Norwegian University of Science and Technology

Exploring Empirical Engineering Approaches in Startup Companies: The Trilateral Hardware Startup Model

Vebjørn Berg

Jørgen Birkeland

Master of Science in Computer Science

Submission date:	June 2018
Supervisor:	Maria Letizia Jaccheri, IDI
Co-supervisor:	Ilias Pappas, IDI
	Anh Nguyen-Duc, Høgskolen i Sørøst-Norge

Norwegian University of Science and Technology

Department of Computer Science

Acknowledgment

We would like to thank Professor Letizia Jaccheri for her supervision during the Master thesis process. Also, we would like to thank Post Doctoral Fellow Ilias O. Pappas and Associate Professor Anh Nguyen-Duc for their co-supervision. All have contributed with their expertise within Software Engineering research. IDI and NTNU deserve a special recognition for their brilliant services and facilitating a great learning experience.

Abstract

Technological advances, decreased component cost, small-batch manufacturing, social connections, and rapid prototyping have lowered the barriers for launching hardware startups. The rising hardware ecosystem has reduced environmental issues related to cost and scalability. Although the obstacles to success are decreasing, hardware startups operate in an environment posing challenges to traditional development and innovation methods. More research should be provided to support engineering activities in the unique and special context of hardware startups.

This Master thesis seeks to explore the work-practices in hardware startups by investigating the role of engineering activities, from idea conceptualization to a launched product. In particular, it investigates factors influencing development speed and agility and explores commonalities, challenges and, situational factors. The research direction was formulated on the basis of *"Software Startup Engineering: A Systematic Mapping Study"* (Appendix B). We state the following research questions:

RQ1 How do hardware startups achieve agility during product development?

- RQ1.1 How do hardware startups develop their products?
- RQ1.2 What kind of challenges are relevant in the hardware startup context?
- RQ1.3 How do internal/external context factors impact the speed of product development?

RQ2 How do hardware startups manage quality concerns of their products?

- RQ2.1 How are hardware products tested?
- RQ2.2 How is technical debt managed in hardware startups?

RQ3 How do hardware startups achieve balance between speed and quality?

We performed a multiple-case study investigating 13 hardware startups. Data were collected through semi-structured interviews. Transcribed interviews were analyzed using a thematic synthesis process to create a model of higher-order themes. The findings of this study led to the following three themes *third-party dependency*, *hardware-software integration*, and *two-folded product quality trade-off* operating the hardware startup context. Thus, the study contributes to the area of startup engineering as it draws from the *Greenfield Startup Model* and extends it with the three unique themes leading to the creation of the *Trilateral Hardware Startup Model.*

Our study presents initial research results on engineering activities in early-stage European hardware startups. The results indicate that hardware startups achieve rapid prototyping through evolutionary approaches, simple software side solutions, and opportunistic agile practices. Quality assurance is an informal process where testing practices are entrusted to each individual team member. As investing in hardware quality is essential for speed and bringing products fast to market, more research should be provided describing how hardware startups can manage the relationship between restricted resources and increased quality demands.

Sammendrag

Teknologiske fremskritt, reduserte komponent-kostnader, lav-skala produksjon, sosiale nettverk og rask prototyping har senket barrierene for å lansere en hardware startup. Det voksende hardware-økosystemet har redusert problemer knyttet til kostnad og skalerbarhet. Selv om hindrene for suksess er avtagende, opererer hardware startups i et miljø som stiller utfordringer til tradisjonelle utviklings- og innovasjonsmetoder. Mer forskning bør foretas for å støtte produktutviklingsaktiviteter i den unike og spesielle konteksten til hardware startups.

Denne masteroppgaven tar sikte på å utforske arbeidspraksiser i hardware startups ved å undersøke rollen til produktutviklingsaktiviteter, fra idé-konseptualisering frem til et lansert produkt. Spesielt undersøker den faktorer som påvirker utviklingshastighet og smidighet, og utforsker fellestrekk, utfordringer og situasjonsavhengige faktorer. Problemstillingen er formulert på bakgrunn av *"Software Startup Engineering: A Systematic Mapping Study"* (Appendix B). Vi definerer følgende forskningsspørsmål:

RQ1 Hvordan oppnår hardware startups smidig produktutvikling?

- RQ1.1 Hvordan utvikler hardware startups sine produkter?
- RQ1.2 Hvilke utfordringer er relevante i konteksten til hardware startups?
- RQ1.3 Hvordan påvirker interne/eksterne faktorer produktutviklingshastigheten?

RQ2 Hvordan håndterer hardware startups produktkvalitet?

- RQ2.1 Hvordan testes hardware produkter?
- RQ2.2 Hvordan håndteres teknisk gjeld i hardware startups?

RQ3 Hvordan oppnår hardware startups balanse mellom fart og kvalitet?

Vi gjennomførte en multiple-case studie som undersøkte og analyserte 13 hardware startups. Data ble innsamlet gjennom semi-strukturerte intervjuer. Transkriberte intervjuer ble analysert ved bruk av en tematisk prosess for å skape en modell av overordnede temaer. Resultatene ledet til følgende tre tema for hardware startups *tredjepartsavhengighet*, *integrering av hardware-software* og *to-foldet produktkvalitets trade-off*. Studien bidrar til området *startup engineering* da den viderefører kunnskap fra *the Greenfield Startup Model* og utvider den med tre unike tema representert ved *the Trilateral Hardware Startup Model*.

Vår undersøkelse presenterer initielle forskningsresultater for produktutviklingsaktiviteter i tidlig-fase europeiske hardware startups. Resultatene indikerer at hardware startups oppnår rask produktutvikling gjennom evolusjonære prototyper og enkle software-side løsninger. Kvalitetssikring er en uformell prosess hvor testing blir betrodd til hvert enkelt lagmedlem. Ettersom investering i hardware-kvalitet er avgjørende for hastighet og for å bringe produkter raskt til markedet, bør det forskes mer på hvordan hardware startups kan håndtere forholdet mellom begrensede ressurser og økte kvalitetskrav.

Preface

This thesis is submitted to the Norwegian University of Science and Technology (NTNU) as part of the course TDT4900 Computer Science, Master's Thesis. The thesis is to be submitted to the *Journal of Information and Software Technology (Elsevier).*

The work has been performed at the Department of Computer Science, NTNU, Trondheim, under the supervision of Professor Letizia Jaccheri as the main supervisor, and Post Doctoral Fellow Ilias O. Pappas and Associate Professor at the University of Southeast Norway Anh Nguyen-Duc as co-supervisors.

The Master thesis builds on previous work from the course TDT4501 Computer Science, Specialisation Project. The specialization project *Software Startup Engineering: A Systematic Mapping Study* has been submitted to the *Journal of Systems and Software (Elsevier)*, and may be found in Appendix B.

In addition, we have submitted the paper *The Role of Data Analytics in Startup Companies: Exploring Challenges and Barriers* to the 17th IFIP Conference on e-Business, e-Services, and e-Society, Challenges and Opportunities in the Digital Era *(I3E 2018)*, and may be found in Appendix C.

Table of Contents

List of Tables

List of Figures

Abbreviations

CEO	=	Chief Executive Officer
CSO	=	Chief Strategic Officer
CTO	=	Chief Technical Officer
COO	=	Chief Operating Officer
GQM	=	Goal Question Metric
GSM	=	The Greenfield Startup Model
HR	=	Human Resource
IDI	=	Department of Computer Science
IoT	=	Internet of Things
MVP	=	Minimum Viable Product
NSD	=	Norwegian Centre for Research Data
NTNU	=	Norwegian University of Science and Technology
PCB	=	Printed Circuit Board
RC	=	Radio Control
SE	=	Software Engineering
SME	=	Small and Medium Enterprises
SRRN	=	Software Startup Research Network
SWEBOK	=	Software Engineering Book of Knowledge
THSM	=	The Trilateral Hardware Startup Model
UX	=	User Experience
VSE	=	Very Small Entity

Chapter 1

Introduction

The barriers for starting a hardware company have never been lower, a result of the rising hardware ecosystem. Technological advances, rapid prototyping, decreased component costs, small-batch manufacturing, and fundraising platforms have renewed the interest for hardware startups (DiResta et al., 2015; Wei, 2017). Even though the obstacles to success are decreasing, the hardware startup context poses several challenges to traditional product development and innovation methods (Ronkainen and Abrahamsson, 2003). Practitioners need to handle the many dependencies posed by the complex nature of high-tech products like vendor, platform, and competence dependency (Nguyen-Duc et al., 2018). More research should be provided to support engineering activities in the unique and special context of hardware startups.

The objective of this Master thesis is to *create a better understanding of work-practices in hardware startups by investigating the role of engineering activities, from idea conceptualization to a launched product. In particular, we will investigate factors influencing development speed and agility, and explore commonalities, challenges and situational factors.* We present the findings from a multiple-case study investigating 13 hardware startups. This implies several recommendations for future work, and the creation of *The Trilateral Hardware Startup Model (THSM)* to present the overall engineering approach of hardware startups.

The remainder of this chapter proceeds as follows: Section 1.1 presents the motivation for this research. Section 1.2 presents the research questions. Section 1.3 defines the boundaries of the research. Section 1.4 explains the chosen research method and research process. Section 1.5 presents the outline of this thesis.

1.1 Motivation

Technology-based startups have long been an important driver for global economic growth and competitiveness (Unterkalmsteiner et al., 2016). Startups, newly created companies producing cutting-edge technology, have shown to be an important source of technology innovation, and will be influential in what is referred to as the third technological revolution the *Internet of Things* (Jacobson et al., 2017). Despite stories of successful startups, most of them fail, primarily due to self-destruction rather than competition (Marmer et al., 2011; Crowne, 2002). The failures come from financial and market factors like insufficient funding to operate startups activities, failure in finding product-market fit, and building an entrepreneurial team (Giardino et al., 2015). There are also identified unique challenges related to product development and innovation methods (Giardino et al., 2015). Startup researchers have called for a further attention to engineering approaches to support startup activities in all startup evolution stages (Unterkalmsteiner et al., 2016). Previously, most of the research in the field of software engineering has been conducted in relation to the needs and challenges of established companies, first identified by Sutton (Sutton Jr, 2000).

The project thesis (*Software Startup Engineering: A Systematic Mapping Study*) undertaken in *TDT4501 - Computer Science, Specialization Project* serves as the foundation for this Master thesis. It has been submitted to the Journal of System and Software (Elsevier) and can be found in Appendix B. The systematic mapping study identifies a change in focus of research area and contextual factors operating startup research. Directions for future work are suggested, including startup evolution models and human aspects, and a consolidation of the contextual factors of software startups. In addition, the study identifies limited research and knowledge on hardware startups. Current literature on hardware startups indicate that specific development practices are required (Nguyen-Duc et al., 2017a; Stock and Seliger, 2016), one reason being their multifaceted architectures and abstraction layers (Jacobson et al., 2017).

Hardware startups develop products with mixed hardware and software parts, including connected devices, sensor devices and advanced robotics (DiResta et al., 2015; Jacobson et al., 2017). This means that they not only work with hardware, but ”may include a significant amount of software components at the system level and solution level” (Nguyen-Duc et al., 2018). Hardware startups are distinct to software startups as they need to handle hardware design and development, and manufacturing in addition to software development. This implies both increased development time and cost compared to software products (Chen, 2015). Knowledge from development of embedded products in established companies can be related to hardware startups product development, however the startup context is unique and special (Nguyen-Duc et al., 2018; Ronkainen and Abrahamsson, 2003). As hardware startups are a popular phenomenon that are becoming increasingly influential in development of new electronic products, more effort should be made looking into practices and processes of hardware product development.

1.2 Research Questions

The technological lifecycle of companies delivering new innovative electronic products is becoming shorter each year. High quality demands, accelerating adoption rates of customers, and pressure for reduced time-to-market increase the uncertainty related to product development processes of hardware startups. In such a context the combination of speed and agility is essential for delivering innovative customer-driven products (Bosch, 2016). Speed is fundamental for staying alive in highly competitive markets, where creativity needs to be combined with agility to handle uncertainty, and introduce flexibility in the process (Garbajosa et al., 2017). At the same time, speed needs to be managed with caution to avoid expensive rework and bringing the development process to a halt due to low priority of product quality. We aim at exploring how agility, speed, and quality are engineered by practitioners in hardware startups. This has motivated the following research questions:

RQ1 How do hardware startups achieve agility during product development?

- RQ1.1 How do hardware startups develop their products?
- RQ1.2 What kind of challenges are relevant in the hardware startup context?
- RQ1.3 How do internal/external context factors impact the speed of product development?

RQ2 How do hardware startups manage quality concerns of their products?

- RQ2.1 How are hardware products tested?
- RQ2.2 How is technical debt managed in hardware startups?

RQ3 How do hardware startups achieve balance between speed and quality?

1.3 Research Scope

The units of analysis are people involved in product development in startup companies who deliver products with mixed hardware and software parts. This multiple-case study is constrained to the following list of inclusion criteria.

- The startup develops products with both hardware and software parts.
- The startup has been active for at least six months.
- The startup has a first running prototype.
- The startup's ambition is to scale its business.

Interviewees from the relevant startups were eligible for participation if they had experience and/or knowledge about software and/or hardware development. If the candidate met the criteria, he/she was regarded as qualified for contributing to answering the research questions. Data were collected from in-depth semi-structured interviews, pre-interview questionnaires, and relevant startup and incubator websites.

Eight of the participating startups are located in Trondheim, three are located in Oslo, one is located in Italy, and one is located in the Netherlands. All startups have to a various extent received funding from public institutions, private investors or established organizations. The transferability of results is restricted to early-stage European hardware startups, hence the results may not be generalizable to startups located outside Europe.

1.4 Research Process

To address our research questions, we decided to conduct a multiple-case study of hardware startups satisfying the inclusion criteria defined in the research scope. The case study process fitted both the time-constraints and availability of participants and is considered suitable for software engineering research (Pervan and Maimbo, 2005; Runeson and Höst, 2009).

To collect data, we performed semi-structured interviews with 13 hardware startups. All interviews were recorded and transcribed, and all participants signed consent forms. We used the GQM method (Basili, 1992) to develop interview questions appropriate for answering our research questions, enabling us to focus on pre-defined topics while at the same time allowing for an exploratory approach. The interview protocol was designed in accordance with the case and subjects selection criteria, and most interviews were conducted face-to-face on-site for the collection of reliable data (Runeson and Höst, 2009).

To analyze the data, we adapted the thematic synthesis process by Cruzes and Dyba (2011). We used NVivo to code the transcribed interviews. A total of 48 codes were generated from 734 references. An integrated approach was applied in combination with the descriptive coding technique (Saldaña, 2015). The codes were translated into themes by following an axial coding approach (Strauss and Corbin, 1998). These themes were further related to each other using the four-step process for creating a model of higher-order themes (Cruzes and Dyba, 2011). The higher-order themes were together with knowledge from the *Greenfield Startup Model (GSM)* combined to create the THSM, describing the engineering approach of hardware startups. Lastly, we discussed threats to the validity and ethical considerations to ensure the trustworthiness of the study.

1.5 Outline of the Thesis

The rest of the thesis proceeds as follows. Section 2 introduces the background of the study and relevant theoretical frameworks. The section presents both software and hardware startups and their unique needs in terms of practices and processes. Section 3 presents the systematic mapping study undertaken in advance of the Master thesis, and its main findings. Section 4 presents the research method undertaken, threats to the validity of the study, and some ethical considerations when conducting empirical research. Section 5 reports the results of the multiple-case study, including transcribed citations from the participants. Section 6 presents the THSM, representing the engineering activities of hardware startups to bring a product to market. Section 7 discusses the results in relation to the research questions and compares product development in hardware and software startups. Section 8 concludes the paper by answering the research questions and proposes directions for future work.

Chapter 2

Background

The theoretical foundation of this Master thesis builds on our systematic mapping study of software startup engineering, and on literature within embedded systems development. The chapter proceeds as follows: Section 2.1 explains the increasing impact of startups in the global consumer and business market. Section 2.2 presents software startups and characteristics of the startup context. Section 2.3 introduces some startup development methodologies, including the widely employed entrepreneurial framework by Eric Ries, *The Lean Startup*. Section 2.4 presents software development processes in startups and how they have different needs from that of established companies. Section 2.5 presents *The Greenfield Startup Model* and other relevant frameworks. Section 2.6 presents hardware startups and how they are different from software startups. Finally, section 2.7 presents embedded systems development and its relation to hardware startups.

2.1 Startup Movement

Technology is moving forward faster than ever before, changing how we work, relate, communicate, and learn, affecting every industry. The world is full of disruptors that challenge products, services, and business models for companies of all sizes. The life expectancy of the most valuable companies in the world has dropped to an average of 15 years (Gittleson, 2012). This makes focus on innovation a necessity, forcing new products and services to market.

The decreased cost of launching new technology startups and the mobility of online consumers imply that new markets can be reached at a rapid pace. It is easier than ever to successfully start a new technology company (Blank, 2012; Chesbrough, 2006), illustrated by the high number of newly established startups globally each year (U.S., 2017). Startups' influence on consumers and the global business market has become significant (Marmer et al., 2011), considerably contributing to the global industrial development (Unterkalmsteiner et al., 2016). These innovative new products and services not only disrupt themselves, but often end up leading the way for the introduction of even more new and disruptive innovations (Srinivasan et al., 2014). Startups generate new jobs, contributing to the overall wealth and global economy (Unterkalmsteiner et al., 2016).

There exist several initiatives supporting startups with funding, marketing, advisory, and other business-related issues. Venture capital has for a long time been a widely employed business model, where venture capitalists invest in new business ventures (Chesbrough, 2006). The venture capital market is continuously evolving, where crowdsourcing-platform Kickstarter has become the world's largest funding platform for creative projects. Since 2009, 140 000 projects have been successfully funded by more than 14 million people, contributing with $3.5 billion (Kickstarter, 2018).

Business incubators are becoming increasingly popular (Grimaldi and Grandi, 2005), reflected by the emergence of incubators in the Norwegian and European innovation communities, where FinTech F3 and Station F are examples of newly opened business incubators. The latter one is a $265 million investment, housing more than 1000 startups (Agnew, 2017). Incubators help startups survive and grow during their early stages, and facilitate collaboration between companies, investors, mentors, and other stakeholders to create innovations (Aernoudt, 2004). Startups are the heart of these innovative hubs, emphasizing startups' impact in today's innovative business landscape.

Norway has experienced a startup boom over the last years, and the trend is expected further growth. Oslo is one of the fastest growing tech scenes outside of the U.S., and is now featured as one of the 25 top startup hubs in the world (Clark et al., 2017). With a yearly growth in startup investments of 160%, Norway will potentially overtake Finland as the primary tech scene in the Nordic. The high quality tech infrastructure, access to data, adoption rate of digital technology, and high employment in technology sectors are incentives for startups to base themselves in Norway (Clark et al., 2017).

2.2 Software Startups

Since the term *software startup* first was introduced by Carmel (1994), there have been presented several definitions of software startups. Sutton Jr (2000) defined startups as "an organization that is challenged by youth and immaturity, with extremely limited resources, multiple influences, and dynamic technologies and markets". Coleman and O'Connor (2008b) described software startups as "unique companies that develop software through various processes and without a prescriptive methodology". Others have characterized software startups as modern organizations with little or no operating history, aiming at developing high-tech and innovative products, and rapidly scale their business in extremely dynamic markets (Giardino et al., 2014a).

Software startups distinguish themselves from established companies in many ways. Established companies focus on optimizing an existing business model and don't necessarily focus on growing, while startups are temporary organizations seeking a scalable, sustainable, and profitable business model (Blank, 2012). With limited resources, startups often seek funding from venture capitalists or investors (Nguyen-Duc and Abrahamsson, 2017), and so the business model is often based on disruptive technology that can give an immediate competitive advantage. As customers and products often are unknown, the success of startups depends on how fast they can prototype to test business ideas (Nguyen-Duc et al., 2017b; Sutton Jr, 2000). In contrast, established companies are more mature and experienced, with greater resources, and already command a mature market where customers and products are known (Unterkalmsteiner et al., 2016). Most startups fail within the first two years (Giardino et al., 2016; Marmer et al., 2011), one reason being the lack of supportive engineering practices (Kajko-Mattsson and Nikitina, 2008; Klotins et al., 2015; Paternoster et al., 2014). Comprehensive descriptions of investigated startups are important in an applied field as software engineering to be able to transfer results from one environment to another (Klotins et al., 2015).

2.2.1 Startup Lifecycle

There are several frameworks describing the evolution of startups (Crowne, 2002; Reynolds and White, 1997; Ries, 2011). Common is that they divide the startup lifecycle into 3-5 stages, from the stage of an idea to a launched product. Crowne (2002) presented the following four-stage startup lifecycle:

Startup: The startup stage is defined as the time from idea conceptualization to the first sale. A small executive team with necessary skills is required in order to build the product.

Stabilization: The stabilization phase lasts until the product is stable enough to be commissioned to a new customer without causing any overhead on product development.

Growth: The growth phase begins with a stable product development process and lasts until market size, share, and growth rate have been established.

Maturity: The last stage is when the startup has evolved into a mature organization. The product development is robust and easy to predict, with proven processes for new product inventions.

The lifecycle stages illustrate why early-stage startups are different from companies who have passed the maturity stage. After entering more mature lifecycle stages, these companies are often referred to as small to medium sized enterprises (SMEs), operating in stable environments where customers and products are known. In contrast, they have a settled team, and have acquired some experience and operating history, hence research on SMEs is only partially relevant for startups (Tripathi et al., 2016). As the different lifecycle stages of startups pose their own set of challenges, specifying the state of the investigated startup is important for transferring research results to new environments (Klotins et al., 2015).

2.3 Startup Development Methodology

The main objective of all startups is to find a sustainable business model through developing products that meet actual customer needs (Ries, 2011). This will eventually lead to further business growth where product development is robust and easy to predict, with proven processes for new product inventions (Crowne, 2002).

Software startups generally develop products in high-potential target markets without necessarily knowing *what* the customers want (Blank, 2013b; Rafiq et al., 2017). Increasingly more industries experience that new technologies become available to all players at the same time, meaning the benefits of technology-driven innovations decrease. This has lead companies to prioritize customer-driven development, which involves identifying new and unknown customer needs as well as meeting known needs (Bosch, 2016). This relates to market-driven software development, where specific requirement elicitation techniques (e.g., prototyping) and time-to-market are key strategic objectives (Nguyen-Duc et al., 2017b; Rafiq et al., 2017). In a market-driven context, requirements tend to be (1) invented by the software company, (2) rarely documented (Karlsson et al., 2002), and (3) validated only after the product is released in the market (Carmel, 1994; Dahlstedt, 2003; Keil and Carmel, 1995; Rafiq et al., 2017). As to this, products that don't meet customer needs are common, resulting in failure of new product releases (Alves et al., 2006).

To manage the challenges posed by the startup context, software startups need systematic processes, both for their business development and their product development. There exist several entrepreneurial theories and frameworks that can guide practitioners in their pursuit to sustained business growth (e.g., Effectuation Theory (Sarasvathy, 2001) and Discovery and Creation (Alvarez and Barney, 2007)). Blank (2013a) introduced the customer development approach to entrepreneurship, identifying customer discovery and validation as key strategic objectives to find a sustainable business model (i.e., processes concerned with *what* product to develop rather than *how* to). The framework is the main inspiration of the Lean Startup method proposed by Ries (2011). The Lean Startup has been criticized by researchers for being based on personal experience and opinions rather than empirical evidence, however, concepts from the Lean Startup have attracted considerable attention among practitioners (Bosch et al., 2013; Blank, 2013b).

2.3.1 Lean Startup

Ries (2011) presented the Lean Startup method in 2008, based on lean principles first introduced by Toyota (Womack et al., 1990). The method aims at creating and managing startups, to deliver products or services to customers as fast as possible. The method provides principles for how to run a new business, where the main objective is to grow the business with maximum acceleration. By iteratively turning ideas into products, measure customers' satisfiability, and learn from their feedback, startups can accelerate their business. This process is referred to as the build-measure-learn (BML) feedback loop, which is an iterative process, where the goal is to minimize total time through the loop.

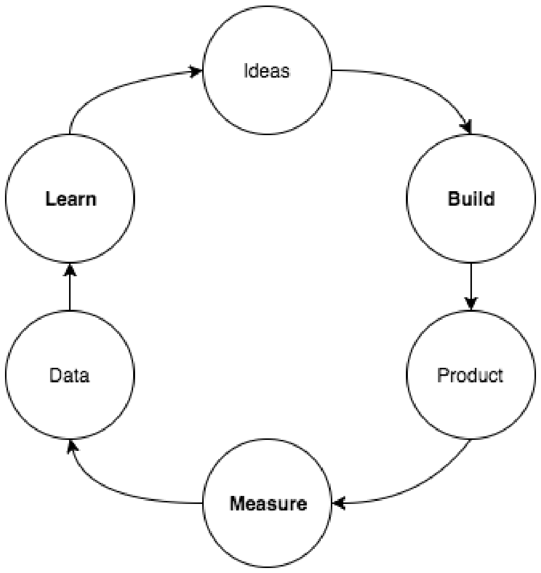


Figure 2.1: The Build Measure Learn Feedback Loop (Ries, 2011)

Key to the BML feedback loop is to do continuous experimentations on customers to test hypotheses. The hypotheses can be tested through building a minimum viable product (MVP), which is the simplest form of an idea, product, or service that can answer the hypotheses. Any feature, process, or effort not directly contributing to answering the hypotheses, is removed. The aim is to eliminate any waste throughout the process. Research has found that MVPs can be used both to bridge knowledge gaps within organizations, and to provide a mutual understanding between customer input and product design (Nguyen-Duc and Abrahamsson, 2016).

When the MVP has been built and the hypotheses tested, the next step is to measure the customer feedback and learn from it. This is referred to as validated learning, which is about learning which efforts are value-creating and eliminate the efforts that aren't necessary for learning customer needs. The final step of the loop is whether to pivot or persevere. A pivot is a structured course correction designed to test a new fundamental hypothesis about the product, strategy, and engine of growth (Ries, 2011). Through an investigation of 49 software startups, Bajwa et al. (2017) identified 10 pivot types and 14 triggering factors, concluding that trying to solve the wrong problem for the customer is the most common reason for pivoting (i.e., ”customer need” pivot). If a pivot isn't required, meaning the MVP was found to be fit to market, the startup perseveres. The BML feedback loop then continues, where new hypotheses are tested and measured.

The Lean Startup method is beneficial for business development and understanding *what* product to develop, emphasizing the importance of getting the product to customers as soon as possible. Startups tend to prefer time and cost over product quality (Yau and Murphy, 2013), neglecting traditional process activities like formal project management, documentation, and testing (Giardino et al., 2016). Shortcuts taken in product quality, design, or infrastructure can eventually inhibit learning and customer satisfaction (Ries, 2011). Software startups need their own development practices to manage the challenges posed by customer development methods such as Lean Startup.

2.4 Software Startup Engineering

The unique characteristics of the startup context pose several challenges related to the development processes in software startups (Giardino et al., 2016). The traditional software development methods of more established companies do not facilitate these constrained conditions. However, software engineering in startups represents a segment that has mostly been neglected in research studies (Paternoster et al., 2014; Unterkalmsteiner et al., 2016), first identified by Sutton Jr (2000). The last few years the interest in research on software startups have gained increased interest in the Software Engineering (SE) community, highlighted by the increased publication frequency (Figure 3.3). Giardino et al. (2016) proposed the new discipline *software startup engineering* to define a first set of concepts, terms, and activities for the software startup phenomenon, defined as "the use of scientific, engineering, managerial, and systematic approaches with the aim of successfully developing software systems in startup companies".

Startups are creative and flexible by nature, and so strict release processes are often overshadowed by quick, inexpensive product releases, with focus on customer acquisition (Wasserman, 2016). This can often result in ineffective software engineering practices (Sutton Jr, 2000). Since startups have limited resources, focus is often directed towards product development, rather than focusing on the establishment of rigid processes (Coleman and O'Connor, 2008b). In contrast to established companies who often have well-defined processes for their business, startups usually have low-ceremony processes where the amount of management overhead is low (Kuhrmann et al., 2016). Instead of utilizing repeatable and controlled processes, startups take advantage of reactive and low-precision engineering practices with focus on the productivity and freedom of their teams (Kakati, 2003; Tanabian and ZahirAzami, 2005). This is why product development in startups often is considered as a set of opportunistic activities, focusing on providing value under constrained conditions (Nguyen-Duc and Abrahamsson, 2017).

The need for product development process in startups is dependent on the lifecycle stage of the company, including aspects like system complexity, business risk, and the number of people involved (Wasserman, 2016). Reactive, low-ceremony processes are powerful in the early stages of software development since speed and learning are important (Ries, 2011). However, as startups enter new more mature lifecycle stages, an increased usage of processes for addressing key customer needs, delivering functional code early and often, and providing a good user experience is required (Kuhrmann et al., 2016). New business issues like hiring, sales, and funding appear, and more users and complex code require extended focus on robustness, scalability, performance, and power consumption (Wasserman, 2016).

Inadequate use of software engineering practices might be a significant factor leading to the high failure rates of software startups (Klotins et al., 2015). In software development, agile methods (Cunningham et al., 2001) have proven to be a powerful tool when the goal is to build a successful, profitable business model. When a company needs to quickly address market and customer needs, agile processes have proven to be much more effective than traditional high-ceremony processes (Wasserman, 2016). Research does not agree on the usage of agile methods in software startups. Some studies suggest agile methods are suitable for startups, as iterative development approaches are adaptive, with short lead time (Pantiuchina et al., 2017; Paternoster et al., 2014). Other studies have found that startups are either reluctant to introducing process (Coleman and O'Connor, 2008b), or that they use their own mix of agile and ad-hoc methods (Giardino et al., 2014a). Small early-stage software startups don't experience the same challenges as larger, more experienced companies, and the cost and time of implementing a rigorous agile methodology may not provide big enough benefits (Yau and Murphy, 2013).

Most research within software startup engineering merely provides a partial depiction of software engineering practices in startups. Current literature is not comprehensive enough to establish an exhaustive understanding of how software engineering practices are applied in startups, emphasizing the need for the creation of a complete scientific body of knowledge to support future software startups (Unterkalmsteiner et al., 2016).

2.5 Theoretical Frameworks for Startups

Entrepreneurial success is complex and depends on several interrelated factors. There exist a variety of theoretical frameworks that allow for positioning research results within a broader context of related concepts (Swanson and Chermack, 2013). Each theoretical framework has its own intention and motivation of use, and benefits startups differently. The frameworks provide the basis for understanding the behavior and engineering approaches of startups, and can help place findings in the context of theory. Section 2.5.1 presents the Greenfield Startup Model, which explains how development strategies are engineered and practices are utilized in software startups. Section 2.5.2 introduces other frameworks relevant to the startup context.

2.5.1 The Greenfield Startup Model

The GSM captures the underlying phenomenon of software development in early-stage startups. From 128 sub-categories clustered into 35 groups, the model consists of seven main concepts at the highest abstraction level. Each of the seven concepts was mapped to the primary papers in the systematic mapping study by Paternoster et al. (2014), to ensure the validity of the model and the conformance to the software startup context. Figure 2.2 shows the causal relationships between the concepts. The model illustrates that quick development is important due to a severe lack of resources, where low attention to quality leads to the accumulation of technical debt. The initial growth hinders the performance and future growth of the company. This section introduces each of the categories in more detail.

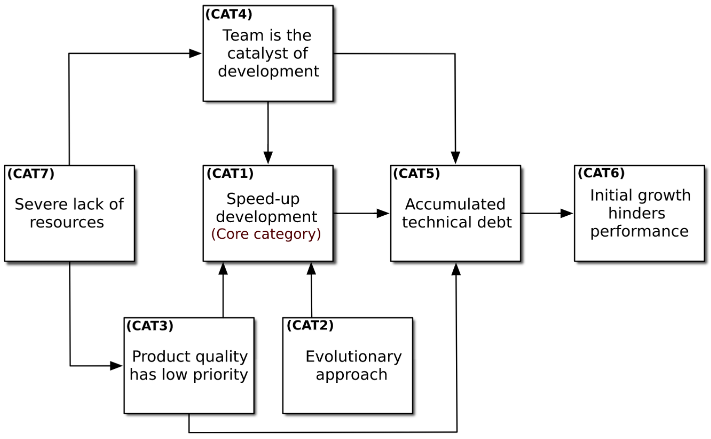

(CAT4)
Team is the catalyst of development
(CAT7)
Severe lack of resources
(CAT1)
Speed-up development
(Core category)
(CAT5)
Accumulated technical debt
(CAT6)
Initial growth hinders performance
(CAT3)
Product quality has low priority
(CAT2)
Evolutionary approach

Figure 2.2: The Greenfield Startup Model (Giardino et al., 2016)

Severe lack of resources. Lack of resources in startups include time-shortage, limited human resources, and limited access to expertise (Paternoster et al., 2014). Time pressure is the most severe due to the need for quick product releases. Limited human resources mean team members often are full stack engineers and employ multiple roles. Startups utilize mentors and advisors to mitigate the limited access to internal expertise. In an environment of strict limitations, development decisions are usually trade-off situations.

Team as the development catalyst. Team members hold multi-roles as they often have to handle both software development and business-related issues like marketing and sales. Small teams in co-located environments facilitate high team-coordination, where tacit knowledge and informal discussions replace most of the documentation. Small teams where members know each other is beneficial to achieve rapid evolvement. The background of the founders and CTO/CEO highly impacts the development approach.

Evolutionary approach. Startups prefer evolutionary prototyping, meaning that they iteratively refine an initial prototype aiming at quickly validating the product/market fit. Throwaway prototypes are mainly used for specification purposes and not as actual building blocks (Nguyen-Duc et al., 2017b). When the product is released, customer feedback highlights new functionalities and improvements. Uncertain conditions mean long-term planning is infeasible, and so flexibility and reactiveness are key priorities. To achieve competitive advantage, startups utilize cutting-edge technologies and follow a customer-driven approach. The main objective is to minimize waste throughout the process by building MVPs that are valuable to customers, to avoid "over-engineering the system" with complex, unvalidated functionalities.

Product quality has low priority. Startups prioritize a limited number of functionalities rather than implementing non-functional requirements to allow for quick product releases. Depending on the type of system being developed, startups focus on usability, UX, and smooth user-flow without interruptions. Startups tend to favor agile practices over quality-related practices (Pantiuchina et al., 2017), creating MVPs which may lack in quality but that are functional enough to pitch and show investors (Yau and Murphy, 2013). Other factors influencing the product quality include the amount of outsourcing (Nguyen-Duc and Abrahamsson, 2017) and the fault-tolerance of users.

Speed-up development. The primary objective of startups is to speed up the product development in the early stages. Companies today must respond to new customer needs and requests at exceptional speeds (Bosch, 2016). It is important both to speed up the learning processes (Nguyen-Duc et al., 2017b) and the decision and design processes (Yau and Murphy, 2013). Speed is achieved through an evolutionary approach with a solid team focusing on developing MVPs to minimize waste. Simple, informal workflows through small self-organized teams allow for flexibility and reactiveness. Decisions are taken as fast as possible through informal and frequent verbal discussions to deal with the unpredictable startup context. The importance of speed also leads to startups neglecting the implementation of development practices and systematic quality assurance activities like efficiency. Startups make use of third-party solutions to deliver scalable products, however, such approaches often lead to interoperability issues. Proven standards and known technologies are often utilized to reduce the need for formal architectural design. Key to the speed of development is the members' product-ownership feeling and their desire to have the product used in the market.

Accumulated technical debt. To achieve speed, startups ignore documentation, software architecture, and processes. Such shortcuts can lead to the accumulation of what is called intentional technical debt (McConnell, 2007). Unintentional technical debt can happen when business model experimentation is leaved out. Not focusing on technical debt will have consequences for the product quality, while constantly changing and improving the business model will be necessary to stay competitive (Yli-Huumo et al., 2015). Requirements engineering processes are replaced by informal specifications of functionalities (e.g., self-exploratory user-tasks on post-its). Traditional analysis and planning are often replaced by an inaccurate feasibility study, however, this can have negative consequences at later stages. Finding the correct balance between quality and speed is essential, a problem also referred to as the developer's dilemma (Terho et al., 2016). It emphasizes the need for managers to communicate the learning goals of the product precisely so that developers can adjust quality accordingly. Not finding the correct match between learning goals and quality will often lead to technical debt, waste, or missed learning. The difficulty of analyzing risks with disruptive technologies often makes it worth keeping the product and processes simple to achieve agility and speed, and so practitioners often prefer using their past experience from similar context when assessing the feasibility of the project. Early decisions are often limited to achieve flexibility of the team, however, it can increase the technical debt. High-level mockups and low-precision diagrams describing the interactions with third-party solutions are prioritized over good architectural design. This will eventually inhibit performance as the team grows and new developers are hired. Automated testing is often replaced by internal smoke tests, so defects are often detected by the users. Rigid project management is replaced by short, informal milestones, low-precision task assignment methods, and low-cost project metrics. Only a final release milestone is viable to allow focus on short-term goals and to put new features in production.

Initial growth hinders performance. When startups enter more mature lifecycle stages, the product becomes more complex, the number of users increases, and the company grows. The chaos encountered in the early-stages forces the company to deal with the accumulated technical debt instead of focusing on acquiring new customers and user requests, hence initial growth hinders the performance. Users demand higher quality of mature products, more funding is required, and new competitors emerge. The current team is often not able to manage the increased demands, code complexity, and scalability issues, while at the same time implementing new features. Informal communication and lack of documentation eventually inhibit the progress, and the business concerns are directed from product development towards business activities. Re-engineering the system by standardizing the codebase with well-known frameworks are required to increase the scalability of the product. Startups gradually introduce traceable systems and metrics for measuring project progress. Startups often allow a drop down in performance at a later stage than losing time in the initial phases of the project.

2.5.2 Relevant Frameworks

Effectuation Theory. The terms effectuation and causation were introduced by Sarasvathy (2001), and are academic theories describing the reasoning behind entrepreneurial actions. Effectual logic is used when the future is unpredictable, while causal logic is used when the future is predictable. In entrepreneurship, effectual reasoning starts with a set of resources, and in the process of using these, goals gradually emerge. On the contrary, causation involves achieving a desired goal through a set of given resources. Effectuation consists of five main principles for effectual entrepreneurs: (1) they start with a set of resources without a given goal, (2) they focus on potential losses and how to minimize these, (3) they cooperate with parties they can trust instead of analyzing competitors to limit potential losses, (4) they try to avoid contingencies where surprises are seen as opportunities, and (5) the future cannot be predicted, but entrepreneurs can control some factors that affect the future.

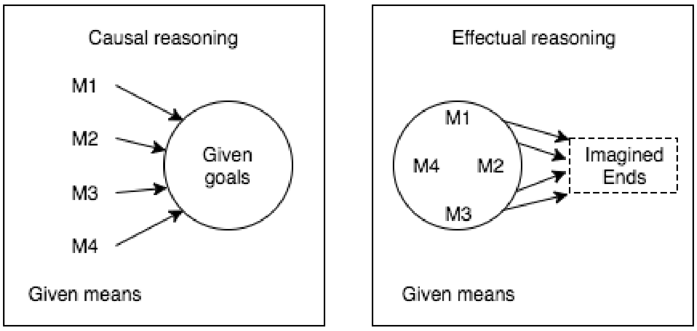
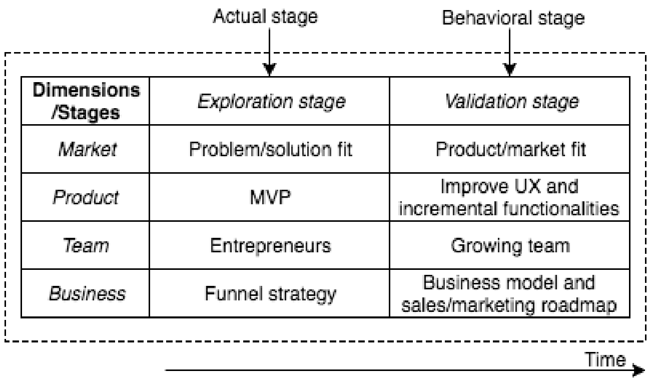


Figure 2.3: Effectuation vs. Causation (Sarasvathy, 2001)

The Behavioral Framework. A behavioral framework identifying why early-stage software startups fail have been introduced by Giardino et al. (2014b). They identified a lack of scientific papers characterizing startup failures. In addition to performing a literature review, they performed a multiple-case study illustrating how inconsistency between managerial strategies and execution can lead to failure. The framework differs between the actual stage (i.e., what startups should focus on) and the behavioral stage (i.e., what they actually focus on, which often is inconsistent with what they planned). The framework works over four dimensions including market, product, team, and business. They found that startups lack problem/solution fit, that is, they focus on validating a product instead of discovering and testing a problem space. Startups also tend to neglect the learning process, meaning they focus on making their customer acquisition process more efficient rather than testing the demand for a functional product. The framework is a first set of concepts, terms, and activities explaining possible reasons for the failure of software startups. More work is required to validate the framework to allow for generalization, as well as providing guidelines for how to prevent a mismatch between business intentions and development execution.



Dimensions /Stages	*Exploration stage*	*Validation stage*
Market	Problem/solution fit	Product/market fit
Product	MVP	Improve UX and incremental functionalities
Team	Entrepreneurs	Growing team
Business	Funnel strategy	Business model and sales/marketing roadmap

Figure 2.4: The Behavioral Framework (Giardino et al., 2014b)

In addition to the theoretical frameworks mentioned above, there exist a wide range of relevant frameworks, both from entrepreneurial and engineering perspectives. The GSM was compared with similar theories by Coleman and O'Connor (2008a,b); Baskerville et al. (2003); Brooks and Kugler (1987). A similar model to GSM, the Academic Startup Model, illustrates how software startups structure and execute their engineering activities (Souza et al., 2017). The Lean Startup methodology builds on several established entrepreneurial theories with many similarities like The New Business Road Test (Mullins John, 2003), Crossing the Chasm (Moore, 2002), and its direct ancestor, from which Eric Ries was inspired and influenced, The Four Steps to the Epiphany (Blank, 2013a).

2.6 Hardware Startups

Hardware startups can be defined as those startups that develop products with mixed hardware and software parts, including embedded systems, sensor devices, and advanced robotics (DiResta et al., 2015; Jacobson et al., 2017). Hardware startups are distinct from pure software startups as they need to handle hardware design and development, and manufacturing in addition to software development. They also have to deal with production and logistics issues like packaging, shipping, and customs (DiResta et al., 2015). Hardware startups need teams with boundary-spanning knowledge, including capabilities within software development, mechanical and electronics engineering, product design, and specific industry knowledge (e.g., experience from working with third-parties) (Nguyen-Duc et al., 2018). This implies higher initial financial and human investments required for hardware startups (Wei, 2017).

Before the dot-com bubble, many startups focused on developing hardware systems (Wei, 2017). However, due to lower investments, return on investment, and risk diversification of software and web-based startups, most venture capitalists' investments were directed away from hardware. The process of designing and developing hardware products takes more time and imply increased costs than that of software products (Chen, 2015), hence most innovations targeted the mechanical part of products as software development merely was limited to mechanical development. Today we have seen a shift in this trend. Increased availability of new technologies such as cloud infrastructure, sensors, microelectromechanical systems (MEMS), and open source software imply that hardware startups have greater access to resources (Wei, 2017). The shifting nature of product innovation, where innovations are increasingly more customer-driven rather than technology-driven (Bosch, 2016), has been reflected by the emergence of startups developing hardware technologies like IoT, cyber-physical systems, and advanced robots. GoPro, Fitbit, DJI, and Xiaomi are all examples of successful hardware startups characterized as "unicorns", valued at more than \$1 billion. By 2020, hardware technology is estimated to be in 95 percent of new electronic products (Gartner, 2017). The number of connected devices in the global market was approximately 15 billion in 2015, a number expected to grow to 75 billion by 2025 (Lucero et al., 2016).

2.7 Embedded Systems Development

The term "hardware" only covers one part of development activities in hardware startups and can be somewhat misleading since hardware startups deliver products with mixed hardware and software parts. Embedded systems are application-specific computing devices consisting of hardware and software components (Ronkainen et al., 2002). Current development methodologies for startups mainly cover specific needs for software startups, only partially applicable to hardware startups as embedded systems development is considerably more complex than software development (Stock and Seliger, 2016). Research on development processes in hardware startups is rare, where exploration of state-of-practice is limited to a few studies (Nguyen-Duc et al., 2018). In this section we introduce research on embedded systems development and initial studies on agile adoption in hardware startups.

In the embedded domain, hardware sets strict requirements for the software. Development of embedded systems require simultaneous development of hardware and software that directly accesses the hardware (i.e., hardware-related software) (Ronkainen et al., 2002). Since software allows for frequent updates and releases, both before and after products are delivered to customers, the system architecture often seeks to separate the hardware from the software to allow for two largely independent release processes (Bosch, 2016). This is illustrated in figure 2.5 where hardware and related software development are distinct processes requiring constant communication and interconnected testing and verification.

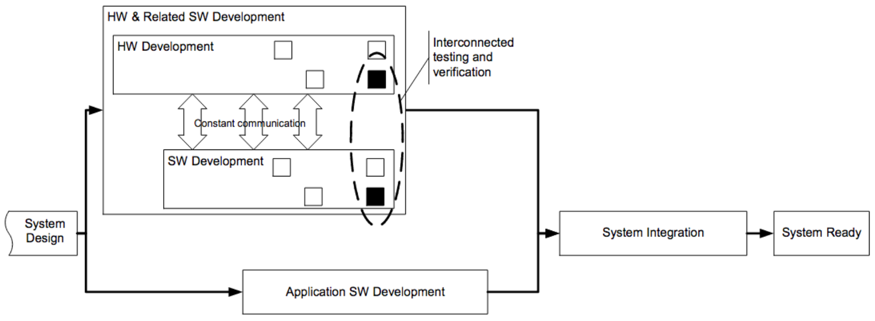


Figure 2.5: Hardware-software co-design process (Ronkainen et al., 2002)

Ronkainen et al. (2002) found four main characteristics of hardware and related software development, including (1) hard real-time requirements, (2) experimental work, (3) documentation requirements, and (4) testing.

1. Real-time requirements include, among other things, factors such as data throughput rates, cycle counts, or function call latency. Hard real-time requirements mean that if software doesn't meet requirements, further system operation may be at risk. To deal with this, hardware simulations can help determine the precise operation of hardware without producing an expensive prototype, and even enable testing of the hardware-software co-operation. Hardware simulations often depend on experts as they are time-consuming to set up and maintain, and because of the many complex hardware-software interactions.

2. Hardware-oriented software development is experimental by nature, and developers need to understand the whole system to deal with all uncertainties related to changes in hardware and how software affects the whole system. Every requirement cannot be known and every decision cannot be made before writing software. Developers should utilize an iterative development approach to deal with all ambiguities of hardware-related software development.

3. The communication among hardware and software developers must work to implement the hardware-software interface efficiently. Information has to be explicit and relies heavily on exact documentation. The iterative approach depends on accurate

documentation to minimize information loss between iterations. However, due to the vast amount of experimental work, too much documentation is not feasible in early stages of product development.

4. Testing is an essential activity in embedded systems development, both due to reliability and device autonomy requirements, and regression tests to ensure parallel development doesn't drift. In addition to independent software and hardware tests, checking the right interaction between hardware and software (i.e., co-verification) is important to ensure the system works as intended.

Instead of applying plan-driven development, Ronkainen et al. (2002) suggested using an iterative approach to deal with the experimental nature of embedded systems development. Kaisti et al. (2013) performed a mapping study identifying agile methods in embedded systems development. They found a general lack of hardware-related agile studies, only customized methods for specific needs (e.g., agile methods scaled and adapted to the needs of large organizations). One paper found that agile methods did not suit the specific needs of the embedded domain, highlighting that new agile methods should focus on meeting the hard real-time requirements. Up-front design and architecture require a certain amount of documentation and specification, also to ease communication and coordination among developers and stakeholders (Ronkainen and Abrahamsson, 2003).

Albuquerque et al. (2012) conducted an investigation into agile methods in embedded systems development, including their benefits, challenges, and limitations. They found that although agile methods and practices had a positive impact on embedded systems development, their use was not widespread. Among the primary papers, many reported experiences regarding the use of a single or a set of agile practices, but not the implementation of complete agile methods. The most widely researched methods were Scrum and XP, however, adoption of such was tailored to individual needs. The benefits of utilizing agile methods were found to be decreased development time, improved productivity, and reduced error rates (i.e., improved system quality). Agile methods also present good outcomes when dealing with changes and uncertainty in environments where developers avoid rigorous processes. The paper identified a need for a coherent understanding of how agile methodologies best fit to embedded systems development, and how such practices can reduce costs and efforts in different phases of the development process (i.e., requirement management, design, and architecture).

Nguyen-Duc et al. (2018) investigated how hardware startups develop their prototypes, and to what extent agile is adopted in hardware startups. As software startups utilize mockup tools to quickly represent product ideas, hardware prototypes are not only used for business experimentation, but are as much a feasibility check. The co-design of hardware and software components, and integration at system level lead to longer Sprint duration. Also the many dependencies with hardware development affect the prototype duration (e.g., vendor dependency and competence dependency). The nature of hardware development affects the adoption of agile practices. Agile methodologies may need to be adapted to fit the complexities of hardware development, and overcome the perceived reluctance to introducing rigid processes in startups. Flexible Sprint duration might be important to handle the many dependencies of hardware development.

Chapter 3

Systematic Mapping Study

Prior to the Master thesis, we conducted a systematic mapping study (Appendix B) to provide an updated view of software startup research in order to identify research gaps. Different from previous mapping studies (Klotins et al., 2015; Paternoster et al., 2014), we aimed at synthesizing startup descriptions in research and observe how software startup research has evolved and possibly matured in some Software Engineering knowledge areas (Bourque and Fairley, 2014). The study provides a mapping of 74 primary papers (in which 27 papers are newly selected) from 1994 to 2017, by expanding previous literature. We discovered that most research has been conducted within the SWEBOK knowledge areas software engineering process, management, construction, design, and requirements, with the shift of focus toward process and management areas. Future work can focus on certain research themes, such as startup evolution models and human aspects, and consolidate the thematic concepts describing software startups. This chapter summarizes our paper *"Software Startup Engineering: A Systematic Mapping Study"*, and proceeds as follows: Section 3.1 explains the research process. Section 3.2 presents the main findings of the mapping, and directions for future work.

3.1 Research Method

Systematic mapping studies can be used in research areas with few relevant primary studies of high quality, as they provide a coarse-grained overview of the publications within the topic area (Petersen et al., 2008). This systematic mapping study covers 74 primary papers, extending the two previous mapping studies (Paternoster et al., 2014; Klotins et al., 2015). As these studies only cover three papers from 2013, the search strategy of this systematic mapping study included papers from 2013 up to October 2017. This approach allowed for merging and comparing the primary literature within the research field for the period 1994-2017.

The main steps of our process are illustrated in figure 3.1, and include the search and study selection strategies, manual search, data extraction, quality assessment, and the data synthesis method. The process led to a total number of 27 new primary papers found in table 3.3.

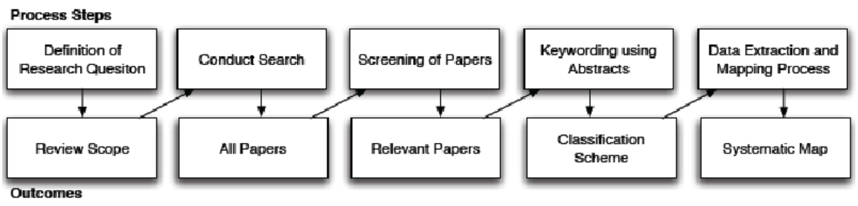


Figure 3.1: The Systematic Mapping Study Process (Petersen et al., 2008)

3.1.1 Research Questions

Since 2015, we observed an increased focus on software startup research (i.e., the organization of three International software startup workshops (ISSW) in 2016 and 2017, and software startups tracks at PROFES 2017 and XP 2017 conferences). The previous systematic review has rapidly gained a large amount of citations (Paternoster et al., 2014). While this implies the further growth in software startup research, a revisit on the area can identify how engineering activities in software startups have changed over time. The objective of this mapping study is to provide an updated view of software startup research in order to identify research gaps. The research objective leads to the following research questions:

RQ1: How has software startup research changed over time in terms of focused knowledge areas?

RQ2: What is the relative strength of the empirical evidence reported?

RQ3: In what context has software startup research been conducted?

The paper presents results from software startup research between 1994-2017. We expand previous literature (Paternoster et al., 2014; Klotins et al., 2015) with the focus on papers published from 2013-2017. We found 27 relevant articles during the last five years. The results were merged and compared to the previous mapping studies. To address RQ1, the papers were structured according to the knowledge areas identified in SWEBOK (Bourque and Fairley, 2014). With RQ2, we evaluated the papers' rigor to compare the quality of papers published before and after 2013. Finally, with RQ3, we examined to what extent the retrieved papers provided sufficient startup descriptions, and if there were similarities in the use of terms describing the startup context between the papers. Our meta-analysis on Software Engineering knowledge area and startup case context reveals important areas for investigation. We also come up with a classification of future research on software startups.

3.1.2 Data Sources and Search Strategy

The systematic search strategy consisted of searches in three online bibliographic databases. The databases were selected from their ability to handle complex search strings, and their general use in similar literature reviews in the software engineering community (Paternoster et al., 2014; Tripathi et al., 2016), and the fact that they index the research articles from other databases. In addition, to ensure the best possible coverage of the literature, we performed complementary searches and forward snowballing (section 3.1.4). Following guidelines from Wohlin (2014), systematic literature studies should use a combination of approaches for identifying relevant literature, where forward snowballing is found particularly useful. Forward snowballing can reduce systematic errors related to the selection of databases and construction of the search string (Wohlin, 2014). To obtain high quality data, the following databases were used:

Database	Papers
Scopus	451
ISI Web of Science	121
Engineering Village Compendex	875
Total	1447

Table 3.1: The searched databases and number of retrievals

Initial searches in the databases were conducted to identify keywords related to software engineering and startups, targeting title, abstract, and keywords. The most frequently used keywords for "startup" were chosen and combined in the search string (Paternoster et al., 2014). The final search string consisted of several search terms combined using the Boolean operator *"OR"*:

> "*(startups OR start-up OR startup) AND software engineering OR (startups OR start-up OR startup) AND software development OR (startups OR start-up OR startup) AND software AND agile OR (startups OR start-up OR startup) AND software process OR (startups OR start-up OR startup) AND software tools*".

3.1.3 Study Selection

The study selection process is illustrated in figure 3.2, along with the number of papers at each stage. Searching the databases Scopus, ISI Web of Science, and Engineering Village using the search string returned 1447 papers, resulting in 1012 unduplicated studies. The searches targeted the document types: book chapters, journal article, conference article, conference proceedings, dissertation, and report chapters.

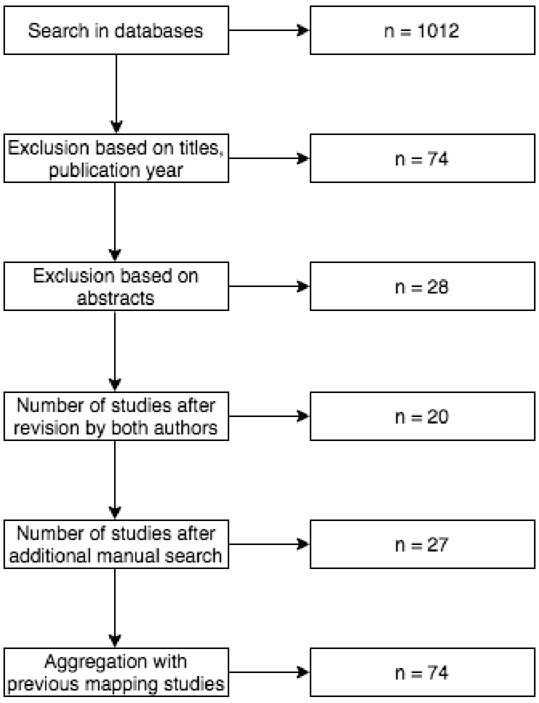

Search in databases
n = 1012
Exclusion based on titles, publication year
n = 74
Exclusion based on abstracts
n = 28
Number of studies after revision by both authors
n = 20
Number of studies after additional manual search
n = 27
Aggregation with previous mapping studies
n = 74

Figure 3.2: The Study Selection Process

Papers were relevant for inclusion in the study if they met the following criteria: (1) investigate concepts/problems/solutions of engineering in software startups, (2) present contributions in the form of lessons learned, framework, guidelines, theory, tool, model, or advice as applied in Paternoster et al. (2014), (3) are not included in any of the previous mapping studies, and (4) studies are written in English. The papers that were selected are scientific peer-reviewed articles, which is independent of the role of authors. We did not find experience reports from entrepreneurs, which might make the sample of papers lean towards the researcher community. To decrease the number of papers into a manageable amount, workshops, and papers based on expert opinion were excluded from the review process.

As common for systematic mapping studies (Petersen et al., 2008), the study focuses on synthesizing empirical research. Empirical studies are important for evidence-based software engineering research and practice, and for generating a knowledge base leading to accepted and well-formed theories (Kitchenham, 2004; Shull et al., 2007). The study provides an overview of empirical research on software startup engineering to date, and how research has evolved and possibly matured over the time period.

The retrieved papers were examined by the first and second author, where each author separately reviewed the papers based on titles and abstracts. Disagreements were resolved by discussion of the full text of the relevant papers. This was necessary as some of the abstracts were incomplete or poor. At this stage another 8 papers were excluded, making the total of newly selected papers 20, before performing the additional manual search.

3.1.4 Manual Search

A manual search was conducted using the forward snowballing technique (Wohlin, 2014), to identify additional papers not discovered by the search string. Google Scholar was used to examine the citations to the paper being examined. The publication lists of frequently appearing authors were also searched. This resulted in several papers as candidates for inclusion. After assessing title, abstract, and finally the full text, 7 more papers were included as primary studies (Nguyen-Duc et al., 2017b; Nguyen-Duc and Abrahamsson, 2016; Bajwa et al., 2017, 2016; Nguyen-Duc et al., 2016; Nguyen-Duc and Abrahamsson, 2017; Nguven-Duc et al., 2017). Among the papers, 21 were conference papers, five were journal papers, and one was a book chapter.

3.1.5 Quality Assessment

To build on previous work, a quality assessment of the new primary papers providing empirical evidence was performed. The total number of eligible papers was 22 (table 3.3). Although systematic mapping studies usually don't evaluate the quality of each paper in such depth as systematic literature reviews, the quality assessment process was undertaken to assess how results were presented in the primary studies. No paper was excluded based on the quality assessment.

To assess the rigour, credibility, and relevance of the papers, we adopted the quality assessment scheme from Nguyen-Duc et al. (2015a). Quality assessment has been identified as important for performing empirical research in software engineering (Kitchenham, 2004; Runeson and Höst, 2009). Table 3.2 illustrates 10 quality evaluation criteria. For each criteria the papers met, they got a score of 1, and otherwise 0. This means that the maximum score a paper could get was 10. A score of 0-3 was regarded as low rigour, 4-6 medium rigour, and 7-10 high rigour.

Problem Statement
Q1. Is research objective sufficiently explained and well-motivated?
Research Design
Q2. Is the context of study clearly stated?
Q3. Is the research design prepared sufficiently?
Data collection
Q4. Are the data collection & measures adequately described?
Q5. Are the measures and constructs used in the study the most relevant for answering the research question?
Data analysis
Q6. Is the data analysis used in the study adequately described?
Q7a. Qualitative study: Are the interpretation of evidences clearly described?
Q7b. Quantitative study: Are the effect size reported with assessed statistical significance?
Q8. Are potential alternative explanations considered and discussed in the analysis?
Conclusion
Q9. Are the findings of study clearly stated and supported by the results?
Q10. Does the paper discuss limitations or validity?

Table 3.2: Quality Assessment Checklist (Nguyen-Duc et al., 2015a)

3.1.6 Data Extraction and Synthesis

After the quality assessment, we defined the classification schema (table 3.3). Data from each of the 27 newly selected primary studies were then systematically extracted into the classification schema, according to the predetermined attributes: (1) SWEBOK knowledge area, (2) Research method, (3) Contribution type, (4) Pertinence, (5) Term for “startup”, (6) Incubator context, (7) Publisher. The chosen attributes were inspired by previous mapping studies (Paternoster et al., 2014; Klotins et al., 2015; Tripathi et al., 2016), and from the process of finding keywords in the abstracts of the retrieved papers (Petersen et al., 2008). Organizing the findings into tabular form enabled for easy comparisons across studies and time periods. In addition to classifying the papers, each paper was scanned for thematic concepts to identify researchers’ descriptions of investigated startups. The thematic concepts were adopted from the recurring themes found in Paternoster et al. (2014). This made it possible to assess the agreement in the community to the definition of startups.

The software engineering book of knowledge (SWEBOK) was created to provide a consistent view of software engineering, and to set the boundary of software engineering with respect to other disciplines (Bourque and Fairley, 2014). SWEBOK contains 15 knowledge areas that characterize the practice of software engineering. The focal point of the paper is to propose research directions based on the knowledge areas following the work done by existing literature. Since the two major mapping studies in the area follow different approaches, that is SWEBOK (Klotins et al., 2015) and focus facets (Paternoster et al., 2014), in the present study we focus on KAs, as this can allow the reader to better comprehend how the two different approaches are connected, thus offering a more holistic understanding of the current status in software startup engineering research. Assigning each paper into the specific knowledge areas was done by the first and second author. Both authors read the titles, keywords, abstracts, and the body of each paper, before evaluating the papers' conformance with each specific knowledge area's description or subareas.

ID	Research Method	Contribution Type	Knowledge Area	Pertinence	Term for startup	Incubator context	Publisher
(Yau and Murphy, 2013)	Case study	Lessons learned	Process	Full	Startup	No	Penn University
(Laporte et al., 2014)	Experiment	Lessons learned	Professional Practice, Management, Quality	Partial	Start-up	No	QUATIC
(Eloranta, 2014)		Framework	Professional Practice	Full	Start-up	No	ACM
(Laporte et al., 2015)	Experiment	Guidelines	Process	Full	Start-up	No	ENASE
(Yli-Huumo et al., 2015)	Case study	Theory	Process, Management, Quality	Partial	Startup	No	Springer
(Edison et al., 2015)	Survey	Tool	Construction	Full	Startup	No	Springer
(Nguyen-Duc et al., 2015b)	Case study	Model	Process, Management	Full	Startup	No	ACM
(Sánchez-Gordón and O'Connor, 2016)	Case study	Lessons learned	Process	Full	Very Small Company	No	Springer
(Wasserman, 2016)		Guidelines	Process	Partial	Startup	No	Springer
(Unterkalmsteiner et al., 2016)		Advice	Process, Management, Professional Practice, Construction, Requirements, Quality, Testing	Full	Startup	No	EISEJ
(Terho et al., 2016)		Advice	Quality, Construction	Full	Startup	No	Springer
(Laporte and O'Connor, 2016)	Case study	Lessons learned	Process, Management	Partial	Very Small Enterprise	No	IEEE
(Giardino et al., 2016)	Design and creation	Model	Models and Methods	Full	Startup	No	IEEE
(Bajwa et al., 2016)	Case study	Lessons learned	Management, Requirements	Full	Startup	No	Springer
(Nguyen-Duc and Abrahamsson, 2016)	Case study	Lessons learned	Management, Design, Construction	Full	Startup	No	Springer
(Nguyen-Duc et al., 2016)	Case study	Model	Methods and Models, Management	Full	Startup	No	IEEE
(Nguyen-Duc and Abrahamsson, 2017)	Case study	Advice	Managenet, Process	Full	Startup	No	EASE
(Bajwa et al., 2017)	Case study	Lessons learned	Management, Testing	Full	Startup	No	Springer
(Nguven-Duc et al., 2017)	Case study	Theory	Management, Process	Full	Startup	No	Springer
(Nguyen-Duc et al., 2017b)	Case study	Lessons learned	Management, Design	Full	Startup	No	Springer
(Pompermaier et al., 2017)	Case study	Lessons learned	Process, Testing	Full	Startup	Yes	KSI Graduate School
(Pantiuchina et al., 2017)	Survey	Lessons learned	Professional Practice	Full	Startup	No	Springer
(Rafiq et al., 2017)	Case study	Lessons learned	Requirements	Full	Startup	No	IEEE
(Souza et al., 2017)	Case study	Model	Professional Practice	Full	Startup	Yes	IEEE
(Chicote, 2017)		Framework	Quality	Full	Startup	No	IEEE
(Chanin et al., 2017)	Case study	Lessons learned	Requirements	Full	Startup	No	IEEE
(Marks et al., 2017)	Case study	Advice	Process	Full	Start-up	No	Springer

Table 3.3: Classification Schema

3.1.7 Threats to Validity

There are several threats to the validity of systematic mapping studies (Zhou et al., 2016). One threat is related to the data extraction from each paper, where results can be biased from researchers personal judgement. To mitigate this threat, and ensure correct classification of each paper into the SWEBOK knowledge areas, this process was performed jointly by the authors at one computer, resolving any conflicts and regulating individual bias.

Threats to the retrieval of relevant papers must also be considered. The inconsistent use of terms for “startup” made it difficult to cover all used terms in the search string. Hence, it appeared terms not considered when constructing the search string. Some of these were “founder teams”, “very small enterprise”, “very small entity”, and “very small company”, which all were used in relation to the startup context. Relevant papers might therefore have been overlooked.

The use of only three bibliographic databases might have affected the number of relevant papers retrieved. Compared to the number of databases used in similar studies, this seems to be at the low-end. The chosen databases are however among the most used ones in the field of software engineering, and the databases that contributed with the most retrieved papers in other studies (Paternoster et al., 2014). The risk of missing papers published the last five years was mitigated by the use of forward snowballing which lead to the retrieval of seven more papers.

To make sure the study selection was not biased from personal opinions, paper selection involved both authors, which allowed a collaborative resolution of conflicted views, following guidelines from Kitchenham Kitchenham (2004). We defined clear inclusion and exclusion criteria, and a quality assessment checklist to assess each paper’s quality. Disagreement to quality assessment was discussed between the authors until consensus was reached. This decreased the risk of miss-classifying any relevant papers.

For the quality assessment, we only used two points to collect answers, as the authors were unfamiliar with the research field. The papers got a score of 1 if they met the criteria, and otherwise 0. Prior studies have used a more fine-grained classification of quality criteria, and even used different criteria in some occasions. It is more likely that the papers in our study obtained higher rigour than they would have received if another more fine-grained assessment method had been used.

3.2 Synthesized Results

This section presents a summary of the results section from the systematic mapping study and is divided into the research questions defined in section 3.1.1.

3.2.1 RQ1: How has software startup research changed over time in terms of focused knowledge areas?

Figure 3.3 shows the number of studies published in relation to software startup engineering between 1994-2017, constituting a total of 74 published papers. We observe that the publication frequency of papers between 2013-2017 is higher than for any period before 2013. From 1994-2013, the highest number of primary papers within a single year was 7 (2008). In comparison, 2016 and 2017 constituted 9 and 11 papers respectively.

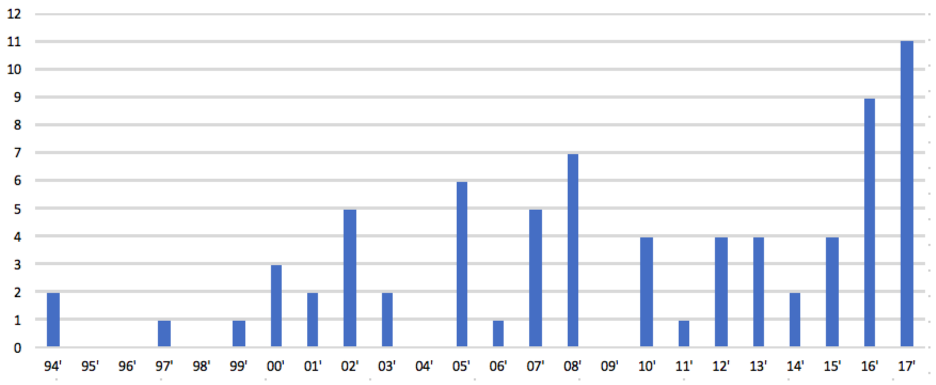
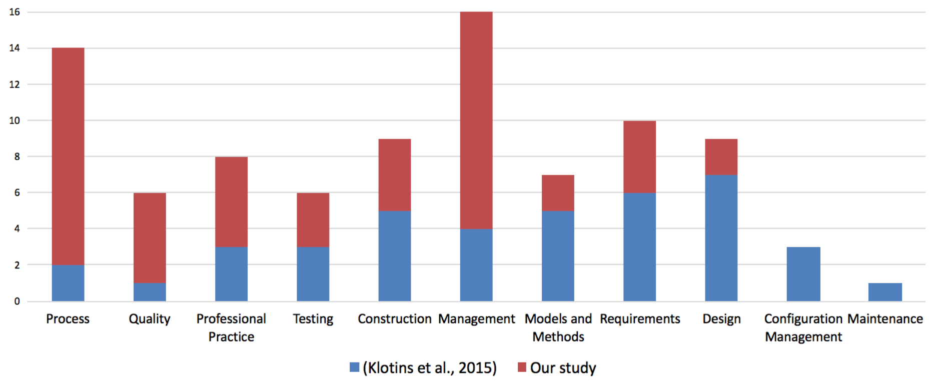


Figure 3.3: Publication Frequency, 1994-2017

The SWEBOK knowledge areas make up 15 categories developed by the software community as a baseline for the body of knowledge within software engineering (Bourque and Fairley, 2014). Figure 3.4 shows the number of papers covering the different knowledge areas from the specified time period. The change of research direction is illustrated by the shift from research focused on software design and software requirements, between 1994-2013, towards more research within software engineering management and software engineering process, between 2013-2017.



.

Figure 3.4: Knowledge area coverage

Paternoster et al. (2014) classified studies into contribution types, originally suggested by Shaw (2003) as a best practice for classifying software research. Figure 3.5 shows that the most frequent contribution types are advice, lessons learned, and models. Tools, guidelines, and frameworks have received little attention in research.

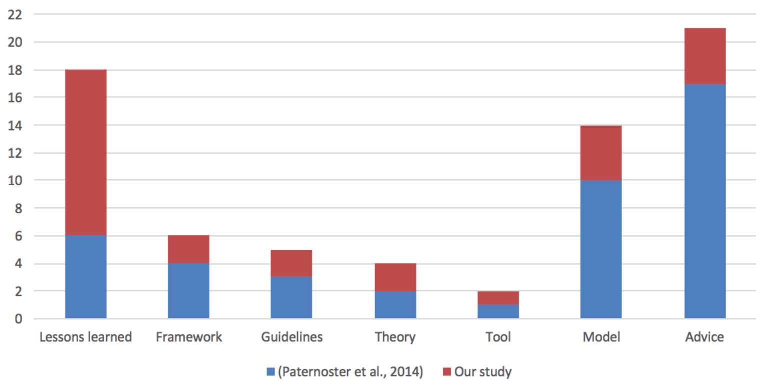


Figure 3.5: Contribution types

3.2.2 RQ2: What is the relative strength of the empirical evidences reported?

Figure 3.6 shows the degree of rigour within each knowledge area between 2013-2017. The x-axis represents the knowledge areas, while the y-axis represents the rigour. Only one paper received low rigour score (Edison et al., 2015), as it didn't provide enough details about the data analysis and included no assessment of the validity of the results. However, as only the papers providing empirical evidence were assessed, it is possible that more papers would receive low rigour as well. In general, the papers received high rigour score, indicating that the quality of research was high.

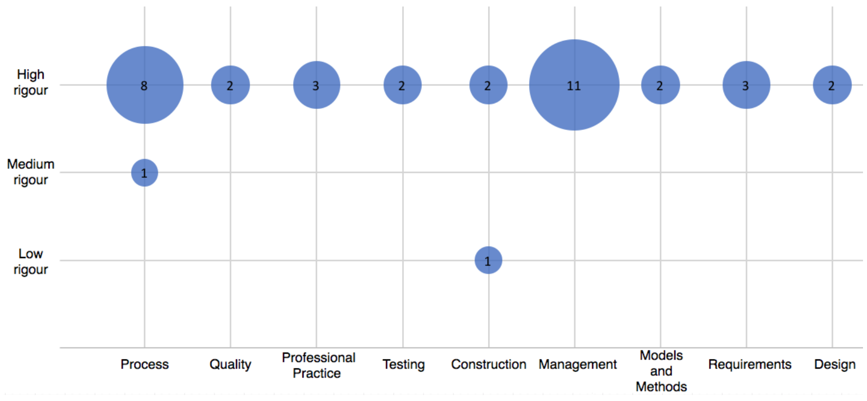


Figure 3.6: Rigour of each covered knowledge area, 2013-2017

Figure 3.7 shows the rigour of the primary studies from Klotins et al. (Klotins et al., 2015), and which research type each constituted. The paper did not specify how they calculated the rigour of each paper. The x-axis represents the research types, and the y-axis represents the rigour. From 14 primary papers, only one provided a contribution of high rigour. Most of the papers (86 percent) obtained low rigour. As to this, the paper concludes that the low rigour of the papers, due to poor contextual descriptions, makes it hard to transfer results from one environment to another.

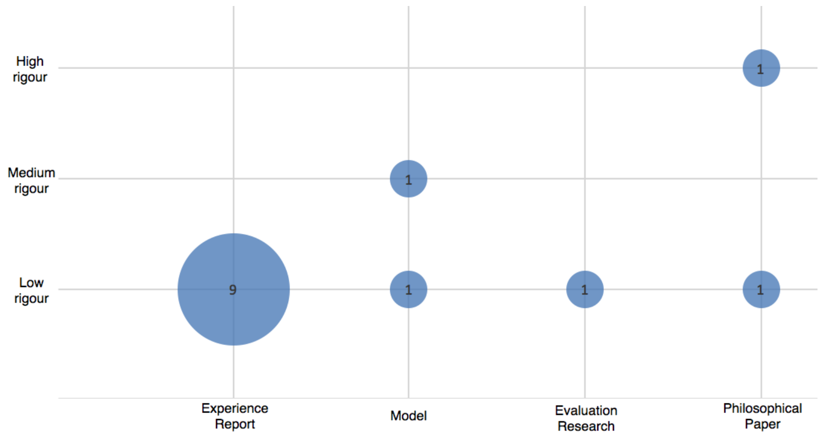


Figure 3.7: Rigour and research type (Klotins et al., 2015)

Figure 3.8 illustrates the rigour of the contribution types provided by each of the primary papers in Paternoster et al. Paternoster et al. (2014). The x-axis represent the contribution types, and the y-axis represents the rigour. The division of rigour score is based on table 7 in the study. Papers that got a total score above 7 received high rigour, between 4 and 7 received medium rigour, while less than 4 received low rigour. 70 percent of the papers in figure 3.8 received a medium score, while 21 got a high score.

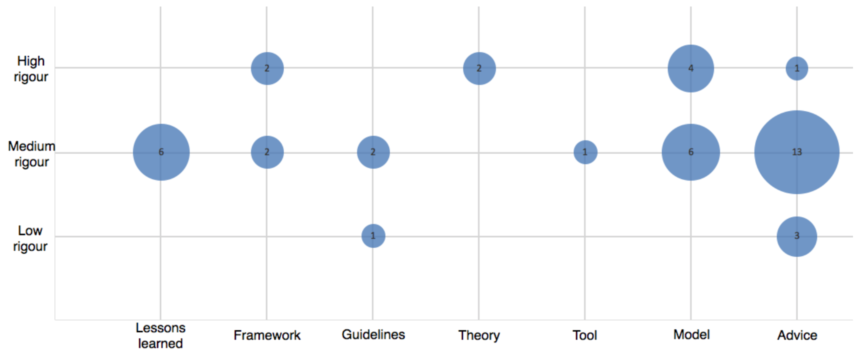


Figure 3.8: Rigour and contribution type (Paternoster et al., 2014)

3.2.3 RQ3: In what context has software startup research been conducted?

Table 3.4 presents a complete usage of thematic concepts operating startup research between 1994-2017. We observe that the characterizations have changed over time (e.g., the most frequently used concept before 2013 was only the fourth most used one after 2013). The differences are significant since it is only four years between the studies. The use of concepts between 2013-2017 is highly inconsistent. There is no single concept that all the 22 empirical papers use for the startups they investigate. The low frequencies of the thematic concepts also illustrate that many of the papers provide poor startup descriptions.

Thematic Concepts	Frequency 13'-17' (#27)	Frequency 94'-13' (#47)
Innovation/Innovative	15	19
Uncertainty	14	15
Small team	11	12
Lack of resources	9	21
Little working/operating history	9	3
Time-pressure	7	17
Rapidly evolving	5	16
New company	5	8
Highly reactive	3	19
Highly risky	3	8
Third party dependency	2	12
One product	2	9
Not self-sustained	1	3
Low-experienced team	0	9
Flat organisation	0	5

Table 3.4: Thematic Concepts, 1994-2017

The primary studies from 2013-2017 that have provided empirical evidence and sufficient contextual descriptions are presented in table 3.5. The relevant context information includes the attributes: (1) number of startups under investigation, (2) size of the company/team, (3) the product orientation of the startups, and (4) other relevant contextual descriptions beyond these three (e.g., lifecycle stage, age/year of establishment, location, software development methodology).

ID	Nr of startups	Company size	Product orientation	Other relevant info
(Yau and Murphy, 2013)	1 startup	5 members	Social network application	Roles: Designer, 1 web/iOS/android dev. each, CEO Approach: Lean Startup/Agile
(Rafiq et al., 2017)	3 startups	4 members	Health	Canada, mobile app, concept stage
		6 members	E-commerce	Italy, mobile and web app, func.stage
		25 members	E-commerce	Brazil, web app, mature stage
(Souza et al., 2017)	4 startups	12 members	Academic business domain	3yrs old
		10 members		1yrs old
		8 members		1yrs old (still incubated)
		10 members		4months old (still incubated)
(Marks et al., 2017)	1 startup	Not specified	Db performance & interoperability	High potential growth firm, spin-out from a university
(Bajwa et al., 2016)	4 startups	2 founders	Video service	Working prototype (14')
		3 founders	SaaS	Func. product, limited users (15')
		2 founders	Event ticketing system	Func. product, high growth (11')
		2 founders	Game-based learning	Mature product (06')
(Nguyen-Duc and Abrahamsson, 2016)	5 startups	6 members	Online photo marketplace	Italy (lean startup/agile,12',impl.phase)
		3 members	Marketplace for food hub	Norway (ad-hoc,15',concept.phase)
		4 members	Collab.platform construction	Norway (Scrum,11',commercial.phase)
		18 members	Sale visualization	Norway (agile,11',commercial.phase)
		3 members	Under-water camera	Finland (ad-hoc,11',impl.phase)
(Nguyen-Duc et al., 2016)	5 startups	20 members	Learning game, B2C	2006, scaling phase
		18 members	Real-time sale management, B2B	2011, scaling phase
		1 member	Photo marketplace, B2C	2012, startup
		3 members	Social platform,B2C	2015, pre-startup
		1 member	Collab.platform construction, B2B	2011, startup
(Nguyen-Duc and Abrahamsson, 2017)	6 startups	6 members	Hyper-local news platform, P2P	Norway (agile,2015,bootstrap)
		9 members	Collab.platform construction, B2B	Norway (scrum,2012,bootstrap)
		3 members	Ticket event system, B2B	Norway (agile,2012,bootstrap)
		5 members	Shipping platform, P2P	UK (agile,2013,early investor)
		12 members	Game learning tool, P2P	UK (dist.agile,2013,bootstrap)
		5 members	Fish farm management, B2B	Vietnam (ad-hoc,2016,bootstrap)
(Laporte et al., 2015)	2 startups	4 members	Not specified	Peru (2012, VSE/start-up term)
		2 members		Canada
(Nguyen-Duc et al., 2015b)	3 startups	4 members	Photo market place, P2P	2011,paying customers
		5 members	Under-water camera, B2B	2009,paying customers
		12 members	Ticketing system, B2P	2011,paying customers
(Sánchez-Gordón and O'Connor, 2016)	3 VSEs	17 members	Enterprise	18yrs active (int.customers)
		10 members	Financial services	9yrs active (int.customers)
		7 members	Enterprise	4.5yrs active (int.customers)
(Giardino et al., 2016)	13 startups	3-20 members 2-6 founders	Not specified	Time-to-market: 1-12months
(Pompermaier et al., 2017)	8 startups	Not specified	Not specified	Incubator-context 62 % used pseudo-agile for reqs. 100 % not documenting many reqs.
(Chanin et al., 2017)	3 startups	Not specified	Food-waste knowledge app	Not specified
			Online debt platform	
			Online investment platform	

Table 3.5: Contextual Descriptions, 2013-2017

3.3 Conclusion

We have applied a systematic mapping method to analyze the literature related to software startup engineering. A total number of 74 primary papers (in which 27 papers are newly selected) were extracted and synthesized. Our study, along with the previous mapping studies, constitute a merging of the primary literature within the field for the period 1994-2017, including the focus and relative strength of research, and the effort that's been made to characterize the software startup context.

The contribution of the mapping study is two-fold. Firstly, the study provides a comprehensive view of software startups for Software Engineering researchers. Possible research gaps are derived for future studies. Secondly, the study provides a map of the contextual setting of investigated startups, inferring the applicability area of empirical findings. This can help to compare and to generalize future research in software startups.

Regards to RQ1, most found software startup research between 2013-2017, are conducted within software engineering management and software engineering process, while software design and software requirements have received most attention between 1994-2013. For the period 2013-2017, software design received far less contributions compared to that in 1994-2013, illustrating a change of research direction. The knowledge areas software engineering models and methods, software quality, and software testing have received little attention from the research community during the period 1994-2017. Apart from these findings, we emphasize the need for more research within all knowledge areas. For the period 2013-2017 software configuration management and software maintenance were not covered at all. As to this, it seems that some of the knowledge areas aren't directly relevant to the startup context. Future mappings should instead use the newly established research themes of Unterkalmsteiner et al. (2016).

Regards to RQ2, we found an increased rigour of primary studies after 2013 in comparison with studies found in 1994-2013. While it is still not clear about the transformation of research results to startup practitioners, startup researchers seem to increase the focus on conducting high-quality research. The increased importance of startups has been an important factor to highlight the need for more research. As startups generally use ad-hoc or opportunistic development methods, practices of startups can be different, meaning that more evidence is needed to generalize work practices to all startups.

Regards to RQ3, we identify a coherent set of concepts that represent the startup context, (1) Innovation/innovative, (2) Lack of resources, (3) Uncertainty, (4) Time-pressure, (5) Small team, (6) Highly reactive, (7) Rapidly evolving. Additionally, aspects like (1) team size, (2) product orientation, (3) number of active years/life cycle stage, (4) number of investigated startups, (5) location, and (6) development method are important to describe sufficiently to be able to transfer results from one environment to another. As only 14 of the primary papers between 2013-2017 provided adequate descriptions, and all primary papers showed an overall inconsistent use of describing thematic concepts, we see a need for a more comprehensive endeavor to describe the engineering context of startups.

Several threats to validity were considered, including the selection of papers, the use of few online bibliographic databases, the selection of keywords, and the coarse-grained classification used for the quality assessment. To ensure the selection process was unbiased, the selection criteria were developed in advance, also the first, second and third author were involved in the selection process. Both the use of few online bibliographic databases, and the identified keywords and search terms might have lead to relevant papers being omitted. This risk was mitigated by performing an additional manual search. For the quality assessment it is likely that the use of only two points have caused the papers to obtain a higher rigour than they would have if a more fine-grained assessment method had been used.

Future work can focus on certain research themes, such as startup evolution models and human aspects, and consolidate the contextual factors of software startups. More work should be conducted for specific business contexts, such as startups that are part of incubators and bigger business ecosystems. As a next step, we seek to address engineering practices in startups who deliver both hardware and software, as no prior studies have been entirely dedicated towards their specific challenges and demands.

Chapter 4

Research Method

Software engineering research is to a great extent concerned with investigating the development, operation, and maintenance of software products. The objective of this study is to *create a better understanding of work-practices in hardware startups by investigating the role of engineering activities, from idea conceptualization to a launched product. In particular, we will investigate factors influencing development speed and agility, and explore commonalities, challenges and situational factors.* We propose a model visualizing the overall engineering approach of hardware startups, serving as a fundament for researchers and practitioners to further explore the hardware startup context. The case study process is considered suitable for such multidisciplinary areas where existing theory may be inadequate (Runeson and Höst, 2009), and so we have designed a case study protocol inspired by Pervan and Maimbo (2005) to guide the collection and analysis of data. The study is of exploratory nature as we seek to create knowledge by investigating events and actions of those who experience them (Oates, 2005).

Several data generation methods were considered (Oates, 2005). Surveys allow for generalization of results, however, the large number of participants required, and the time constraints of this project made surveys an infeasible option. Observations were considered too time-consuming to undertake. Semi-structured interviews of selected participants fitted both the time-constraints and availability of hardware startups and is considered suitable for qualitative data analysis (Oates, 2005). Interviews allowed for a discoverable approach, as interviewees could express themselves more freely and provide their own perspectives on personal experiences related to the research topics.

The rest of this chapter introduces the case study protocol, and proceeds as follows: Section 4.1 introduces and justifies the research questions of the empirical research. Section 4.2 explains how we developed interview questions based on the research questions. Section 4.3 presents the case and subject selections. Section 4.4 explains how the interviews were conducted. Section 4.5 describes the qualitative data analysis process. Section 4.6 presents the steps taken to validate our research. Section 4.7 discusses validity threats to the research design. Finally, section 4.8 explains how we managed ethical considerations. Figure 4.1 illustrates all steps of the research process.

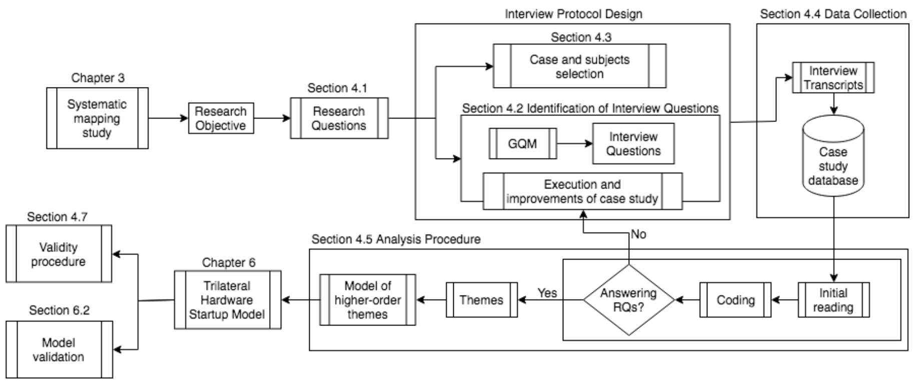


Figure 4.1: Research Process

4.1 Research Questions

From the systematic mapping study, we identified several gaps in the literature on *software startup engineering*, one of them being the rare knowledge on development processes in hardware startups. Among publications the last five years, several of them investigated startups developing products with mixed software and hardware parts, however, there was a lack of validated knowledge focusing on their specific challenges or practices. The context of hardware development poses many constraints and dependencies that affect practices and processes (Ronkainen and Abrahamsson, 2003). Hardware development is considerably more complex than software development, and so there is a need for methodologies and practices that cover the specific concepts for hardware startups (Stock and Seliger, 2016).

The technological lifecycle of companies delivering new innovative electronic products is becoming shorter each year. The high demands and accelerating adoption rates of customers, and the pressure for reduced time-to-market increase the uncertainty related to the product development processes of hardware startups. In such a context the combination of speed and agility is essential for delivering innovative customer-driven products (Bosch, 2016). Speed is fundamental for staying alive in highly competitive markets, where creativity needs to be combined with agility to handle uncertainty, and introduce flexibility in the process (Garbajosa et al., 2017). At the same time, speed needs to be managed with caution to avoid expensive rework and bringing the development process to a halt. We aim at exploring how agility, speed, and quality are engineered by practitioners in hardware startups. This has motivated the following research questions:

RQ1 How do hardware startups achieve agility during product development?

RQ1.1 How do hardware startups develop their products?

RQ1.2 What kind of challenges are relevant in the hardware startup context?

RQ1.3 How do internal/external context factors impact the speed of product development?

RQ2 How do hardware startups manage quality concerns of their products?

RQ2.1 How are hardware products tested?

RQ2.2 How is technical debt managed in hardware startups?

RQ3 How do hardware startups achieve balance between speed and quality?

From the empirical data obtained from the multiple-case study and knowledge from the *Greenfield Startup Model*, we will develop a model describing the engineering approach of hardware startups. The resulting model will explain the priorities of hardware startups, and why introducing process and specific methodologies is hard. The model will be the results of an early investigation of how hardware startups operate and point out opportunities for future research.

Figure 4.2 illustrates the relationships of the formulated research questions, and how they together with the Greenfield Startup Model contribute to the creation of the Trilateral Hardware Startup Model.

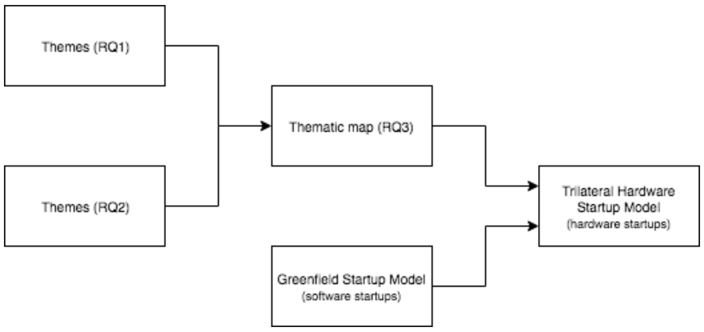

Themes (RQ1)
Themes (RQ2)
Thematic map (RQ3)
Greenfield Startup Model (software startups)
Trilateral Hardware Startup Model (hardware startups)

Figure 4.2: Relationship between research questions and theoretical models

4.2 Identification of Interview Questions

To create a mapping between the research questions and the metrics used to address the questions, we have used the Goal Question Metric (GQM) paradigm (Basili, 1992). Even if the original purpose of the GQM approach was to “define and evaluate goals for a project in a particular environment” (Basili, 1992), the concepts are generic and its use has been expanded to other measurement settings. The paradigm defines a measurement model that can be divided into three main steps: (1) *conceptual level* where an overall objective is defined, (2) *operational level* where a set of questions are defined to achieve a specific objective, and (3) *quantitative level* where a set of metrics are associated with every question for answering it in a measurable way. Further steps include (4) define data collection mechanisms, and (5) collect, validate, and analyze the data in real-time to ensure conformance to the goals and make recommendations for future improvements. This section covers the three first steps of the GQM process in light of our study (i.e., how the research questions of the empirical research are mapped to the interview questions (Appendix A.1)).

The overall objective of this study is to create a better understanding of work-practices in hardware startups by investigating the role of engineering activities, from idea conceptualization to a launched product. To address this research objective, we have defined three research questions with several sub-questions (section 4.1). The next step of the GQM process is to define appropriate measures for the research questions. As this study is a qualitative case study through semi-structured interviews, the interview questions will be the metrics.

Figure 4.3 presents the mapping between the research questions and the associated interview questions. Each interview question is only represented once, with its related code as found in Appendix A.1. The figure illustrates that most interview questions participate in answering several research questions, emphasizing their correlation.

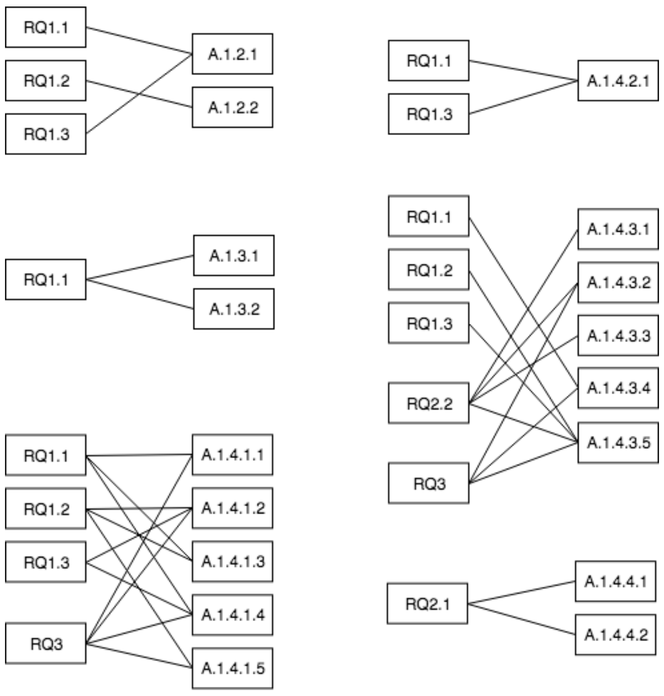


Figure 4.3: Mapping between questions and metrics

4.3 Case and Subjects Selection

The units of analysis are people involved in product development in startup companies that deliver products with mixed hardware and software parts. We defined selection criteria as suggested by Runeson and Höst (2009). Startups were relevant for inclusion in the study if they met the following criteria:

- The startup develops both hardware and software parts.
- The startup has been active for at least six months.
- The startup has a first running prototype.
- The startup's ambition is to scale its business.

People from the relevant startups were eligible for participation if they had experience and/or knowledge about software and/or hardware development. If the candidate met the criteria, he/she was regarded as qualified for contributing to the research study.

We used five different channels to find relevant startups: (1) Innovation Center Gløshaugen, (2) NTNU Accel and FAKTRY, (3) our supervisors' professional networks, (4) OsloTech and StartupLab, and (5) The Hub. Table 4.1 provides an overview of the different communication channels and can help other researchers to find and contact startups.

Channel	**Description**	**Link**
Innovation Center Gloshaugen	The center is located at campus Gloshaugen, and houses various early-stage high-tech startups, mainly to support innovative students.	www.ntnu.no/ig
NTNU Accel and FAKTRY	NTNU Accel is a uni-based accelerator for promising startups. FAKTRY is an incubator which is part of Accel, and houses various hardware startups.	www.ntnuaccel.no, www.faktry.no
Supervisors' professional networks	Letizia Jaccheri: Italian companies (S13) Anh Nguyen-Duc: European and Asian startups Javier Escribano: Spanish and Dutch companies (S11)	
OsloTech and StartupLab	OsloTech manage Oslo Science Park, including incubator StartupLab which has supported more than 200 startups since 2012.	www.oslotech.no www.startuplab.no
The Hub	The Hub is a community platform which gives an overview of Norwegian and Nordic startups. Via the platform, startups can get assistance with recruitment and connection with investors.	www.hub.no

Table 4.1: Startup Channels

We have provided comprehensive contextual descriptions of each case, identified as important to ensure that results are of high rigor and possible to transfer between similar environments (Cruzes and Dyba, 2011). This will allow practitioners to judge the generalizability of the results, and researchers to assess its validity. A textual description of each case can be found in section 5.

4.4 Data Collection Procedure

Our chosen data generation method was interviews, identified as an efficient method for answering research questions in case studies (Oates, 2005). The semi-structured approach enabled discovery of unforeseen information as interviewees could express themselves more freely, and fitted both the time constraints of the project and the availability of startup companies.

The researchers were in direct contact with the subjects, hence the data collection process can be regarded as a first degree data collection technique. First degree data collection requires a significant effort, but allowed both researchers to control what data was collected, ensuring that all pre-defined interview questions were answered sufficiently and exploring new directions by asking follow-up questions (Runeson and Höst, 2009). Both authors attended all interviews to avoid one single interpretation of the respondent's perspective and insight on topics, as qualitative data often can be rich and broad, but less precise.

The interviews were undertaken in the language preferred by the interviewee (English or Norwegian). Several of the interviews were therefore undertaken in Norwegian as this made the interviewees more comfortable. This allowed them to express themselves more freely, and give more in-depth explanations. Because of this, it was necessary to translate some of the interviews when transcribing. As there often doesn't exist a one-to-one relationship between language and meaning (Temple and Young, 2004), the translation of the transcribed interviews was ensured to "express all aspects of the meaning in a manner that is understandable" (Larson, 1991). This implies that not all parts of the interviews were directly translated word-for-word.

All interviews followed the interview guideline (Appendix A.1), which is structured into four main parts. The question were categorized into (1) general information, (2) business background, (3) startup development methodologies, and (4) product development. Each category has several associated questions to ensure sufficient coverage of all topics. All interviews were recorded. Participants signed consent forms (Appendix A.3) before participation. The selection criteria were reviewed by our supervisors to ensure the quality of the study design. All interviews were performed between February and April 2018.

Before the interviews, we looked into the cases' business background, either through their company websites or other relevant incubator or accelerator websites. Additionally, most participants answered a simple questionnaire (Appendix A.2) prior to interviews where they filled out basic information about themselves and the company. These measures allowed for more efficient interviews as the interviewers had more knowledge about the case and could use less time on initial formalities. Since each startup only was interviewed once, it was crucial to perform initial company analysis to get a holistic understanding of each case and to provide stronger evidence for the conclusions drawn from the interviews.

Table 4.2 presents the details of the interviews. All interviews were transcribed shortly after they were conducted. In addition, both authors made initial reflections. These were made to take note of the most important topics from the interview, and for making improvements to the interview questions before the next interview.

Case	Type of Interview	Interview Subject	Duration
Cable cam system	Face to face, not on-site	CTO	40 min
Unmanned aircraft system	On-site	Hardware developer	55 min
Smart gloves	Face to face, not on-site	CEO	35 min
Medtech biosensor	On-site	CEO & software developer	55 min
Physical exercise game	On-site	CTO	40 min
LPG management system	On-site	CEO	40 min
Advanced noise cancellation	On-site	Hardware developer	40 min
Medtech hydration monitoring	On-site	CSO	25 min
Collaborative camera	On-site	Hardware developer	35 min
Digital piggy bank	On-site	CEO	40 min
Interactive children's toy	Skype	CFO	35 min
3D-printer control board	On-site	CEO	25 min
Sensors for IoT	Face to face, not on-site	Hardware developer	25 min

Table 4.2: Case Interviews

4.5 Analysis Procedure

Thematic analysis can be defined as "a method for identifying, analyzing, and reporting patterns (themes) within data" (Braun and Clarke, 2006). We applied the thematic synthesis process which is a codes-to-theory model for qualitative research (Cruzes and Dyba, 2011). The objective of our thematic synthesis process is to both answer the research questions and come up with a new model describing development strategies in hardware startups, focusing on aspects that are unique from software startups. The main steps of the process are illustrated in figure 4.4. The rest of this section explains our data analysis procedure.

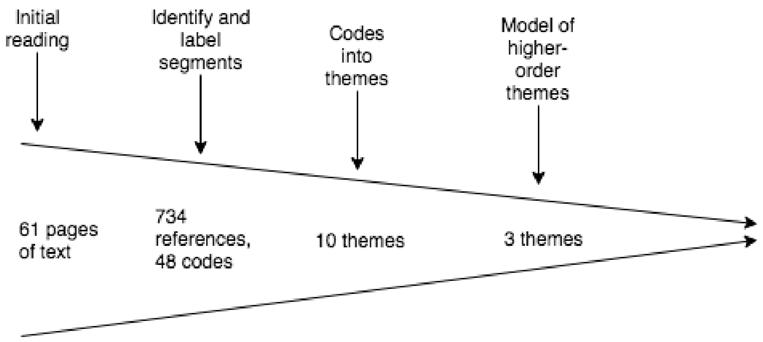


Figure 4.4: Thematic Synthesis Process (Cruzes and Dyba, 2011)

4.5.1 Initial Reading

The first step of the analysis process was to read through the transcribed interviews to generate initial ideas and identify possible patterns in the data. All interviews were transcribed shortly after they were conducted to ensure the actual meaning of interviewees' answers. Both authors discussed the interviews, creating a mind map of central concepts relevant to hardware startups. We also assessed the categories and themes of the GSM to understand the differences to the software context.

The mapping between research questions and interview questions quickly allowed us to address whether interviews provided relevant answers for best addressing the research questions. This facilitated an early and efficient analysis, as we could connect respondents' answers directly to our research objective. This was necessary for making incremental improvements to the interview protocol.

4.5.2 Coding Process

Coding can be seen as the first step of data analysis (Seaman, 1999). To generate initial codes, we applied a descriptive coding technique (Saldaña, 2015). This technique is about summarizing in a word or short phrase the basic topic of the data, to identify interesting concepts, categories, or other findings in a systematic way across the data set. Descriptive coding is useful for inexperienced qualitative researchers and helped us organize and group similar data into categories, which is the first step towards the creation of themes.

The coding process followed an integrated approach (Saldaña, 2015), which is a mix of the inductive and deductive approaches. In inductive coding, data is reviewed line by line and as concepts appear, a code is assigned. In deductive coding, you have a start list of codes based on theories or other key concepts in which you categorize data into. Our approach was slightly more inductive than deductive; we had some clear ideas and thoughts of what we expected to find from the data, however, we created the codes as concepts appeared in the data. This approach allowed us to avoid coding data out of context, while at the same time identifying what the text was saying rather than what we wanted to see.

To help organize and keep an overview of data, we used NVivo, which is a software tool for qualitative data analysis. NVivo can allow for a better understanding and exploration of unstructured data by facilitating quick discovery of key topics and themes. Having in mind that both researchers were new to qualitative data analysis, and the limited time-frame of the research, NVivo was decisive in effectively creating valid and defensible outcomes.

We applied an iterative coding process, to allow for simultaneous data collection and analysis (Runeson and Höst, 2009). The first iteration involved coding the data from the four first interviews. A total of 29 codes were generated from 416 references. The codes were examined by both researchers and one supervisor. Lessons from the evaluation were implemented in the next interviews to generate relevant codes. For the second iteration, we classified text into the codes from the first iteration, while at the same time generating new codes in an inductive manner. The second iteration resulted in a total of 48 codes and 734 references from 13 interviews.

4.5.3 Translate Codes into Themes

A theme can be seen as a way of grouping initial codes into a smaller number of sets, to create a meaningful whole of unstructured codes (Cruzes and Dyba, 2011). The process reaches an end when no new themes emerge from the data, at the point of saturation. We divided related codes into categories and concepts (Strauss and Corbin, 1998). All cases were analyzed separately in relation to their respective case descriptions (section 5.1), to ensure that themes were in line with the associated context. NVivo facilitated independent coding of each case, while at the same time allowing us to classify data from each case into similar codes from other cases.

Hardware startups include a significant amount of software components and face similar business and organizational challenges as software startups. Because of this, we could expect that several of the themes appearing in the data would be matching those created in the GSM. This is a threat to validity, as we expected to find certain things instead of them appearing in the data. Since we are creating a new model describing the engineering approach of hardware startups we mainly focused on creating themes describing aspects that are unique to hardware startups. Some of the themes of the GSM were found to be more significant and even having a different implication for hardware startups. The total number of themes unique to hardware startups ended up being 10.

4.5.4 Model of Higher-Order Themes

The generated themes were further explored and interpreted to create a model of higher-order themes (Figure 5.1). We focused on data and themes unique to hardware startups. The higher-order themes were *third-party dependency, hardware-software integration,* and *two-folded product quality trade-off.* In addition, we identified patterns more general to the startup context. These were similar to several of the categories found in the GSM. Although the names refer to the same concepts, we named them differently from the GSM to avoid any misunderstandings for the reader.

The higher-order themes were in combination with knowledge from the Greenfield Startup Model consolidated to form a conceptual representation named the Trilateral Hardware Startup Model. To create the model we identified connections to *quality, speed,* and *resources*, operating as core elements of the model. Depending on the objective of the project, quality, speed, and resources are all elements that will affect product development. To ensure the correctness of the model, the magnitude of the impact was ranked to be either (1) weak, (2) medium, or (3) strong. Table 4.3 presents the strength of the relationships. Chapter 6 introduces the Trilateral Hardware Startup Model in further detail.

	Restricted resources	Team proactivity	Two-folded product quality trade-off	Third-party dependency	Hardware-software integration	Evolutionary prototyping	Rapid development	Incurred technical debt	Return effects of short-term benefits
Quality	1	1	2	1	1	1	1	3	2
Speed	1	2	2	2	2	3	3	1	2
Resources	3	2	1	2	2	1	1	1	1

Table 4.3: Themes' impact on core model elements

4.6 Validation Procedure

To validate our model's compliance with the hardware startup context, we performed two separate validation activities. The validation and it's results are described in further detail in section 6.2. Firstly, we attended the fortnightly meeting of the Software Startup Research Network, a global network of scientists within software startup research. In this meeting we held a 30-minute presentation, presenting the main findings of the Master thesis and our model describing the engineering approach of hardware startups. Secondly, we contacted all of the investigated cases to assess their conformance with the model. All of the participating startups received the model with the corresponding description. The startups were asked to explain to what extent they found the model useful, and how the model contributed to describing the hardware startup context. Their answers were used to create a mapping between the developed factors and cases (table 6.1), valuable for generalization of the model.

4.7 Validity Procedure

The validity must be addressed for all phases of the case study, to enable replication of our research (Runeson and Höst, 2009), and to ensure the trustworthiness of our findings (Cruzes and Dyba, 2011). The study is classified into four categories of validity concerns used for controlled empirical experiments in a software engineering context (Wohlin et al., 2003). The categories are (1) construct validity, (2) internal validity, (3) external validity, (4) conclusion validity.

4.7.1 Construct Validity

Construct validity relates to our research design, and whether it correctly measures engineering practices in hardware startups. To ensure that the interview questions were suitable for answering our research questions, we defined interview questions through a top-down approach using the Goal Question Metric method. Additionally, we identified topics of interest in hardware startups based on primary papers (e.g., research questions or research directions) from our Systematic Mapping Study on software startups (Chapter 3). The interview questions were revised by our supervisors to ensure that they were appropriate for addressing our research questions. It can be difficult to know beforehand what to focus on and how to perform interviews since relevant information might appear as you start analyzing interviews. To deal with this, we performed interviews in an iterative manner. We transcribed and analyzed the first four interviews before moving on with the nine remaining interviews. This allowed us to make improvements and adjustments to the interview protocol. To minimize deviating results, we conducted interviews in the startups' environments. In cases of restricted office space, we booked private group rooms at the university to reduce noise and disruptions.

4.7.2 Internal Validity

Internal validity is the extent to which bias is minimized and the conclusions are credible. We focused on interviewing software and hardware developers and other people close to the development processes. Some of the interviewees were CEOs, and although they have broad business perspectives, they might have limited expertise and insights into specific development practices. To decrease the risk of biased interpretations, both researchers attended all interviews. We discussed and analyzed answers straight afterward to catch any underlying meanings. We also compared findings to related literature (Giardino et al., 2016; Nguyen-Duc et al., 2018; Ronkainen and Abrahamsson, 2003), examining similarities, contrasts, and explanations. Such comparisons have proven to enhance internal validity and the quality of findings (Eisenhardt, 1989). To ensure that we have synthesized data appropriately and drawn reliable conclusions, we have performed a validation of the model of higher-order themes in section 6.2.

4.7.3 External Validity

External validity relates to our study's generalizability to similar environments. We performed interviews on a sample of hardware startups, based on selection criteria for participation. Interviews were either performed with CEOs, CTOs, or engineers, preferably associated with the companies from the start. The startups were mostly located in the same area, mainly consisting of young, inexperienced entrepreneurs. The study might not apply to all hardware startups. Generalization of the research results is therefore limited to cases with similar characteristics (i.e., early-stage European hardware startups). Case descriptions (section 5.1) allow for identification of key information about each startup and can help guide researchers and practitioners in transferring results to other cases.

4.7.4 Conclusion Validity

Conclusion validity is to which extent our conclusions are reasonable (i.e., the reliability of our study). Since some of the interviews were conducted in Norwegian, it was not always possible to directly translate transcriptions word-for-word. The associated risk of translating interviews is whether we were able to capture the actual meaning provided by the interviewees. We handled this by writing transcripts straight after each interview, as well as having both researchers participating in all interviews. It is difficult to understand a startup and its dimensions within a time-span of 30 minutes. To enhance our understanding of the investigated cases, we collected information about the startups through search engines, company websites, social media, and pre-interview questionnaires. Preliminary investigations enlarged our holistic understanding of the investigated cases. To enhance the reliability of the study all participating startups were included in the process of writing case descriptions to ensure its conformance with reality.

4.8 Intellectual Property Rights

Ethical issues are important to consider when planning and performing empirical research (Oates, 2005; Runeson and Höst, 2009; Singer and Vinson, 2002). Before conducting the interviews, all participants were asked to explicitly agree to participate through an informed consent. To ensure that our research is in line with Norwegian law, the template followed The Norwegian Protection Officer for obtaining consent (Appendix A.3), to inform interviewees that participation in the study is voluntary and that they have every right to withdraw from the study without further notice. The template is in line with the *Policy for the protection and management of intellectual property rights and physical material at NTNU*.

As NTNU has appointed NSD (Norwegian Centre for Research Data) as their Data Protection Official for Research, all NTNU students and researchers are obligated to notify NSD about their project if they are going to process personal data. As our research project won't handle information related to individuals, and as we will only register anonymous information, meaning that the data contains no information that may identify any individual, the project has not been notified to NSD. None of the startups or interviewees were registered or stored by name, or other information that may be used to identify them.

Chapter 5

Results

In this chapter, we present findings from the multiple-case study relevant for answering our research questions. The chapter proceeds as follows: Section 5.1 presents basic information about each of the 13 investigated startups. Section 5.2 seeks to address research question 1 with corresponding sub-questions. Section 5.3 addresses research question 2 and its related sub-questions. The two sections use empirical data from the transcribed interviews to present a meaningful whole from the cases and are based on the thematic synthesis presented in the previous chapter. Section 5.4 uses the findings from the sections 5.2 and 5.3 to explain how hardware startups achieve balance between speed and quality to address research question 3. Additionally, the section presents the model of higher-order themes unique to hardware startups, and a table summarizing the main answers to each of the three research questions.

5.1 Case Descriptions

The following sub-sections present the investigated startups, including product descriptions, business models, and team compositions. Descriptions are made as accurate as possible, without exposing the companies and other sensitive person-identifying information. The case descriptions are made to allow for transferable results and for a better understanding of why the startups operate the way they do (Klotins et al., 2015; Langley, 1999). Table 5.1 presents basic information about each case. The *current stage* in the table is adopted from Crowne (2002), however the first stage *startup* is replaced by *concept* to avoid misunderstandings.

Case	Product	Current Stage	Founded	Location	# of employees
Startup 1 (S1)	Smart gloves	Concept	2016	Norway	18
Startup 2 (S2)	Medtech biosensor	Concept	2017	Norway	5
Startup 3 (S3)	Physical exercise game	Stabilization	2016	Norway	5
Startup 4 (S4)	Unmanned aircraft system	Concept	2016	Norway	7
Startup 5 (S5)	Advanced noise cancellation	Concept	2017	Norway	5
Startup 6 (S6)	Medtech hydration monitoring	Concept	2016	Norway	10
Startup 7 (S7)	LPG management system	Stabilization	2016	Norway	8
Startup 8 (S8)	Cable cam system	Stabilization	2016	Norway	10
Startup 9 (S9)	Digital piggy bank	Concept	2017	Norway	5
Startup 10 (S10)	Collaborative camera	Growth	2014	Norway	50
Startup 11 (S11)	Interactive children's toy	Concept	2015	Netherlands	8
Startup 12 (S12)	3D-printer control board	Growth	2009	Norway	1
Startup 13 (S13)	Sensors for IoT	Growth	2007	Italy	25

Table 5.1: Case Descriptions

S1 Smart Gloves. S1 is a company that develops a human-machine interaction smart glove for intuitive control of machines, with steering drones as their primary target for their MVP. Modern RC controllers are large, complicated, and not very intuitive, resulting in a large number of drone crashes and relatively poor security for low-experienced drone-pilots. S1 replaces the traditional RC controller with a sensor-based glove, aiming at reducing the number of drone crashes, increase security, and give people a more intuitive user experience.

The company was founded in 2016, is purely student-driven, and has received funding from multiple Norwegian innovation initiatives and business angels. The administration consists of CEO, COO, and an HR manager with backgrounds from industrial economy, technology management, and design. The tech team consists of six students from electronics and cybernetics. The business team consists of four students with industrial economy background. Additionally, they have a tech team of five computer engineering students located in the USA. They also collaborate with a local university in a tech-marketing course where a team of five helps them with an extensive market analysis.

Considering that S1 is an early-stage startup, they have a very large team compared to similar companies. This implies that communication is important, also since they have two distributed tech teams. The team-organization allows the company to experiment with different technological solutions, an approach similar to A/B-testing. Working with multiple technical solutions can speed-up development, and facilitate for more customer experimentation. The prototypes have mainly been developed in-house, except for some components that's been developed by an external company to increase the product quality.

S2 Medtech Biosensor. S2 is a medtech startup founded in 2017 that develops a sensor for continuous monitoring of patients. The sensor can extract various data from humans, and visualize this data on iPads and/or laptops. The sensor can replace tasks which today are handled manually by hospital personnel. The original idea was to create sensors set in consumer products, like AppleWatch, to monitor sick people. Through a collaboration with a local hospital, they undertook a product pivot where they went over to creating the sensor as a single unit.

Today the company consists of five part-time employees. The CEO is responsible for business development and the electronics production. The other team members include a software developer, a mechanical developer, a firmware developer, and a signal processing engineer. Each employee has a great responsibility and holds broad expertise in both hardware and software development. To organize the development, they have weekly meetings to synchronize and document changes. This development approach can be seen as a simplified version of Scrum. During its first year, they have had two main prototypes with several versions of each.

The company has received funding from various partners and innovation initiatives, and some members even do consultancy work on the side to generate money for the business. They have invested in an advanced 3D-printer to keep as much development as possible in-house to reduce dependencies and delivery times, and to facilitate for a tight customer feedback-loop with many small, iterative changes. The company have partners in China and USA to help with production and assembling of components, and also outsource parts of their development to meet the strict regulations of the MedTech industry.

S3 Physical Exercise Game. S3 delivers a platform for exercising computer games, in which they build the software side of. The hardware side consists of control-buttons and sensors attached to a bicycle and an intelligent hub that sends data to a PC attached to the handlebar. On the PC, they have a platform running different computer games. Originally a research project, they have received soft funding coupled with a convertible note from private investors. Having identified that their product clearly solves a problem, their main focus is to find the right product/market fit going forward.

The startup was established in 2017 by three people. Currently, the team consists of five full-time employees and two part-time employees. Among the full-time employees, two work on the business side, two work on software development, and the last person does both hardware and software development. Early on, they developed both the hardware and the software in-house. Since hardware development has strict requirements for quality and don't promote flexibility and rapid changes to the same extent as software, the startup has decided to outsource hardware design and development to a nearby consultancy company.

S4 Unmanned Aircraft System. S4 develops an autonomous drone system with a web interface for customers. By putting together already-proven components, they want to deliver drones that are able to autonomously perform specific missions regardless of weather and other external factors. The company was established in 2016 and have received various funding, among others from Innovation Norway. They also partner with local industry to find good solutions and aim at delivering a product that can be used for multiple purposes.

The team consists of seven people, where six work full-time and one is a part-time Master student. Three people have a background from cybernetics and are responsible for electronics and hardware development. Two people have computer science and AI background and are responsible for the IT infrastructure. A mechanical engineer works with drawings and visualizations which has been valuable in receiving funding. The last team member works with business development. The company works closely with business partners and customers to make sure they are building the right product. They try to keep development and prototyping in-house, however, their need for special mechanical and hardware parts require the use of external manufacturing partners.

S5 Advanced Noise Cancellation. S5 is a startup that develops new and advanced noise cancellation technology to ensure that telephone conversations can be performed in privacy, without disturbing people nearby. The product consists of a regular headset fitted with a microphone. It picks up the sound the person makes and generates counter-noise so that the overall effect is that the one you talk to clearly hears you, but the sound you make is reduced in the surrounding environment.

The company is currently at the prototyping stage. They are in constant dialogues with potential customers and have identified a clear need for their product. The team consists of five people, including one full-time employee working on the hardware development, and one part-time employee. In addition, three owners work voluntarily. Originally, the owners and a Master electronics student tried to develop the product, but the full-time employee was hired to help with the hardware development. None of the members have prior industry knowledge.

S6 Medtech Hydration Monitoring. S6 is a startup that develops technology to monitor a person's hydration. The measurement data is sent wirelessly to a hub, and further into the cloud where the data is analyzed. The patch consists of two main parts, one that measures bio-impedance (body conductivity), and one acoustic part measuring the throat. Dehydration causes several health complications and increases the complexity of a patient treatment among elderly. Today hydration measurement is a slow, manual process. S6 has found a market need for a more efficient way of monitoring hydration, to take care of patients and to reduce costs and allow health personnel to work more efficiently.

The company was established in 2016 by five students with technical and entrepreneurial backgrounds. Today the company has 10 employees. Three people work with marketing and funding and one person is a nurse performing tests on patients. The six last members work on the technical solution, including two software developers and three hardware developers who engineer the mechanics and electronics of the product. By means of process, the team tries to avoid strict systems and bureaucracy, but regulations and high quality standards in MedTech will eventually force the company to introduce more processes. They are therefore in the process of introducing a process management tool to describe and create processes.

Currently, their main priority is to develop a product that functions technically, before optimizing it and make it ready for production. A total of three separate prototypes will be collected into one prototype before the final production. Speed has not been a priority as they rather define goals and sub-goals aiming at developing a product that actually works. Since speed partly is a secondary priority, they have a great focus on protecting their intellectual property. This allows them to spend longer time on developing a high-quality solution without others being able to steal their product idea.

S7 LPG Management System. S7 is a startup who develops an IoT solution for gas suppliers. The solution measures content in gas-bottles and sends the measures to suppliers through a web portal. Their objective is to provide LPG suppliers with valuable insight into customer consumption that will help increase profits while making customers happy. Recently, the company has undergone a market pivot towards a more professional sector. This has posed a significant change to their development approach as they no longer are able to produce all prototypes themselves.

The company has a working prototype, and several business partners supporting their work. The team consists of eight employees, including CEO, CTO, one on electronics and IT, one system developer, one part-time employee working on circuit boards, one designer, and one economist.

One of the most important features of their product is the lifetime of the sensor. They have not implemented any specific agile processes or development methods due to the small development team. From the start, they have focused on making prototypes in-house, using molding and 3D-printing. A recent pivot towards a more professional market sector has lead to stricter non-functional requirements and forced them to change product material. This means they now have to outsource the production of some prototype components rather than developing them in-house.

S8 Cable Cam System. S8 develops a camera accessory product used to record film with new camera-angles. The main parts of the system include a wire, a motorized camera-holder moving along the wire, and a remote control to steer it. The design of wire and camera-holder enables access to narrow terrains where for example drones cannot maneuver. The company was established in 2016 by four founders with technical and entrepreneurial backgrounds. Today the company consists of ten employees. In 2017 they launched their product at a Kickstarter campaign, where they pre-sold more than 4000 units to a total sum of $1.1 million. They have also received funding from various innovation initiatives.

The technical team consists of five people. The CTO has a background in mechanical engineering, without similar working experience. The second person is a product designer, with experience from several years working on similar projects. Another person without industry experience works with mechanical analyses and simulations. A part-time cybernetics student with experience from a similar project has recently joined the team. They also have a person from abroad working with both low-level software and hardware development, with many years of industry experience.

All prototypes have been physical, and each new prototype mostly consists of one new feature. This means that several components is reused for each prototype. Initially, the first prototype was developed by a consultancy company. This helped kick-start the business, but the quality of the prototype did not meet the initial expectations. All hardware and mechanical components are produced in China.

S9 Digital Piggy Bank. S9 creates a digital saving device for children, similar to the traditional “saving pig”. The device is connected to the owner’s bank account, and shows balance, deposits, and saving tips. The development is currently in the prototyping stage. The company collaborates with different banks to create a transparent solution which can be used across various banking platforms. This will cause the product to be used by most users without depending on a particular bank affiliation.

The company was founded in February 2017 by three people. They also have two developers working full-time in Serbia. In addition to having two developers in Serbia and one internal employee with PCB circuit board knowledge, the company has used consultants to help with production expertise and UX design. With little initial knowledge about hardware development, the company has used knowledge and contacts of other startup companies to find suitable manufacturers and components.

All prototypes have been physical, and they have used a lot of 3D printing to speed-up development. The small team has not experienced a need to implement specific agile methods or other processes. For communication, they use a Slack channel to connect hardware designers, graphical designers, and other people associated with the development process. Since the development team is not co-located, performing rigorous testing activities can be a tedious process.

S10 Collaborative Camera. S10 offers an intelligent camera platform for online video conversations. The camera is similar to a traditional web-camera, but the camera is better suited for conference meetings. The camera provides HD video and fast data transfer through a USB 3.0 port, it can auto-adjust to the surrounding lighting conditions, and the lens has a 150-degree angle to capture a larger audience. The product also provides visual noise filtering and digital pan and tilt functionality, as well as an intelligent auto-zoom that can detect participants.

The development team consists of 35 people. Most of these work on the software side, where three cross-functional teams work with both software and hardware development. Additionally, they have people in-house working specifically on hardware and mechanics, as well as industrial design and UX design. Instead of outsourcing parts of the development, the company has deliberately hired consultants to allow for a more controlled and flexible development process. Components are mainly produced in China, but the final production takes place in Norway.

S11 Interactive Children's Toy. S11 develops an interactive device for children to play with, supporting active play and encourages children to interact and play together. They deliver a device that seamlessly merges interactive technology with "old-school" games to make sure that kids get up and move.

The company was officially established in March 2016 by two founders. In 2018, they have increased the company with six part-time employees, including one product developer, one sales-person, one on communication, two programmers, and one game developer. Hardware development is outsourced, and the product developer is mainly responsible for managing the communication with the external company to ensure a more controlled and flexible process.

Hardware is developed by an external company, while S11 is responsible for developing the application layer. Outsourcing hardware development has been key to the process so far. Except for some initial communication problems, the outsourcing relationship is now an important feature of their development approach. Recently they have hired an extra person to take care of all communication towards the external partner. This allows S11 to focus on their core business.

S12 3D-printer Control Board. S12 is a small startup developing control systems for 3D-printers. The product consists of a card for controlling printers, and a screen to control the card. The software side of the product is developed as an open source project, that is the company develops hardware, while external developers implement the software functionality.

The company consists of one employee who is responsible for designing the electronics, programming software to fit with the hardware and marketing. The company has used Kickstarter to receive funding and is currently in the "go to market" phase. With one employee, the product development does not follow any specific process. Being an open-source software project, other developers can contribute with software updates to the project repository on GitHub. New hardware versions are sent to the developers who in turn implement new functionality. The company has struggled with components of too low quality, and components that have gone out of production.

S13 Sensors for IoT. S13 is an Italian company specializing in wireless technologies like Wi-Fi and 4G, and IoT sensors for smart cities and various industries. They have collaborated with several companies in research projects, aiming at creating a complete, modular, and scalable offer for different markets and sectors.

The company has competence in design of hardware boards, controllers, basic boards with wireless interfaces, design of firmware, design of small systems with wireless sensors, and IoT. The employees are specialized in design of hardware parts and firmware. The technical team's wide range of competencies is beneficial when delivering and developing IoT solutions.

To manage processes, they have not implemented specific agile practices, rather focusing on being a small team of skilled people, attributes enabling them to be flexible and respond quickly to customer requirements. They focus on reusing components like microcontrollers. To balance hardware and software development, they outsource the production as they cannot build the hardware parts themselves. They have employees working with both software and hardware development. Software development usually starts once the hardware has been designed.

5.2 RQ1 How do hardware startups achieve agility during product development?

5.2.1 RQ1.1 How do hardware startups develop their products?

Hardware startups' products consist of mixed hardware and software parts. When companies are in the prototyping phase, the product is usually in the form of a MVP. The MVP can be seen as the simplest way for a startup to demonstrate its value proposition. Product development refers to the methods and processes hardware startups utilize to develop prototypes and to deliver products to customers. In this section we present how the investigated startups developed their products. We focus on their use of development processes, prototyping, and some of their testing and quality concerns during product development.

Development processes. As hardware and hardware-oriented software development involve a lot of experimental work, developers are encouraged to follow an iterative development approach (Ronkainen and Abrahamsson, 2003). Among the cases, five practiced simplistic versions of Scrum, seven used ad-hoc agile practices, while one startup did not follow a defined agile development process. In some startups there were not identified a need to implement specific development methods, one reason being small team sizes. This was especially the case in early stages when tech teams were co-located and introduction of formal communication processes would inhibit the agility and freedom of the team. In the startup where the development team only consisted of one person, the degree of process was almost absent.

> S5 - *"Since the team is so small, communication is easy. We have not seen a need to implement any specific agile methods or other lean practices."*

In other startups, the nature of hardware development made the use of agile principles an intricate endeavour. Practices like regular refactoring and frequent release are not necessarily suitable for hardware development. To maintain speed, hardware startups tried to limit administrative overhead, similar to low-ceremony processes in the software context (Wasserman, 2016). Most startups preferred a more ad-hoc approach customized to their own needs.

> S13 - *"I don't think agile practices are applicable to hardware development, for example you cannot frequently re-design a port as it involves great costs."*

> S8 - *"In hardware, the variance of tasks and interrelated dependencies make it more complex than what current Scrum tools like Gira are suited for."*

> S4 - *"Strict Scrum is probably easier to implement for pure software development, so we use a simplified version of it."*

Due to different team sizes, product offerings, and other financial, managerial, and human factors, agile practices were implemented differently among the hardware startups. Sprint duration usually lasted between 1-2 weeks, and goals were defined in weekly meetings. Since development of physical products usually takes longer time than implementation of software, the startups focused on defining measurable sub-goals that were part of a long-term plan.

> S1 - *"We work on a weekly basis where we define goals for each week. These are part of a main goal of what we want to achieve during the semester."*

Startups that served professional business markets usually had longer Sprint duration. They operated in more stable environments where customer demands were easier to predict, not requiring the same level of flexibility and experimentation.

> S6 - *"We follow three-month Sprints that are part of a long-term plan ending in 2019."*

Most cases had the same Sprints for the respective hardware and software development. One startup differentiated between hardware and software Sprints to better handle contingencies of hardware and software development.

> S10 - *"Software development follows two-week Sprints while hardware Sprints last 1-2 months."*

Prototyping. Almost all startups immediately built a physical prototype to elicit requirements and achieve rapid business experimentation. They usually followed an evolutionary approach, performing incremental improvements on an early low-resolution prototype. Rapid prototyping is important to obtain customer feedback, however it can be problematic in the hardware context. Non-functional requirements are more important to hardware startups than software startups because of the perceived customer satisfaction, performance issues, and their ability to develop many prototypes. Third-party dependency is another factor negatively influencing the speed of prototyping in hardware startups, both due to delivery times and the increased cost of high-quality prototypes.

> S10 - *"We made a physical prototype immediately. It looks like today's product, but with many shortcuts and lower quality."*

> S8 - *"We can develop many low-resolution prototypes using our own equipment, but if we want high-quality prototypes we might have to order 10 different parts from 2-3 suppliers. If we spend more money on shipping, delivery times become shorter."*

To deal with their inability to quickly develop prototypes, the startups tried to be flexible on the software side of their products. Since software can be frequently updated and tested by customers, they focused on developing a simple interface between hardware and the software directly accessing the hardware. In this way they could achieve more parallel and independent development of hardware and software. They mainly tried to reuse software, as hardware components were easier to reuse with more refined prototypes.

> S3 - *"We have developed a simple interface between hardware and software so that the development can happen individually."*

> S2 - *"We prefer making changes in the software or firmware. To facilitate this, we have a clearly defined interface between software and hardware."*

Keeping hardware production close to their own business can help hardware startups to achieve more rapid development speed. If they outsource to southeast Asia, communication becomes complicated and delivery of components takes longer time.

> S11 - *"We decided to keep hardware production close to ourselves... Keeping production locally makes the whole process a lot easier."*

Another way they can speed-up development is to have multiple tech teams working on different technological solutions. Multiple teams imply more administrative overhead, posing stronger demands on management to maintain a flexible workflow across the company. Startups that are able to achieve this are better equipped to perform problem space testing.

> S1 - *"The two tech teams work on different technological solutions to quickly find the best solution fit."*

Testing and quality assurance. Testing activities were generally not performed in a systematic way among the investigated startups. Even if the product quality often was important to many of the startups, they did not have a formal procedure to improve product quality. They usually took shortcuts and workarounds to achieve rapid development.

> S4 - *"It does happen that we take shortcuts to make things work. What we make currently doesn't exist in the market, so we might prioritize to make these exist rather than achieving high quality."*

Not until startups had grown and served a more established customer base were testing activities performed in a systematic manner. Until then it was usually the responsibility of each developer to verify new functionality.

S8 *"The person responsible for delivery is also responsible for testing the feature to make sure it works."*

S1 - *"Components are tested independently before they are put together into a single product... We do not have a systematic testing process."*

When developing resource-intensive systems, the product quality is sometimes more important than the development time. Since startups generally want to achieve rapid development speed, finding an optimal trade-off between quality and speed can be difficult for hardware startups.

S5 *"We do have periods where we spend a lot of time waiting on vendors. Then we have time to improve and optimize both software and hardware."*

5.2.2 RQ1.2 What kind of challenges are relevant in the hardware startup context?

There are several challenges associated with product development in hardware startups that were uncovered during the interviews. Even though there still may exist challenges that are central to hardware startups, we believe table 5.2 describes a bigger part of the challenges today's practitioners need to overcome. As each startup was encouraged to come up with the three biggest challenges they had met so far, several of the challenges might be relevant to more of the cases than what is presented in the table. In addition to the challenges the startups identified themselves, we also included challenges uncovered from other parts of the interview.

The challenges were grouped into four categories: financial, human resource, strategic development and engineering. Financial challenges refer to problems or pressure related to the startups' financial resources. Human resource challenges involve any difficulties experienced with or by the startups' employees. These can be individual challenges, organizational challenges, or related to the startups' environment. Strategic development challenges are related to the business and process decisions of the startup. Engineering challenges are related to the hardware and software development activities.

Category	Challenges	Case
Strategic Development	Production and shipping time of third-party vendors	S2, S4, S7, S8, S9 S10, S11, S12
Strategic Development	Delay due to manufacturing defects	S4, S6, S10
Strategic Development	Market specific regulations and restrictions	S2, S6
Strategic Development	Avoiding end-of-life components	S12
Strategic Development	Delay due to design errors of hardware components	S1, S4, S10
Strategic Development	Balancing profitable hardware modules and quality	S6, S9
Strategic Development	Feature creeps	S8, S9
Strategic Development	Prototyping capacity	S1, S3
Human Resource	Internal and external communication	S1, S8
Human Resource	Inexperience of working with third-party vendors	S2, S5, S8, S10
Human Resource	Attracting talented people	S6
Financial	Insufficient funding	S3, S4, S11, S12, S13
Financial	Lack of prototyping equipment	
Engineering	Integration of ready-made or outsourced components	S1, S8
Engineering	Underestimating implementation time	S3, S8
Engineering	Intricate product design	S2, S4, S9, S11
Engineering	Lack of reference system	S2, S8
Engineering	Complexity of electronic product development	S1, S2, S3, S4, S5 S7, S10
Engineering	Realistic test environment	S2, S4

Table 5.2: Challenges encountered by hardware startups

Strategic Development. In relation to strategic development there were several issues appearing in the interviews that more or less were common to all cases. Several of the identified challenges are interrelated and arise from the complex context of hardware startups. Central is how startups can achieve speed when working with restricted resources and external partners. As development of high-tech electronic products is extremely resource demanding, lack of resources can be even more severe for hardware startups than software startups.

The most time consuming process of hardware prototyping is the long production and shipping times, as production usually is located in China or other countries in southeast Asia. This means that not only will the delivery time of necessary parts depend on the vendor's own schedule, but also the shipping method used. Several of the investigated cases spent a significant amount of money on speeding up production and shipping time of manufactured components.

> S8 - *"All parts of the prototypes must be ordered, mostly from China, with long delivery times. We spend a lot of money making delivery times shorter."*

Among the investigated startups several experienced quality issues working with their external partners. Manufacturing defects of crucial prototype components caused extra delays, which is critical considering the valuable time already spent waiting for the components. Cooperating with professional actors can decrease the risk of quality issues, and enhance communication.

> S4 - *"We have outsourced production of mechanical parts and circuit boards. Some of the components we have received from the manufacturer have been in bad condition and with significant defects."*

As high-tech prototyping is a time demanding process, there might go several years from the startup is founded before a finalized product is ready to be released to the market. This implies that vendors' dependability also is of importance. Choosing components that with certainty will be available the entire prototyping stage is crucial.

> S12 - *"The first version of the screen went out of production. This was the most important component and it took a lot of time to fix the problem."*

To achieve speed, product quality often gets low priority in startups. However, because of the vendor dependency of hardware startups, hardware development should receive higher focus on quality. Making shortcuts in hardware design, and not assuring that the design is of sufficient quality before sending the specifications for production might be costly. Findings suggest that hardware startups are investing more in quality of their hardware components than their software components. Making improvements and changes to software is usually easier, and can often be handled in a single day.

> S1 - *"We spent more than $500 on a single component we could not use. In addition we had to spend more time redesigning the board, and wait for it to be produced."*

A major difference to software startups is the ability hardware startups have to test their product with a larger customer base. Testing is usually performed with few pilot customers or by team members experienced with similar products. As each prototype is related to individual development cost and time, the testing ability in hardware startups will rely on the startups capacity to produce prototypes.

> S3 - *"There is a great number of people who want to test our product, however we do not have the capacity to produce enough prototypes. The main reason for this is hardware production, which happens in China, and the manual assembly we do ourselves."*

Startups work with innovative products for different markets. Two of the investigated startups work with MedTech, and technologies and solutions aimed at solving health issues. Several markets may pose specific regulations and restrictions that affect process, testing, documentation, etc. Customer testing becomes a tangled process usually involving a significant amount of paperwork, and standards must be followed to handle quality and process.

> S2 - *"As we want to test our product in a clinical context there is a lot of paperwork. It is our biggest brake pad."*

> S6 - *"Since this is a medical company, there are very strong requirements for ISO certification and standards. It must be possible to trace all our components back to the manufacturer."*

Human Resource. Lack of experience can be critical when working with third-party vendors. Startups often consist of students with little or no experience from working with bigger production companies. Because of the pressured financial resources and the small production batches it can be hard for startups to find manufacturers invested in the startups' success. Working with vendors producing components of high quality at an affordable cost will be an advantage. The big geographical distance, and the difference in language and culture may also challenge the communication skills of the team, as effective communication is important to receive service as paid for.

> S10 - *"The first step is to find good suppliers that can help you. As a startup, this is not always easy, low volume, and no financial security to show, mean that you may not get the suppliers you really want. We have seen that as we have grown, we have been able to work with better suppliers producing at higher quality, which in turn has helped us prototype faster."*

> S2 - *"We are building on networks from previous startup experience. We have worked with our partners in both China and Texas before... Previously, we chose the cheapest suppliers, but then we also got components in bad condition, there were communication problems, and it usually took more than 4 weeks to get the products. When we finally got the components, we had to spend time fixing the production defects."*

To handle the unique context of startups and the many challenges it poses, hardware startups need team members that are dedicated to all aspects of the development process. As hardware startups have to deal with many factors besides software there are higher demands to expertise and experience of team members. Team members of hardware startups will preferably need knowledge about both application domain, systematic development, software and hardware development, mechanical engineering, and experience of working with third-party companies. Attracting knowledgeable people is hard as startups rarely can provide good salaries.

> S6 - *"Finding talented people is hard. Since we are a startup we cannot give very good salary. This is why we try to attract people who see that the product may provide great value in the future."*

Financial. Doing what is possible to speed up prototyping is essential for hardware startups as third-party dependency greatly affects development time. Having access to prototyping equipment will be an important asset, reducing both development time and prototyping cost. With 3D-printers startups can do a lot of the prototyping themselves, and make rapid changes based on customer feedback. This enables faster problem space testing.

> S2 - *"With a 3D-printer we can make products that look and feel real. This is a huge advantage. We can literally do almost everything apart from the electronics production ourselves and put it together almost for free."*

> S9 - *"We have done a lot of 3D-printing. Without access to useful equipment prototyping would have been very expensive and taken more time."*

Engineering. Hardware startups generally work with off-the-shelf components from suppliers in addition to the components they develop themselves. Today's access to components at a low cost means startups can choose between a wide variety of components. Finding the best component to suit their specific needs might be difficult. Making the wrong decision can lead to later struggles and increased development time. With components of various types, the complexity increases fast.

> S8 - *"We are struggling to control the engine component. The choice of engine was done almost two years ago, and we do not yet have a well-functioning engine that works the way we want it to."*

Hardware startups often develop products with intricate designs. This implies unique requirements to the shapes of components which needs to be tailored to fit inside the product. Developing components of unusual types, and making them fit into the product shape affect development speed.

> S11 - *"We have made several trade-offs affecting development. Our product has a very specific form-factor."*

Testing is central to hardware startups. High quality in hardware development is important both because of the cost associated with mistakes from production, but also as quality greatly affects the perceived functionality of the product. As hardware startups rarely have the capacity to produce many prototypes for testing, optimizing valuable learning from each prototype is important. In contrast to software products, it is challenging to implement changes and make improvements to the quality after the product has been produced and assembled. As a consequence, focus on non-functional attributes at the prototyping stage is essential.

Several startups faced the challenge of testing their product in realistic environments because of legal restrictions related to privacy and public safety. Simulations and dummy-data can be alternatives to early testing.

> S4 - *"Setting up a foundation for doing robust tests is a challenge. When developing drones it is not easy to perform testing, it requires specific experience and knowledge."*

5.2.3 RQ1.3 How do internal/external context factors impact the speed of product development?

Hardware startups' context is distinct from pure software startups as they need to handle hardware design, development, and manufacturing in addition to software development. Working with several technology domains and third-party vendors increase the complexity and lengthen the prototyping stage. The unique context poses high demands for team knowledge, skills, and capabilities for creating innovative products.

We have used the *reference framework of situational factors* developed by Clarke and O'Connor (2012) to analyze the context of hardware startups, identifying relevant contextual factors affecting speed. The framework is intended for researchers and practitioners to design software process to development setting. As hardware startups include a significant amount of software, the framework will cover a major part of influential context factors. In table 5.3 the sub-factors are adapted to the relevant situation in hardware startups.

Classification	Factor	Sub-factor
Personnel	Turnover	Probability of team members quitting due to long prototyping stage
	Team size	Limited access to personnel
	Culture	Efficient communication among team members
	Experience	Ability to handle unforeseen events
	Skill	Business, programming, and testing capabilities
	Productivity	Minimize activities not adding value
	Commitment	Team members working towards a common goal
Application	Performance	Real-time requirements is critical to the product's perceived functionality
	Complexity	Diverse interconnected components
	Reuse	Evolutionary approach
	Quality	Flexible architecture and system design
Technology	Knowledge	Experience with hardware and software technology
Organization	Facilities	Workspace arrangement
Business	External dependency	Third-party components and hardware/mechanics production

Table 5.3: Situational factors influencing the speed of hardware startups (Clarke and O'Connor, 2012)

Personnel. Documentation is rarely a formal process in early-stage startups. Knowledge is usually in the form of tacit knowledge. As the prototyping stage of hardware startups often is significantly longer than that of software startups, due to the co-design of hardware and software components, and integration at the system level, hardware startups deal with a higher risk of people quitting before introducing their product to the market. Losing important personnel may hamper the prototyping stage and greatly affect the development speed.

> S4 - *"Sometimes it becomes challenging to keep the knowledge of people who quit, the knowledge often accompanies that person. This leads to extra costs and effort."*

Communication skills are important to increase efficiency, reach goals, and avoid conflicts from misunderstandings. Team diversity and production partners introduce culture and language differences. Overcoming communication barriers is important for maintaining product development speed.

> S8 - *"We have a diverse team. One team member is from Iran, so we speak English. We have a production partner from China, who speaks quite poor English. Communication is important to the process so that everyone fully understands the purpose of what we are doing, and the decisions we make to achieve what we want."*

Application. Embedded systems have strict requirements to performance. Meeting the hard real-time requirements of hardware-related software development is essential as it greatly will affect the perceived functionality of the product. If the product fails to meet timing and performance requirements the entire system operation can be jeopardized. This can be critical for certain products as it may be potentially dangerous and present hazard to consumers. Meeting the hard real-time requirements is also important as later refactoring of embedded software can be challenging (Ronkainen and Abrahamsson, 2003). As hardware startups seem to follow an evolutionary approach, pre-development system design is necessary to facilitate for rapid changes at a later stage.

> S4 - *"Performance requirements are important as failures can cause additional work and costs, and at worst present safety issues."*

Development time can be significantly reduced by reusing components. Prototyping in hardware startups follows an evolutionary approach, meaning that there are made incremental improvements for each prototype where several components are reused if possible. Among the investigated startups there was a more extensive reuse of software than hardware. Hardware and mechanical components were easier to reuse with more refined prototypes than early low-resolution prototypes. By only working with evolutionary prototypes feature creep might become a problem. Almost every startup started out building a physical prototype, as prototyping is as much a feasibility test as a way of testing out business hypotheses and eliciting customer requirements. The cases made little use of mock-up tools, and so throwaway prototypes seem to take little part of the prototyping stage of hardware startups.

> S2 - *"All the prototypes have been physical. We try to reuse as much as possible from each prototype. We divide the code into different modules, so that if we replace any hardware component we only need to change that specific part of the code."*

> S10 - *"We made a physical prototype immediately. It looks like today's product, but with many shortcuts and lower quality."*

> S7 - *"We tried to reuse the electronics, but it was harder than expected. So there are mostly new components every time... The software is mostly the same from prototype to prototype."*

> S9 - *"Every prototype we have made have been physical... At first we had too much functionality and had to remove some to obtain a more reasonably priced product."*

Time to market is central to describing the startup context. Being able to prototype fast for testing new ideas and product features is crucial to learn faster than competitors. Facilitating for changes to be made on the software side will positively influence development time, as changes can be done in a few days. Keeping software development in-house, with a clearly defined interface between software and hardware is beneficial.

> S3 - *"When we outsourced software development changes took a lot of time. Things that should have taken a few days took a few weeks. In software we need to make changes weekly. In hardware it is okay that things take a bit more time."*

> S2 - *"Our biggest strength is that all development happens in-house without consultants... We prefer making changes in the software or firmware; the dream is to make as little hardware as possible. To facilitate this, we have a clearly defined interface between software and hardware."*

Technology. Hardware startups need knowledge about a wide range of different technologies. In the early stages of the startup, having few employees implies that team members must spend time learning specific hardware or software technologies necessary for developing the product. This causes additional overhead and underlines the importance of having a team with boundary-spanning knowledge.

> S1 - *"When I started I did not have much knowledge about PCB design. I had to spend six weeks learning circuit board design to make it work."*

Organization. Common to the investigated startups is that they share working facilities with several other startups. Being able to cooperate with other startups that have faced similar challenges and situations was named as a big advantage. In the huge number of vendors for production of hardware and plastic components, finding a good partner can be one of many challenges. Asking more experienced entrepreneurs and startups for help can prevent struggles in the early stages of the startup.

Business. The major factor increasing development time in hardware startups is external dependencies. Relying on third-party vendors in production causes a significant increase in development time. Making use of local vendors can be an advantage to handle the communication issues and delivery times of unfamiliar, far-away vendors. The local closeness might lead to a more productive cooperation as both companies have interest in each other's success.

> S11 - *"We decided to keep hardware production close to ourselves. We can immediately check if things work as intended, and they can respond very quickly to any changes we might need... Keeping production locally makes the whole process a lot easier."*

Planning ahead when working with vendors is important. Time is a limited resource for every startup, and being kept idle waiting for parts is not desirable. As the goal of startups should be to minimize total time through the build-measure-learn feedback loop to accelerate the business as fast as possible, time should be spent on value-adding activities. Testing problem space and learning customer needs should be of higher priority than making incremental quality improvements.

> S5 - *"We do have periods where we spend a lot of time waiting on vendors. Then we have time to improve and optimize both software and hardware."*

5.3 RQ2 How do hardware startups manage quality concerns of their products?

To achieve speed, product quality gets low priority as thorough documentation and testing practices are neglected. Software startups utilize ad-hoc testing methods (Giardino et al., 2014a), and replace strict product development processes with a set of opportunistic activities, focusing on providing value under constrained conditions. This section presents our findings in terms of how hardware startups manage quality concerns of their products.

5.3.1 RQ2.1 How are hardware products tested?

Most interviews were performed with CEO's, and so their answers were biased towards managerial aspects of testing and quality assurance practices. This section presents initial findings of testing practices and strategies from the cases.

To achieve quick development speed in early stages, testing activities generally receive little focus in hardware startups. Before a feature is guaranteed to be part of the final product, it is more important to verify that the feature adds value to the customers. Until then, the time spent on testing activities is minimized. This is also evident in software startups, where developers avoid wasting time on improvements of not-validated functionalities (Giardino et al., 2016).

> S2 - *"We prefer to work fast, as writing tests can double the development time... If parts are to be replaced, then we think there's no point in spending time on testing."*

The hardware startups relied on each individual developer to test features as they were implemented. In that way the person responsible for the code was also responsible for its quality and functioning with the rest of the system. A frequently used testing activity among the cases was manual smoke tests (i.e., ensuring that major functionalities work before undertaking more formal testing procedures). Prototypes were manually tested by internal employees to identify the most prominent defects before testing prototypes with early adopters or customers.

> S8 - *"We test the subsystems ourselves, but do not have a structured system for testing... The person responsible for delivery is also responsible for testing the feature to make sure it works."*

> S1 - *"People inside the startup who have experience with similar solutions test the product before it is tested with pilot customers."*

An alternative method to testing used by hardware startups was simulations. Hardware simulations can help determine the precise operation of hardware without producing an expensive prototype, and enable testing of the hardware-software co-operation (Ronkainen et al., 2002). Several startups found it challenging to test their product in realistic environments, both due to memory and performance constraints and because of privacy and public safety issues. Since planning is difficult in the startup context, test plans were often changed, hence these were often neglected. Simulations helped testing the product and code base before production, postponing the split between hardware and software functionality.

> S4 - *"At an early stage, things don't always go as planned. Other things than what you test for fail, so test and project plans often change a lot... In addition to performing many simulations, we use basic tuning of attitude control to avoid simple mathematical errors."*

Among the investigated startups we found that startups in later lifecycle stages implemented more systematic testing activities. As they got more customers, quality and testing activities became more important. Established customers have stricter demands than pilot customers. To deal with increased quality requirements, the startups implemented formal processes for testing.

> S10 - *"In software we have a great focus on testing. When software is modified, we run automatic tests to ensure that everything works... In hardware we test that the product functions in different climates, and perform various mechanical tests... We have also outsourced much manual testing to a company to check more parts of the product."*

5.3.2 RQ2.2 How is technical debt managed in hardware startups?

Technical debt has been illustrated by Brown et al. (2010), stating that "developers sometimes accept compromises in a system in one dimension (e.g., modularity) to meet an urgent demand in some other dimension (e.g., a deadline), and that such compromises incur a 'debt' on which 'interest' has to be paid and which the 'principal' should be repaid at some point for the long-term health of the project". The development of minimum viable products and releasing the product as fast as possible often require the development team to take shortcuts and workarounds. Shortcuts can lead to the accumulation of what is called intentional technical debt (McConnell, 2007). Unintentional technical debt can happen when business model experimentation is leaved out. No matter how good the idea may seem, not validating it with customers can lead to the development of unnecessary features.

Not focusing on technical debt will have consequences for the product quality, while constantly changing and improving the business model will be necessary to stay competitive (Yli-Huumo et al., 2015). Finding the correct balance between learning goals and quality is therefore important in order to minimize waste and to manage technical debt (Terho et al., 2016). This section presents how hardware startups manage technical debt of their products.

Intentional technical debt. By accepting that time to market is more important than product quality, hardware startups incur intentional technical debt. Business experimentation to build new features is performed in small iterative cycles with minimal effort on product quality to receive fast customer feedback. This is especially the case for the software components of the product. Since software can be changed quickly, shortcuts and workarounds are more easily taken on the software side than on the hardware side of the product. Implementing new functionality receives more focus than the quality of the code. Hence, the temporary low-quality solutions will eventually lead to accumulation of technical debt.

> S2 - *"Software changes all the time... To make things work straight away, we'd rather take a shortcut and fix it later. We know we're building up technical debt, but it's on purpose to be able to test the product on customers as quickly as possible."*

Documentation. On the software side of the product, the common perception is that since the developers work on the code-base every day, documentation activities lead to additional overhead. Tacit knowledge seem to be a common practice in hardware startups.

> S3 - *"We spend less time on documentation to speed things up, development is our main focus. It is also because software development is in-house. We work on it daily and understand the code."*

High-tech products include a lot of different sub-systems and technologies, and so product complexity increases fast. This implies that documentation of components should receive a bigger attention in hardware startups. In worst case, lack of quality and documentation can put all development on hold.

> S2 - *"Instead of updating documentation and quality, we did things as fast as possible, which eventually led to a lot of extra work."*

The prototyping stage in hardware development is often significantly longer than that of software development. Since it might take years before hardware startups have a functioning product ready for the market, there's a great probability of people quitting the project before it is finished. As to this there should be increased focus on documentation in hardware startups, since knowledge often accompanies the person quitting.

> S4 - *"Sometimes it becomes challenging to keep the knowledge of people who quit, the knowledge often accompanies that person. This leads to extra costs and effort."*

The choice of outsourcing companies can greatly impact the amount of documentation. Good partners usually provide well-documented solutions and components, which can help manage technical debt.

> S3 - *"We received an 80-page user manual from the consultants who developed the hardware."*

For some startups, market regulations imply that they must follow strict processes. Companies operating in the market will need to document all parts of their product and meet the high standards of quality required. Hence, market segment will greatly affect the severity of technical debt.

> S6 - *"We are weak on processes and document management, it is very adhoc. Soon we will introduce a process tool and a document management tool. This is necessary if we are to meet the ISO standard requirements and get it approved as a medical product."*

> S13 - *"The documentation is part of the development process... We have an ISO certification that says we are certified according to that quality process. They have strict requirements on how documentation should be kept, including the flow of the documentation and what kind of documentation to write."*

To help startups perform documentation, there exist multiple tools lowering the barriers for writing documentation. Examples of tools include Wikis, Google Spreadsheets, and Confluence. Utilizing tools can help decrease the amount of rework in the long run. Also thorough documentation can allow for more efficient integration of new employees in the development process.

> S2 - *"Previously we have spent a lot of extra time due to a lack of documentation. Instead of stopping, we did things as fast as possible without performing documentation. This eventually lead to a lot of extra work."*

> S4 - *"We have a wiki for internal documentation. It is quite low effort to write something on it."*

Unintentional technical debt. The reasons for unintentional technical debt are often out of control and can be difficult to be aware of. The main reason is the lack of business experimentation. Hardware startups often have a clear conception of how their products should be, including design and functionality, and focus on validating their initial business idea. The evolutionary approach helps demonstrate problem/solution fit through discovering the needs of early customers, enhancing the effectiveness of the product and prevent exhaustion of resources. However, the perceived difficulty of testing problem space can lead hardware startups to implement unnecessary features, too expensive products, or low-quality prototypes.

One reason for the perceived difficulties of testing problem space in hardware startups is their ability to produce prototypes. Physical prototypes are resource-intensive to develop, and in contrast to pure software products, one cannot necessarily deliver a new digital software update to customers. Lack of financial resources and long delivery times make it challenging to test the product on a broader specter of customers. The investigated cases relied on a small unit of pilot customers for feedback.

S9 - *"At first we had too much functionality and had to remove some to obtain a more reasonably priced product... Now we focus more on creating a good user experience instead of implementing features that customers and investors don't want."*

S8 - *"We can develop many low-resolution prototypes, but not more than 4-5 high-resolution prototypes."*

5.4 RQ3 How do hardware startups achieve balance between speed and quality?

Hardware startups operate in a competitive environment, where time-to-market is critical to the startups' success. Speed is essential to avoid being outperformed by startups with similar business ideas. At the same time, our findings indicate that hardware quality is essential to maximize valuable learning from the build-measure-learn feedback loop and prevent expensive rework. Meeting hard real-time requirements requires pre-development system design, and sets the stage for later flexibility and rapid changes. In this section, we have synthesized our findings from the previous sections to present how hardware startups balance speed and quality in the prototyping stage.

	Hardware	Software
Speed	Off-the-shelf components Local manufacturing	Component reuse Development shortcuts Tacit knowledge
Quality	Up-front system design Hard real-time requirements	

Table 5.4: Balance of speed and quality in hardware startups

Speed. Hardware startups can take advantage of using ready-made components to speed up prototyping. As the value proposition from the products delivered by hardware startups most often is provided by its unique software, hardware startups should look to off-the-shelf hardware components when possible. Hardware development is a time consuming process, and so minimizing necessary work for experimenting business ideas is essential.

Manufacturing of hardware and mechanical parts is the most time-consuming process of prototyping in hardware startups. Communication, delivery time, and the risk of receiving bad-quality components are common issues. In addition, the highly experimental nature of hardware startups and the uncertainty of customer requirements at the prototyping stage imply that product features and design might change after components have been sent for manufacturing, or during production. For the outsourcing partnership to become a success, startups need committed partners who understand the dynamic startup context. Making use of local vendors can be a feasible option, contributing to prototyping speed.

Evolutionary approaches with extensive reuse of components are the most suitable for software development (Giardino et al., 2016). Our findings indicate that this also is the case for hardware startups. Hardware startups have bigger tendency to reuse software due to the nature of hardware components. Evolutionary prototyping promotes flexibility and reactiveness and allows hardware startups to move quickly forward while making rapid changes.

Hardware startups intentionally take technical debt on the software side of the product. Software for prototypes is developed as fast as possible with minimal amount of code and focus on quality. Allowing for shortcuts and workarounds at the prototyping stage increases the speed of the customer feedback cycle. As there are higher risks associated with technical debt in hardware development, hardware startups seem to be less willing to prioritize speed over quality for hardware components.

Software documentation relies on tacit knowledge instead of formal documentation. Prototyping is characterized by rapid changes, and so documentation would need to be updated every time the source code is changed. As hardware startups try to implement most feature changes in software, effort and time can be spared relying on the knowledge of team members.

Quality. Up-front system design is unavoidable when working with hardware. As hardware-related code is complex and sensitive to changes (Ronkainen and Abrahamsson, 2003), preliminary architecture design is necessary to facilitate iterative development, and flexibility to handle rapid changes. As hardware startups intentionally try to force changes on the software side, neglecting up-front design may cause bugs that are not easily detected. Investing in architecture quality may eventually lead to speed in business experimentation.

Several hardware startups develop products where attention to non-functional requirements is inevitable. Some products might cause dangerous situations if they fail, or bad performance might affect perceived functionality and feedback from customers. Meeting the hard real-time requirements will affect prototyping time.

Thematic map. The compromise between speed and quality in hardware startups is strongly related to how hardware startups develop products and manage quality. Findings indicate that hardware quality demands make a difficult setting for prototyping speed. The higher-order themes arising from the data contribute to answering why balancing speed and quality can be problematic.

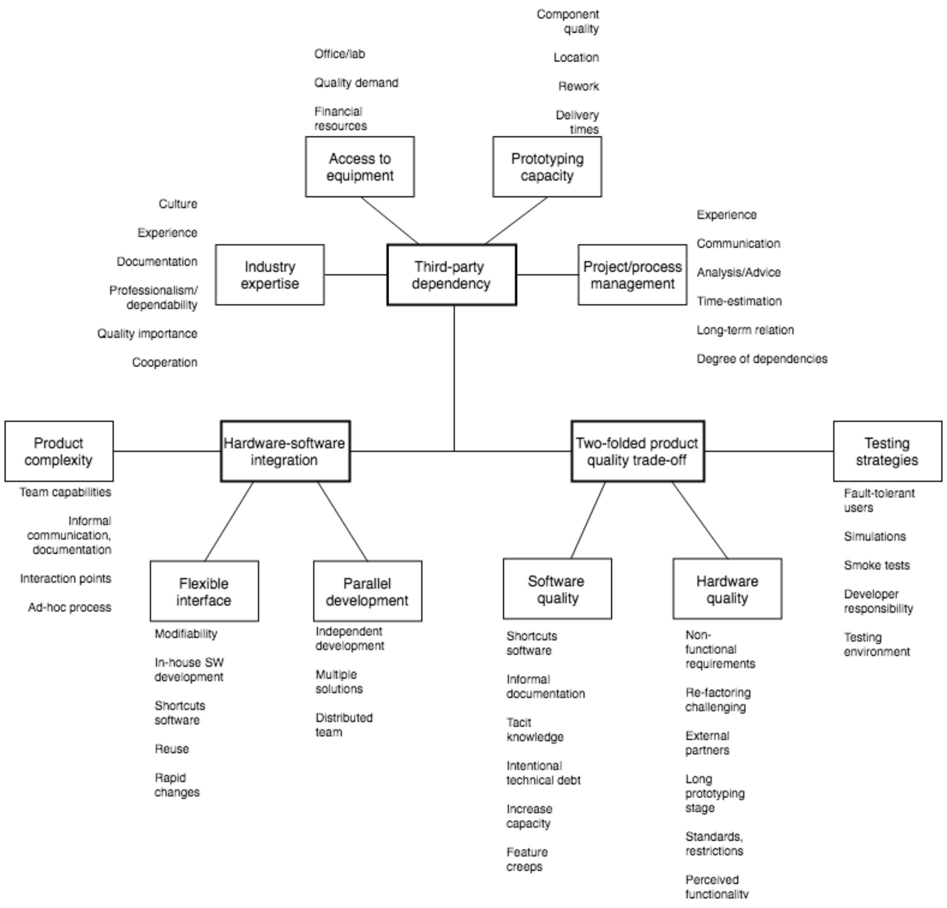


Figure 5.1: Model of higher-order themes unique to hardware startups

Summary. Table 5.5 presents the findings (i.e., current status from the investigated startups) based on our research questions: (RQ1) *How do hardware startups achieve agility during product development?* (RQ2) *How do hardware startups manage quality concerns of their products?* (RQ3) *How do hardware startups achieve balance between speed and quality?*

Research Question	Findings
RQ1	Hardware startups implement opportunistic agile practices. Rapid prototyping is achieved through evolutionary approaches, simple software side solutions, cooperation with local partners, and facilitation for parallel work on multiple solutions.
RQ2	Hardware startups utilize simulations and informal quality assurance practices to deal with intricate testing environments. Testing is an unsystematic process entrusted to each individual team member. They incur technical debt through shortcuts on flexible software side and because problem space testing is a difficult process.
RQ3	The competitive environment of hardware startups makes speed inevitable. Investing in hardware quality is necessary for bringing products fast to market.

Table 5.5: Summary of results

Chapter 6

The Trilateral Hardware Startup Model

Based on the thematic synthesis, we have created a model describing the context and overall engineering approach of hardware startups. The interview base of 61 pages of text allowed us to identify a total of three higher-order themes unique to hardware startups, before compiling the Trilateral Hardware Startup Model constituting a total of nine higher-order themes as illustrated in Figure 6.1. The rest of this chapter proceeds as follows: Section 6.1 presents the model notation, the three elements, and each of the nine higher-order themes (factors), constituting an overview of the most prominent findings from the case study. Section 6.2 presents a validation of the model. Implications of the model are discussed in section 7.4, and includes contribution and recommendations for future work. A comparison of our model to the Greenfield Startup Model can be found in section 7.5.

6.1 Model overview

Notation. This paragraph will explain the notation and structure of the model.

- *Elements* are represented as circles and are essential to the priorities and restrictions of product development in hardware startups.
- *Factors* are represented as rectangles and describe activities and context factors contributing to achieving important objectives (e.g., rapid development under restricted resources) in hardware startups. Depending on the type of project or product, each factor may significantly impact one or more of the elements resources, speed, and quality.
- Arrows indicate that there exists a relationship between elements and/or factors. There are several arrows in the model:

- Arrows between elements indicate that the elements have a direct influence on each other. An example is that restricted resources (e.g., initial capital) may directly slow down development speed in hardware startups (e.g., they won't be able to recruit required expertise due to salaries).
- An arrow from a factor to an element implies that the factor directly affects the element. An example is the arrow from *restricted resources* to *resources*. Hardware startups do not have access to unlimited financial and human resources, hence resources in hardware startups are restricted.
- An arrow from one factor to an arrow between two elements means that the factor has an effect on the relationship between the elements. An example is the arrow from *team proactivity* to the arrow between *resources* and *speed*. Due to restricted resources, hardware startups depend on the proactivity of their teams to drive development forward, hence teams are highly affecting speed (positively or negatively) in hardware startups.

The three elements. Based on our collected data, we identified connections to *quality, speed,* and *resources*, operating as core elements of the Trilateral Hardware Startup Model. Depending on the objective of a project, quality, speed, and resources are factors influencing product development. There may exist other elements that can contribute to describing the engineering activities in hardware startups (i.e., scope of work), however we found resources, quality, and speed to be the most prominent to the startup context.

Resources is the major element affecting hardware startups' ability to both achieve rapid development and high product quality. Experienced by most startups is that resources are restricted in most areas of the business, be it time, money, team capabilities etc. For software startups, lack of resources is the most significant factor operating their context (Paternoster et al., 2014). Limited access to resources sets strict restrictions and boundaries to product development in both software and hardware startups.

Operating in competitive business environments characterized by extreme uncertainty, hardware startups must strive to obtain speed. Through evolutionary prototyping, rapid development, and simplified solutions, they continuously experiment to identify markets and customers. The importance of speed in startups has been emphasized by The Lean Startup method where the main objective is to grow the business with maximum acceleration (Ries, 2011).

In pursuing development speed, product quality tends to be less prioritized. In the model, speed and quality are connected by a two-way arrow, illustrating a trade-off. Hardware startups are to a larger extent dependent on the quality of their products. Speed is achieved through simplified solutions on the software side while spending more time on the quality of hardware.

The nine higher-order themes can be seen as factors operating the hardware startup context. They impact each of the three elements or relations between them in various ways and extent. The following sub-sections will introduce each of the factors in greater detail.

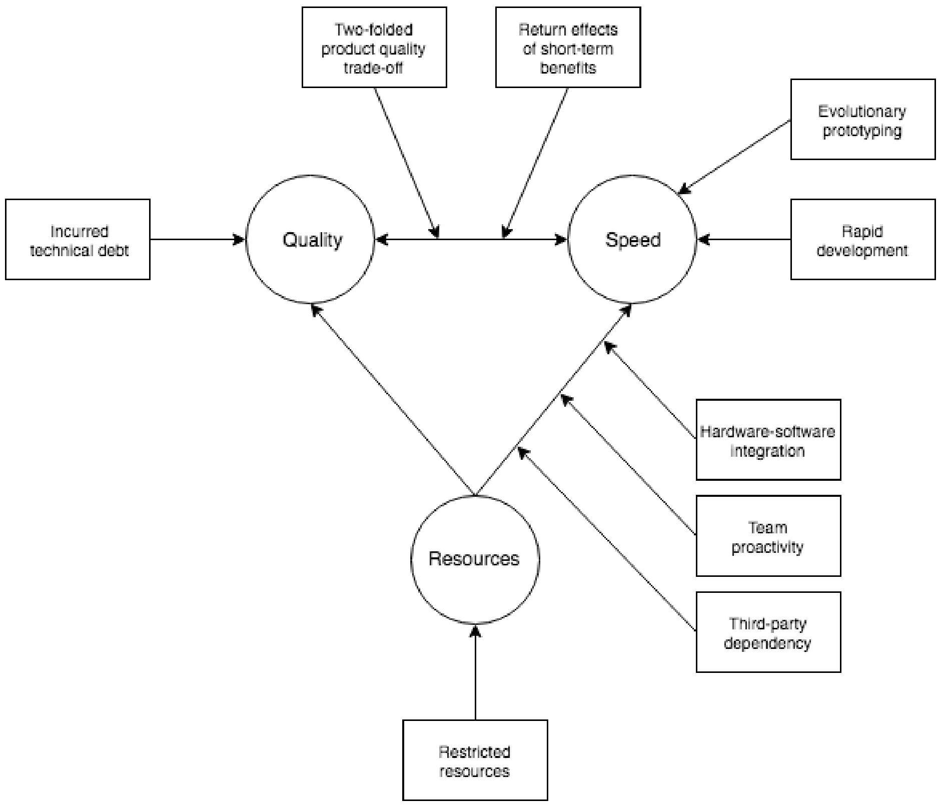


Figure 6.1: The Trilateral Hardware Startup Model

6.1.1 Restricted resources

Lack of resources is the major contextual factor affecting not only hardware startups, but practically every startup company. Software startups have a general lack of human, physical, and economical resources (Paternoster et al., 2014). These factors imply several constraints both as to how they manage and aim to grow their business and how they develop their products in extremely uncertain and dynamic market conditions (Coleman and O'Connor, 2008b). Hardware startups experience similar restricted resources, evermore severe and harmful than software startups. Since they rarely have the capacity to develop prototypes themselves, they greatly depend on third-parties to deliver ready-made or customized components in time. The resource-intensity and complexity of embedded systems development leave immense demands to the capability of development teams, however, there's restricted access to dedicated people with both technical and entrepreneurial skills. Poor economic conditions make recruitment challenging to perform. The severe lack of resources leads decision-making into a series of trade-offs balancing the various dimensions and needs of product and business development.

6.1.2 Team proactivity

The proactivity of the team significantly affects speed in a context of restricted resources. Anticipatory, change-oriented, and self-initiated team members are a necessity in the fast-changing, high-risk environment of startups. Hardware startups need team members dedicated to all aspects of the development process, including knowledge within the application domain, systematic development, software and hardware development, mechanical engineering, and experience of working with third-party companies. Working with several technology domains and external partners increase the complexity and lengthen the prototyping stage, and leave higher demands to skillful teams and entrepreneurial capabilities. Members should have experience from working with bigger product companies as stringent financial resources and small production batches make it hard for startups to find manufacturers invested in the startups' success. Attracting experienced and knowledgeable people is hard as startups rarely can provide good salaries, hence startups often consist of students with little or no experience from high-tech product development. Distributed development across big geographical distances and different cultures and languages can challenge the communication capabilities of the team. Communication is important to increase efficiency, reach goals, and avoid conflicts from misunderstandings. Since formal documentation practices imply documentation must be updated every time software is changed, documentation extensively relies on tacit knowledge to increasingly facilitate for rapid prototyping and implementation of new features. Effort and time can be spared when relying on the knowledge of team members.

6.1.3 Two-folded product quality trade-off

Being able to prototype fast for testing new ideas and product features is crucial to learn faster than competitors. Business experimentation to build new features is performed in small iterative cycles with minimal effort on product quality to achieve speed. Since software can be changed and modified quickly, shortcuts and workarounds are more easily taken on the software side of hardware startups' products. Software testing is not systematically performed, rather the responsibility of each developer. While pursuing speed for the software development of the prototype, the nature of hardware development inhibits hardware startups in achieving rapid prototyping. The importance of non-functional requirements and the dependability to third-parties are factors greatly affecting their ability to perform frequent release. It is challenging to implement changes and make improvements to hardware after the product has been produced and assembled. Product complexity and strict non-functional requirements will eventually make refactoring a complicated undertaking. The influence of product quality is a two-folded trade-off between hardware and software. Investing in hardware quality is necessary for realizing speed in software development.

6.1.4 Third-party dependency

Without industry knowledge and ability to mass-produce prototypes, hardware startups need external competence. Hardware startups depend on third-parties for hardware production and physical components. Third-party dependency is the most prominent factor influencing product development in hardware startups. Long production and shipping times, manufacturing defects, end-of-life components, cost of rework, communication, and culture differences are some central issues affecting hardware startups' ability to perform rapid prototyping and business experimentation. Because of limited financial resources and small production batches, it can be difficult to find manufacturers invested in the startups' success. Working with local vendors producing components of high quality at an affordable cost is advantageous. Long-term relationships with professional actors can enhance product quality and reduce the degree of dependencies. Access to prototyping equipment can reduce dependency to external partners, and have a positive effect on development time and prototyping costs, enabling faster problem space testing. Hardware development is a time-consuming process, and so minimizing necessary work for experimenting business ideas is essential.

6.1.5 Hardware-software integration

Hardware startups' dependability on third-parties for components and manufacturing inhibits both their flexible development approach and their ability to quickly develop prototypes. To achieve rapid prototyping for testing new ideas and product features, they facilitate for changes in software. Software can be modified quickly according to changing customer demands, allowing for frequent releases. A flexible interface between hardware and software can enable more parallel and independent development of hardware and software, and promote work on multiple solution methods. Another important asset provided by a flexible interface is the increased ability to handle product complexity, by allowing for parallel work and informal work-flow. Documentation is to a large extent based on informal communication among the team members. The complex nature of hardware development and the numerous interaction points require information exchange to be explicit, however, too much documentation is not feasible in early development stages. Formal meetings consolidating efforts in hardware and software development is unavoidable, synchronizing and prioritizing new tasks.

6.1.6 Evolutionary prototyping

Hardware startups usually start building a physical prototype to elicit requirements and achieve fast business experimentation. They extensively try to reuse software components, as physical components are easier to reuse with more refined prototypes. This is similar to an evolutionary approach, as they perform incremental improvements on an early low-resolution prototype. The approach can help them demonstrate problem/solution fit through discovering the needs of early customers, enhancing the effectiveness of the product and prevent exhaustion of their severely limited resources. Evolutionary prototyping promotes flexibility and reactiveness, however, hardware and hardware-related code is sensitive to frequent changes and refactoring due to the hard real-time requirements and third-party dependencies. Frequent changes might unconsciously change system behavior, and so rapid prototyping depends on pre-development system design and planning activities beyond the purpose of the evolutionary approach.

6.1.7 Rapid development

The most important priority of hardware startups is to achieve quick development speed. Testing and quality assurance practices are usually inferior to speed-related activities. The importance of testing new ideas and features on customers is crucial to learn faster than competitors, but achieving this in environments of severely restricted resources and third-party dependencies can be somewhat of a challenge. Hardware startups generally minimize any degree of process, preferring ad-hoc development approaches customized to their own needs. They seek to utilize a small, flexible team without any bureaucracy, capable of responding quickly to changes. Informal communication and workflows are favored to formal documentation practices. Having a skilled, boundary-spanning team often counterbalances the lack of process. The same relates to testing practices which highly depend on individual efforts, manual smoke tests, and simulations in early phases. By following an evolutionary prototyping approach, hardware startups aim to test an initial prototype to elicit requirements. Problem space testing is a resource-intensive activity since each prototype involves individual production costs, emphasizing the importance of maximizing valuable learning from each. Having a professional local vendor can help decrease delivery times and manufacturing defects. Access to prototyping equipment can decrease development time and costs as third-party dependencies are reduced, further enhancing the ability to perform problem space testing. A flexible hardware-software interface increases the level of parallel and independent development and can help hardware startups better manage quality concerns in later stages. Since improvements and changes to software are quicker and easier, hardware startups should keep software development in-house.

6.1.8 Incurred technical debt

As hardware startups accept that time to market is a more important objective than product quality, development teams take shortcuts and workarounds. New features are implemented in small, iterative cycles to perform rapid business experimentation, with minimal effort on quality assurance and documentation practices. Software features are implemented with a minimal amount of functionality. As the documentation would need to be updated for every change made to the code base, developers rely on their own knowledge instead of updating formal documentation. Since hardware startups rarely have the capacity to produce many prototypes, problem space testing becomes a challenging endeavor. The evolutionary approach increases the chance of feature creeps. Restricted resources and need for rapid development speed lead to the accumulation of technical debt.

6.1.9 Return effects of short-term benefits

With business growth as the main objective in early phases, hardware startups will eventually have to slow down development to meet the ever-increasing needs of established customers. The evolutionary approach promoting reuse of components will lead to components holding too low quality or features not contributing to the core-delivered value of the product. The numerous interaction points between software and hardware components are vulnerable to later changes. Updating or removing code base can potentially change the entire system behavior, as failing to meet timing and performance constraints can jeopardize the system operation. This means refactoring quickly becomes an immensely complex endeavour. Hardware startups favor informal communication and simple work flows. Shortcuts can speed-up development in early phases, but might cause a severe amount of rework in the long run. In the worst case, lack of documentation and quality can put all development on hold. When scaling the business to a larger customer base, new employees are needed. Tacit knowledge makes it hard to integrate new people and can inhibit further growth. The introduction of more rigorous processes is necessary in the long run, but will forcibly deny the initial speed and flexibility of hardware startups.

6.2 Model Validation

Evaluation with the Software Startup Research Network. SSRN is a global network of scientists within software startup research, many of which have made scientific contributions to our study. Two of the authors of the GSM are part of the network, implying that they have in-depth knowledge about the research area and relevant research method. During the closing stages of our research project we attended a meeting with 17 people from the network where we held a 30 minute presentation, presenting and discussing the results of our research. The network provided useful feedback for making the final improvements to the model and the Master thesis. The presentation was valuable for placing findings and results within theory, and confirming its novelty and contribution to the research community.

Figure 6.2 illustrates an earlier version of the created model presented at the meeting with SSRN. The feedback from the network, along with our response to each comment, can be summarized as follows:

(1) "*Where does the triangle come from? In project management one can basically only choose two points.*"
Following this comment, we included an explanation of each of the three elements in section 6.1. The model should not be confused with traditional project management models where practitioners must choose between two out of three elements. Rather, the model illustrates the three main elements affecting engineering approaches of hardware startups.

(2) "*How is the work connected to, or builds on the Greenfield Startup Model?*"
The empirical investigation combined with knowledge from the GSM lead to the creation of the THSM. The two models illustrate engineering approaches of software and hardware startups respectively. Through a comparison of the models (section 7.5), we identify similarities and differences, exploring how they are connected and inter-related while operating under similar constraints (i.e., the startup context).

(3) "*What does the dotted line indicate?*"
In response to this comment, the dotted lines between elements were replaced with arrows to reflect and clarify the direction of the respective relationships.

(4) "*What is the usefulness of the model?*"

To address this comment we have improved the conclusion to emphasize the usefulness for practitioners and researchers. The model provides practitioners with a better understanding and awareness of their own context by giving a simple illustration of the hardware startup context and the associated engineering approach. For researchers the model provides a first step towards understanding hardware startups, outlining directions for future work.

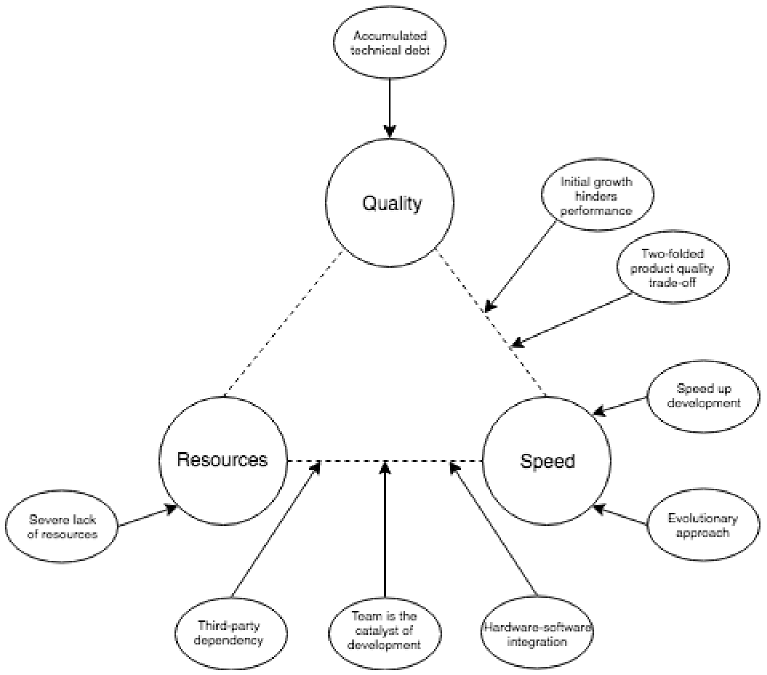

Accumulated technical debt
Quality
Initial growth hinders performance
Two-folded product quality trade-off
Speed up development
Resources
Speed
Severe lack of resources
Evolutionary approach
Third-party dependency
Team is the catalyst of development
Hardware-software integration

Figure 6.2: Earlier version of the created model

Category representation in investigated cases. In the final stages of the Master thesis all 13 startups participating in the study received the newly developed model along with the corresponding description. From the obtained feedback we performed a mapping of the model's factors to each of the investigated cases. Table 6.1 provides an overview of the mapping. Empty columns indicate that the startup did not respond before the deadline for delivering the Master thesis. The mapping indicates the possible generalization of THSM to early-stage European hardware startups.

Among the startups responding to the evaluation of the model only one startup related to the factor *return effects of short-term benefits*. The low representation might be a result of the lifecycle stage of the investigated startups. Since nine of the startups participating in the study are between one and two years old, most of them are still in a lifecycle stage were speed and business experimentation are the main priorities. Future research can investigate the consequences of incurred debt in hardware startups, and how it influences future growth. The model should also be verified by exploring engineering activities of hardware startups at different lifecycle stages.

Seven of the nine startups responded that rapid development is central to their development approach. This implies that similarly to software startups, speed is the main priority of hardware startups. Hardware startups utilize an evolutionary approach, however they need to overcome challenges posed by third-parties. Based on the low release frequency of hardware startups, future work should explore options to speed-up the development process. The potential of throwaway prototyping in hardware startups can be investigated further, as none of the participating startups utilized such approaches for business experimentation.

The nature of hardware development requires increased attention to hardware quality. Four startups responded that quality assurance was two-folded, with stricter demands to hardware. Early-stage startups may not realize the long-term consequences of not investing in hardware quality. With larger production volumes, hardware defects can considerably slow down development and harm product integrity. Future work should provide startups with quality assurance practices consistent with restricted resources.

> S4 - *"Hardware errors can be catastrophic, especially if the production volume is large."*

As the startups only received a brief description of the model and related factors, participants' understanding of the model should be considered. In addition, poor understanding of their own context could prevent them from responding correct. As the response varied between two and seven factors, this can be one reason for the inconsistency. However, the current validation provides an indication to the generalizability of the model.

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13
Restricted resources	X			X	X		X	X	X	X			
Team proactivity	X	X		X	X	X	X		X	X			
Two-folded product quality trade-off	X			X				X		X			
Third-party dependency	X	X		X		X	X			X			
Hardware-software integration	X	X		X		X	X			X			
Evolutionary prototyping	X	X		X	X	X	X			X			
Rapid development	X	X		X	X		X	X		X			
Incurred technical debt	X			X			X	X					
Return effects of short-term benefits										X			

Table 6.1: Mapping between themes and cases *(X indicate match between startup and factor)*

Chapter 7

Discussion

In this chapter we discuss the findings from the cases presented in chapter 5 and the model introduced in chapter 6. Section 7.1 introduces the use of agile practices in hardware startups. Section 7.2 presents how hardware startups manage the quality of their products. Section 7.3 introduces how hardware startups balance development speed and product quality. Section 7.4 presents implications of the THSM. Finally, section 7.5 provides a comparison of the THSM to the GSM to highlight how the hardware startup context is different from that experienced by software startups.

7.1 Agility in hardware startups

Our study presents the characteristics of high-tech product development in hardware startups. In this fast-changing and high-risk environment iterative and incremental development approaches is a necessity, which is what agile intend to provide (Ronkainen and Abrahamsson, 2003). However, not all agile practices are suited for hardware and hardware-related software development. The investigated cases give an overview of current operationalization of agile practices in hardware startups.

Hardware startups use agile planning in their product development. Short-term planning is important for responding to changes in an environment of high uncertainty. Among the startups practicing agile seven planned their product development pipelines one or two weeks ahead. As hardware development usually requires more time than software development one might argue that Sprint duration of one or two weeks are too short for any progress to be made. However, the startups managed to divide the work into small tasks possible to execute in a single week. Startups operating with longer time frames experienced less chaotic environments as they cooperated closely with professional business partners and customers.

Even though short-term planning is adaptable to hardware development, frequent release is not implemented to the same extent. As non-functional attributes need to be assured at the prototyping stage (Nguyen-Duc et al., 2018), and hardware startups deal with third-party dependency, release frequency is low compared to software startups. Most of the investigated startups practiced release frequency of six months or more. This is disadvantageous as continuous experimentation is important for startups to grow (Fagerholm et al., 2014).

Refactoring is about improving the non-functional attributes of a component. Our research indicate that regular refactoring is not practiced in hardware startups, either for software or hardware development. Prototyping consists to a large degree of shortcuts and workarounds, especially for the software components. The nature of hardware development is not compatible with regular refactoring, as frequently redesigning components involves significant costs. This relates to software startups as well. Research state that refactoring rarely is implemented in the early stages of the startup, but as the startup grows, returning the accumulated technical debt is needed to meet more quality-demanding customers and scalability issues (Giardino et al., 2016).

Testing is central to embedded system development, as hardware startups need to assure non-functional attributes at an early stage. Testing must ensure conformance between hardware and hardware-related software. However, the test-first approach is problematic because of the severe memory and performance constraints of embedded systems (Ronkainen and Abrahamsson, 2003), in addition to the restricted resources of hardware startups. Hence, test-first may not be applicable to product development in hardware startups. Among the investigated startups testing procedures were to a large degree a responsibility of each individual developer. Only the more mature startups implemented systematic formal testing processes.

Our findings argue that daily standup meeting is not used in hardware startups. The small team sizes implies that formal processes to aid communication between developers are not necessary as informal communication happens frequently. Several of the startups had however implemented weekly meetings to synchronize hardware and software development, similar to retrospective and review meetings. The co-design of embedded system development implies that development usually is distributed, hence communication between teams is a central aspect of the hardware and software integration efforts necessary in hardware startups.

The use of agile practices in hardware startups is highly opportunistic. The combination of speed-up development and small team sizes affect what practices are implemented. Even if some startups related to agile methods in the form of simplistic versions of Scrum, current implementation of agile practices are found to be ad-hoc. Hardware startups follow an iterative and incremental approach, with short-term planning to allow for flexibility. As in software startups, speed is preferred over quality-related practices. For agile methods to be successfully practiced in hardware startups, methods must be adapted to the stringent constraints of both embedded development and the startup context. The conflicting requirements for quality and speed mean development methods will need to balance strict process for hardware development, while allowing for speed and flexibility in software development. Preliminary architecture and up-front design to meet real-time requirements are activities not in line with current agile guidelines (Kaisti et al., 2013).

In addition to agile process, the investigated hardware startups achieved agility by facilitating for simultaneous work on multiple possible solutions. Implementation of ready-made or outsourced components can be a significant struggle as hardware startups rarely develop all components themselves. System design and architectural decisions are made in advance of development, and may greatly affect later system integration of components. As development in hardware startups can be considered a test of feasibility, development methods should facilitate for experimentation of multiple solution methods.

The nature of hardware development makes embedded systems sensitive to rapid changes in hardware or hardware-related software. The smallest changes may impact important quality attributes or behavior that can be difficult to detect (Ronkainen and Abrahamsson, 2003). However, several of the investigated startups invested heavily in the design and architecture of the interface between hardware and hardware-related software to facilitate for flexibility and quick changes on the software side of their product. As software can be changed multiple times a day, this is essential to the speed of prototyping.

7.2 Quality of high-tech products

Our research presents knowledge on testing and quality assurance practices in hardware startups. Hardware startups' need for development speed is often at the expense of product quality. Instead of applying best practice engineering principles, we found that development teams prefer simple solutions to achieve rapid business growth. Speed-related activities lead to accumulation of technical debt, which eventually inhibit further business growth. Literature on software startups state that shortcuts in quality, design, and infrastructure will lead to long-term quality problems, eventually inhibiting learning and customer satisfaction (Giardino et al., 2016).

The complexities and uniqueness of hardware development imply that hardware startups need to prioritize product quality differently from software startups in order to speed-up development. The investigated startups tried to facilitate for changes in the software side of their products while keeping the amount of hardware rework to a minimum as it was challenging and time-consuming to implement changes and improvements in hardware parts after the product was produced and assembled. Hardware quality is often necessary to meet real-time performance requirements of embedded systems (Ronkainen and Abrahamsson, 2003). Enabling the hardware-software co-operation is an intricate process due to the complex control and testing support required over hardware, and the fast time-to-market cycles require simultaneous software and hardware design (Ronkainen et al., 2002). The hardware startups invested in a simple interface combined with a skilled team to increase the amount of parallel development, facilitating for two largely independent development processes of hardware and software.

While forcing rapid changes on the flexible software side, the hardware startups incurred intentional technical debt. Since the software developers constantly worked with the code base, they relied on tacit knowledge instead of formal documentation. Hardware documentation seemed to be of higher importance due to the many stakeholders involved in hardware development. Intentional technical debt is a frequent problem in software startups, but can be even more harmful for hardware startups due to the change-sensitivity of the numerous complex hardware-software interactions (Ronkainen and Abrahamsson, 2003). Refactoring of code base can cause changes in system behaviour or even jeopardize system operation. Even if software shortcuts make sense in the short-run, our findings indicate that the complex nature of high-tech products may cause a severe amount of rework in the long-run. Hardware startups should invest in documentation tools to lower the barriers for formal documentation. Adoption of agile methods has proven to be efficient in reducing error rates (Albuquerque et al., 2012), however current usage of such is restricted to a subset of agile practices customized the individual needs of hardware startups.

The investigated hardware startups incurred unintentional technical debt due to the difficulty of testing problem space. They performed usability and acceptance tests on a small group of pilot customers, as a lack of financial resources and third-party dependencies (e.g., delivery times) made it challenging to test the product on a broader spectre of customers. By immediately building a physical prototype, the startups focused on validating a product instead of discovering a problem space. This is similar to the failure reasons of software startups described in the *Behavioral Framework* (Giardino et al., 2014b), as they focus on making their customer acquisition processes more efficient rather than testing the demand for a functional product. The hardware startups' inability to produce many prototypes inhibited business experimentation and lead to feature creeps. Feature creeps in hardware startups may similarly to software startups be harmful to the production and maintenance of core functionality (Nguyen-Duc et al., 2017b).

We found that testing practices were implemented to various extent among the hardware startups, among other things, because the testing environment was different from the development environment. Memory and performance constraints can also affect hardware startups' testing ability (Ronkainen and Abrahamsson, 2003). The investigated startups relied on individual developers' efforts to ensure quality of new functionality. Manual smoke tests and simulations were favored to professional engineering activities. Findings indicate that rigorous testing practices were not implemented before later lifecycle stages. Specific testing approaches are required for hardware startups to deal with quality constraints of complex high-tech products under restricted resources. In addition to ordinary software tests, we argue that hardware startups need to apply tests focusing on the functionality of hardware and the related software. Automatic tests is one measure hardware startups can use to identify failures in early stages for avoiding starvation of resources.

This study is only an initial investigation into quality assurance practices in hardware startups. However, the study highlights the compromise hardware startups makes between quality and speed. Quality is of higher significance, and more research should be provided identifying valuable activities and approaches for hardware startups dealing with restricted resources. We encourage researchers to explore the long-term effects of technical debt, as our results are based on a small sample of early-stage hardware startups. Hardware startups need specific guidelines for performing problem space testing, and research should verify the consequences of its absence. In addition, future research should investigate how hardware startups can ensure safety and security standards when developing highly safe systems, following standards like IEC61508 (Japan, 2012). The results are partly based on managerial viewpoints, hence missing important links to everyday testing activities performed by engineers and developers.

7.3 Balancing speed and quality of high-tech product development

Balancing speed and quality in the hardware startup context is somewhat of an intricate endeavour. Agile practices related to speed are commonly used by both software startups and established software companies alike, whereas the associated dependencies (i.e., third-parties and hardware-software integration) and non-functional requirements of hardware development pose restrictions to agile's applicability in hardware startups. From an engineering perspective, we see a need for specific guidelines for how hardware startups can improve their current speed-adding activities and practices.

We have seen that hardware startups' overall strategy is to spend more time on quality-adding activities in hardware, while speeding up software development. We argue that a considerable challenge for agile in hardware startups is to provide more sophisticated methods for recognizing the required amount of documentation at any given time, rather than sticking to the general idea from the software startup context that working software is sufficient. Information exchange in hardware startups cannot solely consist of face-to-face communication with source code as the only documentation (Ronkainen et al., 2002). Developers in hardware startups need knowledge of how to balance quality demands and the need for rapid experimentation, which is similar to the developer's dilemma in software startups (Terho et al., 2016). Not only must agile methods facilitate for the independent needs and co-design of hardware and software, they also have to offer more refined documentation and testing activities while maintaining the informal workflow of the team. The need for explicit information exchange implies that hardware startups need customized solutions (e.g., documentation tools or testing frameworks) to meet the ever-increasing quality demands of their products. A method adjusted to the demands of hardware and software can allow for distributed development teams simultaneously working on multiple solutions and technologies.

One of the most resource-intensive aspects related to hardware and embedded systems development is the number of third-party dependencies. Hardware startups need processes fitted to their relationships with external partners. Important objectives are to minimize the amount of idle time and optimize the amount of value-adding activities during production lead times. As all hardware startups (to various extent) are dependent on external partners for components or production of prototypes, processes must assist in managing these relationships (e.g., balancing and prioritizing different stakeholder needs both in the short and long run). External partners have their own customer deadlines and time schedules, and production and delivery times may vary accordingly. Hardware startups need flexibility towards third-parties and methods to maintain product development effectiveness. External pressure and expectations are majorly affecting business operation, leaving great demands on processes and project management activities to be suited the diverse needs posed by the ever-growing crowd of hardware startups.

The unique requirements to product quality and external stakeholders are extensively affecting hardware startups' ability to frequently release new products and features, inhibiting their ability to perform problem space testing. Evolutionary prototyping is widely employed among hardware startups, but as they have different needs than what the method currently supports (e.g., pre-architecture planning), hardware startups should consider alternative problem space testing methods. Lack of human and financial resources are severely affecting them since physical prototypes are associated with high individual development cost and time. As the early stages of startups not only should be about failing fast, but failing cheap (i.e., eliminating waste), hardware startups should extensively make use of throw-away prototypes. Throw-away prototypes can help startups visualize and present business ideas at an early stage, and reduce costs and time. The potential of throw-away prototyping in hardware startups should be further explored.

7.4 Implications of the Trilateral Hardware Startup Model

The THSM explains the priorities of hardware startups in their engineering approach and provides a simple illustration of the hardware startup context. The model presents the specific needs for process in managing the relationship between quality and speed under restricted resources. It can help practitioners obtain a better understanding and awareness of their own context. This is useful for hardware startups in understanding the underlying motivation for introducing specific practices and activities, and why some practices may counteract the overall objective of startups. The improved understanding will help practitioners in making technical and business-related decisions of sustainable character. With the THSM we aim to contribute to the research community by providing a first step towards understanding the context and engineering activities in hardware startups, and outline potential areas for future research. Researchers can use the model to draw parallels to other contexts (e.g., software startups), enabling transfer of research results between the contexts. Since the model includes aspects applicable to software startup as well, it presents new insights from a different viewpoint than what is provided by current literature.

Even if the model is based on a small sample of early-stage European hardware startups, it is possible to draw initial conclusions and directions for future work. Currently, most research has focused on how to speed-up development with restricted resources, whereas little work has been done to improve quality under similar constraints. As hardware startups need more attention to hardware quality to allow for evolutionary prototyping and speed, there should be engineering approaches describing how hardware startups can manage the relationship between restricted resources and increased quality demands.

The increased demand for quality in the prototyping stage also implies the support needed for hardware startups developing highly safe systems. Hardware startups develop a wide range of products, including systems that may cause critical situations if system operation fails. Testing these products require a certain level of quality and safety assurance. More work should be provided hardware startups for developing embedded systems that can be used safely.

Future work should verify the model with other startup companies to find its applicability in other environments (e.g., lifecycle stages, countries etc.), enabling generalization to a larger startup audience. More investigations should be undertaken to understand the role of scope in the engineering activities of hardware startups, and how it can be included in the model. Discovering the right scope can greatly improve development speed, by identifying the necessary features and effort. Similar findings from the systematic mapping study suggest customer collaboration processes as a potential area for future research, allowing startups to test the problem before releasing the product to market. Seeing this research as an initial step towards understanding engineering approaches in hardware startups, more work should be undertaken to explore the specific factors of hardware startups (i.e., hardware-software integration, third-party dependency, and trade-off between hardware and software quality).

7.5 Comparing the Trilateral Hardware Startup Model to the Greenfield Startup Model

With the Greenfield Startup Model serving as the basis for the Trilateral Hardware Startup Model, we provide a comparison of the two models. A comparison of main similarities and differences will allow for a better understanding of the respective contexts. Table 7.1 illustrates factors operating the two contexts. The Trilateral Hardware Startup Model column shows the main characteristics of the hardware startup context, while the Greenfield Startup Model column shows the characteristics of the software startup context. The rest of this section introduces the factors.

Factors	Trilateral Hardware Startup Model	Greenfield Startup Model
Restricted resources	Yes	Yes
Team proactivity	Full-stack software, hardware, mechanics, and electronics engineers	Full-stack software engineers
Two-folded product quality trade-off	Hardware high priority Software low priority	Software low priority
Third-party dependency	Yes	To some extent
Hardware-software integration	Yes	No
Evolutionary prototyping	Yes	Yes
Rapid development	Yes, but not at all costs	Yes
Incurred technical debt	Yes	Yes
Return effects of short-term benefits	Yes	Yes

Table 7.1: Factors operating the context of hardware and software startups

Restricted resources. Limited access to resources is a commonality among all startups and the main reason for the uncertainty of development strategies both in hardware and software startups. Similar to software startups, hardware startups struggle with time-shortage, limited human resources, limited access to expertise, and limited financial resources. Lack of financial resources can be more severe for hardware startups as they require more initial capital due to the costs associated with rapid prototyping of hardware. This includes material costs and manufacturing fees, as compared to software development which mainly is associated with labor costs (Wei, 2017). Access to prototyping equipment can help hardware startups perform more rapid prototyping, however, their lack of capital and facilities (e.g., labs or proper testing environments) hamper their prototyping capacity.

Team proactivity. Similar to hardware and software startups is that developers have great responsibilities and employ multiple-roles. Software startups favor generalists and full-stack engineers who are able to quickly learn new technologies and rapidly move among multiple tasks. Skilled developers in small co-located teams with informal workflows are essential for high-speed development. Hardware and embedded systems development demand team members possessing a wider range of expertise than that of software development, as development depends on knowledge within hardware, mechanics, and electronics in addition to software. As to this, hardware startups demand more boundary-spanning capabilities of their team members than what is required for software startups. Another commonality of hardware and software startups is that they cannot afford to offer good salaries. Therefore their teams often consist mainly of students or graduates with little or no industry knowledge. For hardware startups especially, lack of third-party experience can largely slow down the development process.

Two-folded product quality trade-off. Whereas product quality has low priority in software startups, product quality in hardware startups can be seen as a two-folded trade-off between the quality of software and hardware. Hardware quality is often necessary to meet non-functional requirements and dependency towards third-parties. Although the software and hardware contexts share many of the same characteristics, the specific hardware tasks and dependencies imply that speed sometimes is achieved through quality-adding activities. Shortcuts are mainly taken on the flexible software side. Hardware changes are kept to a minimum, hence requiring greater quality focus in early stages.

Third-party dependency and hardware-software integration. Represented as two separate themes in the Trilateral Hardware Startup Model, they pose the main distinction points between hardware and software startups. Software startups often depend on third-party solutions (e.g., cloud computing) to achieve speed, but the extent of the dependency is significantly different from what hardware startups experience. While software startups choose to use third-party solutions to speed-up development, hardware startups are genuinely dependent on third-parties, a dependency slowing down development in hardware startups. Hardware-software integration is an activity not evident in software startups. Software startups develop products or services usually consisting only of software components, whereas hardware startups have to balance between hardware and software components and development. A well-functioning hardware-software integration can decrease the degree of third-party dependency and help hardware startups deal with the increasing product and development complexity.

Evolutionary prototyping. To enable quick customer verification, both software and hardware startups follow an evolutionary approach where small iterations are performed on an initial low-resolution prototype. Software startups favor this approach to avoid ”over-engineering the system” by building complex not-validated functionalities. The nature of hardware development requires hardware startups to invest more in pre-development of system design, which is necessary to achieve rapid speed at later stages. Hardware rework is a costly and time-consuming activity since each prototype is associated with individual development cost and time. Designing an architecture facilitating changes in software can enhance their ability to perform problem space testing.

Rapid development. The most important priority in early startup stages is to speed up development. Hardware and software startups generally minimize the effort spent on introducing processes to maintain the flexibility of their small co-located teams. Even if agile practices are suited for software development, software startups ignore them to accommodate the flexibility of their teams. The complex dependencies of hardware development are not compatible with the current scope of agile practices like Scrum, resulting in them utilizing customized methods. Although hardware and software startups aim for speed, different contextual factors affect its achievement in the respective contexts. Third-party dependency, unique product quality demands, team capabilities, and hardware-software integration are factors implying that hardware startups sometimes have to prioritize quality over speed.

Return technical debt. Both hardware and software startups incur technical debt, which eventually will hinder further performance and business growth. As both take shortcuts on software, spending little time on communication and documentation activities, they incur intentional debt. Increased customer base leaves stricter requirements for performance and maintainability, which require refactoring of the code base. Refactoring is a challenge in software startups due to the fear of changing a working product. This relates to hardware startups as well, but also as hard real-time requirements and numerous complex interactions between hardware and software components can be challenging. Both hardware and software startups may struggle with unintentional technical debt if business experimentation is leaved out. Hardware startups’ inability to produce numerous prototypes affects their opportunity for problem space testing, leading to the implementation of unnecessary features. For software startups, feature creeps are easier to handle since they don’t have the same dependencies, including reduced need for initial system architecture. Even if both software and hardware startups incur technical debt, the long-term effects of it can be more harmful to hardware startups.

Chapter 8

Conclusion

Hardware startups develop physical products with mixed hardware and software components, requiring expertise within a broad range of technological fields. In addition to software development hardware startups deal with production and logistics issues, factors implying higher initial financial and human investments than what is experienced by software startups. From a multiple-case study investigating 13 hardware startups, this Master thesis presents the role of engineering activities from idea conceptualization to a launched product, and factors influencing development speed and agility. The findings of this study led to the following three themes *third-party dependency*, *hardware-software integration*, and *two-folded product quality trade-off* operating the hardware startup context. Thus, the study contributes to the area of startup engineering as it draws from the *Greenfield Startup Model* and extends it with the three unique themes leading to the creation of the *Trilateral Hardware Startup Model*.

Our research results indicate that hardware startups achieve rapid prototyping through evolutionary approaches, simple software side solutions, and opportunistic agile practices. Hardware startups incur technical debt, both due to informal testing and quality assurance activities, and because prototyping capability and intricate testing environments inhibit their ability of testing problem space (i.e., leading to feature creeps). Testing is an unsystematic process mainly entrusted to each individual team member. The competitive environment of hardware startups makes speed inevitable, where investing in hardware quality will be necessary for bringing products fast to market.

With the THSM, this study explains the priorities of hardware startups in their engineering approach, providing a simple illustration of the hardware startup context. It presents the specific needs for process in managing the relationship between quality and speed under restricted resources, providing practitioners with a better understanding and awareness of their own context. The improved understanding will help practitioners in making technical and business-related decisions of sustainable character.

For researchers, the THSM provides a first step towards understanding the context and engineering activities in hardware startups, outlining potential areas for future research. Future work should verify the model with other startup companies to find its applicability in other environments, enabling generalization to a larger startup audience. More investigations should be undertaken to understand the role of scope in the engineering activities of hardware startups, and how it can be included in the model. In addition, the three specific factors should be further explored in later studies. As hardware startups need more attention to hardware quality to allow for evolutionary prototyping and speed, there should be engineering approaches describing how hardware startups can manage the relationship between restricted resources and increased quality demands.

There are identified several limitations to this study. Having based our study on qualitative measures, results and implications are subject to bias. To mitigate the risk of misunderstandings or wrong interpretations, both researchers attended all interviews. Whenever possible, interviews were performed face-to-face on-site. Recordings were transcribed and translated shortly after each interview to ensure respondents' meanings were preserved. Another limitation is the insufficient knowledge on technical decisions and product development challenges provided by some interviewees (i.e., knowledge of business executives is often based on managerial viewpoints). The results would benefit from a greater amount of participants providing insights into every-day engineering activities of hardware startups.

Another shortcoming to the study is the diversity of the investigated startups, as the selection constituted early-stage European hardware startups. The study would profit from a wider collection of data, both to discover more relevant themes and to ensure credible conclusions (i.e., generalizability of the results). The model might not necessarily be applicable to the global population of hardware startups. Further investigations of hardware startups operating in different markets, lifecycle stages, and various geographical locations can improve the reliability of the research results.

Bibliography

Aernoudt, R., 2004. Incubators: tool for entrepreneurship? Small Business Economics 23 (2), 127–135.

Agnew, H., 2017. Emmanuel macron thinks big in vision for french tech unicorns. Access date: 2017-10-26.
URL `https://tinyurl.com/yaw8btmv`

Albuquerque, C. O., Antonino, P. O., Nakagawa, E. Y., 2012. An investigation into agile methods in embedded systems development. In: International Conference on Computational Science and Its Applications. Springer, pp. 576–591.

Alvarez, S. A., Barney, J. B., 2007. Discovery and creation: Alternative theories of entrepreneurial action. Strategic entrepreneurship journal 1 (1-2), 11–26.

Alves, C., Pereira, S., Castro, J., 2006. A study in market-driven requirements engineering.

Bajwa, S. S., Wang, X., Duc, A. N., Abrahamsson, P., 2016. How do software startups pivot? empirical results from a multiple case study. In: International Conference of Software Business. Springer, pp. 169–176.

Bajwa, S. S., Wang, X., Duc, A. N., Abrahamsson, P., 2017. "failures" to be celebrated: an analysis of major pivots of software startups. Empirical Software Engineering 22 (5), 2373–2408.

Basili, V. R., 1992. Software modeling and measurement: the goal/question/metric paradigm. Tech. rep.

Baskerville, R., Ramesh, B., Levine, L., Pries-Heje, J., Slaughter, S., 2003. Is" internet-speed" software development different? IEEE software 20 (6), 70–77.

Blank, S., 2012. The startup owner's manual: The step-by-step guide for building a great company. BookBaby.

Blank, S., 2013a. The four steps to the epiphany: successful strategies for products that win. BookBaby.

Blank, S., 2013b. Why the lean start-up changes everything. Harvard business review 91 (5), 63–72.

Bosch, J., 2016. Speed, data, and ecosystems: The future of software engineering. IEEE Software 33 (1), 82–88.
URL http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7368022

Bosch, J., Olsson, H. H., Björk, J., Ljungblad, J., 2013. The early stage software startup development model: a framework for operationalizing lean principles in software startups. In: Lean Enterprise Software and Systems. Springer, pp. 1–15.

Bourque, P., Fairley, R. E., 2014. Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0. IEEE Computer Society Press.

Braun, V., Clarke, V., 2006. Using thematic analysis in psychology. Qualitative research in psychology 3 (2), 77–101.

Brooks, F., Kugler, H., 1987. No silver bullet. April.

Brown, N., Cai, Y., Guo, Y., Kazman, R., Kim, M., Kruchten, P., Lim, E., MacCormack, A., Nord, R., Ozkaya, I., 2010. Managing technical debt in software-reliant systems. In: Proceedings of the FSE/SDP workshop on Future of software engineering research. ACM, pp. 47–52.

Carmel, E., 1994. Time-to-completion in software package startups. In: 1994 Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences.

Chanin, R., Pompermaier, L., Fraga, K., Sales, A., Prikladnicki, R., 2017. Applying customer development for software requirements in a startup development program. In: Proceedings of the 1st International Workshop on Software Engineering for Startups. IEEE Press, pp. 2–5.

Chen, E., 2015. Bringing a Hardware Product to Market: Navigating the Wild Ride from Concept to Mass Production. CreateSpace Independent Publishing Platform.

Chesbrough, H. W., 2006. Open innovation: The new imperative for creating and profiting from technology. Harvard Business Press.

Chicote, M., 2017. Startups and technical debt: Managing technical debt with visual thinking. In: 2017 IEEE/ACM 1st International Workshop on Software Engineering for Startups (SoftStart), 21 May 2017. 2017 IEEE/ACM 1st International Workshop on Software Engineering for Startups (SoftStart). Proceedings. IEEE Computer Society, pp. 10–11.
URL http://dx.doi.org/10.1109/SoftStart.2017.6

Clark, G., Couturier, J., Moonen, T., 2017. Oslo: State of the city. Report, access date: 2018-03-01.
URL https://issuu.com/oslo2015/docs/oslostateofthecity_2017/1?ff=true&e=16401528/47513404

Clarke, P., O'Connor, R. V., 2012. The situational factors that affect the software development process: Towards a comprehensive reference framework. Information and Software Technology 54 (5), 433–447.

Coleman, G., O'Connor, R., 2008a. Investigating software process in practice: A grounded theory perspective. Journal of Systems and Software 81 (5), 772–784.

Coleman, G., O'Connor, R. V., 2008b. An investigation into software development process formation in software start-ups. Journal of Enterprise Information Management 21 (6), 633–648.
URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-55349133834&doi=10.1108%2f17410390810911221&partnerID=40&md5=9a7aca62e6f24c6416fd3034dbb66b0a

Crowne, M., 2002. Why software product startups fail and what to do about it. evolution of software product development in startup companies. In: Engineering Management Conference, 2002. IEMC'02. 2002 IEEE International. Vol. 1. IEEE, pp. 338–343.

Cruzes, D. S., Dyba, T., 2011. Recommended steps for thematic synthesis in software engineering. In: Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on. IEEE, pp. 275–284.

Cunningham et al., W., 2001. The agile manifesto. Access date: 2017-11-12.
URL http://www.agilemanifesto.org

Dahlstedt, A., 2003. Study of current practices in market-driven requirements engineering. In: Third Conference for the Promotion of Research in IT at New Universities and University Colleges in Sweden.

DiResta, R., Forrest, B., Vinyard, R., 2015. The Hardware Startup: Building Your Product, Business, and Brand. " O'Reilly Media, Inc.".

Edison, H., Khanna, D., Bajwa, S. S., Brancaleoni, V., Bellettati, L. U., 2015. Towards a software tool portal to support startup process. In: 16th International Conference on Product-Focused Software Process Improvement, PROFES 2015, December 2, 2015 - December 4, 2015. Vol. 9459 of Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer Verlag, pp. 577–583.
URL http://dx.doi.org/10.1007/978-3-319-26844-6_43

Eisenhardt, K. M., 1989. Building theories from case study research. Academy of management review 14 (4), 532–550.

Eloranta, V.-P., 2014. Towards a pattern language for software start-ups. In: 19th European Conference on Pattern Languages of Programs, EuroPLoP 2014, July 9, 2014 - July 13, 2014. Vol. 09-13-July-2014 of ACM International Conference Proceeding Series. Association for Computing Machinery.
URL http://dx.doi.org/10.1145/2721956.2721965

Fagerholm, F., Guinea, A. S., Mäenpää, H., Münch, J., 2014. Building blocks for continuous experimentation. In: Proceedings of the 1st international workshop on rapid continuous software engineering. ACM, pp. 26–35.

Garbajosa, J., Magnusson, M., Wang, X., 2017. Generating innovations for the internet of things: agility and speed. In: Proceedings of the XP2017 Scientific Workshops. ACM, p. 10.

Gartner, 2017. Gartner top strategic predictions for 2018 and beyond. Access date: 2018-02-15.
URL `https://www.gartner.com/smarterwithgartner/gartner-top-strategic-predictions-for-2018-and-beyond/`

Giardino, C., Bajwa, S. S., Wang, X., Abrahamsson, P., 2015. Key challenges in early-stage software startups. In: International Conference on Agile Software Development. Springer, pp. 52–63.

Giardino, C., Paternoster, N., Unterkalmsteiner, M., Gorschek, T., Abrahamsson, P., 2016. Software development in startup companies: The greenfield startup model. IEEE Transactions on Software Engineering 42 (6), 585–604.
URL `http://dx.doi.org/10.1109/TSE.2015.2509970`

Giardino, C., Unterkalmsteiner, M., Paternoster, N., Gorschek, T., Abrahamsson, P., 2014a. What do we know about software development in startups? IEEE Software 31 (5), 28–32.
URL `http://dx.doi.org/10.1109/MS.2014.129`

Giardino, C., Wang, X., Abrahamsson, P., 2014b. Why early-stage software startups fail: a behavioral framework. In: International Conference of Software Business. Springer, pp. 27–41.

Gittleson, K., 2012. Can a company live forever? Access date: 2017-11-07.
URL `http://www.bbc.com/news/business-16611040`

Grimaldi, R., Grandi, A., 2005. Business incubators and new venture creation: an assessment of incubating models. Technovation 25 (2), 111–121.

Jacobson, I., Spence, I., Ng, P.-W., 2017. Is there a single method for the internet of things? ACM Queue 15 (3), 20.

Japan, I., 2012. Embedded System development Process Reference guide. Information-technology Promotion Agency, Japan.
URL `http://www.ipa.go.jp/english/sec/`

Kaisti, M., Rantala, V., Mujunen, T., Hyrynsalmi, S., Könnölä, K., Mäkilä, T., Lehtonen, T., 2013. Agile methods for embedded systems development-a literature review and a mapping study. EURASIP Journal on Embedded Systems 2013 (1), 15.

Kajko-Mattsson, M., Nikitina, N., 2008. From knowing nothing to knowing a little: Experiences gained from process improvement in a start-up company. In: International Conference on Computer Science and Software Engineering, CSSE 2008, December 12, 2008 - December 14, 2008. Vol. 2 of Proceedings - International Conference on Computer Science and Software Engineering, CSSE 2008. IEEE Computer Society, pp. 617–621.
URL `http://dx.doi.org/10.1109/CSSE.2008.1370`

Kakati, M., 2003. Success criteria in high-tech new ventures. Technovation 23 (5), 447–457.

Karlsson, L., Dahlstedt, A., och Dag, J. N., Regnell, B., Persson, A., 2002. Challenges in market-driven requirements engineering-an industrial interview study. In: Eighth International Workshop on Requirements Engineering: Foundation for Software Quality.

Keil, M., Carmel, E., 1995. Customer-developer links in software development. Communications of the ACM 38 (5), 33–44.

Kickstarter, 2018. Kickstarter stats. Access date: 2018-03-01.
URL `https://www.kickstarter.com/help/stats`

Kitchenham, B., 2004. Procedures for performing systematic reviews. Keele, UK, Keele University 33 (2004), 1–26.

Klotins, E., Unterkalmsteiner, M., Gorschek, T., 2015. Software Engineering Knowledge Areas in Startup Companies: A Mapping Study. Vol. 210 of Lecture Notes in Business Information Processing. pp. 245–257.
URL `<GotoISI>://WOS:000365180900024`

Kuhrmann, M., Münch, J., Richardson, I., Rausch, A., Zhang, H., 2016. Managing Software Process Evolution: Traditional, Agile and Beyond–How to Handle Process Change. Springer.

Langley, A., 1999. Strategies for theorizing from process data. Academy of Management review 24 (4), 691–710.

Laporte, C. Y., O'Connor, R. V., 2016. Implementing process improvement in very small enterprises with iso/iec 29110: A multiple case study analysis. In: Quality of Information and Communications Technology (QUATIC), 2016 10th International Conference on the. IEEE, pp. 125–130.

Laporte, C. Y., O'Connor, R. V., Ieee, 2014. Systems and software engineering standards for very small entities: Implementation and initial results. 2014 9th International Conference on the Quality of Information and Communications Technology (QUATIC), 38–47.
URL `<GotoISI>://WOS:000364237700005`

Laporte, C. Y., O'Connor, R. V., Paucar, L. H. G., 2015. Software engineering standards and guides for very small entities: Implementation in two start-ups. pp. 5–15.

URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-84933558276&partnerID=40&md5=02b5f237bb268c0133caa7c6cb17a44a

Larson, M. L., 1991. Translation: theory and practice, tension and interdependence. John Benjamins Publishing.

Lucero, S., et al., 2016. Iot platforms: enabling the internet of things. IHS Technology white paper.

Marks, G., O'Connor, R. V., Clarke, P. M., 2017. The impact of situational context on the software development process – a case study of a highly innovative start-up organization. Vol. 770. pp. 455–466.

Marmer, M., Herrmann, B. L., Dogrultan, E., Berman, R., Eesley, C., Blank, S., 2011. Startup genome report extra: Premature scaling. Startup Genome 10.

McConnell, S., 2007. Technical debt-10x software development — construx.

Moore, G. A., 2002. Crossing the chasm.

Mullins John, W., 2003. The new business road test.

Nguven-Duc, A., Dahle, Y., Steinert, M., Abrahamsson, P., 2017. Towards understanding startup product development as effectual entrepreneurial behaviors. In: International Conference on Product-Focused Software Process Improvement. Springer, pp. 265–279.

Nguyen-Duc, A., Abrahamsson, P., 2016. Minimum viable product or multiple facet product? the role of mvp in software startups. In: International Conference on Agile Software Development. Springer, pp. 118–130.

Nguyen-Duc, A., Abrahamsson, P., 2017. Exploring the outsourcing relationship in software startups: A multiple case study. In: Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering. ACM, pp. 134–143.

Nguyen-Duc, A., Cruzes, D. S., Conradi, R., 2015a. The impact of global dispersion on coordination, team performance and software quality–a systematic literature review. Information and Software Technology 57, 277–294.

Nguyen-Duc, A., Jabangwe, R., Paul, P., Abrahamsson, P., 2017a. Security challenges in iot development: a software engineering perspective. In: Proceedings of the XP2017 Scientific Workshops. ACM, p. 11.

Nguyen-Duc, A., Khan, K., Lønnestad, T., Bajwa, S. S., Wang, X., 2018. Product development in hardware startups - a software engineering perspective.

Nguyen-Duc, A., Seppanen, P., Abrahamsson, P., 2015b. Hunter-gatherer cycle: A conceptual model of the evolution of software startups. In: International Conference on Software and Systems Process, ICSSP 2015, August 24, 2015 - August 26, 2015. Vol. 24-26-August-2015 of ACM International Conference Proceeding Series. Association for Computing Machinery, pp. 199–203.
URL http://dx.doi.org/10.1145/2785592.2795368

Nguyen-Duc, A., Shah, S. M. A., Ambrahamsson, P., 2016. Towards an early stage software startups evolution model. In: Software Engineering and Advanced Applications (SEAA), 2016 42th Euromicro Conference on. IEEE, pp. 120–127.

Nguyen-Duc, A., Wang, X., Abrahamsson, P., 2017b. What influences the speed of prototyping? an empirical investigation of twenty software startups. In: International Conference on Agile Software Development. Springer, pp. 20–36.

Oates, B. J., 2005. Researching information systems and computing. Sage.

Pantiuchina, J., Mondini, M., Khanna, D., Wang, X., Abrahamsson, P., 2017. Are software startups applying agile practices? the state of the practice from a large survey. In: 18th International Conference on Agile Software Development, XP 2017, May 22, 2017 - May 26, 2017. Vol. 283 of Lecture Notes in Business Information Processing. Springer Verlag, pp. 167–183.
URL `http://dx.doi.org/10.1007/978-3-319-57633-6_11`

Paternoster, N., Giardino, C., Unterkalmsteiner, M., Gorschek, T., Abrahamsson, P., 2014. Software development in startup companies: A systematic mapping study. Information and Software Technology 56 (10), 1200–18.
URL `http://dx.doi.org/10.1016/j.infsof.2014.04.014`

Pervan, G., Maimbo, M., 2005. Designing a case study protocol for application in is research. In: Proceedings of the Ninth Pacific Asia Conference on Information Systems. PACIS, pp. 1281–1292.

Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M., 2008. Systematic mapping studies in software engineering. In: EASE. Vol. 8. pp. 68–77.

Pompermaier, L., Chanin, R., Sales, A., Fraga, K., Prikladnicki, R., 2017. An empirical study on software engineering and software startups: Findings from cases in an innovation ecosystem. In: 29th International Conference on Software Engineering and Knowledge Engineering, SEKE 2017, July 5, 2017 - July 7, 2017. Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE. Knowledge Systems Institute Graduate School, pp. 48–51.
URL `http://dx.doi.org/10.18293/SEKE2017-115`

Rafiq, U., Bajwa, S. S., Xiaofeng, W., Lunesu, I., 2017. Requirements elicitation techniques applied in software startups. In: 2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 30 Aug.-1 Sept. 2017. 2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA). IEEE Computer Society, pp. 141–4.
URL `http://dx.doi.org/10.1109/SEAA.2017.73`

Reynolds, P. D., White, S. B., 1997. The entrepreneurial process: Economic growth, men, women, and minorities. Praeger Pub Text.

Ries, E., 2011. The lean startup: How today's entrepreneurs use contstant innovation to create radically successful businesses. Crown Books.

Ronkainen, J., Abrahamsson, P., 2003. Software development under stringent hardware constraints: Do agile methods have a chance? In: International Conference on Extreme Programming and Agile Processes in Software Engineering. Springer, pp. 73–79.

Ronkainen, J., Taramaa, J., Savuoja, A., 2002. Characteristics of process improvement of hardware-related sw. In: International Conference on Product Focused Software Process Improvement. Springer, pp. 247–257.

Runeson, P., Höst, M., 2009. Guidelines for conducting and reporting case study research in software engineering. Empirical software engineering 14 (2), 131.

Saldaña, J., 2015. The coding manual for qualitative researchers. Sage.

Sarasvathy, S. D., 2001. Causation and effectuation: Toward a theoretical shift from economic inevitability to entrepreneurial contingency. Academy of management Review 26 (2), 243–263.

Seaman, C. B., 1999. Qualitative methods in empirical studies of software engineering. IEEE Transactions on software engineering 25 (4), 557–572.

Shaw, M., 2003. Writing good software engineering research papers. In: Software Engineering, 2003. Proceedings. 25th International Conference on. IEEE, pp. 726–736.

Shull, F., Singer, J., Sjøberg, D. I., 2007. Guide to advanced empirical software engineering. Springer.

Singer, J., Vinson, N. G., 2002. Ethical issues in empirical studies of software engineering. IEEE Transactions on Software Engineering 28 (12), 1171–1180.

Souza, R., Malta, K., Almeida, E. S. D., 2017. Software engineering in startups: A single embedded case study. In: 1st IEEE/ACM International Workshop on Software Engineering for Startups, SoftStart 2017, May 21, 2017. Proceedings - 2017 IEEE/ACM 1st International Workshop on Software Engineering for Startups, SoftStart 2017. Institute of Electrical and Electronics Engineers Inc., pp. 17–23.
URL http://dx.doi.org/10.1109/SoftStart.2017.2

Srinivasan, S., Barchas, I., Gorenberg, M., Simoudis, E., 2014. Venture capital: Fueling the innovation economy. Computer 47 (8), 40–47.

Stock, T., Seliger, G., 2016. Methodology for the development of hardware startups. Advanced Materials Research 1140.

Strauss, A., Corbin, J., 1998. Basics of qualitative research: Procedures and techniques for developing grounded theory.

Sutton Jr, S. M., 2000. Role of process in a software start-up. IEEE Software 17 (4), 33–39.
URL http://dx.doi.org/10.1109/52.854066

Swanson, R. A., Chermack, T. J., 2013. Theory building in applied disciplines. Berrett-Koehler Publishers.

Sánchez-Gordón, M.-L., O'Connor, R. V., 2016. Understanding the gap between software process practices and actual practice in very small companies. Software Quality Journal 24 (3), 549–570.

Tanabian, M., ZahirAzami, B., 2005. Building high-performance team through effective job design for an early stage software start-up. In: Engineering Management Conference, 2005. Proceedings. 2005 IEEE International. Vol. 2. IEEE, pp. 789–792.

Temple, B., Young, A., 2004. Qualitative research and translation dilemmas. Qualitative research 4 (2), 161–178.

Terho, H., Suonsyrja, S., Systa, K., 2016. The developers dilemma: Perfect product development or fast business validation? In: 17th International Conference on Product-Focused Software Process Improvement, PROFES 2016, November 24, 2016 - November 26, 2016. Vol. 10027 LNCS of Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer Verlag, pp. 571–579.
URL http://dx.doi.org/10.1007/978-3-319-49094-6_42

Tripathi, N., Annanpera, E., Oivo, M., Liukkunen, K., 2016. Exploring Processes in Small Software Companies: A Systematic Review. Vol. 609 of Communications in Computer and Information Science. pp. 150–165.
URL <GotoISI>://WOS:000382651100012

Unterkalmsteiner, M., Abrahamsson, P., Wang, X. F., Anh, N. D., Shah, S., Bajwa, S. S., Baltes, G. H., Conboy, K., Cullina, E., Dennehy, D., Edison, H., Fernandez-Sanchez, C., Garbajosa, J., Gorschek, T., Klotins, E., Hokkanen, L., Kon, F., Lunesu, I., Marchesi, M., Morgan, L., Oivo, M., Selig, C., Seppanen, P., Sweetman, R., Tyrvainen, P., Ungerer, C., Yague, A., 2016. Software startups - a research agenda. E-Informatica Software Engineering Journal 10 (1), 89–123.
URL <GotoISI>://WOS:000387014900006

U.S., S. B. A., 2017. Frequently asked questions about small businessAccess date: 2018-02-27.
URL https://www.sba.gov/sites/default/files/advocacy/SB-FAQ-2017-WEB.pdf

Wasserman, A. I., 2016. Low ceremony processes for short lifecycle projects. In: Managing Software Process Evolution. Springer, pp. 1–13.

Wei, J., 2017. State of the hardware incubators and accelerators in the united states [society news]. Ieee Consumer Electronics Magazine 6 (1), 22–23.

Wohlin, C., 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: Proceedings of the 18th international conference on evaluation and assessment in software engineering. ACM, p. 38.

Wohlin, C., Höst, M., Henningsson, K., 2003. Empirical research methods in software engineering. In: Empirical methods and studies in software engineering. Springer, pp. 7–23.

Womack, J. P., Jones, D. T., Roos, D., 1990. Machine that changed the world. Simon and Schuster.

Yau, A., Murphy, C., 2013. Is a rigorous agile methodology the best development strategy for small scale tech startups?

Yli-Huumo, J., Rissanen, T., Maglyas, A., Smolander, K., Sainio, L.-M., 2015. The relationship between business model experimentation and technical debt. In: 6th International Conference on Software Business, ICSOB 2015, June 10, 2015 - June 12, 2015. Vol. 210 of Lecture Notes in Business Information Processing. Springer Verlag, pp. 17–29.
URL `http://dx.doi.org/10.1007/978-3-319-19593-3_2`

Zhou, X., Jin, Y., Zhang, H., Li, S., Huang, X., 2016. A map of threats to validity of systematic literature reviews in software engineering. In: Software Engineering Conference (APSEC), 2016 23rd Asia-Pacific. IEEE, pp. 153–160.

Appendix A

A.1 Interview Protocol

A.1.1 General Information

A.1.1.1 Practicalities

1. Interview Id:
2. When:
3. Where:
4. Duration:
5. Interviewers:

A.1.1.2 Interviewee

1. Interviewee Id:
2. Company Id:
3. Position in company:
4. Years at company:
5. Educational background:

A.1.2 Business Background

1. Describe the technical competence of the team i.e. how is your team organized?
2. Please name the three largest challenges encountered during the startup phase.

A.1.3 Startup Development Methodologies

1. How do you make sure that you are building the right product?
2. How do you manage customer feedback?

A.1.4 Product Development

A.1.4.1 Engineering Practices

1. Do you implement agile practices, i.e. refactoring, test-first, sprint planning, frequent releases, and/or daily stand up meetings?
2. How do external dependencies/factors influence product development e.g. speed?
3. How do you balance hardware and software development?
4. Have you outsourced development i.e. prototyping, conceptualization, design, marketing, testing?
5. Have any technical/business decisions slowed you down or affected the overall quality of the product?

A.1.4.2 Requirements

1. How do you elicit requirements i.e. prototyping, mock-ups, interviews, analysis of similar products, brainstorming?

A.1.4.3 Software Structure and Architecture

1. How do you manage documentation?
2. How do you handle quality, i.e. technical debt, security, and privacy?
3. How often do you refactor the code? How much rework has been done after the prototyping?
4. To what extent do you reuse components of earlier prototypes?
5. How has tacit knowledge affected you?

A.1.4.4 Testing

1. How do you perform hardware and software testing?
2. When do you start writing tests?

A.2 Pre-Interview Questionnaire

The following list contains the pre-interview questions all participants had to fill out before the actual interviews.

1. Briefly describe your product.
2. Briefly explain your role and responsibilities in the company.
3. Briefly describe your company i.e. history and current headcount.
4. Have you received any funding?

A.3 Consent Form

Inquiry about participation in research project

"Software Engineering in Startups Delivering Both Hardware and Software Parts"

Purpose of the study
Empirical analysis, through a multiple case-study, of product development methods in startup companies who deliver products with both hardware and software parts. This is part of a Master Thesis at the Department of Computer Science, NTNU, Trondheim, under the supervision of Professor Letizia Jaccheri. The purpose of the study is to create a better understanding of decision-making and problem-solving in technology startups who deliver both hardware and software parts.

The research questions that will be analyzed are:

RQ1 How do hardware startups achieve agility during product development?

RQ2 How do hardware startup manage quality concerns of their product?

RQ3 How do hardware startups achieve balance between speed and quality?

Selection
Candidates are eligible for participation if they meet the following criteria:

- They have experience and/or knowledge about software or hardware development.
- They work or have worked in a technology startup who have made an initial prototype of their product, or delivered a product with hardware and software parts to paying customers.

If the candidate meets the criteria, he/she is regarded as qualified for contributing to answering the research questions.

What does participation in the study involve?
Participation in the study involves being an interviewee in one or more interviews. Interviews will last for a maximum of 60 minutes. If more data is needed, it is desirable to extend the interview time, or perform additional interviews.

The interviews will be recorded, and then transcribed. This is part of a thematic analysis that will be conducted by the researchers. Participants do not need to provide sensitive information like name or person-number, as this is irrelevant for answering the research questions. Sensitive person-data will not be gathered from other sources like journals or registers. The questions will mainly deal with the interviewee's role in the company, the company's evolvement from inception till today, work methods and cooperation, and characteristics of the startup environment. All participants are allowed to read the thesis before it is published.

What happens with the information about you?
All sensitive person information will be stored confidentially. The project group and the supervisor are the only persons with access to this information. To ensure confidentiality, recordings will be stored on the students' local computer in a separate folder.

The interviewees' role in the company is relevant for the study, and will therefore be described in the thesis. If this information makes it possible to recognize the person, we will perform the necessary measures to ensure complete confidentiality. Company name will not be used in the thesis, however descriptions of how the company operate and what product they deliver will be necessary to build generalizable conclusions.

The final date of the project is 10.06.18. All data material will be made anonymous at this date.

Voluntary participation
It is voluntary to participate in the study. Participants can withdraw at any given time, and all information about the person will then be made anonymous.

If you want to participate in the study or have further questions, please contact Jorgen Birkeland (+47 90676597) or Vebjørn Berg (+47 48268999).

Consent to participation in the study

I have received information about the study, and I am willing to participate

____

(Signed by the participant, date)

Appendix B

Software Startup Engineering: A Systematic Mapping Study

Vebjørn Berg, Jørgen Birkeland, Anh Nguyen-Duc, Ilias Pappas, Letizia Jaccheri*

IT-bygget, Sem Sælands vei 9, 7034 Trondheim, Norway

Abstract

[Context] Software startups have long been a significant driver in economic growth and innovation. The on-going failure of the major number of startups calls for a better understanding of state-of-the-practice of startup activities. [Objective] With a focus on engineering perspective, this study aims at identifying the change in focus of research area and thematic concepts operating startup research. [Method] A systemic mapping study on 74 primary papers (in which 27 papers are newly selected) from 1994 to 2017 was conducted with a comparison with findings from previous mapping studies. A classification schema was developed, and the primary studies were ranked according to their rigour. [Results] We discovered that most research has been conducted within the SWEBOK knowledge areas software engineering process, management, construction, design, and requirements, with the shift of focus towards process and management areas. We also provide an alternative classification for future startup research. We find that the rigour of the primary papers was assessed to be higher between 2013-2017 than that of 1994-2013. We also find an inconsistency of characterizing startups. [Conclusions] Future work can focus on certain research themes, such as startup evolution models and human aspects, and consolidate the thematic concepts describing software startups.

Keywords: Software development, Systematic mapping study, Startup, Software startup, Software engineering

1. Introduction

Technology-based startups have long been an important driver for global economic growth and competitiveness [1]. Software startups, newly created companies producing cutting-edge software technology, have shown to be an important source of software innovation. Despite stories of successful startups,

*Corresponding author. Tel: +4748268999; Tel: +4790676597.
Email address: vebjorbe@stud.ntnu.no, jorgebi@stud.ntnu.no (Vebjørn Berg, Jørgen Birkeland, Anh Nguyen-Duc, Ilias Pappas, Letizia Jaccheri)

90 percent of them fail, primarily due to self-destruction rather than competition [2, 3]. The failures come from financial and market factors, for example, insufficient funding to operate startups activities, failure in finding product-market fit, and building an entrepreneurial team [4]. However, there are also identified unique challenges related to software development and innovation methods [4]. Software startup engineering can be defined as "the use of scientific, engineering, managerial, and systematic approaches with the aim of successfully developing software systems in startup companies" [5]. Startup researchers have called for a further attention to engineering approaches to support startup activities in all startup evolution stages [1]. Previously, most of the research in the field of software engineering has been conducted in relation to the needs and challenges of established companies, first identified by Sutton [6].

Startups are at the forefront of applying new technologies in practice. From an engineering perspective, developing technology products is challenging as the startup context presents a dynamic and fast-changed environment, making it difficult to adopt prescriptive engineering practices [5]. Despite the rapid growth of the population of startups, the research on software engineering in startups is still at an early stage [1].

One of the most extensive literature reviews in the field is the systematic mapping study of Paternoster et al. [7], reviewing a total of 43 primary studies from 1994 until 2013. This review shows a lack of high quality studies in the field. While a large amount of Software Engineering practices were extracted from startups, the practices were chosen randomly and adopted under the constraints imposed by the startup context. Thus, an updated systematic mapping is required as it will identify the current status in the area and pave the way for more empirical studies examining startups.

Since 2015, we observed an increased focus on software startup research (i.e., the organization of three International software startup workshops (ISSW) in 2016 and 2017, and software startups tracks at PROFES 2017 and XP 2017 conferences). The previous systematic review has rapidly gained a large amount of citations [7]. While this implies the further growth in software startup research, a revisit on the area can identify how engineering activities in software startups have changed over time. The objective of this mapping study is to provide an updated view on software startup research in order to identify research gaps. Different from the previous mapping studies [7, 8], we aim at synthesizing startup descriptions in research and its associated software engineering knowledge areas. Beside market factors and financial factors, knowledge about engineering factors and how they affect the startup initiatives and development would be helpful for entrepreneurs in understanding their startups' challenges.

We assume that startups perform various types of software engineering activities, as described in SWEBOK [9]. We would like to observe how software startup research has evolved and possibly matured in some Software Engineering knowledge areas. SWEBOK is previously used in Klotins et al. [8], which allow for easy comparisons and make it possible to identify changes in terms of research direction for the last five years.

The research objective leads to the following research questions:

1. RQ1: How has software startup research changed over time in terms of focused knowledge areas?
2. RQ2: What is the relative strength of the empirical evidence reported?
3. RQ3: In what context has software startup research been conducted?

In this article, we present results from systematic mapping studies of software startup research from 1994 to 2017. To do so, we expand previous literature [7, 8] with the focus on papers published from 2013-2017. We found 27 relevant articles during the last five years. The results were merged and compared to the previous mapping studies. To address RQ1, the papers were structured according to the knowledge areas identified in SWEBOK [9]. With RQ2, we evaluated the papers' rigour to compare the quality of papers published before and after 2013. Finally, with RQ3, we examined to what extent the retrieved papers provided sufficient startup descriptions, and if there were similarities in the use of terms describing the startup context between the papers. Our meta-analysis on Software Engineering knowledge area and startup case context reveals important areas for investigation. We also come up with a classification of future research on software startups.

The contribution of this mapping study is two-fold. Firstly, the study provides a comprehensive view of software startup for Software Engineering researchers. Possible research gap is derived for future study. Secondly, the study provides a map of the contextual setting of investigated startups. Contextual map infers the applicability area of empirical findings from the startups. This would help to compare and to generalize future research in software startups.

The paper proceeds as follows: Section 2 introduces the background of the study and the related mapping studies. Section 3 presents the research method undertaken and threats to the validity of the mapping study. Section 4 reports the results and visualizes both our findings and the findings of the previous mapping studies. Section 5 discusses the results in relation to the research questions. Section 6 concludes the paper by answering the research questions and presents implications and future work.

2. Background

2.1. Software startups

A startup can be defined as "an organization that is challenged by youth and immaturity, with extremely limited resources, multiple influences, and dynamic technologies and markets" [6]. More specifically, Coleman and O'Connor [10] describe software startups as "unique companies that develop software through various processes and without a prescriptive methodology". Others have characterized software startups as modern organizations with little or no operating history, aiming at developing high-tech and innovative products, and rapidly scale their business in extremely dynamic markets [11].

Software startups develop innovative software products in environments of time-pressure and a lack of resources, constantly searching for sustainable and scalable business models. This is in contrast to established companies, that have more resources and already command a mature market [1]. While established companies focus on optimizing an existing business model, startups focus on finding one, which requires experimentation of various products in different markets [12]. Instead of developing software for a specific client, software startups develop systems which have market-driven requirements, meaning they have no specific customers before their product is released [13, 14].

There exist many processes to manage product development (i.e., processes concerned with *how* to develop a product), like agile and waterfall methods. (e.g., agile and waterfall). However, these processes do not focus on addressing *what* product to develop, which is essential in the startup context where both problems and solutions tend to be poorly understood [15]. The high failure rates of software startups are often caused by a lack of customers rather than product development issues [2, 13].

2.2. Startup Development Methodology

Software startups generally develop products in high-potential target markets [16], without necessarily knowing *what* the customers want [14]. This relates to market-driven software development, which emphasizes the importance of specific requirement elicitation techniques (e.g., prototyping), and time-to-market as key strategic objectives [14, 17]. In a market-driven context, requirements tend to be (1) invented by the software company, (2) rarely documented [18], and (3) validated only after the product is released in the market [14, 19, 20, 21]. As to this, products that don't meet customer needs are common, resulting in failure of new product releases [22]. Entrepreneurial and customer focused development approaches like [23, 24, 25, 26] have received attention from the research community. The customer development process introduced by Blank [23] can be divided into four phases: (1) customer discovery, (2) customer validation, (3) customer creation, and (4) company building. A frequently applied entrepreneurship theory among entrepreneurs is Lean Startup, which builds on the principles from Blank. The method has been criticized by researchers for being based on personal experience and opinions rather than empirical evidence, however, concepts from the Lean Startup have attracted considerable attention among practitioners [15, 16].

2.2.1. Lean Startup

Ries [12] presented the Lean Startup method in 2008, based on lean principles first introduced by Toyota [27]. The method aims at creating and managing startups, to deliver products or services to customers as fast as possible. The method provides principles for how to run a new business, where the goal is to grow the business with maximum acceleration. By iteratively turning ideas into products, measure customers' satisfiability, and learn from their feedback, startups can accelerate their business. This process is referred to as the build-measure-learn (BML) feedback loop, which is an iterative process, where the goal is to minimize the total time through the loop.

Key to the BML feedback loop is to do continuous experimentation on customers to test hypotheses. The hypotheses can be tested by building a minimum viable product (MVP), which is the simplest form of an idea, product, or service that can answer the hypotheses. Any feature, process, or effort not directly contributing to answering the hypotheses, is removed. The aim is to eliminate any waste throughout the process. Empirical research has found three main types of MVP usage, including (1) MVP as a design artifact, (2) MVP as a boundary-spanning object, and (3) MVP as a reusable artifact [28]. MVPs can be used to bridge knowledge gaps within organizations or to provide a mutual understanding between customer input and product design.

When the MVP has been built and the hypotheses tested, the next step is to measure the customer feedback and learn from it. This is referred to as validated learning, which is about learning which efforts are value-creating and eliminate the efforts that aren't necessary for learning customer needs. The final step of the loop is whether to pivot or persevere. A pivot is a structured course correction designed to test a new fundamental hypothesis about the product, strategy, and engine of growth [12]. Bajwa et al. [29] identified 10 pivot types and 14 triggering factors, concluding that trying to solve the wrong problem for the customer is the most common reason for pivoting (i.e., customer need pivot). If a pivot isn't required, meaning the MVP was found to be fit to market, the startup perseveres. The BML feedback loop then continues, where new hypotheses are tested and measured.

The Lean Startup method is beneficial for business development and understanding *what* product to develop, emphasizing the importance of getting the product to customers as soon as possible. Startups tend to prefer time and cost over product quality [30], neglecting traditional process activities like formal project management, documentation, and testing [5]. Shortcuts taken in product quality, design, or infrastructure can eventually inhibit learning and customer satisfaction [12]. Software startups need their own development practices to manage the challenges posed by customer development methods such as Lean Startup.

2.3. Software Engineering in Startups

Software startup engineering can be defined as "the use of scientific, engineering, managerial, and systematic approaches with the aim of successfully developing software systems in startup companies" [5]. The degree of process in software development is dependent on system complexity, business risk, and the number of people involved [31]. The impact of the inadequate use of software engineering practices might be a significant factor leading to the high failure rates [8]. As time and resources are extremely scarce in environments of high market and technology uncertainty, software startups need effective practices to face those unique challenges [11]. The need for process depends on the lifecycle stage of the company, divided into four stages [3].

- Stage 1: The startup stage is defined as the time from idea conceptualization to the first sale. A small executive team with necessary skills is required in order to build the product.

- Stage 2: The stabilization phase lasts until the product is stable enough to be commissioned to a new customer without causing any overhead on product development.
- Stage 3: The growth phase begins with a stable product development process and lasts until market size, share, and growth rate have been established.
- Stage 4: The last stage is when the startup has evolved into a mature organization. The product development is robust and easy to predict, with proven processes for new product inventions.

Startups are creative and flexible by nature, and so strict release processes are often overshadowed by quick, inexpensive product releases, with focus on customer acquisition [31]. This can often result in ineffective software engineering practices [6]. Since startups have limited resources, the focus is often directed towards product development, rather than focusing on the establishment of rigid processes [10].

It is important to notice that in terms of communication and cooperation dynamics, startups and established companies have different software engineering experiences and needs [30]. While established companies have well-defined processes for their business, startups usually have low-ceremony processes [32], which means that the amount of management overhead is low. Instead of utilizing repeatable and controlled processes, startups take advantage of reactive and low-precision engineering practices with a focus on the productivity and freedom of their teams [33, 34, 35].

Reactive, low-ceremony processes are powerful in the early stages of software development since speed and learning are important [12]. However, as startups enter new lifecycle stages, an increased usage of processes for addressing key customer needs, delivering functional code early and often, and providing a good user experience is required [32]. New business issues like hiring, sales, and funding appear, and more users and complex code require an extended focus on robustness, scalability, performance, and power consumption [31]. The use of methods like the Lean Startup is one of the reasons why software startups need and sometimes apply their own software engineering practices, which pose challenges when it comes to software engineering. Lean Startup is beneficial for business and product development, but when it comes to software development, a more hybrid approach of agile and lean may provide the most benefits in terms of cost, time, quality, and scope [30].

2.4. Existing literature reviews

Since the gap in research specific to software engineering in startups first was identified [6], there have only been undertaken two mapping studies entirely dedicated to the research area [7, 8]. There also exist relevant work related to SMEs (small and medium-sized enterprises) [36], and VSEs (very small entities), which become more relevant as startups enter more mature lifecycle stages, however, the early stages of startups pose some specific challenges and needs (e.g., little working/operating history).

The first systematic mapping by Paternoster et al. [7] covered studies up to December 2013, aiming at structuring and analyzing state-of-the-art on software startup research. The conclusion of the paper is that there existed few high-quality studies contributing to the body of knowledge and that there is a need for more studies supporting startups for all lifecycle stages. From a total of 43 primary studies, only 4 papers [10, 37, 38, 39] were considered as strong contributions and entirely dedicated to software engineering activities in startups. The results showed that startups choose their software engineering practices opportunistically, and adapt them to their own context.

Klotins et al. [8] conducted a mapping study published in 2015, classifying 14 primary studies on software startup engineering into 11 of 15 SWEBOK knowledge areas. The paper concludes as Paternoster et al. [7], that research did not provide support for any challenges or engineering practices in startups, and that available research results were hard to transfer between startups due to low rigour. This was explained by the lack of contextual information in the studies, and how the studies were performed.

3. Research Methodology

A systematic mapping study was undertaken to provide an overview of the research available in the field of software engineering specific to startups, following guidelines from Kitchenham [40] and several steps of the standardized process for systematic mapping studies, as illustrated in figure 1 [41].

Systematic mapping studies can be used in research areas with few relevant primary studies of high quality, as they provide a coarse-grained overview of the publications within the topic area [41]. This systematic mapping study covers 74 primary papers, extending the two previous mapping studies [7, 8]. As these studies only cover three papers from 2013, the search strategy of this systematic mapping study included papers from 2013 up to October 2017. This approach allowed for merging and comparing the primary literature within the research field for the period 1994-2017.

The main steps of our process are illustrated in figure 1, and include the search and study selection strategies, manual search, data extraction, quality assessment, and the data synthesis method. The process led to a total number of 27 new primary papers found in table A.7.

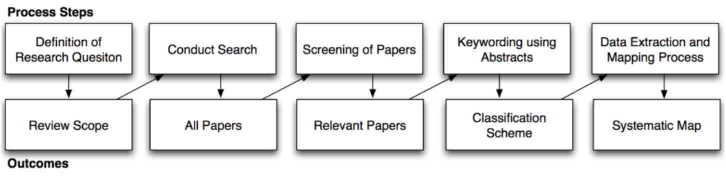


Figure 1: The Systematic Mapping Study Process [41]

3.1. Mapping Procedure

Step 1: Pilot search. Pilot searches were performed in online databases to find an optimal search string and the most suitable databases. The searches helped to define the criteria for inclusion and quality assessment and the classification schema.

Step 2: Search strategy and study selection. Based on the search string, a total number of 1012 unduplicated papers were retrieved. This was further limited to 74 (titles), 28 (abstracts), and finally, 20 papers after a collaborate effort from the first and second author was conducted. The full-text of the remaining 20 papers was read.

Step 3: Additional manual search. A manual search was performed to find more relevant papers. The publication lists of relevant authors were scanned, and the forward snowballing technique was used [42]. For the forward snowballing, Google Scholar was used to examining the citations of the papers retrieved. This resulted in seven more relevant papers. These were either not published in the databases, or were overlooked in step 2.

Step 4: Quality assessment. To identify the rigour of the remaining papers, a quality assessment was performed on the papers that provided empirical evidence. The complete assessment can be found in table B.10.

Step 5: Data extraction and synthesis. From the primary papers, relevant data and information were extracted into a classification schema. A multi-step synthesis was performed to answer the research questions.

3.2. Data Sources and Search Strategy

The systematic search strategy consisted of searches in three online bibliographic databases. The databases were selected from their ability to handle complex search strings, their general use in similar literature reviews in the software engineering community [7, 36], and the fact that they index the research articles from other databases. In addition, to ensure the best possible coverage of the literature, we performed complementary searches and forward snowballing (section 3.4 Manual Search). Following guidelines from Wohlin [42], systematic literature studies should use a combination of approaches for identifying relevant literature, where forward snowballing is found particularly useful. Forward snowballing can reduce systematic errors related to the selection of databases and construction of the search string [42]. To obtain high-quality data, the following databases were used:

Database	Papers
Scopus	451
ISI Web of Science	121
Engineering Village Compendex	875
Total	1447

Table 1: The searched databases and number of retrievals

Initial searches in the databases were conducted to identify keywords related to software engineering and startups, targeting title, abstract, and keywords. The most frequently used keywords for "startup" were chosen and combined in the search string [7]. The final search string consisted of several search terms combined using the Boolean operator *"OR"*:

> "*(startups OR start-up OR startup) AND software engineering OR (startups OR start-up OR startup) AND software development OR (startups OR start-up OR startup) AND software AND agile OR (startups OR start-up OR startup) AND software process OR (startups OR start-up OR startup) AND software tools*".

3.3. Study Selection

The study selection process is illustrated in figure 2, along with the number of papers at each stage. Searching the databases Scopus, ISI Web of Science, and Engineering Village using the search string returned 1447 papers, resulting in 1012 unduplicated studies. The searches targeted the document types: book chapters, journal article, conference article, conference proceedings, dissertation, and report chapters.

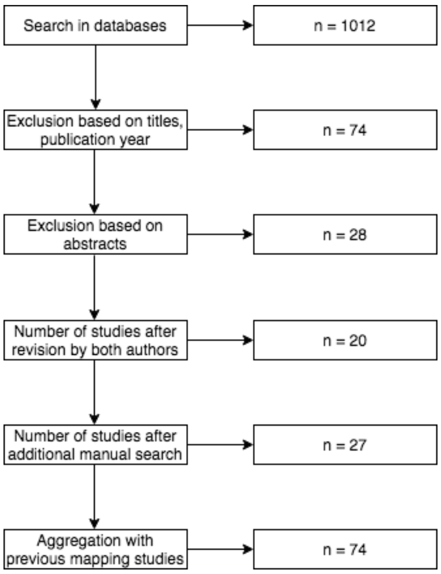

Search in databases
n = 1012
Exclusion based on titles, publication year
n = 74
Exclusion based on abstracts
n = 28
Number of studies after revision by both authors
n = 20
Number of studies after additional manual search
n = 27
Aggregation with previous mapping studies
n = 74

Figure 2: The Study Selection Process

Papers were relevant for inclusion in the study if they met the following criteria: (1) investigate concepts/problems/solutions of engineering in software startups, (2) present contributions in the form of lessons learned, framework, guidelines, theory, tool, model, or advice as applied in Paternoster et al. [7], (3) are not included in any of the previous mapping studies, and (4) studies are written in English. The papers that were selected are scientific peer-reviewed articles, which is independent of the role of authors. We did not find experience reports from entrepreneurs, which might make the sample of papers lean towards the researcher community. To decrease the number of papers into a manageable amount, workshops, and papers based on expert opinion were excluded from the review process.

As common for systematic mapping studies [41], this study focuses on synthesizing empirical research. Empirical studies are important for evidence-based software engineering research and practice, and for generating a knowledge base leading to accepted and well-formed theories [40, 43]. This study provides an overview of empirical research on software startup engineering to date, and how research has evolved and possibly matured over the time period.

The retrieved papers were examined by the first and second author, where each author separately reviewed the papers based on titles and abstracts. Disagreements were resolved by discussing the full text of the relevant papers. This was necessary as some of the abstracts were incomplete or poor. At this stage another 8 papers were excluded, making the total of newly selected papers 20, before performing the additional manual search.

3.4. Manual Search

A manual search was conducted with the participation of the third author, using the forward snowballing technique [42], to identify additional papers not discovered by the search string. Google Scholar was used to examine the citations to the paper being examined. The publication lists of frequently appearing authors were also searched. This resulted in several papers as candidates for inclusion. After assessing title, abstract, and finally the full text, 7 more papers were included as primary studies [17, 28, 29, 44, 45, 46, 47]. Among the papers, 21 were conference papers, five were journal papers, and one was a book chapter.

3.5. Quality Assessment

To build on previous work, a quality assessment of the new primary papers providing empirical evidence was performed. The total number of eligible papers was 22 (table A.7). Although systematic mapping studies usually don't evaluate the quality of each paper in such depth as systematic literature reviews, the quality assessment process was undertaken to assess how results were presented in the primary studies. No paper was excluded based on the quality assessment.

To assess the rigour, credibility, and relevance of the papers, we adopted the quality assessment scheme from Nguyen-Duc et al. [48]. Quality assessment has been identified as important for performing empirical research in software engineering [40, 49]. Table 2 illustrates 10 quality evaluation criteria. For each criterion the papers met, they got a score of 1, and otherwise 0. This means that the maximum score a paper could get was 10. A score of 0-3 was regarded as low rigour, 4-6 medium rigour, and 7-10 high rigour. The complete quality assessment can be found in table B.10.

Problem Statement
Q1. Is research objective sufficiently explained and well-motivated?

Research Design
Q2. Is the context of study clearly stated?
Q3. Is the research design prepared sufficiently?

Data collection
Q4. Are the data collection & measures adequately described?
Q5. Are the measures and constructs used in the study the most relevant for answering the research question?

Data analysis
Q6. Is the data analysis used in the study adequately described?
Q7a. Qualitative study: Are the interpretation of evidences clearly described?
Q7b. Quantitative study: Are the effect size reported with assessed statistical significance?
Q8. Are potential alternative explanations considered and discussed in the analysis?

Conclusion
Q9. Are the findings of study clearly stated and supported by the results?
Q10. Does the paper discuss limitations or validity?

Table 2: Quality Assessment Checklist [48]

3.6. Data Extraction and Synthesis

After the quality assessment, we defined the classification schema (table A.7). Data from each of the 27 newly selected primary studies were then systematically extracted into the classification schema, according to the predetermined attributes: (1) SWEBOK knowledge area, (2) Research method, (3) Contribution type, (4) Pertinence, (5) Term for "startup", (6) Incubator context, (7) Publisher. The chosen attributes were inspired by previous mapping studies [7, 8, 36], and from the process of finding keywords in the abstracts of the retrieved papers [41]. Organizing the findings into tabular form enabled for easy comparisons across studies and time periods. In addition to classifying the papers, each paper was scanned for thematic concepts to identify researchers' descriptions of investigated startups. The thematic concepts were adopted from the recurring themes found in Paternoster et al. [7]. This made it possible to assess the agreement in the community to the definition of startups.

The software engineering book of knowledge (SWEBOK) was created to provide a consistent view of software engineering, and to set the boundary of software engineering with respect to other disciplines [9]. SWEBOK contains 15 knowledge areas that characterize the practice of software engineering. The focal point of the paper is to propose research directions based on the knowledge areas following the work done by existing literature. Since the two major mapping studies in the area follow different approaches, that is SWEBOK [8] and focus facets [7], in the present study we focus on KAs, as this can allow the reader to better comprehend how the two different approaches are connected, thus offering a more holistic understanding of the current status in software startup engineering research. Assigning each paper into the specific knowledge areas was done by the first and second author. Two researchers read the titles, keywords, abstracts, and the body of each paper, before evaluating the papers' conformance with each specific knowledge area's description or subareas.

3.7. Threats to Validity

There are several threats to the validity of systematic mapping studies [50]. One threat is related to the data extraction from each paper, where results can be biased from researchers personal judgment. To mitigate this threat, and ensure correct classification of each paper into the SWEBOK knowledge areas, this process was performed jointly by the first and second author at one computer, resolving any conflicts and regulating individual bias.

Threats to the retrieval of relevant papers must also be considered. The inconsistent use of terms for “startup” made it difficult to cover all used terms in the search string. Hence, it appeared terms not considered when constructing the search string. Some of these were “founder teams”, “very small enterprise”, “very small entity”, and “very small company”, which all were used in relation to the startup context. Relevant papers might therefore have been overlooked.

The use of only three bibliographic databases might have affected the number of relevant papers retrieved. Compared to the number of databases used in similar studies, this seems to be at the low-end. The chosen databases are however among the most used ones in the field of software engineering, and the databases that contributed to the most retrieved papers in other studies [7]. The risk of missing papers published the last five years was mitigated by the use of forward snowballing which lead to the retrieval of seven more papers.

To make sure the study selection was not biased from personal opinions, paper selection involved the first, second, and third author of the paper, which allowed a collaborative resolution of conflicting views, following guidelines from Kitchenham [40]. We defined clear inclusion and exclusion criteria, and a quality assessment checklist to assess each paper's quality. Disagreement to quality assessment was discussed between author one and two until consensus was reached. This decreased the risk of miss-classifying any relevant papers.

For the quality assessment, we only used two points to collect answers, as the first and second author were unfamiliar with the research field. The papers got a score of 1 if they met the criteria, and otherwise 0. Prior studies have used a more fine-grained classification of quality criteria and even used different criteria in some occasions. It is more likely that the papers in our study obtained higher rigour than they would have received if another more fine-grained assessment method had been used.

4. Results

This section presents the extracted data of the primary studies. The section is divided into the three research questions to allow for better visualization and presentation of the most relevant findings. The final number of primary papers ended up being 74, adding the 27 new papers to the existing 44.

4.1. RQ1: How has software startup research changed over time in terms of focused knowledge areas?

This section is divided into two sub-sections. Section 4.1.1 presents the publication frequency of primary studies from 1994-2017. Section 4.1.2 presents the SWEBOK knowledge areas, contribution types, and empirical evidence between 1994-2017.

4.1.1. Publication Frequency

Figure 3 shows the number of studies published in relation to software startup engineering between 1994-2017, constituting a total of 74 published papers. We observe that the publication frequency of papers between 2013-2017 is higher than for any period before 2013. From 1994-2013, the highest number of primary papers within a single year was 7 (2008). In comparison, 2016 and 2017 constituted 9 and 11 papers respectively. The pertinence of the papers published between 2013-2017 was generally higher than what was found for the period 1994-2013. 85 percent of the papers from 2013-2017 had high pertinence, meaning that they were entirely dedicated to software engineering activities in startups. The remaining four papers were focusing on activities of small software companies, and so set to partial pertinence (table A.7). Although their focus was not entirely dedicated to startups, some of them [51] performed empirical studies on startups, referring to them as "very small entities" or "small software companies". In the period 1994-2013, 57 percent of the papers had high pertinence, while 20 percent had partial pertinence.

Klotins et al. [8] only includes 4 unique primary papers [52, 53, 54, 55], as the remaining 10 papers were included among the 43 primary papers in Paternoster et al. [7]. The 4 papers are from 1994, 2000, 2008, and 2013.

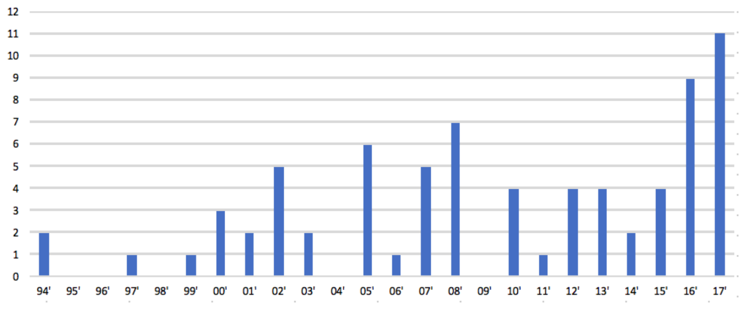


Figure 3: Publication Frequency, 1994-2017

4.1.2. Knowledge Areas

The 27 new primary studies were classified into the knowledge areas of SWEBOK [9]. The categories were developed by the software community as a baseline for the body of knowledge within software engineering. Figure 4 illustrates what knowledge areas that have received most attention the last five years, and to what extent empirical studies have been undertaken. The figure shows which research methods that have been used to address each of the knowledge areas. Only the papers providing empirical evidence (22 papers) were included in the figure, covering a total of 9 knowledge areas. Some of the papers covered one or more knowledge areas (e.g., Nguyen-Duc et al. [56]).

The assessed research methods followed guidelines from Oates [57], and include (1) survey, (2) design and creation, (3) experiment, (4) case study, (5) action research, and (6) ethnography (table A.8). Case study was the most frequently used empirical research method (81 percent), followed by experiments (10 percent), surveys (6 percent), and design and creation (3 percent). Action research and ethnography were not used as research methods in any of the primary studies.

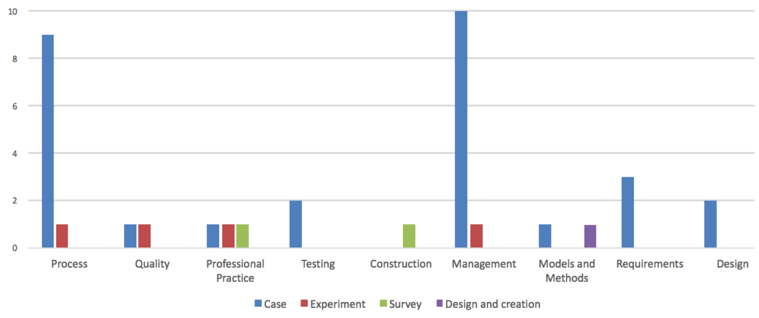


Figure 4: Empirical Evidence, 2013-2017

Figure 5 illustrates contribution types (as applied in Paternoster et al. [7], originally suggested by Shaw [58]) within each each knowledge area between 2013-2017. The 9 different knowledge areas are represented a total of 49 times through 7 different contribution types. These include (1) model, (2) theory, (3) framework, (4) guidelines, (5) lessons learned, (6) advice, and (7) tools (table A.9). Lessons learned is the most frequently used contribution type (43 percent), followed by advice (25 percent), model (12 percent), theory (10 percent), framework (5 percent), guidelines (5 percent), and tools (3 percent).

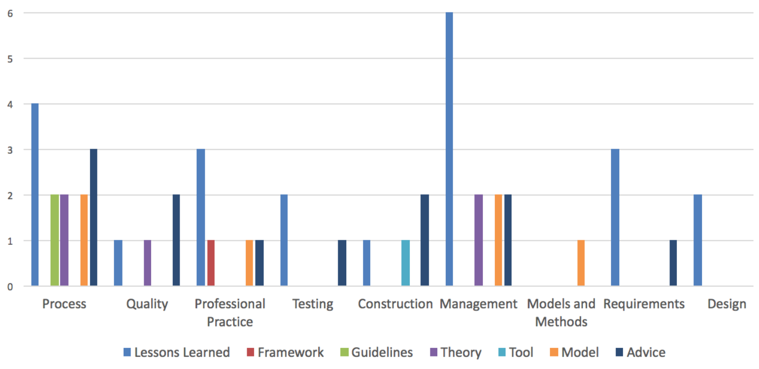


Figure 5: Contribution Types, 2013-2017

Figure 6 presents the number of papers that cover the different knowledge areas in our study (red columns) and Klotins et al. [8] (blue columns). The total number of primary papers in Klotins et al. [8] was 14. Both mapping studies include papers that cover more than one knowledge area.

The newly selected primary papers from 2013-2017 cover 9 of 15 knowledge areas. The ones missing are (1) software configuration management, (2) software engineering economics, (3) software maintenance, (4) computing foundations, (5) mathematical foundations, and (6) engineering foundations.

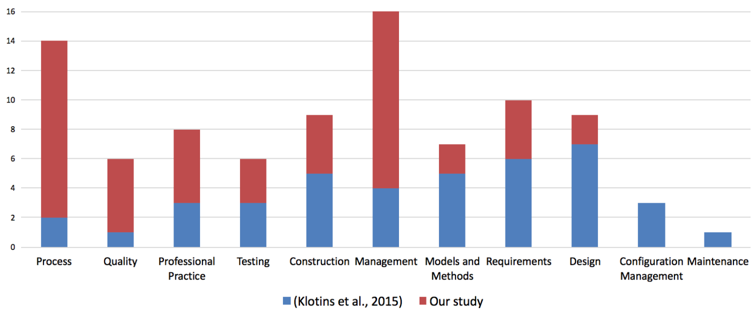


Figure 6: Knowledge Area Coverage, 1994-2017

From figure 6, we see that there is a significant change in the research direction for the last five years. Between 1994-2013 “software design” and “software requirements” are the most represented knowledge areas, whereas “software engineering process" and "software management" have received significant attention from the community between 2013-2017. "Software configuration management" and "software maintenance" are only covered between 1994-2013.

Paternoster et al. [7] did not present any results in relation to the SWEBOK knowledge areas. However, they provided the contribution type of each primary study. Figure 7 shows the contribution types of primary papers between 1994-2017, separating the periods before and after 2013. The most frequently provided contribution types between 1994-2013 were advice and model, while lessons learned was most represented between 2013-2017. The least frequently used ones combined from both studies were framework, guidelines, and tools.

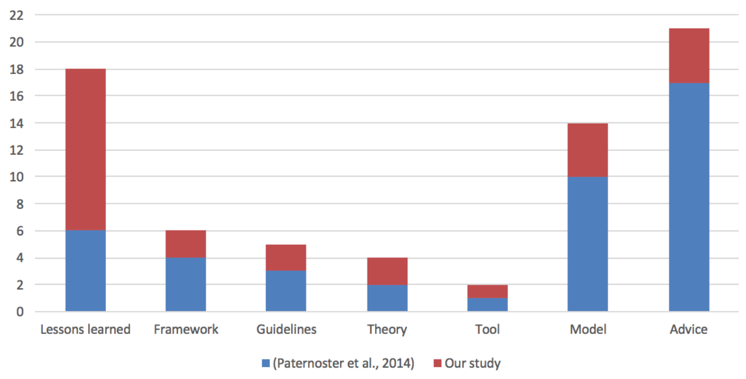


Figure 7: Contribution Types 1994-2017

4.2. RQ2: What is the relative strength of the empirical evidence reported?

To address this research question, we have made a bubble chart of each knowledge area with the corresponding rigour-rating. Among the 27 new primary papers, only the papers that provided empirical evidence were evaluated (22 of 27). The quality assessment will be compared to both of the previous mapping studies.

4.2.1. Rigour of Primary Studies 2013-2017

Publication venue can be interpreted as an initial indicator as to whether the papers provide scientific quality. Among the newly selected primary studies, 21 were conference papers, 5 were journal papers, and 1 paper was a book chapter. However, it is necessary to perform a more comprehensive quality assessment process in order to compare results across different studies.

Figure 8 shows the degree of rigour within each knowledge area between 2013-2017. The figure is based on the quality assessment (table B.10). The x-axis represents the knowledge areas, while the y-axis represents the rigour.

Only one paper received low rigour score [59], as it didn't provide enough details about the data analysis and included no assessment of the validity of the results. However, as only the papers providing empirical evidence were assessed, it is possible that more papers would receive low rigour as well. In general, the papers received high rigour score, indicating that the quality of research was high.

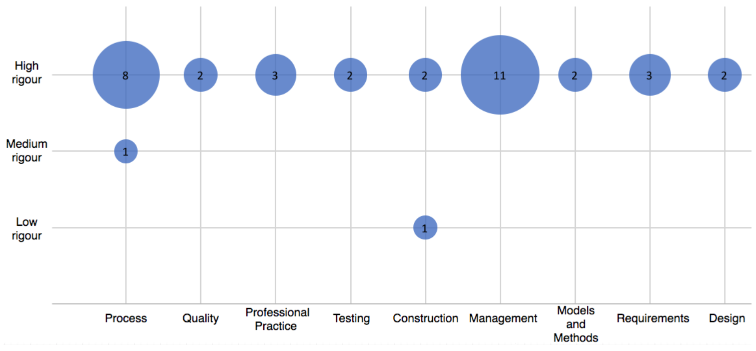


Figure 8: Rigour of each covered knowledge area, 2013-2017

4.2.2. Rigour of Primary Studies 1994-2013

Figure 9 shows the rigour of the primary studies from Klotins et al. [8], and which research type each constituted. The paper did not specify how they calculated the rigour of each paper. The x-axis represents the research types, and the y-axis represents the rigour. From 14 primary papers, only one provided a contribution of high rigour. Most of the papers (86 percent) obtained low rigour. As to this, the paper concludes that the low rigour of the papers, due to poor contextual descriptions, makes it hard to transfer results from one environment to another.

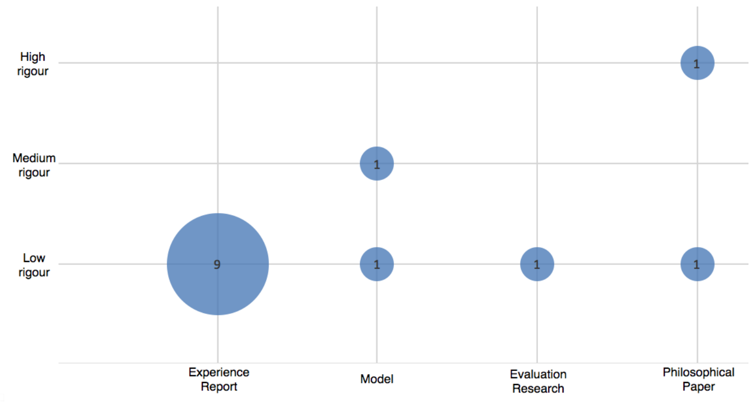


Figure 9: Rigour and Research Type [8]

Figure 10 illustrates the rigour of the contribution types provided by each of the primary papers in Paternoster et al. [7]. The x-axis represents the contribution types, and the y-axis represents the rigour. The division of rigour score is based on table 7 in the study. Papers that got a total score above 7 received high rigour, between 4 and 7 received medium rigour, while less than 4 received low rigour. 70 percent of the papers in figure 10 received a medium score, while 21 got a high score.

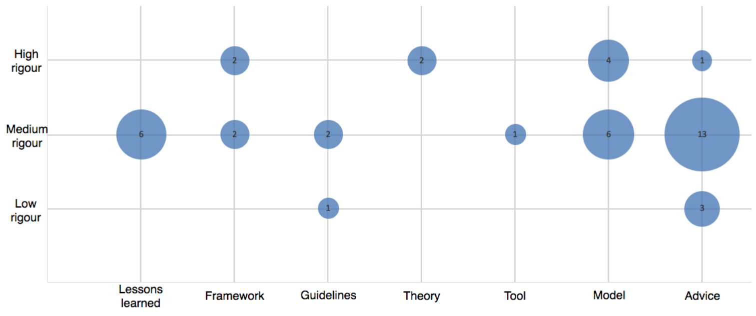


Figure 10: Rigour and Contribution Type [7]

Comparing the tables it becomes evident that the rigour of primary papers has increased from the period 1994-2013 to 2013-2017. Recently software process and management have received a significant amount of high-quality research. The other knowledge areas have received little attention, however of high-quality. The quality assessments are subject to bias from several reasons: (1) different quality assessment criteria, (2) different rating systems, (3) different researchers providing various experience and knowledge to the assessment process, (4) contributions from multiple authors from different time periods. To mitigate systematic errors, we defined clear inclusion criteria, and author one and two collaboratively assessed the newly selected primary papers.

4.3. RQ3: In what context has software startup research been conducted?

This section is divided into three sub-sections identifying the contextual descriptions provided in the period 1994-2017. Section 4.3.1 shows how papers characterize the startup context, and how they use the term for "startup company" differently. Section 4.3.2 shows whether the papers from 2013-2017 focus on startups in the context of incubators. Section 4.3.3 presents in detail the contextual descriptions provided by papers between 2013-2017.

4.3.1. Thematic Concepts and Term Frequency

To illustrate how researchers use different definitions and thematic concepts in their characterizations of startups, we extracted the thematic concepts (i.e., referred to as recurring themes in Paternoster et al. [7]) between 1994-2017. Paternoster et al. [7] extracted 15 themes from 43 papers (explained in table B.11). As to this, it is possible that other selections of papers would have provided a different set. The thematic concepts constitute a solid base for characterizing startups, presenting what are the most common perceived characteristics when talking about startups among research. A coherent use of thematic concepts to characterize software engineering can help researchers and practitioners judge whether research results can be generalized and transferred to other startup engineering contexts. To extract the frequency between 2013-2017, the first and second author read the full text of the primary papers. In addition, searches were performed in the pdf-version of each paper to find the frequency of thematic concepts.

Table 3 presents a complete usage of thematic concepts operating startup research between 1994-2017. We observe that the characterizations have changed over time (e.g., the most frequently used concept before 2013 was only the fourth most used one after 2013). The differences are significant since it is only four years between the studies. The use of concepts between 2013-2017 is highly inconsistent. There is no single concept that all the 22 empirical papers use for the startups they investigate. The low frequencies of the thematic concepts also illustrate that many of the papers provide poor startup descriptions.

Thematic Concepts	Frequency 13'-17' (#27)	Frequency 94'-13' (#47)
Innovation/Innovative	15	19
Uncertainty	14	15
Small team	11	12
Lack of resources	9	21
Little working/operating history	9	3
Time-pressure	7	17
Rapidly evolving	5	16
New company	5	8
Highly reactive	3	19
Highly risky	3	8
Third party dependency	2	12
One product	2	9
Not self-sustained	1	3
Low-experienced team	0	9
Flat organisation	0	5

Table 3: Thematic Concepts, 1994-2017

Table 4 shows the number of primary papers from 1994-2017 using the specified terms for "startup company". In situations where the title did not use any of the terms, the abstract was revised. Several papers [60, 61, 62, 63, 64] did not use any of the terms or was not found. From the table, it can be observed that the use of terms for startup companies has changed. The most significant finding is that the term "startup" is more frequently used now than before. 75 percent of the studies from 2013-2017 used the term "startup", compared to 48 percent in 1994-2013. 15 percent used the term "start-up" in the period 2013-2017, while 48 percent used the term "start-up" between 1994-2013. The inconsistent use of terms is one of the main challenges for developing a coherent body of knowledge within software startup engineering. Even though 40 studies used the term "startup", the context for which they were used was not the same, or the study context was poorly described.

Term	Frequency 13'-17' (#27)	Frequency 94'-13' (#42)
Startup	20	20
Start-up	4	20
Very small entity	1	0
Very small company	1	1
Very small enterprise	1	1

Table 4: Term Frequencies, 1994-2017

4.3.2. Incubated Companies

Figure 11 shows the percentage of the newly selected empirical papers that have performed research in the context of incubators. That is, mentioning incubators or presenting research on startups that are part of incubator environments. As illustrated, 91 percent of the papers focused on startups outside of incubator context or did not mention this in their description. Two papers focused on incubated startups [65, 66].

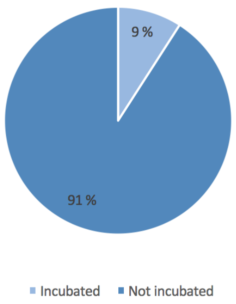


Figure 11: Percentage of Incubated Companies, 2013-2017

4.3.3. Contextual Descriptions

The primary studies from 2013-2017 that have provided empirical evidence and sufficient contextual descriptions are presented in table 5. The relevant context information includes the attributes: (1) number of startups under investigation, (2) size of the company/team, (3) the product orientation of the startups, and (4) other relevant contextual descriptions beyond these three (e.g., lifecycle stage, age/year of establishment, location, software development methodology).

As illustrated in the figure, 14 of the 22 studies showed a sufficient amount of contextual description. The contextual descriptions in the remaining eight papers were either absent or not sufficiently explained. The papers not providing empirical evidence were not evaluated. The two last papers in the table are subject to omission, as both have two fields of "not specified".

The following list presents some of the descriptions of companies that have participated in empirical research on startups, and explanation of non-trivial information.

- The number of startups under investigation is in the range from 1-20 startups. The most frequently used number of startups was found to be 3-5.
- The number of employees is usually in the range of 2-25, depending on the lifecycle stage of the company. At early stages, the number of employees tends to be equal to the number of founders, which seems to be in the range of 2-6. At later stages, more employees are needed. For the scaling phase, most companies have 10-20 employees.
- It is usual that researchers specify the product orientation of the startups (e.g., B2B/B2C).

- The age of the investigated companies is usually in the range 1 month to 3 years. Some papers investigated VSEs, one of them 18 years active [67]. Companies beyond three years of age tend to be past the scaling phase.
- Startups use different software development methods. The most usual methods found were agile, scrum, or ad-hoc.
- No more than two papers mentioned whether the investigated companies had received any funding, and if so what kind of funding they had received.
- Even if some of the investigated startups develop products with mixed software and hardware parts, no paper focused on their specific challenges or practices.
- A bootstrap startup is a company that started out without initial funding and resources [46].
- "VSEs" and "high growth firms" can in the related studies be regarded as startup companies.
- In relation to the startup stages presented in section 2.3: (1) Concept and pre startup stage are similar to stage 1. (2) Implementation, functional, and startup stages are similar to stage 2. Commercial and scaling stages are similar to stage 3. (4) Mature stage can be either stage 3 or 4.

ID	Nr of startups	Company size	Product orientation	Other relevant info
[30]	1 startup	5 members	Social network application	Roles: Designer, 1 web/iOS/android dev. each, CEO Approach: Lean Startup/Agile
[14]	3 startups	4 members	Health	Canada, mobile app, concept stage
		6 members	E-commerce	Italy, mobile and web app, func.stage
		25 members	E-commerce	Brazil, web app, mature stage
[66]	4 startups	12 members	Academic business domain	3yrs old
		10 members		1yrs old
		8 members		1yrs old (still incubated)
		10 members		4months old (still incubated)
[68]	1 startup	Not specified	Db performance & interoperability	High potential growth firm, spin-out from a university
[44]	4 startups	2 founders	Video service	Working prototype (14’)
		3 founders	SaaS	Func. product, limited users (15’)
		2 founders	Event ticketing system	Func. product, high growth (11’)
		2 founders	Game-based learning	Mature product (06’)
[28]	5 startups	6 members	Online photo marketplace	Italy (lean startup/agile,12’,impl.phase)
		3 members	Marketplace for food hub	Norway (ad-hoc,15’,concept.phase)
		4 members	Collab.platform construction	Norway (Scrum,11’,commercial.phase)
		18 members	Sale visualization	Norway (agile,11’,commercial.phase)
		3 members	Under-water camera	Finland (ad-hoc,11’,impl.phase)
[45]	5 startups	20 members	Learning game, B2C	2006, scaling phase
		18 members	Real-time sale management, B2B	2011, scaling phase
		1 member	Photo marketplace, B2C	2012, startup
		3 members	Social platform,B2C	2015, pre-startup
		1 member	Collab.platform construction, B2B	2011, startup
[46]	6 startups	6 members	Hyper-local news platform, P2P	Norway (agile,2015,bootstrap)
		9 members	Collab.platform construction, B2B	Norway (scrum,2012,bootstrap)
		3 members	Ticket event system, B2B	Norway (agile,2012,bootstrap)
		5 members	Shipping platform, P2P	UK (agile,2013,early investor)
		12 members	Game learning tool, P2P	UK (dist.agile,2013,bootstrap)
		5 members	Fish farm management, B2B	Vietnam (ad-hoc,2016,bootstrap)
[69]	2 startups	4 members	Not specified	Peru (2012, VSE/start-up term)
		2 members		Canada
[56]	3 startups	4 members	Photo market place, P2P	2011,paying customers
		5 members	Under-water camera, B2B	2009,paying customers
		12 members	Ticketing system, B2P	2011,paying customers
[67]	3 VSEs	17 members	Enterprise	18yrs active (int.customers)
		10 members	Financial services	9yrs active (int.customers)
		7 members	Enterprise	4.5yrs active (int.customers)
[5]	13 startups	3-20 members 2-6 founders	Not specified	Time-to-market: 1-12months
[65]	8 startups	Not specified	Not specified	Incubator-context 62 % used pseudo-agile for reqs. 100 % not documenting many reqs.
[13]	3 startups	Not specified	Food-waste knowledge app Online debt platform Online investment platform	Not specified

Table 5: Contextual Descriptions, 2013-2017

Table 6 presents a summary of the main findings contributing to addressing our research questions: (RQ1) *How has software startup research changed over time in terms of focused knowledge areas?* (RQ2) *What is the relative strength of the empirical evidence reported?* (RQ3) *In what context has software startup research been conducted?*

Research Question	Findings
RQ1	Most research has been conducted within the knowledge areas software process, management, construction, design, and requirements, with the shift of focus toward process and management areas. Researchers have provided lessons learned and advice studies, paying less attention to specific tools and frameworks.
RQ2	The rigour of primary papers was higher between 2013-2017 than that of 1994-2013. Two reasons for this are increased importance of startups, and increased focus on researchers providing high-quality research.
RQ3	Thematic concepts representing the software startup context include innovation, lack of resources, uncertainty, time-pressure, small team, highly reactive, and rapidly evolving. Startup literature provides an inconsistent use of thematic concepts describing startups.

Table 6: Summary of results

5. Discussion

In this paper, we have applied a systematic mapping method to analyze the 74 primary papers, to observe how software startup research has evolved and possibly matured in some Software Engineering knowledge areas. This makes it possible to identify changes in research direction for the last five years. This section presents our discussion of the newly selected papers along with the SWEBOK knowledge areas, identifying state-of-the-practice and pointing out existing research gaps. From the extracted context features, we provide a synthesized description of the startup context.

5.1. RQ1: How has software startup research changed over time in terms of focused knowledge areas?

5.1.1. SWEBOK Knowledge Areas

Software Engineering Process. The need for the software development process to be adapted to a project's scope, magnitude, complexity, and changing requirements is generally acknowledged, however, there exists a lack in guidance on how software startups can adapt their process to their situational context [68]. The situational context consists of a large number of concerns and factors, as found in the "reference framework" [70], indicating why software engineering is so hard [68]. The situational factors in the reference framework explain the need for startups' own software development processes, and why strictly following the agile methodology is often outside the scope of small startups [30]. In early-stage software startups, research shows that systematic software engineering processes often are replaced by light-weight ad-hoc processes [11].

Software startups need a model fitted to both the innovation, and engineering processes in startups' complex and chaotic situations. The Hunter-gatherer cycle is one model proposed to help startups in all phases of the company, from their evolution from innovative ideas to commercial products. The model differentiates between the hunting cycle, and the gathering cycle, which covers the innovation and engineering activities. The model is at a preliminary stage and requires more empirical evidence in order to be generalized to all software startups [56].

Software Engineering Professional Practice. Software engineers need to possess the required knowledge, skills, training, and experience to practice professional and responsible software engineering [9]. Standards like ISO are meant to ensure high quality and reliability of software products [71], and can thus help developers to practice software engineering at a level in line with these objectives. The paper by Laporte and O'Connor [51] presents results from early trials of the ISO/IEC 29110 standard for very small entities and concludes that international certifications can enhance small software companies' chances for success.

Developers in software startups typically prioritize speed related agile practices rather than quality related ones [72]. Standards like ISO, tailored to the startup context, can help software developers combine quality and speed, which in turn can increase the chances for success. However, as the ISO/IEC 29110 standard mainly is intended for very small companies, it is only partially relevant for startups. Future work should be undertaken to develop an ISO standard tailored to the startup context, to support developers in practicing professional software engineering. In general, there was a lack of research supporting professional practice in software startups.

Engineering foundations is one of the 15 knowledge areas that was not covered in any paper. It is about the application of knowledge in the engineering discipline, allowing engineers to develop and to maintain software more efficiently and effectively, and help practitioners to adopt professional software engineering principles. As to this, more work should be undertaken to identify the engineering foundations of software developers in startups. Most prior research has focused on the needs of established companies. A possible research area could be to investigate which engineering practices graduates and other engineers should possess if they are to work in a software startup, and how universities and other educational institutions can facilitate learning and other services to support the specific needs of practitioners that are to work in software startups.

Software Engineering Management. Software engineering management concerns about a wide range of different areas, including planning, measuring, coordinating, and reporting activities to support systematic software development and maintenance [9]. For startups, software engineering management relates to, among other things, business model experimentation and customer development.

Three primary studies that have identified software engineering management are [73, 56, 74]. However, these papers primarily focus on software engineering processes [73, 56] and software quality [74]. Although the Hunter-gatherer cycle presented by Nguyen-Duc et al. [56] presents how startups can handle the dynamic evolution of product-market fit, which is part of both business experimentation and customer development, it is primarily focused towards software engineering processes in startups.

The managerial part of software engineering has been identified by Nguyen-Duc and Abrahamsson [46], exploring the outsourcing relationship in software startups. They concluded that outsourcing is a feasible option for early-stage startups. The authors are underway to provide a guideline with best practices for outsourcing in startups.

Other papers have focused on principles from Lean Startup, especially the role of pivoting in software startups. This includes why startups pivot [29], and which pivot types exist [44]. More work is required to address the consequences and relationship among different pivot types, both from a business and technical perspective. How pivoting should be performed at different lifecycle stages, both in terms of system complexity and modifiability, may affect the pivoting decision.

The research agenda [1] has addressed a need for more research to identify how startups explicitly manage risks, and how startups can model and measure risks. This further relates to which tools and techniques they should utilize to preserve agility and speed in dynamic environments of high uncertainty. Lean Startup offers entrepreneurs a method to handle such environments, but more empirical evidence is needed to understand how software startups apply this theory in practice so that researchers can develop tailored models and frameworks to reduce business and technical risks.

Software Quality. A frequent issue in terms of software quality for startups is technical debt. The development of minimum viable products, and releasing the product as fast as possible, often require the development team to take shortcuts and workarounds. Steve McConnell showed that technical debt can be divided into intentional and unintentional debt [75]. Shortcuts can lead to the accumulation of intentional technical debt, while unintentional technical debt can happen when business model experimentation is leaved out [74]. No matter how good the idea may seem, not validating the idea with customers could lead to the development of unnecessary features.

Not focusing on technical debt will have consequences for the product quality, while constantly changing and improving the business model will be necessary to stay competitive [74]. Finding the correct balance is therefore essential. The same problem is referred to as the developer's dilemma [76]. The developer's dilemma also emphasizes the need for managers to communicate the learning goals of the product precisely so that developers can adjust the quality accordingly. Not finding the correct match between learning goals and quality will often lead to technical debt, waste, or missed learning.

To help startups focus on technical debt, one estimation method is proposed based on Visual Thinking [77]. The technique is based on "duck taping" each part of the code that is developed or fixed in a messy way, to keep an overview of what might cause quality issues in the future. Measuring technical debt is hard, and as the author also concludes, the method needs more empirical evidence as to whether it actually is capable of solving issues related to technical debt.

Software Construction. There exists a wide range of various software tools to speed up the development processes in software startups. However, as Edison et al. [59] suggest, there does not exist a clear understanding of how entrepreneurs can use the different tools efficiently to meet their specific needs. As to this, the paper describes the outline of a system that provides a software tool portal that supports and recommends which tools to use in the construction of software products and services. The portal can be directly connected to the research agenda [1], which addresses a need for how software tools can be recommended and used by entrepreneurs.

According to our findings, there is a general lack of research within the field of software construction in startups. A software tool portal can indeed be helpful to support software construction. However, such a portal is not specifically addressing how to construct software. Software construction includes the management and practicalities of construction and the use of technologies and tools to develop software [9]. As to this, it can be feasible to address software construction through sub-categories, like design, testing, and verification.

Software Engineering Methods and Models. The models and methods knowledge area aims at making software processes more success-oriented through systematic and repeatable activities at different lifecycle stages [9]. Topics include principles and properties of models, analysis of models, and various software development methods.

Startups need software development methodologies and techniques tailored to their specific contexts. These should be based on Lean Startup and agile principles [7]. Researchers are encouraged to identify what engineering methods and models that are used today, and whether they work in a startup context [1].

The Greenfield Startup Model (GSM) aims at explaining how development strategies and practices are engineered and utilized in startups [5]. A similar model, the academic startup model, was created by Souza et al. [66], which illustrates how software startups structure and execute their engineering activities. Both papers conclude that early-stage software startups do not adopt traditional development methodologies. Instead, rapid prototyping and continuous experimentation are in focus, hence engineering practices are adapted to each startup's specific context. These models provide development objectives that software engineers in startups can use, as well as guidelines for future research aiming at improving the current state-of-the-art.

We see an existing need to validate the software engineering models adapted to the startup context. This includes areas like technical debt management for particular contexts, and how new models from academia and industry can be applied in the startup context [5].

Software Testing. Software validation and testing are essential parts of all software engineering processes. Software testing is both costly and time-consuming, and without sufficient knowledge about customers and users, it can be difficult for startups to apply necessary testing practices in the development of high-quality software products and services.

Pompermaier et al. [65] found that testing is critical to startups' success. However, in the construction phase of the first version of the system, technical teams did not use any software testing techniques. This changed in the following phases, where 75 percent of the technical teams used software testing techniques. The most common testing techniques were unit tests (37 percent), pilot clients (25 percent), functional tests (25 percent), and specialist testers (13 percent).

Due to the importance of testing, startups should apply testing techniques at a more consistent and detailed level to enhance the quality and professionalism of their development processes. Apart from the results presented by Pompermaier et al. [65], more research is required to identify and develop methods for how startups can enhance their current testing processes, even in contexts of scarce resources and time-pressure. Research should look at how startups can learn from established companies' systematic testing processes, even if they have significantly different needs for, and usage of such methods. Finding an optimal balance between cost/time spent on testing activities and how this evolves over time in startups can help them in the introduction of good software testing practices [1].

Software Requirements. Software requirements engineering activities include elicitation, negotiation, analysis, specification, and validation of requirements [9]. As startups lack knowledge about their customers and users, it becomes difficult to identify and also verify all requirements. How much time should be spent on requirements is challenging to estimate when you don't know whether the requirements actually will be implemented. This, in turn, makes it difficult to estimate time and cost of software development. To deal with these ambiguities, startups should apply techniques from the Lean Startup methodology [12]. Prototyping, continuous experimentation of minimum viable products, and pivoting are effective tools and methods startups can utilize in their requirements engineering processes [1].

Rafiq et al. [14] found that there was a lack of studies investigating how software startups perform requirements engineering processes. The study found that requirements mainly were elicited through the founders' assumptions and interpretations of the market. These were based on several different requirements elicitation techniques, including prototyping, interviews, questionnaires, feedback comments analyses, competitor analyses, similar product analyses, collaborative team discussions, and model users. Although elicitation techniques were used, the startups did not define the requirements explicitly. This resulted in a lack of formal documentation, both before and after the elicitation process.

Future research should investigate a larger number of software startups to identify a greater amount of elicitation techniques and to provide stronger evidence of the findings in Rafiq et al. [14]. More research should be undertaken to identify negotiation, specification, and validation techniques. In addition, research is necessary to identify requirements engineering for different lifecycle stages to help startups in specific situational contexts identify appropriate requirements engineering techniques.

Software Design. The role of MVPs in software startups has been addressed by Nguyen-Duc and Abrahamsson [28]. They suggest that MVPs are effective tools for requirements elicitation, and for bridging knowledge gaps between entrepreneurs, investors, and software developers - emphasizing that MVPs can serve as a multiple facet product. A research topic requiring more work is how software prototype practices can be applied in an agile development context, and how startups can benefit from adopting open source software in prototyping.

The speed of prototyping has been addressed by Nguyen-Duc et al. [17]. The factors that influence the speed of prototyping can be grouped into artifacts, team competence, collaboration, customer, and process dimensions. These factors, along with the uncertainties of the startup context make it important to define practices and processes to support decision-making in prototyping. While throw-away prototypes are used mainly for specification and experiments, evolutionary prototypes provide a basis for complete systems, usually developed with extensive reuse. Customer feedback is an essential part of business experimentation and is mainly done through prototyping. More work is required to identify what kinds of learning different prototypes provide and to identify effective prototyping and development patterns among software startups.

5.1.2. Startup Research 1994-2017

Matching the primary papers with the right knowledge area can be challenging, one reason being their relevance to the startup context. Another issue is that different perceptions of knowledge areas can give different classifications. Different authors' biases in terms of knowledge and personal opinions will also lead to different classifications.

Looking at the knowledge areas covered between 1994-2017, we see that software maintenance and software configuration management have received few contributions. As one of the most important objectives for startups is to grow and scale their business, both maintenance and configuration management becomes more important at more mature lifecycle stages. No papers between 2013-2017 focused on these knowledge areas, illustrating their irrelevance to the startup context.

Four knowledge areas were not covered at all (computing, mathematical, and engineering foundations, and software engineering economics). They characterize the educational requirements of software engineering, hence not particularly relevant for specific software startup research. However, we argue that more research should be provided within engineering foundations, as it can serve as a prerequisite for software practitioners in startups. Apart from these findings, we observe that the areas models and methods, testing, and quality have received few contributions. In contrast to the educational requirements, these are of greater importance in all startup lifecycle stages, and should thus be given more attention in future work.

Areas with numerous contributions include software engineering process, software engineering management, software construction, software design, and software requirements. Management was suggested by Klotins et al. [8] as a potential area for future work. Recently, several papers have contributed to important managerial aspects like pivoting, experimentation, and the role of prototypes to define and assess business and development scope. It is clear that the startup context requires fast and effective decision-making, both at a managerial and technical level. Software requirements engineering is important to manage in order to minimize time and costs and avoid feature creeps related to prototyping and business experimentation.

Another area not sufficiently covered between 1994-2013 was software engineering process. This is in contrast to the last five years, where process has received most contributions. Klotins et al. [8] argue that software engineering process becomes relevant for the maturity phase when product development is more robust and processes more predictable. As to this, the software process knowledge area is more relevant for SMEs. In our study, however, we have regarded process as relevant for early-stage development as well, illustrating the different interpretation among researchers.

The publication frequency of primary papers between 1994-2017 indicates that increasingly more papers are published. No other year is more represented than 2017, which indicates that there is an increased focus on research within the field. This can be seen as a direct response to the research agenda's [1] identified need for more research and the increased impact and importance of startups in today's technology innovation processes. The highly dynamic markets and ever-increasing customer demands lead to a high failure rate among startup companies. Empirical studies have found that although startups try to adopt Lean Startup principles and agile methods, they generally find it hard to apply them [5, 66]. More research is thus required to support entrepreneurs and software developers to enhance their chances of success. With the increased publication frequencies in mind, it seems that more work is undertaken to address startups' unique needs.

Software startups find it hard to apply theory in practice, a claim supported by both empirical research and the high failure rates. Looking at the contribution types from 1994-2017, we observe that the most frequent ones are advice, lessons learned, and models, while the least frequent ones are tools, guidelines, and frameworks. Between 2013-2017, lessons learned has been the most popular contribution type, while previously advice was more popular. What we can make from these numbers is that researchers mainly have focused on providing advice and learnings to the startup community. As to this, we suggest that researchers should provide knowledge from state-of-the-practice to support startups with specific tools and frameworks. This could allow for a broader coverage of startups' needs and unique requirements.

5.1.3. Identification of Research Gaps

By structuring literature within the field from 1994-2017, this study allows for identifying whether research from the last five years has addressed research gaps suggested by the previous mapping studies. In addition, this section will point out directions for future research.

Paternoster et al. [7] expected more studies to contribute to the adoption of agile practices in startups. In particular, they recommended the need for future studies to provide techniques for aligning business goals of software startups with the execution of specific development practices. Another area suggested for future research was the development of customer collaboration processes for requirement elicitation, allowing for testing the problem before releasing the product to market. Lastly, they identified the need for improved verbal communication with the introduction of new tools and techniques to enhance knowledge transfer in startups. These research gaps have only partly been addressed the last five years. Towards agile methodologies and techniques tailored to the startup context Pantiuchina et al. [72] provide a better understanding of the current adoption of agile practices in software startups. The study indicates that speed-related agile practices are more frequently used than quality-related practices. Comparable findings have been presented by Yau and Murphy [30], stating that a rigorous agile methodology intended for established companies may not be applicable to the startup environment. In relation to customer development Chanin et al. [13] present the results from applying a customer development process to three startups, indicating that the process can improve the requirements process. Others have also contributed to addressing customer development and requirements elicitation [14, 28, 44]. We could not identify research directly related to team communication, documentation, or knowledge transfer.

Klotins et al. [8] stated that there is an insufficient understanding of quality requirements role in software startups, and that maintenance of product integrity in startups is yet to be explored. This is especially relevant due to the evolutionary approach and restricted resources of startups. Similar to Paternoster et al. [7], Klotins et al. [8] highlight the need for addressing the relation between software technical decisions and organizational business goals, and a better understanding for human capabilities in startups. Comparable to the need for customer development processes presented by Paternoster et al. [7], Klotins et al. [8] identified the need to further investigate the role of scope in software startups. Discovering the right scope can greatly improve development speed, by identifying the necessary features and effort. Since 2013 empirical research has been undertaken to explore state-of-the-practice in testing activities of software startups [65], and the accumulation of technical debt [74, 76, 77].

We suggest future work to compile a set of agile practices that provide enough benefits to be adopted in the startup context, overcoming challenges related to cost and time of implementing a process model without compensating the demand for speed in early-stage startups. A development methodology should include specific practices related to communication in the growing number of stakeholders. We also emphasize future studies dedicated to the role of human capital in startups, investigating capabilities and engineering foundations of startup practitioners. In addition, we highlight the need for studies exploring challenges and engineering approaches of startups developing products with mixed hardware and software parts. Finally, more work is needed to cover the partly filled research gaps identified by the previous mapping studies.

5.1.4. Future Classification

Unterkalmsteiner et al. [1] identified more than 70 research questions in different areas supporting activities of software startups. The researchers contributing to the paper are all part of a network (The Software Startup Research Network) of researchers that have created eighteen research track descriptions to ease the presentation and discussion of the research agenda. These eighteen research tracks were grouped into six themes based on similarities.

Classifying the newly selected primary studies according to the SWEBOK knowledge areas resulted in 9 out of 15 of the areas being addressed. This indicates that some of the knowledge areas might not be related to startups, while some are of big interest. The same pattern was discovered in Klotins et al. [8], which used SWEBOK for lack of a better alternative. The low coverage is most certainly because most of the research in the field of software engineering is undertaken in relation to established companies, from which the SWEBOK knowledge areas are developed.

For future mappings, it would be sensible to categorize the papers into the newly established research themes [1]. These are better suited for startup research and can help guide researchers in providing knowledge to the specific areas that are most important for the challenges faced by startups. Do notice that number 7 and 8 require more evidence as to whether they can be related to software startup engineering: (1) Supporting startup engineering activities, (2) Startup evolution models and patterns, (3) Human aspects in software startups, (4) Applying startup concepts in non-startup environments, (5) Startup ecosystem and innovation hubs, (6) Methodologies and theories for startup research, (7) Marketing, (8) Economics and business development.

5.2. RQ2: What is the relative strength of the empirical evidence reported?

Paternoster et al. [7] provided a mapping of the research within software development in startups for the period 1994-2013, including 43 primary studies. Each study provided empirical evidence as this was a quality criterion of the mapping. Overall, only 4 of these papers were found to be (1) contributions entirely dedicated to engineering activities in startups, (2) providing a strong contribution type, and (3) conducted through an evidence-based research approach [10, 37, 38, 39].

The mapping study Klotins et al. [8] found that (1) most of their primary studies did not compare and analyze data from more than one case, and (2) most studies had low rigour, making it difficult to compare results. As to this, they emphasized the need for more empirical research to provide stronger evidence and enable results to be generalized to all software startups. More specifically, the paper identified a lack of studies related to requirements processes, the "developer's dilemma" (as discussed in section 5.1.1), software architecture, and software engineering processes.

Figure 4 shows the areas that have received most contributions in terms of empirical research between 2013-2017. These are software engineering process, software engineering management, and software engineering professional practice. On the contrary, five knowledge areas received less than five scientific contributions, arguing the need for more research, even within areas that have gotten attention from the research community. This is also in line with the research agenda's addressed need for further empirical evidence [1].

Comparing the provided quality between papers published before and after 2013 we see that the quality of work is improving. As for the previous studies, the reported rigour of the primary papers was at a generally lower level than what was found in the newly selected papers, where only one paper obtained low rigour. Possible explanations are the different quality assessment methods used, and the increasing number of researchers contributing to the field. Another reason may be the assessment bias of different researchers.

Several of the researchers who have contributed to the newly selected primary papers are members of The Software Startup Research Network, whose aim is to provide entrepreneurs and the research community with novel research findings within the area of software startups. Anh Nguyen-Duc, one of the members of this network, has participated in six of the primary studies, in which all received a high rigour score. Another researcher who has contributed to three of the primary studies is Rory V. Connor, where all three papers received a high rigour score. He participated in three primary studies in the previous systematic mapping study as well, all of which obtained high rigour. As to this, it seems that the quality of research is becoming increasingly high compared to before, justifying the high rigour obtained by the quality assessment in this mapping study. The quality of work was reported to be a problem area in both of the previous mapping studies. However, as our findings suggest, there is an increased focus on providing high-quality research with several researchers contributing with multiple papers, as illustrated by specific initiatives that promote scientific work.

5.3. RQ3: In what context has software startup research been conducted?

The most frequently used term for referring to startup companies is "startup", with 54 percent of the 74 primary papers using this term. Between 2013-2017, the same percentage has increased to 75 percent. In order to create a coherent definition for startups, ideally, only one term should be used. The research community has moved towards a common use of "startup", and this should thus be used for future research when referring to companies in the startup context. Inconsistent usage of the term, like "start-up", "start up", or "very small entities" in startup context should be avoided, and makes it difficult for both practitioners and researchers to adopt relevant results.

Primary papers showed an inconsistent use of thematic concepts when describing startup companies, with no single factor being used by all papers. As stated in the results, this also relates to the poor contextual descriptions found in several of the papers. We observe that the usage has changed significantly between 1994-2017, where only "Time-pressure" was used to the same extent before and after 2013. Interestingly, the least used concept between 1994-2013 is the fifth most used concept by the papers between 2013-2017. As to this, it seems that some of the thematic concepts found in the previous mapping study are no longer the ones used by the community.

The many different descriptions of startups make it challenging to develop a coherent definition and body of knowledge for the startup context. Based on the frequency of concepts found in the primary papers between 1994-2017, we argue that at least the concepts occurring in more than 25 percent or more of the studies should be part of a unique definition of startups. The following thematic concepts were: (1) Innovation/innovative, (2) Lack of resources, (3) Uncertainty, (4) Time-pressure, (5) Small team, (6) Highly reactive, (7) Rapidly evolving. Thematic concepts that were not very relevant for startups include not (1) self-sustained, (2) low-experienced team, (3) one product, and (4) flat organization. These concepts should be avoided as the primary definition by researchers in the community.

Many of the newly selected primary studies did not explain the startups under investigation sufficiently. From the 22 papers providing empirical evidence, only 14 of these provided a sufficient amount of descriptions (table 5). Interestingly, only two papers mentioned incubators as part of the startup context. Without a unique definition in literature, the importance of precise descriptions becomes even bigger, especially for transferring results from one environment to another [8].

"Team size" received little focus. A startup with 5 employees have different needs and challenges from a startup with more than 150 employees (e.g., communication needs) [78]. Even though team size will affect engineering practices to a large degree, too few of the primary papers presented the team size of the startups investigated. More research is needed to understand how software engineering practices change according to team size, and to what extent team size should be part of a unique definition of startups. We found that the usual number of employees in investigated startups was 2-25. The number depends on their respective lifecycle stage or the age of the company. A startup usually starts with 2-6 founders, but as the business scales, more employees are required. This will, in turn, affect the startup's need for software processes. As to this, we emphasize that researchers must be aware of which describing concepts that are relevant for the startups they are investigating, and that they specify this in their work. More consistent focus on the situational context is a vital step towards a more coherent body of knowledge.

The research track of Unterkalmsteiner et al. [1] aims at developing a software startup context model that would allow a coherent characterization of software startups. Since there is no agreement on a standard definition, it is challenging to provide coherent contributions to the research area.

6. Conclusion

In this study, we have applied a systematic mapping method to analyze the literature related to software startup engineering. A total number of 74 primary papers (in which 27 papers are newly selected) were extracted and synthesized. Our study, along with the previous mapping studies, constitute a merging of the primary literature within the field for the period 1994-2017, including the focus and relative strength of research, and the effort that's been made to characterize the software startup context.

The contribution of this mapping study is two-fold. Firstly, the study provides a comprehensive view of software startups for Software Engineering researchers. Possible research gaps are derived for future studies. Secondly, the study provides a map of the contextual setting of investigated startups, inferring the applicability area of empirical findings. This can help to compare and to generalize future research in software startups.

Regards to RQ1, most found software startup research between 2013-2017, are conducted within software engineering management and software engineering process, while software design and software requirements have received most attention between 1994-2013. For the period 2013-2017, software design received fewer contributions compared to that in 1994-2013, illustrating a change of research direction. The knowledge areas software engineering models and methods, software quality, and software testing have received little attention from the research community during the period 1994-2017. Apart from these findings, we emphasize the need for more research within all knowledge areas. For the period 2013-2017 software configuration management and software maintenance were not covered at all. As to this, it seems that some of the knowledge areas aren't directly relevant to the startup context. Future mappings should instead use the newly established research themes of Unterkalmsteiner et al. [1].

Regards to RQ2, we found an increased rigour of primary studies after 2013 in comparison with studies found in 1994-2013. While it is still not clear about the transformation of research results to startup practitioners, startup researchers seem to increase the focus on conducting high-quality research. The increased importance of startups has been an important factor to highlight the need for more research. As startups generally use ad-hoc or opportunistic development methods, practices of startups can be different, meaning that more evidence is needed to generalize work practices to all startups.

Regards to RQ3, we identify a coherent set of concepts that represent the startup context, (1) Innovation/innovative, (2) Lack of resources, (3) Uncertainty, (4) Time-pressure, (5) Small team, (6) Highly reactive, (7) Rapidly evolving. Additionally, aspects like (1) team size, (2) product orientation, (3) number of active years/life cycle stage, (4) number of investigated startups, (5) location, and (6) development method are important to describe sufficiently to be able to transfer results from one environment to another. As only 14 of the primary papers between 2013-2017 provided adequate descriptions, and all primary papers showed an overall inconsistent use of describing thematic concepts, we see a need for a more comprehensive endeavor to describe the engineering context of startups.

Several threats to validity were considered, including the selection of papers, the use of few online bibliographic databases, the selection of keywords, and the coarse-grained classification used for the quality assessment. To ensure the selection process was unbiased, the selection criteria were developed in advance, also the first, second and third author were involved in the selection process. Both the use of few online bibliographic databases and the identified keywords and search terms might have lead to relevant papers being omitted. This risk was mitigated by performing an additional manual search. For the quality assessment, it is likely that the use of only two points have caused the papers to obtain a higher rigour than they would have if a more fine-grained assessment method had been used.

Future work can focus on certain research themes, such as startup evolution models and human aspects, and consolidate the contextual factors of software startups. More work should be conducted for specific business contexts, such as startups that are part of incubators and bigger business ecosystems. As a next step, we seek to address engineering practices in startups who deliver both hardware and software, as no prior studies have been entirely dedicated towards their specific challenges and demands.

Appendix A.

Appendix A presents the classification schema (A.7), which includes the classification of each primary paper. Table A.8 and table A.9 explain some of the attributes of the classification schema in more detail.

ID	Research Method	Contribution Type	Knowledge Area	Pertinence	Term for startup	Incubator context	Publisher
[30]	Case study	Lessons learned	Process	Full	Startup	No	Penn University
[51]	Experiment	Lessons learned	Professional Practice, Management, Quality	Partial	Start-up	No	QUATIC
[79]		Framework	Professional Practice	Full	Start-up	No	ACM
[69]	Experiment	Guidelines	Process	Full	Start-up	No	ENASE
[74]	Case study	Theory	Process, Management, Quality	Partial	Startup	No	Springer
[59]	Survey	Tool	Construction	Full	Startup	No	Springer
[56]	Case study	Model	Process, Management	Full	Startup	No	ACM
[67]	Case study	Lessons learned	Process	Full	Very Small Company	No	Springer
[31]		Guidelines	Process	Partial	Startup	No	Springer
[1]		Advice	Process, Management, Professional Practice, Construction, Requirements, Quality, Testing	Full	Startup	No	EISEJ
[76]		Advice	Quality, Construction	Full	Startup	No	Springer
[73]	Case study	Lessons learned	Process, Management	Partial	Very Small Enterprise	No	IEEE
[5]	Design and creation	Model	Models and Methods	Full	Startup	No	IEEE
[44]	Case study	Lessons learned	Management, Requirements	Full	Startup	No	Springer
[28]	Case study	Lessons learned	Management, Design, Construction	Full	Startup	No	Springer
[45]	Case study	Model	Methods and Models, Management	Full	Startup	No	IEEE
[46]	Case study	Advice	Management, Process	Full	Startup	No	EASE
[29]	Case study	Lessons learned	Management, Testing	Full	Startup	No	Springer
[47]	Case study	Theory	Management, Process	Full	Startup	No	Springer
[17]	Case study	Lessons learned	Management, Design	Full	Startup	No	Springer
[65]	Case study	Lessons learned	Process, Testing	Full	Startup	Yes	KSI Graduate School
[72]	Survey	Lessons learned	Professional Practice	Full	Startup	No	Springer
[14]	Case study	Lessons learned	Requirements	Full	Startup	No	IEEE
[66]	Case study	Model	Professional Practice	Full	Startup	Yes	IEEE
[77]		Framework	Quality	Full	Startup	No	IEEE
[13]	Case study	Lessons learned	Requirements	Full	Startup	No	IEEE
[68]	Case study	Advice	Process	Full	Start-up	No	Springer

Table A.7: Classification Schema

Research Methods

Research Method	Description
Survey	Obtain the same kinds of data from a large group of people in a standardized, systematic way to find patterns through statistics.
Design and creation	Development of new IT products or artifacts, or even a model or method.
Experiment	Investigation of cause and effect relationships through hypotheses-testing and proofs. Typically "before" and "after" measurements.
Case study	Focusing on one instance of the "thing" being investigated to obtain a rich, detailed insight into the case and its complex relationships and processes.
Action research	Plan to do something in real life, do it, and reflect on the outcome and learnings.
Ethnography	Focusing on understanding the ways of seeing a specific group of people through field research.

Table A.8: Research Methods [57]

Contribution Types

Contribution Type	Description
Model	Representation of an observed reality by concepts or related concepts after a conceptualization process
Theory	Construction of cause-effect relationships from determined results
Framework/methods	Models related to constructing software or managing development processes
Guidelines	List of advices, synthesis of the obtained research results
Lessons learned	Set of outcomes, directly analyzed from the obtained research result
Advice/implications	Discursive, and generic recommendation, deemed from personal opinions
Tool	Technology, program or application used to create, debug, maintain or support development processes

Table A.9: Contribution Types [7]

Appendix B.

Study	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Score
[30]	1	1	0	1	1	1	0	0	1	0	6
[51]	1	1	1	1	1	1	1	0	1	0	8
[69]	1	1	1	1	1	1	1	0	1	0	8
[74]	1	1	1	1	1	1	1	0	1	1	9
[59]	1	1	0	0	0	0	0	0	1	0	3
[56]	1	1	1	1	1	0	0	1	1	1	8
[67]	1	1	1	1	0	1	1	0	1	1	8
[73]	1	1	1	1	1	1	1	0	1	0	8
[5]	1	1	1	1	1	1	1	1	1	1	10
[65]	1	1	1	1	1	1	1	0	1	1	9
[72]	1	1	1	1	1	1	1	0	1	1	9
[14]	1	1	1	0	1	0	1	1	1	1	8
[66]	1	1	1	0	1	1	1	0	1	1	8
[68]	1	1	1	1	1	1	1	0	1	0	8
[44]	1	1	1	0	1	1	1	0	1	1	8
[28]	1	1	1	1	1	1	1	0	1	1	9
[45]	1	1	1	1	1	1	1	1	1	1	10
[46]	1	1	1	1	1	1	1	1	1	1	10
[29]	1	1	1	1	1	1	1	1	1	1	10
[47]	1	1	1	0	1	1	1	0	1	0	7
[17]	1	1	1	1	1	1	1	1	1	1	10
[13]	1	1	1	1	1	0	1	1	1	0	8

Table B.10: Quality assessment 2013-2017, based on table 3 in Nguyen-Duc et al. [48]

Recurring Themes	Explanation
Lack of resources	Economical, human, and physical resources are very scarce or limited.
Highly reactive	Startups can react very fast to changed market conditions, technologies, or changed customer demands.
Innovative	The startups focus on innovative market segments, most likely where they can disrupt markets.
Uncertainty	The ecosystem in which the startups operate within are very uncertain, wrt. customers, competition, technologies.
Rapidly evolving	Startups' objective is to grow and scale rapidly.
Time-pressure	The market and environment demands fast product releases and constant pressure.
Third party dependency	Startups need to rely on external entities and technologies in their lack of time and resources.
Small team	The startup consist of a small number of individuals.
One product	The startup is only concerned with the development of one product.
Low-experienced team	Maximum five years of experience or newly graduated students.
Highly risky	The failure rate of startups is high.
New company	The company is newly established.
Flat organization	All individuals in the company have shared responsibility, no high-management.
Not self-sustained	External funding is required, especially in the early-stages.
Little working/operating history	There is a lack of organizational culture as the startup is young.

Table B.11: Explanation of recurring themes, based on table 6 in Paternoster et al. [7]

References

[1] M. Unterkalmsteiner, P. Abrahamsson, X. F. Wang, N. D. Anh, S. Shah, S. S. Bajwa, G. H. Baltes, K. Conboy, E. Cullina, D. Dennehy, H. Edison, C. Fernandez-Sanchez, J. Garbajosa, T. Gorschek, E. Klotins, L. Hokkanen, F. Kon, I. Lunesu, M. Marchesi, L. Morgan, M. Oivo, C. Selig, P. Seppanen, R. Sweetman, P. Tyrvainen, C. Ungerer, A. Yague, Software startups - a research agenda, E-Informatica Software Engineering Journal 10 (1) (2016) 89–123. doi:10.5277/e-Inf160105.
URL <GotoISI>://WOS:000387014900006

[2] M. Marmer, B. L. Herrmann, E. Dogrultan, R. Berman, C. Eesley, S. Blank, Startup genome report extra: Premature scaling, Vol. 10, 2011.

[3] M. Crowne, Why software product startups fail and what to do about it. evolution of software product development in startup companies, in: Engineering Management Conference, 2002. IEMC'02. 2002 IEEE International, Vol. 1, IEEE, 2002, pp. 338–343.

[4] C. Giardino, S. S. Bajwa, X. Wang, P. Abrahamsson, Key challenges in early-stage software startups, in: International Conference on Agile Software Development, Springer, 2015, pp. 52–63.

[5] C. Giardino, N. Paternoster, M. Unterkalmsteiner, T. Gorschek, P. Abrahamsson, Software development in startup companies: The greenfield startup model, IEEE Transactions on Software Engineering 42 (6) (2016) 585–604. doi:10.1109/TSE.2015.2509970.
URL http://dx.doi.org/10.1109/TSE.2015.2509970

[6] S. M. Sutton Jr, Role of process in a software start-up, IEEE Software 17 (4) (2000) 33–39. doi:10.1109/52.854066.
URL http://dx.doi.org/10.1109/52.854066

[7] N. Paternoster, C. Giardino, M. Unterkalmsteiner, T. Gorschek, P. Abrahamsson, Software development in startup companies: A systematic mapping study, Information and Software Technology 56 (10) (2014) 1200–18. doi:10.1016/j.infsof.2014.04.014.
URL http://dx.doi.org/10.1016/j.infsof.2014.04.014

[8] E. Klotins, M. Unterkalmsteiner, T. Gorschek, Software Engineering Knowledge Areas in Startup Companies: A Mapping Study, Vol. 210 of Lecture Notes in Business Information Processing, 2015, pp. 245–257. doi:10.1007/978-3-319-19593-3_22.
URL <GotoISI>://WOS:000365180900024

[9] P. Bourque, R. E. Fairley, Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0, IEEE Computer Society Press, 2014.

[10] G. Coleman, R. V. O'Connor, An investigation into software development process formation in software start-ups, Journal of Enterprise Information Management 21 (6) (2008) 633–648. doi:10.1108/17410390810911221.
URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-55349133834&doi=10.1108%2f17410390810911221&partnerID=40&md5=9a7aca62e6f24c6416fd3034dbb66b0a

[11] C. Giardino, M. Unterkalmsteiner, N. Paternoster, T. Gorschek, P. Abrahamsson, What do we know about software development in startups?, IEEE Software 31 (5) (2014) 28–32. doi:10.1109/MS.2014.129.
URL http://dx.doi.org/10.1109/MS.2014.129

[12] E. Ries, The lean startup: How today's entrepreneurs use contstant innovation to create radically successful businesses, Crown Books, 2011.

[13] R. Chanin, L. Pompermaier, K. Fraga, A. Sales, R. Prikladnicki, Applying customer development for software requirements in a startup development program, in: Proceedings of the 1st International Workshop on Software Engineering for Startups, IEEE Press, 2017, pp. 2–5.

[14] U. Rafiq, S. S. Bajwa, W. Xiaofeng, I. Lunesu, Requirements elicitation techniques applied in software startups, in: 2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 30 Aug.-1 Sept. 2017, 2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA), IEEE Computer Society, 2017, pp. 141–4. `doi:10.1109/SEAA.2017.73`.
URL `http://dx.doi.org/10.1109/SEAA.2017.73`

[15] J. Bosch, H. H. Olsson, J. Björk, J. Ljungblad, The early stage software startup development model: a framework for operationalizing lean principles in software startups, in: Lean Enterprise Software and Systems, Springer, 2013, pp. 1–15.

[16] S. Blank, Why the lean start-up changes everything, Harvard business review 91 (5) (2013) 63–72.

[17] A. Nguyen-Duc, X. Wang, P. Abrahamsson, What influences the speed of prototyping? an empirical investigation of twenty software startups, in: International Conference on Agile Software Development, Springer, 2017, pp. 20–36.

[18] L. Karlsson, Dahlstedt, J. N. och Dag, B. Regnell, A. Persson, Challenges in market-driven requirements engineering-an industrial interview study, in: Eighth International Workshop on Requirements Engineering: Foundation for Software Quality, 2002.

[19] E. Carmel, Time-to-completion in software package startups, in: 1994 Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences, 1994.

[20] A. Dahlstedt, Study of current practices in market-driven requirements engineering, in: Third Conference for the Promotion of Research in IT at New Universities and University Colleges in Sweden, 2003.

[21] M. Keil, E. Carmel, Customer-developer links in software development, Communications of the ACM 38 (5) (1995) 33–44.

[22] C. Alves, S. Pereira, J. Castro, A study in market-driven requirements engineering.

[23] S. Blank, The four steps to the epiphany: successful strategies for products that win, BookBaby, 2013.

[24] S. A. Alvarez, J. B. Barney, Discovery and creation: Alternative theories of entrepreneurial action, Strategic entrepreneurship journal 1 (1-2) (2007) 11–26.

[25] S. D. Sarasvathy, Causation and effectuation: Toward a theoretical shift from economic inevitability to entrepreneurial contingency, Academy of management Review 26 (2) (2001) 243–263.

[26] A. Maurya, Running lean: iterate from plan A to a plan that works, " O'Reilly Media, Inc.", 2012.

[27] J. P. Womack, D. T. Jones, D. Roos, Machine that changed the world, Simon and Schuster, 1990.

[28] A. Nguyen-Duc, P. Abrahamsson, Minimum viable product or multiple facet product? the role of mvp in software startups, in: International Conference on Agile Software Development, Springer, 2016, pp. 118–130.

[29] S. S. Bajwa, X. Wang, A. N. Duc, P. Abrahamsson, "failures" to be celebrated: an analysis of major pivots of software startups, Empirical Software Engineering 22 (5) (2017) 2373–2408.

[30] A. Yau, C. Murphy, Is a rigorous agile methodology the best development strategy for small scale tech startups?

[31] A. I. Wasserman, Low ceremony processes for short lifecycle projects, in: Managing Software Process Evolution, Springer, 2016, pp. 1–13.

[32] M. Kuhrmann, J. Münch, I. Richardson, A. Rausch, H. Zhang, Managing Software Process Evolution: Traditional, Agile and Beyond–How to Handle Process Change, Springer, 2016.

[33] M. Tanabian, B. ZahirAzami, Building high-performance team through effective job design for an early stage software start-up, in: Engineering Management Conference, 2005. Proceedings. 2005 IEEE International, Vol. 2, IEEE, 2005, pp. 789–792.

[34] S. Chorev, A. R. Anderson, Success in israeli high-tech start-ups; critical factors and process, Technovation 26 (2) (2006) 162–174.

[35] M. Kakati, Success criteria in high-tech new ventures, Technovation 23 (5) (2003) 447–457.

[36] N. Tripathi, E. Annanpera, M. Oivo, K. Liukkunen, Exploring Processes in Small Software Companies: A Systematic Review, Vol. 609 of Communications in Computer and Information Science, 2016, pp. 150–165. doi:10.1007/978-3-319-38980-6_12.
URL <GotoISI>://WOS:000382651100012

[37] G. Coleman, R. O'Connor, Investigating software process in practice: A grounded theory perspective, Journal of Systems and Software 81 (5) (2008) 772–784.

[38] G. Coleman, R. O'Connor, Using grounded theory to understand software process improvement: A study of irish software product companies, Information and Software Technology 49 (6) (2007) 654–667.

[39] M. Kajko-Mattsson, N. Nikitina, From knowing nothing to knowing a little: Experiences gained from process improvement in a start-up company, in: International Conference on Computer Science and Software Engineering, CSSE 2008, December 12, 2008 - December 14, 2008, Vol. 2 of Proceedings - International Conference on Computer Science and Software Engineering, CSSE 2008, IEEE Computer Society, 2008, pp. 617–621. doi:10.1109/CSSE.2008.1370.
URL http://dx.doi.org/10.1109/CSSE.2008.1370

[40] B. Kitchenham, Procedures for performing systematic reviews, Keele, UK, Keele University 33 (2004) (2004) 1–26.

[41] K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, Systematic mapping studies in software engineering, in: EASE, Vol. 8, 2008, pp. 68–77.

[42] C. Wohlin, Guidelines for snowballing in systematic literature studies and a replication in software engineering, in: Proceedings of the 18th international conference on evaluation and assessment in software engineering, ACM, 2014, p. 38.

[43] F. Shull, J. Singer, D. I. Sjøberg, Guide to advanced empirical software engineering, Springer, 2007.

[44] S. S. Bajwa, X. Wang, A. N. Duc, P. Abrahamsson, How do software startups pivot? empirical results from a multiple case study, in: International Conference of Software Business, Springer, 2016, pp. 169–176.

[45] A. Nguyen-Duc, S. M. A. Shah, P. Ambrahamsson, Towards an early stage software startups evolution model, in: Software Engineering and Advanced Applications (SEAA), 2016 42th Euromicro Conference on, IEEE, 2016, pp. 120–127.

[46] A. N. Duc, P. Abrahamsson, Exploring the outsourcing relationship in software startups: A multiple case study, in: Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering, ACM, 2017, pp. 134–143.

[47] A. Nguven-Duc, Y. Dahle, M. Steinert, P. Abrahamsson, Towards understanding startup product development as effectual entrepreneurial behaviors, in: International Conference on Product-Focused Software Process Improvement, Springer, 2017, pp. 265–279.

[48] A. Nguyen-Duc, D. S. Cruzes, R. Conradi, The impact of global dispersion on coordination, team performance and software quality–a systematic literature review, Information and Software Technology 57 (2015) 277–294.

[49] M. Ivarsson, T. Gorschek, A method for evaluating rigor and industrial relevance of technology evaluations, Empirical Software Engineering 16 (3) (2011) 365–395.

[50] X. Zhou, Y. Jin, H. Zhang, S. Li, X. Huang, A map of threats to validity of systematic literature reviews in software engineering, in: Software Engineering Conference (APSEC), 2016 23rd Asia-Pacific, IEEE, 2016, pp. 153–160.

[51] C. Y. Laporte, R. V. O'Connor, Ieee, Systems and software engineering standards for very small entities: Implementation and initial results, 2014 9th International Conference on the Quality of Information and Communications Technology (QUATIC) (2014) 38–47`doi:10.1109/quatic.2014.12`.
URL `<GotoISI>://WOS:000364237700005`

[52] E. Klotins, M. Unterkalmsteiner, T. Gorschek, Software engineering in start-up companies: An analysis of 88 experience reports, Empirical Software Engineering (2018) 1–35.

[53] K. Kautz, Improvement in very small enterprisese: Does it pay off, Softw. Process Improv. Pr 226 (1988) (2000) 209–226.

[54] S. Jansen, S. Brinkkemper, I. Hunink, C. Demir, Pragmatic and opportunistic reuse in innovative start-up companies, IEEE software 25 (6).

[55] S. Shakir, J. Nørbjerg, It project management in very small software companies: A case of pakistan, in: Americas Conference on Information Systems, 2013, pp. 1–8.

[56] A. Nguyen-Duc, P. Seppanen, P. Abrahamsson, Hunter-gatherer cycle: A conceptual model of the evolution of software startups, in: International Conference on Software and Systems Process, ICSSP 2015, August 24, 2015 - August 26, 2015, Vol. 24-26-August-2015 of ACM International Conference Proceeding Series, Association for Computing Machinery, 2015, pp. 199–203. `doi:10.1145/2785592.2795368`.
URL `http://dx.doi.org/10.1145/2785592.2795368`

[57] B. J. Oates, Researching information systems and computing, Sage, 2005.

[58] M. Shaw, Writing good software engineering research papers, in: Software Engineering, 2003. Proceedings. 25th International Conference on, IEEE, 2003, pp. 726–736.

[59] H. Edison, D. Khanna, S. S. Bajwa, V. Brancaleoni, L. U. Bellettati, Towards a software tool portal to support startup process, in: 16th International Conference on Product-Focused Software Process Improvement, PROFES 2015, December 2, 2015 - December 4, 2015, Vol. 9459 of Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Springer Verlag, 2015, pp. 577–583. doi:10.1007/978-3-319-26844-6_43.
URL http://dx.doi.org/10.1007/978-3-319-26844-6_43

[60] M. Häsel, T. Kollmann, N. Breugst, It competence in internet founder teams, Business & Information Systems Engineering 2 (4) (2010) 209–217.

[61] R. Stanfill, T. Astleford, Improving entrepreneurship team performance through market feasibility analysis, early identification of technical requirements, and intellectual property support, in: Proceedings of the American Society for Engineering Education Annual Conference & Exposition, 2007.

[62] K. Kuvinka, Scrum and the single writer, Proceedings of Technical Communication Summit (2011) 18–19.

[63] S.-l. Lai, Chinese entrepreneurship in the internet age: Lessons from alibaba. com, World Academy of Science, Engineering and Technology 72 (2010) 405–411.

[64] D. B. Yoffie, M. A. Cusumano, Building a company on internet time: Lessons from netscape, California Management Review 41 (3) (1999) 8–28.

[65] L. Pompermaier, R. Chanin, A. Sales, K. Fraga, R. Prikladnicki, An empirical study on software engineering and software startups: Findings from cases in an innovation ecosystem, in: 29th International Conference on Software Engineering and Knowledge Engineering, SEKE 2017, July 5, 2017 - July 7, 2017, Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE, Knowledge Systems Institute Graduate School, 2017, pp. 48–51. doi:10.18293/SEKE2017-115.
URL http://dx.doi.org/10.18293/SEKE2017-115

[66] R. Souza, K. Malta, E. S. D. Almeida, Software engineering in startups: A single embedded case study, in: 1st IEEE/ACM International Workshop on Software Engineering for Startups, SoftStart 2017, May 21, 2017, Proceedings - 2017 IEEE/ACM 1st International Workshop on Software Engineering for Startups, SoftStart 2017, Institute of Electrical and Electronics Engineers Inc., 2017, pp. 17–23. doi:10.1109/SoftStart.2017.2.
URL http://dx.doi.org/10.1109/SoftStart.2017.2

[67] M.-L. Sánchez-Gordón, R. V. O'Connor, Understanding the gap between software process practices and actual practice in very small companies, Software Quality Journal 24 (3) (2016) 549–570.

[68] G. Marks, R. V. O'Connor, P. M. Clarke, The impact of situational context on the software development process – a case study of a highly innovative start-up organization, Vol. 770, 2017, pp. 455–466. doi:10.1007/978-3-319-67383-7_33.

[69] C. Y. Laporte, R. V. O'Connor, L. H. G. Paucar, Software engineering standards and guides for very small entities: Implementation in two start-ups, 2015, pp. 5–15.
URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-84933558276&partnerID=40&md5=02b5f237bb268c0133caa7c6cb17a44a

[70] P. Clarke, R. V. O'Connor, The situational factors that affect the software development process: Towards a comprehensive reference framework, Information and Software Technology 54 (5) (2012) 433–447.

[71] ISO, https://www.iso.org/home.html, access date: 2017-11-12 (2017). [link].
URL https://www.iso.org/home.html

[72] J. Pantiuchina, M. Mondini, D. Khanna, X. Wang, P. Abrahamsson, Are software startups applying agile practices? the state of the practice from a large survey, in: 18th International Conference on Agile Software Development, XP 2017, May 22, 2017 - May 26, 2017, Vol. 283 of Lecture Notes in Business Information Processing, Springer Verlag, 2017, pp. 167–183. doi:10.1007/978-3-319-57633-6_11.
URL http://dx.doi.org/10.1007/978-3-319-57633-6_11

[73] C. Y. Laporte, R. V. O'Connor, Implementing process improvement in very small enterprises with iso/iec 29110: A multiple case study analysis, in: Quality of Information and Communications Technology (QUATIC), 2016 10th International Conference on the, IEEE, 2016, pp. 125–130.

[74] J. Yli-Huumo, T. Rissanen, A. Maglyas, K. Smolander, L.-M. Sainio, The relationship between business model experimentation and technical debt, in: 6th International Conference on Software Business, ICSOB 2015, June 10, 2015 - June 12, 2015, Vol. 210 of Lecture Notes in Business Information Processing, Springer Verlag, 2015, pp. 17–29. doi:10.1007/978-3-319-19593-3_2.
URL http://dx.doi.org/10.1007/978-3-319-19593-3_2

[75] S. McConnell.

[76] H. Terho, S. Suonsyrja, K. Systa, The developers dilemma: Perfect product development or fast business validation?, in: 17th International Conference on Product-Focused Software Process Improvement, PROFES 2016, November 24, 2016 - November 26, 2016, Vol. 10027 LNCS of Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Springer Verlag, 2016, pp.

571–579. doi:10.1007/978-3-319-49094-6_42.
URL http://dx.doi.org/10.1007/978-3-319-49094-6_42

[77] M. Chicote, Startups and technical debt: Managing technical debt with visual thinking, in: 2017 IEEE/ACM 1st International Workshop on Software Engineering for Startups (SoftStart), 21 May 2017, 2017 IEEE/ACM 1st International Workshop on Software Engineering for Startups (SoftStart). Proceedings, IEEE Computer Society, 2017, pp. 10–11. doi:10.1109/SoftStart.2017.6.
URL http://dx.doi.org/10.1109/SoftStart.2017.6

[78] R. Deias, G. Mugheddu, O. Murru, Introducing xp in a start-up, in: Proceedings 3rd International Conference on eXtreme Programming and Agile Processes in Software Engineering (XP), 2002, pp. 62–65.

[79] V.-P. Eloranta, Towards a pattern language for software start-ups, in: 19th European Conference on Pattern Languages of Programs, EuroPLoP 2014, July 9, 2014 - July 13, 2014, Vol. 09-13-July-2014 of ACM International Conference Proceeding Series, Association for Computing Machinery, 2014. doi:10.1145/2721956.2721965.
URL http://dx.doi.org/10.1145/2721956.2721965

Appendix C

The Role of Data Analytics in Startup Companies: Exploring Challenges and Barriers

Vebjørn Berg$^{1[0000-0001-5611-964X]}$, Jørgen Birkeland$^{1[0000-0003-3444-4075]}$, Ilias O. Pappas$^{1[0000-0001-7528-3488]}$, and Letizia Jaccheri$^{1[0000-0002-5547-2270]}$

Department of Computer Science, Norwegian University of Science and Technology, Sem Sælandsvei 9, 7491 Trondheim, Norway
postmottak@idi.ntnu.no

Abstract. The advancement in technology is transforming societies into digital arenas and paves the way towards the achievement of digital transformation. With every transaction in the digital world leading to the generation of data, big data and their analytics have received major attention in various fields and different contexts, examining how they may benefit the different actors in the society. The present study aims to identify how startups that create both software and hardware products can generate value from data analytics and what challenges they face towards this direction. To this end, we performed a multiple-case study with early-stage startups and employed qualitative analysis on a dataset from 13 startups. Through semi-structured interviews, we examine how these companies use data analytics. The findings show that although the benefits from data analytics are clear, multiple barriers and challenges exist for the startups to be able to create value from them. The major ones are about their resources, including human skills, economical resources, as well as time management and privacy issues.

Keywords: Startups · Data analytics · Empirical research

1 Introduction

In the digital era of the 21st century information and knowledge becomes readily available to more and more people every day. Societies generate vast amounts of data every moment from multiple sources, transforming them into landscapes mediated by different digital media platforms, digital services, and technologies, leading to the creation of big data and business analytics ecosystems [1]. The different actors of the society (i.e., industry, public and private organizations, entrepreneurs, academia, civil society) are increasingly realizing the potential of the generated data which can lead to value creation, business change, and social change. To this end, many entrepreneurs and startups are actively trying to harness the power of big data and create software and hardware with the potential to increase value, gain a competitive advantage, and improve various aspects of human life [2].

Startups are newly created companies producing cutting-edge technology, having a major impact on the global economy [9]. In a context of extreme uncertainty and restricted economical, human, and physical resources, startups have unique challenges related to product development and innovation methods [10]. This results in a high number of failures, primarily due to self-destruction rather than competition [11][10]. Operating in fast-changing, competitive high-risk environments, continuous experimentation is essential for learning and bringing products fast-to-market [12].

There is increasing literature on how big data analytics can generate value towards business or societal transformation [3][4], however further work is needed in order to identify and overcome existing barriers that will allow practitioners to generate value from big data and analytics [5]. Digitization and big data analytics have disrupted business models and can be essential tools to reduce increasing failure rates of established companies [6]. Innovative startups profit on reduced barriers for entering markets with technologies disrupting current distribution channels, customer demands, and customer relationships [7]. Big data analytics plays a crucial role in complementing and even substituting labor for machines, especially in the context of value-creating managerial decisions [8]. Even if the barriers to entry are lowered, startups operate in a context of restricted resources and a lack of technical and managerial skills [13]. However, startups have some characteristics (e.g. ability to quickly change and scale business model) enabling them to compete with mature companies. The role and widespread of data analytics in startups is yet to be explored, even if utilization of such can be a major success factor in the ever-increasing competitive business landscapes [4].

This study focuses on how software and hardware startups can benefit from big data and seeks to identify the challenges they face which will allow them to make data-driven decisions and generate value from big data analytics. To this end, this paper will offer insight into software and hardware startup companies by answering the following research questions:

RQ1 How do startups create value from (big) data and analytics?

RQ2 What are the barriers for working with (big) data analytics in hardware startups?

To address these questions this study performs a multiple-case study investigating early-stage European startups that develop both hardware and software. Findings indicate that most startups do not utilize data analytics for various reasons. To this end, there are identified several challenges and barriers for working with data analytics in such startups, including limited data variety and difficulty of performing business experimentation.

The rest of this paper is organized as follows: Section 2 presents background literature. Section 3 explains our research method, including case selections and data analysis procedure. Section 4 presents the findings from the interviews. Section 5 discusses the results, and highlights directions for future research.

2 Background

2.1 Product Development in Startups

The primary objective of startups is to speed up the product development in the early-stages, streamlining the learning process [12]. Startups must respond to fast-changing customer needs and requests [14], both by speeding up the decision and design processes [15]. Startups typically do so by utilizing an evolutionary prototyping approach, meaning that they iteratively refine an initial prototype aiming at quickly validating the product/market fit. Customer feedback highlights new functionality and improvements. As long-term planning is infeasible in the chaotic environment of startups, flexibility and reactiveness are necessary.

Instead of utilizing repeatable and controlled processes, startups take advantage of reactive, low-precision engineering practices with focus on the productivity and freedom of their teams [16]. Startups prefer ad-hoc development approaches customized to their own needs, limiting the administrative overhead. In an experimental environment constantly compromising between speed and quality, certain agile practices might not be beneficial (e.g. regular refactoring and test-first), as excessive administrative overhead can inhibit business experimentation [17]. To bring innovative products fast forward, startups depend on team members and resources dedicated to all aspects of the development process, and to be change-oriented and self-initiated. Startups capability to enter new markets and disrupt current business models is largely associated with the uniqueness of human capital and the different approaches they employ.

2.2 The Importance of Data Analytics

In the ever-increasing digital world, businesses need to develop and evolve their (big) data analytics capabilities and competencies which are key to achieving successful digital business [4][20]. The evolution of the digital economy and its combination with (big) data analytics is challenging current business models with many startups disrupting well-established companies [21]. Big data refers to expansive collections of data (large volumes) that are updated quickly and frequently (high velocity) and that exhibit a huge range of different formats and content (wide variety) [22]. Yet, there is limited understanding of how entrepreneurs and startups need to change to embrace such technological innovations and generate value in the digital economy. Indeed, they need to build upon their main resources that include people, processes, and technology [23]. This is very important, as it allows businesses and decision-makers to respond almost instantaneously to market needs, thus increasing their operational agility. An iterative and incremental approach combined with frequent releases is essential for startups ability to quickly accommodate frequent change, and adapt prototyping to business strategy [18].

Startups and the individuals working there have the opportunity to take advantage of the available data and create new products transforming a market or an industry [3], and big data analytics may be viewed as resources in this process that enable value creation and digital transformation. Many software startups are using existing ecosystems (e.g., Apache Hadoop) to build value-added software and solutions [24]. Nonetheless, since various challenges exist in improving the value creation process, significant research is targeted on addressing these challenges taking into account engineering issues related to specifications, design, or requirements in software development [25]. However, a similar approach is not that easy to be followed by startups that develop both hardware and software. Availability of resources, as well as external and development dependencies, pose restrictions to the implementation of hardware [19], thus influencing the ability of these startups to utilize big data and analytics.

3 Research Method

To explore the research questions we performed semi-structured interviews on 13 early-stage European startups that develop hardware and software. Semi-structured interviews are considered suitable for qualitative data analysis, and allowed for a discoverable approach as interviewees could express themselves more freely and provide their own perspectives on personal experiences related to the research topics [26]. The rest of this chapter presents our research process, including case selections and the collection and analysis of data.

3.1 Case and Subjects Selection

The units of analysis are people involved in product development in startup companies that deliver products with mixed hardware and software parts. We defined selection criteria as suggested by Runeson and Höst [27]. Table 1 presents basic information about each case. The *current stage* in the table is adopted from [28], however the first stage *startup* is replaced by *concept* to avoid misunderstandings.

Table 1. Case Descriptions

Case	Product	Current Stage	Founded	Location	# of employees
Startup 1 (S1)	Smart gloves	Concept	2016	Norway	18
Startup 2 (S2)	Medtech biosensor	Concept	2017	Norway	5
Startup 3 (S3)	Physical exercise game	Stabilization	2016	Norway	5
Startup 4 (S4)	Unmanned aircraft system	Concept	2016	Norway	7
Startup 5 (S5)	Advanced noise cancellation	Concept	2017	Norway	5
Startup 6 (S6)	Medtech hydration monitoring	Concept	2016	Norway	10
Startup 7 (S7)	LPG management system	Stabilization	2016	Norway	8
Startup 8 (S8)	Cable cam system	Stabilization	2016	Norway	10
Startup 9 (S9)	Digital piggy bank	Concept	2017	Norway	4
Startup 10 (S10)	Collaborative camera	Growth	2014	Norway	50
Startup 11 (S11)	Interactive children's toy	Concept	2015	Netherlands	8
Startup 12 (S12)	3D-printer control board	Growth	2009	Norway	1
Startup 13 (S13)	Sensors for IoT	Growth	2007	Italy	25

Fig. 1. Product illustration from the investigated startups

Startups were relevant for inclusion in the study if they met the following criteria: (1) The startup develops both hardware and software parts. (2) The startup has been active for at least six months. (3) The startup has a first running prototype. (4) The startup's ambition is to scale its business. People from the relevant startups were eligible for participation if they had experience and/or knowledge about software and/or hardware development. If the candidate met the criteria, he/she was regarded as qualified for contributing to the research study.

We used five different channels to find relevant startups: (1) Innovation Center Gløshaugen, (2) NTNU Accel and FAKTRY, (3) our professional networks, (4) OsloTech and StartupLab, and (5) The Hub. Figure 1 presents examples of the products developed by the startups of this study.

3.2 Data Collection and Analysis Procedure

Data was collected using a semi-structured interview guideline between February and April 2018. Author one and two attended all interviews to avoid one single interpretation of the respondents' perspectives and insights on topics. This first-degree data collection approach allowed us to control what data was collected, ensuring that all pre-defined interview questions were answered sufficiently, and exploring new directions by asking follow-up questions [27]. All interviews were recorded and transcribed shortly afterward. Before each interview, we looked into the cases' business background, either through their company websites or other relevant incubator or accelerator websites. Additionally, participants were encouraged to answer a simple questionnaire prior to interviews filling out basic information about themselves and the company. The following list presents the main topics and interview questions of the interview guideline:

- Business background
  - Describe your product and team.
  - Name the three largest challenges you have encountered.
- Product development
  - What development process do you use?
  - How are internal/external factors influencing product development?

- Data analytics
  - How do you collect customer data?
  - Have you used data analytics for requirements elicitation?
  - What are challenges related to data analytics?

The interviews were undertaken in the language preferred by the interviewee (English or Norwegian). Several of the interviews were therefore undertaken in Norwegian as this made the interviewees more comfortable. This allowed them to express themselves more freely, and give more in-depth explanations. Because of this, it was necessary to translate some of the interviews when transcribing. As there often doesn't exist a one-to-one relationship between language and meaning [29], the translation of the transcribed interviews was ensured to "express all aspects of the meaning in a manner that is understandable" [30]. This implies that not all parts of the interviews were directly translated word-for-word.

A total of 68 pages of interview transcripts were analyzed using thematic coding analysis [31]. The transcripts were coded and analyzed using NVivo. Firstly, all authors read through the transcribed interviews to generate initial ideas. Secondly, descriptive coding was applied through an inductive coding approach to systematically identify concepts and topics of interest [33]. Related codes were combined into themes to create patterns and a meaningful whole of the unstructured codes [31]. Section 4 presents the findings from the analysis process.

3.3 Validity Procedure

The validity must be addressed for all phases of the case study to enable replication of research [27] and to ensure findings are trustworthy [31]. To ensure validity, we followed guidelines used in controlled empirical experiments in software engineering [34].

Interviewees were either CEOs or engineers with insight into business- and technical-related aspects. As the startups were mostly located in the same area, mainly consisting of young, inexperienced entrepreneurs, generalization is limited to cases with similar characteristics (i.e. early-stage European startups). To decrease the risk of biased interpretations, author one and two attended all interviews. Some interviews were in Norwegian, hence transcripts were not always verbatim to preserve the actual meaning of respondents. Recordings were transcribed shortly after each interview to mitigate bias. Since it is difficult to understand a startup and its dimensions within a time-span of 30 minutes, we collected data about the startups through incubator and company websites prior to interviews.

4 Results

4.1 Utilization of data analytics

Among the investigated startups, the usage of data analytics methods was generally limited. Operating in early stages, they were often determined to rapidly develop new features and perform customer validation. The startups in this study mostly relied on qualitative measures (e.g., interviews and observations) to obtain customer feedback. *"We have not used data analytics, and do not collect customer data."* When focusing on the short-term business goals, they minimized any effort spent on data analytics, rather focusing on the core-delivered values of their products to quickly release a minimum viable product to customers. Improving data collection measures was considered as a rather time-consuming activity. *"Data analytics is not something we currently spend time on."*

Although the startups commonly spent little time on gathering or learning from data analytics efforts, some had a clear perception of the possible business opportunities and benefits from utilization of such. Even if so, data analytics was usually outside their business scope. *"We have looked at some future possibilities of data analytics, but it is not something we currently focus on."* A brake-pad in introducing greater focus towards data analytics was that the startups in this study did not have large amounts of data at their disposal. The restricted access to useful data inhibited potential value-adding activities from data analytics. *"It's too early for us to get something valuable from data analytics."*

The capabilities of team members greatly influence the associated success of startups. From the investigations, we saw an increased focus on data analytics in startups with team members having experience or expertise within the field. Despite for the general limited use of data analytics, possessing the required knowledge and skills of such can have a positive impact on its widespread adoption within a startup organization. *"We work with data analytics and do most of it ourselves [...] It requires that your company is able to get that expertise."*

Although some of the investigated startups were aware of opportunities and benefits associated with utilizing data analytics for decision-making and requirements elicitation, they mainly focused on the core-delivered functionalities of their products to speed-up development. The findings show, that value-adding activities related to data analytics were considered as less important compared to product development activities.

4.2 Barriers for obtaining deeper customer insight

Experimentation, testing, and assessment can be a challenge to startups developing products including both software and hardware components. Physical prototypes are more resource-intensive to develop, in contrast to pure software products, thus limiting startups' ability to test products with a larger customer base. The testing ability of these startups will largely depend on their capacity (i.e., third-party dependency, financial and human resources) to produce prototypes: *"There is a great number of people who want to test our product, however, we do not have the capacity to produce enough prototypes. The main reason for this is hardware production, which happens in China, and the manual assembly we do ourselves."*

Findings from the investigated startups indicate that the amount of collected data in early stages is limited in terms of volume, velocity, and variety, as the data are generated mainly from one prototype used by a couple of users, thus restricting data capture along with their ability to generate value from them. This relates strongly to the early stages of a startup characterized by the existence of only a few customers, as well as to startups developing evolutionary independent systems. Startups may be reluctant to invest in data analytics due to the perceived limitations of the available data: *"The data amount is still a little too small to do any proper analysis of it, and we do not collect enough personal info yet to perform the analysis."*

Acquiring people with the necessary knowledge and skills in data analytics is one of the major challenges in generating value from (big) data. With startups looking for team members with knowledge in a wide area of fields (boundary-spanning knowledge), it is not easy to put a significant focus on data analytics skills and knowledge. The investigated startups had limited expertise in performing data analytics, and knowledge about available tools suited to address startups' concerns or requirements. The findings show that attracting knowledgeable people is quite hard and with resources being severely restricted, hiring specialized people only to work with data analytics is rarely an opportunity, not to mention a priority of startups: *"Finding talented people is hard. Since we are a startup we cannot give very good salary [...] If we had more money we would employ someone to analyze product and customer data [...] I see the value of it, but for the time being, it is not a priority."*

The highly competitive environment of startups and severely limited resources imply startups strict priorities. Data analytics efforts may exhaust the already constrained financial and time resources. In addition, collecting the necessary data may present an additional cost of components (e.g., sensors and IoT technology) and human investments. This may be a priority startups are not willing to take: *"At the time this [data analytics] is not something we prioritize."*

Startups work with innovative technology and products for a wide area of markets. Among the investigated startups some were developing medical products. Certain markets may pose specific restrictions and regulations for data collection. This makes the customer testing an intricate process, involving a significant amount of paperwork. Storing customer data for later analyses may be illegal or too entangled, preventing the use of data analytics. Startups need guidelines for handling privacy (e.g., General Data Protection Regulation - GDPR) and security issues to fully take advantage of the benefits of data analytics: *"When working with hospitals, data becomes more complicated due to privacy."*

The uncertain conditions and fast-changing environment of startups mean long-term planning is not part of their business model, as this is not the way they operate. Some of the investigated startups' business managers lacked the required knowledge to implement data analytics and the potential value in their business plan: *"I see data analyses as the next step for our business [...] Currently we do not even know what our data can be used for."*

5 Discussion and Conclusions

This study examines how startups can generate value by employing data analytics methods. With the majority of the literature focusing on startups that create software, here, we choose to investigate startups that develop both hardware and software. This specific category of startups presents great interest due to specific challenges that differentiate them from typical software startups. Indeed these startups are more likely to face challenges such as limited availability of resources or to be dependant on external factors linked with hardware development [19]. Such challenges are expected to affect their ability to use big data and analytics in order to generate value.

The findings show that some of the startups are aware of the potential benefits from using (big) data analytics, however, they face various barriers and challenges which limit them from utilizing them in their business models and business process. Table 2 presents the main barriers to working with (big) data analytics as identified in this study. In detail, the startups face challenges related with their prototyping capacity, as they are able to develop only limited amount of hardware prototypes, thus limiting the number of users that can use them at the same time. This is directly linked with the limited financial resources that young startups have, as well as with the time-shortage that characterizes startups, since they are forced to work on short deadlines and intensive processes.

The challenge with the limited prototyping capacity can indirectly affect data availability. In detail, limited hardware and users lead to an impact to generated data. However, such limitations could be overcome by better planning and more focused testing of their products with their end-users. Furthermore, some of the startups mention that they face specific security and privacy issues related with the use of personal data, due to the nature of their business (e.g., medical technology tested at hospitals). Nonetheless, such barriers can be overcome with the collaboration of the different actors in the society (i.e., industry, government, academia), and the recently directive from EU on data protection (i.e., GDPR) is a step towards that direction. Finally, the startups indicate that generating knowledge from data analytics is not a primary objective for them, thus it is not included in their overall business strategies. This is also linked with the other barriers, regarding prototyping capacity and resource availability, since they believe that they are not able to achieve their short-term goals using data analytics.

Some business managers mention that they possess limited knowledge on what additional value data analytics could provide to their decision-making and design process. Increasing business managers' awareness around the potential knowledge and presenting them with practical information and knowledge will increase the potential of including data analytics in their business models. This can be achieved by offering to startups validated learning, through the use of cohort metrics (e.g., actionable, accessible, and auditable metrics) and analysis. As startups are characterized by short-term planning and frequent releases, utilizing big data analytics will allow startups to make data-driven decisions,

Table 2. Barriers for working with (big) data analytics

Barrier	Description
Prototyping capacity	Physical prototypes are associated with individual development costs and time (e.g., third-party dependency).
Limitations of data	Data in early startup stages are characterized by low volume, velocity, and variety.
Team capabilities	Startups have high demands for skillful teams with entrepreneurial capabilities. Experience using data analytics will positively impact its widespread organizational adoption.
Financial resources	Hardware development includes production, manufacturing, and logistics, which require more initial human and financial investments.
Time-shortage	The uncertain high-risk environment forces startups to release their products fast and to work under constant pressure.
Security & privacy issues	Collecting customer and usage data for (big) data analytics have associated privacy and security issues.
Integration with business strategy	Data analytics activities are usually outside the short-term business goals of startups.

which can be faster and with increased quality, thus being consistent with the agile environment that most startups operate.

As with all empirical studies, this study has some limitations. Qualitative data collection measures imply that results and implications are subject to bias. To mitigate the risk of wrong interpretations, author one and two attended all interviews, preferably face-to-face on-site. Recordings were transcribed shortly afterward to preserve respondents' actual meanings. Furthermore, the study would profit from a wider collection of data, both to discover more challenges and to ensure credible conclusions. Also, employing quantitative methods would allow for data triangulation.

This study provides initial knowledge on data analytics in startups, however, future work should investigate more startups both to identify other challenges and barriers, and for generalization of results to a larger startup population (e.g., operating in different markets and lifecycle stages, and various geographical locations). Seeing that the widespread of data analytics is limited, startups need specific methods for utilizing analysis tools in early startup stages. Startup managers need guidance to understand how their data can generate revenues, and what knowledge is required for their organization to thrive from data analytics. Startups need directions for how to implement a data analytics strategy to benefit the company in the long run.

Acknowledgments

We would like to thank the startups that participated in this study.

References

1. Pappas, Ilias., Jaccheri, Letizia., Mikalef, Patrick., and Giannakos, Michail: Social Innovation And Social Entrepreneurship Through Big Data: Developing A Reseach Agenda. (2017)

2. Otero, Carlos E., Peter, Adrian: Research directions for engineering big data analytics software. IEEE Intelligent Systems, vol. 30, pp.13-19. IEEE (2015)
3. George, Gerard., Haas, Martine R., Pentland, Alex: Big data and management. Academy of management Journal, vol.57, pp.321-326. Academy of Management (2014)
4. Mikalef, Patrick., Pappas, Ilias O., Krogstie, John., Giannakos, Michail: Big data analytics capabilities: a systematic literature review and research agenda. Information Systems and e-Business Management, pp.1-32. Springer (2017)
5. Vidgen, Richard., Shaw, Sarah., Grant, David B.: Management challenges in creating value from business analytics. European Journal of Operational Research, vol.261, pp.626-639. Elsevier (2017)
6. Weill, Peter., Woerner, Stephanie L.: Thriving in an increasingly digital ecosystem. MIT Sloan Management Review, vol.56, p.27. Massachusetts Institute of Technology, Cambridge, MA (2015)
7. Lucas Jr, Henry C., Agarwal, Ritu., Clemons, Eric K., El Sawy, Omar A., Weber, Bruce: Impactful Research on Transformational Information Technology: An Opportunity to Inform New Audiences. Mis Quarterly, vol.37 (2013)
8. Loebbecke, Claudia., Picot, Arnold: Reflections on societal and business model transformation arising from digitization and big data analytics: A research agenda. The Journal of Strategic Information Systems, vol.24, pp-149-157. Elsevier (2015)
9. Unterkalmsteiner, M., Abrahamsson, P., Wang, X. F., Anh, N. D., Shah, S., Bajwa, S. S., Baltes, G. H., Conboy, K., Cullina, E., Dennehy, D., Edison, H., Fernandez-Sanchez, C., Garbajosa, J., Gorschek, T., Klotins, E., Hokkanen, L., Kon, F., Lunesu, I., Marchesi, M., Morgan, L., Oivo, M., Selig, C., Seppanen, P., Sweetman, R., Tyrvainen, P., Ungerer, C., Yague, A.: Software Startups - A Research Agenda. E-Informatica Software Engineering Journal, vol.10, pp.89-123. (2016)
10. Giardino, Carmine., Bajwa, Sohaib Shahid., Wang, Xiaofeng., Abrahamsson, Pekka: Key challenges in early-stage software startups. International Conference on Agile Software Development, pp.52-63. Springer (2015)
11. Marmer, Max., Herrmann, Bjoern Lasse., Dogrultan, Ertan., Berman, Ron., Eesley, C., Blank, S.: Startup genome report extra: Premature scaling. Startup Genome, vol.10. (2011)
12. Nguyen-Duc, Anh., Wang, Xiaofeng., Abrahamsson, Pekka: What Influences the Speed of Prototyping? An Empirical Investigation of Twenty Software Startups. International Conference on Agile Software Development, pp.20-36. Springer (2017)
13. Paternoster, N., Giardino, C., Unterkalmsteiner, M., Gorschek, T., Abrahamsson, P.: Software development in startup companies: A systematic mapping study. Information and Software Technology, vol.56, pp.1200-18. (2014)
14. Bosch, Jan: Speed, Data, and Ecosystems: The Future of Software Engineering. IEEE Software, vol.33, pp.82-88. (2016)
15. Yau, Alex., Murphy, Christian: Is a Rigorous Agile Methodology the Best Development Strategy for Small Scale Tech Startups?. (2013)
16. Tanabian, MM., ZahirAzami, B.: Building high-performance team through effective job design for an early stage software start-up. Engineering Management Conference, Proceedings, vol.2, pp.789-792. IEEE (2005)
17. Pantiuchina, Jevgenija., Mondini, Marco., Khanna, Dron., Wang, Xiaofeng., Abrahamsson, Pekka: Are software startups applying agile practices? The state of the practice from a large survey. 18th International Conference on Agile Software Development, XP 2017, vol.283, pp.167-183. Springer Verlag (2017)

18. Coleman, G., O'Connor, R. V.: An investigation into software development process formation in software start-ups. Journal of Enterprise Information Management, vol.21, pp.633-648. (2008)
19. Ronkainen, Jussi., Abrahamsson, Pekka: Software development under stringent hardware constraints: Do agile methods have a chance?. International Conference on Extreme Programming and Agile Processes in Software Engineering, pp.73-79. Springer (2003)
20. Pappas, Ilias O., Mikalef, Patrick., Giannakos, Michail N., Krogstie, John., Lekakos, George: Social media and analytics for competitive performance: a conceptual research framework. International Conference on Business Information Systems, pp.209-2018. Springer (2016)
21. Chen, Hsinchun., Chiang, Roger HL., Storey, Veda C: Business intelligence and analytics: from big data to big impact. MIS quarterly, pp.1165-1188. JSTOR (2012)
22. Davis, Charles K.: Beyond data and analysis. Communications of the ACM, vol.57, pp.39-41. ACM (2014)
23. Carlsson, Christer: Decision analytics - Key to digitalisation. Information Sciences. Elsevier (2017)
24. Tan, Wei., Blake, M Brian., Saleh, Iman., Dustdar, Schahram: Social-network-sourced big data analytics. IEEE Internet Computing, vol.17, pp.62-69. IEEE (2013)
25. Otero, Carlos E., Peter, Adrian: Research directions for engineering big data analytics software. IEEE Intelligent Systems, vol.30, pp.13-19. IEEE (2015)
26. Oates, Briony J.: Researching information systems and computing. Sage (ISBN: 1446235440) (2005)
27. Runeson, Per., Höst, Martin: Guidelines for conducting and reporting case study research in software engineering. Journal: Empirical software engineering vol.14, p.131. Springer (2009)
28. Crowne, Mark: Why software product startups fail and what to do about it. Evolution of software product development in startup companies. In: Engineering Management Conference, 2002, pp.338-343. IEMC'02. IEEE
29. Temple, Bogusia., Young, Alys: Qualitative research and translation dilemmas. Journal: Qualitative research, vol.4, pp.161-178. Sage Publications London, Thousand Oaks, CA and New Delhi (2004)
30. Larson, Mildred: Translation: theory and practice, tension and interdependence. John Benjamins Publishing (1991)
31. Cruzes, Daniela S., Dyba, Tore: Recommended steps for thematic synthesis in software engineering. Book: Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on, pp.275-284. IEEE (2011)
32. Wohlin, Claes., Höst, Martin., Henningsson, Kennet: Empirical research methods in software engineering. Book: Empirical methods and studies in software engineering, pp.7-23. Springer (2003)
33. Saldaña, Johnny: The coding manual for qualitative researchers. Sage (2015).
34. Stanfill, R., Astleford, Ted: Improving entrepreneurship team performance through market feasibility analysis, early identification of technical requirements, and intellectual property support. Proceedings of the American Society for Engineering Education Annual Conference & Exposition (2007)