product development, 2.7 Agile methods in non-software development and 2.8 JIRA software. Agile development is a brief history of the subject and common agile methods. Scrum introduces the most used agile framework to date. Large Scale Scrum (LeSS) introduces a scaled agile methodology for multiple Scrum Teams working on a same product. Kanban introduces a different approach to agile development using Kanban board and restricting work in progress. Scrumban combines best practices from Scrum framework and Kanban. Traditional project management, in this context a stategate and a waterfall model, are introduced to compare plan-driven development to iterative and Incremental development. Agile methods in non-software development introduces agile development in other fields of engineering. JIRA software gives an overview of issue and project tracking software used in a target company.

Chapter three includes the methodology of the thesis. There is introduced research strategy and design which includes research questions and research path. Also, research techniques and procedures are presented here. Chapter four includes quantitative and qualitative results as well as best practices for implementing agile methods at target company. Chapter five is conclusions which gives a reader an analysis of the key results, research implications, assessment of the approach and design and future research needs.

2. THEORETICAL BACKGROUND

This chapter provides theory and background of the thesis subject. Theoretical background begins with agile development in general. Next is introduced the selected agile frameworks and methodologies: Scrum, LeSS, Kanban and Scrumban. Traditional product development and agile methods in non-software development are introduced. Last section provides overlook and introduction of issue and project tracking software Jira.

2.1 Agile Development

Agile product development methods are a rising trend in industrial applications and are widely used especially in software development (Becerril et al., 2018; Bohmer et al., 2017; Feldmuller, 2018; Ovesen, 2012). In 2001 seventeen well-known software developers signed *the Manifesto for Agile Software Development* which declares four values and 12 principles (Beck et al., 2001). The four values are:

- Individuals and interactions over processes and tools
- *Working software (product)* over comprehensive documentation
- *Customer collaboration* over contract negotiation
- *Responding to change* over following a plan

While there are items both in left and right, agile methods proponents value the items on the left more (Beck et al., 2001). The 12 principles stated in the manifesto are (Beck et al., 2001):

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

- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software (product) is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers (engineers), and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity, the art of maximizing the amount of work not done, is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

As discussed earlier the manifesto is done by software developers to software developers but there is a lot of discussion about how to integrate this type of thinking to a complex mechatronic product development (Becerril et al., 2018; Bohmer et al., 2017; Goevert et al., 2017).

Agile Development is an umbrella term for various but somewhat similar methods that are based in an iterative and Incremental approach to development (Ovesen, 2012). Some of widely used and known agile methods are (Ashmore & Runyan, 2014):

- Scrum
- Dynamic Systems Development Method (DSDM)
- Lean development
- Kanban Method

Other agile methods worth of mention is:

- Extreme Programming
- Crystal
- Feature-Driven Development

For this thesis the Scrum and Kanban method was chosen as a reference since these has been already studied from the aspect of hardware and mechanical development. (Ovesen, 2012; Reynisdottir, 2013)

Based on the conference paper by Klein et al. (2016), agile procedures have widely spread among and beyond software projects (Komus, 2012). It is proved that agile methods have their advantages in efforts, flexibility and customer satisfaction and affects to reduction of lead times, reworks and engineering costs (Middleton & Joyce, 2012; Müller & Padberg, 2002; Sjoberg et al., 2012). In a big picture, Klein et al. (2016) states that agile methods focus on the people around the project, transparency, customer needs as well as periodic and continuous delivery (Abrahamsson et al., 2002; Sommerville, 2016).

Abrahamsson et al. (2002) states that software development is agile when it is incremental (small software releases with rapid cycle), cooperative (customer and developers working constantly together with close communication), straightforward (the method itself is easy to learn and to modify, well documented), and adaptive (able to make last moment changes).

Jim Highsmith, former Agile Practice Director in Cutter Consortium states in the foreword of the book *Agile Estimating and Planning* that continuous learning and adaptation are core to agile development (Cohn, 2005). It is also stated that core of agile project management is managing the impact of complexity and uncertainty on a project. Working on these core statements in everyday project work can lead to shorter cycle and product delivery times which are crucial in today's highly competitive and rapid industry. (Dyba & Dingsoyr, 2015) In this thesis it is discussed about agile development methods in a new product development context.

2.2 Scrum

Scrum was first introduced in 1986 by Professor Hirotaka Takeuchi and Emeritus Professor Ikujiro Nonaka in their Harvard Business Review article "The New Product Development Game". As stated in the article the holistic approach has six characterizes: built-in instability, self-organizing teams, overlapping development phases, multilearning, subtle control, and organizational transfer of learning. (Takeuchi & Nonaka, 1986)

Scrum framework, widely used in software development, was developed by Jeff Sutherland and Ken Schwaber in the 90s and the initial version was first introduced in an OOPSLA (Object-Oriented Programming, Systems, Languages & Applications), an annual ACM (Association for Computing Machinery) research conference in 1995 (Schwaber, 1997). Since then the Scrum approach has evolved from the field of software to systems development (Ovesen, 2012). A fifteenth State of Agile report shows that Scrum is most popular agile method used. The survey received a total of 4182 responses (State of Agile, 2021).

As stated in the Scrum Guide, Scrum is an iterative and Incremental approach to optimize predictability and to control risk (Schwaber & Sutherland, 2020). Transparency is one element which makes Scrum effective. Transparency enables inspection and emergent process and work must be visible for those who does the work and for those who receive it. (Schwaber & Sutherland, 2020)

Scrum process is repetitive process which aims to improve the product in every cycle. In the Figure 1 is represented the Scrum framework.

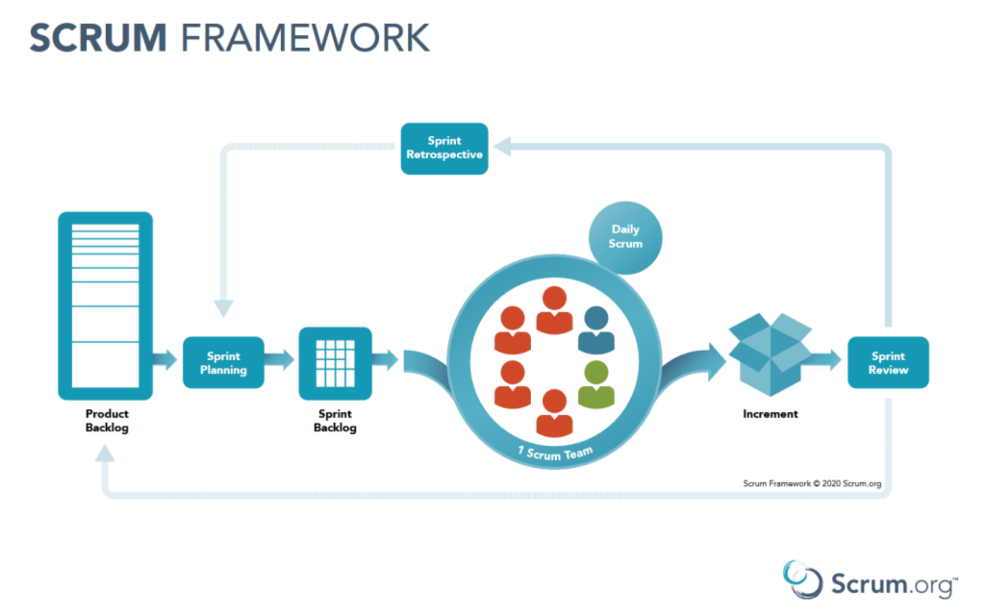

Figure 1. *Scrum framework* (Scrum.org, 2021)

There are three key elements which makes Scrum process effective. These are: *Scrum Team*, *events,* and *artifacts.* The following explanation of the Scrum process is from the official Scrum Guide 2020. (Schwaber & Sutherland, 2020)

2.2.1 Scrum roles

The **Scrum Team** is a small team of people, and it consists of one *Scrum Master*, one *Product Owner* and *Developers*. It is a unit of professionals with no hierarchies or sub teams and shares the same *product goal*. Scrum Team is cross-functional, so the members have all the skills necessary to create value for the product in each *Sprint.* (Schwaber & Sutherland, 2020)

Scrum Master is a servant leader of the Scrum Team. Main responsibility is to facilitate Scrum events and to make sure that Scrum is established as it should be by helping everyone understand Scrum theory and practice. In the Figure 2 can be seen Scrum Master's principal responsibilities. (Schwaber & Sutherland, 2020)

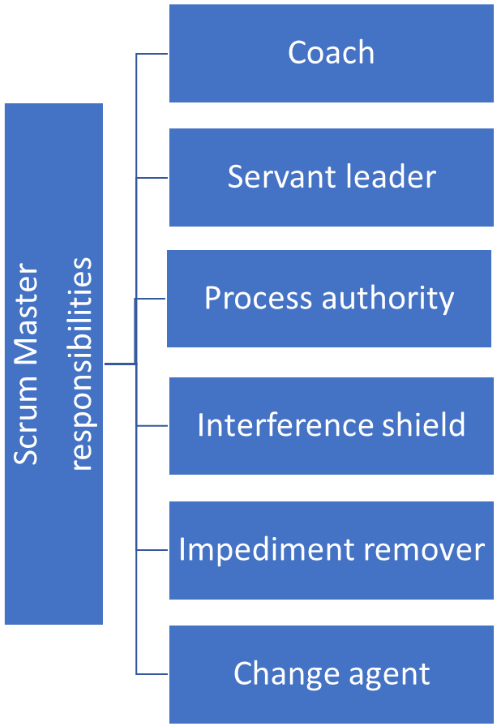

Figure 2. *Scrum Master responsibilities* (adopted from Rubin, 2012, Fig. 10.1.).

Scrum Master serves the Scrum Team by coaching the team members in self-management and cross-functionality and removes any impediments that can slow down Scrum Team's progress. Scrum Master serves Product Owner by helping the Scrum Team to properly understand the Product Backlog items (Schwaber & Sutherland, 2020)

Product Owner is part of a Scrum Team and interacts simultaneously between stakeholders and Scrum Team. Product Owner must understand stakeholders, customers and users' needs as well as communicate to the development team what to build and which order. (Rubin, 2012, Ch 9) In the Figure 3 is shown the interaction between Scrum Team and stakeholders.

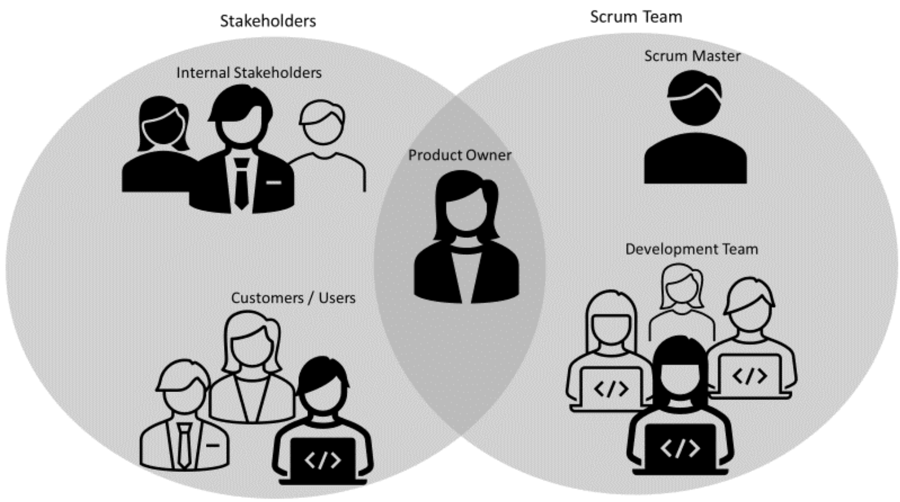

Figure 3. *Product Owner interacts two directions simultaneously* (adopted from Rubin, 2012, Fig. 9.1.).

Product Owner is accountable for maximizing the value of the product resulting from the work of the Scrum Team. Product Owner has right to manage *Product Backlog*, give work for the development team, renegotiate, and cancel the Sprint and release *Increments.* Effective Product Backlog management includes (Schwaber & Sutherland, 2020):

- Developing and explicitly communicating the product goal
- Creating and clearly communicating Product Backlog items
- Ordering Product Backlog items
- Ensuring that Product Backlog transparent, visible, and understood

Product Owner is a full-time role with significant responsibilities. It can be compared to traditional project manager in the matter of the large scope of work. Principal responsibilities among maintaining the Product Backlog are:

- Managing economics
- Participate in planning
- Define acceptance criteria and verify that they are met
- Collaborate with the development team
- Collaborate with the stake holders

Developers in a Scrum Team creates every aspect of usable *Increment* in each Sprint in other word adds value to the product. Developers are accountable for (Schwaber & Sutherland, 2020):

- Creating a plan for the Sprint and Sprint Backlog

- Ensuring quality work by adhering to a *definition of done*
- Adapting plan each day toward Sprint goal

2.2.2 Scrum events

Scrum events are crucial to successfully establish Scrum process. All events take place during one Sprint. To reduce complexity, it is preferred that all events take place at the same time and place. These events are specifically designed to enable the transparency required. (Schwaber & Sutherland, 2020)

Sprints are containers for all events and the heartbeat of Scrum. Typical Sprint is a fixed length of two to four weeks. The goal in every Sprint is to develop a functionality or enhance the product to produce new Increments towards the product goal. Sprint starts with a *Sprint Planning* and continues with *Daily Scrums* and ends in *Sprint Review* and *Sprint Retrospective*. (Schwaber & Sutherland, 2020)

During **Sprint Planning** the Product Owner offers the Sprint goal, development team decides how much work is brought into the Sprint and creates Sprint Backlog items. Scrum Master facilitates and interacts between Product Owner and development team. (Schwaber & Sutherland, 2020)

Daily Scrums are short daily meetings (maximum of 15 minutes) where development team review the progress and updates Sprint Backlog. Scrum Master is responsible to teach how to run the Daily Scrum. If Scrum Master and Product Owner are actively working on items in the Sprint Backlog, they participate in Daily Scrums as developers. Benefits of Daily Scrums are that these improve communication, identifies impediments, and makes progress more transparent. (Schwaber & Sutherland, 2020)

Sprint Review is an event where the outcome of the Sprint is reviewed. Attendees are Product Owner who explains which Product Backlog items is done, development team which demonstrates the results of the Sprint, Scrum Master who facilitates event and key stakeholders which provides feedback. Product Backlog can also be adjusted to meet new opportunities which have appeared during the Sprint. Sprint Review length is maximum of four hours long for one-month Sprint. (Schwaber & Sutherland, 2020)

During **Sprint Retrospective** Scrum Team reviews last Sprint and inspects what can be improved. It is an event where developers can identify ways to continuously improve themselves, processes, and development practices. The most impactful improvements can also be added to Sprint Backlog. Length of the Sprint Retrospective is timeboxed to a maximum of three hours for one-month Sprint. (Schwaber & Sutherland, 2020)

2.2.3 Scrum artifacts

Scrum artifacts are designed to maximize transparency of key elements and represent work or value provided to product. Artifacts are called *Product Backlog, Sprint Backlog,* and *Increment.* All three contains an own commitment towards transparency and progress which are *product goal, Sprint goal* and *definition of done*. (Schwaber & Sutherland, 2020)

Product Backlog consists of all items needed to improve the product e.g., product requirements, functionalities, deliverables, technology issues and risks. Backlog items are prioritized and managed by Product Owner. Product goal is a long-term target for the Scrum Team to fulfil and it is in the Product Backlog. (Schwaber & Sutherland, 2020)

Sprint Backlog consists of all items selected from the Product Backlog and a plan for delivering the Increment for the Sprint. It is a plan by and for developers and it must be highly visible, real-time picture of the work, so the developers can proceed towards the Sprint goal. During Sprint Planning the Sprint goal is created and added to the Sprint Backlog. Sprint goal is one objective for the Sprint and by doing the exact work in the Sprint Backlog the Sprint goal is achieved. (Schwaber & Sutherland, 2020)

Increment is an outcome of each Sprint and is a part of the whole product. One Sprint can contain multiple Increments and to provide value, the Increment must be usable. The commitment for the Increment is a definition of done which is an agreement that defines quality of development team's work. As soon as a Product Backlog item meets the definition of done, an Increment is made. If a Product Backlog item does not meet the definition of done it is returned to Product Backlog for future consideration. Definition of done can be for example an organizational standard and it must be followed as a minimum. It is an agreement that defines quality of development team's work. (Schwaber & Sutherland, 2020)

In conclusion, Product Owner manages a Product Backlog. The Scrum Team turns a selection of the Product Backlog into an Increment of value during a Sprint. The Scrum Team and its Stakeholders inspect and reviews the results and adjust those for the next Sprint. (Schwaber & Sutherland, 2020)

2.3 Large Scale Scrum (LeSS)

LeSS Framework was created in 2005 when Bas Vodde and Craig Larman worked together in Nokia Siemens Networks (Less.works, 2021a). Where original Scrum Framework is for one Scrum Team, the LeSS Framework is scaled agile method for multiple teams working on the same product. There are two variations of LeSS Framework:

smaller LeSS for two to eight development teams and LeSS Huge for more than eight development teams (Vodde & Larman, 2016, Ch 2). In this thesis the scope is in smaller LeSS Framework. In the Figure 4 is shown the LeSS Framework.

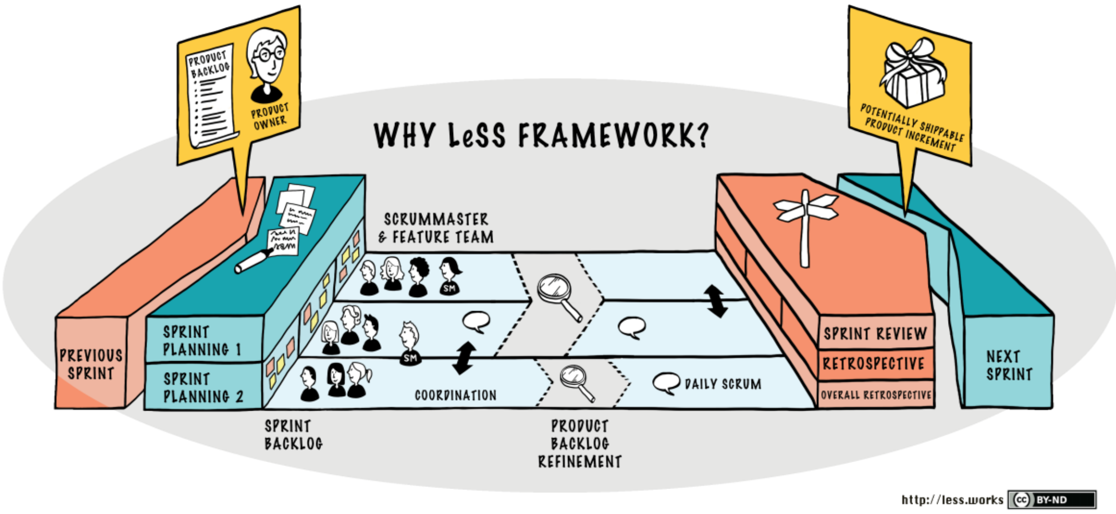

Figure 4. *LeSS Framework is like one-team Scrum* (Less.works, 2021c).

LeSS is very similar to one Scrum Team. Both frameworks have single Product Backlog, Product Owner, and definition of done. Despite the amount of Scrum Teams, all of them work together in order to deliver common, shippable product at the end of each Sprint (O'Connor, 2021). Scrum Teams in one-team Scrum and LeSS are both cross-functional with no single specialist team (Less.works, 2021c).

Although LeSS and original Scrum shares similarities and principles there are some differences between frameworks. Sprint Planning in one-team Scrum is held once before every Sprint and the whole Scrum Team is present. As stated in chapter 2.2 during the planning session the Product Owner offers the Sprint goal, and the development team decides what items to brought to Sprint Backlog from Product Backlog. In LeSS the planning is split into two separate sessions called Sprint Planning 1 and Sprint Planning 2. The former is a higher-level planning session to which are present Product Owner, Scrum Master, and people from all teams. The target is to let members self-manage to decide their division of Product Backlog items and discuss opportunities to find shared work and cooperate. Sprint Planning 2 is held parallel by each team and the scope is to split divisions of Product Backlog from session 1 to smaller, executable form which can be done during the Sprint. (Less.works, 2021c)

Product Backlog Refinement is done per team for the items they are likely going to implement in the coming Sprints. Multi-team Product Backlog Refinements are recommended since those increase shared understanding and exploit coordination opportunities. Product Backlog Refinements are done by Product Owner together with Scrum Teams. (Vodde & Larman, 2016)

Sprint Reviews are attended by Product Owner, participants from all teams and relevant customers, users, and stakeholders. Teams presents here their product Increments and items they have developed. The idea is the same as in one-team Scrum but instead multiple teams are present. (Less.works, 2021c)

Overall Retrospective is new meeting not found in one-team Scrum. Its purpose is to define improvements to overall process and maximum duration is 45 minutes per week of Sprint. Meeting includes Product Owner, Scrum Masters, and rotating representatives from each Scrum Team. (Less.works, 2021c)

2.4 Kanban

Kanban stands for Japanese words kan, meaning visual and ban, meaning card. Together it means visual card, which is a Kanban system which is set up to track the work in progress (Hammarberg & Sunden, 2014). Kanban origins from lean production developed in Toyota as a scheduling system for JIT (just-in-time) manufacturing in the Toyota Production Systems in the 1940s (Womack et al., 1990).

Kanban has three basic principles which are the foundation for the method. These are:

- Visualize
- Limit work in process
- Manage flow

Later, these basic principles have evolved to six practices which are referred as the core practices of the Kanban. The remaining are:

- Make process policies explicit
- Implement feedback loops
- Improve collaboratively, evolve experimentally

Starting from the top, visualize, meaning that by visualizing the workflow using a whiteboard and sticky notes can illustrate the team how the work really works. It could have a huge impact just by making the information visible rather than hide it in electronic databases. The Kanban board can be a physical one, like a white board or a white wall, in a use of co-located team or a virtual board which is used by a global or a remote working team. In a Figure 5 is an example of a Kanban board with working items and columns to visualize the workflow.

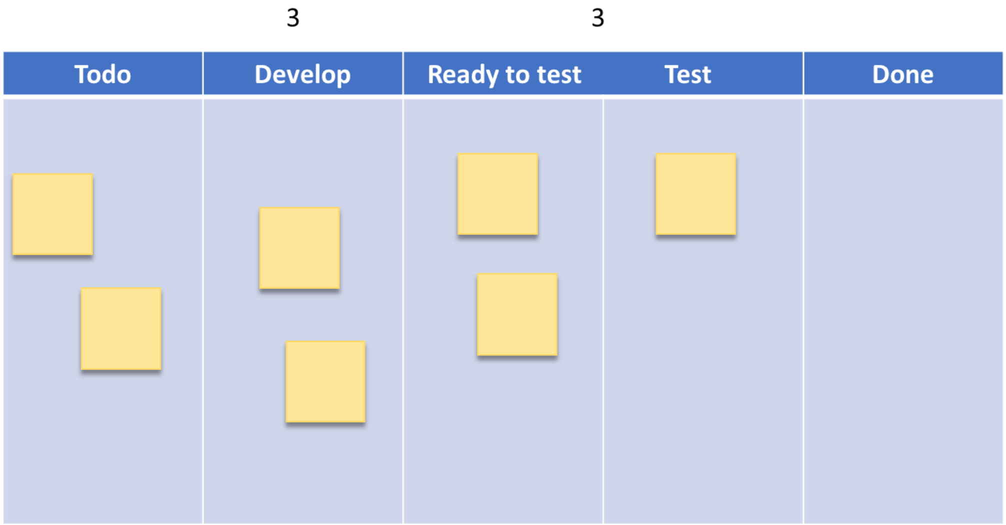

Figure 5. *Example of a Kanban board* (adopted from Hammarberg & Sunden, 2014).

Working items can be as simple as a sticky note or an index card and it must be movable on the board to maintain the work progress. It contains all the information how to work with a working item. A few useful information to work with working items are a clear *description* which tells what the work is about, *deadlines, tracking IDs* used in electric databases and *type of work* which helps prioritizing items against each other. Using avatars or personalized magnets helps team to know who is working on what.

As seen in the Figure 5 some columns have numbers after the column name. These numbers limit the WIP for the phase meaning that the number is the maximum amount of the tickets accepted. The second practice, limiting work in process (WIP), means that by limiting parallel work can speed up the ongoing work done. This refers to Little's law so that the more the WIP, the longer the cycle time is. The goal is to improve the flow by limiting WIP.

Third practice, managing flow, is about monitoring and measuring how the workflows. There are several different approaches to manage flow: limiting WIP as stated previously, reducing waiting times, removing blockers, avoiding rework, using cross-functional teams, and using timeboxes.

As in Scrum, also Kanban can have daily stand-ups which can clarify the process. The idea behind daily stand-ups is to keep it short, focused, and regular where the team can daily check the Kanban board and review what they are working on.

Kanban is a popular agile method used in software development. In this thesis Kanban is one of the possible solutions to improve remote teams' collaboration during product development. It also makes the process more transparent which is important especially when people are working from home and face-to-face talks are non-existent.

2.5 Scrumban

Scrumban was first introduced by Corey Ladas in his paper *Scrum-ban* in 2008 which evolved into a book *Scrumban* in 2009 (Ladas, 2009). Scrumban is a hybrid of Scrum and Kanban (Ladas, 2009; Reddy, 2015; Stellman, 2020). Main differences between Scrum and Kanban are shown in Table 1.

Teams and organizations that have already been using Scrum as a default Agile approach have come across a need for more evolved development framework (Ladas, 2009). As Reddy (2015) states in his book, Scrumban have evolved to become a family of principles and practices that can help teams and organizations to accelerate their transition to Scrum as well as help them to overcome challenges that Scrum causes. Key idea for implementing the Scrumban is to maintain core elements of both Scrum and Kanban – combining the structure of Scrum with the flow-based methods of Kanban. It must preserve the empirical process control of Scrum as well as implement a Kanban pull system (Stellman, 2020). To make a success in Scrumban one must understand both practices discussed in chapters 2.3 and 2.4.

Table 1. *Differences between Scrum and Kanban* (Rehkopf, 2021b)

	Scrum	**Kanban**
Cadence	Regular fixed length Sprints (i.e., from 2 to 4 weeks)	Continuous flow
Releases	At the end of each Sprint	Continuous delivery
Roles	Product Owner, Scrum Master, development team	No required roles
Key metrics	Velocity	Cycle time, throughput, WIP
Change philosophy	No changes during the Sprint	Changes can happen anytime

Common misconceptions are that Scrumban is a tool to organize work and make it more efficient or that Scrumban is Scrum without iterations combined with Kanban board

(Stellman, 2020). As discussed in chapters 2.2 and 2.4 Scrum is a framework for a project management and Kanban is a method to improve process. Ladas, (2009) states in his paper that:

“At this point, we are now operating a simple Kanban pull system. We still have our time-boxed iteration and planning cycle, so perhaps we might call such a thing a Scrumban system!”

2.6 Traditional product development

Traditional product development models such as Waterfall model and Stage-Gate model follows linear progression where previous phase is done before moving to the next phase. This type of linear progression, later known as the Waterfall model, is often cited in the articles by Dr. Winston W. Royce (Royce, 1987). Unmodified version of the Waterfall model is illustrated in Figure 6.

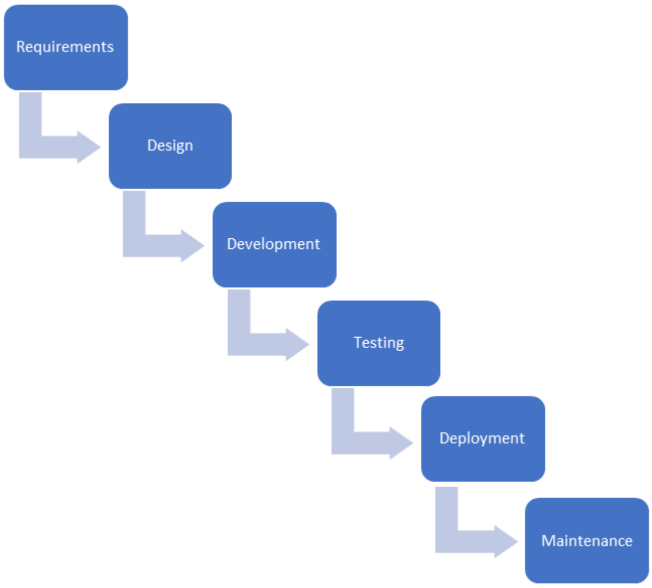

Figure 6. *Unmodified Waterfall model.* (adopted from Royce, 1987).

2.7 Agile Methods in non-Software Development

It is stated that large established companies do not yet use agile development as state of the art (Goevert et al., 2017) and it is new topic in manufacturing industry (Becerril et al., 2018). Agile methods have great potential to become more common in mechatronic engineering and product development. Feldmuller (2018) states in his paper that agile practices are discussed for mechatronic product development and is usually mixed with stage-gate process to combine best practices from both approaches to make a hybrid approach.

Because of the differences between software development and new physical product development, the transformation from traditional methods to agile methods is rarely straightforward. Table 2 shows differences between software development and hardware development.

Table 2. *Differences between software and hardware development (Alten, 2019)*

Attribute	Hardware (Mechanical and Electrical	Software
General Process	• Design, Refine, Prototype, Test	• Create Test, Implement Code, Refactor, Repeat
Producibility	• Requires drawing, quotes, build, assembly, and testing • May take days to months	• Requires compiling code to create an executable / image • Takes on the order of minutes to hours
Component and Testing Costs	• Prototype cost can be high • In addition to labour, testing costs typically include custom test and production equipment, plus assembly and packaging	• Little to no parts or distribution costs • Testing costs is primarily labor
Modularity	• High degree of integration on most products to minimize size, weight, and therefore costs • Modularity usually designed in only for known changes / upgrades	• High modularity is encouraged as a best practise
Cost of Change	• Most changes become exponentially more difficult and expensive to implement the later in a project they occur (especially when they impact production or test equipment)	• If well-architected, the cost of change is mostly stable over a project • Labor cost only
Team Composition	• Team members are likely to be specialized in some area • Skills often not interchangeable	• Many skills are interchangeable

A case study shows that it is possible to follow the Scrum framework in new product development with minor tweaks for rules and criteria, yet still following the principles successfully (Reynisdottir, 2013). On the other hand it is stated that benefits of agile methods within hardware development is minimal. Thompson (2011) states that: "historically, the view was, 'Let's make the hardware first, then develop the software, and get it to work well". Sceptics in the industry have not seen the benefits for agile methods within hardware product development since circuit boards and chips are expensive and hard to change late in the product development cycle. (Gustavsson & Rönnlund, 2013)

2.8 JIRA software

Jira is a family of products built to help all types of teams manage their work. Jira platform includes products and apps which help teams plan, assign, track, report, and manage work. The products built on the platform are Jira Software, Jira Service Management and Jira Core. Jira Align is an enterprise scale solution for agile planning. (Atlassian, 2021b)

Jira Software (later just Jira) is an issue and project tracking software from Australian company called Atlassian founded by Mike Cannon-Brookes and Scott Farquhar (Atlassian, 2021b, 2021a). The company is also known for its collaborative tools Confluence and Trello which can be integrated to Jira (Atlassian, 2021d). Jira is one of the most used agile project management tools available and (State of Agile, 2020) report claims that 67 % of the respondents were using Jira. In the same report it is stated that 78 % of the respondents would recommend Atlassian's Jira as a project management tool.

Jira supports different agile methodologies as well as mixed methodologies. Scrum, Kanban and Scrumban is discussed in depth in chapters 2.2, 2.3 and 2.5. Kanplan is Kanban with separate backlog. Instead of using Kanban boards To Do -column as a backlog, Kanplan introduces separate backlog for items. As Kanban is a pull, rather than push method, the separate Product Backlog keeps the Kanban board cleaner and simpler allowing development team to pull items from separate backlog to Kanban boards To Do -column (see Figure 5). (Atlassian, 2021e)

Although Jira is primarily used in software companies it is a useful project management tool for other industries as well. It increases trackability and transparency of the project which benefits both managers and engineers. (Atlassian, 2019)

Jira Scrum workflow can be seen in Figure 7. For managers, Scrum Masters and Product Owners, Jira supports whole workflow from project configuration to wrapping up the project. Jira users which in this context are developers and engineers are mainly involved in Sprints.

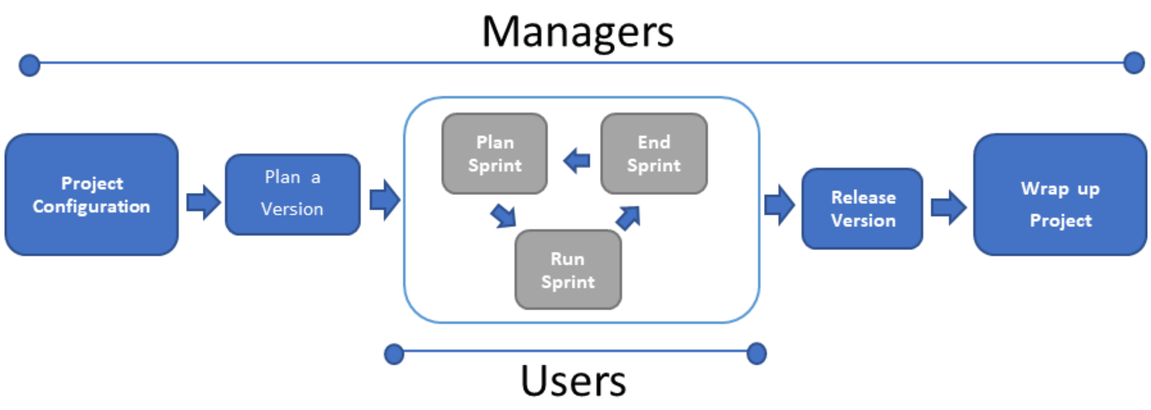

Figure 7. *Jira Software workflow* (adopted from Atlassian, 2021c)

Before starting agile project, one usually chooses agile Scrum development or Kanban development which both are fully supported in the tool (Atlassian, 2021f). In project configuration phase manager adds users for the project and configures estimation and tracking. This is essential for understanding how much work the team has and how much can be done (Atlassian, 2021f). Estimation (used for measuring the amount of work in the backlog and calculating velocity) can differ from tracking (used for measuring burndown of hours used during the Sprint) in terms of units (Atlassian, 2021f). Estimating unit for issues can be for example a story point which is abstract measure of effort required to implement it, or simply a time-based measurement like an hour or a workday.

Creating a backlog for the project is essential. As discussed in the chapter 2.2 backlog consist of a list of tasks that represents the work in a project. In Jira individual pieces of work that must be completed are called issues. Rehkopf (2021a) states in Atlassian Agile Coach tutorial about issues that depending how the team uses Jira, an issue could represent a project task, a helpdesk ticket, and a leave request form. In mechatronic product development an issue could be a product requirement which is split into tasks that eventually fulfils the requirement. Ranking of the issues in order of priority is essential and allows the development team to focus on more crucial tasks (Rehkopf, 2021a). In the Figure 8 is an overview of the Scrum backlog with separate Sprint Backlog.

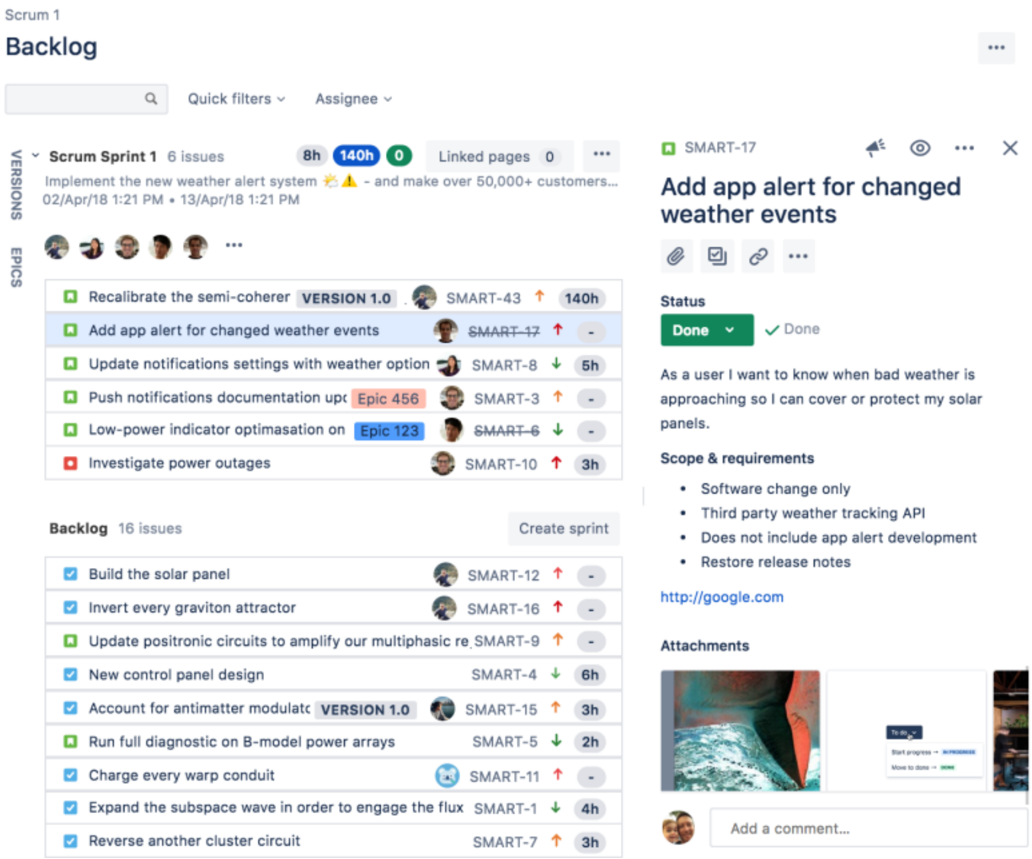

Figure 8. *An overview of Jira Scrum backlog and Sprint Backlog* (Atlassian, 2021f).

Task management and structuring a project in Jira can be challenging when adopting agile methodologies to product development. When this is done properly and customized for the Scrum Team it can be very beneficial. One example how to divide project into smaller sections are shown in the Table 3.

Table 3. *Structuring backlog items from large body of work to single task.*

PRODUCT BACK-LOG ITEM	DESCRIPTION	SCOPE	TIME-BOX
EPIC	Large body of work which can be split into several smaller stories	One or more Scrum Teams	Multiple Sprints
STORY	Short requirements or requests which can be split into smaller tasks	One Scrum Team	One Sprint
TASK	A piece of work to fulfil a story	One responsible Scrum Team member	A few days

Epics are larger bodies of work which can be split into smaller user or technical stories. Epic ties together stories and tasks that forms a concrete target. Usually, Epics are built from multiple Sprints. People who commit for the Epic can be from multiple Scrum Teams. Stories can be user stories or technical stories. The difference between these two are that user stories are from the user (e.g., customer) point of view and technical stories are not visible to the user, like technical requirement. Stories add value to the product. Tasks are concrete work to fulfil a story. Usually, a task is pointed for one person and a story can have multiple tasks. Time-box with stories is one Sprint and with single task, a few days. This must keep in mind when estimating the backlog items.

During Sprint Planning and Sprint cycle the users are involved as well. As discussed in chapter 2.2 Sprint Planning is a part of a Scrum workflow and important part of the time boxed Sprint. Sprint Planning is done together with the Product Owner, Scrum Master, and development team. Before starting a Sprint in Jira, a Sprint Backlog must be created in a Sprint Planning.

During Sprints the development team works with tasks that are in the Sprint Backlog. Tasks are always assigned to the developer who is working with it. Managers and team leads can view burndown charts to check how the team is progressing in the current Sprint. A burndown chart compares work required to complete the Sprint with actual work completed by development team (Atlassian, 2021f).

3. METHODOLOGY

This thesis follows an abductive approach which is also known as abductive reasoning (Tuomi & Sarajärvi, 2002, p. 99). In Table 4 is introduced key elements of abductive approach to this study. It is stated that instead of moving from theory to data as in deductive approach or data to theory as in inductive approach, an abductive approach combines both moving back and forth between theory and data (Suddaby, 2006). In this approach the analysis of the data is not directly based on theory. Although the data can be linked to theory and this way the theory can give explanations and confirmation to support the data and findings. (Aaltola & Valli, 2001, pp. 133-157)

Table 4. *Abduction as an approach for research* (Saunders et al., 2019).

	ABDUCTION
LOGIC	Known premises are used to generate testable conclusion
GENERALISABILITY	Generalising from the interactions between the specific and the general
USE OF DATA	Data collection is used to explore a phenomenon, identify themes and patterns, locate these in a conceptual framework and test this through subsequent data collection and so forth
THEORY	Theory generation or modification; incorporating existing theory where appropriate, to build new theory or modify existing theory

3.1 Research strategy and design

Research strategy and design is built around a case study which purpose is to give answers to research questions:

1. How agile methods can be used in hardware development?
2. How agile methods are currently seen in the target company?
3. How to get started with agile development?

Research questions and how these are linked to research methods and results can be seen in Figure 9.

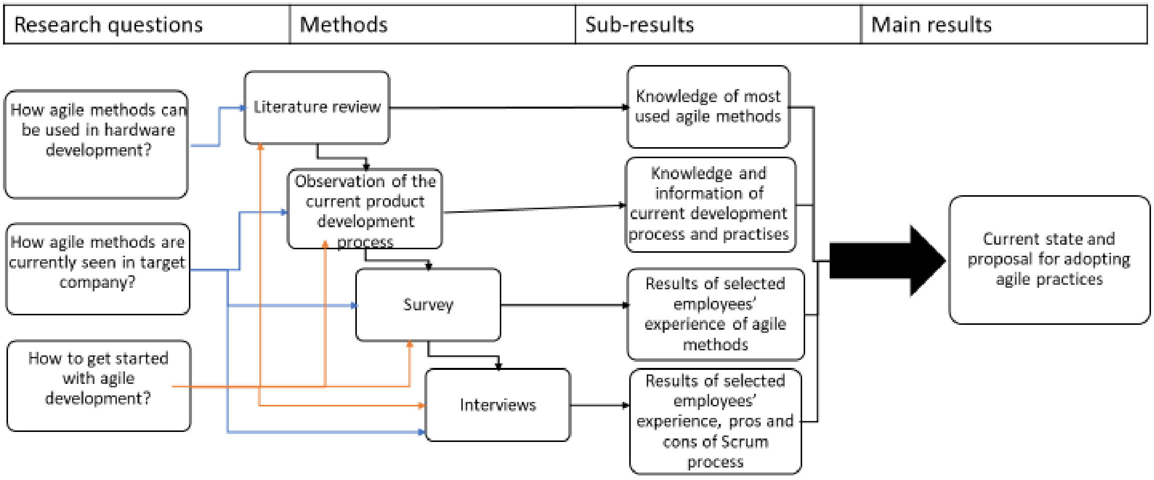

Figure 9. *Research path of the thesis.*

Research path starts with three research questions. These questions are linked to research methods which are used to gather knowledge and information of the subject. Literature was mainly used to study the first research question. For the second research question an observation, a survey and interviews were used to gather knowledge. For the third research question all the methods were used to form a starting point for the target company.

A case study is an in-depth research into a topic within its real-life setting (Yin, 2017). As Saunders et al. (2019) states "case" in a case study can be a person, a group, an organization, an association, a change process, an event, and other types of case subjects. In this thesis "case" is a group, a hardware team which includes electronics engineers and mechanical engineers.

Under the more generalized category of case study exist several subdivisions, each of which is custom selected for use depending upon the goals and/or objectives of the investigator, e.g. descriptive, explanatory and exploratory (Becker et al., 2012). This thesis follows exploratory (also known as pilot) case study which basic function is to help identify questions and select types of measurement prior to main investigation. This thesis is a pre-study of agile methods in hardware development and the main investigation would be a real-life piloting of the suggested agile practices.

In this thesis the data collection is done in a single phase through survey and interviews. It is said that collecting qualitative and quantitative data and comparing how these data sets support one another, a concurrent triangulation design is used (Saunders et al., 2019). Triangulation involves using multiple sources of data and method of collection to confirm the validity, the credibility, and the authenticity of research data (Saunders et al., 2019). This study uses three different research methods to collect data: survey, interviews, and observation as well as mixed methods, qualitative and quantitative, to collect data. In the survey it is used embedded mixed methods, where one methodology supports the other (Creswell & Plano Clark, 2011). This means that some qualitative, open-ended questions, are included in the survey which on the other hand is a quantitative method with closed-ended questions (Saunders et al., 2019). It is said that combining quantitative and qualitative methods can provide better opportunities to answer research questions (Tashakkori & Teddlie, 2010). This is true since in this study much of the data and background information is collected through survey and interviews. Combining those two methods can give deeper information than using only other method.

3.2 Research technique and procedures

Research follows the research path in the Figure 9. Literature review gives theoretical background of the subject. Observation on the other hand gives insight into current product development process. With the information and knowledge from observation the survey questions are prepared. Combined with the theoretical background the survey has its final form. Survey results gives a direction for the interview flow and questions. Interview structure and questions is formed with the data from survey and observation together with knowledge and information from theory part of the thesis.

3.2.1 Observation of the Scrum workflow

Classic observation roles in Figure 10 shows relationship between roles and researcher's activity and identity. In this thesis the closest role is the observer as participant. This means that observation in this role is mostly restricted to being present at events and meetings. Participation in the activity is low. People in the meeting are also aware that researcher is conducting a study and knows his or her identity. (Saunders et al., 2019)

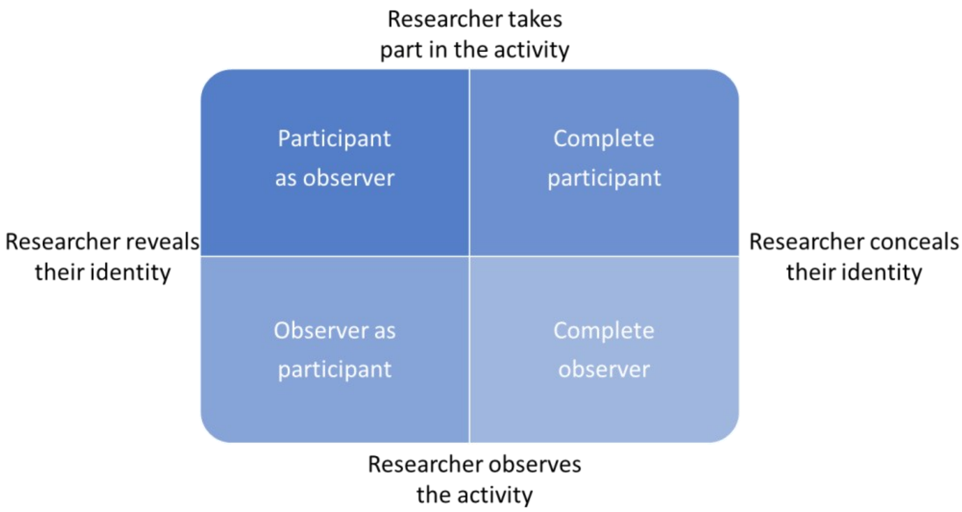

Figure 10. *Classic observation roles* (adopted from Saunders et al., 2019, Fig 9.3).

Observation of the Scrum process was done by attending to Daily Scrum meetings and observing usage of JIRA. Daily Scrums were held from Monday to Thursday at the same time at 13:00 (though some meetings were cancelled). Some but not all meetings included round table check in. During the check in every team member told what he/she was done in the past 24 hours, what he/she was going to do and were there any impediments.

3.2.2 Survey

Survey strategies using questionnaires are popular as they allow the collection of standardized data from many respondents economically, allowing easy comparison (Saunders et al., 2019). Survey was made with an online survey creator tool called Microsoft Forms. Overview of the tool used can be seen in the Figure 11. The tool had all the necessary and needed features to create a survey which included qualitative open-ended questions as well as quantitative close-ended questions and statements. The survey had five sections with different themes. Themes were chosen based on observation phase of the Scrum process. In the first section there were three background close-ended questions about agile methods in general. Second section was about agile methods at the target company. It provided a few close-ended questions and matrix question statements. Third section was about agile teams and collaboration at target company. It provided matrix question statements and open-ended question. Fourth section was about agile events and usage of JIRA at the target company. It provided a few matrix questions statements, a few close-ended questions as well as a few open-ended questions. Fifth section provided background questions about the respondent. There was also a price which was raffled among the respondents to attract more people to answer the survey.

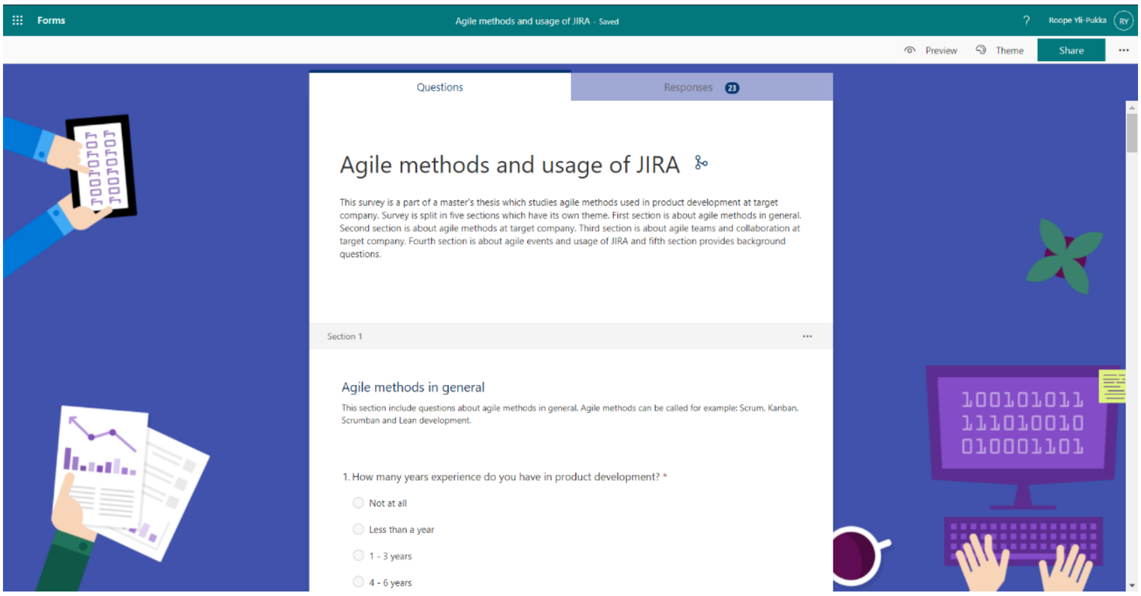

Figure 11. *Overview of the survey tool.*

One important feature was a matrix survey question type which made possible to use Likert scale in the statements. Likert scale is a commonly known five-point bipolar response asking people to indicate how much they agree or disagree a given statement (Allen & Seaman, 2007; Vehkalahti, 2014, p. 35). In this survey there were five options which to choose from: strongly disagree, disagree, neutral, agree and strongly agree.

Information about the survey was given in a project meeting via Teams and after the meeting the survey was sent via email to target audience. The recipients were people from software, electronics, validation, and mechanical engineering teams. The survey was open from March 19th to March 31st, 2021, and it was answered anonymously. People were asked a few times in daily meetings if they were already answered to the survey and reminder email was sent a few days before end of the survey.

3.2.3 Interviews

For this thesis it was chosen semi-structured theme interviews. Saunders et al. (2019) states that semi-structured are likely to be most advantageous approach to attempt to obtain data when questions are complex or open ended. Semi-structured method is useful when the order and logic of questioning may need to be varied (Saunders et al., 2019). Thesis interviews included open ended questions and the questions order varied between interviewees depending on their background and interview flow.

Interviews included three themes and one to two questions on each theme which were asked from every interviewee. The goal was to keep the interviews as a conversation and other questions were asked on the fly. The themes and theme related questions were:

1. Information flow in own team and between teams
    a. Is there enough information shared in your team?
    b. Is there enough information shared between different teams?
2. Scrum process in a current project
    a. Is Scrum as a project workflow suitable for this type of product development?
    b. How is Scrum Product Backlog built?
3. Usage of Jira in a current project
    a. How the usage of Jira has been helpful?

Interviewees were asked to participate in the interview via email and the potential interviewees were selected from software, electronics, and mechanical engineering teams. Due to global pandemic and remote working the interviews was held in Teams and was recorded for the further analyze phase.

Although the scope of the thesis is in the electronics and mechanical engineering, also software team leader was interviewed. They have been using Scrum and Jira in their daily work successfully, so it was interesting to get details about that. From electronics and mechanical engineering point of view, the scope was more on their experiences from the current Scrum process and usage of Jira.

4. RESULTS

This chapter provides results from observation, survey, and interviews. In the end is presented an agile method which can be suitable for the target company.

4.1 Observation

Current development process was observed two months from a single project point of view from beginning of February to end of March of 2021. The process can be described as a hybrid model between traditional and agile. The development follows traditional development process (2.6) in which have been added agile practices from Scrum like daily stand-ups, Product Backlog (requirements), Sprint Backlogs and Sprints.

Focus was on Daily Scrum meetings and the usage of Jira during these meetings. Daily Scrums were a possibility to share information between electronics engineering team and mechanical engineering team. In February the daily sessions were planned to be maxi mum of 15 minutes which followed the Scrum guideline. As said in chapter 2.2: "Daily Scrums are short daily meetings (maximum of 15 minutes) where development team review the progress and updates Sprint Backlog." Daily meetings often went overtime. This was mostly because of the large team size. The number of participants on each daily was approximately 20 which led to that there were a lot of Jira tasks to discuss about.

During February observation, first Sprint was started, and the sessions were hosted by a Scrum Master. There were no proper Sprint Planning or Sprint goal for the Sprint. Some of the tasks that had to be done during the Sprint, were added to the Sprint Backlog. Some tasks were added during the Sprint by the engineers which was responsible of that work and some tasks were added by team leaders and were assigned to themselves or directly to a responsible engineer. During the Sprint there were held occasionally round table check in. During that check in, everyone had a chance to tell what they were working on at that moment and did they confront any impediments.

On March, first Sprint of the project was finished. There were many tasks which was not finished during the Sprint, and these were added to the new Sprint Backlog. There was no Sprint closing, Sprint Retrospective, or new Sprint Planning. Next Sprint was started right after the other one was closed.

During March, Daily Scrum meetings were combined with design update meeting. The idea was to discuss more widely about current design in electronics and mechanics point of view. Usually, the host of the meeting was a team leader, and the scope were to share updates about the project. There were little to no Sprint Backlog updates and Jira went quickly out of date. Product Backlog included out of date requirements and contribution towards the usage of Jira was decreasing. There was also a need for wider view of the tasks and work than the four-week Sprint. An Excel sheet was made which included a weekly plan and coming milestones and releases. At that moment, there were two places which shared the same information but a bit differently. Jira still had all the tasks (although some tasks were still missing) and the weekly plan was build based on information from Jira. Weekly Excel was maintained by team leaders.

4.2 Survey results

Survey reached 46 co-workers via email. Total of 23 participated to the survey thus answer rate was 50%. 15 respondents from the total of 23 were either from electronics engineering or mechanical engineering team. This is considered a good answer rate and the survey gave plenty of data to analyze. In the most relevant questions, the answers from electronics and mechanical engineering team are examined separately and compared to the full dataset. Full survey can be found in the appendix B.

4.2.1 Background questions

Participants were either from electronics engineering, mechanical engineering, software engineering or validation engineering team. The distribution of teams is seen in the Figure 12.

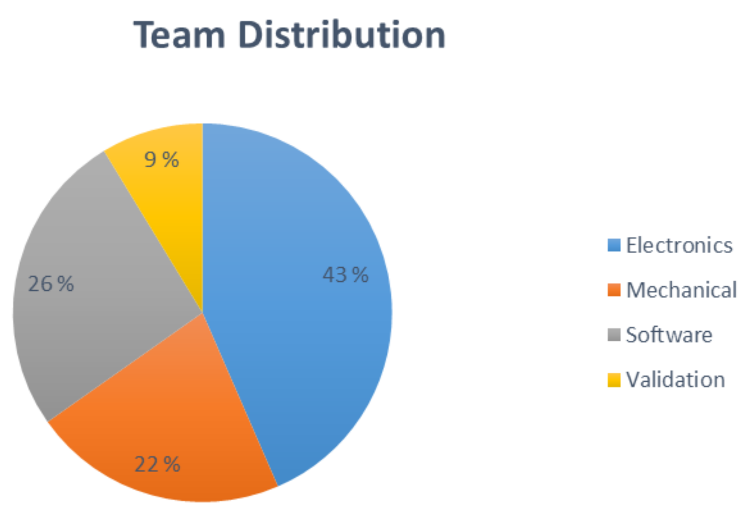

Figure 12. *Participant team distribution*

Most of the participants were from electronics engineering and mechanical engineering teams. Participants were asked about their work experience in product development and clearly most of them had over 10 years of experience. Education level was split nearly 50-50 between bachelor's degree or equivalent and master's degree or equivalent. Most of the participants were engineers or developers, two were (engineering) interns and a few were team leaders or managers. Most of the participants were born in the 1970s so they already had a long product development experience.

Some background questions were also asked about agile methods. Questions were:

- How much experience do you have with agile methods in practice?
- What agile methods have you used?

Results for the former question including all participants can be seen in Figure 13. Answers from electronics engineering (EE) and mechanical engineering (ME) team can be seen in Figure 14.

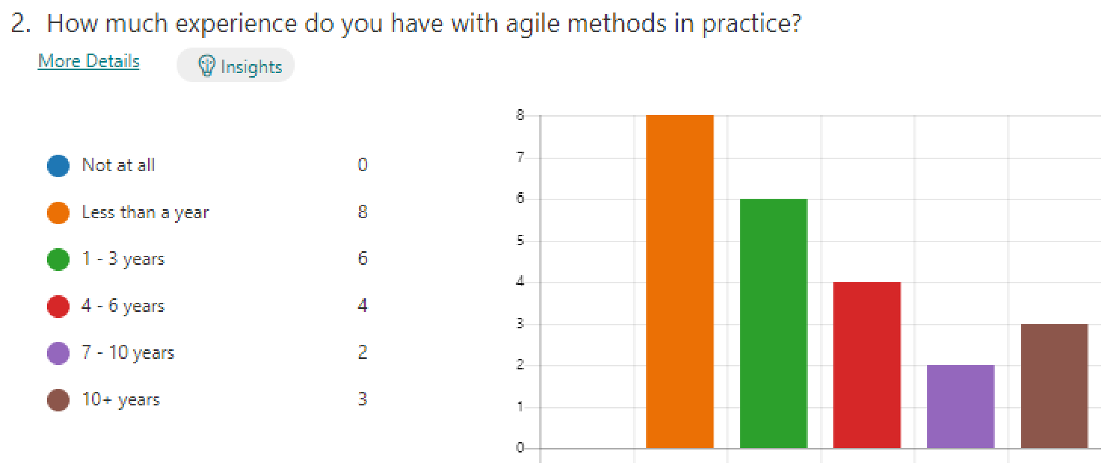

Figure 13. *Agile methods experience results including all answers.*

Although participants had a long product development experience it didn't correlate to agile methods experience. Most of the participants had experience less than a year and a few had more than ten years. If compared to team distribution, electronics and mechanical engineering teams have less experience about the agile methods than software and validation team.

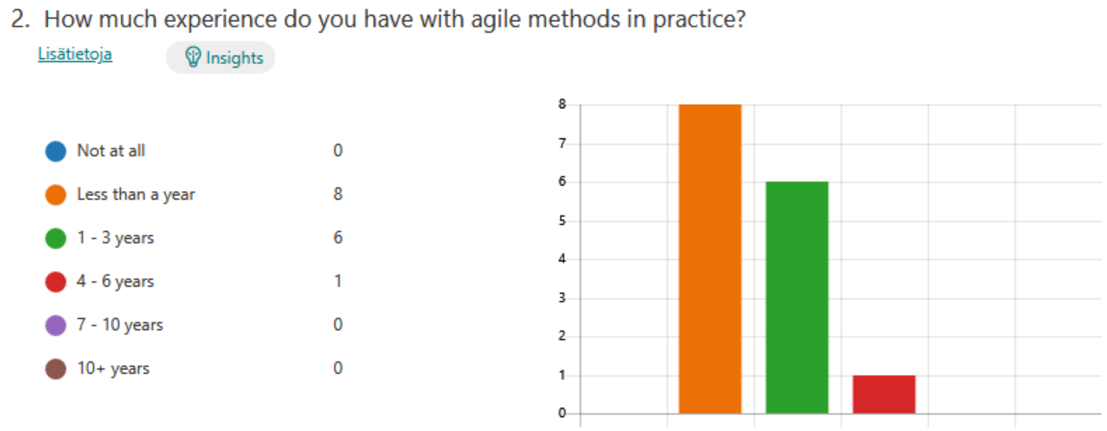

Figure 14. *Agile methods experience in EE and ME teams.*

In Figure 15 can be seen the usage of different agile methods.

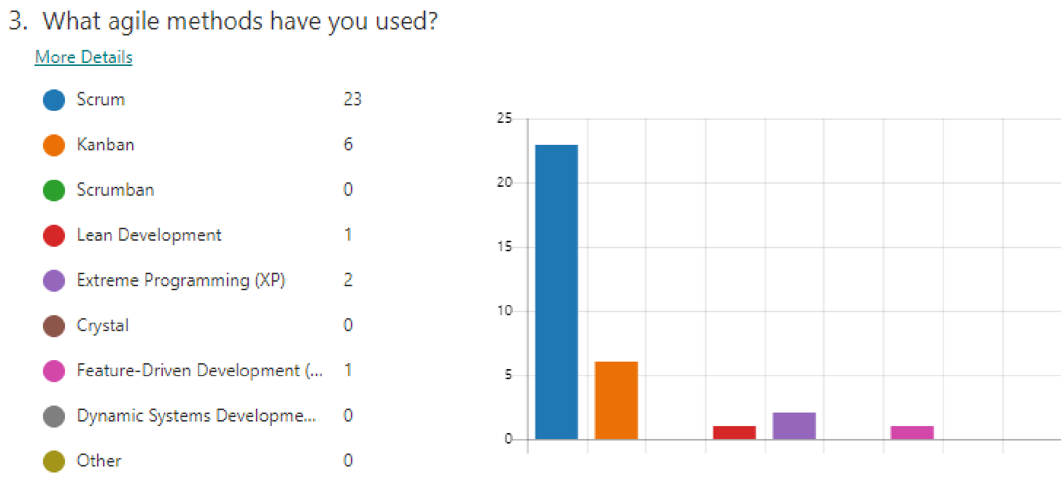

Figure 15. *What agile methods have you used?*

Clearly most of the participants have used Scrum as an agile framework. As stated in the chapter 2.2 and 15th annual State of Agile (2021) report most used agile methodology is Scrum. The connection can be seen in the report and in the survey that Scrum is most popular framework of agile methodologies worldwide and in the target company. EE and ME teams had used only Scrum or Kanban as an agile method.

4.2.2 Agile methods at target company

This section included three questions about agile methods at the target company. The questions were:

- How familiar are you with agile methods?
- Matrix question about current experience with agile methods
- How important you consider agile methods in your team in the future?

Participants were asked about their knowledge of agile methods and the responses were scaled with Likert scale. Majority of the participants were either somewhat familiar or not so familiar of agile methods and a few were very familiar or extremely familiar with the methodologies. Typically, SW and Validation engineers were more familiar with the agile methods than EEs and MEs. All results from the matrix question can be seen in Figure 16. EE and ME answers are separated in the Figure 17 for the comparison.

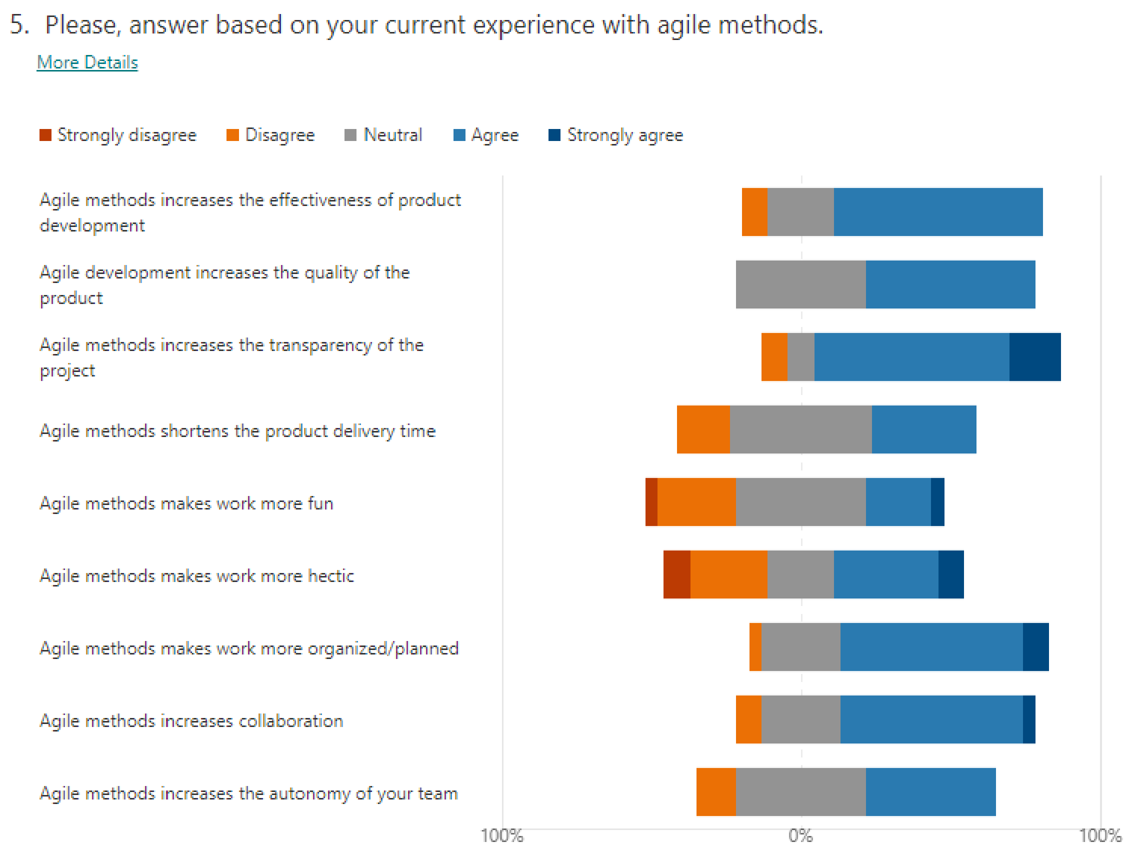

Figure 16. *Matrix question results of agile methods of all participants.*

Overall responses were more on the positive side than the negative side. Third claim "Agile methods increases the transparency of the project" was answered mostly "Agree" or "Strongly agree". As stated in the theoretical background, transparency is one element in which agile methods focus and improves. Same is with claims "Agile methods makes work more organized/planned" and "Agile methods increases collaboration".

Compared EE and ME answers to all answers shows more disagreement in the claim "Agile methods makes work more fun". Also, in the claim "Agile methods makes work more hectic" ME and EE answers are more on the agree side than disagree while all answers show nearly 50-50 distribution. EE and ME answers to the claim "Agile methods makes work more organized/planned" increased a little the disagreement and neutral answers but also increased the agreement in the extreme end.

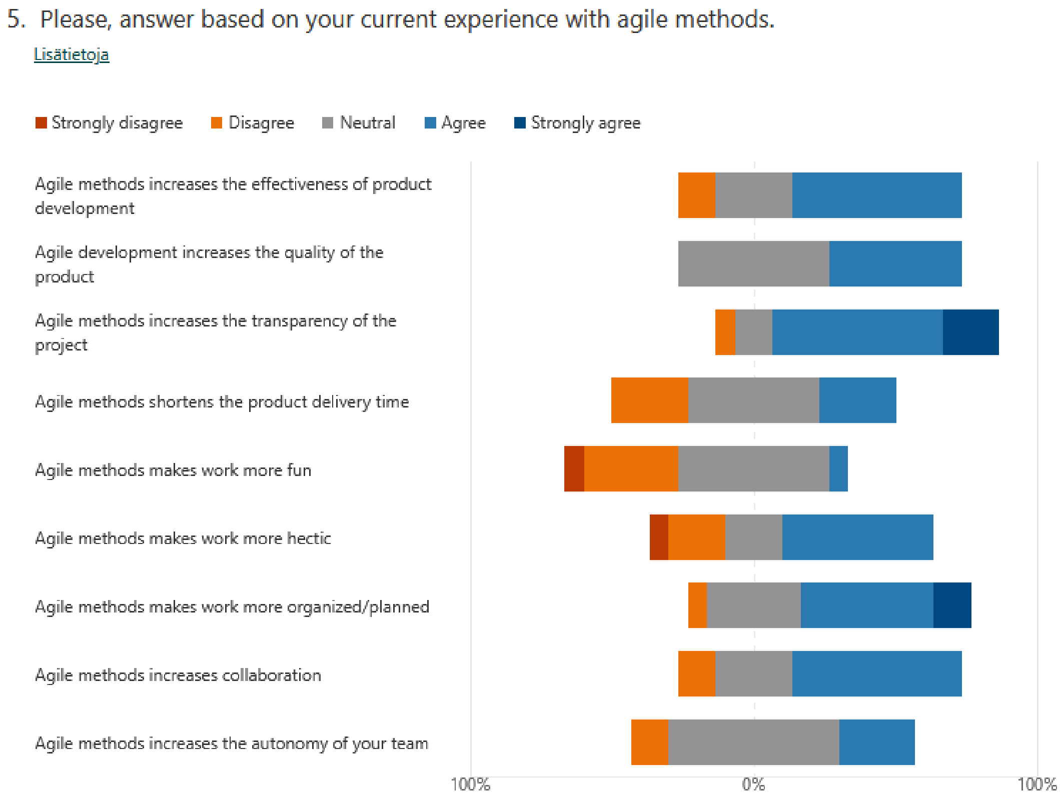

Figure 17. *Matrix question results from the EE and ME teams.*

People were also asked that how important agile methods are in the future. Majority of the responses showed that agile methods should be considered somewhat important and a few extremely important. 26% answered “neutral” and 13% answered “somewhat not important”. None of the participants answered, “extremely not important”. There was no major difference between answers from ME and EE compared to all answers.

4.2.3 Agile teams and collaboration at target company

In this section people were asked questions about agile teams and collaboration at the target company. Section included one matrix question with seven claims answered using Likert scale and one open ended question. These were:

- Matrix question of current experience about team and collaboration
- Open ended question about teams and collaboration

Results for the matrix question of all participants can be seen in Figure 18 and separated answers from EE and ME teams can be seen in Figure 19.

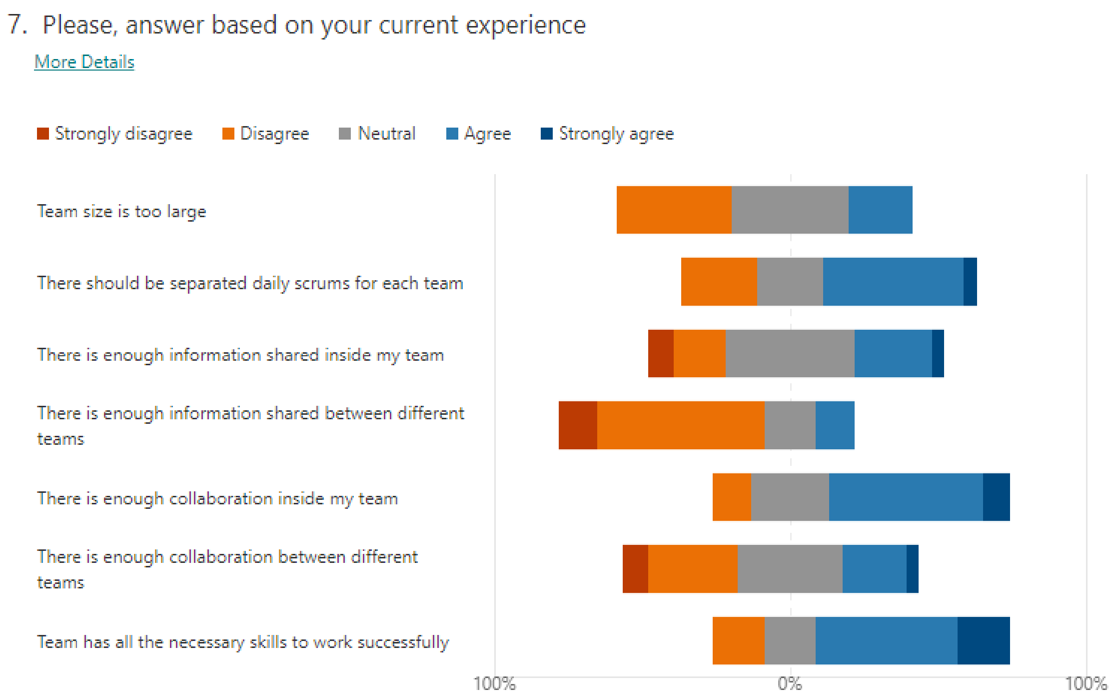

Figure 18. *Answers from all participants.*

Time size shared opinions and it was mostly disagreed that the size is too large. EE and ME results had more disagreement and agreement decreasing the number of neutral answers. Based on all answers it was mostly agreed on that teams should have separated daily meetings. EE and ME answers shared opinions 50-50 regarding to Daily Scrums. Claim about information sharing inside own team had strong disagreement and strong agreement. EE and ME answers had a few more answers on both extreme ends. Claim about information sharing between teams had strong disagreement in both cases. Collaboration inside own team was mostly agreed and didn't differ between all answers and ME and EE. Collaboration between different team was either disagreed or agreed. ME and EE had fewer neutral answers and a few more answers in both extreme ends. Last claim was mostly agreed on. ME and EE responds had a few more agreements and a fewer neutral answer.

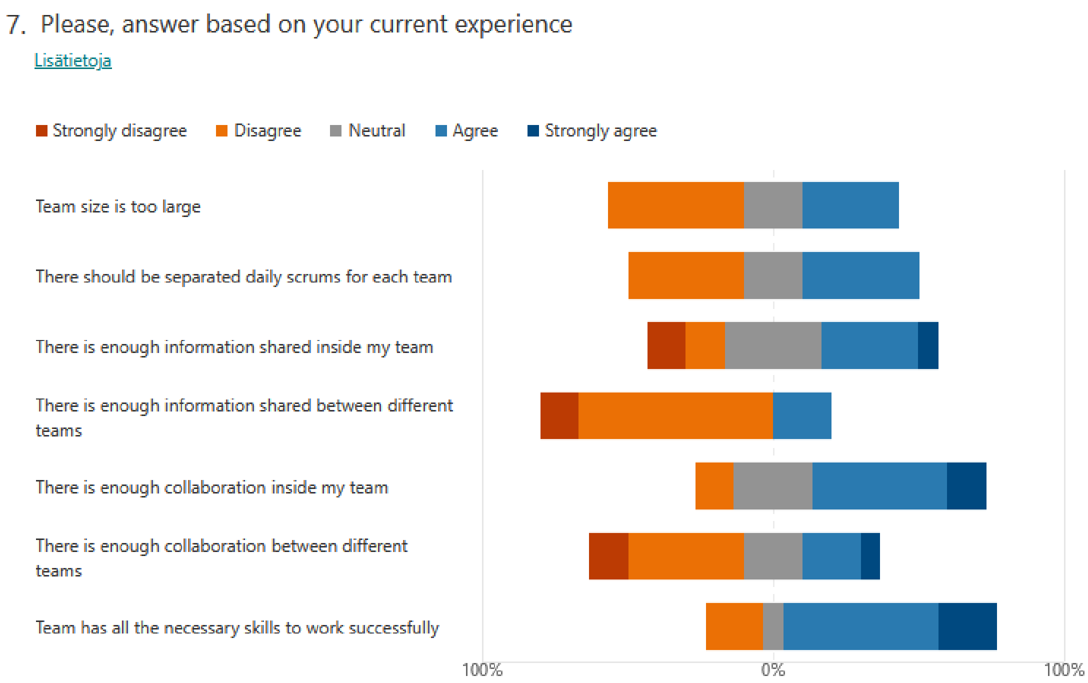

Figure 19. *Answers from EEs and MEs.*

In the open-ended question people had a free word of agile teams and collaboration. Two responses had similarities, and these are shown below:

“It's everyone's responsibility to give visibility to others. By doing that, we help inclusion to be built.”

“All team members should take part of collaboration.”

Answers indicates that as an individual one must be active and contribute to team.

4.2.4 Agile events and usage of JIRA at target company

In this section participants were asked various questions of agile events and usage of Jira at the target company. There were total of seven questions including matrix question and close and open-ended questions. These were:

- Matrix question about agile events using Likert scale
- How many times do you use Jira in a work week?
- How much time do you spend using Jira in a work week?
- Matrix question about usage of Jira
- What do you consider as the main benefits of using Jira?
- What do you consider as the main challenges of using Jira?

- Anything else you would like to comment about using agile methods and Jira?

Matrix question results of agile events of all participants are shown in the Figure 20. Separated answers from EE and ME are shown in the Figure 21.

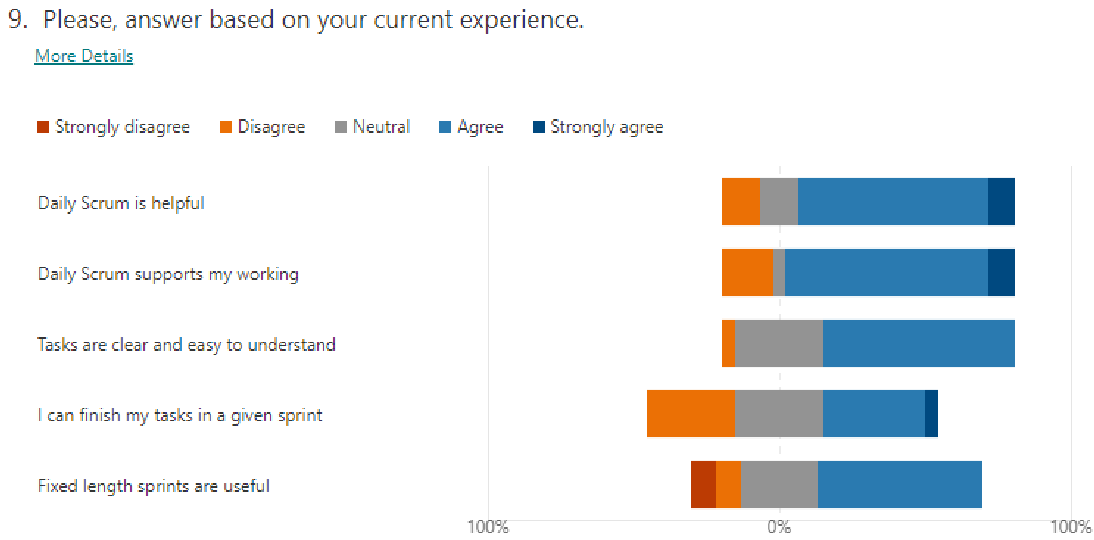

Figure 20. *Matrix question results of agile events of all participants.*

Majority agreed that Daily Scrums are helpful. EE and ME answers differed a little by having no strong agreement on the claim, having a few more neutral answers and a few more disagreement. Same applied on the second claim. Majority agreed that Daily Scrum supports the working. ME and EE had a few more disagreements. Third claim was nearly identical between all answers and EE and ME answers. Majority agreed that tasks are clear and easy to understand. EE and ME had a few more disagreements and neutral answers. Second last claim shared opinions. EE and ME had nearly 50-50 disagreement and agreement while a few strongly agreed to the claim. According to last claim about fixed length Sprints the majority agreed on that those are useful. Although there were a few strong disagreements in all answers and EE and ME answers.

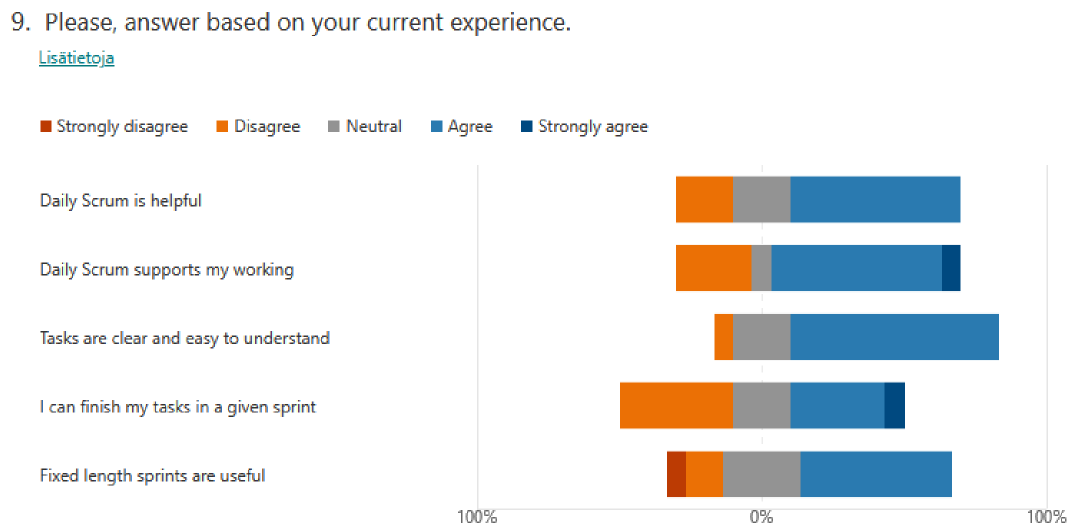

Figure 21. *Separated answers from EEs and MEs.*

People were asked how many times they use Jira in a work week. 52% answered “a few times a week” while 35% answered “every day”. Majority of the participants use Jira a few times a week or more. 13% answered “about once a week” and none of them answered less than once a week. There were also asked how much time you spend using Jira in a work week. 39% answered 15-30 minutes, 26% 30-60 minutes, 22% more than 60 minutes and 13% less than 15 minutes.

Results for the matrix question about usage of Jira of all participants can be seen in Figure 22. Separated answers from EE and ME can be seen in the Figure 23.

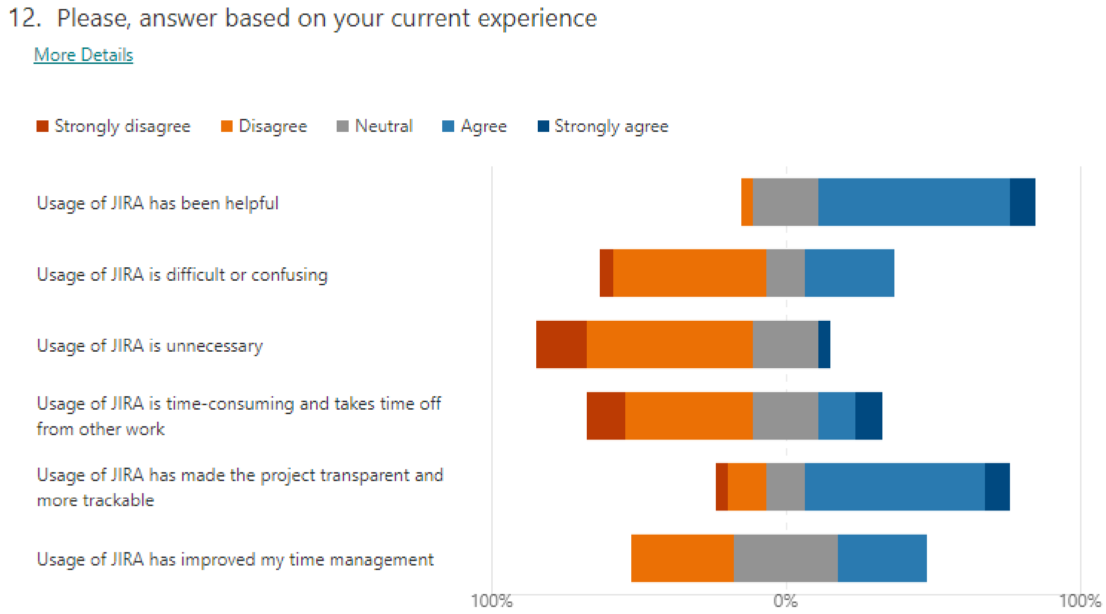

Figure 22. *Matrix question results about usage of Jira from all participants.*

Majority of the participants agreed that usage of Jira is helpful and that it makes project more transparent and trackable. There was no major difference between whole dataset and EE and ME answers. Usage of Jira was seen a bit more time consuming in EE and ME answers than in whole dataset. Also, more people in EE and ME team disagreed that usage of Jira has improved their time management compared to the whole dataset.

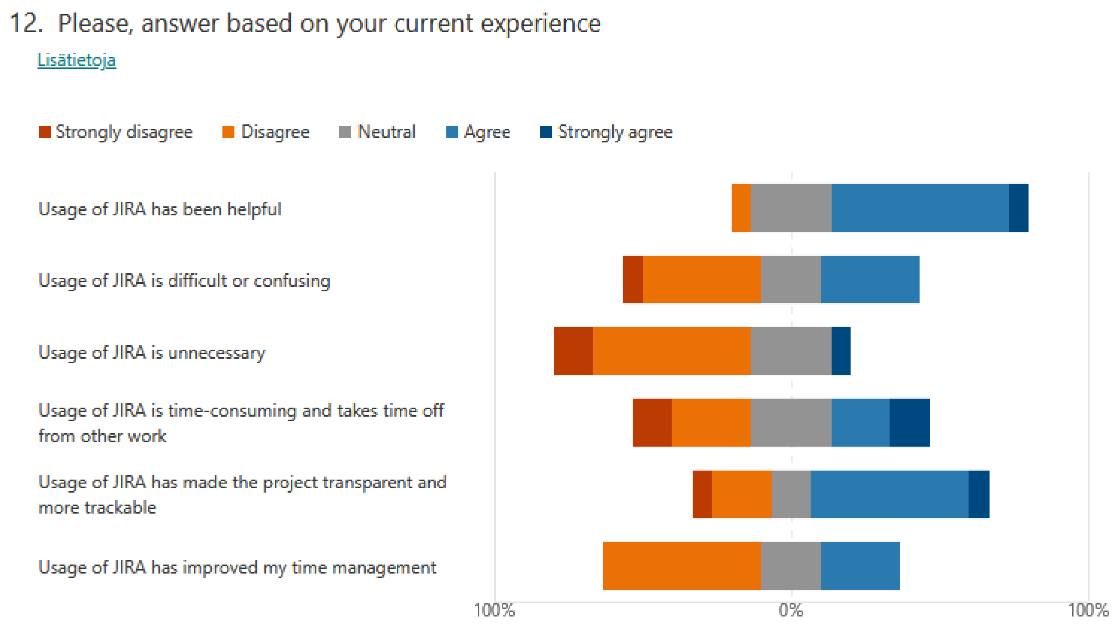

Figure 23. *Answers from EEs and MEs.*

It was also asked that what are the main benefits and main challenges of using Jira. These responds are collected to the Table 5.

Table 5. *Main benefits and challenges of using Jira.*

Benefits	**Challenges**
Traceability and tracking of tasks	Getting people to use it actively and properly
Transparency and shared information	Hard to find information / Too much information
Information in the same tool	Lack of training and experience
Documentation and results	Defining tasks and detail level of each task
Organized structure	Customization

Many of the respondents said that the benefits of using Jira are traceability, trackability and transparency. Many claimed that there is a lack of training and low experience using these types of software in a daily basis. Also, it is seemed somewhat complex or overwhelmed if there is a lot of information available.

4.3 Interview results

Interview results are split into three themes: information flow, Scrum and Jira. As stated in the chapter 3.2.3 interview was a semi-structured with three main themes and each theme included 1-2 questions. In all themes the scope is at each team's point of view.

As seen in the Table 6, total of five people was interviewed. Interviewees were from different teams and had different roles and backgrounds. Scope was more on the electronics and mechanical engineering side.

Table 6. *Specialist interviews*

Interviewee's team	**Interviewee's role**	**Date and length of the interview**
Mechanical	Mechanical engineer, team lead	6.4.2021, 45 min
Electronics	Electronics engineer	7.4.2021, 30 min
Mechanical	Mechanical Engineer	8.4.2021, 40 min
Software	Software engineer, team lead, Scrum Master	9.4.2021, 37 min
Electronics	Electronics engineer, Scrum Master	9.4.2021, 48 min

4.3.1 Information flow

In the mechanical engineering team's point of view, it was said that team's internal communications and information flow were good. Daily Scrums have also improved information sharing between mechanical and electrical engineering teams. It was also said that sharing information is not a problem. During the remote working it needs more effort to get the information. Inside own team the information from outside is usually shared through team leader who participates in cross-team meetings.

Participants in electrical engineering team had similarities in answers compared to mechanical engineering team members. It was said that usually the information is shared well inside the team. Sometimes when there is smaller group's meeting inside the team the information won't always reach other team members. It was also said that in remote working the information sharing is more challenging than on-office. Coffee table conversations are surprisingly effective way to share information especially between different teams. It was said that there should be someone responsible to share information between teams and meeting minutes memos should be done more often and shared to everyone.

From software engineering point of view the Daily Scrums makes effective information sharing possible. There is still missing Scrum of Scrums which connects multiple teams who need to work together. Currently information is shared between teams in daily stands for team leaders and line managers. It was also said that during remote working individuals must be active in Teams and share the most obvious information also.

4.3.2 Scrum process

From the mechanical engineering point of view Scrum is seen as a potential agile method but it has its own challenges when it comes to mechanical and hardware development. Especially in the conception and prototype planning phase the requirements and specifications can change, or something is unknown. Is the fixed Sprint still reasonable? It was said that defining Product Backlog and Sprint Backlog can be difficult. For example, how to define the size of the task and how to convert product requirements into product features. It was said that current process follows waterfall model, and that Scrum is not used as it is intended to. To become agile there should be flatter hierarchies inside engineering team. Engineers and developers should take more responsibility and be involved in decision making. It was also said that Sprint Planning should be done by the agile team that do the tasks. Agile team should be multi-talented and consists of maximum of 10 people. In the other words in electrical, electronic, and mechanical development an agile team should have experience in all the required fields. Hardware development requires a lot of skills and people and usually is very dependent in third parties e.g., component suppliers and subcontractors.

Interviewees from electrical engineering team had somewhat similar answers than mechanical engineering team members. It was said that four weeks Sprint is a minimum for this type of development. Two-week Sprint is either too short or there is needed a lot more planning. One good point from mechanical and electrical point of view was that Sprints should have clear goals and those should be linked more on the progression. One major problem was pointed out. There is no proper Sprint Planning, and the tasks are added to the backlog when those are already done or in progress. Also, information flow between Scrum Masters should be better. It was said that it would help to generate the tasks to the backlog. It was also said that there is resistance to change according to Scrum and it is seemed somewhat non-essential.

Software engineering team has been using agile methods for years and the Scrum framework is familiar for them. It was said that the main reason to use Scrum in hardware development was to get projects more transparent during remote working. Scrum was chosen as an agile method because the software team has used it so at least someone in the house knew the framework. In software team the product requirements were transferred to epics in Jira and epics was split into smaller user stories. The problem was that the extent of the requirements varies a lot – some are large requirements which needs more effort, and some are smaller, simpler requirement. It was said that fixed length Sprints are more suitable when there is already something done, and it needs small Increments. Also, it was agreed that for hardware development the Sprints should rather be linked to some known milestones than use fixed Sprints. Another idea was to use Kanban board with continuous flow and limit WIP as stated in the chapter 2.3. In the end it was discussed about how the software and hardware development differs regarding agile development. For example, if the supplier changes or there is other major change the on-going Sprint should be closed and start a new one with added or modified tasks. It was also stated that in hardware development the entities are large. Could there be something that adds value to be delivered after the Sprint so that the Sprints won't be too long.

4.3.3 Usage of Jira

As discussed with mechanical engineers Jira is seen as a useful tool. It is seen as a suitable tool for documentation and sharing tasks to developers and engineers. It makes working more organized and there is always the information available that who is working on what. Currently there is no roadmap or longer period schedule available which would be much welcomed. Usage of Jira could be more versatile than it is now. There are possibilities for different kind of plugins and third party supports but the platform is under a larger organization so the changes can be difficult. It was also said that adding tasks to Jira after those are started is frustrating. Currently there is no common guidelines on how to use Jira which might affect to user experience.

From the electrical engineering point of view Jira is seen as a good tool for task tracking, documentation, reporting and test plans. It was also mentioned that there is need for clear guidelines on how to use Jira. Currently the user experience from the Scrum Master point of view is not flawless. Changing the status of the task is seen as time consuming and difficult. It was also discussed that the team is not fully committed to using Jira.

From the software engineering point of view Jira is seen as a good tool for documentation, task tracking and linking the dependencies between issues. It was also discussed that since Jira is under larger organization the changes are harder to push through because the changes affect everyone.

4.4 Implementing agile methods

Scrum is suitable agile framework for one Scrum Team which is multiskilled and self-organized. For multiple teams working on same product, scaled agile methods are more suitable. For target company, LeSS is seen as a potential framework to begin with.

Teams had been adopting Scrum practices and are somewhat familiar with the Scrum framework therefore LeSS was chosen as an agile method. There is a possibility to lose the benefits of Scrum by modifying it too much in the beginning (Benefield, 2008). Same applies to LeSS. In the beginning it is easier to follow the steps provided here than modify the framework before it is implemented. As stated before, Scrum (and LeSS) is empirical process. Over time it will be developed to meet the needs of the target team.

The chapter 2.3 introduces LeSS as a framework. This chapter provides detailed steps to getting started with LeSS as well as best practices found to suite the target team. Findings from observation, survey and interviews are also reflected to the suggestions.

4.4.1 Getting started

Vodde & Larman (2016) states that successful adoption requires challenging peoples' assumptions about how organizations should work and simplifying the organizational structure. It is harder to implement new framework to ongoing project so the best time to become agile is before new project. Here is presented five steps to getting started with LeSS framework (Vodde & Larman, 2016, Ch 3):

1. Educate everyone
2. Define “product”
3. Define “done”
4. Have appropriately structured teams
5. Only the Product Owner provides work for the teams

First step is learning and training. It is important to understand Scrum framework, what it is and what it is not. After that comes LeSS training where the focus is on scaled framework. When people know the framework and its capabilities it is more motivating to implement it from theory to practice. It was said in the survey and interviews that Scrum training was inadequate, but still agile methods were seen positive. With enough training and workshops, it's easier to find right mindset.

The second step is defining the product. This determines later the content of the Product Backlog and who makes a suitable Product Owner. The product definition must be practical enough to start. The product can be defined narrowly which leads to non-customer-centric technical Product Owner and technical items on the Product Backlog. The product can be defined broadly and if it defined too broadly it becomes impractical. While it is inevitable that product needs technical requirements in the beginning is more important to think broader and more customer centric. (Vodde & Larman, 2016, Ch 7)

Third step is defining Definition of Done. This clarifies the Team's understanding of what must be accomplished while creating each deliverable, testing the deliverable, and fixing defects found in it (Thompson, 2015). Vodde & Larman (2016, Ch 7) states that Definition of Done is an agreement between the teams, the Product Owner, and managers on which activities are performed during the Sprint. A few examples of Definition of Done in hardware development could be:

- Module passes all required tests
- Mechanical and electrical drawings, schematics, documents etc. are up to date
- Certifications and standards are fulfilled

Definition of Done should be defined together with the team and Product Owner and if needed with other stakeholders.

Fourth step is moving from traditional competence-based teams to cross-functional teams. As stated in the chapter 2.2.1 the team has all the necessary skills to create value for the product in each Sprint. Team should also be dedicated, stable, long-lived and co-located (Vodde & Larman, 2016, Ch 3). As stated in the interviews a hardware Scrum Team should consist skills in electrical, electronic, and mechanical engineering.

The idea in fifth step is that all the work for the development teams comes from Product Owner. It is important that initial teams have their focus on shared goals for the product. The focus is easily lost if the work comes from many sources with no proper prioritization. The work is pulled from the Product Backlog which is maintained and prioritized by Product Owner. (Vodde & Larman, 2016, Ch 3)

4.4.2 Teams and responsibilities

LeSS roles are important in scaled Scrum process. In the Figure 24 is presented one possible block diagram of Scrum roles for the hardware development in the target company. This could be scaled to include software development as well.

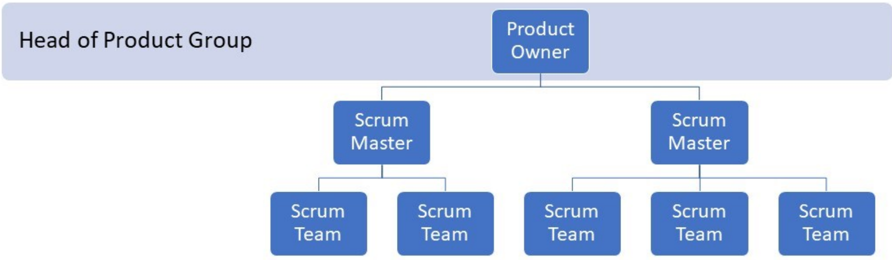

Figure 24. *Roles implemented from LeSS Framework.*

The Figure 24 should not be read as hierarchies between roles. Connections in the diagram presents interactions and communication between roles. Flat organizational hierarchy is a key point in Scrum and LeSS. Product Owner in LeSS has the same responsibilities as in one-team Scrum which are described in the chapter 2.2.1. Scrum Masters are servant leaders of the Scrum Team. As stated in the chapter 2.2 the main responsibility is to facilitate Scrum events and help and train team members to establish and understand Scrum theory and practice. Scrum Master serves Product Owner by helping the team to understand Product Backlog items created by Product Owner.

Scrum Team should be multi-talented and include members with different background so the team would be skilled enough to finish tasks from Sprint Backlog. This means that electronics engineers and mechanical engineers should be merged into one or two Scrum Teams. Idea is to have knowledge from both fields of engineering in both teams. Smaller teams increase collaboration and individuals are more easily involved to the planning and development. Having multiple Scrum Teams developing one product requires more organizing and collaborating between teams.

Suggestion is that line managers become Product Owners and Scrum Masters are Scrum Team members at the beginning. It is worth considering hiring experienced Scrum Master with the knowledge and experience of LeSS. Therefore, Scrum Master can be a coach and increase the understanding of agile. As stated earlier, there can be one large team or two smaller teams. LeSS and Scrum are empirical frameworks so if one is not working try another. It is more important to start with something than condemn the framework without trying it.

4.4.3 Planning and artifacts

Before the first Sprint it is important to have the Product Backlog with prioritized Product Backlog Items (PBI). Product Backlog in LeSS is the same as in one-team Scrum environment. A good Product Backlog should have (Vodde & Larman, 2016):

- estimates for all items
- finer grained items at the top and coarser grained items further down
- prioritized items

A target company uses Jira for managing the Product Backlog and Sprints. In the interviews and during observation it was noticed that initial PRD document (Product Requirement Definition) was done in Excel. When the PRD document was mature enough it was converted to Jira tasks where one task represented a product requirement. LeSS and Scrum guides suggests using Epics, User Stories and Tasks as PBIs (Vodde & Larman, 2016). At the beginning it would be less confusing to stick with a PBI that adds value to the product and divide that into tasks if needed. Idea is to make PBI items small enough so that it can be done in a single Sprint and create a task so that can be done in a day or two.

As stated in the Scrum Guide (Schwaber & Sutherland, 2020) and scaled version (Vodde & Larman, 2016, Ch 2) there should be only one Product Backlog for the Product. Key reason is to increase overall transparency and increase complexity so the teams can keep the whole-product focus. Currently there is separate Backlogs for hardware and software development which decreases the transparency. Better idea is to share the Product Backlog and if needed, categorize inside the backlog hardware and software. The direction should be in the integrated and goal-oriented backlog.

Liminka, (2019) found out in his thesis that Sprint goal was an important part of the Sprint cycle. He states that with Sprint goal everyone could get a better view of the big picture and had a common objective. A clear common Sprint goal should be defined before the Sprint, and it works as a motivator for the developers and engineers. A Sprint goal is decided by the development team.

Estimating a backlog item can be difficult but over time this helps teams understand how much effort is needed and how much they can achieve in a period. The idea is to estimate all backlog items. There are different methods to do the estimation but one is planning poker which involves the whole team participation (Radigan, 2021). In this method each item is discussed briefly, and each member will formulate an estimate. Then everyone holds up a card with the number that reflects their estimate. If there is variety between estimations, it should be discussed briefly and decide the estimation for the item. A suggested method for rating the items is to use relative estimation with Fibonacci sequence. When the estimation is done for the backlog items a burndown and burnup charts in Jira software can be used for tracking the project's progress.

4.4.4 Meeting practices

Product Backlog Refinement (PBR) is essential part in LeSS. It is a workshop where teams clarify upcoming items with users and stakeholders, split big items and re-estimate items (Vodde & Larman, 2016, Ch 11). LeSS rules for Product Backlog refinement are following (Vodde & Larman, 2016, Ch 11):

- PBR is done per team for items they are likely going to implement in the future. Multi-team PBR is preferred to increase shared understanding.
- PBR is done together with Product Owner and development teams.
- Product Owner does the prioritization

For the target company it is suggested to have a **multi-team Product Backlog Refinement** followed by a **team-specific Product Backlog Refinement**. For the start these should be done weekly to keep Backlog items up to date and refined. Multi-team PBR is attended by project leaders (Head of Project Group), Product Owner(s) and representatives of each development team. The focus in multi-team PBR is to choose the items which would be refined in upcoming team specific PBR. It is a good idea to rotate representatives from each team. In team specific PBR the focus is to define and estimate the items more accurately for upcoming Sprints. In this way the Product Backlog is constantly updated and refined. Attendees in team specific PBR are the Product Owner, Scrum Master, and the members of the Scrum Team.

Before the first Sprint there is **initial Product Backlog Refinement**. Once the Product Owner is identified, the development teams and their members have been decided and Product Backlog is made it is time to refine the Backlog before the first Sprint. Attendees are Project Leaders, Product Owner(s), all team members for all teams, Scrum Masters. customers/users and other stakeholders. The basic objective is to sufficiently refine enough items so all teams can be productive in the first Sprint. (Vodde & Larman, 2016)

The Sprint starts with **Sprint Planning 1** which is attended by Product Owner, and representative from each team. General idea is to decide which team will work on which items. In the Figure 25 is presented the LeSS Sprint Planning.

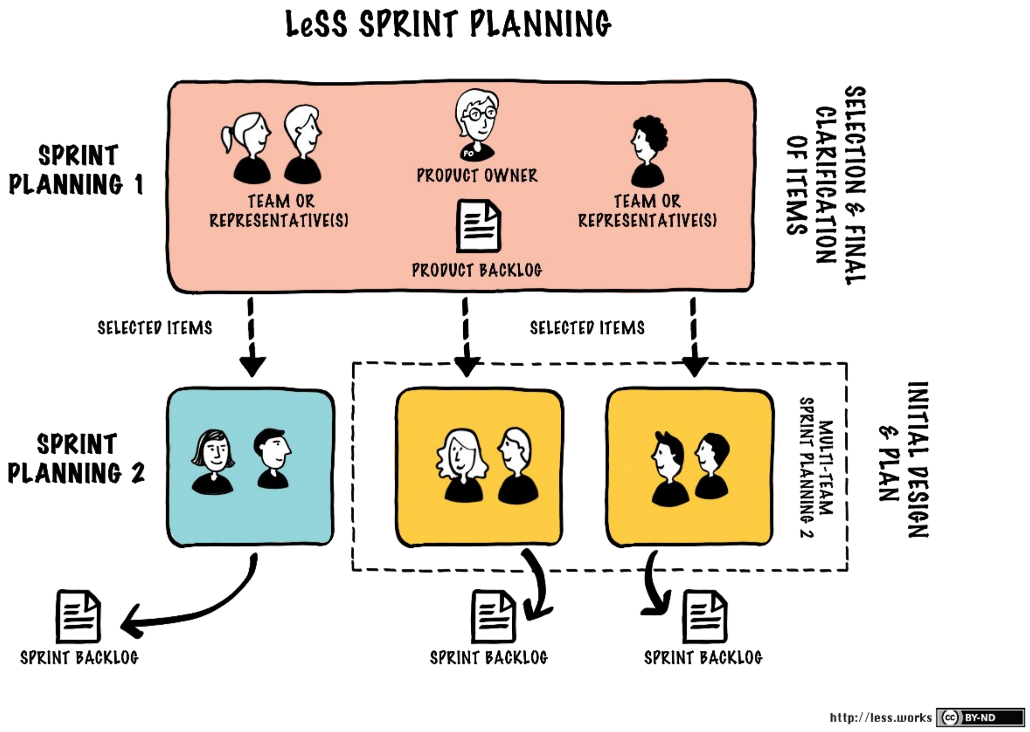

Figure 25. *LeSS Sprint Planning 1 and 2* (Less.works, 2021e).

During Sprint Planning 1 it is suggested to examine the high-level schedule which could be done using Jira roadmaps. During the Sprint Planning 1 the Sprint goal is decided for the upcoming Sprint. Less.works, (2021c) provides tips for getting the Sprint Planning 1 to work well:

- Limit the participants. If there are only two teams, then the whole team can attend. If there are more than two teams, then send one or two representatives per team.
- Product Owner presents the highest priority items from the Product Backlog to the team.
- The teams apportion the items amongst themselves.
- Any open questions related to the items (that weren't resolved in earlier Product Backlog Refinement) are discussed.
- Discuss the need for coordination between teams during the upcoming Sprint.

Sprint Planning 2 is held right after the first planning session, and it is same as in one-team Scrum discussed in chapter 2.2.2. This is longer meeting which is facilitated by Scrum Master for Scrum Team. If seen necessary, the Product Owner can attend the meeting as well. In that case the scheduling should be staggered so the Product Owner can join every team planning session. The agenda in Sprint Planning 2 is to plan on how to get the items done during the Sprint. Often this requires splitting the requirements and PBIs into smaller, executable tasks.

Daily Scrum is same as in one-team Scrum. All teams should have their own short daily meeting scheduled at the same time every day. During the Daily Scrum everyone tells what they did yesterday, what will they work on today and is there something on their way. It is a good opportunity to discuss about impediments and problems and arrange a follow up discussion outside the Daily. In LeSS it is also possible that representatives from other teams join the Daily Scrum.

Scrum of Scrums is a cross-team coordination technique. Although it is not part of the LeSS Framework it could be useful in hardware development. Scrum of Scrums is like Daily Scrum, held a few times in a week. Scrum of Scrums is attended by Scrum Master, representatives from every development team, Product Owner and Project Leader if seen necessary. Idea is to manage dependencies and cross-team interactions.

Sprint Reviews and **Sprint Retrospectives** are essential part of the Sprint. Especially when learning LeSS and agile practices these are the two meetings that can improve the forthcoming Sprints. These are held in the end of the Sprint. Vodde & Larman, (2016, Ch 14) points out six principles when scaling reviews and retrospectives: customer-centric, transparency, continuous improvements towards perfection, empirical process control, whole-product focus, and Systems thinking. During the Sprint Review, customers and stakeholders examine what the teams built during the Sprint and discuss changes and new ideas (Less.works, 2021f). Attendees in Sprint Review are the development teams, Product Owner, customers, and stakeholders. While Sprint Reviews is more about the outcome of the Sprint, the retrospectives idea is to improve the Scrum process. As Schwaber & Sutherland (2020) states in the Scrum Guide, the purpose of the Sprint Retrospective is to plan ways to increase quality and effectiveness. In LeSS retrospectives have divided into two separate meetings as seen in the Figure 26.

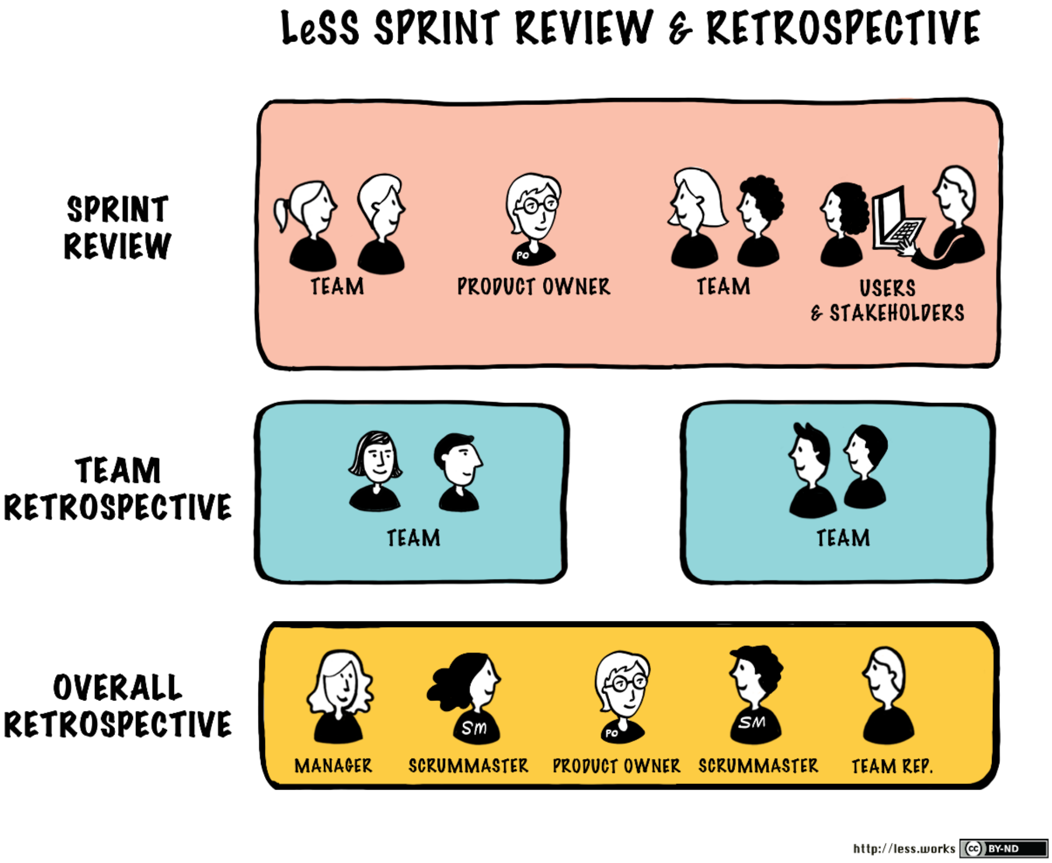

Figure 26. *Sprint Review and Retrospective in LeSS* (Less.works, 2021d).

Team Retrospective is same as in one-team Scrum discussed in the chapter 2.2.2. Overall Retrospective is new meeting in LeSS where the attendees are Scrum Masters, Product Owner, Team representatives and managers. Topics that might be discussed in an Overall Retrospective are (Less.works, 2021d):

- How well are the teams working together?
- Is there something that a team should share to the rest?
- Are there systemic organizational issues that cause problems in how teams operate?
- Is the Product Owner doing well?

4.4.5 Timing of the LeSS meetings

This chapter provides an example schedule for LeSS meetings during the Sprint. It is important to keep the schedule consistent through Sprints but allow some modifications if those suits the development better. In the Figure 27 is presented an example schedule for two-week-long Sprint (for simplicity). As stated earlier the Sprint length can be anything from two to eight weeks. It is up to the Scrum Teams to decide which length is the most suitable for them.

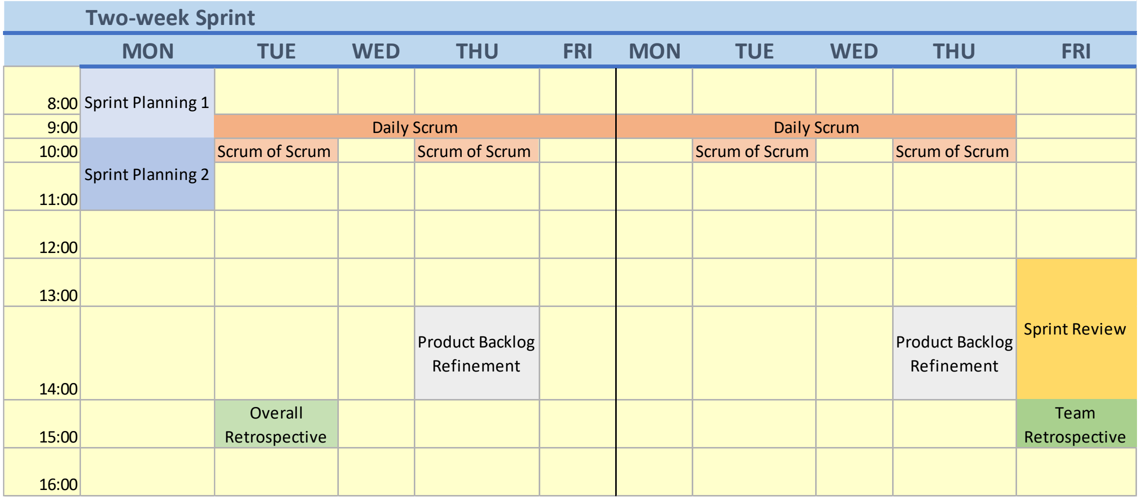

Figure 27. *Two-week LeSS Sprint.*

The Sprint starts with Sprint Planning sessions which are divided to Sprint Planning 1 and Sprint Planning 2. Planning sessions together are timeboxed at eight hours or less for a four-week Sprint. The shorter the Sprint the shorter the timeboxes. Daily Scrum meetings are held daily expect of planning day and review day and are timeboxed to 15 minutes. Scrum of Scrums are held a few times a week to get synchronized with other teams and are not limited to 15 minutes timebox. Product Backlog Refinement, including multi-team and single team, are timeboxed to four to eight hours for two-week Sprint. Sprint Review and Sprint Retrospective are held at the end of the Sprint. Reviews are timeboxed to four hour or less for one-month Sprint. Even though Overall Retrospective is at the beginning of the Sprint it is a retrospective of last Sprint. There is already plenty of things to discuss in the end of the Sprint on Friday, so the Overall Retrospective is moved to the start of the next Sprint. Retrospectives together are timeboxed to three hour or less for one-month Sprint.

In conclusion: educate everyone, define product, define done and have appropriately structured teams. LeSS is scaled Scrum developed to multiple teams who work together on a single product. LeSS is about figuring out how to apply the principles, purpose, elements, and elegance of Scrum in a large-scale context, as simply as possible (Less.works, 2021b).

5. DISCUSSION

The main objective of this study was to find out what would be suitable agile framework for hardware development. First section provides key results and answers to research questions. In second and third section is discussed about research implications and assessment of approach and design. In the last section is discussed about future research needs.

5.1 Key results

The key results of the thesis are discussed under each research questions. To find out the answers for the research questions a literature review, process observation, survey and interviews were conducted. Survey questions was analysed also from the EE and ME point of view. These answers did not differ that much from the total answers. There was slightly positive feedback towards agile methods from the answers including SW and Validation team answers. That is probably because they have been adopting agile practises longer than EE and ME and are familiar with the concept.

1) How agile methods can be used in hardware development?

Agile product development methods are widely used especially in software development (Becerril et al., 2018; Bohmer et al., 2017; Feldmuller, 2018; Ovesen, 2012). Agile methods are recently seen as an opportunity in other fields as well when a need for new development methods are generally acknowledged (Ovesen, 2012). A recent report (State of Agile, 2021) shows that other areas than software development have adopted agile principles and practices as well.

In chapter 2.7 are discussed about the differences between hardware and software development. Table 2 shows that when the maturity of the product increases the cost of changes increases as well. This is because hardware development requires physical components and mechanics and replacing them in a latter phase is expensive. What is truly beneficial in the hardware development using agile methodologies is the increased transparency and information flow. Shared Product Backlogs keep the upcoming work in one place and when teams are fully committed to agile mindset the focus is maintained better.

Hardware development requires a broad skillset and thus engineers from different background (e.g., mechanics, electronics, regulatory, validation) are needed. This means that an agile development team should consists of either several engineers with different skillset or fewer engineers with a broad skillset. This usually leads to the situation where it is needed multiple teams to work with the same product. Scrum is suitable agile framework for one team which is multiskilled and self-organized. A case study shows that it is possible to follow the Scrum framework in new product development with minor tweaks for rules and criteria, yet still following the principles successfully (Reynisdottir, 2013). For multiple teams working on same product, scaled agile methods, like LeSS, are more suitable. LeSS is introduced in chapter 2.3 and best practices are discussed further in chapter 4.4.

2) How agile methods are currently seen in target company?

When Scrum was introduced to the electrical and mechanical engineering teams there were little to no experience about the framework. During the survey EE and ME teams had already used some practices from the Scrum framework. Survey results showed that majority of the respondents had experience with agile methods in practice one to three years or less than a year. Although some of the Scrum practices had been used for a year, clearly majority of the respondents were somewhat familiar or not at all familiar with agile methods. Naturally SW and Validation were more familiar with agile methods. During the interviews the opinion was also clear that there was no proper training for the framework or no experienced Scrum Masters to facilitate the Scrum events. Agile development differs from traditional plan driven development, so the traditional mindset needs to be challenged to become agile. Proper training and knowledge of the agile development with workshops is the key to implement agile development practices.

Survey shows that agile methods are beneficial for all the teams. EE and ME participants mostly agreed that agile development practices:

- increases the effectiveness of product development
- increases the transparency of the project
- makes work more organized and planned
- increases collaboration

Especially Daily Scrums are seen helpful and supportive. Doing Daily Scrums daily increases the collaboration and transparency of the project. The observation of the hardware development process and participating to daily meetings showed the lack of planning of the Sprint. There were also many participants, so it became frustrating to ask everyone about what they have been doing and is there any impediments. This is one reason to keep the agile teams small, so the meetings are efficient, and the focus is kept in progression.

Interviews showed that combining mechanical engineering and electronic engineering teams into one hardware team increased the collaboration and information flow. Downside was that the team size became large. It was also pointed out that working completely remote the information sharing becomes even more important. While working on-site simple coffee-talk can have a lot of information which flows beyond team boundaries.

Although survey results pointed out that fixed Sprint was more agreed than disagreed, the interviewees challenged the use of fixed Sprints. It was stated in the interview that fixed length Sprint is suitable when there is already something done, and it need small Increments. During the conception and prototype planning phase requirements and specifications can change and the uncertainty is high. A suggestion was that in early development phase the Kanban board with continuous flow could be used instead of fixed Sprint. This allows pulling work items from the backlog in the prioritized order rather than planning a work into a fixed length Sprint.

The usage of Jira was seen helpful, and it has made the project more transparent and trackable. Jira supports fully Scrum and Kanban development. Main benefits and challenges are listed in the Table 5. Scrum and Kanban boards can be used without any software tools using only a whiteboard and sticky notes. In large projects this can be hard to maintain but the idea is to focus on the people doing the agile rather than the tools. One of the core values in Agile Manifesto is "individuals and interactions over processes and tools" (Beck et al., 2001).

3) How to get started with agile development?

LeSS is suggested as an agile framework for the target company. Whereas Scrum is for single development team, LeSS is developed for two to eight teams working on the same product. The teams had already some knowledge of Scrum framework so the decision to choose LeSS as a potential framework was justifiable. In the chapter 4.4 is discussed more thoroughly about LeSS and how to get started with it. Everything starts with proper education and training where everyone is introduced to Scrum framework and after that comes LeSS training. It is important to get an agile mindset after years of traditional plan driven development. Hiring an experienced Scrum Master can help to get started and she or he can be a good mentor and teacher of agile mindset during the learning process. The whole organization must become agile to successfully implement agile methods for development process. This is usually more difficult to large organizations like the target company than smaller organizations.

It is important to define the product which is been developed. The product determines the Product Owner and the content of the Product Backlog. Defining Definition of Done clarifies the team's understanding of what must be accomplished for each deliverable (Thompson, 2015). It is an agreement between the development team, Product Owner and manager on which activities are performed during the Sprint to close the task (Vodde & Larman, 2016, Ch 7).

Development teams should be cross-functional, multiskilled and optimally maximum of ten people large (Vodde & Larman, 2016, Ch 3). In hardware development there should be skills and competence in electronics, mechanics, and electrics. All the work for the development team should flow through Product Owner. The focus is easily lost if the work comes from many sources with no proper prioritization. The work is pulled from the Product Backlog which is maintained and prioritized by Product Owner (Vodde & Larman, 2016, Ch 3).

5.2 Research implications and impact

Transformation from traditional product development to agile product development doesn't happen in a week, month or even in a year. Adopting agile practises is Incremental process and it's against agile nature to have fixed roadmap for the changes needed. This thesis with its key results provides a starting point for adopting agile practises in a hardware development. Survey and interviews provide important feedback about the current state of hardware development using agile practices and tools.

Agile methodologies are widely used in software development and in hardware development the potential have been recognized in the past a few years (Feldmuller, 2018; State of Agile, 2021). The key result of this thesis shows that adopting agile practices have increased transparency and trackability of the project as well as increased collaboration and information flow. These are important findings to support the theory behind agile methodologies. Survey shows that attendees have positive experience of agile practices and agile is seen as potential way of working in the future.

5.3 Assessment of the approach and design

This thesis is a case study of agile practices in hardware development at the target company. Case study as a research design was suitable for this type of thesis and topic. Mixed methods, like quantitative and qualitative data from survey and qualitative data from interviews alongside with current state observation and literature review gave plenty of information and knowledge of the subject.

It was challenging to find relevant literature of agile methods used in other industries than software development. As stated in the theoretical background, Agile Manifesto was done for software developers by software developers. True success stories of using agile in hardware development was hard to find.

Survey reached 46 co-workers via email. Total of 23 participated to the survey thus answer rate was 50%. This is considered a good answer rate and the survey gave plenty of data to analyze. Although survey gave good amount of data the validity of the answers should considered. Some people might have been in a hurry and didn't focus on the questions. Some of them could have answered randomly to the questions without even reading the questions. More answers would have given more data which would have been beneficial for the study.

Interviewees were from electronic, mechanical and software engineering teams. Total of five people were interviewed. Considering the restricted time to conduct the study five people was seen a good number of interviewees. The questions asked and the conversations gave great answers but afterwards it would have been better to ask more about challenges and personal opinions on how to improve current development process.

Considering the challenging topic and relatively low amount of literature the research design was suitable for the thesis. Qualitative and quantitative methods was a must to gather background knowledge and experience of the subject. It would have been great to pilot the framework with a real-life project to see how these practises suits the development team.

6. CONCLUSION

This thesis was done as a part of an internship in a large multinational leading technology company. There are teams in hardware design (combining electronics and mechanics design), software development and validation. The primary function of the R&D unit is to take risks and develop and test new ideas quickly.

The hardware team adopted from software team an agile approach to product development and started to implement some agile practices from the Scrum Framework such as Product Backlog, Sprints and Daily Scrums. In this thesis it was studied that how well an agile approach suits the hardware development and what will be the best practices.

Literature review of the thesis introduces agile development and most used agile frameworks. Research method of the thesis is a case study which was built around observing, survey and interviews. These methods completed the knowledge and background of the subject in the case study company.

The thesis gives answers to three main research questions which are:

1) How agile methods can be used in hardware development?
2) How agile methods are currently seen in target company?
3) How to get started with agile development?

With regarding to the first question: agile methods can be used in hardware development much like in the software development. The principals of the frameworks are the same but there must be considered the difference between software development and hardware development. Hardware development requires a broad skillset and thus engineers from different background (e.g., mechanics, electronics, regulatory, validation) are needed. This means that an agile development team should consists of either several engineers with different skillset or fewer engineers with a broad skillset. This usually leads to the situation where it is needed multiple teams to work with the same product. In conclusion: pick an agile approach that suits the teams' needs and start using it. Agile approaches are iterative and Incremental – most likely some tweaks are needed during the development process to meet the needs of the development team.

Agile methods are seen as an opportunity in the target company. Survey shows that agile methods are beneficial for all the teams. Team members in electronics engineering and mechanical engineering teams mostly agreed that agile development practices:

- increases the effectiveness of product development

- increases the transparency of the project
- makes work more organized and planned
- increases collaboration

Especially Daily Scrums are seen helpful and supportive. Doing Daily Scrums daily increases the collaboration and transparency of the project.

LeSS was chosen as an agile development framework for the target team. It is scaled Scrum framework and shares same principals. Whereas Scrum is for one development team, the LeSS can be scaled up to eight teams. To get started with LeSS there are five steps to follow:

1. Educate everyone
2. Define “product”
3. Define “done”
4. Have appropriately structured teams
5. Only the Product Owner provides work for the teams

It can be concluded that it is easier to adapt to simpler flamework and build on it, than try to adopt to more complex one and customize it down to the team specific use.

6.1 Future research needs

The thesis gives a starting point to adopt agile methodologies and practices to hardware development. It is needed more research of the subject especially about the agile hardware development. Agile development has potential outside of the software development and it could be one solution to respond to rapidly changing and dynamic environment.

More proof of suitability of agile methods in hardware development is needed in practice. Even though agile development is seen potential there is not much proof that it really works in hardware development. Thus, it is needed more pilots to adapt agile practises in the development process. Next steps at the case study company would be piloting LeSS framework with a real project and get the agile mindset for the whole R&D unit. In the end it is the people who makes the development process agile.

REFERENCES

Aaltola, J., & Valli, R. (2001). Ikkunoita tutkimusmetodeihin. 2, Näkökulmia aloittelevalle tutkijalle tutkimuksen teoreettisiin lähtökohtiin ja analyysimenetelmiin. In *Ikkunoita tutkimusmetodeihin. 2, Näkökulmia aloittelevalle tutkijalle tutkimuksen teoreettisiin lähtökohtiin ja analyysimenetelmiin*. PS-kustannus.

Abrahamsson, P., Salo, O., Ronkainen, J., & Valtion teknillinen tutkimuskeskus. (2002). *Agile software development methods : review and analysis*. VTT.

Allen, I. E., & Seaman, C. A. (2007). Likert scales and data analyses. *Quality Progress*, *40*, 64–65. http://rube.asq.org/quality-progress/2007/07/statistics/likert-scales-and-data-analyses.html

Alten. (2019). *4 Agile Best Practices for Mechanical and Electrical engineering project*. https://www.alten.com/mechanical-electrical-engineering-agile-method/. [Accessed: 8.6.2021].

Ashmore, S., & Runyan, K. (2014). *Introduction to Agile Methods*. Addison-Wesley Professional.

Atlassian. (2019). *Jira Software: How can non-software teams use Jira Software? e*. https://confluence.atlassian.com/confeval/jira-software-evaluator-resources/jira-software-how-can-non-software-teams-use-jira-software. [Accessed: 15.7.2021]

Atlassian. (2021a). *Atlassian Company b*. https://www.atlassian.com/company. [Accessed: 15.7.2021]

Atlassian. (2021b). *Atlassian Jira Software a*. https://www.atlassian.com/software/jira/guides/getting-started/overview#jira-software-hosting-options, [Accessed: 15.7.2021]

Atlassian. (2021c). *Atlassian Jira Workflow*. https://confluence.atlassian.com/jirasoftwareserver085/getting-started-with-jira-software-981155999.html, [Accessed: 15.7.2021]

Atlassian. (2021d). *Atlassian Main c*. https://www.atlassian.com/. [Accessed: 15.7.2021]

Atlassian. (2021e). *Atlassian Mixed Methodologies d*. https://www.atlassian.com/software/jira/agile#other-frameworks, [Accessed: 15.7.2021]

Atlassian. (2021f). *Jira Software Data Center and Server Documentation v. 8.5 e*.

https://confluence.atlassian.com/jirasoftwareserver085/jira-software-server-8-5-documentation-981155994.html, [Accessed: 15.7.2021]

Becerril, L., Hollauer, C., & Lindemann, U. (2018, December). Assessing the Agility of Teams within Mechatronic Product Development. *2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*. https://doi.org/10.1109/IEEM.2018.8607700

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., & Cunningham, W. (2001). *Manifesto for Agile Software Development*. https://agilemanifesto.org/, [Accessed: 23.4.2021]

Becker, B., Dawson, P., Devine, K., Hannum, C., Hill, S., Leydens, J., Matuskevich, D., Traver, C., & Palmquist, M. (2012). *Writing@CSU*. Colorado State University. https://writing.colostate.edu/guides/guide.cfm?guideid=60, [Accessed: 5.11.2021]

Benefield, G. (2008). Rolling Out Agile in a Large Enterprise [Proceeding]. *Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008)*, 461–461. https://doi.org/10.1109/HICSS.2008.382

Bohmer, A. I., Schweigert, S., Devecka, J., Grauvogl, C., Becerril, L., Bahrouni, Z., & Lindemann, U. (2017, June). Towards agile development of physical products a startup case study. *2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC)*. https://doi.org/10.1109/ICE.2017.8279872

Cohn, M. (2005). *Agile Estimating and Planning*. Prentice Hall PTR.

Creswell, J. W., & Plano Clark, V. L. (2011). *Designing and Conducting Mixed Methods Research* (2nd ed.). Sage.

Dyba, T., & Dingsoyr, T. (2015, May). Agile Project Management: From Self-Managing Teams to Large-Scale Development. *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. https://doi.org/10.1109/ICSE.2015.299

Feldmuller, D. (2018). Usage of agile practices in Mechatronics System Design Potentials, Challenges and Actual Surveys. *Proceedings of the 2018 19th International Conference on Research and Education in Mechatronics, REM 2018*, 30–35. https://doi.org/10.1109/REM.2018.8421803

Goevert, K., Gokdemir, A., Peitz, C., & Lindemann, U. (2017, December). Challenges of agile development implementation in mechatronic development processes. *2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*. https://doi.org/10.1109/IEEM.2017.8289981

Gustavsson, T., & Rönnlund, P. (2013). *Agile adoption at Ericsson hardware product development*. https://doi.org/10.13140/2.1.3781.3447

Hammarberg, M., & Sunden, J. (2014). *Kanban in Action*. Manning Publications Co.

Klein, T. P., Drescher, B., & Reinhart, G. (2016). Agile Engineering in Mechatronic Education: Interdisciplinary Development of a Mechatronic System Using Scrum. *2016 11th France-Japan & 9th Europe-Asia Congress on Mechatronics (MECATRONICS) /17th International Conference on Research and Education in Mechatronics (REM)*.

Komus, A. (2012). *Status Quo Agile*.

Ladas, C. (2009). *Scrumban: Essays on Kanban Systems for Lean Software Development*. Modus Cooperandi Press.

Less.works. (2021a). *About Less*. https://less.works/resources/about, [Accessed: 17.5.2021]

Less.works. (2021b). *About Less*. https://less.works/less/framework/introduction, [Accessed: 17.5.2021]

Less.works. (2021c). *LeSS Framework*. https://less.works/less/framework, [Accessed: 17.5.2021]

Less.works. (2021d). *Overall Retrospective*. https://less.works/less/framework/overall-retrospective, [Accessed: 17.5.2021]

Less.works. (2021e). *Sprint Planning One*. https://less.works/less/framework/Sprint-planning-one, [Accessed: 17.5.2021]

Less.works. (2021f). *Sprint Review*. https://less.works/less/framework/Sprint-review, Accessed: [17.5.2021]

Liminka, J. (2019). *Implementing Agile Methods in Large Scale Hardware Development Projects*. (Master's thesis). Vaasa University of Applied Sciences.

Middleton, P., & Joyce, D. (2012). Lean software management: BBC worldwide case study. *IEEE Transactions on Engineering Management*, *59*(1), 20–32. https://doi.org/10.1109/TEM.2010.2081675

Müller, M. M., & Padberg, F. (2002). *Extreme Programming from an Engineering Economics Viewpoint*.

O'Connor, T. E. (2021). *The Large-Scale Scrum (LeSS) framework*. Atlassian Agile Coach. https://www.atlassian.com/agile/agile-at-scale/less, [Accessed: 17.3.2021]

Ovesen, N. (2012). *The Challenges of Becoming Agile: Implementing and Conducting Scrum in Integrated ProductDevelopment (PhD Thesis)*. Aalborg University.

Radigan, D. (2021). *Story points and estimation*. Atlassian Agile Coach.

Reddy, A. (2015). *The Srumban [R]Evolution : getting the most out of Agile, Scrum, and lean Kanban*. Addison-Wesley.

Rehkopf, M. (2021a). *Agile Coach: Learn Issues with Jira Software*. Agile Tutorials. https://www.atlassian.com/agile/tutorials/issues

Rehkopf, M. (2021b). *Atlassian Agile Coach: Kanban vs. Scrum: which agile are you?* https://www.atlassian.com/agile/kanban/kanban-vs-Scrum

Reynisdottir, T. (2013). *Scrum in Mechanical Product Development: Case Study of a Mechanical Product Development Team using Scrum (Master's thesis)*. Chalmers University of Technology.

Royce, W. (1987). Managing the development of large software systems: concepts and techniques. *Proceedings of the 9th International Conference on Software Engineering*, 328–338.

Rubin, K. S. (2012). *Essential Scrum: a practical guide to the most popular Agile process*. Addison-Wesley Professional.

Saunders, M. N. K., Thornhill, A., & Lewis, P. (2019). *Research Methods for Business Students* (8th ed.). Pearson Education, Limited, .

Schwaber, K. (1997). SCRUM Development Process. In C. and M. J. and P. P. and H. G. Sutherland Jeffand Casanave (Ed.), *Business Object Design and Implementation* (pp. 117–134). Springer London.

Schwaber, K., & Sutherland, J. (2020, November). *The 2020 Scrum Guide*. https://www.Scrumguides.org/Scrum-guide.html

Scrum.org. (2021). *Scrum Framework*. https://www.Scrum.org/resources/Scrum-framework-poster

Sjoberg, D. I. K., Johnsen, A., & Solberg, J. (2012). Quantifying the Effect of Using Kanban versus Scrum: A Case Study. *IEEE Software*, *29*(5). https://doi.org/10.1109/MS.2012.110

Sommerville, I. (2016). *Software Engineering* (10th ed.). Pearson Education Limited.

State of Agile. (2020). *14th Annual State of Agile Report*. https://stateofagile.com/

State of Agile. (2021). *15 th State of Agile Report*. https://stateofagile.com/, [Accessed: 30.8.2021]

Stellman, A. (2020). *What Is Scrumban?* O'Reilly Media.

Suddaby, R. (2006). From the Editors: What Grounded Theory is Not. *Academy of Management Journal*, *49*(4). https://doi.org/10.5465/amj.2006.22083020

Takeuchi, H., & Nonaka, I. (1986). *The New New Product Development Game*. https://hbr.org/1986/01/the-new-new-product-development-game, [Accessed 6.4.2021]

Tashakkori, A., & Teddlie, C. (2010). *SAGE Handbook of Mixed Methods in Social & Behavioral Research*. SAGE Publications, Inc. https://doi.org/10.4135/9781506335193

Thompson, K. (2011). *Scrum in the Enterprise: How to Manage the Common Issues of Scrum Projects*. cPrime.

Thompson, K. (2015). *Agile Processes for Hardware Development*.

Tuomi, J., & Sarajärvi, A. (2002). *Laadullinen tutkimus ja sisällönanalyysi*. Tammi.

Vehkalahti, K. (2014). *Kyselytutkimuksen mittarit ja menetelmät*. Helsingin yliopisto. https://doi.org/10.31885/9789515149817

Vodde, B., & Larman, C. (2016). *Large-Scale Scrum: More with LeSS* [Book]. Addison-Wesley Professional, pp. 368.

Womack, J. P., Jones, D. T., & Roos, D. (1990). *The Machine That Changed the World*. Free Press.

Yin, R. K. (2017). *Case Study Research and Applications: Design and Methods* (6th ed.). Sage.

APPENDIX A: SURVEY QUESTIONS

1. How many years experience do you have in product development? *

- ○ Not at all
- ○ Less than a year
- ○ 1 - 3 years
- ○ 4 - 6 years
- ○ 7 - 10 years
- ○ 10+ years

2. How much experience do you have with agile methods in practice? *

- ○ Not at all
- ○ Less than a year
- ○ 1 - 3 years
- ○ 4 - 6 years
- ○ 7 - 10 years
- ○ 10+ years

3. What agile methods have you used? *

- ☐ Scrum
- ☐ Kanban
- ☐ Scrumban
- ☐ Lean Development
- ☐ Extreme Programming (XP)
- ☐ Crystal
- ☐ Feature-Driven Development (FDD)
- ☐ Dynamic Systems Development Method (DSDM)
- ☐ Other

4. How much do you estimate your knowledge of agile methods? *

- ◯ Extremely familiar
- ◯ Very familiar
- ◯ Somewhat familiar
- ◯ Not so familiar
- ◯ Not at all familiar

5. Please, answer based on your current experience with agile methods. *

	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
Agile methods increases the effectiveness of product development	◯	◯	◯	◯	◯
Agile development increases the quality of the product	◯	◯	◯	◯	◯
Agile methods increases the transparency of the project	◯	◯	◯	◯	◯
Agile methods shortens the product delivery time	◯	◯	◯	◯	◯
Agile methods makes work more fun	◯	◯	◯	◯	◯
Agile methods makes work more hectic	◯	◯	◯	◯	◯
Agile methods makes work more organized/planned	◯	◯	◯	◯	◯
Agile methods increases collaboration	◯	◯	◯	◯	◯
Agile methods increases the autonomy of your team	◯	◯	◯	◯	◯

6. How important you consider agile methods in your team in the future? *

- ◯ Extremely important
- ◯ Somewhat important
- ◯ Neutral
- ◯ Somewhat not important
- ◯ Extremely not important

7. Please, answer based on your current experience *

	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
Team size is too large	◯	◯	◯	◯	◯
There should be separated daily scrums for each team	◯	◯	◯	◯	◯
There is enough information shared inside my team	◯	◯	◯	◯	◯
There is enough information shared between different teams	◯	◯	◯	◯	◯
There is enough collaboration inside my team	◯	◯	◯	◯	◯
There is enough collaboration between different teams	◯	◯	◯	◯	◯
Team has all the necessary skills to work successfully	◯	◯	◯	◯	◯

8. Anything else you would like to share about teams and collaboration?

Enter your answer

9. Please, answer based on your current experience. *

	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
Daily Scrum is helpful	○	○	○	○	○
Daily Scrum supports my working	○	○	○	○	○
Tasks are clear and easy to understand	○	○	○	○	○
I can finish my tasks in a given sprint	○	○	○	○	○
Fixed length sprints are useful	○	○	○	○	○

10. How many times do you use JIRA in a work week? *

- ○ Every day
- ○ A few times a week
- ○ About once a week
- ○ Less than once a week
- ○ Never

11. How much time do you spend using JIRA in a work week? *

- ○ None
- ○ Less than 15 minutes
- ○ 15 - 30 minutes
- ○ 30 - 60 minutes
- ○ More than 60 minutes

12. Please, answer based on your current experience *

	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
Usage of JIRA has been helpful	○	○	○	○	○
Usage of JIRA is difficult or confusing	○	○	○	○	○
Usage of JIRA is unnecessary	○	○	○	○	○
Usage of JIRA is time-consuming and takes time off from other work	○	○	○	○	○
Usage of JIRA has made the project transparent and more trackable	○	○	○	○	○
Usage of JIRA has improved my time management	○	○	○	○	○

13. What do you consider as the main benefits of using JIRA?

Enter your answer

14. What do you consider as the main challenges of using JIRA?

Enter your answer

15. Anything else you would like to comment about using agile methods and JIRA?

Enter your answer

16. In what year were you born? (enter 4-digit birth year; for example, 1976) *

Enter your answer

17. What is your current education level? Choose the highest completed degree. *

- ◯ Upper secondary school or vocational school
- ◯ Bachelor's degree or equivalent
- ◯ Master's degree or equivalent
- ◯ Licentiate or doctoral degree

18. How long have you worked at *

- ◯ Less than 1 year
- ◯ 1 - 2 years
- ◯ 3 - 5 years
- ◯ 6 - 7 years
- ◯ 8 - 10 years
- ◯ 10+ years

19. I work in a (team) *

- ◯ SW
- ◯ HW
- ◯ ME
- ◯ Validation
- ◯ Other

20. What is your job role? Pick one that describes your current role the best. *

- ◯ Team leader
- ◯ Manager / Management
- ◯ Engineer / Developer
- ◯ Intern
- ◯ Scrum Master
- ◯ Product Owner
- ◯ Other