Software Startup Engineering: A Systematic Mapping Study

Vebjørn Berg and Jørgen Birkeland

December 13th, 2017

TDT4501 PROJECT THESIS
Department of Computer Science, NTNU

Supervisor: Letizia Jaccheri
Co-supervisor: Ilias O. Pappas
Abstract

**Context:** Software startups are newly created companies producing cutting-edge technology in extremely dynamic environments of high uncertainty, making them unique from more established companies. Despite the stories of successful startups, 90 percent of them fail, primarily due to self-destruction rather than competition. Hence, there is a need for research to support startup activities in all lifecycle stages, as most prior research in the field of software engineering has been conducted in relation to the needs of established companies.

**Objective:** The research objective of this thesis is to provide an up to date overview of the research field. The lack of strong contributions dedicated to support startup engineering activities, and research questions from the previous mapping studies, have motivated the following research questions:

- RQ1: How has software startup research changed over time?
- RQ2: What is the relative strength of the empirical evidences reported?
- RQ3: What effort has been made to characterize the context of software engineering in startups?

**Method:** We have applied a systemic mapping method to analyze the literature related to software startup engineering. A classification schema was developed, and the primary studies were ranked according to their rigour.

**Results:** A total number of 27 primary papers from 2013-2017 were assessed and classified into the classification schema. The papers were merged and compared with 47 unique primary papers for the period 1994-2013 from two previous mapping studies. Most research has been conducted within the SWEBOK knowledge areas software engineering process, management, construction, design, and requirements. The rigour of the primary papers was assessed to be higher between 2013-2017 than that of 1994-2013.

**Conclusion:** This thesis presents the literature on software startup engineering from 2013-2017 in comparison to the literature from 1994-2013. More work is still required to support the unique software development challenges of startups. Future work should try to make up a common software startup definition to help in the creation of a coherent body of knowledge, implying that researchers must devote more attention to describe central situational aspects of the investigated startups. The knowledge gained from this mapping study will be used to design an investigation to be undertaken during spring 2018.

**KEYWORDS**
Empirical Research, Innovation, Lean Startup, Quality Assessment, Software Engineering, Software Startup, SWEBOK Knowledge Area, Systematic Mapping Study
Preface

This thesis is submitted to the Norwegian University of Science and Technology (NTNU) as part of the course TDT4501 Computer Science, Specialisation Project.

The work has been performed at the Department of Computer Science, NTNU, Trondheim, under the supervision of Professor Letizia Jaccheri as main supervisor and Post Doctoral Fellow Ilias O. Pappas as co-supervisor.
Acknowledgment

We would like to thank Professor Letizia Jaccheri for her supervision during the specialisation project. Also, we would like to thank Post Doctoral Fellow Ilias O. Pappas for his co-supervision. Both have contributed with their expertise within research in software engineering. We must not forget to send our thanks to Anh Nguyen-Duc for his support and guidance in the area of software startup engineering. IDI and NTNU deserve a special recognition for their brilliant services and facilitating for a great learning experience.

NTNU, December 13th, 2017
Vebjørn Berg and Jørgen Birkeland
# Contents

Abstract .................................................................................................................. i  
Preface ..................................................................................................................... ii  
Acknowledgment ................................................................................................... iii  
List of Figures ......................................................................................................... vii  
List of Tables .......................................................................................................... viii  
Abbreviations ......................................................................................................... ix  

1 Introduction ........................................................................................................ 1  
1.1 Motivation ....................................................................................................... 1  
1.2 Research Questions ....................................................................................... 2  
1.3 Research Process ............................................................................................ 3  
1.4 Outline of the Thesis ...................................................................................... 3  

2 Background ......................................................................................................... 4  
2.1 Closed Innovation .......................................................................................... 5  
2.2 Open Innovation ............................................................................................. 7  
2.3 Startups ........................................................................................................... 10  
2.4 Lean Startup ................................................................................................... 11  
2.5 Software Engineering ...................................................................................... 13  
   2.5.1 Software Engineering in Startups ............................................................ 15  

3 Research Method ................................................................................................ 18  
3.1 Mapping Procedure ....................................................................................... 19  
3.2 Research Questions ....................................................................................... 20  
3.3 Data Sources and Search Strategy ............................................................... 20  
3.4 Study Selection ............................................................................................... 21  
3.5 Manual Search ............................................................................................... 22  
3.6 Data Extraction .............................................................................................. 23  
3.7 Quality Assessment ....................................................................................... 23  
3.8 Data Synthesis ............................................................................................... 24
### Results

4.1 RQ1: How has software startup research changed over time?

4.1.1 Publication Frequency

4.1.2 Knowledge Areas

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

4.2.1 Rigour of Primary Studies 2013-2017

4.2.2 Rigour of Primary Studies 1994-2017

4.3 RQ3: What effort has been made to characterize the context of software engineering in startups?

4.3.1 Themes and Term Frequency

4.3.2 Incubated Companies

4.3.3 Contextual Descriptions

### Discussion

5.1 RQ1: How has software startup research changed over time?

5.1.1 SWEBOK Knowledge Areas

5.1.2 Startup Research 1994-2017

5.1.3 Future Classification

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

5.3 RQ3: What effort has been made to characterize the context of software engineering in startups?

### Planning of the Investigation

6.1 Mapping Study Findings and Observations

6.2 Research Questions and Problem Areas

6.3 Research Plan and Approach

6.3.1 Research Process

6.3.2 Participants

6.3.3 Paradigm

6.3.4 Deliverables

### Conclusion

7.1 RQ1: How has software startup research changed over time?

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

7.3 RQ3: What effort has been made to characterize the context of software engineering in startups?

7.4 Implications and Future Work
# CONTENTS

A  **Classification Schema**  65  
   A.1  Research Methods  67  
   A.2  Contribution Types  68  

B  **Quality Assessment**  69  

**Bibliography**  71
List of Figures

2.1 Background Overview .................................................. 5
2.2 The Closed Innovation Process (Chesbrough, 2006) ................ 6
2.3 The Open Innovation Process (Chesbrough et al., 2014) .......... 8

3.1 The Systematic Mapping Study Process (Petersen et al., 2008) ... 19
3.2 The Study Selection Process ............................................. 22

4.1 Publication Frequency, 2013-2017 .................................... 28
4.2 Publication Frequency, 1994-2017 .................................... 29
4.4 Contribution Types, 2013-2017 ........................................ 31
4.5 Knowledge area coverage, this study and Klotins et al. (2015) ... 32
4.6 Contribution types, this study and Paternoster et al. (2014) ....... 33
4.7 Rigour of each covered knowledge area, 2013-2017 ............... 34
4.8 Rigour and Research Type (Klotins et al., 2015) .................. 35
4.9 Rigour and Contribution Type (Paternoster et al., 2014) .......... 35
4.10 Percentage of Incubated Companies, 2013-2017 ................. 40

6.1 The Research Process (Oates, 2005) .................................. 60
**List of Tables**

3.1 The Searched Databases and Number of Retrievals ........................................... 21  
3.2 Quality Assessment Checklist (Nguyen-Duc et al., 2015a) ................................. 24  
4.1 Theme Descriptions, 2013-2017. ................................................................. 37  
4.2 Theme Descriptions, 1994-2013 (Paternoster et al., 2014). ............................. 38  
4.3 Term Frequencies, 2013-2017. ................................................................. 39  
4.4 Term Frequencies, 1994-2013 (Paternoster et al., 2014). ............................. 39  
4.5 Contextual Descriptions, 2013-2017 ............................................................. 41  
A.1 Classification Schema ....................................................................................... 66  
A.2 Research Methods (Oates, 2005) .................................................................. 67  
A.3 Contribution Types (Paternoster et al., 2014) ............................................... 68  
B.2 Explanation of themes, based on table 6 in Paternoster et al. (2014). ............... 70
Abbreviations

BML  Build-measure-learn
ISO  International Organisation of Standardization
IP   Intellectual Property
MVP  Minimal Viable Product
R&D  Resource and Development
SME  Small and Medium-sized Enterprises
SMS  Systematic Mapping Study
SW   Software
SWE  Software Engineering
SWEBOK Software Engineering Book of Knowledge
VSE  Very Small Entities
Chapter 1

Introduction

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" (Giardino et al., 2016). Software startups present a unique combination of characteristics which pose several challenges to traditional software development and innovation methods (Giardino et al., 2015). This thesis aims at providing an up to date overview of the research within software startup engineering, structuring and analyzing the literature between 2013-2017.

The introduction consists of five subsections. Section 1.1 presents the motivation and research objective of this research. Section 1.2 presents the research questions. Section 1.3 explains the chosen research method and the research process. Section 1.4 presents the outline of this thesis.

1.1 Motivation

Software startups are newly created companies producing cutting-edge technology in extremely dynamic environments of high uncertainty. Without any significant operating history, they are unique from more established companies (Giardino et al., 2015). Despite stories of successful startups, 90 percent of them fail, primarily due to self-destruction rather than competition (Marmer et al., 2011). Hence, there is a need for research to support startup activities in all lifecycle stages. Previously, most of the research in the field of software engineering has been conducted in relation to the needs and challenges of established companies. This gap was first identified by Sutton Jr (2000).

The importance and influence of startups in today's digital business innovation is increasing, illustrated by a shift from a closed to a more open, collaborative innovation paradigm. Established companies utilize external resources to a larger extent. In five years, the collaboration
with startups is expected to contribute with 20 percent to the total digital revenue of large companies, and create 10 million new youth jobs (Accenture, 2015). The cost of launching a new technology startup is lower than ever, and with increasingly more people connected to the Internet through mobile devices, it is possible to reach customers and new markets at a rapid pace. Startup companies’ influence on people’s lives is more significant than ever before (Marmer et al., 2011; EY and Cisco, 2016).

Two mapping studies have previously been conducted within the field of software engineering for startup companies, identifying publications from 1994 to 2013 (Paternoster et al., 2014; Klotins et al., 2015).

1. Paternoster et al. (2014) assessed a total of 43 primary papers, all providing empirical evidence. The study extracted recurrent themes that characterize the startup context, and identified the transferability of results to industry through quality assessment of all primary papers. It also identified work practices in startups.

2. Klotins et al. (2015) assessed 14 primary papers, however 10 of these papers were also covered in Paternoster et al. (2014). The study focused on the primary papers’ coverage of the SWEBOK knowledge areas, and whether the primary papers provided sufficient rigour and industry relevance.

Although both studies covered papers up to and including 2013, no more than three papers were included from 2013. Both primary studies concluded that there was a lack of high-quality papers supporting software engineering activities in any phase of the startup lifecycle, and so transfer of results to other startups was difficult. Hence, there exist a need for an up to date overview of the available research, to identify research quality and gaps, and suggest potential research topics for future research within the field.

1.2 Research Questions

The research objective of this thesis is to identify the latest research within software engineering specific to the startup context. The lack of strong contributions dedicated to support startup engineering activities, and research questions from the previous mapping studies, have motivated the following research questions:

1. RQ1: How has software startup research changed over time?

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

3. RQ3: What effort has been made to characterize the context of software engineering in startups?
1.3 Research Process

To address the research objective and the research questions, we will undertake a systematic mapping study (Petersen et al., 2008). Due to the time-constraints of the research project, the newness of the research topic, and the fact that both authors are new to the research area, a systematic mapping is regarded as a more suitable approach than a systematic literature review. The systematic mapping will follow guidelines from Kitchenham (2004), and several steps of the standardized process for systematic mapping studies (Petersen et al., 2008). The main steps of the process includes the research questions to be answered, search and study selection strategies, manual search, data extraction, quality assessment, and the method for synthesizing data.

Based on the search string (section 3.3), a total number of 1012 unduplicated papers were retrieved. This was further limited to 74 (based on titles), 28 (based on abstracts), and finally 20 papers after a collaborate effort from both authors was conducted. An additional manual search, using the forward snowballing technique (Wohlin, 2014), resulted in a total number of 27 primary papers. The full-text was read of all 27 papers, and relevant information was extracted into a classification schema (table A.1). Quality assessment (table B.1) was performed on all papers providing empirical evidence. Lastly, a thematic analysis (section 4.1.2) was conducted to allow for comparison of our findings to that of the previous mapping studies.

1.4 Outline of the Thesis

The rest of the thesis proceeds as follows. Section 2 introduces the background of the study, and the related work. The section explains how startups have become so influential in light of a shift to a more open, collaborative innovation paradigm, and some of the characteristics of startups from a software engineering perspective. Section 3 presents the research method undertaken, and threats to the validity of the mapping. Section 4 reports the results of the mapping study, 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 presents future work and the research plan of an in-depth investigation that will be conducted as part of a Master thesis during spring 2018. Section 7 concludes the paper by answering the research questions. Implications and future work are also presented.
Chapter 2

Background

The background consists of five subsections, illustrated by figure 2.1. These present how startups have become so influential, and why there is a need for more research supporting startups’ engineering activities, which are unique to those of established companies. Section 2.1 introduces the topic Closed Innovation, and illustrates how established companies went through a transformation during the mid 1980’s. The transformation lead to a breakage of the Closed Innovation paradigm, where established companies opened up their innovation processes, allowing for collaboration with external actors in the development of new products and services. This shift of strategy is referred to as the Open Innovation paradigm, which is introduced in section 2.2. The Open Innovation paradigm is one of the main reasons why startups have become so important in today’s innovation processes. Section 2.3 further presents software startups and characteristics of the startup context, which introduces several unique challenges. A method startups can follow to deal with some of the challenges posed by the startup context, is Lean Startup. Section 2.4 presents the Lean Startup method, and how this method poses several issues in the context of software engineering in startups. Section 2.5 presents the topic software engineering, with a particular focus on different stages of software development and agile methodologies. The section is further divided into software engineering in the context of startups (section 2.5.1). Finally, section 2.5.1.1 presents related work in the field of software startup engineering.

Figure 2.1 illustrates the main topics of the background section, and how these relate. Startups are part of both the Closed and Open Innovation paradigms, but in different ways and extent for each of them. Software startups have special challenges and needs in terms of software engineering practices, and use alternative management methods like Lean Startup to deal with the challenges posed by the startup context.
CHAPTER 2. BACKGROUND

2.1 Closed Innovation

Closed Innovation can be defined as “a process where a company’s intellectual property (IP) is developed and kept within the company boundaries until a new product or service is released in the market” (Chesbrough, 2006). In terms of software, IP refers to everything encompassing patents, copyrights, trade secrets, and trademarks, and IP rights can be used to protect the technology or new software programs from competitors (Chesbrough, 2006). In Closed Innovation, companies have their own department of research and development (R&D) to completely control the development of new products and services.

Among the different research projects within a company’s internal R&D department, only some are in line with the company’s current business model, as illustrated by the “firm boundaries” in figure 2.2. The business model’s role is to create an internal logic for how value is created and maintained by connecting a company’s technical inputs to its economic outputs (Chesbrough, 2006). The projects not in line with the business model are often rejected and turn into external spin-offs, as the company doesn’t see further potential for the projects within its boundaries. The Closed Innovation approach is good at eliminating projects that initially look good, but later turn out to be disappointing, or so-called false positives. To eliminate false positives, companies need to terminate projects that show any signs of potential failure. However, this approach might lead to many false negatives, which means that the terminated project turns out to be a success in another market through a spin-off with another business model.
Between the end of World War II and the mid 1980s, most R&D processes followed the principles of Closed Innovation. The companies applying the Closed Innovation approach consisted of deep, vertically integrated R&D processes; these companies discovered new technological breakthroughs, developed and built them into products or services in their factories, and managed the distribution, financing, and service of these; all on their own (Chesbrough, 2006).

For certain markets and products, the Closed Innovation process was and still is efficient. Examples of industries dependent on Closed Innovation practices are nuclear reactors, mainframe computers, and aircraft engines. Common to these industries is that they:

- Rely on internal ideas
- Have low labor mobility
- Have little venture capital
- Consist of few and weak startups
- Aren't dependent on universities

Closed Innovation encourages companies to be self-reliant, because one cannot be sure of others' ability to deliver quality on a timely basis. Since the intellectual property arising from internal R&D is strongly guarded, others cannot exploit these ideas for their own profit (Chesbrough, 2006). Successful commercialization of a new technology involves the management of both technical and market uncertainty. Until you know the most valuable uses of a technology and which markets to target, you do not know how to manage and where to focus the development activity. By making an initial product, you can learn what customers want and adjust plans as more information becomes available. Established companies find such experimentation hard.
to do. They tend to have well-developed processes for their current business - not for new, uncertain markets that eventually lead to new businesses (Chesbrough, 2006).

The Closed Innovation approach has resulted in a wide range of new discoveries and technological breakthroughs (Chesbrough, 2006). It is effective for improving existing products and services, or even creating adjacent ones (EY and Cisco, 2016). On the other hand, it is often unsuccessful at creating genuine breakthroughs. The best ideas don’t necessarily come from within, as the external pools of creative people is way larger than internal ones. In the mid 1980’s, the Closed Innovation principles became fundamentally obsolete as the landscape of knowledge was rearranged (Chesbrough, 2006). This rearrangement was mainly caused by the following factors:

- Increased availability and mobility of skilled people.
- The venture capital market.
- External options, like external startups in new markets for ideas sitting on the shelf.
- The increased capability of external suppliers.

These factors led to a breakage of the Closed Innovation approach. Venture capital was a process for creating companies to commercialize new discoveries through fast decision-making and actions (Chesbrough, 2006). The ventures estimated market and customer needs with real products, taking advantage of third party technologies to compensate for their own shallow technology. Promising, internal R&D projects of established companies that got rejected found value in new markets through external spin-offs and startups. Skilled workers were allowed to pursue these projects outside the established companies, independent of the internal bureaucracy and slow, vertically integrated processes (Chesbrough, 2006). To stay competitive, the established companies had to find new ways to develop and commercialize new products and services through collaboration with innovative startups (EY and Cisco, 2016), to strategically manage the knowledge sharing, uncertainty, and ambiguity of these changed landscapes (Accenture, 2016).

### 2.2 Open Innovation

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. This puts innovation at the front for most organizations, challenging them to develop new products and services faster (EY and
Companies that expect their technology not to change and don’t innovate, will be disrupted. The strong competition can be seen on the estimated life expectancy for companies. For companies on the S&P 500 Index, this has dropped to an average of 15 years, which implies that three quarters of the index in 2020 will be companies we have not heard of yet (Gittleson, 2012).

This new situation and fast changing environment requires a new approach to innovation. The term “Open Innovation” was introduced by Henry Chesbrough in 2003 and is an approach to innovation where Closed Innovation is no longer sustainable. This new paradigm “assumes that firms can and should use external ideas as well as internal ideas, and internal and external paths to market, as the firms look to advance their technology” (Chesbrough, 2006). The process facilitates collaboration with external partners to solve problems by exploiting ideas, people, and knowledge from inside and outside the organization to create value (EY and Cisco, 2016).

The Open Innovation process can be divided into stages, as illustrated in figure 2.3. The first stage represents the research process of the organization. This is where the company tries to develop as many ideas as possible around interesting areas for the company. The ideas can come from internal research processes or by harnessing the great amount of knowledge from people outside of the company to access potential interesting ideas. The porous boundary of the company means that ideas can exit or enter at any stage in the process. During the development stage of the process, ideas are refined and tested. Ideas that initially don’t seem interesting in the current market of the company, exit in the form of startups with the organization’s own
employees trying out the idea for a different market. These are called false negatives, and might turn out to be surprisingly valuable and used by the company in the future. Letting ideas flow in and out of the company and the use of approaches like Lean Startup at this stage gets the ideas faster to market, which leads to faster feedback, and hence learning for the company. Fast learning will be critical to the success of the company and important to outperform other organizations. The ideas that fit the business model of the company follow the internal paths of the organization which in the same way as Closed Innovation eliminates false positives. These are projects that initially seem promising, but later turn out to be of little or no value (Blosch and Burton, 2016). The last stage of the process can be referred to as the transfer stage or growth phase. At this stage, the ideas have been developed and tested against the market, and it is time to launch the idea. This implies, among other things, manufacturing, marketing and setting up supporting functionality (EY and Cisco, 2016).

Collaboration between startups, crowds, or established companies is an important part of the Open Innovation process. As suggested by Trott and Hartmann (2009), actors can collaborate through licensing, supplier relations, outsourcing, joint or non-joint ventures, R&D consortia, industry clusters, or innovation networks. Important reasons for collaboration include:

- Accessing capital, marketing, new technologies, or management skills
- Shared risk and liability
- Better relationship with strategic partners
- Reduced R&D costs

The degree of collaboration between actors is dependent on corporate needs and market forces. Research differs between four main categories of Open Innovation, based on the degree of collaboration between the actors (Accenture, 2015):

- Corporate ventures: They are efficient for de-risking financial bets on internal R&D through external investments and scouting for next-generation technologies.

- Incubators/accelerators: They are becoming increasingly important, as large companies seek to make startups more influential to their business. Today, incubators/accelerators are used by one third of large companies (Accenture, 2015).

- Joint innovation: This enables the participants to collaborate more broadly towards maximizing market opportunities more effectively than they could have independently. Startups are expected to contribute their ideas within large companies’ environments to help solve their problems and needs.
**Ecosystem innovation:** The creation of an ecosystem of partners to jointly develop new technologies or solutions and integrate these, e.g. through a digital platform. It enables companies to bring in external ideas more quickly to create shared value at the intersection of corporate performance and society to solve big problems, and serves as the ultimate destination for Open Innovation in maximizing the value of collaboration (Accenture, 2015).

Collaboration, especially with startups, is expected to be a crucial way to increase digital revenues. The proportion of revenues generated today by innovation collaboration with startups and entrepreneurs already represents a significant 9 percent of large companies’ total revenues. In five years, the proportion is expected to increase to 20 percent as collaboration accelerates (Accenture, 2015).

### 2.3 Startups

A startup can be defined as “an organisation that is challenged by youth and immaturity, with extremely limited resources, multiple influences, and dynamic technologies and markets” (Sutton Jr, 2000). More specifically, Coleman and O’Connor (2008) describes software startups as “unique companies that develop software through various processes and without a prescriptive methodology”, while 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., 2014; Ries, 2011).

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 (Unterkalmsteiner et al., 2016). While established companies focus on optimizing an existing business model, startups focus on finding one, which might require experimentations of different products in different markets (Ries, 2011).

Over the past ten years, there has been a significant rise in the number of new software startup companies (EY and Cisco, 2016). This is partly due to the following reasons (Marmer et al., 2011):

- The cost of launching a technology startup has decreased.
- Increasingly more people own mobile, digital devices and have Internet connection.
- It is easier to access potential customers and new markets.
The increasing importance of startups is highlighted by the Open Innovation paradigm, which illustrates how established companies can make use of startups in their innovation processes (Chesbrough, 2006), e.g. through business incubators, which has become an increasingly popular approach in today’s innovation communities.

Inspired by successful startups, many new software companies are created every day (Paternoster et al., 2014). Although the number of new startups is increasing, Marmer et al. (2011) found that 90 percent of them fail, primarily due to self-destruction rather than competition. Although market factors and financial issues are important reasons why startups fail before realizing any significant achievements (Crowne, 2002), the impact of inadequate use of software engineering practices might be a significant factor leading to the high failure rates (Klotins et al., 2015). As time and resources are extremely scarce in environments of high market and technology uncertainty, software startups need effective practices to face with those unique challenges (Giardino et al., 2014).

Planning and forecasting are only accurate when based on a long, stable operating history and static environments. As startups have neither, old management methods aren’t necessarily sustainable for their survival and further business growth. Startups need a strategy to achieve its vision. The strategy includes a business model, a product roadmap, a point of view about partners and competitors, and a target market, where a product or service is the end result of the strategy (Ries, 2011).

Instead of developing software for a specific client, software startups are developing products in high-potential target markets (Blank, 2013). This relates to market-driven software development (Alves et al., 2006), which emphasizes the importance of time-to-market as a key strategic objective (Sawyer et al., 1999). In a market-driven context, requirements tend to be (1) invented by the software company (Potts, 1995), rarely documented (Karlsson et al., 2002), and (3) validated only after the product is released in the market (Dahlstedt, 2003; Keil and Carmel, 1995; Carmel, 1994). As to this, products that don’t meet customer needs are common, resulting in failure of new product releases (Alves et al., 2006). The Lean Startup methodology is a method startups can use to develop products that are fit-to-market and that meet actual customer needs (Ries, 2011).

### 2.4 Lean Startup

The Lean Startup 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 busi-
ness, where the goal is to grow the business with maximum acceleration (Ries, 2011). 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 experimentations on customers to test hypotheses. The two most important hypotheses startup companies make, are the so-called leap-of-faith assumptions:

- A value hypothesis to test whether a product really delivers customer-value.
- A growth hypothesis to test how new customers will discover the product.

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.

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 what customers want. To prove that you actually are learning, it is important to use the right metrics to answer the hypothesis. This is referred to as innovation accounting (Ries, 2011). It is beneficial to use cohort-based metrics. They are actionable, accessible, and auditable, and are suitable for measuring hypotheses as they give concrete and tangible cause-and-effect relations through customer-feedback.

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). 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 aim of every startup is to build the right product to find out whether the product or service is fit to market. The Lean Startup method only works if you are able to build an organisation as adaptable and fast as the challenges it faces (Ries, 2011). This is also one of the limitations of the Lean Startup method when applied in a software startup development context. Validated learning and MVPs emphasize the importance of getting the product to customers as soon as
possible, but the BML feedback loop is a continuous process - you don't stop after creating one MVP. Shortcuts taken in product quality, design, or infrastructure today might slow down a company tomorrow. A low-quality product can inhibit learning, and prevent customers from experiencing the product's benefits (Ries, 2011). When you're going too fast, you cause more problems.

The previous paragraph illustrated an example of what we refer to as technical debt, which was originally introduced by (Cunningham, 1992) as the possible cost of choosing an easy solution now, instead of a better approach which might lead to additional re-work at a later stage. 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”. Technical debt can hurt companies and have important consequences (Chicote, 2017). The Lean Startup method is great when dealing with general business and product development processes and principles that are well-suited for software projects. However, it does not address engineering practices and issues particular to software development (Yau and Murphy, 2013), like technical debt.

### 2.5 Software Engineering

Software engineering is an engineering discipline that covers all aspects of software production (Sommerville, 2011). The term was first introduced in 1968, as individual program development approaches did not scale up to large, complex systems that tended to be unreliable, over budget, and delivered late. This led the way for the development of a variety of new software engineering techniques and methods during the 1970s and 1980s, including tools and object-oriented programming.

To characterise the software engineering discipline, 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 characterise the practice of software engineering.

Well developed software products are characterized by (1) maintainability, which means that software can evolve to meet changing customer needs, (2) dependability and security, which includes reliability, security and safety, (3) efficiency, which includes responsiveness, processing time, and memory utilization, and (4) acceptability, which means that software must be accepted by the intended users. To meet these criteria, software development should include the
following stages of software engineering:

1. Software specification. End-users and engineers determine and define what software that is going to be developed, including all constraints of the development process.

2. Software development. The stage where the software is actually designed and developed.

3. Software validation. Involves to check and ensure that the software meets customer requirements through testing.

4. Software evolution. Software is modified to meet changed customer- and market requirements.

Together, these stages are referred to as a software process, leading to the production of a software product (Sommerville, 2011). Different software products require different software engineering methods and techniques, but common to all is that the development must be managed and follow a planned development process to meet issues related to dependability, security, requirements, and reuse.

Software development is generally divided into two main approaches. These are plan-driven methodologies and agile methodologies. Whether to use an agile or plan-driven approach depends on the type of software being developed, the capabilities of the development team, and the culture of the developing company (Sommerville, 2011).

Agile methods have become more and more influential in the development processes of most software production. Important features of the method include (1) rapid development, (2) frequent releases, (3) reducing process overhead, and (4) producing high-quality code, and (5) that customers are directly involved in the process. These fundamentals are part of the agile manifesto (Cunningham et al., 2001), which was written by practitioners in 2001. There exist several agile methods, including lean software development, scrum, and extreme programming.

Agile methods have been criticized by several practitioners and academics because, among other things, they mean it is mainly suitable for small companies (Cohen et al., 2004) and that there exist little scientific support for such methods (McBreen and Foreword By-Beck, 2002). A systematic review of the research within the field of agile methodologies was performed by Dybå and Dingsøyr (2008). They divided the studies into four thematic groups, so that readers from industry could investigate relevant studies further and compare the settings to their own situation. They further concluded that there was a need for future research to expand the number and quality of studies on agile software development.
2.5.1 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" (Giardino et al., 2016). The management of software development can be achieved through software processes, as explained in section 2.5. The degree of process in software development is dependent on system complexity, business risk, and the number of people involved (Wasserman, 2016). The need for process depends on the lifecycle stage of the company. According to Crowne (2002), the startup lifecycle can be divided into four stages.

- 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 (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, 2008).

It is important to notice that in terms of communication and cooperation dynamics, startups and established companies have different software engineering experiences and needs (Yau and Murphy, 2013). While established companies have well-defined processes for their business, startups usually have low-ceremony processes (Kuhrmann et al., 2016), 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 focus on the productivity and freedom of their teams (Tanabian and ZahirAzami, 2005; Chorev and Anderson, 2006; Kakati, 2003).

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 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 appears, and more users and complex code require extended focus on robustness, scalability, performance, and power consumption (Wasserman, 2016). The usage 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 (Yau and Murphy, 2013).

A general lack of studies in the area of software engineering in startups was first noted by Sutton Jr (2000), where he claimed that “software startups represent a segment that has been mostly neglected in process studies”. Further evidence for this observation is provided by Coleman and O’Connor (2008) and Giardino et al. (2014). Although research exists on the challenges software startups face, there is no study dedicated to their success factors (Unterkalmsteiner et al., 2016). According to Giardino et al. (2016), several models have been introduced to drive software development activities in startups, however without delivering significant benefits (Sutton Jr, 2000; Coleman and O’Connor, 2008).

### 2.5.1.1 Related Work Software Startup Engineering

Since the gap in research specific to software engineering in startups first was identified (Sutton Jr, 2000), there have only been undertaken two mapping studies entirely dedicated to the research area (Paternoster et al., 2014; Klotins et al., 2015). One systematic review on software processes in small software companies was also found, which to some degree relate to the startup context (Tripathi et al., 2016). As discovered in the previous mappings, and in the search process of this mapping, there also exist relevant work related to SMEs (small and medium-sized enterprises), and VSEs (very small entities), which is outside the startup context. These type of businesses share some of the same characteristics as startups, however the early lifecycle stages of startups pose some specific challenges and needs (e.g. little working/operating history). As startups enter new, more mature lifecycle stages, studies related to SMEs and VSEs become more relevant, hence they will not be discussed any further in this study.

The first systematic mapping by Paternoster et al. (2014) covered studies up to December 2013. The mapping study aimed at structuring and analyzing the state-of-art on software startups research. The conclusion of the paper was that there exist 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 (Coleman and O’Connor, 2008; Coleman and O’Connor, 2008, 2007; Kajko-Mattsson and Nikitina, 2008) 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. (2015) conducted a mapping study classifying all studies into the SWEBOK knowledge areas. The paper was published in 2015, and mapped studies from 1994 to 2013. The paper concluded, as Paternoster et al. (2014), that existing research does not provide support for any challenges or engineering practices in startups, and that available research results are hard to transfer between startups due to low rigour. This is explained by the lack of contextual information in the studies, and how the studies were performed.

The systematic review Tripathi et al. (2016) focused on companies of up to 50 people, but was undertaken recognizing the lack of systematic reviews for small software companies and startups. The review analyzed papers published between 2004-2014. The results of the review were challenges to the software processes, and the related experience of small companies. The results showed, among other things, the absence of process standards, documentation, and the burden of quality assurance and testing.

Common to the systematic mappings and the review, is the recognition of how few papers that address the different areas within software startup engineering. To date, there exists no foundation to support any software activities in startups. All papers recognized the need for more empirical research in the startup context to address the identified gap, and to understand how software engineering is performed in these companies. The papers showed an inconsistent use of terms describing the startup context, identifying the need for a coherent body of knowledge.
Chapter 3

Research Method

A systematic mapping study will be undertaken to provide an overview of the research available in the field of software engineering specific to startups. Systematic mapping studies are good in research areas with few relevant primary studies of high quality, as it provides a coarse-grained overview of the publications within the topic area (Petersen et al., 2008). The previous systematic mapping studies (Paternoster et al., 2014; Klotins et al., 2015) covered research up to and including 2013, but only three papers were included from 2013. Hence, to assure that all relevant publications from 2013 are covered, this mapping study will focus on research from 2013 up to October 2017. This approach will allow for the results of this systematic mapping study to be merged and compared with the previous mapping studies (Paternoster et al., 2014; Klotins et al., 2015), to provide an up to date overview of the research field.

Due to the time-constraints of this research, the newness of the research topic, and the fact that both authors are new to the research area, a systematic mapping is a more suitable approach than a systematic literature review. Systematic reviews can allow for more general conclusions, but they require a major effort and in-depth analyses. Systematic mappings require less effort, but at the same time, they can provide an overview of a research area, and allow for identification and quantification of research and results. They have also proven to be more presentable to industrial software engineers, due to its visual appeal (Petersen et al., 2008).

This systematic mapping study will follow guidelines from Kitchenham (2004), and several steps of the standardized process for systematic mapping studies, as illustrated in figure 3.1 (Petersen et al., 2008). The main steps of our process are explained in section 3.1, and includes the research questions, search and study selection strategies, manual search, data extraction, quality assessment, and the data synthesis method. The process lead to a total number of 27 primary papers, which can be found in the classification schema in Appendix A.
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 data extraction form.

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 both authors was conducted. The full-text was read of the remaining 20 papers.

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 (Wohlin, 2014). For the forward snowballing, Google Scholar was used to examine the citations of the papers retrieved. This resulted in 7 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 and quality 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.1.

Step 5: Data analysis
From the primary papers, the relevant data and information were extracted into the classification schema, and a thematic analysis was conducted to allow for comparison of our findings to that of the previous mapping studies. This analysis was used to visualize relevant data in the
3.2 Research Questions

Both of the previous mapping studies recognized a lack of high-quality papers addressing software engineering in startups, and highlighted a need for more empirical evidence. The newness and growing interest in the research field, argues the need for a new mapping study to identify the focus and quality of the last five years of research. The lack of reliable support for software engineering activities in any stages of the startup’s lifecycle, and research questions from the previous mapping studies, have motivated the following research questions:

1. RQ1: How has software startup research changed over time?
2. RQ2: What is the relative strength of the empirical evidences reported?
3. RQ3: What effort has been made to characterize the context of software engineering in startups?

To provide an up to date overview of the research results within the field, the results will be merged and compared to the previous mapping studies. To address RQ1, the papers will be structured according to the knowledge areas identified in SWEBOK (Bourque and Fairley, 2014). As SWEBOK is used as the focus facet of Klotins et al. (2015) this will allow for easy comparisons, and make it possible to identify changes in terms of research direction for the last five years. With RQ2, we intend to evaluate the papers’ rigour to compare the quality of papers published before and after 2013. Finally, with RQ3, we will examine to what extent the retrieved papers provide sufficient contextual descriptions, and if there are similarities in the use of terms describing the startup context between the papers.

3.3 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 use in the software engineering community. To obtain high quality data, the following databases were used.
Initial searches in the databases were conducted to identify keywords related to software engineering and startups. These initial searches discovered the inconsistent use of terms for “startup” in the research community, also presented in Paternoster et al. (2014). For that reason, the most frequently used keywords for “startup” were chosen and combined in the search string. The final search string consisted of several search terms combined using the Boolean operator “OR”.

- (startups OR start-up OR startup) AND software engineering
- (startups OR start-up OR startup) AND software development
- (startups OR start-up OR startup) AND software AND agile
- (startups OR start-up OR startup) AND software process
- (startups OR start-up OR startup) AND software tools

### 3.4 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. These were imported into EndNote X8.
The retrieved papers were examined by both authors, where each author separately reviewed the papers based on titles and abstracts. Studies were relevant for inclusion if they presented contributions to the body of knowledge in the field of software engineering in the startup context, and if they were not included in any of the previous mapping studies (Paternoster et al., 2014; Klotins et al., 2015). Studies written in English (from 2013 up to an including October 2017) by students, researchers, and professional software developers were included. To decrease the number of papers into a manageable amount, workshops, conference proceedings, “lessons learned” papers without a clear research question, and papers based on expert opinion were excluded from the review process.

When both authors had reviewed papers separately, the number of similar papers was 21 out of 28 (75 percent). A collaborate effort was then undertaken to determine which studies to include for the data extraction and quality assessment phase. The final number of studies was 20, which were selected as primary studies.

### 3.5 Manual Search

Following the systematic 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 pa-
pers were included in the primary studies (Bajwa et al., 2016; Nguyen-Duc and Abrahamsson, 2016; Nguyen-Duc et al., 2016; Duc and Abrahamsson, 2017; Bajwa et al., 2017; Nguyen-Duc et al., 2017; Nguyen-Duc et al., 2017), making the total number of primary studies 27. Nearly 80 percent of these were conference papers, while the rest either were journal papers or part of scientific books.

### 3.6 Data Extraction

After the manual search, ending up with the resulting 27 primary studies, we defined the classification schema in Appendix A. Each of the primary studies were then systematically classified into the classification schema, according to the predetermined attributes. The chosen attributes were inspired by previous mapping studies discovered during the data collection process (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).

- Knowledge area
- Research method
- Contribution type
- Pertinence
- Term for startup
- Research type
- Incubator context
- Publisher

In addition to classifying each paper into the classification schema, each paper was scanned for recurrent themes to see how researchers use the term "software startup", as done in Paternoster et al. (2014). This was performed to answer RQ3.

### 3.7 Quality Assessment

To be able to build on the previous work of Klotins et al. (2015) and Paternoster et al. (2014), a quality assessment of the primary papers providing empirical evidence was done. As shown in table A.1, the total number of eligible papers was 22. Although systematic mappings 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, and to answer RQ2.

The quality assessment followed table 3 of Nguyen-Duc et al. (2015a), to assess the rigour, credibility and relevance of the papers, which Kitchenham (2004) has identified as important for performing empirical research in software engineering. 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. The complete table of all papers can be found in Appendix B.

<table>
<thead>
<tr>
<th>Problem Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1. Is research objective sufficiently explained and well-motivated?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Research Design</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q2. Is the context of study clearly stated?</td>
</tr>
<tr>
<td>Q3. Is the research design prepared sufficiently?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data collection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q4. Are the data collection &amp; measures adequately described?</td>
</tr>
<tr>
<td>Q5. Are the measures and constructs used in the study the most relevant for answering the research question?</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q9. Are the findings of study clearly stated and supported by the results?</td>
</tr>
<tr>
<td>Q10. Does the paper discuss limitations or validity?</td>
</tr>
</tbody>
</table>

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

### 3.8 Data Synthesis

Thematic analysis was selected as the synthesis method. As done in the mapping Klotins et al. (2015), we identified the SWEBOK knowledge areas that each primary paper covered. Having categorized them into knowledge areas, we further divided them into which research method
that was used in each paper, and which type of contribution each paper provided. Additionally, each paper was categorized according to research type, as well as the publishing venue. The papers that provided empirical evidence were also assessed as to what rigour they had. This allowed for comparison of the results, and to present an up to date overview of the research field. This was relevant for answering both RQ1 and RQ2.

For each paper, the term used for "startup" was classified. This, along with mapping the usage of recurrent themes describing startups, made it possible to assess if there exist an agreement in the community to the definition of startups. The recurring themes were adopted from Paternoster et al. (2014), and our results were compared to the findings of this mapping study. To extract the themes, both authors read the full text of the papers. Additional searches were performed in the pdf-version of each paper to provide even stronger evidence. Lastly, we examined the papers providing empirical evidence as to which degree they included sufficient contextual descriptions of the startups under investigation.

3.9 Validity of Review

There are several threats to the validity of systematic mapping studies (Zhou et al., 2016). One threat identified for this and most other literature mappings and reviews, is the publication bias. Positive results are more often published than negative results (Kitchenham, 2004). For this mapping study, none of the primary studies focused on failed startups, although the failure-rate of startups is high. Also, none of the primary studies focused on the used software engineering activities of failed startups, or presented failure from implementing specific methods or practices in startups.

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 seem 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). Since the precision of the chosen databases only was 2 percent (20 out of 1021), the probability that other smaller databases would provide many relevant studies is
low. The risk of missing papers published the last five years was mitigated by the use of forward snowballing which lead to the retrieval of 7 more papers.

To make sure the study selection was not biased from personal opinions, the final selection of the primary studies was undertaken collaboratively by both authors, resolving any conflicts, following guidelines from Kitchenham (2004). This decreased the risk of excluding any relevant papers.

For the quality assessment, we only used two points to collect answers, as both authors are 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. An example of this is from Nguyen-Duc et al. (2015a), where they gave each quality criterion a score between 0-4 instead of 0-1. This paper used the exact same quality criteria as we have used. As to this, it is more likely that the papers in our study obtained higher rigour than they would have gotten if another, more fine-grained assessment method had been used.
Chapter 4

Results

This section presents the results from the extracted data of the primary studies. The section is divided into the three research questions presented in section 3.2, to allow for better visualization and presentation of the most relevant findings. From initially 1012 unduplicated papers, the final number of primary papers ended up being 27.

4.1 RQ1: How has software startup research changed over time?

This section is divided into two sub-sections. The first sub-section, section 4.1.1, presents the publication frequency of primary studies, and compares this mapping study to both of the previous mapping studies (Paternoster et al., 2014; Klotins et al., 2015). The second sub-section, section 4.1.2, presents the different SWEBOK knowledge areas that have been covered by the papers in this study, along with the empirical evidence and the contribution types of each knowledge area. This is followed by a comparison to the results of the previous mapping studies.

4.1.1 Publication Frequency

Figure 4.1 shows the number of studies published in relation to software engineering in startups, each year from 2013 to 2017. The total number of published papers was 30. Three of the papers from 2013 are from the previous mapping studies, including (Shakir and Nørbjerg, 2013; Bosch et al., 2013; Blank, 2013). These were not included in our study as they have already been assessed. Since more than 60 percent of the papers are from the period 2016-2017, there is an increasing amount of papers being published.

85 percent of the papers from our study had high pertinence, which means that they were entirely focusing on software engineering activities in startups. The remaining four papers were focusing on activities of small software companies, and so set to partial pertinence. Although their
focus was not entirely dedicated to startups, some of them, like Laporte et al. (2014), performed empirical studies on startups, referring to them as "very small entities" or "small software companies". All studies were related to engineering activities, and so no paper got marginal pertinence.

Figure 4.1: Publication Frequency, 2013-2017

Figure 4.2 shows the complete number (74) of publications from 1994 to 2017, including all studies from both Paternoster et al. (2014) and Klotins et al. (2015). 57 percent of the papers in this period had high pertinence, while 20 percent had partial pertinence.

The number of unique primary papers in Klotins et al. (2015) was 4, as 10 of the 14 primary papers in this study also were among the 43 primary papers in Paternoster et al. (2014). The 4 papers were from the years 1994, 2000, 2008, and 2013 respectively. These numbers suggest that Paternoster et al. (2014) provides a more comprehensive mapping than Klotins et al. (2015). In fact, only 4 of the 10 top-ranked papers from Paternoster et al. (2014) were included in Klotins et al. (2015).
Looking at the numbers presented in figure 4.2, we observe that the publication frequency of papers between 2013-2017 is higher than for any period before 2013. From 1994 to 2013, the highest number of primary papers within a single year was 7 in 2008. In comparison, 2017 constituted 11 papers. The pertinence of the papers published between 2013-2017 was generally higher than what was found for the period 1994-2013.

4.1.2 Knowledge Areas

To classify our primary studies (2013-2017), the knowledge areas of SWEBOK (Bourque and Fairley, 2014) were chosen, which make up 15 categories in total. The categories were developed by the software community as a baseline for the body of knowledge within software engineering. By using the SWEBOK knowledge areas to classify the publications, comparisons can be made to the previous mapping study by Klotins et al. (2015). This will help structure the publications, and can thus be useful for addressing RQ1.

To illustrate what knowledge areas that have received most attention the last five years, and to what extent empirical studies have been undertaken, figure 4.3 was made. The figure shows which research methods that have been used to address each of the knowledge areas in our study. 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, like Nguyen-Duc et al. (2015b), which covers both software engineering process and software engineering management.
CHAPTER 4. RESULTS

Figure 4.3: Empirical Evidence, 2013-2017

The research methods that have been assessed followed the guidelines from (Oates, 2005), and include (1) survey, (2) design and creation, (3) experiment, (4) case study, (5) action research, and (6) ethnography. Each of the research methods are explained in greater detail in table A.2. The most frequently used research method was case study, which was used 81 percent of the times empirical studies was undertaken. The second most frequently used research method was experiments (10 percent), followed by surveys (6 percent) and design and creation (3 percent). Action research and ethnography were not used as research method in any of the primary studies, and were thus not included in the figure.

To illustrate which types of contributions that have been made within each knowledge area between 2013-2017, figure 4.4 was made. The figure shows the contribution type of each of the addressed knowledge areas. All 27 primary papers in our study are included in the figure. The 9 different knowledge areas are represented a total of 49 times through seven different contribution types, as several papers covered more than one knowledge area.
We differ between seven contribution types, as applied in Paternoster et al. (2014), originally suggested by Shaw (2003). These include (1) model, (2) theory, (3) framework, (4) guidelines, (5) lessons learned, (6) advice, and (7) tools (described in table A.3). All of the seven contribution types are represented. Lessons learned is the most frequently used contribution type, as it is used in 43 percent of the knowledge areas. The second most frequently used one is advice (25 percent), followed by model (12 percent), theory (10 percent), framework (5 percent), guidelines (5 percent), and tools (3 percent). Each of the contribution types are explained in Appendix A.

Only 9 of the 15 knowledge areas are covered in our primary papers. 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. Software engineering process has gotten the most contributions (13), followed by software engineering management (12) and software engineering professional practice (6).

Figure 4.5 shows the number of papers that cover the different knowledge areas in our study (red columns) and in Klotins et al. (2015) (blue columns). The total number of primary papers in Klotins et al. (2015) was 14. Both Klotins et al. (2015) and our mapping study include papers that cover more than one knowledge area. Klotins et al. (2015) did not differ between different contribution types.
From figure 4.5, we see that in Klotins et al. (2015) "software design" and "software requirements" are the most represented knowledge areas, while the other categories are fairly equally represented. Compared to the primary papers of our mapping study, we see that there is a significant change in the research direction for the last five years. Among the primary papers published between 2013 and 2017, "software engineering process" and "software management" have received significantly more attention from the community and are by far the most represented knowledge areas. Even though Klotins et al. (2015) only covers 14 papers, it covers two more knowledge areas than our mapping study. These are "software configuration management" and "software maintenance".

Paternoster et al. (2014) did not present any results in relation to the SWEBOK knowledge areas. However, it is interesting to look at the type of contribution each of their primary studies provided, and see how this relates to figure 4.4, which shows the contributions of our primary studies. This is illustrated in figure 4.6.
CHAPTER 4. RESULTS

Figure 4.6: Contribution types, this study and Paternoster et al. (2014).

The most frequently provided contribution type in Paternoster et al. (2014) was advice, followed by model and lessons learned. In our study, lessons learned was the most used one, followed by advice and model. The least frequently used ones combined from both studies were framework, guidelines, and tools. A discussion of these findings can be found in section 5.1.2.

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

To address this research question, we have made a bubble chart of each knowledge area, with corresponding rigour-rating, from our study. Only the papers that provided empirical evidence were evaluated, constituting 22 out of 27 primary papers. 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 our primary studies, 21 were conference papers, 5 were journal papers, and 1 paper was part of a book. However, it is necessary to perform a more comprehensive quality assessment process in order to compare results across different studies.

Figure 4.7 shows the degree of rigour within each knowledge area from our study. The figure is based on the quality assessment table found in Appendix B. The papers that have been assessed are the ones that provided empirical research from 2013-2017 (22 of 27). The x-axis represents the knowledge areas, while the y-axis represents the rigour (as explained in section 3.7).
Figure 4.7: Rigour of each covered knowledge area, 2013-2017

Only one paper got low rigour score (Edison et al., 2015), as it didn’t provide enough details about the data analysis and 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 get low rigour as well. In general, the papers got a high rigour score, which means that the quality of research was high.

4.2.2 Rigour of Primary Studies 1994-2017

A similar figure to figure 4.7 can be found in Klotins et al. (2015), as illustrated in figure 4.8. The figure shows the rigour of each of the primary studies, 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 4.8: Rigour and Research Type (Klotins et al., 2015)

Figure 4.9 illustrates the rigour of the contribution types provided by each of the primary papers in 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.

Figure 4.9: Rigour and Contribution Type (Paternoster et al., 2014)

70 percent of the papers in figure 4.9 received a medium score, while 21 percent scored high. The rest received a low score. Only 6 percent of the advice-papers got a high score, while 40 percent of the model-papers got high. All theory-papers received high rigour score, but they only
constituted 2 papers.

Comparing our results to the ones found in the previous papers, it is clear that our primary papers got a generally higher rigour score than papers in the previous mappings. Possible explanations for this are presented in both section 3.9 and in section 5.2.

### 4.3 RQ3: What effort has been made to characterize the context of software engineering in startups?

The following section is divided into three sub-sections. Together, they intend to identify the contextual descriptions provided by the papers from the last five years, and overall in the period 1994-2017. Section 4.3.1 shows how papers use the term for "startup" company differently, comparing the results from this mapping study to the results from Paternoster et al. (2014). Section 4.3.2 shows whether the papers from 2013-2017 focus on startups in the context of incubators. Section 4.3.3 illustrates how the papers from 2013-2017 describe the situational contexts of the startups under investigation.

#### 4.3.1 Themes and Term Frequency

To illustrate how researchers use different definitions and terms in their characterizations of startups, we extracted the themes in “Theme Description” in table 4.1 from each of the 27 primary studies. The themes followed the same guidelines as suggested by Paternoster et al. (2014). An explanation of the different theme descriptions can be found in Appendix B. To extract the themes, both authors read the full text of the papers. In addition to this, searches were performed in the pdf-version of each paper to provide even stronger evidence.
Table 4.1 shows that the use of describing themes in the primary papers of this mapping study are highly inconsistent. There are no single term that all the 22 empirical papers use for the startups they investigate. The low frequencies of the recurring themes also show that many of the papers have poor contextual descriptions.

Table 4.2 shows the usage of different themes from Paternoster et al. (2014). The study elicited the themes from 43 primary studies, and are ordered according to their frequency (the number of papers using the respective themes to describe the startup context). A total of 15 themes were extracted from the papers. As to this, it is possible that other selections of papers would have provided a different set of themes. However, the themes do constitute a solid base for characterizing the context of startups.
### Theme Descriptions, 1994-2013 (Paternoster et al., 2014)

<table>
<thead>
<tr>
<th>Theme ID</th>
<th>Theme Description</th>
<th>Frequency (nr of papers)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>Lack of resources</td>
<td>18</td>
</tr>
<tr>
<td>T2</td>
<td>Highly reactive</td>
<td>17</td>
</tr>
<tr>
<td>T3</td>
<td>Innovation/Innovative</td>
<td>15</td>
</tr>
<tr>
<td>T4</td>
<td>Uncertainty</td>
<td>14</td>
</tr>
<tr>
<td>T5</td>
<td>Rapidly evolving</td>
<td>14</td>
</tr>
<tr>
<td>T6</td>
<td>Time-pressure</td>
<td>13</td>
</tr>
<tr>
<td>T7</td>
<td>Third party dependency</td>
<td>10</td>
</tr>
<tr>
<td>T8</td>
<td>Small team</td>
<td>9</td>
</tr>
<tr>
<td>T9</td>
<td>One product</td>
<td>9</td>
</tr>
<tr>
<td>T10</td>
<td>Low-experienced team</td>
<td>8</td>
</tr>
<tr>
<td>T11</td>
<td>New company</td>
<td>7</td>
</tr>
<tr>
<td>T12</td>
<td>Flat organisation</td>
<td>5</td>
</tr>
<tr>
<td>T13</td>
<td>Highly risky</td>
<td>5</td>
</tr>
<tr>
<td>T14</td>
<td>Not self-sustained</td>
<td>3</td>
</tr>
<tr>
<td>T15</td>
<td>Little working/operating history</td>
<td>3</td>
</tr>
</tbody>
</table>

Comparing our results to the ones found in Paternoster et al. (2014), we observe that the use of theme descriptions have changed quite a lot. The most frequently used theme in the previous mapping study was only the fourth most used one in our study. From the recurring themes, T6 was the only theme being used to the same extent before and after 2013. The differences are significant, both since the themes initially were extracted from 43 research papers, and taking into account that it is only four years between the studies.

Table 4.3 shows the number of primary papers from 2013-2017 using the specified terms for "startup company". As the table shows, the term was found to be used quite differently in the primary papers. 75 percent of the studies from 2017 used the term “startup”, compared to 50 percent in 2014. 15 percent used the term “start-up” in 2017, while 50 percent used the term “start-up” in 2014. The inconsistent use of terms are one of the main challenges to develop a coherent body of knowledge within software engineering for startups. Even though 20 studies used the term “startup”, the context for which they were used was not the same, or the study context was poorly described.
CHAPTER 4. RESULTS

<table>
<thead>
<tr>
<th>Term</th>
<th>Frequency (nr of papers)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Startup</td>
<td>20</td>
</tr>
<tr>
<td>Start-up</td>
<td>4</td>
</tr>
<tr>
<td>Very small entity</td>
<td>1</td>
</tr>
<tr>
<td>Very small company</td>
<td>1</td>
</tr>
<tr>
<td>Very small enterprise</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.3: Term Frequencies, 2013-2017.

Table 4.4 shows the term frequencies from Paternoster et al. (2014), and are based on the usage extracted from the primary papers’ titles. In situations where the title didn't use any of the terms, the papers were searched for and found through Google Scholar, for revision of the abstract. The papers Häsel et al. (2010); Stanfill and Astleford (2007); Kuvinka (2011); Lai (2010); Yoffie and Cusumano (1999) did not use any of the terms, or was not found. Three of the terms were not used.

<table>
<thead>
<tr>
<th>Term</th>
<th>Frequency (nr of papers)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Startup</td>
<td>19</td>
</tr>
<tr>
<td>Start-up</td>
<td>19</td>
</tr>
<tr>
<td>Very small entity</td>
<td>0</td>
</tr>
<tr>
<td>Very small company</td>
<td>0</td>
</tr>
<tr>
<td>Very small enterprise</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.4: Term Frequencies, 1994-2013 (Paternoster et al., 2014).

Comparing our results to the ones in table 4.4, it can be observed that the usage of terms for startup companies is quite different. In the previous study, papers either used "startup" or "start-up", with each constituting half of the frequencies. In our study, "startup" is used in almost 6 times as many papers as "start-up". As to this, "startup" is becoming a more frequently used term than "start-up" compared to before.

4.3.2 Incubated Companies

Figure 4.10 shows the percentage of the 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 (Pomermaier et al., 2017; Souza et al., 2017) focused on incubated startups. The first one presents learnings from empirical studies of eight startups in an incubator context, where they focus on software methodologies and testing practices. The second one presents an academic startup model based on case studies of four startups in an incubator environment.
4.3.3 Contextual Descriptions

The primary studies from 2013-2017 that have provided empirical evidence and sufficient contextual descriptions are presented in table 4.5. The relevant context information includes the attributes: (1) number of startups under investigation, (2) size of the company/team, (3) the domain in which the startups operate within, and (4) other relevant contextual descriptions beyond these three. Examples of (4) are lifecycle stage, age/year of establishment, location, software development methodology, or whether the companies have received funding.

As illustrated in the figure, 14 of the 22 studies that provide empirical evidence showed a sufficiently amount of contextual description. The contextual description in the remaining eight papers were either absent or not sufficiently explained. The papers that did not provide any empirical evidence were not evaluated. The two last papers in the table are subject to omission, as both have two fields of "not specified". They are examples of what we regarded as a minimum threshold when searching for context descriptions in the papers.

Apart from what is presented in table 4.3.3, an interesting finding from Laporte et al. (2015) is that the authors actually characterize the enterprises by their size. This is in contrast to many of the other papers that provide contextual descriptions. Micro enterprises consist of 1-9 employees, and have less than 2M dollars in annual turnover. Small enterprises have 10-49 employees, and less than 10M in annual turnover. Medium enterprises have 50-249 employees, and less than 50M in annual turnover. However, the paper did not specify more than the company size in their characterization of the startup companies under investigation, and was thus not included.
Table 4.5: Contextual Descriptions, 2013-2017

The following list shows some of the descriptions of companies that have participated in empirical research on startups.

- The number of startups under investigation is in the range from 1-13 startups. The most
frequently used number of startups was found to be 3-5.

- The number of employees is usually in the range between 2-25, depending on the lifecycle stage of the company. At early stages, the number of employees tend 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. 25 employees is the highest number found in the papers. This was an E-commerce company, which was at a mature lifecycle stage.

- It is usual that researchers specify the domain in which the startups operate within. The most common explanations is whether the business is B2B/B2C, or specification of the product/service being developed.

- The age of the investigated companies is usually in the range 1 month to 3 years. Some papers, like Sánchez-Gordón and O’Connor (2016), investigated VSEs, one of them 18 years active. 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 adhoc.

- No more than two papers mentioned whether the investigated companies had received any funding, and if so what kind of funding they had received.
Chapter 5

Discussion

This section will present discussion and analysis of each of the three research questions. The analyses are mainly based on the data presented in the results section.

5.1 RQ1: How has software startup research changed over time?

To address this research question, we will present each of the nine knowledge areas that our primary studies covered, and compare the different contributions within each. This will allow for identifying the research within the field between 2013-2017. Furthermore, a comparison of these findings will be made to the two previous mapping studies. A merging of the papers is necessary to understand how software startup research has changed over time. Lastly, we will present a suggestion for how future classifications of research within the field should be conducted for best addressing startup-related issues.

5.1.1 SWEBOK Knowledge Areas

The primary studies were classified into 9 of the 15 SWEBOK knowledge areas. This section will present what our primary papers have addressed and highlighted as needed for future work, allowing for a better understanding of what the research between 2013-2017 has covered.

5.1.1.1 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. The situational context consist of a large number of concerns and factors, as found in the “reference framework” (Clarke and O’Connor, 2012), indicating why software engineering is so hard (Marks et al., 2017).
CHAPTER 5. DISCUSSION

The situational factors in the reference framework explains the need for startups’ own software development processes, and why strictly following the agile methodology is often outside the scope of small startups (Yau and Murphy, 2013). In early-stage software startups, research shows that systematic software engineering processes often are replaced by light-weight ad-hoc processes (Giardino et al., 2014).

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 (Nguyen-Duc et al., 2015b).

5.1.1.2 Software Engineering Professional Practice

This knowledge area requires that software engineers possess the required knowledge, skills, training, and experience to practice professional and responsible software engineering (Bourque and Fairley, 2014). Standards like ISO are meant to ensure high quality and reliability of software products (ISO, 2017), and can thus help developers to practice software engineering at a level in line with these objectives. The paper Laporte et al. (2014) 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 (Pantiuchina et al., 2017). 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, and is about the application of knowledge in the engineering discipline. This knowledge can allow engineers to develop and maintain software more efficiently and effectively, and help practitioners to practice professional software engineering. As to this, more work should also be un-
dertaken 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. The issues of engineering competencies and competency needs in software startups have also been addressed by the research agenda (Unterkalmsteiner et al., 2016).

5.1.1.3 Software Engineering Management

Software engineering management is concerned around a wide range of different areas, including planning, measuring, coordinating, and reporting activities to support systematic software development and maintenance (Bourque and Fairley, 2014). 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 (Laporte and O’Connor, 2016; Nguyen-Duc et al., 2015b; Yli-Huumo et al., 2015). However, these papers primarily focus on software engineering processes (Laporte and O’Connor, 2016; Nguyen-Duc et al., 2015b) and software quality (Yli-Huumo et al., 2015). Although the Hunter-gatherer cycle presented by Nguyen-Duc et al. (2015b) 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.

Apart from these, much work has been done to identify the managerial part of software engineering. Duc and Abrahamsson (2017) explores the outsourcing relationship in software startups, and find that outsourcing is a feasible option for startups, also in the early stages. 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 (Bajwa et al., 2017), and which pivoting types exist (Bajwa et al., 2016). 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 (Unterkalmsteiner et al., 2016) has addressed a need for more research
to identify how startups explicitly manage risks, and how startups actually should 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 actually apply this theory in practice, so that researchers can develop tailored models and frameworks to reduce business and technical risks.

5.1.1.4 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. Shortcuts can lead to the accumulation of what is called intentional technical debt (Yli-Huumo et al., 2015). The paper by Yli-Huumo et al. (2015) showed that technical debt should be divided into intentional and unintentional debt. Unintentional technical debt can happen when business model experimentation is left out. No matter how good the idea may seem, not validating the idea with customers, could lead to 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 is therefore essential. The same problem is referred to as the developer’s dilemma (Terho et al., 2016). The developer’s dilemma also emphasises 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 (Chicote, 2017). The technique is based on “duck taping” each part of 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.

5.1.1.5 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. (2015) suggests, 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 to 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 RQ4 in section 3.1.6 of the research agenda (Unterkalmsteiner et al., 2016), 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 usage of technologies and tools to develop software (Bourque and Fairley, 2014). As to this, it can be feasible to address software construction through sub-categories, like design, testing, and verification.

5.1.1.6 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 (Bourque and Fairley, 2014). Topics include principles and properties of models, analysis of models, and various software development methods.

Startups need software development methodologies and techniques tailored for their specific contexts. These should be based on Lean Startup and agile principles (Paternoster et al., 2014). In RQ3 of section 3.1.1 from the research agenda (Unterkalmsteiner et al., 2016), researchers are encouraged to identify what engineering methods and models that are used today, and whether they work in a startup context.

The Greenfield Startup Model (GSM) aims at explaining how development strategies and practices are engineered and utilized in startups (Giardino et al., 2016). A similar model, the academic startup model, was created by Souza et al. (2017), which illustrates how software startups structure and execute their engineering activities. Both papers conclude that early-stage software startups aren't adopting traditional development methodologies. Instead, rapid prototyping and continuous experimentation are in focus, and so engineering practices are adapted to each startup's specific context. These models provide development objectives that software engineers in startups can use. They also provide 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 (Giardino et al., 2016).
5.1.1.7 Software Testing

As we have seen in section 2.5, software validation and testing is an essential part 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.

Through interviews and observations of eight software startups, Pompermaier et al. (2017) found that testing is critical to startups' success. However, in the construction phase of the first version of the system, the technical teams did not use any software testing techniques. This scenario did however change 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. (2017), 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 (Unterkalmsteiner et al., 2016).

5.1.1.8 Software Requirements

Software requirements engineering activities include elicitation, negotiation, analysis, specification, and validation of requirements (Bourque and Fairley, 2014). 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 (Ries, 2011). Prototyping, continuous experimentation of minimum viable products, and pivoting are effective tools and methods startups can utilize in their requirements engineering processes (Unterkalmsteiner et al., 2016).

Rafiq et al. (2017) found that there was a lack of studies that had investigated how software startups actually perform requirements engineering processes. The study identified the require-
ments elicitation techniques of three software startups, and 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 use of 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 even more elicitation techniques, and to provide stronger evidence of the findings in Rafiq et al. (2017). As that study only looked at elicitation techniques, more research should be conducted to identify negotiation, specification, and validation techniques. More research is also necessary to identify requirements engineering for different lifecycle stages. This can help startups in specific situational contexts to identify which requirements engineering techniques to use.

5.1.1.9 Software Design

The role of MVPs in software startups has been addressed by Nguyen-Duc and Abrahamsson (2016). They suggest that MVPs not only are effective tools for requirements elicitation, but also that they are good for bridging knowledge gaps between entrepreneurs, investors, and software developers - emphasizing that MVPs can serve as a multiple facet product and not only for business experimentation. 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 is another topic that has been addressed (Nguyen-Duc et al., 2017). 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 makes 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 a real system and are often developed from many previous prototypes. Customer feedback is an essential part of business experimentation, and is mainly done through prototyping. With time-pressure, the speed of prototyping is important, but quality can often be equally important. The authors concluded that 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

This section presents our findings in comparison to the previous mapping studies. It is challenging to match primary papers with the right knowledge areas. One reason for this is that some of the knowledge areas aren’t directly relevant 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 can also lead to different classifications.

Looking at the knowledge areas that’s been covered between 1994-2017, we can see that software maintenance and software configuration management have not gotten many contributions. As one of the most important things 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, which illustrates 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, and are thus not particularly relevant for specific software startup research. However, we have argued (section 5.1.1.2) that more research should be conducted within the area 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 gotten quite 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 that have gotten many contributions include software engineering process, software engineering management, software construction, software design, and software requirements. Management was an area which Klotins et al. (2015) identified that more work should be conducted within. 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 cost, to avoid implementing unimportant functionality. This is closely related to prototyping and business experimentation, which are essential aspects of Lean Startup. Prototyping is also an essential part of software design, which was the most covered knowledge area in Klotins et al. (2015).

Another area that was not sufficiently covered between 1994-2013 was software engineering
process. This is in contrast to the last five years, where process has been the area with most contributions. Klotins et al. (2015) argue that software engineering process is more 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. This illustrates that different authors have different interpretations of the knowledge areas.

Looking at the publication frequencies of primary papers between 1994-2017, it is clear that increasingly more papers are being 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 (Unterkalmsteiner et al., 2016) 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 actually apply them (Giardino et al., 2016; Souza et al., 2017). 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 and more work is underway 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. 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 pay even more attention to 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 Future Classification

The paper by Unterkalmsteiner et al. (2016) has 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 together 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 primary studies according to the SWEBOK knowledge areas only 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. (2015), which also classified the papers into the categories of each knowledge area. This resulted in less than 50 percent coverage, implying that our mapping would get an even lower coverage. A reason behind the low coverage is most certainly that 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. This was also acknowledged by Klotins et al. (2015), which used SWEBOK for lack of a better alternative.

For future mappings, it would be sensible to categorize the papers into the newly established research themes (Unterkalmsteiner et al., 2016), instead of the knowledge areas found in SWEBOK. These are better suited for startups research, and can help guide researchers to provide more knowledge into the specific areas that are most important for the challenges faced by startups. The following list presents the different themes. 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 evidences reported?

The systematic mapping study by Paternoster et al. (2014) provided a mapping of the research within software development in startup companies for the period 1994-2013, and included 43
primary studies. Each of these studies provided empirical evidence, as this was a quality crite-
ria 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 (Coleman and O’Connor, 2008; Cole-

Another mapping study (Klotins et al., 2015) within software engineering knowledge areas in
startup companies 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 com-
pare 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 pa-
per identified a lack of studies related to requirements processes, the “developer’s dilemma” (as
discussed in section 5.1.1.4), software architecture, and software engineering processes.

In our systematic mapping, 80 percent (22 of 27) of the primary papers provided empirical ev-
idence, covering 9 of the 15 knowledge areas. These papers had significantly higher rigour and
quality compared to the primary papers in the previous mapping studies. Figure 4.3 in sec-
tion 4.1.2 shows that the areas that’s received most contributions in terms of empirical research
between 2013-2017 are software engineering process, software engineering management, and
software engineering professional practice. On the contrary, 5 of the knowledge areas received
less than five scientific contributions, which argues that more research is required, even within
areas that have gotten attention from the research community. This is also in line with the re-
search agenda’s addressed need for further empirical evidence (Unterkalmsteiner et al., 2016).
Although some of these research questions have been identified, more work is required to add
to the body of knowledge. This is supported by our discussion in section 5.1.

Comparing our findings to that of the previous mapping studies, we find that the papers are
quite distinct in terms of quality (rigour) of the provided empirical evidence. As for the previous
studies, the reported rigour of the primary papers was at a generally lower level than what was
found in our study, where only 2 of 22 papers did not achieve high rigour. This was found to be
almost exactly opposite to what was reported in Klotins et al. (2015). Possible explanations to
this have been discussed in section 3.9, where one logical reason was that different quality as-
essment methods have been used. However, as more researchers are contributing to the field,
the quality of research is increasing.

Many of the authors who have contributed to our primary papers (including the research agenda
and the two previous mapping studies) are members of «The Software Startup Research Net-
work», 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 our primary studies, in which all received a high rigour score. Another author who's contributed to three of our 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 in 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 conducting high-quality research with several researchers contributing with multiple papers, as illustrated by specific initiatives that promote scientific work.

5.3 RQ3: What effort has been made to characterize the context of software engineering in startups?

The most frequently used term for referring to startup companies is “startup”, with almost 75 percent of our primary papers that are directly related to the startup context using this term. The rest of the papers used the term “start-up”. The papers in Paternoster et al. (2014) used these two terms for "startup" the same number of times. In order to create a coherent definition for startups, ideally only one of the terms should be used. Between 2013-2017, 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 find relevant results.

Table 4.1 presented in section 4.3 illustrates the use of describing themes in our primary studies. The primary papers showed a highly inconsistent use of the different terms when describing startup companies, with no single term being used by all papers. As stated in the results, this also relates to the poor contextual descriptions found in many of the papers. Comparing it to the original figure in Paternoster et al. (2014), we observe that the usage has changed quite significantly. Only theme T6 was being used to the same extent before and after 2013, shown by table 4.1 and 4.2. Interestingly, the least used term in Paternoster et al. (2014) is the fifth most used term by the papers between 2013-2017. As to this, it seems that some of the recurrent themes 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 defini-
tion and body of knowledge for the startup context. Based on the recurring themes found in the primary papers, we argue that at least the themes occurring in more than 25 percent or more of the studies should be part of a unique definition of startups. The following themes were:

- Innovation/innovative
- Uncertainty
- Lack of resources
- Small team
- Little working/operating history
- Time-pressure

Theme descriptions that were not very relevant for startups include “third-party dependency”, “not self-sustained”, and “flat organisation”. These terms should be avoided as the primary definition by researchers in the community.

Many of the primary studies did not explain the situational context of the startups under investigation sufficiently. Of 27 primary papers, 22 provided empirical evidence. Only 14 of these provided a sufficiently amount of contextual description of the investigated startups, as illustrated in table 4.5. Interestingly, only two papers mentioned incubators as part of the startup context. Without a unique definition in literature, the importance of precise contextual descriptions becomes even bigger, especially for transferring results from one environment to another (Klotins et al., 2015).

One contextual factor that has received surprisingly little focus, is team size. A startup with 5 employees have different needs, and challenges from a startup with more than 150 employees (Deias et al., 2002), e.g. communication needs. 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 changes 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 contextual factors 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 in section 3.1.1 of Unterkalmsteiner et al. (2016) aims at developing a software startup context model that would allow a coherent characterization of the startup context. Since there is no agreement of a standard definition, it is challenging to provide coherent contributions to the research area.
Chapter 6

Planning of the Investigation

The purpose of the systematic mapping study was to provide an up to date overview of the research results within the field of software startups, by identifying the focus and quality of the last five years of research, and merge this with the results of previous mapping studies. This was done by structuring and analyzing the literature on software development in startup companies from the last five years. Some of the research gaps found in this study form the basis for an investigation that will be conducted by the authors during spring 2018. The results of the investigation will be presented in a Master thesis at Department of Computer Science, NTNU.

The rest of section 6 will introduce the planning of this investigation. Firstly, section 6.1 presents a short summary of the findings and observations of the systematic mapping. Secondly, section 6.2 introduces the research questions and problem areas that will be investigated. Lastly, section 6.3 presents the research plan, including research process, participants, paradigm, and deliverables.

6.1 Mapping Study Findings and Observations

Our systematic mapping study identified several research gaps and potential future work. All knowledge areas of SWEBOK presented the need for more empirical investigations. Most interesting was how little attention the primary papers devoted to the domain (market sector) in which the startups operated, and showed overall poor contextual descriptions. This makes it hard both to compare software engineering activities, and to transfer knowledge between startups in similar domains. The purpose of the Master thesis will thus be to address domain-specific software startup challenges.
6.2 Research Questions and Problem Areas

The importance of collaboration in today’s competitive environment has been described by the Open Innovation paradigm (section 2.2). For some market sectors, the use of startups in innovation has increased significantly the last few years. The financial sector is one of these industries, where the global investment in financial technology (fintech) ventures climbed to 22.3 billion dollars in 2015, from 1.8 billion dollars in 2010 (Accenture, 2016). Initially, most startups wanted to compete with the traditional companies in the financial sector. Today however, there is more collaboration, where startups partner with, or are acquired by established companies. This is showed by the high percentage (83 percent) of the investments going to collaborative fintech (Accenture, 2016).

New digital technology and startups’ ability to innovate are expected to change several industries by 2020. From over 103 CEOs and Sr. Business Executives in the financial sector, 57 percent expected the industry to change substantially, or to change to the unrecognizable in five years’ time. In the same sector, 70 percent meant competition from digital-enabled companies entering the industry from other industries was a bigger threat than traditional industry competitors (Gartner, 2016). This highlights the need for all companies in these highly competitive industries to open up for more collaboration to innovate, for which business incubators are a great tool.

Through the use of business incubators, established companies can work more closely with startups. This will allow for more rapid business model testing and experimentation, to learn fast, and quickly find the best path forward. For this, business incubators have become an increasingly popular approach in large companies’ innovation processes (Grimaldi and Grandi, 2005). Today, this is highlighted by their emergence in the Norwegian and European innovation communities, where FinTech F3, Trondheim 2017, and Station F, Paris 2017, are examples of newly opened business incubators. The latter one is a 265 million dollars investment, housing more than 1000 startups (Agnew, 2017). The president of France, Emmanuel Macron, has stated that “entrepreneurs are the new France” and that “he wants France to be a country of unicorns (companies valued at more than 1Bn dollars)” (Agnew, 2017).

One of the findings from the systematic mapping study was that little research has been conducted in relation to startups in incubators. Having in mind that incubators are so influential in today’s innovation processes, more research should be conducted to investigate incubated startups. Business incubators can help startups survive and grow during their early stages. They allow startups and entrepreneurs to learn from each other, and facilitate collaboration between companies, investors, mentors, and other stakeholders, to create innovative products and ser-
vices. The term business incubation refers to an “interactive development process where large companies can encourage people to start their own business and support startups in the development of new, innovative products” (Aernoudt, 2004). If the innovative ideas or products are promising, the large companies can access these at an early stage, or even acquire the startup to get a competitive edge.

Even though the financial sector and fintech have received most attention the latest years, other industries have also opened up for more joint innovation and collaboration with other established companies and startups. In the healthcare sector, new technology referred to as medtech will change the way consumers interact with their healthcare providers. Consumers are already using mobile applications as their personal diagnostic, and doctors will use artificial intelligence and big data to faster come up with the correct diagnosis, and provide more personalized treatment. Other industries expected to go through a digital transformation the coming years are the real estate industry and the education sector, having received their own buzzwords “proptech” and “edtech”.

The lack of research papers addressing market-specific software engineering practices, and the increased use of startups in established companies’ innovation processes, have motivated the following research questions:

1. RQ1: To what extent does market sector pose any differences in software engineering practices for startups?
2. RQ2: What situational factors characterize the specific market sectors?
3. RQ3: What is the need for incubators specific to market sector?

### 6.3 Research Plan and Approach

The research plan follows guidelines for research processes as suggested by (Oates, 2005), illustrated in figure 6.1. The autumn project consisted of (1) motivation and initial research questions (section 1), and (2) the systematic mapping study resulting in the conceptual framework and the proposed research questions in section 6.2. This section (6.3) presents the research plan for the investigation of the research questions. The results will be presented in the Master thesis.

#### 6.3.1 Research Process

For the Master thesis, survey and case study are considered the most suitable approaches for addressing the research questions, following the guidelines from Oates (2005). Surveys can
allow for generalization of the results to a larger population through web-distributed questionnaires, however the large number of participants required, and the time-constraints of the project makes this infeasible. Observations would also be a desirable approach, but as the project only spans for one semester this would be challenging. Hence, method- and strategy triangulation will not be undertaken.

Figure 6.1: The Research Process (Oates, 2005). The dashed boxes illustrates the systematic mapping study, while the green boxes illustrates the Master thesis.

To address the research questions, a case study will be performed. Semi-structured interviews allow for a discoverable approach, as interviewees can express themselves more freely. This approach fits both with the time constraints of the project, and the availability of both incubators and software startups in the area. To obtain results closer to the reality and provide stronger validity, allowing for analytical generalisation through qualitative research, we will perform 12 to 15 interviews of maximum 45 minutes (due to time-pressure of startups) in collaboration with minimum three incubators and associated startups. The number of interviews depends on the interviewees’ ability to participate, and to which degree each interview provides useful insight.

6.3.2 Participants

Both authors are responsible for planning and conducting the research. The supervisor is Letizia Jaccheri and the co-supervisor is Ilias O. Pappas; both will contribute with knowledge about research within software engineering. Associate professor Anh Nguyen-Duc will contribute with knowledge about software startup research.

For the Master thesis, software developers in incubated startups are regarded as main participants, but all people attached to either a startup or an incubator can contribute. The incubators FinTech F3 and FinStart Nordic and their associated startups have agreed to participate in the studies. The following list contains incubators that are regarded as relevant participants.
• Fintech F3 (Trondheim) - houses four startups.
• FinStart Nordic (Stavanger) - opens in January 2018.
• Oslotech (Oslo) - Norway’s largest incubator with over 70 startups.
• Kjeller Innovasjon (Lillestrøm) - focusing on greentech startups.
• EdTech (Helsinki) - Nordic’s largest incubator for learning solutions.
• Station F (Paris) - Europe’s largest incubator.

All participants must sign informed consent forms to participate. Data from participants will be kept anonymous, and will not be used outside of this research.

6.3.3 Paradigm

In the Master thesis, we will address the research questions through a case study. This approach is related to the interpretivism paradigm, as we will generate knowledge by investigating the meaning of events and actions of those who experience them, and generalize the results to a larger population. Such qualitative methods are usually performed on few instances through semi-structured interviews, which is what we intend to conduct. As researchers, we are not neutral in the research process since we will interact with the participants. To provide more trustworthy and credible results, both researchers will participate in the interviews, and we must be aware of how we influence the respondents and how we interpret their answers.

6.3.4 Deliverables

The systematic mapping study and the research results from the case study will be presented in two deliverable documents, Project thesis and Master thesis. The systematic mapping study is planned to be submitted to the Software Engineering conference XP2018 in collaboration with our supervisors and Anh Nguyen-Duc. The Master thesis is planned to be submitted to ICSE2018, the 40th International Conference on Software Engineering.
Chapter 7

Conclusion

In this study, we have applied a systematic mapping method to analyze the literature related to software startup engineering. A total number of 27 primary papers for the period 2013-2017 have been presented. These papers have been compared to the previous mapping studies (Paternoster et al., 2014; Klotins et al., 2015), which include a total number of 47 unduplicated primary papers for the period 1994-2013. 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.

7.1 RQ1: How has software startup research changed over time?

In the period 1994-2017, most contributions have been provided between 2013-2017, showed by the publication frequencies in figure 4.2. From 2013-2017, most research has been 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 has gotten far less contributions compared to 1994-2013. These findings illustrate a change of research direction. The knowledge areas software engineering models and methods, software quality, and software testing have gotten 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, as suggested in section 5.1.1.

Between 1994-2017, four knowledge areas (computing, mathematical, and engineering foundations, and software engineering economics) were not covered at all. Additionally, for the period 2013-2017, two more knowledge areas (software configuration management and software maintenance) were not covered. 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) (section 5.1.3), as these are better suited for startup research. The SWEBOK knowledge areas are difficult to classify due to different authors' biases in terms of knowledge and personal opinions.

The empirical investigations were generally based on data from few startups, and so there's a need for more empirical research, investigating a larger number of startups. 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. Pantiuchina et al. (2017) stands out from the rest of the primary papers as an example of empirical investigation of several startups, performing a survey on 1526 startups.

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

The previous mapping studies reported a lack of high-quality research, which made it hard to transfer results to industry. However, the rigour of the primary papers in this study was generally high. One obvious reason for the significant differences is that different quality assessment methods have been used, where the previous mappings provided an even more fine-grained representation than our study. Apart from this, we argue that there has been an increased focus on conducting high-quality research, with several authors contributing with multiple papers. The increased quality is also a result of the initiative "The Software Startup Research Network", which promotes scientific work. The increased importance of startups, illustrated by a shift to a more open, collaborative innovation paradigm, has been an important factor to highlight the need for more research. High-quality research is of great importance in order to reduce the high failure rates of software startups in the coming years.

7.3 RQ3: What effort has been made to characterize the context of software engineering in startups?

We emphasize that inconsistent usage of terms for "startup company" should be avoided. Researchers and practitioners should use the term "startup" for future work. To develop a coherent definition and body of knowledge for the startup context, the themes (1) innovation/innovative, (2) uncertainty, (3) lack of resources, (4) small team, (5) little working/operating history, and (6) time-pressure should be part of the contextual description of the investigated startup company to regard the research as highly relevant. Additionally, aspects like (1) team size, (2) domain, (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 incubators increasingly more often are part of the startups environment, this may also affect the context and challenges of startups, and should be part of a complete description. This will allow practitioners to learn from each other, and incorporate results from the latest of research. Only 14 of the primary papers between 2013-2017 provided adequate descriptions, and all primary papers showed an overall inconsistent use of describing terms. This illustrates that there still is a need for a more comprehensive endeavor to describe the startup context. We highlight the importance of future work to provide extensive contextual descriptions.

7.4 Implications and Future Work

We conclude that research is of higher quality now than before, but still more work is required to support startups’ software engineering practices, to prevent startups from failing. For startups to be able to benefit from research, and transfer results to industry, thorough contextual descriptions are important. Future work should try to make up a common software startup definition, making use of the frequency of recurrent themes, and central aspects of the contextual description presented in this mapping study. This would help in the creation of a coherent body of knowledge, overall strengthening the rigour of studies supporting the engineering activities of startups. With the high number of dedicated and knowledgeable researchers within the field of software startup engineering, we expect more quality research the coming years.

As a next step, we seek to address market-specific software engineering practices. The primary papers devoted little attention to the domain (market sector) in which the startups operated, and provided few results in relation to startups in incubators. Having in mind that incubators are so influential in today’s innovation processes, more research should investigate incubated startups. Business incubators can help startups survive and grow during their early-stages, and can allow established companies to learn fast, to quickly find the best path forward. Through empirical investigations, we will try to discover whether different market sectors pose differences in software engineering practices for startups, what situational factors that characterize the specific market sectors, and whether there is a need for market-specific incubators.
Appendix A

Classification Schema

Appendix A presents the classification schema (A.1), which includes the classification of each primary paper. Table A.2 and table A.3 explain some of the attributes of the classification schema in more detail.
<table>
<thead>
<tr>
<th>ID</th>
<th>Research Method</th>
<th>Contribution Type</th>
<th>Knowledge Area</th>
<th>Pertinence</th>
<th>Term for startup</th>
<th>Research method</th>
<th>Incubator context</th>
<th>Publisher</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Yau and Murphy, 2013)</td>
<td>Case study</td>
<td>Lessons learned</td>
<td>SWE Process</td>
<td>Full</td>
<td>Startup</td>
<td>Experience paper</td>
<td>No</td>
<td>Penn University</td>
</tr>
<tr>
<td>(Laporte et al., 2014)</td>
<td>Experiment</td>
<td>Lessons learned</td>
<td>SWE Professional Practice, SWE Management, SWE Quality</td>
<td>Partial</td>
<td>Start-up</td>
<td>Evaluation research</td>
<td>No</td>
<td>International Conference on the Quality of Information and Comm Tech</td>
</tr>
<tr>
<td>(Eloranta, 2014)</td>
<td>Framework</td>
<td>SWE Professional Practice</td>
<td></td>
<td>Full</td>
<td>Start-up</td>
<td>Philosophical paper</td>
<td>No</td>
<td>Association for Computing Machinery</td>
</tr>
<tr>
<td>(Laporte et al., 2015)</td>
<td>Experiment</td>
<td>Guidelines</td>
<td>SWE Process</td>
<td>Full</td>
<td>Start-up</td>
<td>Evaluation research</td>
<td>No</td>
<td>10th International Conference on Evaluation of Novel Approaches to SWE</td>
</tr>
<tr>
<td>(Edison et al., 2015)</td>
<td>Survey</td>
<td>Tool</td>
<td>SWE Construction</td>
<td>Full</td>
<td>Startup</td>
<td>Solution proposal</td>
<td>No</td>
<td>Springer</td>
</tr>
<tr>
<td>(Nguyen-Dac et al., 2015b)</td>
<td>Case study</td>
<td>Model</td>
<td>SWE Process, SWE Management</td>
<td>Full</td>
<td>Startup</td>
<td>Solution proposal</td>
<td>No</td>
<td>Association for Computing Machinery</td>
</tr>
<tr>
<td>(Sánchez-Gordón and O'Connor, 2016)</td>
<td>Case study</td>
<td>Lessons learned</td>
<td>SWE Process</td>
<td>Full</td>
<td>Very Small Company</td>
<td>Experience paper</td>
<td>No</td>
<td>SW Quality Journal</td>
</tr>
<tr>
<td>(Wasserman, 2016)</td>
<td>Guidelines</td>
<td>SWE Process</td>
<td></td>
<td>Partial</td>
<td>Startup</td>
<td>No</td>
<td>Springer, Cham</td>
<td></td>
</tr>
<tr>
<td>(Terho et al., 2016)</td>
<td>Case study</td>
<td>Lessons learned</td>
<td>SWE Process, SWE Management</td>
<td>Partial</td>
<td>Very Small Enterprise</td>
<td>Evaluation research</td>
<td>No</td>
<td>IEEE</td>
</tr>
<tr>
<td>(Giardino et al., 2016)</td>
<td>Design and creation</td>
<td>Model</td>
<td>SWE Models and Methods</td>
<td>Full</td>
<td>Startup</td>
<td>Solution proposal</td>
<td>No</td>
<td>Institute of Electrical and Electronics Engineers Inc.</td>
</tr>
<tr>
<td>(Bajwa et al., 2016)</td>
<td>Case study</td>
<td>Lessons learned</td>
<td>SWE Management, SWE Requirements</td>
<td>Full</td>
<td>Startup</td>
<td>Experience paper</td>
<td>No</td>
<td>Springer International</td>
</tr>
<tr>
<td>(Nguyen-Dac et al., 2016)</td>
<td>Case study</td>
<td>Model</td>
<td>SWE Methods and Models, SWE Management</td>
<td>Full</td>
<td>Startup</td>
<td>Solution proposal</td>
<td>No</td>
<td>IEEE</td>
</tr>
<tr>
<td>(Duc and Abrahamsson, 2017)</td>
<td>Case study</td>
<td>Advice</td>
<td>SWE Management, SWE Process</td>
<td>Full</td>
<td>Startup</td>
<td>Solution proposal</td>
<td>No</td>
<td>International Conference on Evaluation and Assessment in SWE</td>
</tr>
<tr>
<td>(Bajwa et al., 2017)</td>
<td>Case study</td>
<td>Lessons learned</td>
<td>SWE Management, SWE Testing</td>
<td>Full</td>
<td>Startup</td>
<td>Experience paper</td>
<td>No</td>
<td>Springer</td>
</tr>
<tr>
<td>(Nguyen-Dac et al., 2017)</td>
<td>Case study</td>
<td>Theory</td>
<td>SWE Management, SWE Process</td>
<td>Full</td>
<td>Startup</td>
<td>Solution proposal</td>
<td>No</td>
<td>Springer, Cham</td>
</tr>
<tr>
<td>(Nguyen-Dac et al., 2017)</td>
<td>Case study</td>
<td>Lessons learned</td>
<td>SWE Management, SWE Design</td>
<td>Full</td>
<td>Startup</td>
<td>Solution proposal</td>
<td>No</td>
<td>Springer, Cham</td>
</tr>
<tr>
<td>(Pompermaier et al., 2017)</td>
<td>Case study</td>
<td>Lessons learned</td>
<td>SWE Process, SWE Testing</td>
<td>Full</td>
<td>Startup</td>
<td>Experience paper</td>
<td>Yes</td>
<td>Knowledge Systems Institute Graduate School</td>
</tr>
<tr>
<td>(Panttiuchina et al., 2017)</td>
<td>Survey</td>
<td>Lessons learned</td>
<td>SWE Professional Practice</td>
<td>Full</td>
<td>Startup</td>
<td>Experience paper</td>
<td>No</td>
<td>Springer</td>
</tr>
<tr>
<td>(Raliq et al., 2017)</td>
<td>Case study</td>
<td>Lessons learned</td>
<td>SW Requirements</td>
<td>Full</td>
<td>Startup</td>
<td>Experience paper</td>
<td>No</td>
<td>IEEE Computer Society</td>
</tr>
<tr>
<td>(Souza et al., 2017)</td>
<td>Case study</td>
<td>Model</td>
<td>SWE Professional Practice</td>
<td>Full</td>
<td>Startup</td>
<td>Solution proposal</td>
<td>Yes</td>
<td>Institute of Electrical and Electronics Engineers Inc.</td>
</tr>
<tr>
<td>(Chicote, 2017)</td>
<td>Framework</td>
<td>SWE Quality</td>
<td></td>
<td>Full</td>
<td>Startup</td>
<td>Opinion paper</td>
<td>No</td>
<td>IEEE Computer Society</td>
</tr>
<tr>
<td>(Chanin et al., 2017)</td>
<td>Case study</td>
<td>Lessons learned</td>
<td>SW Requirements</td>
<td>Full</td>
<td>Startup</td>
<td>Evaluation research</td>
<td>No</td>
<td>IEEE</td>
</tr>
<tr>
<td>(Marks et al., 2017)</td>
<td>Case study</td>
<td>Advice</td>
<td>SWE Process</td>
<td>Full</td>
<td>Start-up</td>
<td>Experience paper</td>
<td>No</td>
<td>Springer, Cham</td>
</tr>
</tbody>
</table>

Table A.1: Classification Schema
# A.1 Research Methods

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

Table A.2: Research Methods (Oates, 2005)
### A.2 Contribution Types

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

Table A.3: Contribution Types (Paternoster et al., 2014)
# Appendix B

## Quality Assessment

<table>
<thead>
<tr>
<th>Study</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
<th>Q6</th>
<th>Q7</th>
<th>Q8</th>
<th>Q9</th>
<th>Q10</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Yau and Murphy, 2013)</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>(Laporte et al., 2014)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>(Laporte et al., 2015)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>(Yli-Huumo et al., 2015)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>(Edison et al., 2015)</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>(Nguyen-Duc et al., 2015b)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>(Sánchez-Gordón and O’Connor, 2016)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>(Laporte and O’Connor, 2016)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>(Giardino et al., 2016)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>(Pompermaier et al., 2017)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>(Pantiuchina et al., 2017)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>(Rafiq et al., 2017)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>(Souza et al., 2017)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>(Marks et al., 2017)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>(Bajwa et al., 2016)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>(Nguyen-Duc and Abrahamsson, 2016)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>(Nguyen-Duc et al., 2016)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>(Duc and Abrahamsson, 2017)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>(Bajwa et al., 2017)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>(Nguyen-Duc et al., 2017)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>(Nguyen-Duc et al., 2017)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>(Chanin et al., 2017)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Theme (T)</th>
<th>Theme Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>Economical, human, and physical resources are very scarce or limited.</td>
</tr>
<tr>
<td>T2</td>
<td>Startups can react very fast to changed market conditions, technologies, or changed customer demands.</td>
</tr>
<tr>
<td>T3</td>
<td>The startups focus on innovative market segments, most likely where they can disrupt markets.</td>
</tr>
<tr>
<td>T4</td>
<td>The ecosystem in which the startups operate within are very uncertain, wrt. customers, competition, technologies.</td>
</tr>
<tr>
<td>T5</td>
<td>Startups' objective is to grow and scale rapidly.</td>
</tr>
<tr>
<td>T6</td>
<td>The market and environment demands fast product releases and constant pressure.</td>
</tr>
<tr>
<td>T7</td>
<td>Startups need to rely on external entities and technologies in their lack of time and resources.</td>
</tr>
<tr>
<td>T8</td>
<td>The startup consist of a small number of individuals.</td>
</tr>
<tr>
<td>T9</td>
<td>The startup is only concerned with the development of one product.</td>
</tr>
<tr>
<td>T10</td>
<td>Maximum five years of experience or newly graduated students.</td>
</tr>
<tr>
<td>T11</td>
<td>The company is newly established.</td>
</tr>
<tr>
<td>T12</td>
<td>All individuals in the company have shared responsibility, no high-management.</td>
</tr>
<tr>
<td>T13</td>
<td>The failure rate of startups is high.</td>
</tr>
<tr>
<td>T14</td>
<td>External funding is required, especially in the early-stages.</td>
</tr>
<tr>
<td>T15</td>
<td>There is a lack of organisational culture as the startup is young.</td>
</tr>
</tbody>
</table>

Table B.2: Explanation of themes, based on table 6 in Paternoster et al. (2014).
Bibliography


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

