VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Jyri Liminka

Implementing Agile Methods in Large Scale Hardware Development Projects

School of Technology
2019

VAASAN AMMATTIKORKEAKOULU
Ylempi ammattikorkeakoulututkinto, Projektinhallinta

TIIVISTELMÄ

Tekijä	Jyri Liminka
Opinnäytetyön nimi	Implementing Agile Methods in Large Scale Hardware Development Projects
Vuosi	2019
Kieli	Englanti
Sivumäärä	73
Ohjaaja	Adebayo Agbejule

Tämän opinnäytetyön tavoitteena on tutkia, miten ketterä projektimalli voidaan ottaa käyttöön laitteistokehityksessä ja mitä haasteita siihen liittyy. Tutkitaan myös ketterien menetelmien skaalausta isommille projekteille ja ohjelmille, joissa työskentelee useita tiimejä eri aloilta.

Teoreettinen viitekehys muodostettiin kirjallisuuskatsauksen avulla, joka kattoi erilaisia projektimalleja, ketterän mallin käyttöä laitteistokehityksessä ja menetelmiä ketterän mallin skaalausta varten. Tapaustutkimus tehtiin yritykselle, jossa oli hiljattain siirrytty käyttämään Scrum-prosessia laitteistokehityksessä. Tapaustutkimuksen tiedot kerättiin tekemällä kuusi henkilökohtaista haastattelua yrityksen kahdessa laitteistokehitysprojektissa työskentelevien Project Leadereiden, Product Ownereiden ja Scrum Mastereiden kanssa.

Tapaustutkimuksen tulokset tukevat aikaisemmista tutkimuksista saatuja tietoja. Riittävä koulutus ennen ketterän mallin käyttöönottoa ja valmennus siirtymävaiheen aikana todettiin tärkeiksi mahdollistamaan sujuva siirtyminen uuteen projektimalliin. Testaus määriteltiin pullonkaulaksi ketterässä laitteistokehityksessä ja investointeja testausautomaatioon tulisi harkita tämän ongelman lievittämiseksi. Myös täysi tuki organisaatiolta ketterään malliin siirryttäessä todettiin tärkeäksi. Lisäksi haastatteluista saatiin muita huomioita ja parannusehdotuksia.

Tapaustutkimuksen kohteena ollut yritys voi käyttää tämän opinnäytetyön löydöksiä prosessiensa arviointiin ja mahdollisten muutosten tekoon niihin. Muut yritykset, jotka ovat suunnittelemassa ketterään malliin siirtymistä fyysisten tuotteidensa kehityksessä, voivat löytää tutkimuksesta hyödyllistä tietoa ja täten olla paremmin valmistautuneita sen mukanaan tuomiin haasteisiin.

Avainsanat ketterät menetelmät, laitteistokehitys, projektimalli, scrum

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Master of Engineering in Project Management

ABSTRACT

Author	Jyri Liminka
Title	Implementing Agile Methods in Large Scale Hardware Development Projects
Year	2019
Language	English
Pages	73
Name of supervisor	Adebayo Agbejule

The objective of this thesis was to examine how to implement an agile project model for hardware development and what the challenges related to it are. Scaling agile methods for larger projects and programs, with multiple teams from different fields was studied as well.

The theoretical framework was formed from a literature review, covering different project models, an agile model in hardware development and methods for scaling an agile model. A case study was made for an organization where they had recently started using the Scrum process in hardware development. Data for the case study was collected by conducting six individual interviews with Project Leaders, Product Owners and Scrum Masters from two hardware projects in the company.

The case study findings supported the information obtained from the existing literature. Adequate training for the personnel before implementing the agile model and coaching during the transition period was found to be important to allow for a smooth transition to the new project model. Testing was determined to be a bottleneck in agile hardware development and investing in test automation should be considered to mitigate this issue. Full support from the organization when transitioning to the agile model, was deemed important as well. Additionally, other observations and improvement ideas were gained from the interviews.

The findings from this thesis can be used by the case study company to evaluate their processes and make possible adjustments to them. Other organizations planning to switch from a traditional project model to an agile project model in their physical product development may find useful information from the research and thus be better prepared for the challenges it brings.

Keywords agile, hardware development, project model, scrum

TABLE OF CONTENTS

1 INTRODUCTION 8

1.1 Research objectives and research questions 9

1.2 Structure of the thesis 10

2 LITERATURE REVIEW 11

2.1 Project management 11

2.2 Traditional project models 13

2.2.1 Drawbacks of traditional project models 15

2.3 Agile project models 16

2.4 Extreme Programming 19

2.5 Scrum 20

2.5.1 Scrum roles 21

2.5.2 Scrum events 22

2.6 Hybrid project model 25

2.7 Agile methods in hardware development 26

2.8 Scaling agile methods 30

3 THEORETICAL FRAMEWORK 34

4 METHODOLOGY 38

4.1 Research method 38

4.2 Case study 38

4.3 Collecting and analyzing data 38

4.4 Validity and reliability 39

5 FINDINGS 41

5.1 Description of the projects 41

5.2 Team compositions and responsibilities 42

5.3 Collaboration with other teams 45

5.4 Feedback from the customer 46

5.5 Planning tools 47

5.6 Planning methods 47

5.7 Work effort estimations 48

5.8 Meeting practices 49

5.9 Development ... 53
5.10 Testing ... 55
5.11 Positive experiences from Scrum adoption ... 56
5.12 Challenges in implementing Scrum ... 58
6 DISCUSSION ... 60
6.1 First research question ... 60
6.2 Second research question ... 62
7 CONCLUSION ... 64
7.1 Summary ... 64
7.2 Theoretical contribution ... 65
7.3 Practical contribution ... 65
7.4 Limitations of the research ... 65
7.5 Suggestions for future research ... 66
REFERENCES ... 67

LIST OF ABBREVIATIONS

3D	= Three dimensional
ART	= Agile Release Train
CPM	= Critical Path Method
DAD	= Disciplined Agile Delivery
DevOps	= Development and Operations
HIP	= Hardening, Innovation and Planning
IPMA	= International Project Management Association
LeSS	= Large Scale Scrum
NDA	= Non-Disclosure Agreement
PERT	= Program Evaluation Review Technique
PI-Planning	= Program Increment Planning
PMBOK	= Project Management Body of Knowledge
PMI	= Project Management Institute
SAFe	= Scaled Agile Framework
VoC	= Voice of Customer
XP	= Extreme Programming

LIST OF FIGURES AND TABLES

Figure 1.	The waterfall development model.	p. 14
Figure 2.	The Stage-Gate model.	p. 14
Figure 3.	A simple example of a V-model.	p. 15
Figure 4.	An agile project model.	p. 18
Figure 5.	The Scrum model.	p. 24
Figure 6.	An example of the Agile-Stage-Gate hybrid project model.	p. 26
Table 1.	Comparison of traditional and agile project models.	p. 36

LIST OF APPENDICES

Appendix 1.	List of the interviews.
Appendix 2.	The main interview questions.

1 INTRODUCTION

The environment for product development has become more unpredictable and more competitive, requiring companies to focus on change management and getting their products to the market as fast as possible, without compromising quality. In the past years, agile project models have become the standard models used in software development projects. They are based on frequent, incremental product releases in order to get feedback from the customers, which allows for early discovery of problems and makes it possible to direct the product development to the most important areas.

A survey with 57,075 respondents to the question concluded that 85.9% of professional software developers were using agile methods in their work and 63.2% of the developers were using Scrum in 2018 (Stack Overflow 2018). However, the introduction of agile methods to hardware development is a quite recent phenomenon and the traditional waterfall model is still the preferred method of managing hardware development projects. There are many reasons why developing hardware incrementally is challenging, the most obvious one being that building a physical product and testing it is a considerably more time-consuming procedure, than creating and testing software. This makes it difficult to make iterative product releases and to get customer feedback frequently. Despite the challenges, several studies have determined that it is possible to develop hardware incrementally and it is beneficial over the traditional style as well (Cooper and Sommer 2016b; Friis Sommer et al. 2015; Gustavsson & Rönnlund 2013; Karlstrom & Runeson 2005; Punkka 2016; Reynisdóttir 2013; Serrador & Pinto 2015).

The research and development department of the case study company had already been utilizing agile methods in software development for a few years prior to this study and they had recently introduced them to hardware development as well. The organization’s goal was to decrease the time to market for products by being able to react quicker to changing market conditions and customer requirements. Their solution was a hybrid project model, where Scrum was utilized for development and testing. The hardware department’s iteration cycles were in the same cadence with the software department to allow better co-operation between them, as the product was highly dependent on the integration of hardware and software. The company wanted to know how well the arrangement was working for the development teams and if changes were needed to the current system.

1.1 Research objectives and research questions

The objective of this research is to find out how suitable an agile project model is for hardware development, how well the Scrum model is working for a large scale product development program in the hardware development department of the case study company and what possible improvements to the processes could be beneficial for the future. This thesis intends to answer the following two research questions:

First research question

What aspects should be considered when transitioning from a traditional project model to an agile project model in hardware development projects?

Hardware development differs from software development in many ways and introduces new challenges when an iterative design process is being adopted. Suitable methods have to be identified for enabling efficient product development and for overcoming the obstacles.

Second research question

How to manage communication between different projects and Scrum teams in a large scale product development program?

The Scrum process emphasizes the importance of frequent communication between team members and between the team and the customers. However, when scaling into bigger projects with multiple teams and into bigger programs involving multiple projects, new solutions are needed to ensure a sufficient flow of information between all participants.

1.2 Structure of the thesis

In chapter two the existing literature is reviewed to gain a profound understanding of the management methods used in product development projects. The chapter starts by describing project management in general and continues by examining different project models, followed by analyzing the advantages and challenges which agile models bring to hardware development. Moreover, frameworks for scaling agile methods are studied.

In chapter three a theoretical framework is formed according to the information gathered from the literature.

Chapter four explains the methodology used in the case study, examines the methods for collecting and analyzing the data, and assesses the reliability and validity of the research.

Chapter five presents the findings from the interviews and the company documents by separating the information into different categories.

In chapter six the findings are analyzed and compared to the literature of the topic area. The research questions are discussed and answered, based on the evaluation of the results.

Chapter seven concludes the thesis by summarizing the results and analyzing the limitations of the research. It also explains the theoretical and practical contributions of the study and gives suggestions for future research.

2 LITERATURE REVIEW

The literature review presents the relevant information gathered from previous studies concerning project management methods in product development. It begins by outlining the basic principles of project management, followed by introducing different project models and how they have evolved. Scrum is described more in depth, as it is the main subject for this thesis. Agile methods for hardware development are discussed by concluding findings from different studies, explaining the advantages and challenges an agile model brings when compared to traditional project models. Lastly, the scaling possibilities of agile methods are explored by introducing the existing scaling frameworks and by examining related studies.

2.1 Project management

Project management is a relatively recent method of operation in companies. Program Evaluation Review Technique (PERT) and the Critical Path Method (CPM) were invented in 1958, which was arguably the starting point for modern project management. The International Project Management Association (IPMA) was founded in 1965 and the Project Management Institute (PMI) was founded in 1969. IPMA promotes project management mainly in Europe and Asia, while PMI is based in the U.S. PMI publishes The Project Management Body of Knowledge (PMBOK), which is a collection of best practices for project management. (Seymour & Hussein 2014)

PMI defines a project as follows: “A project is a temporary endeavor undertaken to create a unique product, service, or result. The temporary nature of projects indicates that a project has a definite beginning and end. The end is reached when the project’s objectives have been achieved or when the project is terminated because its objectives will not or cannot be met, or when the need for the project no longer exists (PMI 2013a, 2).”

The definition of project management according to Kerzner (2013, 4) is: “Project management is the planning, organizing, directing, and controlling of company resources for a relatively short-term objective that has been established to complete specific goals and objectives. Furthermore, project management utilizes the systems approach to management by having functional personnel (the vertical hierarchy) assigned to a specific project (the horizontal hierarchy)."

PMI (2013a, 60) describes 47 separate processes for project management and these processes have been categorized into five groups:

- Initiation
- Planning
- Execution
- Monitoring and controlling
- Closing

Furthermore, each of the 47 processes is linked to one of the ten project management knowledge areas:

- Project Integration Management
- Project Scope Management
- Project Time Management
- Project Cost Management
- Project Quality Management
- Project Human Resource Management
- Project Communications Management
- Project Risk Management
- Project Procurement Management
- Project Stakeholder Management

As an example, changes are handled in a process called *"Perform Integrated Change Control"*, which belongs under *"Monitoring and controlling"* process group and is linked to *"Project Integration Management"* knowledge area. It is the process of reviewing all change requests by the project manager, change control board or other responsible group or individual. The change requests can come from stakeholders or they can be produced by multiple other processes at any time during the project lifecycle. The changes can be approved, rejected or put on hold. All change requests are documented in a change log.

A project manager should be able to identify requirements, communicate with stakeholders to address their needs and find a balance between different project constraints. The primary constraints are time, cost and performance, where the performance can be scope, quality or technology. These are called the triple constraints, which have been traditionally used as criteria for measuring project success. Secondary constraints include resources, risks, value, and image/reputation among others. The constraints compete with each other and compromises are usually necessary to complete the project successfully. Different projects and different phases of the projects might require prioritizing of one constraint over another, depending on the situation. (Kerzner 2013, 8; PMI 2013a, 5)

Kerzner (2013, 3) determines that project management has been successful when the project objectives have been achieved within the allocated time and cost limits, at the wanted performance and technology levels. In addition, efficient and effective use of assigned resources and customer acceptance are required.

2.2 Traditional project models

Product development projects have been traditionally executed by following a linear progression, in which the planning is finished before the actual development begins and testing is done after completing the development phase. This practice was originally described by Dr. Winston W. Royce in 1970 (Royce 1970) and later the process became known as *the waterfall development model*. The structure resembles a waterfall, where every step is completed before moving on to the next one, as illustrated in Figure 1.

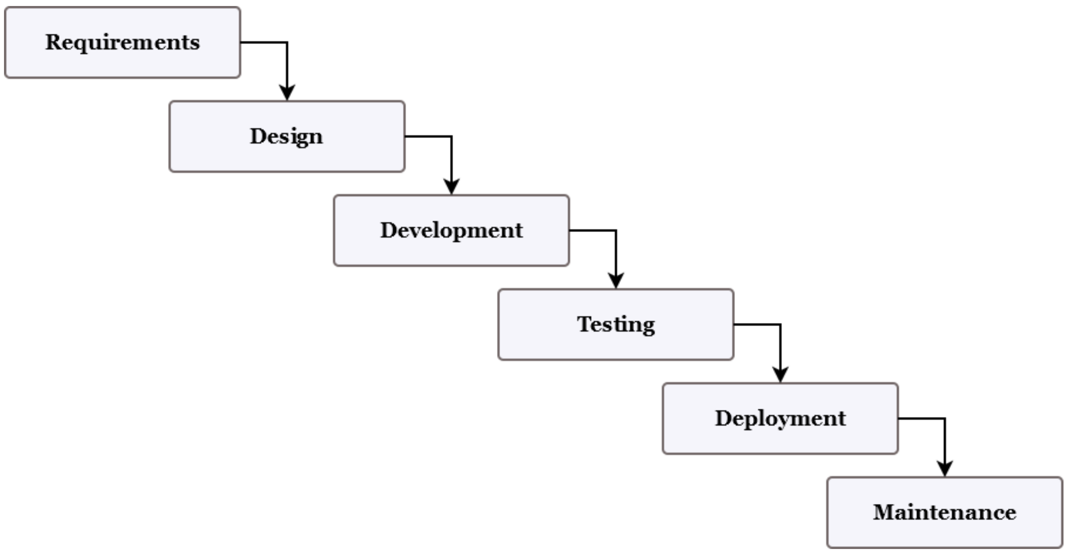

Figure 1. The waterfall development model.

The Stage-Gate model was developed by Robert G. Cooper in the 1980s to improve the effectiveness and quality of the product development process. This is a more refined version of the waterfall model and the development still happens in the same linear fashion. The system divides the whole project from idea to launch in typically four to seven stages and each stage is followed by a gate, which is the point of deciding if the project is worth continuing (see Figure 2). A determined set of deliverables must be ready at every gate and their quality is checked before the project can proceed to the next stage. The number of stages can vary, depending on the project size and complexity. (Cooper 1990; Cooper 2014).

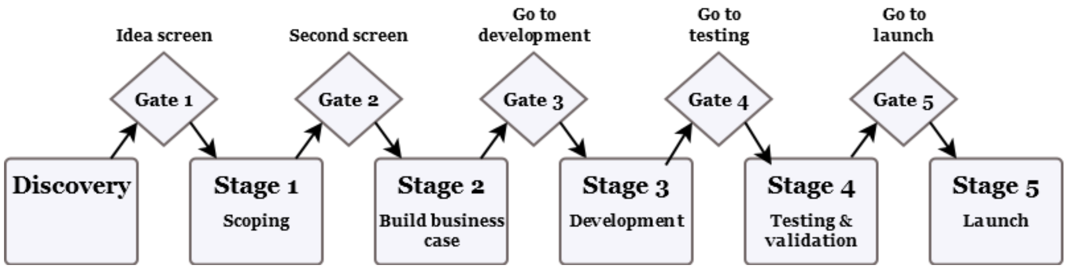

Figure 2. The Stage-Gate model.

The V-Model is an adaptation of the waterfall model as well and follows a similar sequential process, but the emphasis is on testing. The planning and designing are performed in phases, starting from requirements and high-level plans, advancing down the left side of the V-shape to more detailed levels. The verification procedures for each level are planned at the same time. The implementation starts after everything has been planned and designed, including the tests. After the implementation is completed, the testing starts from the most detailed level and advances up the right side of the V-shape to higher levels, according to the verification and validation plans created for each level (see Figure 3). (Mathur & Malik 2010; Munassar & Govardhan 2010)

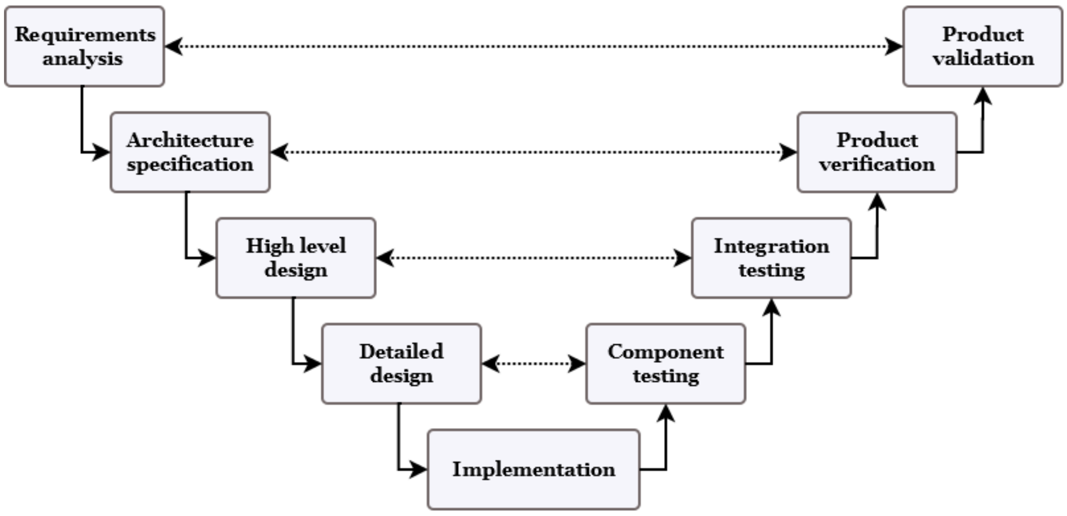

Figure 3. A simple example of a V-model.

2.2.1 Drawbacks of traditional project models

The waterfall model and the more advanced versions of it have been utilized by organizations for decades, but many of them have been starting to look for better solutions to keep up in the current competitive environment. The sequential nature of the development limits the model's effectiveness when fast time to market is critical and change management becomes an increasingly important factor for project performance.

One of the main disadvantages of traditionally operated projects is that the most time-consuming phase is the planning phase, which includes the engineering of requirements. A lot of the already approved requirements can ultimately be dismissed due to altering circumstances while the creation process is still ongoing, and implementing changes takes a long time (Cervone 2011; Petersen et al. 2009).

This is a significant issue as efficient change control is recognized as one of the key success factors for projects (Cooke-Davies 2002; Lechler 2002). The design and development work for the project is stalled, while the change control process is underway, which is inefficient and slows down the progress.

Another shortcoming of the waterfall model is that verification and validation are started only after the product is finished. It is usually much more difficult to fix defects that come up when the product is already completed, in comparison to finding the issues in an earlier phase of the development (Royce 1970; Petersen et al. 2009; Petersen & Wohlin 2010). This means that the cost of correcting problems rises towards the end of the project. When a crucial component is determined as faulty or inadequate for the product, also other parts might have to be substituted to maintain interoperability. The project lead time and costs can increase dramatically or even double in worst situations, where the product has to be redesigned and the requirements need to be revised (Royce 1970). Additionally, when there have already been delays in the project, there is a risk that the verification and validation have to be rushed to meet the deadlines, which is likely to negatively affect the quality of the product (Petersen et al. 2009).

2.3 Agile project models

In 2001, seventeen software development professionals representing different schools in development methods, like Scrum, Extreme Programming and Crystal, sat together to talk about different ways of creating software and aimed to find common ground. The result of that meeting was “Manifesto for Agile Software Development” (Beck et al. 2001):

“We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

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

That is, while there is value in the items on the right, we value the items on the left more."

In addition, the participants specified the following twelve principles:

1. "Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly."

Agile practices were created to address the weaknesses in traditional methods of development, which include the tedious planning phase and the problems related to late testing. Originally these ideas were directed to software development, but later they have been successfully applied to other sectors as well (Gustavsson & Rönnlund 2013; Punkka 2012; Reynisdóttir 2013).

Agile models use an incremental development style where the work is done in short iterations (see Figure 4), every iteration produces something functional and testing is included in the process from the beginning (Highsmith 2002). Planning is also done incrementally instead of using the traditional method of planning the whole project in advance. The short cycles enable early discovery of possible mistakes made in planning and allow the developers to get quick feedback from the testers. Furthermore, agile principles encourage frequent dialogue between different parties involved in the project, including the customers, to increase the flow of information and reduce misunderstandings. In a large study by PMI (2013b), involving 742 project management practitioners, 148 executive sponsors, and 203 business owners, effective communications to all stakeholders was determined as the most important success factor in project management. Increased communication is considered as one of the main advantages of an agile model, as the team members maintain a good understanding what kind of tasks and issues others are dealing with, which leads to quicker detection of problems and possible solutions to them (Begel & Nagappan 2007; Paasivaara et al. 2008). Feedback from the customer can change priorities and have an impact on the direction of the project when the gained information is taken into account in the next iteration cycle.

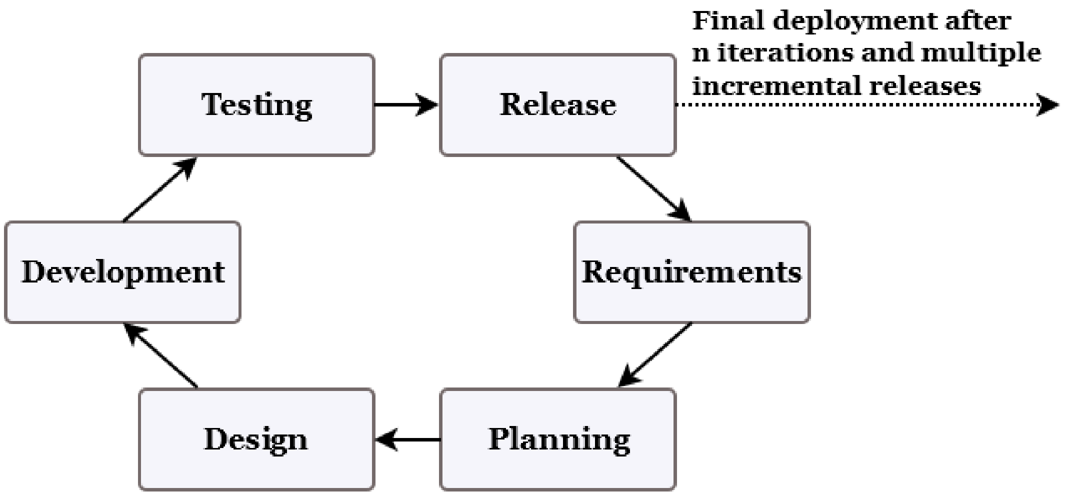

Figure 4. An agile project model.

2.4 Extreme Programming

Extreme Programming (XP) is an iterative development model created in the late 1990s by Kent Beck, Ron Jeffries, and Ward Cunningham. It is designed specifically for creating software effectively by combining ideas from different frameworks and models. The work is done small teams, typically with less than 10 members and the customer is sitting with the team on site. The customer decides which features (stories) they want to have implemented for each release and when the release should happen. The developers have estimated the required time to implement each story and the customer selects the most important ones, which fit in the desired schedule. This is called the **planning game** in XP terms. New requirements from the customer are added to the list of stories and they can be selected for the next release if their priority is high enough for the customer. (Beck 1999; Paulik 2001)

The developers divide the stories into smaller tasks and create automated unit tests for them in advance, which can be used to verify that the task is complete. The created tests are run repeatedly until the end of the project. The team members choose the tasks they want to complete during the iteration, aiming to share the workload evenly across the team. During the development, the customer specifies the functional tests for verifying each story, the test cases can be implemented by the team or by the customer. (Beck 1999; Paulik 2001)

One of the methods in XP is **pair programming**, where two developers work at the same computer. The practice is considered to be more productive than programming individually, as the developers learn from each other. XP emphasizes simplicity in the design and **refactoring** is used to optimize and simplify the code, without changing its behavior. **Collective ownership** means that any developer can refactor any part of the code when room for improvement is detected. **Continuous integration** is the principle of integrating new code from each task to the system very soon after the code has been created and it has passed the unit tests. After integration, every existing test must pass or the integration is canceled. At the end of the iteration, all the previously created test cases and the functional tests provided by the customer have to be passed. The next iteration starts again with the planning game, where the customer selects the most important stories, which can be finished during the selected time period. (Beck 1999; Paulik 2001)

2.5 Scrum

The ideas behind Scrum were first introduced in 1986 (Takeuchi & Nonaka 1986) and further developed in the 1990s (Schwaber 1997; Beedle et al. 1999). The concept is a product development framework based on flexibility, continuous learning and the ability to deal with changes as efficiently as possible, rather than trying to avoid them. The project team is constantly adapting as conditions change and new data becomes available (Schwaber 1997). Communication is one of the main focus points and various techniques are used to improve information flow inside the project. Knowledge sharing between team members is encouraged and stakeholders are regularly updated about the state of the project (Schwaber 1997; Schwaber & Sutherland 2017). The main purpose of Scrum is to make product development more efficient and decrease the time to market while ensuring that customers get the product they desire (Cervone 2011).

The work is done in iterations called **Sprints**, which are typically between one and four weeks long and there are no breaks between them. The length of the Sprints stay constant during the project and will be changed only if there is an apparent need for it, but never in the middle of a Sprint. Each Sprint is meant to produce completed deliverables, which can be for example finished components for the product under development, documentation or completed tests. Together these deliverables form a **product increment**, which is a jointly agreed goal for the Sprint, with a clear definition of done. The ideal product increment would be a potentially shippable product, but in practice, this is not always a realistic goal. (Deemer et al. 2012; Schwaber 1997; Schwaber & Sutherland 2017)

All requirements for the product, sorted by priority, are listed in the **Product Backlog**. The Product Backlog items are constantly updated when new information becomes available, for example when a newly found defect has to be fixed or when new functionality is requested by the customer. Items with higher priorities have more detailed descriptions and effort estimations, which are refined during the Sprints. (Deemer et al. 2012; Schwaber & Sutherland 2017)

The **Sprint Backlog** is a collection of items, selected from the Product Backlog for the current Sprint and further divided into tasks. It serves as a plan on how to achieve the Sprint goal. Each task includes a description of the work needed and an estimate for the amount of time it takes to complete the task. Each item has a definition of what is required for the item to be determined as done. (Cervone 2011; Schwaber & Sutherland 2017)

The team members update the Sprint Backlog daily by reporting the amount of time they have used for each task and by estimating how much time is still needed to complete them. Original effort estimations can be changed by the team if a task proves to be easier or more complex than expected. These updates are visualized in the Sprint Burndown Chart, which is a graph of the amount of work left for the team in the current Sprint (Cervone 2011; Deemer et al. 2012). The chart gives an overall picture of how well the Sprint is progressing and if the line is not descending at a reasonable pace, it signals that maybe something should be changed in the planning or in the ways of working.

2.5.1 Scrum roles

Scrum Team consists of Product Owner, Scrum Master and typically five to nine professionals, who are usually developers and testers in product development projects. The professionals will be referred to as “the team” from this point forward.

The team is cross-functional, meaning it includes people with different skills and has the capability to build the product with the combined expertise of the team members. Moreover, it is self-organizing, which implies that no one is actively managing the work done by the team members. They independently select how many tasks they take for the Sprint and decide how they are going to complete them. The optimal team size is from five to nine members, larger teams require too much coordination. (Cervone 2011; Deemer et al. 2012; Schwaber & Sutherland 2017)

The Product Owner is the person responsible for creating and managing the Product Backlog, so it is important for him or her to thoroughly understand the product and needs of the customers. He or she acts as a link between the team and the stakeholders and makes sure that the requirements are prioritized correctly to ensure that the team's capabilities are always focused on the most important items. Additionally, the Product Owner is an internal customer for the team. He or she evaluates the outcomes of each Sprint and makes decisions on possible changes to the product and what should be the target for future Sprints. (Cervone 2011; Deemer et al. 2012; Schwaber & Sutherland 2017)

The Scrum Master caters the team, the Product Owner and the organization by advising everyone involved in matters concerning the Scrum process. He or she arranges and conducts Scrum meetings, eliminates obstructions hampering the team's productivity and ensures that the team members are able to do their work as effectively as possible. The Scrum Master serves more as a coach and not as a manager or an authority to the team, as he or she does not supervise the work and is not in charge of the development. Anyone capable can take the Scrum Master role, excluding the Project Owner. For example, a member of the team can act as the Scrum Master on the side while continuing development or testing work for the project, but especially bigger teams might benefit from a dedicated Scrum Master. (Deemer et al. 2012; Schwaber & Sutherland 2017)

2.5.2 Scrum events

There are four main recurring meetings in the Scrum process, which are Sprint Planning, Daily Scrum, Sprint Review and Sprint Retrospective. All of these meetings are chaired by the Scrum Master. There are also Product Backlog refinement sessions, in which the Scrum Master is not required to attend, but they can be more irregular events. During these sessions, the team gives the Product Owner estimates for the time it takes to complete each Product Backlog item, which can then be added to a future Sprint Backlog. If an item is stated as too big, it is divided into smaller parts and the definition of done is determined for each task. The refinement can be done when needed and it does not have to be scheduled to always happen at regular intervals. (Cervone 2011; Deemer et al. 2012; Schwaber & Sutherland 2017)

The Sprint starts with **Sprint Planning**, where the team draws items from the Product Backlog to the Sprint Backlog. The planning is divided into two parts. In the first part, the Product Owner discusses with the team about the goal of the Sprint and what is possible to accomplish in the time period. This gives everyone an understanding of the overall objective and awareness of possible limitations that have to be taken into account. In the second part of the Sprint Planning, each member of the team chooses how many tasks they will take for the upcoming Sprint, trying to take the maximum amount of work they believe to be able to accomplish during the Sprint. The tasks are preferably selected in order of priority, but items with lower priority can be added to the Sprint as well, for example when it is seen as more efficient to complete a lower priority item concurrently with another task. The duration of the Sprint Planning meeting is usually a few hours, depending on the length of the Sprint and the size of the team. (Cervone 2011; Deemer et al. 2012; Schwaber & Sutherland 2017)

Daily Scrum is a short stand-up meeting held every day, which should not take more than 15 minutes, with the purpose to keep everyone updated on the Sprint progress. The Product Owner is not required to attend the meeting. Each team member provides answers to the following three questions:

1. What have I done since the last Daily Scrum?
2. What will I do before the next Daily Scrum?
3. Are there any obstacles or problems affecting the advancement of the Sprint?

The answers to the questions are kept brief and if a more detailed discussion is needed, it happens after the Daily Scrum. The point of the meeting is that the team members are able to share information with each other and learn about matters that can influence their work. (Cervone 2011; Deemer et al. 2012; Schwaber & Sutherland 2017)

Sprint Review is scheduled to the end of each Sprint, with the purpose of reviewing the results of the past Sprint. The attendees can include customers and other project stakeholders in addition to the Scrum Team. The team members discuss with the Product Owner about what has been completed during the Sprint and give demonstrations of the work that has been completed. Possible changes and additions to the Product Backlog are also considered. (Deemer et al. 2012; Schwaber & Sutherland 2017)

Sprint retrospective meeting, held after the Sprint Review, focuses on improving the Scrum process and the ways of working. Scrum Master and the team analyze the possible issues encountered, seeking to learn about avoiding problems in the future. Furthermore, they reflect on positive aspects and on how to continue on a successful path. (Deemer et al. 2012; Schwaber & Sutherland 2017)

The next iteration of Scrum starts with the Sprint Planning meeting on the following working day after the Sprint Review and the Sprint Retrospective, usually after a weekend and the cycle repeats again from the beginning (see Figure 5). The Sprint resembles a mini project inside the project, as it has a planning phase, followed by development phase and it ends in reviewing the results and analyzing the lessons learned, preferably resulting in releasing the next version of some part of the product. (Schwaber & Sutherland 2017)

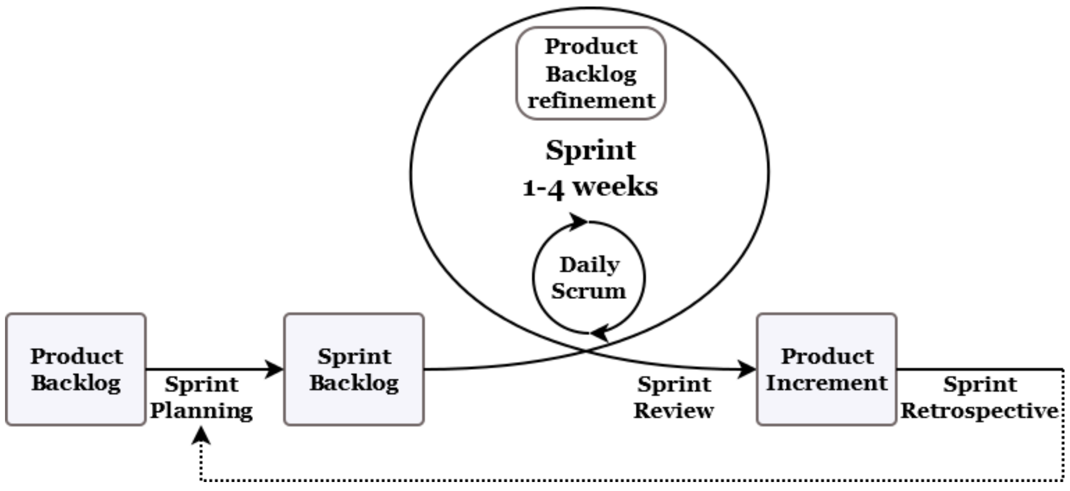

Figure 5. The Scrum model.

2.6 Hybrid project model

The Stage-Gate model has been evolving over the years and many companies have implemented their own versions of the system to better accommodate their specific needs (Cooper 2014). Other systems, like Lean and Six-Sigma, have also been integrated into the Stage-Gate model by different organizations to make it more suitable for the modern demands (Cooper 2008; Cooper 2014). Agile methods have been added into the mix correspondingly in recent years. The most common way of integrating agile methods into the Stage-Gate model is by using it in the development and testing phases of the project, but agile methods have been successfully utilized in other project phases as well (Cooper & Sommer 2016a). An example of the Agile-Stage-Gate hybrid project model is illustrated in Figure 6.

Positive results have been reported by many companies, after introducing agile processes into their projects. Serrador and Pinto (2015) found in their research, which included not only software projects but 1002 projects from various sectors that increased utilization of agile methods correlated with higher project success in all three of the measured categories: efficiency, stakeholder satisfaction, and project performance. Other studies have noticed increased efficiency and productivity when agile methods have been combined with the existing Stage-Gate systems (Cooper & Sommer 2016b; Friis Sommer et al. 2015). One major cause for the improved performance is better communication inside the team, which leads to fewer misunderstandings (Cooper & Sommer 2016b). The methods to achieve increased information flow include daily stand up meetings and organizing the office space in a way that allows the team members to sit close to each other (Karlstrom & Runeson 2006). In their earlier study, Karlstrom and Runeson (2005) observed a considerable increase in product quality in one of the case study organizations. The quality improvement was believed to be caused at least partly by the continuous integration performed during the agile development. This avoids the usual rush to integrate everything at once near the milestones, which is common in traditional project models. In the same study, the teams perceived to be able to manage their activities better, due to the work being divided into smaller, well-defined tasks.

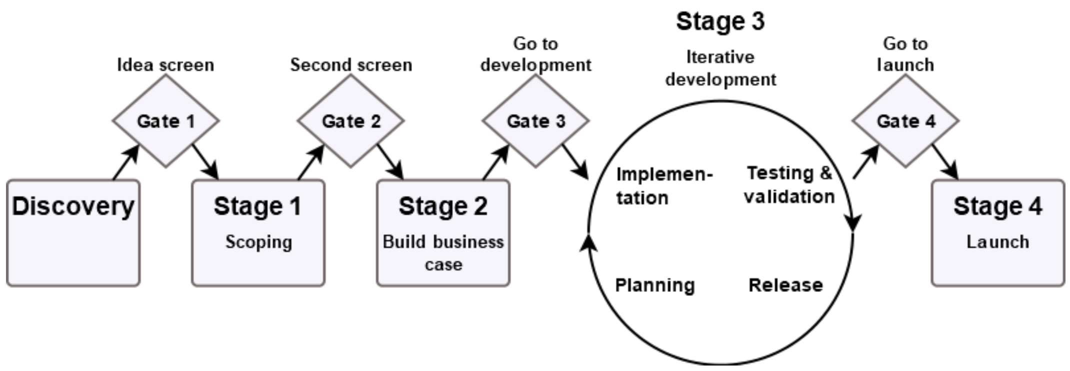

Figure 6. An example of the Agile-Stage-Gate hybrid project model.

2.7 Agile methods in hardware development

Although agile methods were originally created to be used in software development projects, they are making their way into hardware development as well. Today it is common to incorporate software into hardware products and in this case, utilizing an agile model also for the hardware part is especially beneficial. It enables the software team to start their development work earlier by using the first versions of the hardware instead of waiting for a finished product. In addition, the software team is able to give feedback to the hardware team and suggest changes to them if deemed necessary. For example, when developing embedded systems, it might become apparent that a faster processor or more memory is needed for the system to perform optimally.

However, implementing agile methods into hardware development introduces new challenges, which are not as big of a concern in pure software projects. Changing code is quite straightforward, compared to changing components in a physical product, making proper planning very critical in hardware design. This means that usually, change comes with a much higher cost when dealing with hardware and the cost will increase significantly towards the end of the project. Testing is typically less automated and more time-consuming in hardware projects, in comparison to software development, where the testing can be done very frequently, even after only small changes to the code. These key differences can make it seem like it is nearly impossible to adopt agile ways of working for hardware design, but studies made on the subject have shown that this is not true.

Gustavsson and Rönnlund (2013) determined that agile methods can work in hardware development. It allowed for faster results, compared to earlier projects completed with traditional methods and the reception for the new approach was positive overall in the case study organization. Punkka (2012) emphasizes the system-level adoption of agile methods and the same cadence for hardware and software development. He proposes that utilizing an agile model in hardware development benefits the whole company, especially if it is already used in their software department and they are working on the same final product. In a later study by Punkka (2016), it was concluded that organization-wide support for agility is required in order to fully benefit from it. In a paper by Cooper & Sommer (2016b), an Agile-Stage-Gate hybrid model was used in a physical product development project, where Scrum was employed in the development and implementation phases. The result was considerably increased productivity, when compared to earlier projects, mainly due to improved communication. A case study by Reynisdóttir (2013) concludes that it is feasible to make use of Scrum in mechanical product development, with some adaptations to the process. It was found that the communication between the project team and the stakeholders was improved. Furthermore, everyone included was able to get a clearer picture of the whole project and the amount of redundant work was reduced. A combined literature review and case study by Punkka (2016) shows that an agile project model can be advantageous also for projects, which do not involve software development, as it improves team collaboration and accelerates learning, among other benefits.

As mentioned earlier, adopting agile methods into hardware development is usually not quite as simple as using it in software design projects. There are multiple aspects to consider and some adjustments can be needed for the process, to make the iterative model better suited for creating physical products. For example, it might not be possible to produce a complete, working product increment after every Sprint when using Scrum. In this case, designing only a smaller part of the product or an improvement to it can be specified as the goal of the Sprint (Cooper & Sommer

2016b; Reynisdóttir 2013). However, it is possible to lose the benefits Scrum brings, if it is modified too much (Benefield 2008).

Several studies have pointed out that the ability to create prototypes quickly, efficiently and as early as possible in the project, is one of the most important aspects of agile hardware development (Cooper & Sommer 2016b; Punkka 2012; Punkka 2016; Stelzmann 2011). Previously prototypes have been commonly created at a later stage in the project to validate the already mostly finished design, but in agile development it has been found to be advantageous to change this practice and utilize up-front prototyping instead, which means using the prototyping process as a learning and experimenting tool and not only as a validating tool (Cooper & Sommer 2016b; Punkka 2012; Punkka 2016). Early and frequent creation of prototypes accelerates learning and the whole development process by making it easier to discover design errors, which otherwise might have been left unnoticed for a long time. Moreover, it allows the customers and other stakeholders to see the progress and give feedback for the product. The result does not have to be perfect for the prototype to fulfill its purpose and mistakes should not be feared excessively, because accepting failures as a part of the process creates a more encouraging environment for experimentation, driving the development forward (Cooper & Sommer 2016b; Punkka 2016; Zanotti et al. 2017).

The main disadvantage of the recurring prototyping are the extra expenses related to it. The focus of the prototyping process should be in minimizing the costs and time to demonstrate the desired functionality. Vertical slicing technique can be used to design different parts of the product in parallel, which together accomplish a larger, verifiable progress increment for the product. Simulation, 3D printing, and breadboards, among others, are good tools for this (Punkka 2012; Zanotti et al. 2017). Additionally, it is important to avoid redundant and unnecessary work by taking the time to study and test the previous prototypes carefully, before creating the next one (Punkka 2016). Even though the costs of prototyping are greater when compared to traditional project model, the iterative model makes the costs for the whole project more predictable. If the first prototype, created at the end of the development process, reveals a critical design fault, it can mean that significant amounts of time and money have to be allocated for fixing the issue (Punkka 2012).

The required time and resources allocated for testing in hardware projects is typically much greater in comparison to software development, mainly due to the lack of automation in the testing practices. As every prototype has to be tested, the testing should ideally be performed continuously beside the development and it can be a considerable challenge to adapt the verification and validation processes for the iterative development style. When using Scrum, longer Sprints might be needed for this to be possible, but they should not be extended too much either. Eloranta et al. (2013) found that Sprints lasting over four weeks result in lower work efficiency at the beginning of the Sprint, which causes the tasks to be unfinished at the end of the Sprint. Furthermore, the tasks tend to grow too large and some of the work done might become obsolete by the end of the Sprint. Moving the testing to the next Sprint can be a necessary compromise, even though it contradicts the principles of Scrum. Increasing the level of automation in testing would help the adoption of Scrum in hardware development, but the large initial investment can be hard to justify (Punkka 2016).

One of the original values in the Agile Manifesto is “Working software over comprehensive documentation” (Beck et al. 2001), but if the amount of documentation is reduced too much, it can be detrimental for the project (Cervone 2011). Definition of done must be clearly stated for every work item in the Sprint backlog and the creation of necessary documentation should also be included in the tasks or specified as separate work items. In some cases, the documents can even be automatically generated to reduce the workload (Punkka 2016; Karlstrom & Runeson 2005). The amount of documentation should be just enough to cover everything essential, without slowing down the development too much. Eloranta et al. (2013) observed that if a company only partly adopts Scrum and still requires a comprehensive requirements documentation instead of just using the product backlog, the team might find it difficult to fully understand the requirements and their order of priority.

Introduction of agile methods to the hardware development domain is still a quite recent phenomenon and it will probably take some time before it becomes widely accepted there. Because of this, the majority of the experts on the field might not yet be very familiar with the concept of Scrum and agile methods. Coaches for agile methods can alleviate this issue by offering guidance and support to the employees in issues related to the adoption of the Scrum process and how to incorporate it in their daily work. The company can organize lectures where an expert on agile methods teaches the stakeholders and professionals everything they need to know about the subject and the coach can then continue to help them during the projects. Studies from Carlson and Turner (2013) and Laanti (2016) point out the importance of the coaches.

2.8 Scaling agile methods

Scrum and agile methods, in general, were originally targeted to be used by one small team, working on a project on its own. While agile methods have continued to gain popularity, there has been a growing demand for solutions, which would allow them to be scaled for larger projects with multiple teams, possibly located in different countries. The definition of large scale is not distinctly defined, but typically it means two or more teams coordinating their efforts to create a relatively demanding product. (Dikert et al. 2016; Dingsøyr et al. 2014; Larman & Vodde 2013)

Multiple frameworks, using different approaches, have been created in order to make scaling agile methods for bigger projects possible. Alqudah and Razali (2016) determined that Disciplined Agile Delivery (DAD), Large Scale Scrum (LeSS) and Scaled Agile Framework (SAFe) were the most commonly used models designed for this purpose at the time of their study.

Disciplined Agile Delivery (DAD) is the most complex of the frameworks and the most challenging to implement for an organization, as it combines practices from several different existing models, including Scrum, Lean, Kanban, Extreme Programming, Agile Modeling, and others. It is a hybrid method as it has separate phases to cover the whole product delivery lifecycle instead of focusing only on the development part. The process starts from the initial scoping, financing and scheduling, then it continues to the construction phase, followed by the final release and deployment of the product. DAD consists of four models, called lifecycles, which are utilizing different sets of processes and methods. Each lifecycle accommodates to different circumstances and the organization selects the one that is the most appropriate for them at the time. DAD is suitable for large projects with over 200 people, but it can be used for smaller projects as well. (Alqudah & Razali 2016; Ambler 2013; Vaidya 2014)

Scaled Agile Framework (SAFe) describes four layers of organization: Portfolio, Large Solution, Program and Team. Depending on the size and needs of the organization, different configurations consisting of two to four layers can be employed. The Portfolio layer is for managing the project portfolio and doing business decisions concerning investments and strategy while making use of Lean methods. The Large Solution layer concentrates on delivering large and complex systems to the customers, unlike the Portfolio layer, which is better suited for managing multiple separate programs. The Program layer consists of agile teams, including Business Owner Team, Release Management Team, DevOps team and System team, all with different responsibility areas. In addition to the agile teams, Product Manager serves as a head Product Owner, Release Engineer acts as a head Scrum Master for the program and User Experience Designer focuses on the keeping the design consistent from the users perspective across the program. The teams at the Program layer use a concept called Agile Release Train (ART), containing four iteration cycles, which are two weeks long and lastly a HIP (Hardening, Innovation, and Planning) iteration, which is three weeks long. Potential releases are scheduled to happen quarterly. The Team layer utilizes standard Scrum teams, but they are not required to be cross-functional, one Scrum Master can be shared between a few teams and Extreme Programming (XP) principles are used in combination with Scrum. The teams are still required to be able to create and test product increments by themselves. The release frequency is once a quarter at the Team layer as well. One program should consist of 50–125 members divided into five to twelve agile teams. (Alqudah & Razali 2016; Scaled Agile, Inc. 2018; Vaidya 2014)

Large Scale Scrum (LeSS) expands the standard Scrum process for multiple teams working on the same product. There are two variations of the framework: LeSS for up to ten Scrum teams and LeSS Huge for more than ten teams and up to a few thousand people involved. LeSS Huge integrates multiple LeSS groups to work in parallel. In addition to normal Scrum roles, there is a common Product Owner for all teams and LeSS Huge also makes the addition of Area Product Owners for different requirement areas. All teams share the same Product Backlog and aim to deliver an increment for the same product after each Sprint. The standard Scrum meetings are included with some additions. Sprint Planning is divided into two parts. Part one is attended by two members of each team and the common Product Owner. Part two is team specific, but members from other teams may attend for information sharing purposes. Scrum of Scrums resembles a Daily Scrum meeting, but it is kept between the teams with a representative from each one and it is not required to be arranged every day. Light Product Backlog refinement meeting between the teams is added, with two members of each team participating. Joint Sprint Retrospective meeting includes the Scrum Masters and one member from each team. Moreover, LeSS Huge introduces Pre-Sprint Product Owner Team Meetings, Area-Level Meetings, Sprint Reviews at the product level and Sprint Retrospectives at the product level in addition to LeSS. (Alqudah & Razali 2016; Larman & Vodde 2013; Vaidya 2014)

Dikert et al. (2016) concluded in their systematic literature review of industrial large scale agile transformations that even though these ready-made frameworks exist, it is rare that any of them are a perfect fit for an organization’s needs and can be used as such without modifications. They also listed success factors for the large scale agile transformations, most important ones being “choosing and customizing the agile approach”, “mindset and alignment”, “management support” and “training and coaching”. Furthermore, many companies mentioned the lack of guidance from literature to be a major challenge. Increased communication has been clearly shown to be a major advantage in agile projects (Begel & Nagappan 2007; Paasivaara et al. 2008; Cooper & Sommer 2016b), but too many meetings can also slow down the progress, because the added overhead can hinder the actual development work (Begel et al. 2009). Paasivaara et al. (2012) studied the usefulness of Scrum of

Scrum meetings and found that they were not very beneficial at the case study organizations, as there were too many people with different areas of interest and the participants were confused about what they should share with the other teams. Feature-specific Scrum of Scrums with fewer teams involved worked better, but there were still issues with coordination at the project level.

3 THEORETICAL FRAMEWORK

An agile project model differs from the traditional product development project model in many ways, as shown in Table 1. Scrum was originally created for software development projects, but the literature shows that is has proven to be an effective project model for hardware development as well. Studies made about the adoption of agile methods in physical product development projects point out several advantages over traditional methods, including:

- increased productivity and faster results (Cooper and Sommer 2016b; Friis Sommer et al. 2015; Gustavsson & Rönnlund 2013);
- a clearer picture of the project for the whole team (Reynisdóttir 2013);
- improved communication (Cooper & Sommer 2016b; Reynisdóttir 2013);
- less redundant work (Reynisdóttir 2013);
- higher project success in several categories (Serrador & Pinto 2015);
- increased product quality (Karlstrom & Runeson 2005);
- improved team collaboration (Punkka 2016); and
- accelerated learning (Punkka 2016).

There are also many challenges mentioned in the literature. Agile hardware development benefits from the ability to create prototypes quickly, especially in the early stages of the project, as it reveals design errors and accelerates learning (Cooper & Sommer 2016b; Punkka 2012; Punkka 2016; Stelzmann 2011). The main downside of frequent prototyping are the costs, which can be reduced by utilizing simulation, 3D printing and other tools (Punkka 2012; Zanotti et al. 2017). Producing a complete product increment after every Sprint is often not possible when building physical products, however, the team can still achieve progress with partial improvements and designs (Cooper & Sommer 2016b; Reynisdóttir 2013). This requires that the definition of done is clearly defined for every task, so the progress can be tracked.

Hardware testing is predominantly performed manually and the setups can be complex, which makes the testing very time-consuming. For this reason, fitting the testing and development in the same Sprint might not be a possibility. Building automated test systems for hardware products is considerably more difficult and expensive than for software products, but investing in test automation would help in the adoption of agile methods for hardware development (Punkka 2016).

Agile development and Scrum are still often new concepts for hardware developers, which makes proper training very important. Everyone should understand the principles and why certain practices exist in the process. Studies suggest that even after agile methods are already in use, it would be beneficial to have designated coaches to help the team with issues related to the process (Carlson & Turner 2013; Laanti 2016).

Efficient communication is a key factor for project success (PMI 2013b) and improved communication is regarded as one of the most important advantages of Scrum when compared to traditional project model (Begel & Nagappan 2007; Cooper & Sommer 2016b; Paasivaara et al. 2008; Reynisdóttir 2013). When agile methods are used for large scale product development projects and programs, managing the information flow between teams and between different projects has to be planned properly. Multiple frameworks have been created to address this challenge, the most popular ones being Disciplined Agile Delivery (DAD), Scaled Agile Framework (SAFe) and Large Scale Scrum (LeSS) (Alqudah & Razali 2016), but usually they require customizations in order to achieve a suitable solution for the organization (Dikert et al. 2016).

	Traditional project model	**Agile project model**
Requirements	Detailed specification at the beginning of the project.	Rough initial specification, detailed only for the near future.
Planning	Long term, everything is planned at once.	Mainly short term, continuous throughout the project.
Development	Completed all at once, after everything has been planned.	Incremental and iterative, learning from each iteration.
Testing	Started after the development of the product is done.	Performed simultaneously with development.
Product delivery	One time delivery, after the product is completely finished.	Many versions delivered during the project.
Customer involvement	Low during the project, feedback after the project.	High during the project, providing constant feedback.
Communication style	Mainly formal, through meetings.	Mainly informal, through daily interaction.
Responding to changes	Slow, changes are handled in a separate process.	Quick, frequent changes are expected.
Documentation	Vast and detailed.	Only essential documentation created.
Lessons learned	At the end of the project.	Multiple times during the project, after every iteration.

Table 1. Comparison of traditional and agile project models.

Shields & Young (1989) introduced the “Seven Cs” model, which is a behavioral model for implementing cost management systems. The model focuses on human behavior and incentivizing the employees to commit to the goals of the organization to achieve continuous improvement. The system is consists of the following seven categories:

1. **Culture:** Companies should strive for a strong functional culture to be successful in the long term. It can be achieved by having motivated employees, who feel that increasing their work performance benefits them as well as the organization. Characteristics of a strong culture include clear goals, strong and innovative leaders, specific norms of behavior, and effective communication networks.

2. **Champion:** Almost all successful innovations come from strong individuals, i.e. champions. Top management should support these champions and give them the resources to develop their ideas.
3. **Change process:** The change process is implemented by the champion, using controls, compensation, and continuous education, which are described later in the list. Enough time must be allocated for completing the change process.
4. **Commitment:** The organization and the employees should be committed to continuous improvement. Controls and compensation can be used to improve commitment.
5. **Controls:** The implemented controls should help in achieving the goals of the company and in accelerating continuous improvement. Flatter organizational structure and teams structured around the products of the company are recommended.
6. **Compensation:** Compensation can be used to motivate the employees to focus on continuous improvement. Innovation must be encouraged by allowing risk-taking. Compensation should be based on long term performance of the individuals and the teams. Non-financial compensation should be used to create an atmosphere of positive reinforcement.
7. **Continuous education:** Employees are the most important assets of the company and improving their skills increases the value of the company. Separate training sessions should be augmented by employees constantly learning from each other. The employees should also be educated on the compensation system by explaining how they benefit from improving product quality.

The Seven Cs model can provide guidance when an agile project model is being implemented in an organization. The framework can be utilized in order to reduce the resistance to change and make the transition smoother.

4 METHODOLOGY

The research method for this thesis and different types of case study research are described in this chapter. The methods for collecting and analyzing the data are explained and lastly, the validity and reliability of this study are assessed.

4.1 Research method

Quantitative research aims to explain, answering the question "why?" and it is used to measure quantity or amount. Qualitative research aims to understand, answering the questions "how?" or "what?" and the results are not in quantitative form (Harling 2012; Kothari 2004, 3). This study is qualitative, as the information is collected from interviews and the research questions fit in the description as well. Furthermore, nothing is being measured or compared statistically, which are tools used for quantitative research.

4.2 Case study

Cousin (2005) describes a case study as follows: "Case study research aims to explore and depict a setting with a view to advancing understanding of it". A case study can be intrinsic or instrumental in nature. Intrinsic case study concentrates only on a particular case, trying to understand the unique phenomenon and it does not make generally applicable conclusions. Instrumental case study, on the other hand, aims to gain general understanding, using one or more cases (Bassey 2009, 29; Harling 2012). This thesis is an instrumental single case study.

4.3 Collecting and analyzing data

Interviews are a common way of collecting data in qualitative studies. Interviews can be structured, semi-structured or unstructured. Structured interviews use predetermined questions, which are asked in the same order from every interviewee. Semi-structured interviews allow the researcher to ask follow-up questions if the interviewee seems to have additional input to the subject. Unstructured interviews enable maximum flexibility, but the analyzing of the responses can be difficult. Semi-structured interviews fit well to case studies, as the interviewees can express themselves better when they are not as restricted to the researcher’s perspective. (Hancock & Algozzine 2006, 40; Kothari 2004, 97–98)

The case study consists of six face-to-face interviews with two Project Leaders, two Product Owners, and two Scrum Masters. The interviews are semi-structured and the focus on different subjects varies a bit naturally, depending on the person’s knowledge areas and their position in the organization (the main questions are listed in Appendix 2). The interviews are recorded and later transcribed from the recordings. The transcribed interviews are synthesized by grouping together similar statements and separating the findings into categories. Furthermore, documents provided by the case study organization are used to supplement the data. The documents include descriptions of the organization’s previous traditional processes and new hybrid agile processes. They also include the material from the training day for agile methods, which was conducted for the hardware development department prior to the interviews.

4.4 Validity and reliability

Golafshani (2003) says that “Although the term ‘Reliability’ is a concept used for testing or evaluating quantitative research, the idea is most often used in all kinds of research. Reliability and validity are conceptualized as trustworthiness, rigor and quality in qualitative paradigm”. On the other hand, Stenbacka (2001) suggests that the concept of reliability is irrelevant and misleading in qualitative research because there is no measurement method. She concludes that “A thorough description of the whole process, enabling conditional intersubjectivity, is what indicates good quality when using a qualitative method”.

For this thesis, six people, covering three different roles in the company have been selected for the interviews (listed in Appendix 1). Two interviewees for each role allow the researcher to get reliable information from each perspective. The interviews are conducted face-to-face in a private meeting room using the Finnish language, which is everyone’s mother tongue, so there is no language barrier, which could possibly cause misunderstandings or restrict the conversation. The interviews are recorded and later carefully listened and transcribed to make sure that every bit of information is included. I was an employee of the case study company at the time of the interviews but was working on a different project, which was not under the same program with the case study projects.

5 FINDINGS

The information, gathered from the interviews and the company documents, is structured under 12 sections. Sections 5.1–5.4 explain the nature of the projects, the team compositions, how the teams collaborate with other teams and how the projects interact with the customers. Sections 5.5–5.7 describe the planning tools, planning methods and the effort estimation system used by the teams. In sections 5.8–5.10, the meeting practices of the projects are introduced and the development and testing processes are clarified. Each section includes opinions and possible improvement ideas from the interviewees. In sections 5.11–5.12, the positive experiences from Scrum adoption, as well as challenges in implementing Scrum are evaluated. The interviewees are not distinctly singled out and exact direct quotes are not used for privacy reasons.

5.1 Description of the projects

The organization had started aligning its strategy towards agile development to decrease the time to market for the products. At the time of the study, the product development department was using a hybrid Stage-Gate project model, where Scrum was utilized for development and testing. The hardware development department was working on two separate hardware projects under the same program, which will be called “project 1” and “project 2”. Project 1 had two Scrum teams, with the other one having 13 members and the other one having six full-time members and one half-time member. Project 2 had one Scrum team with 19 members, of which five were part-time members and did work for other projects as well. The work for both projects incorporated electrical engineering, electronic engineering, and mechanical engineering. The planned lengths for the projects were around three years and the implementation of Scrum happened in the middle of the projects. At the time of the interviews, both projects had been running Scrum for approximately 12 weeks.

Their idea was to adopt Scrum as it is, with little to no modifications to the process at first, to see how it works and to avoid endangering its efficiency by making unnecessary changes, when the model was still new and unfamiliar to almost everyone involved. The only modification they agreed to make was to leave out the first Daily Scrum meeting on the next day after Scrum Planning meetings because everyone already knew what they should be working on.

The hardware department had been using Sprints already before the Scrum was fully adopted and the length of the Sprints were initially six weeks, but later the projects decided to switch to a three-week Sprint cycle to match the software department and all the other teams under the same program. The six-week Sprints were found to be too long, as the teams perceived that it was too laborious and time-consuming to plan the tasks so far ahead and more work could go wasted, if the need for unplanned changes would arise near the end of the Sprint, due to unexpected discoveries or new information obtained. Several interviewees pointed out that the three-week iteration cycle is appropriate for them, as it is important to often enough and regularly check that the team is working on the correct issues. This allows them to quickly adjust their focus according to prevailing circumstances and make changes if needed. In the beginning, there were concerns that a three-week cycle is not long enough for hardware development and the overhead from the Scrum meetings would take too much time away from the actual development duties, but after a couple of Sprints, no more opposing opinions were heard.

5.2 Team compositions and responsibilities

Most of the personnel working on the projects had no knowledge or earlier experience of Scrum. There had been a one-day training session for the employees about the Scrum process and how it will be implemented in the organization. One of the instructors was a Product Owner from one of the projects, who also had certification for the role and previous work experience in both software and hardware development using Scrum. More training had already been planned for the Product Owners and the Scrum Masters in the near future.

The teams in the hardware department were previously divided by competences, meaning all electronics designers formed one team, as did the mechanics designers and electrical engineers who were working with higher voltages. After Scrum, they were distributed into separate cross-functional Scrum teams, which had the ability to create product increments independently. Testing had been integrated into the Scrum by sharing the testing resources between the projects and the testers were part-time members of the Scrum teams for one to two Sprints, when their services were needed. The testers were not in more than one Scrum team at a time, because the overhead from the Scrum process and all the meetings would have taken too much of their time. They also did not change teams in the middle of a Sprint and the same rule applied to the part-time specialist members who were helping with technical issues. Some teams had recently taken a step forward and integrated testers into their Scrum teams as permanent team members.

Each project had a **Project Leader**, who managed the money, resources, and risks for the project, much like project managers in the traditional model. The difference to the old model was that they were not directing the everyday work of the team, as they were not a part of the Scrum process originally, although later they were given the role of administering the Sprint Planning 1 meeting. The Project Leaders were the head of a Core Team, which consisted of the Product Owners and project managers from the product administration and supply chain organizations. The members of the Core Team reported to the Project Leader, who reported to the program management and from there the information went to the steering group. Given the large size of the program, it was necessary to have several levels of administration to be able to manage it adequately. The program had a manager for the Product Owners, but the manager did not take part in running the Scrum.

The Product Owners were former hardware development project managers, who still had some project manager duties on top of their role in Scrum. One person was responsible for all of the verification and validation performed and he resembled an extra Product Owner for both projects. He maintained a document containing all the planned tests for the products and from that document, the test cases were transferred to the Product Backlogs of both projects.

Line managers were serving as **Scrum Masters**, which was not an optimal solution, but there were plans for giving the roles for team members in the future. This subject was discussed in every interview and everyone agreed that there were no issues at the moment and the arrangement was satisfactory. The team members did not fear to communicate the problems they faced in their work to the Scrum Master, even though they were talking to their line manager. Some commented that this was probably due to the low hierarchy in the company culture, where the superiors were not seen as intimidating authoritarian figures, but rather as colleagues. This was also considered as the easiest solution when Scrum was just being introduced, to avoid additional confusion, when the members of the teams already had so much new information and new concepts to absorb. It would be better, that everyone was familiar with their role as a Scrum team member first, before adding a Scrum Master role on top of that.

There were no subcontractors included in the Scrum teams, as not much of outside work was used, according to the interviewees from both projects. Parts for the prototypes were ordered from outside of the company and some analysis was done by third-party laboratories, but the actual development was performed fully inside the organization.

It was expressed that maybe there could be a possibility to divide the product into smaller parts to reduce the team sizes. Although it would be difficult because there were so many competence areas to cover in hardware development. Many commented that at the moment, it was not possible to make the teams smaller. It would likely force some members to work for more than one team and hinder communication. One person mentioned that it would be good if they could get to a point, where one person could design e.g. a circuit board and some other developer could make improvements for the design later. At that time, the convention was that whoever did the original design, would be responsible for all of the future modifications as well.

Most of the interviewees stated that the daily duties of the team members did not change considerably when Scrum was introduced, but one commented that the responsibility areas might have been expanded a little. The person continued that the developers were more frequently doing tasks, which they were less familiar with, but had the skills to complete them. This would likely have a positive effect on the team’s competence and flexibility. Previously the tasks of a particular type were usually always assigned to the same person and this had not allowed the team members to gain as much experience outside of their usual tasks. The Product owners did not see that much had changed in their work, compared to the previous project manager duties, but Scrum Masters, on the other hand, were a totally new concept, which the line managers had to learn.

The majority did not think that more employees were needed to be hired as a direct consequence of the transition to Scrum, as every role could be filled by the existing personnel. However, one person envisioned that it could indirectly lead to a demand for more people to improve the test automation, as it would be the key area to focus on, in order to achieve more frequent product increments. One interviewee also brought up that hiring a few more employees could be beneficial because some people, like the verification responsible, were working for both projects and it would be better if they were not required to divide their efforts between the projects. Another interviewee said that hiring more people to the teams could lead to a situation, where everyone did not have enough tasks to do.

Two interviewees brought up that there should be monetary compensation for the Scrum Master role, to encourage the team members to take the duties away from the line manager, because there are some amount of extra work and responsibilities attached to the role.

5.3 Collaboration with other teams

The project teams collaborated with other project teams, some of which were located in different countries. This was necessary because they were using the same platform design, even though the products were different, so some components were shared between the products. All projects used the same design tools to be able to transfer data between the teams effectively. Furthermore, the hardware projects had a lot of collaboration with software projects, due to numerous dependencies between them. Software teams needed hardware to develop software and hardware teams needed software to be able to run and test their products. One interviewee described that there were also situations, in which one team needed the help of an expert in another team to resolve issues or to act as a substituent when someone got sick or was on a holiday at a critical moment. Moreover, the projects distributed some documentation tasks from both projects evenly for all teams, which included, for example, the planning documents and specifications shared between the projects.

5.4 Feedback from the customer

As was noted in one interview, Scrum emphasizes frequent communication with the customer to be able to quickly react to the feedback and make the required changes to the product. However, in these projects, there was not just one specific end customer, as the products developed were general purpose products aimed for broad markets. The feedback from the customers to the team came through the Product Owners who had connections to the product management, which gathered feedback from the sales department and from the end customers. The products would be presented for selected customers a few times during the development and some of them would become pilot customers to use the product and give feedback for it when the product was functional enough, this would take place at about halfway through the projects. The customers would have to sign NDAs to not leak the information to competitors and due to this nature, no end customers were involved in the Sprint demo sessions.

One interviewee mentioned that maybe Scrum would be even more suitable for projects where the product was developed for one customer as a commissioned work. Then it would be possible to frequently engage with the customer and include them in the demo sessions to get constant feedback. Now it was not possible to fulfill every desire of every customer, but rather strive to understand the general consensus amongst them, steer the development in that direction and make the necessary compromises to achieve the best results. The company does customer specific projects as well and in those, the full potential of Scrum could be taken advantage of.

5.5 Planning tools

Microsoft Excel was used for managing the Product Backlogs and Microsoft Planner was used for managing the Sprint Backlogs, although the software development department of the company used a different specialized backlog management tool for handling both. The interviewees explained that Excel was selected for its simplicity, so the tool would not start to dictate the process. Other tools made for the purpose would have had a learning curve for everyone and further complicated the adoption of the new ways of working. The new tools might also have had limitations, which could have forced modifications to the processes to bypass them. After enough experience would have been gained from Scrum, they could consider switching to a more specialized tool, as then they would know which features were essential and which were not needed at all. With the gained knowledge, it would then be possible to select an appropriate backlog management tool for them. Microsoft Planner was chosen as a Sprint Backlog management tool for the projects as a more advanced version of sticky notes on a board because the sticky notes approach would not work very well for teams with members from different cities or countries, who would attend the Scrum meetings using a video conferencing application. Furthermore, these tools did not add any extra costs, as both Excel and Planner were included in the Microsoft Office suite that the company had already paid for. The interviewees were satisfied with the tools and did not see a reason to search for replacements very soon.

5.6 Planning methods

The organization’s software development teams utilized user stories, epics, and themes in their planning work, but the hardware projects took a different approach and simplified this convention to only having a backlog item in the Product Backlog for every requirement. The reason for this was to reduce the complexity and new concepts required to be familiarized when adopting Scrum. The Product Owner would manage the Product Backlog and order the items by priority. The goal was to make backlog items small enough to be possible to complete during a single Sprint. These items were then be split into tasks for the Sprint Backlogs, which would typically take one or two days to finish. Sometimes longer tasks of e.g. five days were also be added, instead of using too much effort in splitting them, if it was not practical to do that. It was emphasized in the interviews that defining a clear definition of done for the tasks was very important. The test cases from the verification plan document, maintained by the verification responsible, were also added to the Product Backlogs to be able to see the total amount of work left to be done and to get a realistic picture of the progress.

Even though the Large Scale Scrum process suggests that all the teams under the same project should use only one Product Backlog shared between them, all of the interviewees who commented on the subject thought that it was better to have separate backlogs for each Scrum team, because the shared backlog would be more difficult to manage.

One interviewee said that setting the Sprint goal at the start of the Sprint had proven to be an important part of the process, as then everyone could get a better view of the big picture and had a common objective, even though they were concentrating on their own tasks. The Sprint goal had been brought up in the meetings and it had likely increased the motivation of the team. Another person suggested that the Sprint goal should be made even clearer so that everyone would certainly understand, how their efforts brought the team closer to the shared goal.

The amount of documentation was thought to be adequate for the projects and it was said that it would not be wise to try to reduce the number. There has to be sufficient documentation available because after the development is finished and the responsibility for the product is transferred to the maintenance department, they have to be able to get the information they need.

5.7 Work effort estimations

Project 1 used a relative estimation method based on the Fibonacci sequence to estimate the amount of time needed to complete the tasks. Each team could decide what amount of time each point represents, but in practice, every team used one point for one day of work. It was pointed out that it would be hard to accurately estimate the needed time allocation for possible upcoming prototype creation work or other larger backlog items, so they would give those tasks a generous estimate to be on the safe side and to avoid issues with the schedule.

The general consensus among the interviewees was that the estimations had been fairly accurate even for the larger backlog items, at least when given by the more experienced team members. No interviewees saw that making the estimations was too hard for the teams or that it was a problem in any way. It was also mentioned that sometimes there has been unexpected issues and the completion of an item has been delayed considerably, especially if problems were found in the testing or if the team had to wait for some approval that was required before continuing, but this was not a frequent issue.

A burn-up chart was used to estimate the progress of the project and the sufficiency of the resources. From the chart, it could be determined if the project was on schedule and if more resources were needed to not fall behind. Project 2 was not as far in adopting Scrum as project 1 at the time of the interviews, as they did not use days or points for estimating the required work for items in the Product Backlog at all. During the Sprint planning, the team members would just choose the tasks they believed they could be able to finish.

5.8 Meeting practices

The meeting conventions in both projects were adapted from the Large Scale Scrum (LeSS) framework, but the teams were not rigorously following the practices defined in the framework. Some meetings were left out and some were added from outside of Large Scale Scrum. For example Scrum of Scrums meetings were excluded and Core Team meetings, Voice of Customer workshop and PI-Planning meetings from the SAFe model were included.

The Sprints started with **Sprint Planning 1** meeting, which was originally organized for each team individually, but later it was changed to a cross-project meeting with Project Owners and representatives from each Scrum team and a representative from supply chain operations attending. The Project Leader was administering the one-hour meeting. The agenda was to examine the general, high-level schedule strategy, which was a Gant chart in Excel and to prepare backlog items for the Sprint Planning 2 meeting, which would be conducted right after the first planning. The attendees estimated, which items they could be able to finish during the Sprint, according to the mutually agreed definition of done for each item.

Sprint Planning 2 was a longer meeting, lasting up to three hours. It was held separately for each Scrum team and the Scrum Master was administering it, but also the Product Owner was attending. The team went through the previously selected backlog items and confirmed which ones would be chosen for the Sprint and included into the Sprint goal. The items were split into smaller tasks for the Sprint Backlog. After this, the Product Owner was fully aware of what the team would be doing for the next three weeks.

There was a suggestion that the Sprint Planning 2 meeting could be improved by making the team members split the backlog items into tasks in small groups, instead of doing it all together because it would be faster and the role of the Product Owner was too prevalent in the process they were using. The product owner could even be excluded from the meeting completely in the future. Moreover, it was said by two people that three backlog refinements might be needed in the first few Sprints of the project, but after that, they could be reduced to one or two sessions.

Daily Scrum meetings administered by the Scrum Master were organized every day, excluding the day after the Sprint planning meetings. This meeting followed the Scrum process without any modifications, meaning the team members communicated what they had done, what they were going to do and did they face any issues in their work. The Product Owner also sometimes attended the meeting to keep up with the progress and gain information. After the Daily Scrum, they had an optional extension of 30 minutes for solving problems if there were any. The time for the problem-solving session was also reserved in everyone’s calendars for every day, so there was always time for that if it was needed.

Once a week they had scheduled a **cross-project Backlog Refinement Planning** meeting, attended by the Project Leaders, the Product Owners and representatives of each team from both projects. In this meeting, the teams prepared the contents for the next team-specific **Backlog Refinement** meetings, which would be held later in the week. The Product Owners suggested which backlog items they should refine in the next session. The attendees also identified the possible needs for inviting specialists from outside the teams into the following refinement to help with more intricate technical details and, additionally, they planned how to handle possible overlap between the projects, regarding the specialist attendance.

The **team-specific Backlog Refinement Planning** session was scheduled for every week, in which the Product Owner and the team went through the Product Backlog items selected earlier for refinement. The content and time estimates for the items were defined more accurately and their priority orders were reviewed for upcoming Sprints. The refinement could be done in small groups, where each group would take a subset of backlog items under inspection.

There were three meetings planned for the last day of the Sprint. The first one was the **Sprint review** for each team separately, where they analyzed the Sprint results, describing what tasks were completed and what tasks were not completed. The team members demonstrated the results of their work to the Product Owner, who then officially accepted the results for the Sprint and examined the state of the Product Backlog with the team. They also already started to plan what could be done in the next Sprint.

The second meeting of the day was **Sprint Retrospective**, again organized for each team separately. The goal was to review the previous Sprint from a viewpoint of processes and tools. The team considered methods to improve their ways of working to enable them to complete their tasks easier and faster. They identified what went well and what needed to be fixed for the next Sprint and created a plan for that.

The third and last meeting on the same day was **Overall Retrospective**, attended by Project Leaders, Product Owners and team representatives from both projects. This was similar to the team-specific retrospective, but the analyzing of processes and tools was done using a much bigger scale, covering problems regarding both of the projects and the entire organization. A plan was created for the next Sprint, defining how to deal with the identified issues and describing what was working well and should not be changed.

Every week there was a **Core Team meeting** to share information about what had been achieved, what kind of problems were the teams facing and also analyzed the risks. They went through requirements from different sources for the project and planned on how to distribute them between the Product Backlogs of each team.

At the beginning of the project, there was a **Voice of Customer** (VoC) workshop, which was an event lasting two days, attended by representatives from sales, product administration, product development, and production. The needs and wants of the customers were analyzed. This included deciding which features were the most important to have at the initial launch of the product, to guarantee that the product would be competitive in the market and which features were less important, but could be done if there was time to finish them or could be moved on to the next version of the product. Based on this analysis, the project team knew what to focus on.

PI-Planning (Product Increment Planning) meeting, a part of SAFe (Scaled Agile Framework), was scheduled to be organized once per quarter, i.e. every three months. The meeting had originally over 60 attendees from every project under the same program, including hardware and software development, but was planned to be reduced to about 40 in the future. The aim was to inspect the dependencies among all of the projects and make plans on how to manage them for the next six months forward. The dependencies were examined by writing down all relevant requirements for each project on pieces of paper and laying them across tables. Then the requirements were connected by strings, representing the dependencies between them. Different colors for the strings were used to indicate the state of the dependency, e.g. green meant that everything was fine and progressing as scheduled and red indicated that there were delays or issues, affecting the schedule of the other requirement connected to it. When doing the analysis, it could also be noticed if requirements are missing completely and needed to be added. Furthermore, the agenda of the meeting included presentations from each project for the other attendees to listen, where a representative from the project described what they had accomplished so far and what kind of problems they had been facing in the past three months. In addition, the hardware teams had been arranging mini PI-Planning sessions with the software teams every six weeks to make sure that everyone was focusing their efforts in the correct subjects.

One person stated that the PI-Planning meetings had been the most important tool in managing the dependencies and co-operation between different projects. The identifying of dependencies using the papers and strings method had been working well and the participants had been happy with it.

None of the interviewees were interested in integrating Scrum of Scrums meetings from the Large Scale Scrum framework into their process. Informal discussions were happening during each day, in addition to the scheduled meetings and adding an extra meeting was not seen as a necessity. As stated by an interviewee, this was also in line with the Agile Manifesto principle of "individuals and interactions over processes and tools". Cross-project Sprint Planning 1 and Overall Retrospective meetings were sufficient for communication between the projects. The information flow between the hardware teams was working well and no improvements for that were needed, but the communication with software teams could have been improved to some extent. The Scrum meetings, in general, were not considered as a waste of time, as they reserve only about ten hours per Sprint for each team member and many problems are being solved during the meetings.

5.9 Development

It was said that in hardware development it is not as easy to share the tasks between different team members as it is in software development, because the expertise areas are so specific. In practice, each backlog item is tagged to one person who is the specialist of that competence area and has the capability to complete the tasks to finish that item. The development of the product needs specialists from many different fields, which might lead to larger team sizes compared to what the Scrum model suggests so that the teams maintain the proficiency to create product increments independently.

A common consensus among the interviewees was that it was usually not possible to create a complete, functional product increment in every Sprint, because of the long lead up times of some prototypes and the fact that software development was done under a different project, which sometimes had different priorities than the hardware projects. The tasks in the Sprint Backlog could include, for example, gathering information and creating a document or creating plans for future designs. One person added that this is more common in hardware development, in comparison to software design projects, which can create new versions of the product typically after each Sprint, although there are exceptions as well. One hardware team, working on electronics, could create a prototype in about three Sprints and for other teams, it could take as long as six to twelve months. At the beginning of the project, the time required to create product increments was usually longer than later on when the most pressing issues had already been solved. The testing also took a considerable amount of time as most of the tests were not automated.

The first prototype was typically a very crude one, a minimum viable product, which could be used by the software department to start developing software for it. The goal was to finish it as early as possible because it was necessary to have software for the product to be able to test it properly. High power components were emulated with electronics, so that the software developers could run the devices in their offices safely, without the need to use high voltages. Some mechanical parts of the product could be created relatively quickly using 3D Printing, and simulation was utilized for temperature and some other areas, but simulation still could not replace real-world testing. The simulation, 3D printing, and breadboards had already been used long before Scrum adoption. Some teams were able to do the simulations by themselves and other teams used specialists from outside of the Scrum teams for that purpose. The product or parts of the product could be assembled onto a table without housing, using breadboards and connecting them with wires. All the plastic parts could be 3D printed and also sheet metal had been printed before. Moreover, the production department had its own 3D printers, so they could experiment with different designs and give feedback and suggestions to the mechanics designers in the product development department.

It was suggested that more research was needed in utilizing simulation especially in the earlier stages of the project. It could also be beneficial to study how to develop the end product more as a whole, how to integrate software and hardware together and what kind of agile process models could be used for this. At the time, software and hardware development were performed in separate environments and a new official software version for the hardware department was scheduled to be released only every three months because the software development was lagging behind the hardware projects quite a lot. It would have been better to get a new version of the software after every Sprint, which could have then been used in the tests.

5.10 Testing

An interviewee explained that the planning for future tests can be started right at the start of the project, so there was not too much downtime for the testers who were permanent members of the Scrum teams. Testing was performed constantly, but the need for testing was not steady all the time and sometimes there were spikes in the demand. It was stated that in project 1, the testing of each complete prototype, which included the work from all Scrum teams, lasted about six to eight months. During that time, modifications and software updates for the product were likely needed, which meant that some of the tests had to be performed again. The designing of a new prototype was started already when the previous one was still under testing, so the designers had work to do and they did not have to wait for the testing to be completed. It was said that the developers were also writing design documents and had other responsibilities, like quality control, which were not purely development duties.

It was mentioned that most of the testing could not be automated at that moment, because assembling the test setup already took at least two days for the whole product. The electronics team, on the other hand, could complete functional tests for the circuit boards in only a week, but reliability testing took a longer time for them as well. Final testing for the product, including approvals, was estimated to last one year for project 1 and one and half years for project 2, if everything went well and no modifications were needed. There were plans for more advanced test automation for integrating software and hardware, which could be used to simulate more complex setups.

Until then, the testers had been mainly built the test setups by themselves, using guides made by the developers, but often something had been forgotten or left unclear and this had led to delays in building the setup. An interviewee said that there were plans to get the developers to assist with the test setup. This would accelerate the testing process because the products under testing are usually new and unfamiliar to the tester, but the developers are very well aware of the technical details concerning their portion of the product and could quickly clarify the issues, which are unclear for the tester. This would require multiple developers participating in the process, as each of them have insight for their own area, but usually, none of them are experts on the whole product.

One interviewee brought up that it could be advantageous to include the testers as permanent members in all Scrum teams. There were still doubts because maybe it would not be worthwhile for the testers to participate in every meeting, as all of the subjects might not concern them at all and the time could be used better by performing the tests. At least in some teams, the test tasks were small enough to fit into one Sprint, which is why including the testers to the team was seen as a possibility. In addition, it was mentioned that a representative from supply chain could perhaps be added to the team as well in the future.

5.11 Positive experiences from Scrum adoption

Several interviewees emphasized that they had seen very positive results from Scrum, as the next three weeks were always properly planned and everyone knew what to do and what was expected from them. Before with the traditional model, it had been difficult to estimate how long it would take to complete the larger design tasks. Now the team was better aware of the progress of the product and planning the schedule took much less effort, as all the training sessions and other events unrelated to the project were also taken into account more accurately. The team members had been more committed to their objectives and they had been able to work without distractions because every task they did, came through the backlog and no extra tasks were given to them. Moreover, the frequent feedback from the results of their work had been rewarding for the team members and had been keeping up the motivation. The interviewees could not say at that point if Scrum had made the development faster, but some envisioned that the increased motivation and clarity in the objectives could potentially speed up the work and improve the time to market for the product. None of them wanted to go back to the traditional model.

The Daily Scrums had been perceived as very productive use of time and the performance of the team had been improving for the reason that now the problems could be addressed and resolved immediately, instead of some developers using several days trying to solve the issues by themselves, like had been the case sometimes in the traditional model. Before Scrum was taken into use, the hardware teams had already been using short morning meetings as a temporary solution, when they had been in a hurry to solve some pressing issue and this method had been very successful in the past.

Cross-functional feature teams were seen as a very positive change and it had a big impact on the development performance, as the team members were sitting close to each other and were able to communicate constantly and exchange ideas. The improvement had been noticed especially in the collaboration between mechanics and electronics designers, as the circuit boards had to be integrated into the mechanics. Previously the mechanics designers had been doing work for many projects at the same time and could not have been focusing fully on one product. One downside of this arrangement was that the experts in the same field did not share information between them as much as before when they were situated close to each other. However, this issue was alleviated by the cross-project meetings and other meetings not belonging to the Scrum process, where the experts had a chance to communicate.

Several interviewees mentioned that problems were noticed much earlier now when compared to the traditional waterfall development model and this is apparent every day. The information flow was much better and deficiencies came up earlier. An example was given that in the Daily Scrum someone might remind about a feature, which has not yet been tested and the testing needs to happen before the next design is started, so the possible faults could be noticed and not transferred forward into the next design iteration. No matter how experienced and skillful a person is, they might forget something or not notice an issue. The probability of catching errors is high when things are brought up and discussed every day in the Daily Scrums.

There was a mention that the implementation of Scrum process had been very successful, as after only a few Sprints had passed, there were not many issues brought up anymore in the retrospective meetings related to the process itself. From that point forward, they could concentrate on reviewing the tools and work instructions during the retrospectives.

5.12 Challenges in implementing Scrum

It was brought up that it had been challenging to make the team members internalize the concept that they had the power to decide what tasks they would choose for the next Sprint because they had been used to someone telling them what to do. This would supposedly change when they gained more experience from Scrum and familiarized the new ways of working.

Two interviewees mentioned that there had been initially some resistance against the Daily Scrum meetings among the hardware developers because they had not been used to having to report about their work every day. However, the complaints had been diminished over time and were already nonexistent.

One line manager and Scrum Master commented that it was very important to be able to separate the two roles and forget the role as a manager when conducting the Daily Scrum and other meetings included in the Scrum process. The team members should learn that in the Daily Scrums they were reporting to the whole team and not only to the Scrum Master. He estimated that the total time consumed by the Scrum

Master role amounted in 30 minutes of work each day when distributed equally, which was very reasonable. The one downside of using line managers as Scrum Masters, according to him was, that the line manager was automatically better informed about the matters of the employees who were in the same Scrum team with him. Although this issue was easily fixed by being proactive and making sure to also talk to employees from other Scrum teams frequently enough and ask how they are doing.

One challenge mentioned was that there were differences in adopting the Scrum process between the projects. Project 1 was following the process very well, but project 2 was somewhat lagging in the adoption. According to the interviews, one probable reason was that project 1 had one Product Owner with strong Scrum knowledge and experience and project 2 had no people with previous Scrum experience. The people from project 2 likely did not fully understand their role in the team and did not see the purpose of all the meetings. The other reason could have been that the ways of working were somewhat different in project 2 and maybe it was more difficult to implement Scrum into that. One person added to the subject that a more extensive training for everyone would have been valuable before Scrum was introduced to the development, as then the adaptation would have likely been quicker.

An interviewee contemplated that the basic Scrum process was quite easily adopted in the hardware department, but it would require many years of evolving the system to achieve a truly agile project environment. This would involve large investments in, for example, test automation. Furthermore, the whole company culture would need to change and they would have to get rid of excessive bureaucracy, which is difficult for large organizations in general. He added that quality control is very important in hardware development, which creates challenges for the implementation of agile methods. Specific processes in the development are required, to be able to obtain certain certificates for the products and modifying the processes to fit the requirements could hinder the agility of the development. After a certification has been rewarded, some parts of the processes cannot be changed anymore in order to keep the certificate.

6 DISCUSSION

The findings from the case study are analyzed in this chapter. Both research questions are addressed and potential solutions to them are presented, based on the literature review and the case study findings. The literature review was presented in chapter 2 and the case study findings in chapter 5.

6.1 First research question

What aspects should be considered when transitioning from a traditional project model to an agile project model in hardware development projects?

Testing is usually very laborious and time-consuming in hardware development, which creates a bottleneck when attempting to create complete product increments frequently. It was brought up in one of the interviews that serious investments in test automation are needed for making it possible to produce a new version of the product after every Sprint or even after a few Sprints. The same statement was also made in research by T. Punkka (2016). This does not mean that agile methods should not be implemented at all if most of the test are not automated. Transitioning to Scrum had been perceived as a very beneficial and positive change by most of the interviewees, even when almost all of the testing was done manually. The teams were still able to achieve progress in every Sprint. Similar results were reported in a study by Þ. Reynisdóttir (2013).

The importance of training (Dikert et al. 2016) and coaches for agile methods (Turner 2013; Laanti 2016) were pointed out in the literature. The interviewees mentioned that there should have been more training before the Scrum process was introduced. Furthermore, the project with better success in utilizing Scrum had an experienced and certified Product Owner in one team. The other project had no one with previous Scrum experience, which likely had a negative effect on how the process was being adopted. Continuous education is included in the Seven Cs model, which states that employees are the most valuable asset of the company and improving their skills increases the value of the company (Shields & Young 1989).

The literature suggests that a Scrum team should have no more than nine members, as larger teams require too much coordination (Schwaber & Sutherland 2017; Deemer et al. 2012). However, the team sizes can grow quite large in hardware development projects and limiting the number of members can be difficult, when experts from several fields are needed to create the product. In the interviews, it was mentioned by most that the current team sizes could not be reduced, as it could hinder the development. The only solution would be to divide the product into smaller parts, which could be one thing to consider before starting a new project.

The hardware department in the case study company wanted to implement Scrum with minimal modifications to the process and see how well it suited them, before considering possible changes based on the experiences. This approach was also supported by the literature, as Benefield (2008) concluded that it is possible to lose the benefits of Scrum by altering it too much.

The teams were using six-week Sprints prior to full adoption of the Scrum process and found them to be too long, as the planning was far too time-consuming and there was a greater possibility of wasted work, due to possible changes in the middle of the Sprint. They moved to three-week Sprints to allow for greater flexibility and to match the Sprint cycle of the software department. The change was received positively, after initial concerns by the developers. The research by Eloranta et al. (2013) came to the same conclusion, finding that Sprints over four weeks tend to lower the work efficiency at the beginning of the Sprint, tasks often grow too large and it is more probable that some of the work done becomes obsolete by the end of the Sprint. Moreover, Punkka (2012) proposed that having hardware and software development in the same cadence greatly benefits the company.

It was mentioned in one interview that closer integration of hardware and software development could be advantageous for the future. Hardware and software departments had their own separate implementations of an agile model and different priorities, which slowed down the development of the product. This was also mentioned in the study by Punkka (2012), where he emphasized the benefits of co-design between hardware and software developers.

Two interviewees suggested that there should be monetary compensation for the Scrum Master role, as this was not the policy at the time. The team members should be incentivized to accept the additional responsibilities that come with the role. The Seven Cs model addresses this issue, as one of the Cs included in the model is compensation, which is used in a way that it motivates the employees to focus on continuous improvement (Shields & Young 1989). Additionally, reducing bureaucracy in the organization to improve the agility was mentioned in the interviews. Punkka (2016) concluded that organization-wide support for agility would be needed to be able to fully benefit from it. The Seven Cs model emphasizes the importance of top management support for change processes (Shields & Young 1989). The case study organization had recently started moving towards agile product development, but was in the early stages of the transformation, so the aforementioned problems still existed.

6.2 Second research question

How to manage communication between different projects and Scrum teams in a large scale product development program?

Dikert et al. (2016) reviewed different scaling models for agile methods and came to the conclusion that existing frameworks can rarely be taken into use by organizations without modifications. In the case study company, the meeting practices were adopted mainly from Large Scale Agile (LeSS) framework, but for example Scrum of Scrum meetings were excluded from the model. The interviewees thought that Scrum of Scrums were not needed, because the communication between the teams was already sufficient without them. The sentiment was supported by the literature, as the Scrum of Scrums meetings were determined to be not beneficial (Paasivaara et al. 2012). Moreover, Begel et al. (2009) found that too many meetings can add excessive overhead to the process and slow down the development.

Strong communication networks are one characteristic of a strong organizational culture according to Shields & Young (1989). According to the interviewees, the modified LeSS framework with the additions of Core Team meetings, PI-Planning meetings from the SAFe framework, and Voice of Customer workshops was a satisfactory solution for a large scale implementation of agile methods, when multiple teams from different departments were involved. The PI-Planning was considered as the most important tool for managing dependencies between different projects.

However, there was a mention that the communication between the hardware and software projects could still be further improved. The hardware teams were sharing knowledge with each other informally every day, but there was not as much informal communication with the software teams. There were a lot of dependencies between the departments and although mini PI-Planning meetings were also arranged every six weeks between the hardware and the software projects, it could have been beneficial if the teams in the hardware department were updated more often on the state of the software development.

7 CONCLUSION

In the first three chapters, the results of the case study are summarized, followed by specifying the theoretical and practical contributions of this thesis. The next chapter analyzes the limitations of the study and lastly, subjects for future research are suggested.

7.1 Summary

An agile project model can be utilized effectively in hardware development when it is implemented properly. This was the conclusion from the literature review and the argument was supported by the case study as well. One of the most important areas to focus on is adequate training for everyone involved before agile methods are introduced into their daily work and there should be continuous coaching during the adoption phase as well. Test automation for hardware should be developed to allow the creation of product increments more frequently. Getting the organization to fully support the adoption of agile methods is also an important aspect, including monetary incentives to the employees for accepting additional responsibilities. Furthermore, integrating hardware and software development more closely with common cadence and priorities could be very beneficial. Other observations and improvement ideas were also gained from the interviews.

Sufficient communication in large scale agile programs, including multiple projects and teams, can be accomplished by combining different methods from existing frameworks and models, which are the most appropriate for the company in question. Studying and experimentation are needed for finding the right solution because it is unlikely that any of the existing scaling frameworks would be a perfect fit for an organization as such. In the case study company, a modified LeSS framework was combined with SAFe and other smaller customizations were added in order to achieve a suitable system for them.

7.2 Theoretical contribution

The theoretical contribution of this thesis is to provide more data on the subjects of adopting agile methods to physical product development and scaling agile methods to larger projects and programs. There were not too many case studies available on either of the subjects at the time of this research. The case study supports the existing literature, as the results are mostly similar to the previous studies, while hopefully adding some useful information. This research reinforces the importance of training (Dikert et al. 2016; Turner 2013; Laanti 2016), organizational support (Punkka 2016; Shields & Young 1989) and investments in test automation (Punkka 2016), which were addressed in the literature. It also raises the issue of Scrum team sizes in hardware development, which can grow quite large when experts of several different fields have to be included in order to enable the team to create complete product increments. Furthermore, the study provides an example of a functioning large scale agile communication solution.

7.3 Practical contribution

This research was made for the case study company and the interviews provided experiences and ideas, which the company can use to further improve their processes. For example, adding more automation to hardware testing, closer integration of hardware and software development, and incentivizing the employees to accept the Scrum Master responsibilities by monetary compensation could be considered. Other organizations planning on similar transformations can use the information from this thesis to be better prepared for the process and to avoid early mistakes.

7.4 Limitations of the research

The interviewees for the case study represented only two projects of one company, so vast generalizations cannot be made based on this research alone. Furthermore, the Scrum process had just been fully implemented quite recently and the interviewees still had limited experience in using it, so longer-term benefits and challenges were not covered by this study.

7.5 Suggestions for future research

More case studies are needed on the scaling solutions for implementations of agile methods, especially concerning large projects and programs, which require the collaboration of different departments and integration of software and physical products.

Another important subject of study is integrating automatic testing to agile hardware development and how to make it fast and efficient enough to achieve a truly agile development environment.

REFERENCES

Alqudah, M. & Razali, R. 2016. A Review of Scaling Agile Methods in Large Software Development. International Journal on Advanced Science, Engineering and Information Technology 6 (6), 828–837. 10.18517/ijaseit.6.6.1374.

Ambler, S. W. 2013. Going Beyond Scrum. Disciplined Agile Consortium. [online] Available at: https://disciplinedagileconsortium.org/Resources/Documents/BeyondScrum.pdf [accessed 12.9.2018]

Bassey, M. 2009. Case study research in educational settings. Buckingham: Open University Press.

Beck, K. 1999. Embracing change with extreme programming. Computer 32 (10), 70–77.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland, J. & Thomas, D. 2001. Manifesto for Agile software development. Agile Alliance. [online] Available at: http://agilemanifesto.org/ [accessed 9.8.2018].

Beedle, M., Devos, M., Schwaber, K., Sharon, Y. & Sutherland, J. 1999. Scrum: A Pattern Language for Hyperproductive Software Development. Pattern Languages of Program Design 4, 637–651.

Begel, A. & Nagappan, N. 2007. Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study. Proceedings - 1st International Symposium on Empirical Software Engineering and Measurement, ESEM 2007, 255–264. 10.1109/ESEM.2007.12.

Begel, A., Nagappan, N., Poile, C. & Layman, L. 2009. Coordination in large-scale software teams. 2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering, Vancouver, BC, 1–7. 10.1109/CHASE.2009.5071401.

Benefield, G. 2008. Rolling Out Agile in a Large Enterprise. Proceedings of the Proceedings of the 41st Annual Hawaii International Conference on System Sciences, 461. 10.1109/HICSS.2008.382.

Carlson, R. & Turner, R. 2013. Review of Agile Case Studies for Applicability to Aircraft Systems Integration. Procedia Computer Science 16, 469–474. 10.1016/j.procs.2013.01.049.

Cervone, H.F. 2011. Understanding agile project management methods using Scrum. OCLC Systems & Services: International digital library perspectives 27 (1), 18–22.

Cooke-Davies, T. 2002. The ‘‘real’’ success factors on projects. International Journal of Project Management 20 (3), 185–190.

Cooper, R. G. 1990. Stage-Gate Systems: A New Tool for Managing New Products. Business Horizons 33 (3), 44–54.

Cooper, R. G. 2008. The Stage-Gate Idea-to-Launch Process–Update, What’s New and NexGen Systems. Journal of Product Innovation Management 25 (3), 213–232.

Cooper, R. G. 2014. What’s next? After Stage-Gate. Research Technology Management 157 (1), 20–31.

Cooper, R. G. & Sommer, A.F. 2016a. Agile-Stage-Gate: New idea-to-launch method for manufactured new products is faster, more responsive. Industrial Marketing Management 59 (November), 167–180.

Cooper, R. G. & Sommer, A.F. 2016b. The Agile–Stage-Gate hybrid model: A promising new approach and a new research opportunity. Journal of Product Innovation Management 33 (5), 513–526.

Cousin, G. 2005. Case Study Research. Journal of Geography in Higher Education 29 (3), 421–427.

Deemer, P., Benefield, G., Larman, C. & Vodde, B. 2012. The Scrum Primer, Version 2.0. [online] Available at: http://scrumprimer.org/scrumprimer20.pdf [accessed 11.8.2018].

Dikert, K., Paasivaara, M. & Lassenius, C. 2016. Challenges and success factors for large-scale agile transformations: A systematic literature review. The Journal of Systems and Software 119, 87–108.

Dingsøyr, T., Fægri, T. & Itkonen, J. 2014. What Is Large in Large-Scale? A Taxonomy of Scale for Agile Software Development. Product-Focused Software Process Improvement 8892, 273–276. 10.1007/978-3-319-13835-0_20.

Eloranta, V., Koskimies, K., Mikkonen, T. & Vuorinen, J. 2013. Scrum Anti-patterns – An Empirical Study. 2013 20th Asia-Pacific Software Engineering Conference (APSEC), Bangkok, 503–510.

Friis Sommer, A., Hedegaard, C., Dukovska-Popovska, I. & Steger-Jensen, K. 2015. Improved Product Development Performance through Agile/Stage-Gate Hybrids: The Next-Generation Stage-Gate Process?. Research-Technology Management 58 (1), 34–45.

Golafshani, N. 2003. Understanding Reliability and Validity in Qualitative Research. The Qualitative Report 8 (4), 597–607.

Gustavsson, T. & Rönnlund, P. 2013. Agile adoption at Ericsson hardware product development. 10.13140/2.1.3781.3447.

Hancock, D. R. & Algozzine, R. 2006. Doing case study research. New York, NY: Teachers College Press.

Harling, K. 2012. An Overview of Case Study. SSRN Electronic Journal. 10.2139/ssrn.2141476.

Highsmith, J. 2002. What Is Agile Software Development? The Journal of Defense Software Engineering 15 (10), 4–9.

Karlstrom D. & Runeson, P. 2005. Combining agile methods with stage-gate project management. IEEE Software 22 (3), 43–49.

Karlstrom D. & Runeson, P. 2006. Integrating agile software development into stage-gate managed product development. Empirical Software Engineering 11 (2), 203–225.

Kerzner, H. 2013. Project management. 11th ed. New York, New York: Wiley.

Kothari, C. R. 2004. Research Methodology: Methods and Techniques. 2nd ed. New Delhi: New Age International Publishers.

Laanti, M. 2016. Piloting Lean-Agile Hardware Development. Proceedings of the Scientific Workshop XP2016, 1–6. 10.1145/2962695.2962698.

Larman, C. & Vodde, B. 2013. Scaling Agile Development. CrossTalk 26 (3), 8–12.

Lechler, T. 2002. Plans are Nothing, Changing Plans is Everything: The Impact of Changes on Project Success. Research Policy 33 (1), 1–15.

Mathur, S. & Malik, S. 2010. Advancements in the V-Model. International Journal of Computer Applications 1 (12), 29–34.

Munassar, N. M. A. & Govardhan, A. 2010. A Comparison Between Five Models Of Software Engineering. International Journal of Computer Science Issues 7 (5), 94–101.

Paasivaara, M., Durasiewicz, S. & Lassenius, C. 2008. Distributed Agile Development: Using Scrum in a Large Project. 2008 IEEE International Conference on Global Software Engineering, Bangalore, 87–95. 10.1109/ICGSE.2008.38.

Paasivaara, M., Lassenius, C. & Heikkilä, V. 2012. Inter-team Coordination in Large-Scale Globally Distributed Scrum: Do Scrum-of-Scrums Really Work?

Proceedings of the 2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, 235–238.

Paulik, M. C. 2001. Extreme programming from a CMM perspective. IEEE Software 18 (6), 19–26.

Petersen, K., Wohlin, C. & Baca, D. 2009. The Waterfall Model in Large-Scale Development. Product-Focused Software Process Improvement 32, 386–400.

Petersen, K. & Wohlin, C. 2010. The effect of moving from a plan-driven to an incremental software development approach with agile practices. Empirical Software Engineering 15 (6), 654–693.

PMI. 2013a. A Guide to the Project Management Body of Knowledge (PMBOK Guide). 5th ed. Newtown Square, PA: Project Management Institute.

PMI. 2013b. The Essential Role of Communications. [online] Available at: https://www.pmi.org/-/media/pmi/documents/public/pdf/learning/thought-leadership/pulse/the-essential-role-of-communications.pdf [accessed 25.10.2018].

Punkka, T. 2012. Agile Hardware and Co-Design. Embedded Systems Conference 2012, Boston, ESC-3008.

Punkka, T. 2016. Flexible New Product Development: Using Knowledge Transfer from Agile Software Development as a Catalyst for Adaptation – Case Study and Systematic Literature Review. Licentiate thesis, Aalto University. http://urn.fi/URN:NBN:fi:aalto-201602171404.

Reynisdóttir, Þ. 2013. Scrum in mechanical product development. Master’s thesis, Chalmers University of Technology.

Royce, W. 1970. Managing the development of large software systems: Concepts and techniques. Proc. IEEE WESCOM, IEEE Computer Society Press.

Scaled Agile, Inc. 2018. SAFe® 4.6 Introduction (white paper). [online] Available at: https://www.scaledagile.com/resources/safe-whitepaper [accessed 13.11.2018].

Schwaber, K. 1997. SCRUM Development Process. Business Object Design and Implementation, 117–134. 10.1007/978-1-4471-0947-1_11.

Schwaber, K. & Sutherland, J. 2017. The Scrum Guide, November 2017. [online] Available at: http://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf [accessed 11.8.2018].

Serrador, P. & Pinto, J. K. 2015. Does Agile work? — A quantitative analysis of agile project success. International Journal of Project Management 33 (5), 1040–1051.

Seymour, T. & Hussein, S. 2014. The History of Project Management. International Journal of Management & Information Systems 18 (4), 233–240.

Shields, M. D. & Young, S. M. 1989. A Behavioral Model for Implementing Cost Management Systems. Journal of Cost Management (Winter 1989), 17–27.

Stelzmann, E. 2011. Contextualizing agile systems engineering. 2011 IEEE International Systems Conference. 10.1109/SYSCON.2011.5929068.

Stack Overflow. 2018. Stack Overflow Developer Survey 2018. [online] Available at: https://insights.stackoverflow.com/survey/2018 [accessed 7.12.2018].

Stenbacka, C. 2001. Qualitative research requires quality concepts of its own. Management Decision 39 (7), 551–556.

Takeuchi, H. & Nonaka, I. 1986. The New New Product Development Game. Harvard Business Review 64 (1), 137–146.

Vaidya, A. 2014. Does DAD Know Best, Is it Better to do LeSS or Just be SAFe? Adapting Scaling Agile Practices into the Enterprise. PNSQC 2014 Proceedings, 21–38.

Zanotti, C. C., Kaylor, A. J. & Davidsen, K. L. 2017. Multi-discipline agile development and reliability and maintainability. Proc. 2017 Annual Reliability and Maintainability Symposium (RAMS), 171–173.

APPENDIX 1

List of the interviews:

Interview number	Interviewee	Length	Date
1	Project Leader A	1 h 35 min	27.03.2018
2	Product Owner A	1 h 30 min	19.06.2018
3	Scrum Master A	1 h 7 min	27.06.2018
4	Product Owner B	1 h 19 min	03.07.2018
5	Scrum Master B	47 min	04.07.2018
6	Project Leader B	1 h 10 min	05.07.2018

APPENDIX 2

The main interview questions:

1. What does your team do and what is the product?
2. What are your current duties and how much previous experience of using Scrum or other agile methods do you have?
3. What are the pros and cons of Scrum in hardware development in your opinion? Should something be changed to make the process more suitable for hardware development?
4. Has the line manager role changed after implementing Scrum?
5. Have the roles of the team members changed after adopting Scrum? Have their responsibility areas broadened or gotten smaller?
6. How many members do the Scrum teams have? Should there be changes to the team sizes? What roles are essential for each team?
7. How testing should be integrated into the Scrum process?
8. What is a good length for the Sprint? How to synchronize the Sprints with other teams?
9. How to manage the product backlog or backlogs between several Scrum teams?
10. What are the current meeting practices and how well do they work?
11. How good is the communication between Scrum teams and projects? Could it be improved somehow?