Thesis description

The purpose of this master thesis is to investigate how the Agile approach can be used by startups in the development of physical products. This is investigated by considering if startups use Agile frameworks for physical product development, and by examining how the use of Agile frameworks is different from what is described in literature.
Preface and acknowledgements

This master thesis is written by two master students, studying at the School of Entrepreneurship at the Norwegian University of Science and Technology (NTNU). The thesis represents the concluding scientific work of the authors.

The authors want to thank Vivek Sinha, who has been the supervisor for this paper. His feedback, support, and guidance are much appreciated. The authors would also like to thank Christer Elverum for his guidance in the field of Agile Product Development.

The authors also want to thank the four interviewed organizations for being so collaborative and helpful. Lastly, the authors are grateful for the assistance provided by Øystein Widding in reviewing the thesis, contributing valuable comments and opinions.
Abstract

The Agile approach has for last 25 years revolutionized information technology through its radical alternative to traditional sequential development processes. Agile frameworks are now spreading to improve the development of innovative physical products, and are expected to increase product success rates, speed to market, improved quality based on customer feedback, and boosting the motivation of the product development team.

The literature review in this paper identifies the gap between Agile Software Development and Agile Hardware Development, by an analysis of existing literature on Agile Software Development and Agile Product Development for startups. First, this paper proposes definitions and classifications of the Agile approach. Second, it analyses six Agile frameworks for physical product development up against traditional product development and Agile software frameworks. Third, it briefly compares Agile Software Development and Agile Hardware Development through existing literature on Agile Product Development, by highlighting similarities and differences.

This paper discovers a lack of literature about Agile frameworks for physical products, that Agile Product Development is a relatively new and an undiscovered field, and that there exist no frameworks only used for physical product development. Regarding literature for startups who want to develop physical products with an Agile approach, the authors have found very few, and almost no holistic approaches. Based on this, the authors have done case studies on four hardware startups, and have found several findings which are relevant for the development of the field.

The purpose of this paper is to investigate how the Agile approach can be used by startups in the development of physical products. This is investigated by considering if startups use Agile frameworks for physical product development, and by examining how the use of Agile frameworks is different from what is described in literature.

Findings from the case studies show that hardware startups mix concepts from different Agile frameworks. The startups only use some of the concepts from the Agile frameworks they claim to use, and customize their own frameworks by borrowing concepts from several other Agile frameworks. This can be loosely connected to the finding which shows that all the interviewees have a diverging understanding of what an iteration, a prototype and an increment is. This partly builds on existing theory, were experts state that managers don't understand the Agile approach even when they are confronted with it.
Further the study identifies a fundamental difference between the development of hardware and software products. The length of an iteration seems to vary to a large degree between the case studies, and iterations are much longer than what is presented by existing literature. A consequence of this, shown in other findings, is that the frequency of testing becomes much lower, customer interaction becomes rarer, and taking a product to market takes longer compared to software development.
Sammendrag

Agile tilnærmingen har i de siste 25 årene revolusjonert informasjonsteknologi gjennom sitt radikale alternativ til tradisjonelle utviklingsprosesser. Agile rammeverk sprer seg nå for å forbedre utviklingen av innovative fysiske produkter, og forventes å øke produktsuksessrate og hastighet til marked, forbedre kvalitet basert på tilbakemeldinger fra kunder, og øke motivasjonen til produktutviklingsteamet.


Gjennom oppgaven er det oppdaget mangel på litteratur som omhandler Agile rammeverk for fysiske produkter, at Agile Product Development er et relativt nytt og uoppdaget felt, og at det ikke eksisterer noen Agile rammeverk som bare brukes til fysisk produktutvikling. Når det gjelder litteratur for oppstartselskaper som ønsker å utvikle fysiske produkter med en Agile tilnærming har forfatterne funnet svært få, og nesten ingen helhetlige tilnærminger. Basert på dette har forfatterne gjort case-studier på fire hardware-oppstarter, og avduket flere funn som er relevante for utvikling av feltet.

Formålet med masteroppgaven er å undersøke hvordan den Agile tilnærmingen kan brukes av oppstartselskaper i utviklingen av fysiske produkter. Dette undersøkes ved å vurdere om oppstartene bruker Agile rammeverk for fysisk produktutvikling, og ved å undersøke hvordan bruken av Agile rammeverk er forskjellig fra det som er beskrevet i litteraturen.

Resultater fra case-studiene viser at hardwareoppstarter blander konsepter fra forskjellige Agile rammeverk. Oppstartene bruker bare noen av begrepene fra det Agile rammeverket de hevder å bruke, og tilpasser egne rammeverk ved å låne konsepter fra flere andre Agile rammeverk. Dette kan være løst knyttet til funnene som viser at alle intervjuobjektene har en ulik forståelse av hva en iterasjon, prototype og inkrement er. Dette bygger delvis på
eksisterende teori, der ekspert sier at managere ikke forstår Agile tilnærmingen, selv når de konfronteres med den.

Videre identifiserer studiet en grunnleggende forskjell mellom utviklingen av hardware- og software-produkter. Lengden på en iterasjon synes å variere i stor grad mellom casene, og iterasjonene er mye lengre enn det som er presentert av eksisterende litteratur. En konsekvens av dette er, som vist i andre funn, er at frekvensen for testing blir mye lavere, kontakt med kunde skjer sjeldnere, og at det å ta et produkt til markedet tar lengre tid, sammenlignet med software-utvikling.
Table of contents

1. Introduction .......................................................................................................................... 1
   1.1 The Agile approach to physical product development .................................................. 2
   1.2 Research questions ......................................................................................................... 4

2. Theory .................................................................................................................................. 6
   2.1 Product development ....................................................................................................... 7
   2.2 Agile Product Development .......................................................................................... 8
   2.3 Agile in startups – Entrepreneurship is Agile innovation ............................................ 14
   2.4 Agile frameworks for physical product development .................................................... 16
   2.5 Agile Software Development and Agile Hardware Development ............................. 29
   2.6 Other Agile terms, Agile movement and rhetorical techniques .................................... 35
   2.7 Summary and discussion of theoretical findings ............................................................ 36

3. Method ................................................................................................................................. 38
   3.1 Research design ............................................................................................................. 38
   3.2 Data acquisition ............................................................................................................. 40
   3.3 Analysis of data ............................................................................................................. 41
   3.4 Reflections of method .................................................................................................... 42

4. Case studies ......................................................................................................................... 46
   4.1 Case 1 - Robotics .......................................................................................................... 46
   4.2 Case 2 - Wearables ........................................................................................................ 51
   4.3 Case 3 - Connected devices .......................................................................................... 55
   4.4 Case 4 - Designed products ......................................................................................... 61
   4.5 Case studies summarized ............................................................................................... 67

5. Findings ................................................................................................................................ 71
   5.1 Key finding 1: All the interviewees have a diverging understanding of what an iteration,
   a prototype and an increment is .......................................................................................... 71
   5.2 Key finding 2: The length of an iteration seems to vary to a large degree between the
   case studies ......................................................................................................................... 75
5.3 Key finding 3: The frequency of testing for hardware startups is lower compared to the Agile Manifesto, and there is large variation between different hardware startups. 77

5.4 Key finding 4: Hardware startups mix concepts from different Agile frameworks. 79

6. Analysis and discussion
6.1 Analysis and discussion of key finding 1 82
6.2 Analysis and discussion of key finding 2 86
6.3 Analysis and discussion of key finding 3 88
6.4 Analysis and discussion of key finding 4 92

7. Conclusion
7.1 Theoretical implications 97
7.2 Practical implications 98
7.3 Limitations 99
7.4 Further research 99

References 101
Appendix 103
A. Interview guide 103
B. Agile Manifesto – values and principles 104
1. Introduction

The Agile approach has revolutionized information technology, through increased success rates in software development, improved quality and speed to market, and through boosting the motivation and productivity of IT teams (Rigby et al., 2016b). The Agile approach, with its values and principles, is a product development approach which is spreading across a broad range of industries and functions. Literature shows that Agile’s roots extend far beyond information technology and experts expect that the Agile approach will continue to spread to improve innovation processes in nearly every function of every industry. (Rigby et al., 2016a)

Managers are now increasingly beginning to apply the Agile approach to the development of physical products (hardware) (Cooper and Sommer, 2016). Agile Product Development is the overarching field for both Agile Software Development and Agile Hardware Development, and is a relatively undiscovered field regarding literature. Most literature available on Agile Product Development is about software development, and to a small degree about hardware development. Even more limited is the literature on Agile Product Development for startups. As opposed to well-established firms, most startups have to be more Agile, because they have less resources and people, and work with an unfinished product they have never made before. (Visser and Selnes, 2016)

Therefore, the purpose of this study is:

*To investigate how the Agile approach can be used by startups in the development of physical products.*

In this master thesis, we present the results of a literature review and case studies of hardware startups and their use of the Agile approach. As the Agile approach is a broad concept and covers a wide field, in this chapter, we present the relevance of the Agile approach for product development in startup firms, building upon some definitions and the historical development of the concept. We then present our research questions and the structure of the paper.
1.1 The Agile approach to physical product development

The use of Agile Product Development dates back as far as the mid-1950s (Larman and Basili, 2003), and were according to Cockburn and Highsmith (2001) proposed as a reaction to the traditional methods used at that time. The traditional methods did not account for the increasing changes in the business environment, as they assumed no changes to requirements would happen throughout the project lifecycle (Abbas et al., 2008).

According to Schuh et al. (2016), traditional product development is suitable for evolutionary product development, but is considered too slow, linear and little adaptive for the management of radical product innovation. The principle of an early product definition in the traditional process does not take into consideration the fact that customers usually do not clearly know what they want until they see a prototype with their own eyes. (Schuh et al., 2016)

Over the years, Agile Product Development has come to encompass a broad range of ideas related to agile in product development. These can be understood through the definition of agile, the core characteristics of the Agile approach based on the Agile Manifesto, and through an introduction to Agile frameworks.

Agile within product development is defined by Böhmer et al. (2015) as:

"The capability to react and adapt to expected and unexpected changes within a dynamic environment constantly and quickly; and to use those changes (if possible) as an advantage."

The terms agile and agility are often associated with software development. However, the terms are not unique to either manufacturing, new product development nor software development. (Kettunen, 2009).

Although many of the Agile ideas have been around since the 1970s, it has been most noticeable with its change to software development in the last fifteen years (Abbas et al., 2008). The use of Agile Software Development has significantly increased since the Manifesto for Agile Software Development was published in 2001. The Agile Manifesto¹, shown in the Appendix, set out to establish values and principles to guide a better approach to software development (Abbas et al., 2008). The authors of the manifesto later stated that the

The Agile approach is useful for all types of projects that face uncertainty, not only in the software development domain, where its usage is widespread (Stare, 2014).

Regardless of the field, the Agile approach can be broken down to its core characteristics:

- **Values**: “general assumptions framing the Agile view of the world.”
- **Principles**: “core Agile rules, organizational and technical.”
- **Practices**: “specific activities practiced by Agile teams.”
- **Roles**: “responsibilities and privileges of the various actors in an Agile process.”
- **Artifacts**: “tools, both virtual and material, that support the practices.”

The principles follow from the values, and the practices, roles, and artifacts follow from the principles (Meyer, 2014). The complete list of the core characteristics can be found in section 2.5.2.

The Agile approach is implemented through Agile frameworks. Agile principles and values make up the Agile frameworks and a way of thinking about development. The word method is included in the understanding of frameworks, and can sometimes be used interchangeably. (Meyer, 2014, Böhmer et al., 2015)

Most Agile frameworks available have been made with software in mind. However, some frameworks, like Kanban and Scrum, are originally made for physical product development and have been widely applied to the software domain (Dybå and Dingsøyr, 2008). Scrum for example, was introduced by Takeuchi and Nonaka (1986) almost 30 years ago. From the starting point of the 1986 article, together with several studies done on product management, Ken Schwaber and Jeff Sutherland codified and created the Scrum approach. This was released through the 1995 paper: "Scrum development process" by Sutherland. (Rigby et al., 2016a). Scrum and Kanban have been used to develop new types of bicycles and sports cars, and Design Thinking, another Agile framework, is used across industries. (Böhmer et al., 2015).

Various articles mention Agile frameworks which can be used in physical product development, and according to Rigby et al. (2016a), all development frameworks that align with the Agile Manifesto’s values and principles, regardless of software, would be known as Agile frameworks.

Rigby et al. (2016a) state that even though Agile Software Development is widely used, Agile began outside of IT and is therefore relevant for physical product development as well. The
The iterative nature of Agile requires that many prototypes are built, which makes it contemporary for Agile Product development. Hardware prototyping was usually seen as very resource-demanding, but now, with new rapid prototyping technologies and the development of affordable 3D-printers, it has become easier and cheaper to do frequent prototyping. (Vetter, 2011)

Even though software and physical products are different, non-software developers can use Agile Software Development literature for inspiration and adaptation (Larman and Basili, 2003). Managers are beginning to apply the Agile approach to physical product development as well (Cooper and Sommer, 2016). Several companies have started to integrate elements of Agile development techniques in their traditional product development process (Cooper, 2014).

According Crichton (2014), Silicon Valley and other startup ecosystems, have for the last 30 years been focusing on software startups, and this is now changing. Hardware startups are getting increasingly more investments and the attention in the press is growing; it’s all about developing hardware at the speed of software (Statt, 2015, Constine, 2015).

Also for startups, who are searching for a repeatable business model, through product differentiation in changing markets (Blank and Dorf, 2012), traditional methods are likely to be hindering. Often with a high level of project uncertainty, there is a higher possibility that a project needs to change at some point in the product development stage. (MacCormack and Verganti, 2003).

1.2 Research questions
To be able to investigate how the Agile approach can be used for physical product development, it is necessary to understand where Agile came from, where and how it is currently used, and if it is possible to integrate it with existing management. The definitions and background information presented above established the context for the research questions. Taking this into consideration, and to reach the outlined purpose, the authors will examine the following research questions (RQs):

1. Do hardware startups use Agile frameworks for physical product development?

2. How do hardware startups use Agile frameworks in practice?
Research question one (RQ 1) is chosen because Agile frameworks are used to a large degree within software development and could for that reason be relevant for hardware startups as well. Experts have told the authors that RQ 1 should be answered through an exploratory study, because the field of research is relatively unexplored in literature. This result of this should be an improved research design. RQ 1 does most likely not have a clear answer, since Rigby et al. (2016b) state that Agile frameworks often are combined and customized. It is therefore expected that if Agile frameworks are used, they have been customized, combined or only used to some extent by the case studies.

Agile Product Development has mostly been developed in the software domain, and since literature implies that there is a difference between software and hardware, the authors of this paper wish to research what the implications would be in practice and theory when hardware companies use Agile frameworks. Research question two (RQ 2) is therefore chosen to investigate the link between how hardware startups use Agile frameworks and how Agile frameworks are described in literature.

The structure of this paper is based on a literature review and case studies on startups. A literature review is presented in chapter two, chapter three presents and elaborates on the qualitative research design. Chapter four introduces the four case studies, and chapter five will present the key findings. Chapter six will analyse and discuss these findings up against literature, and chapter seven will conclude this master thesis.
2. Theory

This chapter presents the results and findings from the literature review, and provides an overview of the previous academic research on the Agile approach.

First an introduction to the field of product development is presented to make the reader understand the context of this paper. Further, Agile Product Development will be presented (in chapter 2.2), and is compared up against traditional product development. When doing a literature review on the Agile approach for hardware startups, the authors of this paper were surprised that the Agile approach is almost always discussed up against traditional approaches. This could be because Agile is a relatively new term to product development, and since most product development companies still use traditional development methods or a combination of both traditional methods and Agile approach. This paper will therefore include explanations of the differences, to be able to understand the landscape where and how Agile development is used. As the introduction chapter pointed out, Agile Product Development has mostly been developed in the software domain, and chapter 2.2 will therefore also contain what Agile Software Development and Agile Hardware Development is, and how the principles from Agile Software Development can be employed.

Then, a link between agility and Entrepreneurship will be presented in chapter 2.3. According to Rigby et al. (2016b), innovation is what Agile is all about. Since the case studies study hardware startups, the literature review contains perspectives and views on how Agile can be used for innovate product development projects.

Sub-chapter 2.4 contains an overview of all the Agile frameworks which can be used for physical product development, discussed up against traditional management and Agile Software Development.

The next sub-chapter (chapter 2.5) will show an overview of the differences and similarities between Agile Software Development and Agile Product Development considering the sub-chapters described above. Through a broad overview of where the Agile approach fits within the technology R&D spectrum, it becomes clear what literature points out to be a list of core Agile characteristics for physical product development.

Finally, other Agile terms, the Agile movement and rhetorical techniques will be presented (chapter 2.6), followed by a conclusion and discussion of the theoretical findings (chapter 2.7).
2.1 Product development

The role of product development in companies is becoming increasingly more important. Product development is defined as: “the process until a product can be used: starting with the product planning and the search for ideas, the definition of the product, respectively with the single part production: starting with the order up to the delivery of the product at the customer (Ehrlenspiel, 2003). Through this process, the product gets more and more concrete with every prototype-cycle. The customer of this product can be determined or not when development starts. (Dombrowski et al., 2011)

Product development is typically divided phases which in practice overlap depending on such factors as the product, the organization of the enterprise and the available resources. Roughly these phases are; research, product planning with the customer, design, and development which creates a prototype, testing of this prototype, manufacturing and assembly. (Dombrowski et al., 2011)

Over the years there has been an increasing focus on the processes that assure implementation, and therefore better management of the first parts of the development process, measuring this process and continually improving processes. (Cooper, 2001)

Leonard-Barton (1992) argues that the development of new products and processes is a critical strategic activity for a company wanting to innovate and differentiate. New product development (NPD) is typically seen as the development of new products that have been on the market five years or less (including continuous improvements). Within NPD most companies use some form of the Stage-Gate process. (Cooper, 2001)

Krishnan and Ulrich (2001) define product development as: "the transformation of a market opportunity into a product available for sale." Their extensive review paper focuses on the development of physical goods and technological innovation within single firms, in contrast to other literature, which addresses innovation at the level of an entire industry or an entire firm. (Krishnan and Ulrich, 2001)

As a perspective, product development can be seen as a deliberate business process, where hundreds of decisions need to made in all parts of the organization. This decision perspective makes sense because “while how products are developed differs across firms and within the same firm over time, what is being decided seems to remain fairly consistent at a certain level of abstraction.” (Krishnan and Ulrich, 2001)
Product development has established fields within marketing, organizations, engineering design, and operations management. There are decisions to made within a development project and decisions in setting up a development project. When considering product development, the organization of firms into functions such as engineering, marketing, and operations introduces a discontinuity in ongoing operations. Product development decisions are typically clustered, and multiple functions in the firm are responsible for the same decisions. (Krishnan and Ulrich, 2001). Frameworks support what is going to be decided, and Agile frameworks, with its values and principles, are therefore relevant within the domain of NPD.

From the above, it is clear that product development is a crucial function in a firm, and the different authors of the articles agree that improvements there can yield significant benefits. The Agile approach concepts are therefore very relevant for businesses seeking to extract maximum benefit from improving their product development process.

2.2 Agile Product Development

When talking about Agile Software Development and Agile hardware development, Agile Product Development is the overarching term. Agile Product Development is not a framework in itself and can be seen as an umbrella term which describes several Agile frameworks. How these fields relate to each other, can be seen in figure 11 in 2.4.

From the articles the authors have found, Agile seems to be relatively unknown as a hardware development process. Most articles which mention Agile Product Development, reference Agile Software Development as if it was the Agile origin.

From the numerous articles available for Agile Software Development and the lack of articles for Agile hardware development, it seems that it has been forgotten that Kanban and Scrum originally are Agile Product Development frameworks.

The authors of this paper have not found Agile frameworks only applicable to hardware, but have found several Agile frameworks which can be applied to both software and hardware. Some frameworks, like Feature Driven Development (FDD), seem to be only applicable software. Abrahamsson et al. (2002) state that the FDD approach is based on best practices found to be useful in the software industry.

According to Stare (2014), almost all of the research studies on Agile methods up to 2009 have focused only on software. The one article Stare (2014) refers to as Agile Product
Development is actually about software only. Stare (2014) has also not found Flexible Product Development - first described by Smith (2007) - as a field, or found Scrum as general Agile Product Development framework. This reflects the difficulty of finding relevant articles. Combined with our results, it tells that Agile Hardware Development is a new or nonexistent field. Agile Product Development itself is a narrow and small field.

Of the articles the authors have found, almost all product development papers reference Agile Software Development. The articles also typically criticize traditional product development. It is also difficult to measure efficiency between different ways of performing product development, which is why product development is not described widely in scientific papers. (Ottosson, 2004)

In the following sub-chapters different product development processes will be elaborated. First, the Agile approach will be compared up against traditional product development. Then, Agile Software Development and Agile Hardware Development will be described. Lastly an overview of the terms iteration, increment and prototype are explained to make the reader understand the basic concepts related to the Agile approach.

2.2.1 Traditional product development and the Agile approach
The search for Agile hardware frameworks has also led to findings of the importance of traditional development in combination with Agile. It is a waste of time to “reinvent the wheel”, which means that it is important to consider traditional approaches as a supplement to Agile (Ottosson, 2004). (Cooper, 2016), the inventor of the popular hardware development model Stage-Gate, explains that Agile and Stage-Gate also should be used in combination. Since stage-gate is a macroplanning process and Agile is a micro planning project management methodology, they can be used in combination for the development of physical products. Agile methods are being recognized within traditional system development organizations, and most organizations seem to sustain both forms of development (Vinekar et al., 2006). Rigby et al. (2016b), on the other hand, states that Agile, with new values, principles, practices, and benefits are a radical alternative to many existing development techniques. Vinekar et al. (2006) also recognizes that the two alternatives are radically different, and he studies how an organization can combine the two.

The most widely used traditional physical product development model is the stage-gate model. In the stage-gate process, product development is divided into stages, separated by gates. At each gate, there is a decision whether the process should continue (Schuh et al.,
Classical development has been developed since the 1970s and has mainly existed in the automotive industry. Existing traditional methods, such as Integrated Product Development (IPD), Concurrent Engineering (CE), Simultaneous Engineering (SE), were developed for re-engineering of existing products (Ottosson, 2004). Stage-Gate®, PACE (Product and Cycle-Time Excellence), NPI (New Product Introduction), PDP (Product Development are phased product development systems and encourage substantial up-front planning and sticking to this plan. (Smith, 2007). Ovesen and Sommer (2015) report that some companies in the domain of IPD, Scrum is being integrated as an additional process, instead of a substitute.

The link between traditional plan-driven software engineering and Agile approaches are drawn by several authors (e.g., (Boehm and Turner, 2004) (Cooper, 2014) (Schuh et al., 2016)). In traditional software development, one of the most popular models used is the waterfall method (Baskerville et al., 2003) (Schuh et al., 2016).

Traditional development attempts to minimize the change in the course of the project through an extensive gathering of requirements before the actual development starts. Analysis and design are also done in the beginning to obtain higher quality results under a controlled schedule. (Cockburn and Highsmith, 2001). Agile, on the other hand, focuses on adaptation and innovation rather than prediction and control. (Vinekar et al., 2006)

It can also be noticed that iterations and increments are at odd with traditional development, where a working product often is provided at release or the end of the project. Traditional methods typically plan most development activities before starting and execute them in sequence and sometimes overlap them for speed. This can be seen in figure 1 below. (Smith, 2007)
While traditional approaches are useful, Agile offers several major benefits when the two they are compared through studies and documentations. (Rigby et al., 2016b)

2.2.2 Agile Software Development

“The new agile methods have evoked a substantial amount of literature and debates. However, academic research on the subject is still scarce, as most of the existing publications are written by practitioners or consultants.” - Abrahamsson (2002)

A comprehensive literature review was done by Abrahamsson et al. (2002), to clarify the meaning of Agile Software Development (ASD) and the various applications and methodologies. In principle, they state, ASD is a method focusing on simplicity and speed, team work, customers and especially, working code. In their report, they concluded a definition of what makes software development agile. The definition is 'incremental, cooperative, straightforward and adaptive', which means small software releases, customers and developers work close, that the method is well documented and that it is possible to make last moment changes. Several different methodologies of ASD are described and compared, including (1) Extreme programming, (2) Scrum, (3) Crystal Method, (4) Feature-Driven Development, (5) Adaptive Software Development, (6) Rational Unified Process, (7) Dynamic Systems Development Method, and (8) Open Source Software development, where Extreme Programming has been widely accepted as the foundation for numerous software development methodologies. According to their findings, there are no practices available for a startup to find the 'right' method, as the methodologies mentioned cover different phases and seem 'tailored' for various types of situations. Further, no distinct boundaries are obtained to differentiate ASD from more conventional methodologies, which makes it difficult to spot
whether a practice is 'agile' or not. Abrahamsson (et al.) mention that agile practices have been used in software development since the 1960s, and recently more research has been conducted on the advantages and disadvantages by utilizing these methods.

Still, a summary of 1996 studies conducted by Dybå and Dingsøyr (2008) found that the strength of evidence is very low, and encourage more research on the investigation of how organizations benefit from the use of ASD.

Another comprehensive literature search on Agile Software Development, done by (Chuang et al., 2014), also shows that the academic field is relatively new, even though Agile Software Development is becoming increasingly more mainstream.

2.2.3 Agile Hardware Development

Despite the growing popularity of Agile Software Development, the field Agile Hardware Development has not yet been established (Stare, 2014). On the other hand, Smith (2007) popularized flexible product development (FPD), which can be seen as the Agile Product Development for physical products. FPD draws on the roots of Agile Software Development, and uses additional tools and techniques which apply beyond the software domain. When FPD is being implemented to the development of software products, it can be recognized as Agile Software Development. (Smith, 2007)

The lack of articles in this field is also reflected by Stare (2014) who states he only found one article which explored the Agile approach in physical product development projects. The Berger and Beynon-Davies (2009) article Stare (2014) references do not even directly talk about physical product development, and it seems like Stare (2014) it is trying to make a fit.

There are a few examples of Agile Product Development being used in hardware development. John Deere, a farm equipment company, has used Agile Product Development to develop new machines, and Saab has used it to produce new fighter jets. Often the adaptation and expansion of the Agile approach begin within the company, with a software engineer or manager who has started working on a physical product development project. (Rigby et al., 2016b). Also, the companies Wikispeed and Johnson&Johnson, mentioned previously, are doing Agile hardware development.

Even though Agile has become a popular term, and hardware companies claim they are becoming more nimble, managers don't understand the Agile approach when they are confronted. (Rigby et al., 2016b)
Schuh et al. (2016) combines research on highly iterative innovation processes for the fast realization of physical product ideas and combines these with Scrum. An adapted Scrum process model for physical product development can succeed by demonstration and specific characteristics from Agile. Still, since the vast majority of physical product development projects use traditional development models, Agile has to be modified according to the companies’ environment. Schuh et al. (2016). Ovesen and Sommer (2015) conclude with the same, stating that “Scrum is an additional framework that is only utilized to some extent.”

Flexible techniques must be used with discretion, as it can’t be used for all product categories, for instance, only the parts of a product likely to undergo change are recommended. Some techniques are perfect for one particular project and terrible for another. Each project should be evaluated separately, and the use of flexible techniques must be tailored to the project to make them beneficial, not harmful. (Smith, 2007). This fits with Stare (2014), who argues that only certain parts of the Agile approach will be widely used in physical product development and that there are many implementation challenges.

2.2.4 Iterations, increments and prototypes
At the heart of the Agile approach are the terms iterations, increments and prototypes. This section will briefly explain what the differences are and how they are related. This is important for understanding the literature findings, case findings and discussions. This section is directly related to key finding 1 and 2.

Smith (2007) is the only literature the authors have found which seems to cover an unbiased view of Agile Product Development, and describes iterative and increment in the following way:

- Iterative development is when a product is developed in several short cycles rather than in one linear sequence.
- An increment is one release to the customer.

An iteration is often a time-boxed (fixed time slot) set of activities, lasting from a week to several weeks. Good practice is to provide a prototype or working sample at the end of the iteration, to enable customer feedback. (Smith, 2007)

A product can therefore go through several iterations within an increment and have several increments before the product is complete (Smith 2007). A visual explanation of this is found in 2.1.1 in figure 1.
A prototype on the other hand is an initial model, sample, or release of a product or part of a product. It allows engineers and designers to see what works and what does not. New products often go through several rounds of prototypes before the design is finalized. A prototype may replicate only one part of a design. A prototype could create the look of the object without functionality, or the functionality without the look, and can be categorized in many ways. (Blackwell and Manar, 2015). A prototype is part of an iteration and becomes an increment if the prototype is not discarded in the iteration cycle.

2.3 Agile in startups – Entrepreneurship is Agile innovation

Radical innovations are characterized by a high degree of both market and technological uncertainty (Lynn and Akgün, 1998). For startups, who are searching for a repeatable business model, through product differentiation in changing markets (Blank and Dorf, 2012), traditional methods such as the waterfall method and stage-gate are likely to be hindering. Often with a high level of project uncertainty, there is a greater possibility that a project needs to change at some point in the product development stage. (MacCormack and Verganti, 2003)

Schumpeter (1934) describes economic development as a historical process of structural changes driven by innovation. One of his five types of innovation is using and applying new methods of production or sales of a product not yet proven in the industry. Agile is one of these unproven methods and can be related to traditional innovation management by viewing Agile management as a speedometer. Agile can be used to measure the speed of an innovate product development project. (Morris et al., 2015).

Highsmith (2009) states that innovation and faster development of a product are not enough. Companies need to be able to evolve a product close to the end of the development lifecycle, to be able to create ultimate customer value. Companies who realize that the "ultimate customer value is delivered at point-of-sale, not point-of-plan," will have a huge competitive advantage. A growing number of successful innovative companies are using an Agile approach, which can be seen as a solution to the gap between customers needing new products and companies delivering new products. Within product development, around 60% of newly launched products fail in the market. (Highsmith, 2009)

Early phase development of innovative solutions is not widely described in scientific papers because it is hard to do classical research on unpredictable situations (Ottosson, 2004). From the author's findings, innovate Agile development for physical product development is not
written about. The authors have collected the most relevant findings related to this paper below:

- Boehm and Turner (2004) equate entrepreneurship with agility since a strong discipline without agility would result in an inflexible hierarchy and project stagnation.
- Agile focuses on adaptation and innovation rather than prediction and control. (Vinekar et al., 2006)
- The more innovative your product, the more likely you are to make changes during its development. (Smith, 2007)
- Between 30% and 60% of all newly launched products fail in the market, because of resistance to change in the organization. “In practice, entrepreneurs can be innovative and possess technical skills, but can also lack management process knowledge to the detriment of long-term venture success” (Morris et al., 2015)
- The book Agile Innovation by (Morris et al., 2015) combine Agile and classical innovation and state that the word software in the Agile Manifesto can be replaced with innovation. This is very important, as it justifies why Agile Software Development and Agile Product Development are discussed interchangeably in some contexts.
- According to (Rigby, et al. 2016b), innovation is what Agile is all about. They state that Agile methods usually work best in software applications, where the problem to be solved unknown, complex and product requirements typically change during the project.
- Even though no article mention startups explicitly, Ries' (2011) Lean Startup states that his Agile methodology works for "any human institution designed to create a new product or service under conditions of extreme uncertainty."

Agile frameworks can be placed in a product innovation process like shown in figure 2 below. This illustrates which phases of an innovation process the pictured frameworks can apply.
2.4 Agile frameworks for physical product development

Agile frameworks for physical products can, in general, be explained through the origins the Agile framework Scrum. Takeuchi and Nonaka (1986) compared Agile Product Development with the scrum strategy used in the sport Rugby. In Rugby, the ball (seen as the idea in product development) is passed between the players (seen as the different stakeholders of the company), and the whole team moves together forwards to the goal line. This can be seen in contrast to the traditional product development way of passing the ball in very sequential matter from design to engineering, economy, and marketing. If marketing drops the ball at the end of the development process, the project has to start over.

It can be important to notice that Scrum and Agile approach first of all focus on execution of a project, and not planning the whole project life cycle. In principle, all steps in the process (defining requirements, designing technical aspects, execution, and testing and closing the process) remain the same. The differences lie in the degree of planning (where only a rough sketch could be needed), and the execution phase (which lasts much shorter).

In this sub-chapter, Agile frameworks used for physical product development are presented, starting with a traditional framework used as an explanation of the opposite. An example of an Agile software framework is presented in the end. For easier comparison of the frameworks, the Agile frameworks are structured with a general overview, followed by a description of the process and characteristics.
2.4.1 The waterfall method - the opposite

The traditional *waterfall method*, illustrated below, can be seen as the opposite of Agile methods. With the advent of the personal computer in the 1980s, it was easy to make Gannt charts with detailed descriptions of the whole project. Every delivery date, milestone, goal, and sub-goal could be laid out in detail. One main task leads to next and cascades down to the next like a waterfall (Sutherland et al., 2014).

![Figure 3: The waterfall model: a sequential process, adopted from (Sutherland et al., 2014)](image)

According to Sutherland et al. (2014), the problem with them is that they are always wrong. Requirements that change in the process is common and strict time schedules are almost never met.

In other words, the waterfall method decides variables such as requirements, resources, technologies and tools at the beginning of a project and does not change them. If something in the process fails, everything is back to start. (Schwaber, 2004)

Compared with this traditional method, it can be important to notice that Agile approaches in general first of all focus on execution of a project, and not planning the whole project life cycle. In principle, all steps in the process remain the same. The differences lie in the degree of planning, where only a rough sketch could be needed, and the execution phase, which lasts much shorter.
2.4.2 Scrum framework

Scrum is regarded as one of the process models which follows the rules of the Agile Manifesto. Takeuchi & Nonaka (1986) were the first to mention Scrum as a concept for development.

Scrum emphasizes an empirical process, collecting data to make decisions, rather than a defined process set at the beginning of the whole development. In Scrum, the development is split up in short iterative work cycles called sprints, where the variables mentioned in Figure 4 are continuously revised and controlled (Schwaber, 2004)

Next to being iterative, Scrum is incremental. A product release may require multiple sprints, each iteration of work builds on the previous increment and is often replaced when learning has occurred in previous work. (Schwaber, 2004)

In each sprint team members focus on completing tasks from the sprint backlog, derived from the tasks from the product backlog. Scrum emphasizes self-organizing teams, through sprint planning meeting at the beginning of the sprint, daily scrum meetings at the beginning of each day, and sprint retrospect meetings at the end of the sprints. At the end of each Sprint, there is a potentially shippable product ready, no matter how simple it is. This simple prototype can be iterated on through feedback from the stakeholder (Schwaber, 2004).

![Figure 4: The Scrum framework, with the circular arrow representing a sprint iteration, copied from (Ovesen and Sommer, 2015)](image-url)
Phases

P1 - Planning: All features desired for integration are put in a product backlog list, and prioritized. This list is constantly updated. Definition of teams and tools necessary for the project are done here.

P2 - Architecture: Based on features listed in the product backlog, the basic design of the system is planned. What features to be added in which iteration is decided.

P3 - Development: System is developed through sprints, each typically 2-4 weeks long (Schwaber, 2004). A definition of 'done' is necessary to ensure that requirements are met. Different variables are observed and controlled through Scrum practices. There are typically 3-8 sprints in this phase.

P4 - Postgame: When all new features are added the system is qualified for distribution. This phase includes testing and integration into customers' systems.

Characteristics

- Small teams - (5-9, recommended by (Schwaber, 2004))
- Sprints / Iterations
- Self-organizing teams
- Freeze requirements - during an iteration, no new feature should be added to the backlog. This to ensure focus.
- Daily scrum meetings - track progress and plan new features to be added.
- Task board - to visualize progress.

Do not provide any specific software development practices to be used (Schwaber, 2004)

2.4.3 Lean startup

"Lean, Kanban, and their hybrids are legitimate applications of Agile values and principles. (Rigby et al., 2016a)"

The Lean Startup, written by Eric Ries (2011) uses principles from Agile development and Lean manufacturing, to let startups make better and faster business decisions. The book defines a startup as "a human institution designed to create a new product or service under conditions of extreme uncertainty" (Ries, 2011)
The Lean Startup emphasize prototyping, building a minimum viable product (MVP), and to be able to go through build, measure, learn cycles as fast as possible. The success of this MVP is measured, a conclusion is drawn, and hopefully, validated learning is achieved. The learning, which occurred because hypotheses were validated or not, help to decide if the next move is to pivot or persevere with the current product concept. The length of the build-measure-learn cycle is to be minimized to achieve high adaptability and is run iteratively and incrementally if needed. (Ries, 2011)

Rigby et al. (2016a) argue that Agile methodologies inspired the development of the Toyota Production System, which is the primary source of today's lean thinking. Since Ries' Lean Startup focuses on customer collaboration, Agile practitioners have accepted lean development and other hybrids as legitimate applications of Agile values and principles. (Rigby et al., 2016a)

![Figure 5: the Lean Startup process and cycle, adopted from Ries (2011)](image)

**Phases**

**P1 - Build:** Start out first with building a minimal viable product (MVP) based on hypothesis. Later new builds will be based on both customer feedback and experience from

**P2 - Measure:** Innovation accounting is central here. The MVP is used to establish real data, and measure effectiveness in the marked

**P3 - Learn:** The experiments' results are used to obtain validated learning, from which the startup can decide to continue or pivot - meaning changing - the product.

**Characteristics**

- Speed
- Iterations
- Incremental
- Customer feedback
- Minimal-viable-product
- All team sizes
2.4.4 Kanban

Kanban is a variant of Lean development, influenced by processes developed for Toyota and seeks to minimize work-in-progress by ensuring just-in-time production, driven by demand (Meyer, 2014). In itself, Kanban is typical for physical product development. There is so far no exact Kanban method for software, but WIP-minimization has been found useful to identify for example impediments in Scrum and focus teams on most productive tasks (Meyer, 2014). The main focus of Kanban is to articulate what task is needed to be done, and when. This is done by prioritizing the tasks on 'Kanban cards', often illustrated on a 'Kanban board' with columns for different stages as 'To do, 'Doing', and 'Done'. Cards will be transferred to the next stage when finished, and each stage can only store a fixed amount of cards at a time, to minimize work-in-progress. This enables bottlenecks to be identified and resolved and allows the user to increase or decrease the work-in-progress limit in each column to increase the efficiency of the workflow.

![Kanban Board Diagram](image)

Figure 6: Visualization of tasks in Kanban

Anderson (2010), one of the first to formulate Kanban, propose five core properties. Notice that these are not phases of different processes, but rather simple rules of the methodology.

**Visualization:** Usually a board is used to visualize workflow. This contains features and works to be conducted or problems to be solved.

**Limitation:** For each category column only specific number of work items are allowed. This will ensure identification of bottlenecks which in turn will help to streamline the process.

**Policies:** The team establishes precise rules for how the process is managed. This includes, for example, colored sticky notes for specific meanings or rules of when to move cards from one column to another.

**Measure:** Gather data, as bottlenecks and average cycle times, for comparison of project progress.
**Improvement**: By resolving bottleneck, the work can be managed to deliver the highest value in the shortest amount of time.

**Characteristics**

- Typical for physical product development
- Visualization
- Minimizing work in progress - features are not implemented unless the customer asks for it.
- Incremental
- Customer feedback
- Team focus

Kanban is no alternative for existing processes or other frameworks but rather a supplement which can be applied in every phase of the innovation.

2.4.5 Design Thinking

Through academic investigation and research Plattner et al. (2010) have sought to understand why the design thinking method consistently leads to good results. According to them, the global truth lies in the fact that every physical product delivers a service and that neither of this matters if one does not have an insightful enterprise strategy. The methodology of design thinking, explained in detail by Plattner et al. (2010), is based on four rules:

1. **The human rule**: All design activity is ultimately social in nature - focus on the 'human-centric point of view'.

2. **The ambiguity rule**: Design thinkers must preserve ambiguity - innovation demands experimentation.

3. **The re-design rule**: 'All design is re-design', meaning that the human needs we seek to satisfy, have been with us for millennia. Learn from the past!

4. **The tangibility rule**: Making ideas understandable always facilitates communication.
Figure 7: The five phases of Design Thinking

Phases

**P1 - Empathize:** Customer interaction to understand their basic needs. Design thinking emphasizes the importance of knowing your customer, and caring about their lives.

**P2 - Define:** The right problem is defined, based on customer interactions. This will be the challenge to address.

**P3 - Ideate:** In this phase, the widest possible range of ideas are created, from which one can choose the best solution.

**P4 - Prototype:** Prototyping is done to stimulate the mind, communicate with stakeholders and fail quickly and cheaply.

**P5 - Test:** Feedback from the user might redefine prototype and solution.

By iterating through the process several times, the scope gets narrower and the product more focused.

**Characteristics:**

- Speed (rapid prototypes)
- Prototyping
- Working demos
- Customer interaction / understanding the user
- Iterations
- Experimentation
- Incremental
- Delayed solution articulation

(Brown, 2008) gives several examples of successful applications of design thinking. He demonstrates that the methodology can an aspect of human behaviour, and turn it into both a business value and a customer asset.
2.4.6 Extreme Programming (XP)
The introduction of the extreme programming method has been widely acknowledged as the historical starting point for the various Agile Software Development approaches (Abrahamsson et al., 2002). According to (Smith, 2007), it is perhaps the most widely discussed and most informative way of describing the Agile approach.

This literature review has also found other uses besides software. The company Johnson & Johnson have made an adaptation of XP principles for developing personal care products, such as lotions, creams, shower gels, and soaps (Smith, 2007). Wikispeed has also used XP principles as an inspiration to develop some physical parts of their racing car (Denning, 2012).

![Figure 8: Life cycle of Extreme Programming progress, copied from (Abrahamsson et al., 2002)](image)

**Phases**

**P1 - Exploration:** Features to be added in the software are decided. The team gets familiar with technology and tools needed for the project, and a prototype of the system is built. Duration of weeks to a few months.

**P2 - Planning:** The content of the first product and the order of how the features should be implemented are determined. Duration of first this product launch should not exceed two months. Duration of a planning phase is a couple of days.
P3 - Iterations: Several iterations where features are implemented in the order of customer's choice. After each iteration, functional tests created by the customer are run. After the iteration phase, the system should be ready for production. Duration of iterations are one to four weeks

P4 - Productionizing: Testing of system's performance. Also, new features can be added by choice of customer, or postponed to be implemented in the next phase.

P5 - Maintenance phase: From here the system must be developed and at the same time maintained with for example customer support. This requires a reorganization of resources and personnel.

P6 - Death: Customer no longer has any features to be implemented. The system must satisfy terms of performance and reliability.

Characteristics

- Short iterations and increments
- Open workspace - enabling coordination and communication between team members at all times.
- Pair programming - the code is written by two people on the same computer.
- Shared code ownership - anyone could change any part of the code at any time.
- Customer driven
- Daily builds

Johnson & Johnson used the following characteristics: small releases, simple design, test-driven design, pairing, collective code ownership, continuous integration, customer on the team (Smith, 2007). Wikispeed has used an adaptation of pair programming (Denning, 2012). This shows that, even though XP may not be completely transferable outside the software development world, the underlying values are transferable. (Smith, 2007)
2.4.7 Flexible product development

Flexible product development is described as the ability to make changes to the product in the development process or change the development process when it has already started. This can be done late in the development without being too disruptive. Changes late in the process are often more flexible, less disruptive and demonstrate higher flexibility. (Smith, 2007)

Central to flexible product development are iterations, where each iteration goes through the entire process and receives feedback from tests, customers or management before starting the next loop. This is because requirements will most definitely change unless they are specified at a high level. Due to the uncertainty of flexible projects, the development starts with short iterations which allow for adjustments of the direction taken and should be done in the customer environment. One or multiple iterations make up an increment, which is one release to a customer. A product can have several increments before it becomes mature. (Smith, 2007)

Flexible development has several categories of techniques to keep the cost of change low and to enable late decisions in the process. The techniques tend to be synergetic, and for a given project you would need an assortment of them. These methods include using modular architectures, experimentations, and iterations, set-based design and emergent processes.

Using modular architectures means assigning functional elements of a product to its physical chunks, and describing how these decoupled and independent chunks relate to each other. Modular architectures encapsulate and limit change and accommodate growth. Experimentation and iterations are used to sample the product or parts of the product, and revising them with customers frequently. Experimentations are exploration and show the product developer what works and not. The most common experiments are done through prototyping, testing, and simulation. The author emphasizes on front-loading experiments; getting valuable learning at the beginning of the project. Failure of an experiment should be seen as valuable learning, not something negative. Set-based design, in contrast with point-based design, means building and keeping a range of design options available and open throughout the development process. The design options are narrowed down continuously, and the better options are chosen late in the development. Emergent processes are used to allow flexibility, through letting new processes emerge during the project. Throughout the project, cycle processes are appreciated, eliminated or modified under given circumstances. Smith (2007)
Figure 9: The phases of flexible product development, illustrating increments for each phase

Phases

P1 - Sensing the market: Get in touch with the customer base, understand and learn.

P2 - Specification: Market definition and idea generation.

P3 - Design: Customer needs are met through frequent prototyping, as well as segmentation of the market.

P4 - Testing: Alternative technical solutions are tested on the marked with early prototypes, to establish a direction of project (Iansiti, 1998). Alternative designs are tested quickly and cheaply.

P5 - Integration: The customers' needs are implemented in the technical solution.

P6 - Stabilization: Product is launched and maintained. Monitoring and market response analysis.

Characteristics

- Iterations
- Customer relationship
- Rapid prototyping/testing
- Emergent process
- All team sizes
- Self-organized team
- Visualization
- Team collaboration
2.4.8 Feature Driven Development (FDD)
This methodology describes an adaptive approach for developing systems, focusing on design and building phases of software development and was first used in the development of a large banking application project in the late 1990’s. (Abrahamsson et al., 2002). The approach embodies iterations, quality throughout the process and frequent and understandable product deliveries.

*Figure 10: The five phases of FDD*

**Phases**

**P1 - Develop an Overall Model:** Documentation of specifications and a high-level description of the system are present. An overall model is created.

**P2 - Build a features list:** A proposition of what features to be added. These must be validated by customers.

**P3 - Plan by feature:** Features are prioritized. Each developer is dealt a specific code of which they are responsible.

**P4 & P5 - Design by feature & Build by feature:** Some of the features from P2 are chosen and multiple teams are created, developing different features simultaneously. This phase is iterative, with a time frame of a couple of days to two weeks. After a successful iteration the features are added to the main build, and the team starts phase 4 and 5 again with new features.

**Characteristics**

- Iterations
- Individual code ownership
- Small teams
- Self organizing
- Regular builds
- Incremental
- Working code in a timely manner
- Stakeholder-centric
2.4.9 Others
Ottosson (2004) dynamic product development fits the characteristics of Agile Product Development but is not included because of its focus on integration with traditional product development, which is beyond the scope of this paper. Innovative Lean Development, by Schipper and Swets (2012), is not included because of its reference to Lean development. Schipper and Swets (2012) also doesn’t mention the word Agile or flexible in a relevant context.

2.5 Agile Software Development and Agile Hardware Development

![Diagram]

*Figure 11: Overview of applied literature with product development as the umbrella term*

Literature links between Agile Software Development (ASD) and Agile Hardware Development (AHD) are sporadic, from what the authors have found. None of the articles we found link ASD and AHD. Smith (2007) talks about Agile Product Development and lends ideas from Agile Software Development but does not state that it can be used for hardware explicitly. Smith and Radeka (2009) mention that Agile can be used for NPD. Smith (2007) sees software as a unique medium that lends itself to Agile approaches and states that the Agile software approach cannot be directly translated into physical products. The specific literature on AHD is not found, and is therefore probably new or none-existing, considering the comprehensive search for literature.

Kettunen (2009) searched for the similarities between key concepts of Agile manufacturing and some of the most popular Agile software frameworks, to improve the software development process. The article draws links between Agile Manufacturing and Agile Software Development, and states that Agile manufacturing has had a longer tradition.
Drawing links between Agile Software Development and Agile Hardware Development, therefore, seems to have many advantages.

The benefit of hardware products is that they are visible both when prototyped and simulated. Electrical systems get the advantage of programmable components. By looking at the hardware-software product boundary, electronic technology is a good example of currently being somewhere in between. Electronic devices are becoming more and more programmable. Electronics can shift towards being more like software through microprocessor technology. Microprocessors are cheap, can be easily replaced and are programmable. (Smith, 2007, p. 83)

From an analytical standpoint, it is possible to compare ASD and AHD through the core Agile characteristics, specific to hardware and software. This is done through the mapping of the existing frameworks and the relevant literature which mention both ASD and Agile Product Development.

2.5.1 The technology R&D spectrum and the use of the Agile approach
From studying the product development processes of several startups in a range of industries, Marion et al. (2011) presents a schematic of a technology R&D spectrum. Shown in figure 12, the spectrum shows that companies can range from short-cycle rapid development (e.g., software), to traditional market-facing innovation (e.g., complex consumer products), to long-lead heavy R&D (e.g., biotechnology).

![Figure 12: The technology R&D Spectrum, copied from (Marion et al., 2011)](image)

Software companies can typically do quick prototyping with many iterations and be easily be tested with customers. Traditional development projects are typified by more upfront market research and traditional design methods. Long-lead R&D projects normally require intense
scientific investigation and a large amount of resources, both time and capital. The R&D spectrum also shows that software companies typically develop their products in a shorter time, and more often do prototyping and testing with the customer. The NPD commonalities have not been researched. (Marion et al., 2011). Ottosson (2004) also states that it is important for companies with short product life cycles to develop new products quickly.

2.5.2 Software and hardware applications to Agile Product Development

The following bullet points are listed from Smith (2007) and describe characteristics of software development that fits Agile methodology:

- Object-oriented development presents the possibility for modularization and enables substitutions of modules and isolating changes.
- Automatic testing of builds (iterations/increments) can be done cheaply, frequent and early in the process.
- The logic fundament of software code allows relatively fast automated checks of many types of errors.
- Relatively easy divisibility of product requirements into product features. Development of product features and dividing product features into specific development tasks.
- Relatively easy to find and involve customers in the development.
- No tools except computer and software are necessary.
- Low cost of iteration, as changes to software, through editing code, is easy.

The following statements from Schuh et al. (2016) show the challenges of using Agile with physical products:

- Hardware can usually not be infinitely partitioned into increments.
- There exist availability problems of tools and other production equipment.
- Hardware needs constant consideration of requirements which change from the feedback loops and results in high audit costs.
Table 1: Comparison of software and hardware characteristics

<table>
<thead>
<tr>
<th></th>
<th>Software</th>
<th>Hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency of testing</td>
<td>High</td>
<td>N/A</td>
</tr>
<tr>
<td>Cost of testing</td>
<td>Low</td>
<td>N/A</td>
</tr>
<tr>
<td>Cost of prototyping</td>
<td>Low</td>
<td>N/A</td>
</tr>
<tr>
<td>Cost of iteration</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Ease of tools and equipment</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Ease of increments</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Ease of stakeholder</td>
<td>High</td>
<td>N/A</td>
</tr>
<tr>
<td>involvement</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1 is based on information from the literature review and statements above. The blue boxes show the literature findings. This table can also be found together with the case findings in the conclusion.

2.5.2 Core Agile characteristics

Even though the Agile manifesto (and its principles) is supposed to give all characteristics of Agile, Meyer (2014) proposes a better list. He also includes the existing Agile frameworks which make up Agile. Meyer (2014) considers the Agile Manifesto and Agile Software Development methods and has rewritten the characteristics to be able to distinguish them better.

Table 2: Core characteristics of the Agile approach

<table>
<thead>
<tr>
<th>Agile Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Redefine the roles for developers, managers and customers.</td>
</tr>
<tr>
<td>2) No “Big Upfront” steps.</td>
</tr>
<tr>
<td>3) Iterative development.</td>
</tr>
<tr>
<td>4) Limited, negotiated functionality.</td>
</tr>
<tr>
<td>5) Focus on quality, understood as achieved through testing.</td>
</tr>
</tbody>
</table>
### Agile Principles

<table>
<thead>
<tr>
<th>Organizational</th>
<th>Technical</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Put the customer at the center.</td>
<td>6) Develop iteratively:</td>
</tr>
<tr>
<td>2) Let the team self-organize.</td>
<td>6.1) Produce frequent working iterations.</td>
</tr>
<tr>
<td>3) Work at a sustainable pace.</td>
<td>6.2) Freeze requirements during iterations.</td>
</tr>
<tr>
<td>4) Develop minimal software:</td>
<td>7) Treat tests as a key resource:</td>
</tr>
<tr>
<td>4.1) Produce minimal functionality.</td>
<td>7.1) Do not start any new development until all tests pass.</td>
</tr>
<tr>
<td>4.2) Produce only the product requested.</td>
<td>7.2) Test first.</td>
</tr>
<tr>
<td>4.3) Develop only code and tests.</td>
<td>8) Express requirements through scenarios.</td>
</tr>
<tr>
<td>5) Accept change.</td>
<td></td>
</tr>
</tbody>
</table>

### Agile practices

<table>
<thead>
<tr>
<th>Organizational</th>
<th>Technical</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Daily meeting.</td>
<td>6) Test-driven development.</td>
</tr>
<tr>
<td>2) Planning game, planning poker.</td>
<td>7) Refactoring.</td>
</tr>
<tr>
<td>3) Continuous integration.</td>
<td>8) Pair programming</td>
</tr>
<tr>
<td>4) Retrospective.</td>
<td>9) Simplest solution that can possibly work.</td>
</tr>
<tr>
<td>5) Shared code ownership</td>
<td>10) Coding standards</td>
</tr>
</tbody>
</table>

### Agile artifacts

<table>
<thead>
<tr>
<th>Virtual</th>
<th>Material</th>
<th>Agile roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Use case, user story.</td>
<td>3) Story card.</td>
<td>1) Team</td>
</tr>
<tr>
<td>2) Burndown chart.</td>
<td>4) Story board.</td>
<td>2) Product owner.</td>
</tr>
<tr>
<td></td>
<td>5) Open room.</td>
<td>3) Scrum Master.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4) Customer.</td>
</tr>
</tbody>
</table>

### Agile roles

2.5.3 Comparison of Agile frameworks

Table 3 (below) shows the relationship between the seven Agile frameworks considered in this thesis; Scrum, Lean Startup, Kanban, Design Thinking, Extreme Programming (XP), Flexible Product Development (FPD) and Feature Driven Development (FDD)
Table 3: Table with comparison of Agile frameworks. The asterisks, *, means 'to some degree'

<table>
<thead>
<tr>
<th></th>
<th>Scrum</th>
<th>Lean Startup</th>
<th>Kanban</th>
<th>Design Thinking</th>
<th>XP</th>
<th>FPD</th>
<th>FDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software</td>
<td>X</td>
<td>X</td>
<td></td>
<td>*</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physical</td>
<td>X</td>
<td>*</td>
<td>X</td>
<td>X</td>
<td>*</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Iterations</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Incremental</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Visualization</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rapid prototyping</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Stakeholder interaction</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Focus on team, collaboration and</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>communication</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Self-organizing</td>
<td>X</td>
<td>*</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Freeze requirements</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Limit WIP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adaptation of plans</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

From table 3, it is clear that all of them follow the Agile principles of iterations and increments, rapid prototyping, close interaction with stakeholders, team focus, collaboration and communication, and lastly on the principle of adaption during the process. This goes for all except Kanban, which does not include rapid prototyping. The reason is that Kanban - as a supplement to other frameworks - describes what work needs to be done, not necessary explicit to prototyping. (Meyer, 2014)
The authors have chosen to include the specific characteristics of 'freeze requirements' and 'limit WIP' of respectively Scrum and Kanban, to indicate that there is some difference between them.

Even though 'freeze limitations' are listed as one of the Agile traditional principles, only Scrum mentions this explicitly. The same goes for self-organizing (an Agile organizational principle), which is not explicitly stated in Design Thinking. Further, the limitation of work-in-progress is only mentioned in Kanban.

In the top two rows, it is shown that only Scrum and Design Thinking are utilized actively in both physical product development and software development. Lean Startup and Extreme Programming are mainly made for - and utilized within - software development, but are to some extent used in the development of physical products as well. For Kanban and Flexible Product Development it is the opposite, where the methodologies mainly are used within physical products, but to some extent within the software. Feature Driven Development are on the other hand only used for software development. In general, all the Agile frameworks considered follow the core characteristics of Agile methodology - shown in table 3 - with only minor exceptions.

2.6 Other Agile terms, Agile movement and rhetorical techniques

As the reader probably has noticed, Agile is not mentioned explicitly in all relevant literature. Agile approaches are often not called Agile, even though they are by values and principles. Smith (2007) and Meyer (2014) argue that the term agility should be avoided since the opposite of Agile would be rigid, and no one wants to admit to being rigid. Smith (2007) uses flexible instead of the term Agile, unless he references software development. This is because he wishes to rebuild what Agile practitioners and academics (agilist) have done, and not merely translate it from software. Boehm and Turner (2003) uses the term Agile, but uses discipline and plan-driven as the opposite. Agilist object the term discipline, since Agile needs discipline in an unconventional way. Agilist objects the term plan-driven since these Agile needs planning just as much "plan-driven" approaches. A good middle ground would be traditional as the opposite of Agile (Smith, 2007).

Also worth mentioning is that Agile is part of a movement, started by the authors of the Agile Manifesto. The authors formed the Agile Alliance, a non-profit organization that promotes software development according to the manifesto. After considering a variety of sources, there seems to be criticism towards traditional product development when Agile is mentioned.
Meyer (2014) even goes as far as calling the Agile movement a way of misleading the readers of Agile articles. Meyer (2014) has several examples of rhetorical techniques which are being used in articles about Agile and invites the reader to exert caution.

There seems to be a lot of confusion and misunderstanding around Agile. From what the authors have found, it is unclear to many if Agile and lean are the same, and what definitions exist for Agile Product Development and lean product development. As stated in the section about Agile history, a lot of literature on lean development could be called Agile, with (Schipper and Swets, 2012) as an example. It may seem like this book is talking about Agile even though the book is called ‘Innovative Lean Development: How to Create, Implement and Maintain a Learning Culture Using Fast Learning Cycles’. From the title of the book alone, learning cycles can be spotted. This hint to the Agile characteristics of iteration-driven, stakeholder involvement and validation of ideas. A reader interested in innovative Agile Product Development should also consider this book.

2.7 Summary and discussion of theoretical findings

The literature review has discovered several important findings regarding the Agile approach for physical product development; (1) Agile frameworks seemed to be used to a large degree in software development, and newer literature shows the use of several Agile frameworks for hardware development, and (2) The core characteristics of the Agile approach seem applicable to hardware development

Still, the concept's broad range of topics and its non-consistent definitions has made it difficult to know if all relevant literature is obtained. There seems to be no articles that directly links Agile Software Development to Agile Hardware Development. Even though research has been conducted on Agile Product Development, the results are inconsistent. Several examples are given about the benefits of the field, but there exists a consensus that more research is needed on the field. There also exists a disagreement about the integration of Agile Product Development in traditional product development. Some authors state the importance of combining these, while other glorify the Agile approach as an individual approach. The application of the Agile approach to physical products seems to be disputed.

The six Agile frameworks presented can be confused with each other, and there seems to be only minor differences considering the characteristics. Even though the Agile Manifesto was intended for software development, some of the frameworks fit hardware as well. Some of the frameworks presented even started as a hardware framework, which makes the authors
believe they are still relevant for this purpose, even though they are not widely used today. Considering literature, the following Agile frameworks can be applied to physical product development: Scrum, Kanban, Design Thinking, Flexible Product Development and The Lean Startup. Extreme Programming can be used to some extent.

The authors conclude that one of the reasons why the existing Agile approach doesn’t directly apply to hardware is because of the fundamentally different conditions for software and hardware development. Regardless of conditions the core of the Agile approach seems to be; iterative and incremental development with customer interaction, self-organizing teams, emergent processes and using tests as a key resource.
3. Method

In the following chapter, the method used for the master thesis are presented. A case study research has been chosen to answer the research questions. Firstly, a case study design was created based on the aim of the thesis and the research questions. Then data acquisition was conducted through eight interviews of the four hardware startups, together with a thorough due diligence. Two interviews were conducted for each hardware startup in each of the four subcategories Robotics, Wearables, Connected Devices and Designed Products, as proposed by DiResta (et al., 2015).

3.1 Research design

In this subchapter, the research design will be elaborated. It includes an explanation of the chosen qualitative study, followed by a subchapter regarding epistemology, ontology and methodology. Finally, an explanation of the case studies will be described.

3.1.1 Qualitative research

The aim of the master thesis is to investigate how the Agile approach can be used by startups in the development of physical products. A qualitative research is chosen, as its creative nature is great for understanding the world from the perspective of those studied (Pratt, 2009). Qualitative data often provides good understanding of the dynamics of the relationship, which is crucial to the establishment of internal validity (Eisenhardt, 1989).

3.1.2 Case study

A case study is an in depth study within single settings focusing on understanding the dynamics present (Eisenhardt, 1989), and can involve either single or multiple cases, typically with combined data collection methods (Yin, 1981).

To get a better insight in possible differences within hardware startups, the authors have chosen to divide these into the four subcategories proposed by DiResta et al. (2015). These are Robotics, Wearables, Connected Devices and Designed Products. Consequently, the researchers have chosen to interview one startup from each of these subcategories.

This number is due to Eisenhardt's (1989) argument of four to ten subjects as a proper size, and the fact that (Dalland, 2012) recommend to start small and rather consider expanding later if more data is needed. It is of interest to use a homogeneous sample to ensure internal validity and add construct validity since the likelihood of distortion from extraneous variables is reduced (Calder et al., 1982).
The cases interviewed had to fulfil some criteria. Firstly, they had to develop a physical product, and be regarded as startups. Startups are chosen since they operate with high levels of project uncertainty. Because of this, there is a higher possibility that a project needs to change at some point in product development (MacCormack and Verganti, 2003), and since change is what Agile is all about, startups seems like a better candidate then established firms. Due to the general and broad definition of a startup as "a human institution designed to deliver a new product or service under conditions of extreme uncertainty!" (Ries, 2011), the authors chose to narrow this down by investigating startup companies developing their first product.

This usually means early phase and few resources available. The startup should not be more than 5 years old. The startup chosen for the designed products category was in their fifth year, but accepted due to the difficulty the authors had to find similar startups within a short amount of time. Further, the focus was on Norwegian companies. The reason for this was availability. The researchers wanted to conduct the interviews in person, to get a more dynamic interview. Finally, the startup should be interacting with an established - or potential - customer.

The criterions in selection of cases, summarized:

1. Must be a startup, no more than 5 years old.
2. Developing their first product,
3. Developing a physical product
4. One company from each of the four hardware subcategories
5. Norwegian based company
6. Interacting with an established - or potential - customer.

The design of the study will be as a multiple-case study (Yin, 2013), with a cross case analysis.

**Selection of interview objects**

It is of importance to have criteria for the choice of interview objects, to extract as much relevant information as possible regarding the research questions. The scope of the thesis is to investigate Agile Product Development methodology, thus at least one of the subjects had to be a product developer having personal experiences with the methodology in use in the respective company. The product developer should have been involved in the company through its whole lifetime, to be able to reflect more deeply on possible changes of
methodology and their sense of results. One should also have an interview object from the top management of the company, to verify results regarding the company's main goal and strategy. According to Dalland (2012), the aim should be to find someone in each case study that has been through the whole process. The criterions in selection of interview objects, summarized: (1) Involvement throughout the whole period for at least one of the interview objects, (2) One involved in product development and one with total overview of the company. The two candidates could for example be chief technology officer (CTO) or product developer and chief executive officer (CEO) of the company or a business developer.

3.2 Data acquisition

Data was collected through interviews and background information of the companies - as size, industry, customers and income. According to Eisenhardt (1989) the overall idea should be to become intimately familiar with each case as a stand-alone entity. After clarifying which hardware startups to interview, and getting their approval, the authors started the data acquisition.

The researchers chose two objects from each hardware startup to interview and conducted the data acquisition as a team within a short period of time. Immediately after each interview, the authors made individual reports, discussed contradictions and wrote the case studies. In each interview, one of the authors would handle the interview questions - functioning as the personal interactor - and the other one would take a more distant view, recording notes and observations (Eisenhardt, 1989). This ensured a higher validity and allowed the case to be viewed from different perspectives. The interviews were conducted in a semi-structured way, allowing for dialogue and possibilities of discovering data not covered by the interview questions.

Throughout the interview, the researchers would ask themselves questions like "what am I learning” and "how does this differ from the last case?”, proposed by Eisenhardt (1989) to enhance the learning outcome. The interviews were recorded and transcribed later, to allow for full focus on the interviewee. Before the interview participants was informed about the use of the obtained information, and confidentiality. All data acquired was collected and systemized to obtain a chain of evidence (Yin, 1998).
3.2.2 Interview

From each case, the authors interviewed a business developer and a product developer. Before each new case, a due diligence was performed, by gathering business plans, virtual progress boards (e.g. Trello), Gantt-diagrams, meeting documents, previous prototypes and similar. In total, eight interviews were conducted with two participants from each of the four hardware subcategories. Each interview lasted between 60 and 90 minutes, and was conducted at the premises of the hardware startup, with both researchers present. The interviews was conducted in a semi-structured way. According to Bryman (2008), this is a good method for understanding subjects' perspectives. It was believed that two subjects from each company should be sufficient to capture enough information, both about what is 'actually' happening and the experience of the people directly involved in startups methodology.

The researchers prepared an interview guide related to the research questions. The interview questions were based on the summarized version of the Agile Manifesto, as proposed by Meyer (2014), but in the semi-structured interview these could vary depending on the situation. The aim of the interviews was to look for certain similarities or dissimilarities between the cases related to the characteristics presented in the Agile Manifesto, as this was made specifically for companies developing software products. Additional questions regarding Agile frameworks for physical product development were also added. Covering a broad amount of questions and topics, the focus narrowed to the most interesting findings during the thesis.

Only one interview was conducted per subject, so several test interviews were done on test subjects to ensure that the 'rough' structure of questions was suitable. The main objective of the interviews was to record the subject’s own experience of the product development methodology utilized by the case company, and how it affected them. To enhance the effectivity and validity of data analysis the interviews themselves was divided into several parts with specific themes.

3.3 Analysis of data

After collecting the data, the researchers tried to ensure that the reliability of their measures was satisfactory (Mullen et al., 2009) by doing a case analysis separately on each case immediately after each interview when the experience of these were fresh in mind. Each author wrote down their perception of the interviewee in terms of how their startup utilized
different characteristics from the Agile Manifesto. In addition to this, suggestions for findings were written down and collected in a shared document. Then, the authors compared notes and discussed what aspects seemed most interesting. This would influence the interview guide in terms of new questions or themes. When both interviews of one case was performed, the authors merged their separate analysis into bulletpoints for each of the Agile characteristics, which would form the basis for the case studies (chapter 4). The analysis itself was based on the theoretical framework proposed in chapter 2, with a focus on the research questions. To substantiate the importance of 'trackability and explicity' (Haldorsson & Aastrup, 2003), the data was systemized and structured. For analyzing the researchers kept in mind Dalland's (2012) encouragement to alternate between interpreting and analyzing the given answers from interviews.

A cross-case analysis was conducted and is presented in Findings and Analysis (Chapter 5). Authors tried to find variables within each case that contributed to the effects of what they are searching for, thus looking for similarities and dissimilarities between the groups. The researchers looked for patterns to confront the theoretical framework with empirical information. Comparison was done with both conflicting and similar literature to build internal validity, improve construct definition, sharpen generalizability and raise the theoretical level (Eisenhardt, 1989).

3.4 Reflections of method

There are several limitations of using specific methods to answer research questions. In the following we will consider several challenges of using the chosen research design, discuss the quality of the study and the general limitations of the study.

**Challenge 1:** When choosing a qualitative method, the researchers must know that there are both advantages and disadvantages with the method. By using a qualitative method one could dig deeper into fewer cases, and get a better understanding of the perspectives of those studied (Pratt, 2009), opposed to the quantitative method. Even though the method is likely to be more testable and generate a novel theory, it is difficult to know what the most important relationships in the findings are (Eisenhardt, 1989). Through a qualitative method, the results will also be coloured by the perception of the subject as well as the perception of the researcher.
**Solution 1:** By performing the interviews as a team, doing solid notes on the company, interviewee, situation, and other relevant data, have retrospect discussions and being aware of the 'problem' of perceptions, it would give the research a higher trustworthiness.

**Challenge 2:** The subject might not be completely honest, like giving answers they think the researchers want to hear (Yin, 2014).

**Solution 2:** Building trust with the subject through regular contact before interview, and enlighten them on the purpose of the thesis as well as confidentiality was important for the researchers. If desired, subjects could require that data was deleted after the research, and kept safe during data analysis, as proposed by (Thagaard, 2003). This did not happen in any of the cases. The researchers encouraged subjects to be completely honest for the sake of the thesis. A case study can be affected by external explanatory factors, thus it is important to relate the conducted study to its results.

**Challenge 3:** The questions itself might have been misleading to the actual purpose of the thesis, without the researcher's knowledge. Poorly articulated questions can give 'wrong' answers.

**Solution 3:** By performing the interviews in a semi-structured way, the researchers opened for a dynamic interview. The questions could change during the conversation, if for example more interesting topics was brought up. The problem with this was that the conversation deviates from the original path in such a way that the information gathered could be of less use than in other, more structured interviews. As each interview only was conducted once, and that time was of the issue, it was important to be 'on point' and redirect the conversation when needed.

**Challenge 4:** To strengthen internal validity and reduce likelihood that certain variables distorted the results (Calder et al., 1982), the researchers chose a homogeneous sample. Even though each startup was from a different subcategory, after DiResta et al. (2015) definition, the researchers tried their best to choose as homogeneous samples as possible within these groups. The problem of this could be the lack of generalization, or external validity. This choice also built on the choice of a qualitative research with a low number of case studies.
Solution 4: It is possible for the researchers to do a quantitative research on top of the qualitative, which would give a broader understanding of the topic. Unfortunately, time is a limiting factor which would make it hard to do both case study types in the same thesis. Another possibility for increasing the external validity was to increase the sample size.

Challenge 5: The choice of startups and interview subjects was based on criteria chosen by the researchers. These are based on their own perception of what criteria would fit best to answer the research questions. This also applies to the criteria of the cross-case analysis.

Solution 5: By doing a thorough research on relevant literature and defining the theoretical framework in depth, researchers could choose the relevant criteria with higher confidence. The choice of theoretical framework was dependent on the amount of relevant literature obtained, and influenced the whole research. A good foundation was therefore important.

Challenge 6: Literature gathered, even though seeming accurate, might in fact be misleading or wrong. This could be an effect of sloppy work done by previous researchers or poor articulation of our own research questions.

Solution 6: It is difficult for the researchers to decide whether literature can be trusted or not. In general, research is often conflicting, so comparing the large amount of literature to each other and discarding what is thought to be misleading might be a mistake in itself. The best way to avoid errors in literature is to gather as much information as possible, and be critical under the analysis.

One of the greatest concerns for the researchers was that of construct, internal and external validity. Construct validity is the degree to which a test measures what it claims to be measuring. Internal validity is the extent to which one could say "no other variable than the one we are studying caused the results", that this and only this made that happen. External validity explains how well data and theories from one setting apply to another, in other words how generalizable the findings are.

According to (Cook et al., 1979) one can build construct validity by test for convergence between same 'things' and divergence between related, but different 'things'. This can be done in the data analysis, but one should bare in mind the possibility of choosing the wrong variables to test for is which will affect the results and general conclusion. The researchers have enhanced the internal validity by choosing a homogeneous sample, divided interviews
into themes (organizational and product development) for easier analysis and searched evidence for ‘why’ behind relationships, as proposed by Eisenhardt (1989). The case of external validity is given less priority in the thesis, due to the choice of qualitative research, but can be sharpened by comparing with both conflicting and similar literature (Eisenhardt, 1989).

One of the most limiting factors was that of time. As this study only occurred over a short period of some months, there were no time to go into significant depth which influenced the conclusion.

The amount of relevant literature found on Agile development for physical product development was less than expected which provided a weaker foundation for theoretical framework. This might have something to do with the search engines in use which might not cover all literature, or wrong selection of search terms. Even though this made the research more extensive it also enabled for discoveries of new theory.
4. Case studies

In the following chapter, four case studies will be presented, one for each of the four subcategories - Robotics, Wearables, Connected Devices and Designed Products - as proposed by The Hardware Startup (DiResta et al., 2015). All cases will be anonymous, and only mentioned by their subcategory name. An interview guide was used during the interviews, which covered a broad spectre of questions related to Agile characteristics. This can be found in the Appendix chapter A.

The structure of each case study is made to give a coherent text regarding the respective startups methodology; the first part is related to information about the company, and the second part is related to product development.

The following criteria was used:

(1) Startup, no more than 5 years old
(2) Developing their first product
(3) Developing a physical product
(4) One company from each of the four hardware subcategories
(5) Norwegian based company
(6) Interacting with an established - or potential - customer.

4.1 Case 1 - Robotics

4.1.1 Description of the company

Robotics is a hardware startup established early in 2016, originating from NTNU. They are developing an affordable and easy to set up cable cam, enabling filming in challenging areas where camera drones or likewise gadgets are impractical. The startup consists of two business developers and three product developers and falls in the category of hardware subcategory of Robotics, due to its use of electric motors and automated control system.

**Interview object 1:** Product developer. Master degree in Product Development at NTNU.

**Interview object 2:** CTO. Master degree in Entrepreneurship with a background from Product Development at NTNU.

**Frameworks claimed to be used**

According to the CTO, Robotics use the agile framework Scrum which is something they have learned through courses at NTNU. They have implemented several aspects of this including Scrum Master and Product Owner, though these are not utilized completely after
Scrum-methods, in addition to sprints, sprint log and product backlog. The latter is a list of all possible features for the product. Before each iteration, several features are chosen from the product backlog to be implemented, and added to the sprint log. The product backlog and sprint log is visualized through the online tool, Trello. The product developer points out that the methodology they are using allows them to react quickly to any change.

Roles
CTO has the main responsibility for the product development, everything from product development to production. “I have the administrative responsibility and the high-level managing of the product development”, he says. The CTO does all the sourcing of components and equipment, and makes sure that everyone sticks to the rules and process when working with product development. The lead designer does almost all the graphical work, is responsible for CAD-drawings and other 3D modelling. Everything that has to do with industrial design. The last product developer has the responsibility to find smart solutions for the proposed prototype.

The team uses Scrum but does not use the names for the roles that the Scrum framework proposes. Decisions are made in different hierarchies depending on what needs to be decided. The high level decisions, like decisions which make a big impact on the product, are always made at the “all-meeting”, where all team members from the startups is present. Here information from all team members are shared, including financials from the CEO and user-feedback from the CMO. “So you can say that we together make product owner decisions in that meeting. We have had bad experience with decisions made by one person”, the CTO adds. But on the other hand, the CTO tells us; “when speed is needed in the product development, letting all people in on decisions, can consume too much time, even though small changes in the prototype can drastically change the final product.

For distribution of tasks, the team divides product development tasks between them every week after what needs to be done, even though this goes beyond their set role and responsibility.

Meetings
Robotics practice several different types of meetings. The “all-meeting” is every Tuesday or Wednesday, depending on circumstances. Here, everyone in the startup meets to discuss and take the big decisions regarding product development and business in general. Every Sunday they have a “Product planning meeting” for the product developers. This is a meeting where
they plan activities and tasks a week ahead. In addition, they practice something called “work day” - a day where everyone works together with product development, like a workshop. “This means working and making smaller development decisions with the technical part of the team” the CTO explains. The CTO also reflects on the decision-making process, talking about the balance of making decision with the product development team alone or in the “all-meeting” or with the lead-users. “Not all decisions can be done with the lead-users or the whole team, because it take to long. It’s important to considers if the given decisions affect a lot in the final product.”. The startup has a round with personal feedback two times per year.

**Working environment**
The startup has their office at the premises of NTNU, and utilize a workshop offered through Spark NTNU. All the team members usually works there, but mostly the two business developers and the CTO. The product developer has noticed that the technical team is able to get much more work done if all the team members are located at the same office. Similar, he notice that productivity drops immediately if they do remote working. All team members work long days, usually 50-60 hours each week.

**Company ethics**
According to the CTO, it is very important for them to make a decision through consensus. They also believe in the idea of being able to react quickly to changes. In addition, all prototypes or subversions should be as simple as possible. This is important, as it will reduce the time between each new prototype, and then enable Robotics to fail fast. "Failing is inevitable", the product developer says, "the question is how fast you can make it happen".

**Customer**
Robotics' customers are outdoor enthusiasts, performing for example extreme sports, that want to film themselves in nature. This could be a snowboarder doing a jump, or an off-road cyclist. The startup does not have paying customers, but use their test-users and ambassadors for feedback. The test-users test features and ambassadors promote the product to get free prototypes. The product developer also underlines the fact that, being outdoor enthusiasts, they are themselves potential users of the product they are making.

4.1.2 Product development
The CTO states that they have made 28 prototypes. He further explains that each prototype has a sub-version. For example 28.A, 28.B, and so on, where 28.A differs from prototype 28.B in the way they function. The main concept, and thus the customer segment, has been
the same from early stages. This means no extreme changes has occurred, which has enabled a constant focus with several rapid changing prototypes.

**Requirements**

The startups get most of its requirements from their own use of the product. They also receive feedback from test-users, but not all feedback is relevant. Some feedback is noted as possible ways the product could be, but mostly it is important for the startup to have focus on the requirements they think are important. “We know in the abstract what the product is going to be like, so the planning and execution is for the details, the add-on's, and each sub-part of the total solution. The idea doesn’t change and therefore some high-level planning is possible“, the CTO explains.

The communication with test-user and ambassadors is done through email. The CMO presents their opinions in the “all-meeting”, and the whole team discuss these opinions as if they had one vote in the discussion. “The opinions are often biased and subjective, based on what the test-users normally uses the product for, and it's therefore hard to do evaluate the opinions systematically”, the product developer explains. The startup has used their own product for a long time, and tried to emphasize with the customer - to try to understand what they think. The CTO explains that he “tries to think like the customer.”

**Planning**

When developing a new prototype, the development team typically uses the following process:

1) State the purpose for building the prototype, 2) Find out what needs to be built, from a drawing or requirement, 3) Find out the different ways that prototype can be built and choose one, 4) Consider that it also needs to be tested, 5) Build and 6) Test.

When making the product the startup does iterations on parts of the prototype, because a long time can go before the whole prototypes gets an iterations. At the beginning of the week they plan. Each iterations often involves some drawing, making CAD drawings, printing 3D models and testing them. The CTO tells us: “For each week-iteration everything is planned to great detail. We know what we want to do and how the solution can be best built. If you are told that the solution is supposed to be a certain way, you are supposed to deliver a proposal for that thought out solution.”

The company limits the possibility for change since they have experienced that if this is not done, the rest of the team can disagree with the proposed solution and there work has to be re-
done. The CTO wants to freeze the requirements for each week - allowing no changes to the plan - but thinks that is too hard. “Things change and deadlines are often not met. That’s the way product developments is. Things always last longer than you might expect, and it’s therefore important to accept this. There is always a high degree of uncertainty when planning our deadlines.” the CTO explains. Robotics recognizes the need for better planning or execution, and that specific experience in the right field is needed. "A challenge," the CTO explains, “is the limited experience the product development team has. We do not know all the different techniques available and therefore only prototype with the techniques we know”. The product developer elaborates that it is very realistic that execution can take 100 times longer than the planning expected.

**Building**

The CTO explains that it’s important that the development team works together at the same place and location, to be able to make as many decisions as they can in the least amount of time. “We have no time for surprises the next day. When we are working together we can asks questions at the moment they occur and therefore make most product development decisions together. Micro decisions in the product development can change the final design to a large degree.”

Standardization of components is not an issue for Robotics. “We only have few components and therefore do not standardize. We also don’t use standard solutions because everything needs to be customized.”, CTO says. Sometimes they make something radically different and implement it if it is better. "One time we made shorter arms and ended up using it.". 3D-printing is utilized for almost every component of the prototype, but even though 3D-printing is a new ‘rapid’ method the product developer points out that it takes time. Even a small component can take 10 hours to 3D-print. "It is especially time consuming if something goes wrong with the printer", he says. "That happens all the time".

**Review**

Testing is done continuously, either at the workshop or with test-users. The CTO explains that it is hard to make things with the intent to test them afterwards. Often, they need to talk with experts or find people with experience. “For example, if you need to perfectly click fit two plastic parts together. It’s possible to 3D print it, but impossible to know how they should be made for plastic molding. It’s not possible to know what the parameters should be. So, designing for manufacturing is not always possible.”. Also, testing the product in it's right environment is not straight forward as their product requires alot of space to be used. Their
rapid development of prototypes and sub-versions results in their frequent testing. Still, CTO says, they have never defined a prototype to be done; "there are always improvements to be made". Sometimes, they make prototypes for long term goals like getting to production, a conference, or something similar. They did for example make a demonstrator prototype for the ISPO Conference where they were going to show potential collaboration partners how it the product works. When needed, Robotics involve suppliers or experts on the given field, to make sense of the testing results.

4.2 Case 2 - Wearables

4.2.1 Description of the company
Wearables are developing personal sensor devices which gathers data about the human condition, and was established in the middle of 2016. The product can take the form of a device worn by the user and a mobile application or dashboard which presents the data. The startup is currently working on a sensor which measures hydration of elderly, and consists of three business developers, and two product developers. The startup falls in the subcategory of wearables, as the product "gathers data related specifically to a human subject, then process and display it in a way that makes it easily understandable to human end users" (DiResta et al., 2015)

**Interview object 1:** CEO. Master degree in Entrepreneurship, with background from Mechanical Engineering and Business Administration.

**Interview object 2:** Hardware engineer. Master degree in Electronic Systems Design at NTNU.

**Frameworks claimed to be used**
According to the CEO, Wearables use Kanban for both product development and business development. This is achieved through an online tool called Taiga, which the CEO claims is a combination of both Kanban and Scrum. They utilize the Scrum roles of Product Owner and Scrum master, and use Gant charts, which is a diagram that illustrates a project schedule.

**Roles**
Except from the CEO, and two other business developers, Wearables consists of two product developers - a software developer, which is the CTO, and a hardware engineer, both responsible for their respective fields. According to himself, the CEO currently has the role as the product owner (Scrum), but says this role later will be transferred to the CTO. “The
technical team is self-driven. They work from functional requirements and decide themselves how they want to solve it”, CEO says.

The CEO explains that his most important job as the product owner is to limit the items in the backlog by finding out what relates directly to the core requirements. The core requirements are related to what is prioritized at a given time, the available resources and what is able to make the company money at the end. “You cannot be sure if your product has the right functionality until the customer puts money on the table, or in any way commits financially. That’s the only you will know for sure when you are making something the customer wants”, the CEO points out.

The two business developers only get updates from what the product developers are working with, but can affect the product development by updating the requirements-document. The CEO explains that “Everybody who has been in contact with the customer, needs to update the document where we list the customer requirements.”

The new CTO makes most of the important product development decisions. The CEO can propose a solution or a new requirement for the product and the CTO can then tell the CEO if this is realistic to do considering the time and resources available. The CTO’s experience is the reason his decisions often are chosen.

Meetings
The CEO claims they are not having any meetings anymore. “Meetings are extremely wasteful and we try to let our process do this work. We are focused on communicating goals and activities through documentation after each sprint.” He adds that this works well because the CTO communicates very well. “The CTO is structured in the way he speaks and categorizes everything to make it understandable.”. The meetings are replaced with documentation. Documentation consists of making your goals, process and previous work as clear as possible to the others. According to the CEO this makes it easier to on-board new team members. The CEO also admits that a previous, very skillfully, employee was unstructured, had no plan for his work and was bad at communicating. This has partly made the CEO believe that their new framework is important, and that the next people they employ need to have a good work structure. “Skills are important, but it’s more important to be hungry for knowledge and improvement.”
Working environment
The startup works in a shared office, and the CTO works remote. “The biggest risk of working remote is that people don’t have the same perception of the goals. Remote working works well for us, because the CTO is good to communicate. We would not let everyone work remote.”

The hardware engineer states that they do not practice feedback sessions with all team members involved. The CEO likes to talk about the things that bother him immediately, and sees no reason to involve everyone in this. “We have a culture of saying things straight out, and not beat around the bush.”

Company ethics
The CEO explains that they don’t have strict rules, but value the following practices: 1) Multitasking is not allowed. Stop it at once, and 2) You should not interfere with other people's work. Trust that they will do the job right as long as it is done according to the requirements-document.

The hardware engineer believes it is of great importance that people have specific responsibilities and that everyone trust each other to deliver results based on this. “I don’t believe we have any rules, but we have a certain way of working and we commit to this to let the team-machine work in the same way”, he says.

Customer
Wearables’ customer is the Norwegian Government. These are responsible for the health and well being of elderly in Norway, and are looking for solutions that can help out with the problems related to dehydration. Contact with the customer is regularly through workshops about every third week, and through mail. Wearables do not have a close contact with the end-user.

4.2.2 Product development
According to the CEO, Wearables have made more than 100 prototypes over the lifespan of the startup, where the hardware engineer claims this number is 5 or 6. In the very beginning, Wearables were going to develop an hydration sensor for athletes, but quickly changed to dehydration sensors for elderly after discovering a huge demand for this type of solution. They have evaluated several types of hardware products to do sensing and visualization of data, like a watch, but is now developing a medical patch to be placed on the chest. The prototypes are being developed in-house by the two product developers.
**Requirements**

According to the hardware engineer, they get their requirements directly from the customer. He explains that they try to let the customer requirements go to tasks as fast as possible, in case this affects the current list of requirements. “We go from an Epic - which is a demonstration of functionality, or a milestone - to user stories below this Epic, and the user stories become prioritized tasks.”, the CEO explains. The Epics and user stories are mostly used in the same way every time, and they try to make it into a standard they can use. That way, as little time as possible is used before they can do a task. “We try to be 100% self-organized through the process we have set up, so I don’t have to interfere with how someone does their job. The most important thing I do is to make sure everyone is working towards the same goal and the same understanding of this goal”, the CEO explains.

**Planning**

The CEO’s perception is that it is important start with planning, executing this plan and evaluating this plan afterwards. During an iteration, they find it important to limit changes. “If you change everything while you are working you will never complete anything”, he says. The CEO explains that they have development cycles lasting approximately 1 month, and that these cycles are more high-level. “Each R&D cycle would need to satisfy requirement X, Y and Z, and each of these high level requirements are also put into shorter cycles. The X cycle can last 12 days for example.” Iterations stem from the customer requirements. The Epics become user stories and the user stories become prioritized tasks which are executed on in the development cycles (iterations). The hardware engineer highlights that “Sometimes the technical team needs 2-3 iterations to work on some functionality, because it is technically challenging.”. The hardware engineer has experienced that it is important that you know what tasks will be completed for next Friday, but also sees the problems with planning. “In the beginning you will over- or underestimate how much time a task will take. The next time you will probably improve on your guessing.”

The CEO claims “We don’t set time limits, because Agile states that you are not supposed to do so”. The hardware engineer also exemplifies this practice by stating that an iteration is not finished until the prototype is finished, even though they try to estimate the time they will use on a completing a prototype. The CEO tells us that they have to use time limits because if not, the customer won’t receive their product in time and the company goes bankrupt.
Building

The development is done individually, as both product developers are responsible for their own work. The hardware engineer usually works at the office, while the CTO works from home. Due to the CTOs good communication, this is not a problem. Joint development is almost not done, and only by mistake, the hardware engineer states. It is important for the startup that each team member has their own respective task to work with. The hardware engineer points out that “on the other hand, when an unexpected deadline is near, the whole team can help finishing a task.”

Review

According to the CEO the most important thing after a sprint is that the results are good, and not that the team has learned a lot. The CEO says that “A fundamental thing for me is that the team does not need to be good at everything. You have to trust that your team members do their work well.” After something has been built, it is documented, and this needs to be be very clear and concise, according to the CEO.

The CEO points out that “We make prototypes for the purpose of testing something you otherwise wouldn’t be able to test. It doesn’t make sense to just make something. It’s important to always have a purpose for each prototype.”. At the same time, the hardware engineer states that the first iterations can be exploratory and that the purpose of prototyping in the beginning are very broad. “Sometimes it’s hard to test if you have made a prototype which has served the set purpose, because the exploratory prototyping itself is the purpose. The learning from these prototypes can make the next iteration more specific regarding the purpose.”

4.3 Case 3 - Connected devices

4.3.1 Description of the company

Connected Devices is a spin-off from NTNU, offering monitoring of propane cylinders for gas suppliers. Using GSM-technology, gas suppliers can be notified when the propane cylinders run out of gas and save resources associated with logistics. The company was founded in January 2016, and is made up of 7 part-time members, some of them engineer students from NTNU. The startup falls in the category of connected devices due to their GSM-module which streams information from the product to a device of the customer's choise, like a computer or a mobile phone.
**Interview object 1:** Product developer Took a bachelor in Bergen, and is now studying a master in Product Development at NTNU.

**Interview object 2:** CEO and founder. Studied bachelor in IT, then master in Marine Technology.

**Frameworks claimed to be used**

Connected Devices is trying to implement some Design Thinking here and there, based on a course with the same name. This means that they have a customer focus, and try to develop their product after the customer’s need instead of pushing a product on them. As the product developer says “It is about empathizing with the customer, to get insight in their problems”. According to the product developer, this is something they could have a larger focus on; have more dialogue with the customer. Still, he emphasizes that their product is customized to their customer, not a general need. According to CEO, after they acquired a customer they have also implemented more focused schedules and deadlines.

**Roles**

According to the product developer, the team is very multi disciplinary. The CEO is in overall control and is responsible for market clarification. He also facilitates for the rest of the team, purchase supplies and obtains the expertise needed through networking. As he says; “I have to keep an overview, and try to look into the future. It is my responsibility to make sure of the business’ progress”. Other than that, the team consists of a CTO, a business developer responsible for the economy, two electronics engineers and a web designer. Everybody helps out with anything, like market clarification or business modelling, but as the CEO says; ”it is natural that you work more on what you are good at”. An example is that the product developer took the responsibility of writing the application to the Norwegian Research Council, or that one of the electronic engineers out of the blue built a custom made circuit board. Except from these, they do not have any other defined roles.

According to the product developer, the team has no manager with the ultimate saying. Everybody takes initiative. “Anyone can come up with a suggestion or proposed solution. If it sounds good, we’ll do it”. Still, the CEO is generally the one initiating meetings and handing out tasks, even though it is the team as a whole who decides. “We have never had a decision based on votes. We always end up agreeing on a solution.”

As a way to motivate, Connected Devices have introduced a dynamic model adopted from the book “Slicing Pie” - the more you work, the more shares of the company you get - which was
agreed upon by all team members. This benefits the CEO the most, as he is more dedicated and works full time, but the rest is fully aware of the possibilities in this agreement. The CEO is the only non-student among the team members.

Meetings
Connected Devices have regular meetings every Thursday, and if needed some additional irregular meetings. Typically, the weekly meetings are for planning future events, discussing strategy for product development and business, issues regarding financing and sharing of what has happened since last time. “If CEO has been in Europe on market clarification, he usually talks about what happened on these meetings”, the product developer adds. They both explain that they do not have a structured method for giving feedback. “It is more like; ‘good work’”, the product developer points out. It happens that they agree that they should have said “something different” to people in a meeting, but they rarely (or never) share personal feedback between each other. Usually, though, the CEO claims to try to pay attention to energy levels and if people are motivated, and intervene if necessary.

Working environment
Connected Devices’ only office is the premises of Start NTNU, which often is occupied as it is shared among several other startups at NTNU. Further, the product development usually takes place at the workshop next door, which the product developer describes as “chaos”. “It is not ideal to work here. It is an OK place to start, but we need new premises as fast as possible.”, he explains. The is messy and cramped, with no whiteboards or plants. “We miss whiteboard and projector.”, states the CEO. Within the next few months, the startup will move to their new office with more space.

Company ethics
There are some shared values and principles in Connected Devices. They claim to be open and honest, and try to focus on not being locked to one particular concept. “If we have a concept, we will try to test out other possibilities to see if that works”, the CEO exemplifies. Also, Connected Devices has a focus on close dialogue with customer. The product developer explains that they try not to undermine anyone in a discussion, but always create agreement. For brainstorming they use post-it’s, 3d-prints or sheets of paper.

Customer
Previously, Connected Devices made a product for the business market, typically restaurants. Their requirements where on roughness and lifetime. After a trip to Latvia in February where
the CEO met with the gas company, their focus changed. The gas company proclaimed that they wanted to use the product on the private market. As a result of this, Connected Devices’ focus is not on the end users anymore, which means their product is not the same. They know they have to make something that fits in for the end-user, but they focus on the customer (Intergas) and their requirements. The gas company wanted to pay 15 euros per product, which changed the whole focus of Connected Devices. Another requirement, the product developer highlights, was that the product needed to fit a larger size of gas cylinders. They now needed to redesign the whole product.

One of the reasons why they changed market was the immediate demand. Also, the need of several certifications in the business market - which is expensive - made the private sector more appealing. “To change market was the fastest way to make money”, the CEO explains. Interaction with the customer has been over mail lately, usually every third week or so. The next physical meeting will be in June, which is when the product should be ready.

4.3.2 Product Development
The product developer says they are now on their fourth prototype while the CEO claims this number is seven, as he includes the earliest versions and a web design. As a result of several iterations, due to changed requirements, the current prototype looks very different than the previous versions. Through help from Inventas they now focus on reducing production costs, which Connected Devices do not have much experience from, but which Inventas are good at.

Requirements
The product developer says that they are trying to get as much information before prototyping as possible. As the team is split up into different disciplines, the different team members are responsible to get the requirements they need for their own tasks. They must also communicate internally to share knowledge of different components, if that is needed.

As every prototype has been very different - due to change of requirements and customer focus - they have had to start almost all over again each time. Every time they try to get as much requirements and specifications as possible from their customer in advance. It is clear that these specifications are not very detailed, and usually develops during prototyping. Other requirements may come from the technicians, and are based on gut-feeling or know-how.

Connected Devices try to limit themselves in the sense of what features to implement, and not. The product developer explains that they have a focus on limiting the degree of electricity consumption. This affects what features they implement. “As an example”, the product
developer points out, “we do not implement Bluetooth now if not the customer asks for it explicitly, no matter how useful we think it would be”.

Another way to list requirements is through what they call a ‘customer journey’ or ‘customer road map’. According to the CEO, they try to imagine the whole process, and go through each scenario step-by-step. They do this focused on the customer's needs and so-called touch points - either through acting or drawings - to look for things they might have overlooked earlier and to optimize customer’s impression of them.

With the help from Inventas, Connected Devices listed almost 30 features they could implement in their product and tried to rate them. Based on customer requirements they could now come up with a new design for the next prototype. Their customer main requirement is clear; it must work and be cheap.

**Planning**

The product developer says his understanding of an iteration is “You make something, test it, and improve it. Every prototype is an iteration of the last one”, and states that this is what Connected Devices is doing. They do not have a system for this, as it is more a consequence of where their focus has been. “I imagine that in the future - when we have a more solid focus - will be more iteration driven. Until now, almost every prototype has been different. From now on, iterations will be more important”. The CEO agrees here; “Now we start with the process of what I would call ‘Iterations’.”. They actually have an agreement with the gas company, that from June they will do iterations with durations of one month on the prototype.

As an example of an iteration, the product developer explains that their first focus was the business market, where the requirements were very different. Here, they started with steel, but iterated later to plastic.

Except for some up-front requirements, planning the prototyping in advance is not a major focus for Connected Devices. They use deadlines if the customer requires results within a certain date, but according to the product developer the structured schedules they tried to implement around Christmas have not been followed up. According to the CEO, they actually do have a time schedule, divided into four areas: design, electronics, web interface and business, which they follow week-by-week. CEO explains: “This way we can see if we are ‘on track’. If we notice that we get behind on the time schedule, we will make a plan B”. He explains that it has not always been like this. Their focus has narrowed down lately, which makes it much easier with a more rigid plan, due to less changes.
Connected Devices do not use any method or representation to visualize team progress. They were thinking about defining Key Performance Indicators (KPI’s) and quantify features with a number representing how much value it would add to the prototype. Then, progress could be visualized after each milestone. The reason for this, the product developer explains, would be to define the goal of why they work with the thing they do, and to streamline the work. The idea came from a lecture one of them attended. They say that the reason why it has not yet been implemented, was that it was discussed during a time with a lot to do, and thus was abandoned. “Also”, the product developer continues, “the way we do it now has worked very well, so the need for a change has not been obvious”.

Regarding calculating cost and time of an implementation, they get a specific price and time estimate from Inventas. Other than that, it is not much used. The customer is not involved, as the people they talk to usually is salespeople. They give basic requirements, but leave the technical details to Connected Devices.

**Building**
During prototyping, there are no concrete rules stated by the Connected Devices. It is common that technicians work at the same time, and at the same part, but it is not obligated. The product developer usually works joint with other technicians, but state that those with electronics work more alone. The CEO claims that the Electronics usually work together, but that this joint development did not work for the designers.

Some parts are bought in as standard components, but it is not a conscious choice. As each prototype until now has been very different, all the parts are usually new too. “We do not think about reuse”, the CEO says.

If doing radical changes, Connected Devices test single parts of the prototype before they decide whether to keep or dismiss the change. The product developer explains; "We went from four to one weight springs during the iterations, and the electronics is placed differently, too. There is also a locking mechanism in the new prototype”.

**Review**
After new implementations, focus meetings are used to share the new changes with the other team members that were not involved. “One time, the product developer and I made this prototype. When we were done, we explained it to the rest and made a document elaborating how we proceeded, to make it easier for others to do it again.”, the CEO says. He also underlines how important it is that the design- and electronics engineers all the time knows
what each other is doing, as these are so dependent on each other. Remote working is a big issue.

Testing and verification from customer are tools used to learn from each prototype. The testing is done arbitrary. “We had to bring the last prototype with us to Latvia. That meant we had to delay testing it”. The product developer says. The tests are simple; they put a gas cylinder on top of the weight and are rough with it, to measure how well the product withstands the forces. That is the only test they do, though it was planned to do the testing in freezing temperatures as well. Some testing is done after each prototype, all together 3-4 times over a period of 10 months. They also show the product to the customer, to get feedback. “Once, we tested the prototype in a customer’s restaurant and it didn't work. It was clear that this had to be fixed. Here, the electronic guys take control” the CEO explains. “Another time, we tested the precision of the nacelles and found out that they were not as precise as we thought. Discussing this with our customer we realized that the precision we had were enough.”.

Connected Devices do not have any precise definition of when a product is ‘done’, other than to see if it works. “It is not done before it is done, and when it is done it is done”, the product developer states. They test it in the environments it is designed for; if it does what it is supposed to do, it is done. If the prototype is not what the customer wants, they still test it to see if it worked as they hoped. “The last prototype was made up of two molded parts, and electronics in the middle. We tested it to see if it measured precisely. When it did, we knew that it worked and we were happy.”

4.4 Case 4 - Designed products
4.4.1 Description of the company
Designed Products is delivering a purely mechanical device for easier transportation up and down stairs, particularly aimed at older people. Established in 2012 after an acid test at the NTNU School of Entrepreneurship, the trio consisting of a CEO, CMO and CTO has spent 5 years to develop their product. It was first out for sale in 2015, but the team is doing regular updates on the product to ensure high security, easy integration and attractive design, and they just recently they hired a new product developer. They fall in the designed products category as they develop a purely physical product.
Interview object 1: CMO. Master degree in Entrepreneurship with a background from Nanotechnology at NTNU.

Interview object 2: CTO. Master degree in Entrepreneurship with a background from Product Development at NTNU.

Frameworks claimed to be used
According to CMO, Designed Products do not use any particular methodology. For them, it is about discovering new concepts, by testing out different possibilities. Their self-developed way of doing product development is based on experience. He states that they probably could have learned a lot earlier with supervision from likewise startups, but this was hard to find at that time. According to the CTO, they always try to keep customer and user in focus. She says that they are always open for new ideas, and use ‘green phases' to think outside the box. Now, Designed Products use to teach other startups about their own fails and victories. Changes done to their methodology is provoked through mistakes. “Then we have a greater ownership to the change”, CMO says.

Roles
Designed Products consists of two product developers. Their job is to find out how things work together and come up with new concepts and solutions, but they also do technical drawings in CAD and purchase of equipment or materials. CTO has the last word concerning the product; she decides what to be implemented, and checks the product for errors subsequently. CMO oversees market research and helps out with business development together with CEO. The CEO is also the one facilitating for product development, get capital, resources and partners. Their recent hire, has been assigned his own project. Here, the PD has the responsibility of ordering equipment and test prototypes.

The product developers are responsible for developing prototypes, but always discuss new features with the rest of the team before implementation. This has to do with cost and time. When it comes to costs, it is the CEO who decides in the end. CMO exemplifies: "One time we had a dilemma whether to invest in a new casting mold for 20,000 NOK or not. We would discuss what benefits this would

Meetings
Meetings are held once a week, usually on Mondays, and can last from 30 to 120 minutes. Here, they discuss product development and issues regarding production, together with
updates and general planning. In addition, they have some workshops now and then to brainstorm around new concepts and ideas.

Except from the first couple of years they do not facilitate personal feedback. They have been working together for 5 years now, and know each other’s strengths and weaknesses. “This is better. Then we do not have to change each other” - CMO.

**Working Environment**
The designed product startup rents office space at Digs in Trondheim centrum. Here, they sit with several other startups. They have their own office - a corner office for four people - a large workshop in the basement, and have access to meeting rooms almost all the time. Both CMO and CTO express satisfaction over their premises. As for the working hours, all team members work approximately 40 hours each week.

**Company ethics**
During Designed Products' recent hire of a new product developer, they could instead have chosen an older, more experienced guy. They settled for their new employee as a result of his flexibility and motivation, which is characteristics Designed Products value. They also emphasize the importance of testing a product. “Vi do not sell anything unless it is tested with us first” says the CTO. Sometimes this limits their partners’ possibility to find new solutions or extra parts they want to add to the product. “Security is very important. Either we test it first, or it has to be according to these standards”

Designed Products has come up with the following values for the relationship between customer and product: 1) The user must be stimulated to more activity, 2) The user should be proud over their product, and 3) The products design should fit the user’s home.

**Customer**
The main customer of Designed Products is a governmental help central. Based on an application the governmental help central get from users, they can approve and assign them the product they applied for. Therefore, for Designed Products it is very important that the end users know about them, and that the governmental help central consider them as a worthy product.

Designed Products has tried to involve the governmental help central along the way with several meetings, discussing what the product should contain. They have tried to initiate a
more structured cooperation, but this is not wanted by the governmental help central as they do not want to undermine potential competitors.

Lately, Designed Products has also started to sell in Sweden, Netherlands and Germany, where they sell through distributors.

4.4.2 Product Development
Designed Products is now working with their fifth prototype, each of them being radically different from each other. The first three prototypes were developed over 9 months when they were still students in 2012/13. During the development, both functionality and ease-of-use got better. "This was a result of several increments, and a lot of try-and-fail", CMO says. The fourth prototype was made to prepare for mass production, and was finished april 2014. “We failed with our first step to industrialization”, the CMO says. “It looked very good, but both functionality and ease of use was worse than both prototype 3 and prototype 2”. From this prototype, they learnt how important it was to do proper testing. “We realized that the things we did to make prototype 4 better actually made it worse”. They removed some of the new features and worked hard with strength calculations. The current prototype is now out for sale, but constant iterations are performed to improve the product.

Requirements
From the very beginning, the requirements of Designed Products has been rooted in the customer’s need for safety. The concept was based on a research article, and later input has come from experts such as ergo therapists and physiotherapists, their customer (the governmental help central), and the end users themselves. They explain that some of the problems they try to solve only appear after several months of usage, so feedback is crucial to improve the product. "The folding mechanism worked in the workshop, but after 3-4 months with the end user it started to strike and became difficult to manoeuvre." the CTO exemplifies.

Based on feedback they make a list of criteria’s and features the product must possess, often as many as thirty. They then prioritize them on a list divided into “features we must have”, “features we should have” and “features we could have”, where the prioritization is based on how consistent the feedback is. “Security is on the top. If the feedback is security-related it is always on the ‘must’ list” the CMO explains.
**Planning**

The CMO’s understanding of iterations is “smaller steps”. He explains that it is important to do changes that get you in the right direction as fast as possible. That way you can get more done in less time. For Designed Products, the iterations can last from 1 week up to several months, depending on feature. One time, they made a change on the handle, based on customer feedback. For this, they used a local manufacturer and three iterations to get it fixed, where every new iteration was sold as a consequence of continuous orders on the product. For other, more radical changes, they usually utilize 3d-printing to do fast and cheap experiments.

Time-frames and deadlines is a dilemma for Designed Products. They want a product delivered as fast as possible, but still it must hold their standards of quality. There are situations where, if they do not find anything better within a deadline, start working on something new instead.

To plan for the prototyping Designed Products do not use fictive user-stories, but rather discuss specific customer or installer problems, received from previous feedbacks. They use Trello to visualize progress. There they have checklists, where certain implementations are broken down into as simple steps as possible. The CMO states that this is not used to communicate actively, but by checking tasks 'done', they can keep an eye on the overall progress regarding product development.

Designed Products choose their suppliers through a testing process. They order several samples from different suppliers and examines the quality. Sometimes they also integrate them all the way down to product development, but this do not happen a lot. “We really want more manufacturers to come up with suggestions on how it would be easier to adapt to their production methods” - CMO. They also use their suppliers to estimate costs for production of certain parts. The calculation of time used is more based on experience.

A focus on standardizing has been increasingly important for later prototypes, especially since their product already is out for sale. They think about how the product should be three or more years from now, and thus use extra time to make that standard foundation. "An example", CTO says "is the handrail, adapted to fit almost any type of future handle".

**Building**

Material and parts are bought through suppliers. Most of the product is assembled by the installers at the user’s premises, but some important and difficult parts are assembled by Designed Products themselves in-house. This is of course related to the latest version, which
they sell. At the workshop, frequent iterations on the prototype is done every month. Their latest change in concept, where the new product is supposed to be much cheaper to produce and easier to install, is a relative big change and requires both new equipment and new suppliers. They usually produce a smaller quantity of the iterations they believe in, because they know they must deliver frequently.

When doing changes, they try to focus on isolated changes where the main parts are the same. They also try to limit changes of plastic parts as the molds are very expensive, opposed to machined metal parts which is much cheaper. The CMO states that the changes in prototype mostly is related to costs.

The mere construction of prototypes is done by both the PU and the CTO, where the CTO is the only one allowed to freely do changes. "Still, I always want to make sure people agree on the big changes, before they are executed. Smaller, superficial changes I can do without discussing with the rest of the team", says the CTO.

**Review**

Designed Products always call the user 2 weeks after the user has received the product, to get feedback. “We learn a lot about what they think is good, and what needs to be better.”, CMO says. They also talk to specialists and test the product in the workshop. After they talk to enough people, they can form a picture of what is real and important, and what is not. This information is further used in their prioritized must-should-could list. If a test fails and it is on the must-list, they talk to suppliers and experts to find a solution, no matter what. "With prototype 4 we used 3-4 months before we tested mechanical strength", the CMO explains, "Now, we have a test form with things to test, how to test them, and what we do if it fails". Thus, testing of prototypes has been reduced to two days. There are also some things they do not have test for, like to see how the product behave after for example 10,000 repetitions. It could be dust entering the mechanics, material fatigue, or it could just be used the wrong way.

According to the CMO, they are not very good at defining a finished product and it often ends up with subjective opinions. They say that this is something they can be better at, especially with the recent introduction of the must-should-could list.
4.5 Case studies summarized

Four different hardware startups have been presented in the above case studies. In table 4 below, each characteristic retrieved from Meyer (2014) table of Agile characteristics are presented in each row (in addition to the top row added by the researchers). The researchers have analyzed each of the four cases and summarized their adoption of each characteristic. This can be used as a basis for further discussion, justifying why some of the differences are present (Mullen et al., 2009).

Table 4: Case studies summarized

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Robotics</th>
<th>Wearables</th>
<th>Connected Devices</th>
<th>Designed Products</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frameworks claimed to be used</td>
<td>Mentions Scrum, and uses Kanban through the tool Trello</td>
<td>Mentions Kanban, but uses also Scrum and Lean Startup without mentioning it specifically as a framework.</td>
<td>Try to implement Design Thinking, with a high customer focus through ‘empathizing’. Use time schedules and deadlines</td>
<td>Do not use any particular methodology. Though, use Trello. Self-developed, based on experience.</td>
</tr>
<tr>
<td>Values</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Redefine roles</td>
<td>The managers jobs are limited. The tasks are assigned by the team as a whole. Test-users are involved in the product development.</td>
<td>Task stem directly from the user stories and are picked based on experience and responsibility. Work close with customer</td>
<td>CEO facilitates for the rest of the team. Do not have highly defined roles. The team always come to an agreement on a solution.</td>
<td>CEO facilitates for the product developers. CTO and PU are responsible for the product, but must discuss new radical features with the whole team. Close work with both user and customer.</td>
</tr>
<tr>
<td>No &quot;Big Upfront&quot; steps</td>
<td>The team has an idea of how the product will look like, uses itself as a test user and also has test-users. Use 1 week iterations with Scrum.</td>
<td>To some degree. They have a requirements document which is updated continually, but they do not frequently show prototypes or demonstrators to the customer.</td>
<td>Some requirements are received from the customer up-front. Still, several changes has been done iteratively throughout the development phases. A total of 4-7 prototypes with large changes.</td>
<td>Concept is developed through frequent interaction with customer and end user.</td>
</tr>
<tr>
<td>Iterative development</td>
<td>Yes, one iteration per week. Prioritized list is made by internal team.</td>
<td>Yes. Will select for implementation the function that has the highest Return on Investment (ROI).</td>
<td>Increments are done frequently, iterations are done irregularly.</td>
<td>Frequent iterations, that can vary from 1 week to several months. Implementations are chosen from a prioritized list based on customer feedback.</td>
</tr>
<tr>
<td>Limited, negotiated functionality</td>
<td>Do not limit features based on their business value</td>
<td>Say they want to limit features based on business value.</td>
<td>Only implements features requested by customer. Prototypes are simple.</td>
<td>Use list of prioritized features. This enables them to implement only the most requested features.</td>
</tr>
<tr>
<td>Focus on quality</td>
<td>Frequent testing, after each iteration.</td>
<td>Regular testing. Not every week. Not possible to do for all requirements since some of them have to be considered subjective.</td>
<td>Testing is arbitrary. Ca 3-4 times over the last 10 months. Evaluate every failed test with customer.</td>
<td>Testing is a major issue, but not done regularly. Large focus on security.</td>
</tr>
<tr>
<td>Principles</td>
<td>Put the customer at center</td>
<td>Self-organized team</td>
<td>Work at sustainable pace</td>
<td>Develop minimal functional prototype</td>
</tr>
<tr>
<td>------------</td>
<td>--------------------------</td>
<td>--------------------</td>
<td>--------------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td></td>
<td>Frequent contact with test-user and utilize user stories.</td>
<td>All low-level decisions are made by the developer themselves. All decisions are made on team-basis.</td>
<td>50-60 hours per week</td>
<td>Great focus on doing as little amount of work as possible on each feature, to find out what solution to choose.</td>
</tr>
<tr>
<td></td>
<td>With frequent contact with customer, but not user. They have had workshops and discussed several prototypes.</td>
<td>Their system should allow most team members to work remotely. Claims to be 100% self-organized. Trusts that everyone does their job satisfactorily.</td>
<td>Minimum 60 hours each week</td>
<td>Only build the functionality the customer is willing to pay for.</td>
</tr>
<tr>
<td></td>
<td>Claims to put customer in focus, designs product after customer requirements. Irregular contact, over mail. No contact with end user.</td>
<td>Changes must be discussed in plenum before implementation. Team decides as an entity. Members work when they want to. Everyone is responsible for the requirements needed for their area of responsibility.</td>
<td>Varies among the members. Average of 15 hours/week per person.</td>
<td>In the beginning they tried things based on gut feeling. Now, only implement features requested by customer.</td>
</tr>
<tr>
<td></td>
<td>Requirements are based on customer feedback. The upfront requirements are changed. Regular interaction with both user and customer.</td>
<td>CTO and PU comes up with new solutions. CTO has the last word concerning the product. Decisions related to large cost expenses must be evaluated by the CEO first.</td>
<td>Standard workweek of 37.5 hours.</td>
<td>Requirements are based on customer feedback. Only implement what is requested, after prioritizing them.</td>
</tr>
<tr>
<td>Practices</td>
<td>Daily meetings</td>
<td>Planning game</td>
<td>Continuous integration</td>
<td>Retrospective</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------</td>
<td>---------------</td>
<td>------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td><strong>Express requirements through scenarios</strong></td>
<td>Important requirements are picked from several customers and made into user stories.</td>
<td>Scenarios are made together with the customer and end-users. Divide into 'epics' and 'user stories'.</td>
<td>Use 'customer journey' or 'customer road map'. Focus on customers' needs to optimize interaction. Done through acting or drawings.</td>
<td>Use real life scenarios, based on actual customer/user feedback.</td>
</tr>
<tr>
<td>Joint development</td>
<td>CAD files are systematically shared among the two set of developers. 1 part does the mechanical CAD and makes comments, and the other does the design considerations.</td>
<td>Have not tested it yet, but it makes sense because this can detect mistakes before the whole prototype is tested.</td>
<td>Sometimes two or more work together, but there are no norms related to joint development.</td>
<td>They usually work together, but not at the same thing. Discuss with each other about uncertainties.</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Simplest solution that can possibly work</td>
<td>Have started doing this after time restrictions have become more apparent.</td>
<td>Main focus lies on making simple prototypes for customers.</td>
<td>Prototypes started very simple. Only implements required features. Gets pushed by Inventas to do simple and cheap solutions.</td>
<td>Focus on simple, cheap and isolated implementations. Have a greater idea of what customers want, as they are already selling their product.</td>
</tr>
<tr>
<td>Standardize</td>
<td>Do not use standard components.</td>
<td>Hard to explain a standard in electronic design. Every component is standard to some degree.</td>
<td>Buy standard components, but not planned. Very different prototypes, so there has been less need for this.</td>
<td>Standardizing has been increasingly important, and extra time is used for this. Changes should fit new implementations several years in the future.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Artifacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>User story</td>
</tr>
<tr>
<td>Burndown chart</td>
</tr>
<tr>
<td>Story card</td>
</tr>
<tr>
<td>Story board</td>
</tr>
<tr>
<td>Open room</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team</td>
</tr>
<tr>
<td>Product Owner</td>
</tr>
<tr>
<td>Scrum Master</td>
</tr>
<tr>
<td>Customer</td>
</tr>
</tbody>
</table>
5. Findings

This chapter presents the key findings and analysis that emerged from the case studies. All four hardware startups are presented in the findings both separately and combined. They are presented combined when it is necessary to have a point of reference to companies who develop physical products in general and separately when the cases are clearly different. Each finding below is linked to the research questions and theory above. This chapter will act as a foundation for the next chapter, 6. Discussion, where the key findings and research questions will be discussed together with the theory. Each of the first four sub-chapters represents one key finding, with several findings related to the key finding. The four findings have been chosen for their relevance and impact to the research questions.

5.1 Key finding 1: All the interviewees have a diverging understanding of what an iteration, a prototype and an increment is

The case studies all present a different view of what they define to be an iteration, prototype and increment, and based on literature, some of these assumptions presented in the case studies are wrong. Even the interviewees within the same startup have diverging understanding of the three words. Each case study’s ideas and definitions of the words iterative, increment and prototype are presented in their respective sub-chapter. The last sub-chapter presents the case studies collectively. Since, iterations, prototypes and increments are a central part to all activities in Agile Product Development, the findings below are presented in a holistic manner. Several aspects are considered in the context of the respective case study, and because the different startups develop different type of products and are in different stages in development, the findings cover a large part of the collected data.

5.1.1 Robotics

*Iteration*

Robotics plans each iteration in a detailed manner, through requirements, with two meetings, and delivers on those plans. Iterations are used both for high-level requirements and low-level requirements. Customers are not always included in the low-level iteration. They do iterations on parts of the prototype. For them, an iteration is directly connected to what prototype they are going to build.

*Prototype*

The CTO states that they have made 28 prototypes. Each prototype has a sub-version with slightly different functionality. For example, 28.A, 28.B, and so on. The prototype goes from
28 to 29 when it feels right – it is chosen subjectively. The prototype is never defined as ‘done’.

**Increment**

Increments are made for specific goals and milestones in the project. These goals and milestones are business specific and are not exclusive to product development.

5.1.2 Wearables

**Iteration**

Wearables see a sprint as the same thing as an iteration. An iteration is defined through the prototype, which always needs to serve a purpose. At the end of an iteration the prototype has been built according to the purpose and tested thereafter. Iterations become more specific regarding the purpose after a couple of iterations, where the first iterations often are more exploratory.

The startup uses iterations with high-level requirements which last 1 month, and low-level requirement which can last much shorter, down to 1 week.

The startup defines an iteration as finished when the prototype is finished, and they don’t set time limits – even though they must guess how long it will take to be able to do better planning. While they are in iteration mode, they freeze requirements.

**Prototype**

The CEO tells us they have made over 100 prototypes. The hardware engineer says they have made 5.

**Increment**

The startup has shown several prototypes to the customer, but are not able to tell which of the prototypes these are.

5.1.3 Connected Devices

**Iteration**

The startups have regular meetings every week where they discuss plans for future events and lay a strategy for the product development. They can also discuss the status of product development.

The company has made several prototypes where the requirements have changes after each version. They try to obtain as much information before prototyping as possible, and the
different team members are responsible to get the requirements they need for their own tasks. Except for some up-front requirements, planning the prototyping in advance is not a major focus for Connected Devices. They use deadlines if the customer requires results within a certain date, but the structured schedules are seldom used.

The CEOs understanding of an iteration is: “You make something, test it, and improve it. Every prototype is an iteration of the last one”. In the interviews the CEO points out that they test every prototype, which indicates that they focus on partial results.

**Prototype**
The company made a prototype for their customer but discarded it after the customer told them to change focus to another market the customer had access to.

The engineer in the company claims they have made 4 prototypes, while the CEO claims they have made 7, which includes the earliest versions and a web design.

A prototype is described in both as something which is shown to a customer (increment) and as something used internally to improve on the technical requirement.

**Increment**
The startup has tested some of the prototypes with customers. Most of the prototypes were made for technical testing. The company has not yet shown an increment which the customer has accepted. The last prototype they made was not accepted and they had to start from scratch.

5.1.4 Designed Products

**Iteration**
The startup is iterative in the way they improve their way of working. They improve when they have learned from their mistakes.

The startup defines iterations as “smaller steps”. They are used to take make the right decisions in product develop in a faster way (less time).

**Prototype**
The product developers are responsible for developing prototypes, but always discuss new features with the rest of the team before implementation. This has to do with cost and time.

The startup is developing their fifth prototype, and each prototype is completely different from the other.
**Increment**

The startup defines high-level prototypes, for example prototype 3.0, 4.0 and 5.0, as increments. Low-level versions of this prototype, for example prototype 3.1, 3.2 and 3.3, are called iterations.

The startup did not show their first functional prototype to a customer, since they found ways of improving it themselves. Their second prototype was shown to customers (increment) with an overwhelming good response, and they made changes based on their own technical knowledge since the customer was so pleased to see their solution. Their third prototype was shown several times to customers and was also iterated on. Prototype four was worse in both functionally and usability since they made it for production. Prototype five is their last increment which they are continually improving, and is the prototype they are currently selling.

In general the startup did not make a simple functional prototype for their customer, since the customer wanted to see a generalized solution.

5.1.5 Combined case studies finding and analysis

Table 5 shows which words were mentioned by the different case studies, and table 6 shows if they actually practice the meaning of the words iteration, prototype and increment. Table 5 also includes synonyms or similar meanings for the words, like cycle or loop as a synonym for an iteration, sprint as the believed meaning for the word iteration, and MVP as a synonym for increment. Table 6 is derived from analysing the practices of the case studies above.

**Table 5: Mentioned the word in the case interviews**

<table>
<thead>
<tr>
<th>Word/Case</th>
<th>Robotics</th>
<th>Wearables</th>
<th>Connected Devices</th>
<th>Designed Products</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>Cycle or loop</em></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td><em>Sprint</em></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prototype</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Increment</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MVP</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
Table 6: Uses iteration, prototype or increment in product development (not only mentioned the word)

<table>
<thead>
<tr>
<th>Uses</th>
<th>Robotics</th>
<th>Wearables</th>
<th>Connected Devices</th>
<th>Designed Products</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterations</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Prototyping</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Increments</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

When comparing the two tables it becomes clear that the startups practice iterations, prototyping and increments, even if it is only to some degree, and even if they do not use the established words for their practices. Iteration and increment are not established words in the chosen case studies. Prototyping is well known.

Besides table 5 and 6, the startups all agree that a product release for a customer needs several sprints, cycles, iterations, prototypes and MVPs. And they therefore use incremental development without knowing the word incremental or being able to explain why they show several prototypes to the customer before releasing the final product.

5.2 Key finding 2: The length of an iteration seems to vary to a large degree between the case studies

It is found that all cases practice iterations in various lengths, but distinguishes between long and short ones. The length of the long iterations varied from one to six months, and the length of the short iterations varied from one to two weeks. This key finding is closely related to key finding 1, where this key finding works as an additional example of how startups have diverging understanding of what an iteration is. In the following subchapters, each startup is presented separately and combined.

5.2.1 Robotics

Robotics have a product planning meeting each week, which can be seen as the start of an iteration. The week iterations are often used to finish low-level requirements. Iterations for high-level requirements are not time-boxed and are subjectively chosen as ‘done’. The high-level requirements can therefore last from 1 month to a half year depending on the milestones and goals of the company.
5.2.2 Wearables
Wearables' iterations lasts approximately 1 month. Each 1 month cycle can contain several shorter iterations which satisfy requirements within the 1 month cycle. A short cycle within the 1 month cycle can for example last two weeks. The development team tries to estimate the time it will take to do an iteration, but does not practice time-boxing, since an iteration is not finished until the prototype is not finished.

5.2.3 Connected Devices
Connected Devices have regular meetings every week, but not necessarily discuss product development at every meeting. The startup has not had time-boxed iterations yet, but has planned to meet the customer every month, so each high-level requirement iteration (and increment) approximately lasts 1 month.

5.2.4 Designed Products
The iterations of Designed Products can last from 1 week up to several months, depending on feature. The startup has weekly meetings where they discuss product development, but this is not related to iterations and iterations are not time-boxed.

Since the prototypes need testing over a long period to enable durability testing, the iteration cycles could be 3-4 months long. Some functionality is not possible to test within a short period of time and could take longer than 4 months. The startup always contacts the user two weeks after they have received the prototype to receive feedback, which also make the iteration cycles longer. They are not good at defining at done iteration (or prototype) and the planned iteration time is sometimes doubled or tripled. Shorter iterations are often not used.

5.2.5 Combined case studies finding and analysis
Table 7 is a summary of the individual case findings and shows how long the iterations of each startup is. All the startups distinguishes between long iteration cycles with high-level requirements and short iterations cycles with low-level requirements. The table below shows these findings.
Table 7: Length of iteration and use of time-boxing

<table>
<thead>
<tr>
<th></th>
<th>Robotics</th>
<th>Wearables</th>
<th>Connected Devices</th>
<th>Designed Products</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length of (long) iteration</td>
<td>1-6 months</td>
<td>1 month</td>
<td>1 month</td>
<td>3-4 months (or longer)</td>
</tr>
<tr>
<td>Length of (short) iteration</td>
<td>1 week</td>
<td>2 weeks</td>
<td>1 week (not always)</td>
<td>1 week (or longer)</td>
</tr>
<tr>
<td>Time-boxing</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

The table points out that the long iterations can last from 1 month to a half year depending on the requirement that needs to be fulfilled. Short iterations typically last from 1-2 weeks and included fulfilling the low-level requirements which could be everything a feature of a product, to a part of the final solution. Short iterations were mostly not shown to the customer. Time-boxing is not practiced.

5.3 Key finding 3: The frequency of testing for hardware startups is lower compared to the Agile Manifesto, and there is large variation between different hardware startups.

One of the key concepts in the agile methodology is quality, which is achieved through frequent testing (Agile Manifesto). It is found that, although testing is a crucial part of hardware development, the frequency of testing is much lower than proposed by the Agile Manifesto. Among the four cases, none of them tested on a daily basis and the frequency varied a lot. The usual cases varied from one time every second or third month, to three to four times every month. The companies gave lack of resources and testing equipment, and time-consuming developing processes as the main reasons for this.

5.3.1 Robotics

The Robotics company do some testing after each prototype, with a new iteration every week. The testing, which is performed both at the workshop and with end-users, is twofold; technical tests and user feedback. The CTO explains that some tests are difficult to perform, as for example specific parts might never have been produced before. Also, testing the prototype is extra time consuming due to the large space required (the product includes a 60 feet long wire). Their product is mostly comprised of 3D-printed parts, and by having a 3D-
printer nearby, they reduce the time between each prototype and thus between each test. Still, 3D-printing can be very time consuming, with up to 10 hours for even small components.

5.3.2 Wearables
The CEO of the Wearables company describes the problem of testing by the fact that “testing requires specific things to test, and the right equipment. The reason why we do not test more often is resources”. On the contrary, he explains that they sometimes can test several times a day if the test equipment is already prepared and ready to use. Wearables always make prototypes for the purpose of testing something, but sometimes the testing is hard because the exploratory prototyping itself is the purpose. Wearables have one software developer and one hardware developer, and usually test several times each month.

5.3.3 Connected Devices
Connected Devices has the least frequent testing, only every couple of months. The employees here say that the most important thing with a prototype is that it works based on customer requirements, which is validated through testing. In addition to simple technical tests, verification from customer is also an important feedback. For their newest customer this has only happened once, with the next feedback in July. The startup has developed four to seven prototypes over the course of approximately 1 year. During that period the prototypes has changed alot, and some of the new parts are more difficult and time-consuming to produce. CEO examplifies this with some plastic parts they had to mold themselves. On average, each person in Connected Devices work 15 hours each week, and they rarely work everyone together.

5.3.4 Designed Products
Designed Product startup has a high focus on testing, which they learned after their fourth prototype. As they develop a product related to the safety for elderly, it is of major importance that all components are properly tested. Still, testing is done only after new implementations, which takes time. This is partly due to a time-consuming development process of new components, and partly because they have to wait for suppliers to deliver material and parts. Feedback from end-user is retrieved 2 weeks after every new sale, where they learn alot. Some types of test are not possible before some time has passed, like how the product functions after months in user environments.
5.3.5 Combined case studies finding and analysis

The following table shows control variables for the different cases. The data is retrieved from the case studies, and can contribute to explain why Robotics has the highest testing frequency, and Connected Devices has the lowest testing frequency.

Tabell 8: Analytic description of the cases

<table>
<thead>
<tr>
<th></th>
<th>Robotics</th>
<th>Wearables</th>
<th>Connected Devices</th>
<th>Designed Products</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usual testing frequency</td>
<td>Every week</td>
<td>Several times each month</td>
<td>Every third month</td>
<td>Several times each month</td>
</tr>
<tr>
<td>Average working hours/week/person</td>
<td>50</td>
<td>60</td>
<td>15</td>
<td>37.5</td>
</tr>
<tr>
<td>Product Developers</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>Total working hours (product development)</td>
<td>150</td>
<td>120</td>
<td>75</td>
<td>75</td>
</tr>
<tr>
<td>Age</td>
<td>2 years</td>
<td>2 years</td>
<td>1 year</td>
<td>5 years</td>
</tr>
</tbody>
</table>

5.4 Key finding 4: Hardware startups mix concepts from different Agile frameworks

As presented in theory, there are several Agile frameworks applicable to hardware startups. All cases, except Designed Products, claims to be using several different types of frameworks. The frameworks claimed to be used are Kanban, Scrum, Lean Startup and Design Thinking. In practice, the startups only use some of the concepts from the Agile frameworks they claim to use, and customize their own frameworks by unconsciously borrowing concepts from several other Agile frameworks, with Flexible Product Development as the most prominent.

5.4.1 Robotics

The Robotics company utilize Scrum in several aspects, including having a product backlog and sprint log. The CTO also categorize his own position as a Scrum Master. The traditional Product Owner mentioned in Scrum is here a position shared among the whole team. In addition to this they use Trello - a collaboration software tool that organizes projects into virtual boards. From a framework point of view, Trello is identical to that of Kanban with visualization of progress.
5.4.2 Wearables

Wearables mentions Kanban as their only framework which they use through an online platform called Taiga, but are describing themselves as 'lean'. In fact, they have roles resembling both Product Owner and Scrum Master, and focus on simple prototypes with fast learning cycles, all characteristics of both Scrum and Lean Startup. They do not use time limits and justifies this by their understanding of Agile as an opponent of this.

5.4.3 Connected Devices

Connected Devices use Trello, and claim to be using Design Thinking with a high customer focus. This is said to be retrieved from a school course with the same name. They do have an apparent understanding of their customer's requirements, but the frequency of customer interaction and the frequency of iterations is lower than that of the framework. They also practice the Scrum Roles, with CEO as Scrum Master and the team as a whole as the Product Owner. Unlike Robotics, they do not have a name for these roles.

5.4.4 Designed Products

Designed Product is the only case claiming to have no methodology. They use Trello, but claim that their 'methodology' is self-developed. They have a focus on incremental development, use iterations, focus on stakeholders - both suppliers, customers and users - and have also the roles as Scrum Master (CEO) and Product Owner (CTO), except that they do not call them by their Scrum name. Unaware of it, Designed Products thus actually utilize several of the agile frameworks. One could comment on the fact that this case is much older than the other cases and thus should have a better understanding of what they use, but this could be an argument for the opposite as well.

5.4.5 Combined case studies finding and analysis

In table 9 (below), an analysis with respect to the characteristics of the seven different frameworks - Scrum, Lean Startup (Lean), Design Thinking (DT), Kanban, Flexible Product Development (FPD), Extreme Programming (XP) and Feature Driven Development (FDD) - has been performed. The table shows what characteristics are utilized by the four hardware startups, retrieved from Theory (Chapter 2). As shown, Flexible Product Development (FPD) are dominant in the number of utilized characteristics, even though this framework is not mentioned by any case during the interviews.
Table 9: Characteristics from Agile frameworks

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Robotics</th>
<th>Wearables</th>
<th>Connected Devices</th>
<th>Designed Products</th>
<th>Typical for</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterations</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Scrum, Lean, DT, FPD, XP, FDD</td>
</tr>
<tr>
<td>Incremental</td>
<td>X</td>
<td>X</td>
<td>x</td>
<td>x</td>
<td>Scrum, Lean, Kanban, DT, FPD, XP, FDD</td>
</tr>
<tr>
<td>Visualization</td>
<td>X</td>
<td>X</td>
<td>x</td>
<td>x</td>
<td>Scrum, Kanban, FPD</td>
</tr>
<tr>
<td>Rapid prototyping</td>
<td>X</td>
<td>X</td>
<td>x</td>
<td>x</td>
<td>Scrum, Lean, DT, FPD, XP, FDD</td>
</tr>
<tr>
<td>Stakeholder interaction</td>
<td>X</td>
<td>X</td>
<td>x</td>
<td>x</td>
<td>Lean, Kanban, DT, FPD, XP, FDD</td>
</tr>
<tr>
<td>Focus on team collaboration and communication</td>
<td>X</td>
<td>X</td>
<td>x</td>
<td>x</td>
<td>Kanban, FPD</td>
</tr>
<tr>
<td>Self-organizing</td>
<td>X</td>
<td>X</td>
<td>x</td>
<td>x</td>
<td>Scrum, FPD, FDD</td>
</tr>
<tr>
<td>Freeze requirements</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>Scrum</td>
</tr>
<tr>
<td>Limit WIP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Kanban</td>
</tr>
<tr>
<td>Adaption of plans</td>
<td>X</td>
<td>X</td>
<td>x</td>
<td>x</td>
<td>Scrum, Lean, Kanban, DT, FPD, XP</td>
</tr>
</tbody>
</table>

Illustrates what Agile characteristics - presented in the seven frameworks presented in theory are utilized by the cases investigated. The column to the right depicts
6. Analysis and discussion

In this chapter, the four key findings will be analysed and discussed. In the first two sub-chapters, relevant literature from the theory chapter is included to give the reader an overview of how the key finding could contribute to previous literature. Every sub-chapter will be divided into individual discussion and collective discussion, except from the discussion of key finding 2. Each finding will be discussed up against the research questions, and makes the foundation for conclusion, implications and suggestions for further research.

6.1 Analysis and discussion of key finding 1: All the interviewees have a diverging understanding of the what an iteration, a prototype and an increment is

This section will discuss key finding 1 (chapter 5.1), and how it is related to existing literature.

The first sub-chapter will present how the Agile frameworks from chapter 2.3 relate to the words iterative, increment and prototype. This key finding is also strongly related to chapter 1.2.8, and the reader is encouraged to re-read that chapter to better understand this finding.

In the following sub-chapters, key finding 1 is discussed in five parts. The first four parts are discussions regarding each individual case, and the last discussion part is a collective discussion about all four cases.

6.1.1 Theory

The literature review has shown that all the presented Agile frameworks are both iterative and incremental. When comparing the traditional predictive processes and the processes in the Agile frameworks, the things separating them the most are how they relate to iterations and increments.

The easiest way to understand Agile frameworks in general could be comparing them with traditional development. Like figure 1 in 2.1.1 shows, the predictive (traditional) process has a long-lasting development time-box with a product release at the end, and the incremental (Agile) process has several smaller product (or prototype) releases within the same time-box. Each release is iterated on. (Smith, 2007)

The Scrum framework brands iterations as sprints, but even though all sprints are iterations, not all iterations are sprints (Schwaber, 2004). As an example, you can have a three-week sprint, but have three one week (internal) iterations within that sprint. A sprint is therefore a
fixed timebox in which you do all the activities you need to, from analysis to deployment. This can be further understood through (figure 4 in 2.3.2).

Scrum is also incremental and each iteration of work build on the previous increment. The increment is often replaced when learning has occurred in the previous iteration. (Schwaber, 2004).

The Lean Startup build-measure-learn cycles are the same as iterations, and the minimum viable product (MVP) is the same as an increment.

Kanban is incremental and iterative in improvisation of the process, and is a supplement for other frameworks in the Agile approach. (Meyer, 2014)

Design thinking is iterative, where each time the process is repeated, the scope gets narrower and the increments more focused. (Brown, 2008)

In Flexible product development iterations are central, and one or multiple iterations make up an increment, which is one release to a customer. (Smith 2007).

6.1.2 Robotics
It seems like the startup defines a high-level prototype as an increment, since some iteration include customers, and some do not. This seems to be so, because of the high resource demand for including several customers in each iteration. This seems to fit with the framework Scrum, since they deliver an increment after each high-level iteration. The length of the iteration is different and will be discussed in key finding 2.

6.1.3 Wearables
Since this startup sees a sprint as the same thing as an iteration, and since they do not time-boxing for each iteration, they have misunderstood the word sprint from the framework Scrum.

There also seems to be a varying in-house understanding of what a prototype is and how it is defined. Based on literature, a prototype is already a broad concept, and this startup seems to use the word prototype for both an increment and a functional prototype of part of the solution and for internal technical testing.
6.1.4 Connected Devices

As every prototype has been very different - due to change of requirements and customer focus - they have had to start almost all over again each time. This influences the terms they use.

It seems like the startup has been able to do a lot of minimal functional prototyping, but has not been able to do iterative development. The CEO stated that they would become more iteration driven in the future, because they now have an agreement with their customer.

“I imagine that in the future - when we have a more solid focus - will be more iteration driven. Until now, almost every prototype has been different.” – CEO, Connected devices

For not having a definitive customer, it seems as if the connected devices startup has not done iterations on the exact same product concept every time. Major changes on the requirements have happened and the startup had to start over. At the same time, the startup has made several increments, but has discarded several after customer interaction. This means that they have been doing incremental development with a focus on building minimal functional prototypes. This makes sense when considering the case study.

It seems like it is hard to do focused iterative and incremental development if a company does not have a definitive potential customer who can give feedback on the developed prototypes.

It seems like the startup is looking for a new development technique, because of their sharpened focus towards a specific customer (and not a market). Considering figure 2 in 2.2 by Böhmer et al. (2015), it seems like the startup is in a transitioning from the phase Idea Selection to Concept Development.

6.1.5 Designed Products

Designed Products did not start with the Agile approach. Combined iterative development and incremental development did not start before their third prototype, when they started showing the customer their changes continually.

Instead of developing several functional prototypes the startup developed a more generalized solution which was more complex and time consuming. This looks more like the predictive process, explained in (figure 1 in 2.1.1), compared to several incremental processes.
Since the startup has been doing product development for 5 years and have defined important customer values, it seems probable that they have shown and have tested several prototypes with customers. The startup is therefore probably doing incremental development to some degree without knowing it.

6.1.6 Collective analysis and discussion

**Increment = prototype?**

It seems like the case studies use the word increment and prototype as the same thing. Prototypes seem to be categorized as a both a customer prototype and a technical prototype.

The customer prototype is by some of the startups also called a demonstrator, and is by definition also an increment if it’s shown to a customer. When asked if the technical prototypes could have been shown to the customer, all the interviewees agree. The reason the technical prototypes were not shown to customer is because they were not considered technically sufficient.

For this reason it seems like none of the startups really know the definition of the words they use, and that they are not consistent in the words they use.

**A different understanding of the words prototype, iteration and increment could create problems**

Not having the same understanding of the word prototype, iteration and increment could create problems when communicating the goals and progress of the company to team members, investors, suppliers and other stakeholders. This could also create problems when using different frameworks, where some frameworks advocate only making increments, like the MVP in The Lean Startup, or the Scrum frameworks which proposes to do prototyping (iterations) on increments (prototypes).

It makes sense that the different frameworks use different definitions of the word iteration and increment, as they are all specialized versions of the Agile approach. As we found out in the case studies, it seems like there is no common understanding of what an iteration is. For further research, it therefore would make sense to ask specific iteration questions related to the framework they are currently using. At the same time, the interview objects did not know what the name of the framework they used was, and they also used some frameworks in combination with each other. This could create problems when onboarding new team members if the company has not documented their custom framework well enough.
Startups are Agile to some degree when frequent prototypes are shown to customers

It seems like all the startups value iterative development, since they all build several versions of their products. If they show some or all of the prototypes to the customer, that means they also do incremental development. This could indicate that startups, who do frequent prototyping and get feedback on these prototypes with the customer, use some form of iterative and incremental development, and can therefore be considered as Agile to some degree.

Increments are too expensive and are therefore not considered as a viable development technique

The startups’ lack of understanding of an increment could indicate that an increment of a product often is too expensive, even though literature sometimes claims it not to be. (Smith, 2007) explains that a blend of iteration and incremental development is most the effective, where the choice depends on the companies economic resources. “If the cost of a release is not too great and customers gain value from it, an incremental release has advantages, in part because it shortens time-to-market by reducing development effort.” (Smith, 2007)

One could also point out the fact that the mere development of a hardware product is more time consuming than that of a software program. According to Schuh (et al., 2016), presenting a functioning prototype is much more difficult for hardware as it cannot be partitioned into infinite increments. Also, the decision making can be assumed to involve more stakeholders, because of the higher costs for each prototype. This cost seems very typical for physical products: for a software program most of the components can be built by the whole team of programmers, while a physical product usually consists of components involving several (engineering) disciplines. Many of these components must be bought from suppliers, which in turn increase the development time due to the extra factor of supplier's planning, construct and shipping. For software, the quick and cheap designs results in more frequent increments, which again is easily tested using software's unique super power of automatic testing.

6.2 Analysis and discussion of key finding 2: The length of an iteration seems to vary to a large degree between the case studies

In the following sub-chapter, key finding 2 is discussed collectively and all four cases are considered. This key finding will only be discussed collectively as the researchers want to illuminate the difference and similarities between the cases. It will be discussed how the
length of an iteration for hardware startups is related to existing literature. As with key finding 1, relevant theory will be presented first.

6.2.1 Theory
An iteration is often a time-boxed (fixed time slot) set of activities, lasting from a week to several weeks (Smith, 2007, Meyer, 2014). If time-boxing is used, the iteration is finished within this time-box or terminated prematurely. This also means that if time-boxing is used the fixed time slot cannot be changed when an iteration is on-going. The opposite of time-boxed iterations are iterations with variable length, where the length of an iteration can be changed when an iteration is on-going.

The Agile frameworks presented are not all time-boxed, but all focus on speed.

In Scrum, an iteration (or sprint) is time-boxed, and is finished on the planned date, and not “when it’s ready”. The (time-boxed) iteration can in rare cases be terminated prematurely to start with a new iteration. A sprint in software typically last 1-4 weeks and each sprint can contain shorter iterations of for example 1 week. (Schwaber, 2004).

The Lean Startup cycles (or iterations) are not time-boxed, but Ries (2007) emphasises that “the faster we go around the build-measure-learn cycle, the faster we can validate our business model.” It is also worth noting that a Lean startup iteration is the same as an increment.

Kanban can be used with time-boxing, but does not advocate it. Kanban focuses on minimize work-in progress by ensuring just-in-time production, driven by demand. (Meyer, 2014)

Design thinking emphasize high speed and rapid prototyping Plattner et al. (2010)

In Flexible product development iterations are often time-boxed, where one time slot can last from a week to several weeks. (Smith, 2007)

6.2.2 Collective analysis and discussion
The time used on each iteration seems to be longer for all case studies compared with what is common in software development. Based on the case studies the reason for this could be:

- The procurement of many different parts for prototyping, including waiting for suppliers to deliver (the right) parts. Sometimes parts need to be procured from a foreign supplier which increases shipping time.
Planning with physical product development is bound with high uncertainty when the development team is not experienced or uses a framework.

- The decision-making process can be assumed to involve more stakeholders, because of the higher costs for each prototype.

- For a software program most of the components can be built by the whole team of programmers which creates less delays in the development, while a physical product usually consists of components involving several (engineering) disciplines which means waiting could occur more easily.

- Testing is easier for software with automated unit test, whereas hardware testing too some degree is subjectively measured. Hardware stress tests also require testing over a longer time period to ensure physical quality. In software, stress test can be automated and simulated.

The effect of time delays in development means that iterations sometimes need to be paused, and that the next iterations needs to be started before the previous iteration is done. Overlapping therefore occurs and can create a planning problem. The case studies already present difficulties planning the length of an iteration and an overlapping of iterations makes both time-boxing and keeping to practice of iteration harder.

As presented in section 2.1, by (Dombrowski et al., 2011), the way product development is done is dependent on such factors as the product, the organization and the available resources of the company. If these factors are taking into consideration in the different phases of product development, time spent and resources used, can also vary a lot. It seems especially apparent that the development of complex physical products seems to take a long time.

6.3 Analysis and discussion of key finding 3: The frequency of testing for hardware startups is lower compared to the Agile Manifesto, and there is large variation between different hardware startups

In this section, key finding 3 will be discussed. All four cases will first be discussed separately with relation to the research questions, and then collectively in the end. This key finding is closely related to key finding 2, which is found to be a major factor adding to the low testing frequency of the four cases.

6.3.1 Robotics

Robotics are the hardware startup with the most frequent testing. They use frequent iterations, and try to test as often as possible. Some could rise the question why Robotics are not able to
fulfil the Agile Manifesto's encouragement of daily testing. One thing is their argument that some tests are difficult to make in advance. This seems reasonable, as a physical object has so many variables which could be tested. Also, size and shape will affect the results, so different equipment and test procedures might be needed every time. This is in contrast to software, where every test is non-physical, and can be performed with just the click of a button. Another reason is that their prototype needs space to be tested, also unlike software which need no such thing. Lastly, the use of 3D-printing can speed up the process for development of new and complicated components, but it will take time print. With this method, the startup can produce its own components instead of having to wait for suppliers or the like. The authors have no data to compare this with the development time of different software codes, but it would seem likely that development of even software 'components' can be time consuming.

6.3.2 Wearables
Wearables' testing frequency could be explained by the fact that they only have two product developers. With one software developer and one hardware developer, each increment of the unified prototype will take more time than if they worked separately. The CEO points out that testing of prototypes requires equipment and resources, which not always is available. Even though part of the product is software, and that this part can be tested more often, the testing of the main product requires human test objects and real feedback, thus, more complicated. The product developer in Wearables highlights the fact that, sometimes, the purpose of prototypes is just building them. The reason for this could be that several mistakes can, and probably will, occur while developing, and this way the builders are trained to avoid them. This is probably the same for software developers, but there are differences: 1) software developers can easily test codes continuously and thus discover mistakes earlier, and 2) if there is an error, reconstruction of the code is much cheaper than building a new physical product. When noting this latter fact, it is easy to understand the need of this 'experimentation phase' for hardware startups. If something goes wrong with the 'real' prototype (where for example very expensive materials are used), the consequences could be dramatic. Thus, it could seem as if this resource demanding development of physical products also requires a resource demanding 'experimentation phase'.

6.3.3 Connected Devices
In principle, the testing for Connected Devices' product should not need to take much time. This is reasoned with their claim of few and easy tests of only some parameters. The reason
might be that some of their components take time to develop. When the development is time consuming, this will of course influence the testing frequency. Further, they do not work more than about 15 hours each week per person in average, and rarely work together. With a changing focus and little communication, important things like testing might be forgotten or postponed for unknown reasons. Also, with a customer (and end-user) abroad, it is more difficult to get feedback on new solutions. This way, even though a prototype is ready for customer testing - or technical testing in customer environment - it might take alot more time than expected.

6.3.4 Designed Products

As the most mature hardware startup, Designed Products already has a product for sale. This means that the requirement for quality is very high, and thus that every new component must be developed with great caution. From this one could imagine that a new prototype at the stage of Designed Products takes more time to develop than a new prototype for an early phase startup. Most of the components used in the product is acquired through suppliers, which is time consuming. This highlights another aspect; where the 'components' of software can all be built by the software developers, hardware requires several disciplines and sometimes production facilitiess. Finally, there are tests regarding, for example, durability. A software program would have no such issue as there are no physical components to be exhausted. This is another factor adding to the time delay for development, and thus, testing.

6.3.5 Collective Analysis and discussion

The methods of agile methodology consider tests as a crucial part of every project, and urges no implementation to take place without having a test for it (Meyer, 2014). This fits the Wearables claim that every prototype should have a purpose and always be tested, though they emphasize that this can be very difficult sometimes. Even the Robotics company, who tests very often, find this issue somewhat difficult. The CTO explains how difficult it could be to create a test for a feature before you build it. The reason is that some things might be completely new, and you might need several attempts to iterate something useful.

From the table in key finding 3 it is clear that Robotics is the hardware startup with the highest testing frequency, and Connected Devices is the one with the least frequent testing. There can be several reasons for this. First of all, most of Robotics components are 3D-printed which could save them alot of time. On the other hand, 3D-printing a component can still take som time, and errors can occur. Secondly, the amount of working hours performed by Connected Devices is much less than that of Robotics and is also split between more product
developers. With a more fragmented working day, it would not seem strange that the focus of the team as a whole would drop, leading to less productive work. Thirdly, the customer of Connected Device is located abroad, opposed to Robotics' test-users who are much easier to get in touch with. A delayed user or customer feedback could result in delayed iterations, and thus testing.

There are some significant factors connecting Robotics and Connected Devices to opposite sides of the testing frequency scale. This seems not to be linked directly to the subcategories, but rather the organized structure. One could claim that based on this research, Robotics usually do more 3D-printing than other startups, giving them an edge in the case of frequent testing, but it seems strange that the 3D-printing effect should be that significant alone. Within all the subcategories, products will vary, and the teams developing them will utilize different methodologies.

Each of the four cases has revealed some interesting reasons behind the testing frequency, and they all seem related to the physical aspects of the hardware product. Here, we might be in the core of the 'problems' related to hardware product development. It seems as if the low frequency of testing for hardware startups, compared to the frequency of daily builds suggested by the Agile Manifesto, might be the fundamental difference between hardware and software products. In software, testing can be done continuously with tools that can automatically test thousands of defined tests within a short period of time. This enables them to test several times each day if needed, without the need of any equipment or extra resources (Smith, 2007). This is opposed to hardware, which requires physical testing equipment to perform the different tests. The problem of lack of equipment, as mentioned by Wearables, is also emphasized by Schuh (et.al, 2016) claiming that there exists availability problems of tools and other production equipment. In addition, the mere building of a physical prototype is time consuming and expensive, and one should expect to need several attempts. Another important issue lies in the fact that all software can be written by the software developer, while development of a physical product requires several disciplines. The hardware team will thus be more specialized, making shared ownership - one of the Agile practices - more difficult. In addition, some components can not be created by the startup, but must be manufactured by suppliers. This also shed a light on the importance for hardware startups to not only integrate the customer, but also the supplier.

The reasons mentioned above is closely related to key finding 2. A result of the fundamental difference between software and hardware products will lead to longer development time, and
thus iterations, which again will postpone the testing. This is a major factor adding to the
general problems of testing physical problems, as discussed above, leading to a lower testing
frequency than proposed by the Agile Manifesto.

6.4 Analysis and discussion of key finding 4: Hardware startups mix concepts from
different Agile methodologies

In the following chapter, key finding 4 will be discussed. This key finding is a direct answer
to research question 1. The discussion will elaborate on the fact that all cases seems to use
characteristics from several different Agile frameworks. All cases will be discussed separately
before they are discussed collectively. The majority of the discussions content will be
presented in the last subchapter.

6.4.1 Robotics

Seems to follow the Scrum approach in several aspects, but do not have daily meetings.
Ovesen & Sommer (2015) states that Scrum is an additional framework that is only utilized to
some extent. As with all cases, Robotics utilize aspects of several other Agile frameworks as
well.

6.4.2 Wearables

Their misunderstanding of Agile as an opponent of time limits could be a result of their own
misconception of the Agile methodology, or the fact that they have not read it in details. This
is understandable, as the Agile approach in fact wants to be open for changes. In this sense,
time-boxing would seem like a limiting factor. This is comparable to another misconception;
that Agile - with its self-organizing teams - has no leaders (Meyer, 2014). This highlights the
fact that there are several aspects of Agile Methodology that seems illogical, and that deeper
understanding of the methodology is needed to fully practice it to its full extent.

6.4.3 Connected Devices

It seems clear from both interviews that Connected Devices has a perception of themselves as
following the Design Thinking approach. The most eye-catching result, is their claim to put
customer in center, which they sort of do. All requirements are based on customer
requirements, but the frequency of customer and user interaction is very low. This might have
to do with the fact that their customer and users are settled in another country.

6.4.4 Designed Products

Designed Products claim not to use any known methodology, but still use several aspects of
the Agile approach. A clear difference from the other cases is how long they have been doing
product development - five years opposed to less than two years for the other cases - which for the authors would mean that they should have a better understanding of what methodology they use or not. On the other hand, a mature startup might not have the need of defining their methodology.

6.4.5 Collective analysis and discussion
It would seem as if all the cases utilize aspects from all the seven frameworks to some degree. Particularly visible are the Scrum roles of Product Owner and Scrum Master, as well as the usage of online tools as Trello or Taiga for visualization of tasks - associated with Kanban. In addition, incremental development, rapid prototyping, stakeholder interaction, focus on team collaboration and communication, self-organizing teams and a continuous adaption of plans, are all Agile characteristics that every case practice to some extent. To what degree each case actually practice these characteristics vary among the cases.

The authors have tried to rate each case for each characteristic. This is based on their impression after interviewing and analysing the different cases, together with a deep analysis of the Agile Manifesto, with its values, principles, practices, artifacts and roles. The result is visualized in table 10. The table is identical to that of table 9, except from the numbers in each cell. Instead of a binary division, each cell receives a number describing to what degree these characteristics are utilized. The higher the number, the more the characteristic is utilized (zero being the lowest and two being the highest). It should be noted that the rating also is a relative measurement between the cases.

Table 10: The researchers analysis of how the hardware startups utilize the different characteristics.

<table>
<thead>
<tr>
<th></th>
<th>Robotics</th>
<th>Wearables</th>
<th>Connected Devices</th>
<th>Designed Products</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterations</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Incremental</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Visualization</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Rapid prototyping (MVP)</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Stakeholder interaction</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
Focus on team collaboration and communication | 2 | 1 | 2 | 2
Self-organizing | 1 | 1 | 1 | 1
Freeze requirements | 1 | 1 | 0 | 0
Limit WIP | 0 | 1 | 0 | 0
Adaption of plan | 2 | 2 | 2 | 2

From this, Robotics seem to be more Agile than all the other cases, with Connected Devices as the least Agile hardware startup. All four cases get top score for *incremental development* and *adaption of plans*, and in general an overall Agile approach.

Ovesen & Sommer (2015)'s statement that Scrum is an additional framework that is only utilized to some extent seems to be the case with all the other Agile frameworks too, when utilized by hardware startups. The reason for this strange fact - that all these frameworks are mixed - could be that the startups consciously have chosen to use only parts of different frameworks. This seems not to be the case, though, as there are no data from the cases backing this claim. A more obvious reason is the fact that all the frameworks are very similar to each other. The four frameworks mentioned among the cases - Scrum, Lean Startup, Design Thinking and Kanban - all share very similar characteristics. The precise definition of some of these terms do somewhat differ from literature to public understanding, which is discussed in key finding 1. This confusion of concepts might also be a big reason why the cases mix up the different frameworks.

All of the cases also seem to have characteristics in common with both Extreme Programming and Feature Driven Development - which are both software methodologies (especially FDD) - and also, the more natural Flexible Product Development. This methodology was popularized by Smith (2007), stating that FPD could be seen as the Agile Product Development for physical products. In addition to possess most of the core Agile characteristics, FPD fits all team sizes. It could seem strange, though, that such an applicable Agile hardware framework is not better known. But, for this to happen, hardware startups must learn it, and to learn it you need people to teach you. Considering the difficulty of discovering literature regarding the field of Agile methodology for physical product development (Stare, 2014) this suddenly do
not seem as strange. One would imagine that startups have better things to do than use their time roaming the internet for methodologies that might fit their company.

Considering that all cases are hardware startups, that FPD is an Agile framework made especially for hardware companies, and that the cases as a group has more in common with FPD than any other Agile framework investigated, it is very tempting for the authors to suggest FPD as the unification of Agile methodologies for hardware startups. There are of course some aspects not adopted by FPD, like Kanban's minimization of work in progress or Scrum's daily meetings, which could justify why hardware startups might adopt practices from several frameworks. Still, there are no restrictions preventing these characteristics to be adopted as part of the FPD methodology.

From the four hardware startups investigated, there are significant evidence that Agile frameworks are used for their development of physical products. All cases show traces of every Agile framework considered in this thesis, pointing to the fact that no single framework covers all hardware startup needs. The framework covering most Agile characteristics practiced by the cases evaluated are Flexible Product Development.
7. Conclusion

The authors of this master thesis have investigated how the Agile approach can be used by startups in the development of physical products. This has been done by researching if hardware startups use Agile frameworks and how this related relates to theory.

The literature review done before the case studies looked at the differences and similarities between Agile Software Development and Agile Hardware Development and discovered which Agile frameworks could be used for hardware product development. The link between the two fields is weak or non-existing considering literature, but the literature review has shown that Agile Product Development can be used by hardware startups for physical product development.

By investigating if startups use Agile frameworks for physical product development (RQ 1), the authors found that all the hardware startups utilize Agile values, principles, practices, artifacts and roles to some degree. On the other hand, no single framework is used by any of the cases since they all mix existing Agile frameworks. This is done both through customizing existing frameworks to fit the specific cases, and through not using any specific framework - but borrowing concept from several frameworks. This confirms both Smith (2007) and (Rigby et al., 2016b).

By investigating how the use of Agile frameworks is different from what is described in literature (RQ 2), the authors found that there is a distinct difference between Agile Software Development and Agile Product Development, and that there is a fundamental difference between hardware and software products. The following findings from the case studies show that:

- Iterations are much longer for hardware startups, with lengths varying from 1 week to 6 months. In addition, time-boxing is not practiced.
- Making an increment for each iteration is not possible, and is restricted by the nature of the physical product and the resources available.
- Daily builds (daily testing) - as proposed by the Agile Manifesto - are very difficult. A result is a much lower testing frequency.

The findings of this thesis are summarized in table 11 below, and shows the Agile characteristics for hardware relative to Agile software characteristics, where the blue boxes show the literature findings and the yellow boxes show the findings from the case studies.
Finding 1-3 and 7 can be seen as a theoretical contribution specific to Agile hardware startups, and finding 4-6 can be seen as a contribution to existing theory, although it is specific for startups.

**Table 11: Agile characteristics for hardware relative to Agile software**

<table>
<thead>
<tr>
<th></th>
<th>Software</th>
<th>Hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Frequency of testing</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>2. Cost of testing</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>3. Cost of prototyping</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>4. Cost of iteration</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>5. Ease of tools and equipment</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>6. Ease of increments</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>7. Ease of stakeholder involvement</td>
<td>High</td>
<td>Low</td>
</tr>
</tbody>
</table>

In general, the case studies in this thesis also confirm Smith (2007), who states that the Agile principles which apply to software, as well as hardware, are iterative and incremental development (with customer feedback), self-organizing teams, and emergent processes. Flexible Product Development, by Smith (2007), also seems to be the framework which is most used among the cases, and the one which is considered the least both by startups and in theory.

7.1 Theoretical implications

As of implications to theory, the authors have discovered several aspects of contribution. This thesis is contributing to theory by linking Agile Software Development to Agile Hardware Development, which, according to Stare (2014), has not yet been established.

The case study findings confirm Schuh et al. (2016)'s claims regarding the challenges of using Agile for physical products.

Agile roles - including team, Product Owner, Scrum Master and customer - must also include supplier, when used by hardware startups, since this is an integral part of the product development. This is an addition to Meyer (2014) core characteristics of Agile.

The authors have recognized rapid prototyping tools such as 3D printers as a new benefit, which makes iterative and incremental Agile Hardware Development easier, which confirms
Vetter (2011), but also recognize the challenge Schuh et al. (2016) presents. Not all tools needed for making a functional prototype are as rapid and available as 3D-printers.

Furthermore, the qualitative nature of this study would require some additional research before the findings could contribute to the gap between Agile Software Development and Agile Hardware Development.

7.2 Practical implications

For practical implications, the authors propose several actions for hardware startups.

Since costs of changes are much higher for physical product development than for software development, and since a startup with an unexperienced team has a higher probability for failure, a hardware startup should integrate mentors and experts to get enable rapid feedback and shorter feedback in their development cycles.

Further, the hardware startup should focus on always building the simplest prototypes as possible to reduce the resources and time spent on an iteration or increment. This can be done by choosing for example simple manufacturing methods, such as 3D-printing, or cheaper materials.

For defining what constitutes a ‘done’ sprint, the startup should always define the purpose of the iteration and define what to test before the prototype is built. This is make sure that the startup does not waste an iteration cycle.

Essential for incremental development is good customer and user relationship with regular feedback, preferably before the entire increment is built. This will ensure rapid feedback which will shorten the iteration cycles. Managing the customer and users according to the chosen Agile frameworks is essential. In addition, the integration of suppliers in the product development could be crucial. They can suggest better and cheaper manufacturing methods, and a close relationship could result in more rapid deliveries of parts.

All hardware startups must be consistent, both internally and externally, with their definitions of specific Agile terms. For example the team needs to clearly define iteration, increment and prototype. If team member has a different perception of these terms, it could create problems with communication. In addition, adoption of framework practices could be difficult if framework characteristics and suggestions are misunderstood. Documentation of the custom frameworks could be a way to do it. If Agile frameworks are going to be used, all team members must on board and understands the methodology.
7.3 Limitations

From this research, several limitations could be pointed out. First, all hardware startups where chosen out of a list of criterions. This made the cases slightly homogeneous in the sense of control variables, meaning that the results do not include data for hardware startups outside this focus. Secondly, only four hardware startups are covered. This could be a representative number for an exploratory study like this, but might be limiting by not including enough data to generalize for all hardware startups. Thirdly, the authors have investigated a very broad topic concerning all characteristics of the Agile Manifesto. A thorough analysis would require more data on every case, but this would require much more time which limited the data acquisition process. Thus, there were topics the authors did not have time to investigate further. Also, the interview guide could be improved to get more specific answers related more to the research questions, although the study was exploratory.

7.4 Further research

For further studies regarding using the Agile approach for physical product development, the authors recommend examining the following four suggestions.

1) A more extensive quantitative research is recommended to be able generalize the findings in this paper. The broad theoretical gap between traditional approaches and the Agile approach in physical product development is worth examining, but also the differences between Agile Software Development and Agile hardware development. The findings from this paper and the interview guide in the appendix could act as a foundation for a survey which could cover a large data sample of specific companies.

2) There should be conducted a qualitative study similar to this study with a broader range of companies relative to the technology R&D spectrum presented in 3.4.3, since this study mostly considered relatively simple consumer faced products. Long-lead heavy R&D projects were not considered. Considering a different technology focus for further studies, only analysing companies who do purely hardware development would also be interesting. This would make it possible to understand how Agile works in the broad spectre of physical products and how the absence of software affects their Agile approach.

3) A similar qualitative study could be conducted with companies who have one customer focus. The interviewed companies were operating in both B2B and B2C markets, and showed differing interaction patterns with their customers, end-users and suppliers. A population of companies who only serve B2C markets and have frequent end-user contact could be
interesting. The authors question whether the findings in this paper are applicable for both companies focusing on B2C markets with frequent end-user contact and at the same to also for companies focusing on B2B markets with almost no end-user contact.

4) Finally, the authors would recommend a research to investigate to which degree Agile methodology and frameworks leads to higher or lower success rate among hardware startups, compared to other methodologies. This could help to clarify if, and in what settings, the Agile Approach should be recommended.


PRATT, M. G. 2009. From the editors: For the lack of a boilerplate: Tips on writing up (and reviewing) qualitative research. *Academy of Management Journal*, 52, 856-862.


STATT, N. 2015. For hardware startups, it’s getting less hard. *CNET*.


Appendix

A. Interview guide

A.1 Interview guide questions

The interview questions in the interview guide (at the end) are mostly based on two things: the characteristics from the Agile Manifesto, as proposed by (Meyer, 2014) and the Agile frameworks for physical product development from the theory chapter. Additional question (in the end) were added for clarification or categorization of the questions already asked. The reference column is added to show which questions partly overlap. This can be used to reduce the time of an interview.

A.2 Categorization and clarification of the interview guide

After transcribing the interviews some data was missing. These questions were asked again after the first initial interviews. For writing the case studies, a categorization was used. A cross-reference table was used to categorize questions and are presented in table A2 below.

Table A1: Categorization of questions

<table>
<thead>
<tr>
<th>Intro</th>
<th>Requirements</th>
<th>Planning</th>
<th>Building</th>
<th>After Math</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status regarding prototypes. How many built, etc. Also, has things changed?</td>
<td>When building a prototype, you need requirements. Where do you get this? Up-front. Most important feature. Customer</td>
<td>Now you have the requirements. Before the prototype is built, this must be planned. Do you use iterations? Time-boxing? Are you going to visualize progress? Will you limit change? Etc.</td>
<td>This refer to “during prototyping”. What rules do you follow here? Do you use standard components? Always build 2 at a time? Where do you do this?</td>
<td>How do you make sure you learn anything from these prototypes? Do you do any testing? How do you know that it is done, at all?</td>
</tr>
<tr>
<td>10</td>
<td>4, 5, 6, 14, 20, 21</td>
<td>11, 12, 38, 56, 73, 74, 59, 60</td>
<td>70, 67, 68</td>
<td>57, 15, 16, 17, 18, 72</td>
</tr>
</tbody>
</table>
A.2.1 Structure of the questions
Because some of the questions overlap in terms of what answers the interview subjects would give, the authors recommend to structure the questions with relevant topics to avoid redundant data collection and lengthy interviews. This master thesis used the structure shown in table A1 below. To use this structure, the researchers should know the original questions by heart.

Tabell A2: Structure of interview questions

<table>
<thead>
<tr>
<th>Topic</th>
<th>Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organizational questions</td>
<td>76, 47, 64, 49, 50, 51, 2, 3, 23, 27, 28, 55</td>
</tr>
<tr>
<td>Iteration questions</td>
<td>10, 11, 12, 38, 56, 57</td>
</tr>
<tr>
<td>Prototyping questions</td>
<td>4, 5, 6, 14, 19, 20, 21, 15, 16, 17, 18</td>
</tr>
<tr>
<td>Detailed short questions</td>
<td>72, 73, 74, 70, 67, 59, 60, 46, 68</td>
</tr>
</tbody>
</table>

B. Agile Manifesto – values and principles
Agile development as a term was coined by several leading experts in the field of software development and is elaborated in the Agile Manifesto, through a set of values and principles shown below.

The whole (original) manifesto and corresponding principles are cited below²:

Manifesto for Agile Software Development
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.

Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave

---

### Agile Manifesto Principles

*We follow these 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 behaviour accordingly.
<table>
<thead>
<tr>
<th>Values</th>
<th>Questions</th>
<th>Answers</th>
<th>Ref</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A) Redefine roles</strong></td>
<td>Explanation: Limit managers role by transferring duties to the team. Especially tasks to be performed.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1) What is the role of the various positions in your company?</td>
<td>Team and roles</td>
<td>N1</td>
</tr>
<tr>
<td>2</td>
<td>2) Who decides what needs to be done? Explain</td>
<td>Manager or team/individuals?</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3) How much of the responsibilities are transferred to the team? How much does the team decide (vs individual)?</td>
<td>Self-organized</td>
<td></td>
</tr>
</tbody>
</table>

| **B) Rejection of Big Upfront** | Explanation: - Requirements can not be captured in the beginning of a project, since the customers do not know what they want. If you can get some requirements, they will surely change. - Building a design up front is a waste of time, because we do not know what will work and not. - Instead of a requirements list, Agile require customer interaction. | | |
| 4 | 1) When you build a new prototype, how do you know what you are going to build? | Requirements from customer | [Simple mvp] |
| 5 | 2) How much of the requirements are figured out before/during prototyping? | Big upfront? |
| 6 | 3) How do you figure out these requirements? | Customer interaction |
| 7 | 4) How involved are you with your customer? | Customer interaction | F2 |
| 8 | 5) How often do you interact with your customer, and what do you get out of it? | Customer interaction frequency | F2 |
| 9 | 6) Typically how long time do you use on one prototype? | Iterations | C2 |

| **C) Iterative development (and time-boxed)** | - At the beginning of each iteration, a list of functionalities is set. - From this the team will select implementations with the highest Return on investment (ROI). | | |
| 10 | 1) How many prototypes have you made? | Due diligence |
| 11 | 2) Do you use iterations? How often? How long is each iteration? | Iterations |
| 12 | 3) Bruker dere tidsbokser / deadlines? | Time-boxed, y/n |

| **D) Limited, negotiated functionality** | - Limit features to the most important ones, as measured by their business value (the ROI). - Waste minimization is a core concern. | | |
| 13 | 1) How do you decide what features to implement in your prototype? | Most important features | B5 |
| 14 | 2) Do you have method to keep track of which features are more (or less) important to implement than others? How? | Most important features |

<p>| <strong>E) Focus on quality</strong> | - Continuous testing - Do not continue before all tests are passed. | | |
| 15 | 1) How do you learn from your prototypes? | Testing |
| 16 | 2) When are these tests planned? And how do you perform them? How often? | Testing |
| 17 | 3) Do you implement features which you do not have a test for? | Focus on testing |
| 18 | 4) Do you have a policy to define which classes of non-passing tests that preclude new development, and which are acceptable? | All tests passed |</p>
<table>
<thead>
<tr>
<th>Principles</th>
<th>Questions</th>
<th>Answers</th>
<th>Ref</th>
</tr>
</thead>
</table>
| Organizational | - Invite customers to regular meetings  
- Embedd customer in the team  
- Do not put effort into writing upfront requirements | | |
| F) Customer at the center | 1) Do you have a customer? Tell us about them | [Customer focus] [Due-diligence] | |
| | 2) How often do you interact with your customer? How are they involved in PD? | [Customer interaction] [Customer interaction frequency] | |
| | 3) How do this feedback influence your product development? | [Requirements during product development] | |
| G) Self-organizing team | - Managers do not assign tasks to developers.  
- Scrum is particularly systematic in this respect.  
- Manager encourage initiative from the team members and gradually move the team to a self-organized mode. | | |
| | 1) Who decides what needs to be done? | [Self-organized] A1 | |
| | 2a) How ofte møter teamet? | [Regular meetings] | |
| | 2b) Hva diskuterer dere da? | | |
| | 3) How free are individuals to decide their own work tasks? | [Self-organized] A4 | |
| | 4) Do you use any frameworks in your company? If so, which/how? | [Scrum] [RO3] X | |
| H) Work at at sustainable pace | - Good working conditions  
- Advised to open spaces rather than cubicles  
- Personal and honest feedback. | | |
| | 1) Describe your working environment | [Working environment] O4 | |
| | 2) Do you have regular feedback? | [Feedback] | |
| | 3) Is it usual to give personal feedback (criticism)? | [Honest feedback] | |
| | 4) Do you work in open spaces or cubicles? How is this working out? | [Open spaces] Y5 | |
| I) Produce mvp / only what requested | - If you can decide between several functions, include only what is absolutely needed.  
- What is the simplest thing that can possibly work?  
- Worry only about what is needed here and now. | | |
| | 1) How do you decide what functions/features to include in your product development? | [List of ROI features] [Mvp] B5 | |
| | 2) How do you decide what is most needed? | [List of ROI features] [Mvp] D2 | |
| | 3) Do you create only what you know is needed, or do you make hypothesis of what is needed and implement them as well? | [Only what requested] | |
| | 4) Do you focus on simple or complex prototypes? | [Iterative] C2 | |
### J) Accept change

- In reality this means 'limiting change'. In Scrum, no-one can add or change product requirements during a project development phase (sprint).
- The rule that "every piece of functionality should have an associated test case" especially applies to software (authors remark), since a change here can ruin the whole code.

### Technical

#### K) Develop iteratively (Frequent iterations + freeze requirements)

- Iterations should be short, typically a few weeks. (Needs definition of done)
- Iterations should be frequent!
- Time-boxing. Functionality goes before a deadline.
- Functionality can only be added in sprint planning phase (closed-window rule)
- Highest business value first!

### M) Express requirements through scenarios (use-case)

- A user story describes an elementary unit of interaction. "As a [role], I want to [action] so that [goal]".
- The principle of using scenarios for requirements specification is one of the most widely practiced agile concepts. (Note: this can be dangerous)

### 34

1) Did you use much time to decide requirements before building your prototype? How much?  

<table>
<thead>
<tr>
<th>[%]</th>
<th>B6</th>
</tr>
</thead>
</table>

### 35

1) How do you think this could be done faster?  

<table>
<thead>
<tr>
<th>[%]</th>
<th>[%]</th>
</tr>
</thead>
</table>

### 36

2) Do you use iterations (what about sprints)?  

<table>
<thead>
<tr>
<th>[%]</th>
<th>[%]</th>
</tr>
</thead>
</table>

### 37

3) How frequent are these iterations? (How many prototypes have you made?)  

<table>
<thead>
<tr>
<th>[%]</th>
<th>[%]</th>
</tr>
</thead>
</table>

### 38

4) Do you limit change (freeze requirements) during iterations?  

<table>
<thead>
<tr>
<th>[%]</th>
<th>[%]</th>
</tr>
</thead>
</table>

### 39

1) How often do you usually make prototypes?  

<table>
<thead>
<tr>
<th>[%]</th>
<th>[%]</th>
</tr>
</thead>
</table>

### 40

2) Do you use iterations? How long/short are the time-frames?  

<table>
<thead>
<tr>
<th>[%]</th>
<th>[%]</th>
</tr>
</thead>
</table>

### 41

3) How do you plan what these 'iterations' should include? / Do you have any rules of when it is ok to add requirements?  

<table>
<thead>
<tr>
<th>[%]</th>
<th>[%]</th>
</tr>
</thead>
</table>

### 42

4) How do you decide what to build? How do you know this is the most important?  

<table>
<thead>
<tr>
<th>[%]</th>
<th>[%]</th>
</tr>
</thead>
</table>

### 43

1) How do you make sure you learn something from each prototype?  

<table>
<thead>
<tr>
<th>[%]</th>
<th>[%]</th>
</tr>
</thead>
</table>

### 44

2) When (in the process) do you decide what you are going to test?  

<table>
<thead>
<tr>
<th>[%]</th>
<th>[%]</th>
</tr>
</thead>
</table>

### 45

3) Do you have a policy to define which classes of non-passing tests preclude new development, and which are acceptable?  

<table>
<thead>
<tr>
<th>[%]</th>
<th>[%]</th>
</tr>
</thead>
</table>

### 46

1) Do you ever use some sort of scenario-based user cases to understand the customer (and their needs) better? I think this is very software  

<table>
<thead>
<tr>
<th>[%]</th>
<th>[%]</th>
</tr>
</thead>
</table>
### N) Roles

**Manager:**
- Do not assign tasks
- Do not decide what functions to implement
- Do not direct the work of team members
- Do no request status report
- Establish an environment that enables team to work successfully.
- Ensure smooth interaction with the rest of the organization

**Product Owner:**
- Has the final say over the product and what things to implement.
- Responsible of product backlog: list of features (what to implement)
- Evaluate results at the end.

**Scrum Master:**
- Enforcing agile rules
- Remove impediments identified by team members
- Protect team from distractions

**Members and observers:**
- Members should dominate the discussion, observers standing on the side.

**Team:**
- Teams in Agile are self-organizing
- Cross functional

### Separating roles:
- In general, two tasks are needed:
  1) Directing project, day after day.
  2) Defining what it must do for the business, and assessing whether it actually does it.

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>47</td>
<td>1) What are the different roles in your company, and what are their responsibilities?</td>
</tr>
<tr>
<td>48</td>
<td>2) Do you have any self-made roles?</td>
</tr>
<tr>
<td>49</td>
<td>3) If Scrum, Do you have a Product owner? (See description)</td>
</tr>
<tr>
<td>50</td>
<td>4) If Scrum, do you have a Scrum-Master?</td>
</tr>
<tr>
<td>51</td>
<td>5) Are some of the team-members more committed than others?</td>
</tr>
</tbody>
</table>

### O) General questions regarding Practices

- The following questions are not mentioned on Meyers sheet of Agile Practices.

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>52</td>
<td>1) Do you use sprints (iterations)?</td>
</tr>
<tr>
<td>53</td>
<td>2) Do you freeze requirements during iterations?</td>
</tr>
<tr>
<td>54</td>
<td>3) Do you involve customers in your product development?</td>
</tr>
</tbody>
</table>

109
## Managerial

### O) Daily meetings
- Short. Fifteen minutes is the standard.
- What did you do on the previous day? What will you do today? Any impediments?

<table>
<thead>
<tr>
<th>58</th>
<th>How often do you have meetings, and what do you discuss?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[Daily meetings] [Feedback]</td>
</tr>
<tr>
<td>G2</td>
<td></td>
</tr>
</tbody>
</table>

### P) Planning game
- Figuring out how much time an implementation will take.
- Customer defines the priority of a set of functionalities. Develops estimate the cost (person-days) of implementation.

<table>
<thead>
<tr>
<th>59</th>
<th>1) Do you use any resources calculating cost/time of an implementation?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[Planning game]</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| 60 | 2) Do you involve customers in this estimation?                         |
|    | [Planning game]                                                        |

### Q) Continuous integration / daily builds
- Integrate and test as often as possible

<table>
<thead>
<tr>
<th>61</th>
<th>1) How often do you test/do an overview of what has been built?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[Daily builds]</td>
</tr>
<tr>
<td>E2</td>
<td></td>
</tr>
</tbody>
</table>

### R) Retrospective
- What went well and less well in the last sprint?

<table>
<thead>
<tr>
<th>62</th>
<th>1) If iterations sprint; how do you make sure you learn something from the iteration?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[Feedback]</td>
</tr>
<tr>
<td>H2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>63</th>
<th>2) Do you have a meeting do discuss personal experiences, of what people think went well and not so well?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[Feedback] [Experience]</td>
</tr>
<tr>
<td>H2</td>
<td></td>
</tr>
<tr>
<td>H3</td>
<td></td>
</tr>
</tbody>
</table>

### S) Shared ownership
- Anyone can change anything (extreme version)
- Varies within different agile frameworks

<table>
<thead>
<tr>
<th>64</th>
<th>1) What are your rules regarding responsibilties (who can do what)?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[Shared ownership]</td>
</tr>
</tbody>
</table>

## Technical

### T) Test driven development (TDD)
- Write tests before making changes / adding functionalities
- Very software
- Do not move until all tests succeed.

<table>
<thead>
<tr>
<th>65</th>
<th>1) How do you make sure you learn something from your prototypes?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[Testing, y/n]</td>
</tr>
<tr>
<td>E1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>66</th>
<th>2) Do you have a way to determine what results that prevents further development, and which are 'good-to-go'?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[All tests passed]</td>
</tr>
<tr>
<td>E4</td>
<td></td>
</tr>
</tbody>
</table>

### U) Refactoring
- Changing an existing code step by step, so that errors are less likely to occur.
- Specifically for software coding.

<table>
<thead>
<tr>
<th>67</th>
<th>1) When you are doing a radical change of the prototype, do you do the whole change in 'one go', or do you do these step by step so to minimize the possibility of a failure?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[Refactoring]</td>
</tr>
</tbody>
</table>

### V) Joint development
- Do you have two or more people doing the same thing at the same time (cooperation)?

<table>
<thead>
<tr>
<th>68</th>
<th>Do you have two or more people doing the same thing at the same time (cooperation)?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[Joint development, y/n]</td>
</tr>
</tbody>
</table>

### W) Simplest solution that can work
- Do you make the simplest solution possible, every time?

<table>
<thead>
<tr>
<th>69</th>
<th>1) Do you make the simplest solution possible, every time?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[Simple mvp, y/n]</td>
</tr>
<tr>
<td>B5</td>
<td></td>
</tr>
<tr>
<td>X) Standardize</td>
<td>- Define style rules that the team should apply to all coding</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>70</td>
<td>1) Do you simplify components to standards, or do you customize? [Standardize]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Artifacts</th>
<th>Questions</th>
<th>Answers</th>
<th>Ref</th>
</tr>
</thead>
<tbody>
<tr>
<td>71</td>
<td>1) Do you use user stories / scenarios / use case?</td>
<td>[Scenarios, y/n]</td>
<td>M1</td>
</tr>
<tr>
<td>72</td>
<td>2) When you are developing a new prototype, how do you know it is 'done'? How do you define this?</td>
<td>[Definition of done]</td>
<td></td>
</tr>
<tr>
<td>73</td>
<td>3) Do you use any method/representation to visualize the teams progress?</td>
<td>[Burndown/up chart]</td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>4) Do you have a method to keep track of what features that needs to be implemented? Do you visualize these some-how?</td>
<td>[Story card / story board]</td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>5) How is the working conditions / working environment?</td>
<td>[Open room]</td>
<td>H1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Additional questions</th>
<th>Questions</th>
<th>Answers</th>
<th>Ref</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>Do you use any frameworks in your company? If so, which/how?</td>
<td>[Scrum] [RO3]</td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td>Have you ever changed you development method?</td>
<td>[Iterative improvement]</td>
<td></td>
</tr>
<tr>
<td>A3</td>
<td>How much time do you use on you startup?</td>
<td>[Sustainable pace]</td>
<td></td>
</tr>
<tr>
<td>A4</td>
<td>If you were to interview a new team member, what would you say your values are?</td>
<td>[Values for product development]</td>
<td></td>
</tr>
<tr>
<td>A5</td>
<td>Do you have any core rules? «The best way to do product development is...»</td>
<td>[Principles]</td>
<td></td>
</tr>
<tr>
<td>A6</td>
<td>Could you mention any specific activities you practice? «This is how we do it...»</td>
<td>[Practices]</td>
<td></td>
</tr>
<tr>
<td>A7</td>
<td>Do you use any virtual tools to visualise progress?</td>
<td>[Visualise progress]</td>
<td></td>
</tr>
</tbody>
</table>