The Journal of Systems & Software 144 (2018) 255–274

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Controversy Corner

Software startup engineering: A systematic mapping study

Vebjørn Berg[a], Jørgen Birkeland[a,*], Anh Nguyen-Duc[b], Ilias O. Pappas[a], Letizia Jaccheri[a]

[a] *Department of Computer Science, Norwegian University of Science and Technology (NTNU), Sem Sælands vei 9, Trondheim, 7491, Norway*
[b] *Department of Business and IT, University of South-Eastern Norway*

A R T I C L E I N F O

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

A B S T R A C T

Software startups have long been a significant driver in economic growth and innovation. The on-going failure of the major number of startups calls for a better understanding of state-of-the-practice of startup activities. *Objective* With a focus on engineering perspective, this study aims at identifying the change in focus of research area and thematic concepts operating startup research.

A systematic mapping study on 74 primary papers (in which 27 papers are newly selected) from 1994 to 2017 was conducted with a comparison with findings from previous mapping studies. A classification schema was developed, and the primary studies were ranked according to their rigour.

We discovered that most research has been conducted within the SWEBOK knowledge areas software engineering process, management, construction, design, and requirements, with the shift of focus towards process and management areas. We also provide an alternative classification for future startup research. We find that the rigour of the primary papers was assessed to be higher between 2013–2017 than that of 1994–2013. We also find an inconsistency of characterizing startups.

Future work can focus on certain research themes, such as startup evolution models and human aspects, and consolidate the thematic concepts describing software startups.

1. Introduction

Technology-based startups have long been an important driver for global economic growth and competitiveness (Unterkalmsteiner et al., 2016). Software startups, newly created companies producing cutting-edge software technology, have shown to be an important source of software innovation. Despite stories of successful startups, 90 percent of them fail, primarily due to self-destruction rather than competition (Marmer et al., 2011; Crowne, 2002). The failures come from financial and market factors, for example, insufficient funding to operate startups activities, failure in finding product-market fit, and building an entrepreneurial team (Giardino et al., 2015). However, there are also identified unique challenges related to software development and innovation methods (Giardino et al., 2015). 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). Startup researchers have called for a further attention to engineering approaches to support startup activities in all startup evolution stages (Unterkalmsteiner et al., 2016). Previously, most of the research in the field of software engineering has been conducted in relation to the needs and challenges of established companies, first identified by Sutton Jr (2000).

Startups are at the forefront of applying new technologies in practice. From an engineering perspective, developing technology products is challenging as the startup context presents a dynamic and fast-changed environment, making it difficult to adopt prescriptive engineering practices (Giardino et al., 2016). Despite the rapid growth of the population of startups, the research on software engineering in startups is still at an early stage (Unterkalmsteiner et al., 2016).

One of the most extensive literature reviews in the field is the systematic mapping study of Paternoster et al. (2014), reviewing a total of 43 primary studies from 1994 until 2013. This review shows a lack of high quality studies in the field. While a large amount of Software Engineering practices were extracted from startups, the practices were chosen randomly and adopted under the constraints imposed by the startup context. Thus, an updated systematic mapping is required as it will identify the current status in the area and pave the way for more empirical studies examining startups.

Since 2015, we observed an increased focus on software startup research (i.e., the organization of three International software startup workshops (ISSW) in 2016 and 2017, and software startups tracks at

* Corresponding author.
E-mail addresses: vebjorbe@stud.ntnu.no (V. Berg), jorgen.birkeland1@gmail.com, jorgebi@stud.ntnu.no (J. Birkeland), anh.nguyen.duc@usn.no (A. Nguyen-Duc), ilpappas@ntnu.no (I.O. Pappas), letizia.jaccheri@ntnu.no (L. Jaccheri).

https://doi.org/10.1016/j.jss.2018.06.043
Received 26 February 2018; Received in revised form 11 June 2018; Accepted 18 June 2018
Available online 19 June 2018
0164-1212/ © 2018 Elsevier Inc. All rights reserved.

PROFES 2017 and XP 2017 conferences). The previous systematic review has rapidly gained a large amount of citations (Paternoster et al., 2014). While this implies the further growth in software startup research, a revisit on the area can identify how engineering activities in software startups have changed over time. The objective of this mapping study is to provide an updated view on software startup research in order to identify research gaps. Different from the previous mapping studies (Paternoster et al., 2014; Klotins et al., 2015), we aim at synthesizing startup descriptions in research and its associated software engineering knowledge areas. Beside market factors and financial factors, knowledge about engineering factors and how they affect the startup initiatives and development would be helpful for entrepreneurs in understanding their startups' challenges.

We assume that startups perform various types of software engineering activities, as described in SWEBOK (Bourque and Fairley, 2014). We would like to observe how software startup research has evolved and possibly matured in some Software Engineering knowledge areas. SWEBOK is previously used in Klotins et al. (2015), which allow for easy comparisons and make it possible to identify changes in terms of research direction for the last five years.

The research objective leads to the following research questions:

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

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

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

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

2. Background

2.1. Software startups

A startup can be defined as an organization that is challenged by youth and immaturity, with extremely limited resources, multiple influences, and dynamic technologies and markets (Sutton Jr, 2000). More specifically, Coleman and O'Connor (2008a) describe software startups as unique companies that develop software through various processes and without a prescriptive methodology. Others have characterized software startups as modern organizations with little or no operating history, aiming at developing high-tech and innovative products, and rapidly scale their business in extremely dynamic markets (Giardino et al., 2014).

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 requires experimentation of various products in different markets (Ries, 2011). Instead of developing software for a specific client, software startups develop systems which have market-driven requirements, meaning they have no specific customers before their product is released (Chanin et al., 2017; Rafiq et al., 2017).

There exist many processes to manage product development (i.e., processes concerned with *how* to develop a product), like agile and waterfall methods. (e.g., agile and waterfall). However, these processes do not focus on addressing *what* product to develop, which is essential in the startup context where both problems and solutions tend to be poorly understood (Bosch et al., 2013). The high failure rates of software startups are often caused by a lack of customers rather than product development issues (Marmer et al., 2011; Chanin et al., 2017).

2.2. Startup development methodology

Software startups generally develop products in high-potential target markets (Blank, 2013b), without necessarily knowing *what* the customers want (Rafiq et al., 2017). This relates to market-driven software development, which emphasizes the importance of specific requirement elicitation techniques (e.g., prototyping), and time-to-market as key strategic objectives (Rafiq et al., 2017; Nguyen-Duc et al., 2017). In a market-driven context, requirements tend to be (1) invented by the software company, (2) rarely documented (Karlsson et al., 2002), and (3) validated only after the product is released in the market (Rafiq et al., 2017; Carmel, 1994; Dahlstedt, 2003; Keil and Carmel, 1995). As to this, products that don't meet customer needs are common, resulting in failure of new product releases (Alves et al., 2006). Entrepreneurial and customer focused development approaches like (Blank, 2013a; Alvarez and Barney, 2007; Sarasvathy, 2001; Maurya, 2012) have received attention from the research community. The customer development process introduced by Blank (Blank, 2013a) can be divided into four phases: (1) customer discovery, (2) customer validation, (3) customer creation, and (4) company building. A frequently applied entrepreneurship theory among entrepreneurs is Lean Startup, which builds on the principles from Blank. The method has been criticized by researchers for being based on personal experience and opinions rather than empirical evidence, however, concepts from the Lean Startup have attracted considerable attention among practitioners (Bosch et al., 2013; Blank, 2013b).

2.2.1. Lean Startup

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

Key to the BML feedback loop is to do continuous experimentation on customers to test hypotheses. The hypotheses can be tested by building a minimum viable product (MVP), which is the simplest form of an idea, product, or service that can answer the hypotheses. Any feature, process, or effort not directly contributing to answering the hypotheses, is removed. The aim is to eliminate any waste throughout the process. Empirical research has found three main types of MVP usage, including (1) MVP as a design artifact, (2) MVP as a boundary-spanning object, and (3) MVP as a reusable artifact (Nguyen-Duc and Abrahamsson, 2016). MVPs can be used to bridge knowledge gaps within organizations or to provide a mutual understanding between customer input and product design.

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

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

2.3. Software engineering in startups

Software startup engineering can be defined as "the use of scientific, engineering, managerial, and systematic approaches with the aim of successfully developing software systems in startup companies" (Giardino et al., 2016). The degree of process in software development is dependent on system complexity, business risk, and the number of people involved (Wasserman, 2016). The impact of the 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 those unique challenges (Giardino et al., 2014). The need for process depends on the lifecycle stage of the company, divided into four stages (Crowne, 2002).

- 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, the focus is often directed towards product development, rather than focusing on the establishment of rigid processes (Coleman and O'Connor, 2008a).

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 a 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 appear, and more users and complex code require an extended focus on robustness, scalability, performance, and power consumption (Wasserman, 2016). The use of methods like the Lean Startup is one of the reasons why software startups need and sometimes apply their own software engineering practices, which pose challenges when it comes to software engineering. Lean Startup is beneficial for business and product development, but when it comes to software development, a more hybrid approach of agile and lean may provide the most benefits in terms of cost, time, quality, and scope (Yau and Murphy, 2013).

2.4. Existing literature reviews

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). There also exist relevant work related to SMEs (small and medium-sized enterprises) (Tripathi et al., 2016), and VSEs (very small entities), which become more relevant as startups enter more mature lifecycle stages, however, the early stages of startups pose some specific challenges and needs (e.g., little working/operating history).

The first systematic mapping by Paternoster et al. (2014) covered studies up to December 2013, aiming at structuring and analyzing state-of-the-art on software startup research. The conclusion of the paper is that there existed few high-quality studies contributing to the body of knowledge and that there is a need for more studies supporting startups for all lifecycle stages. From a total of 43 primary studies, only 4 papers (Coleman and O'Connor, 2008a; 2008b; 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 published in 2015, classifying 14 primary studies on software startup engineering into 11 of 15 SWEBOK knowledge areas. The paper concludes as Paternoster et al. (2014), that research did not provide support for any challenges or engineering practices in startups, and that available research results were hard to transfer between startups due to low rigour. This was explained by the lack of contextual information in the studies, and how the studies were performed.

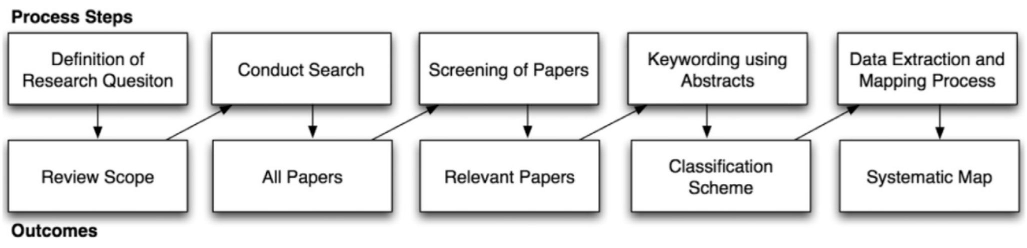

Fig. 1. The systematic mapping study process (Petersen et al., 2008).

Table 1
The searched databases and number of retrievals.

Database	Papers
Scopus	451
ISI web of science	121
Engineering village compendex	875
Total	1447

3. Research methodology

A systematic mapping study was undertaken to provide an overview of the research available in the field of software engineering specific to startups, following guidelines from Kitchenham (2004) and several steps of the standardized process for systematic mapping studies, as illustrated in Fig. 1 (Petersen et al., 2008).

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

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

3.1. Mapping procedure

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

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

Step 3: Additional manual search. A manual search was performed to find more relevant papers. The publication lists of relevant authors were scanned, and the forward snowballing technique was used (Wohlin, 2014). For the forward snowballing, Google Scholar was used to examining the citations of the papers retrieved. This resulted in seven more relevant papers. These were either not published in the databases, or were overlooked in step 2.

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

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

3.2. Data sources and search strategy

The systematic search strategy consisted of searches in three online bibliographic databases (Table 1). The databases were selected from their ability to handle complex search strings, their general use in similar literature reviews in the software engineering community (Paternoster et al., 2014; Tripathi et al., 2016), and the fact that they index the research articles from other databases. In addition, to ensure the best possible coverage of the literature, we performed complementary searches and forward snowballing (Section 3.4 Manual Search). Following guidelines from Wohlin (2014), systematic literature studies should use a combination of approaches for identifying relevant literature, where forward snowballing is found particularly useful. Forward snowballing can reduce systematic errors related to the selection of databases and construction of the search string (Wohlin, 2014). To obtain high-quality data, the following databases were used:

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

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

3.3. Study selection

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

Papers were relevant for inclusion in the study if they met the following criteria: (1) investigate concepts/problems/solutions of engineering in software startups, (2) present contributions in the form of lessons learned, framework, guidelines, theory, tool, model, or advice as applied in Paternoster et al. (2014), (3) are not included in any of the previous mapping studies, and (4) studies are written in English. The papers that were selected are scientific peer-reviewed articles, which is

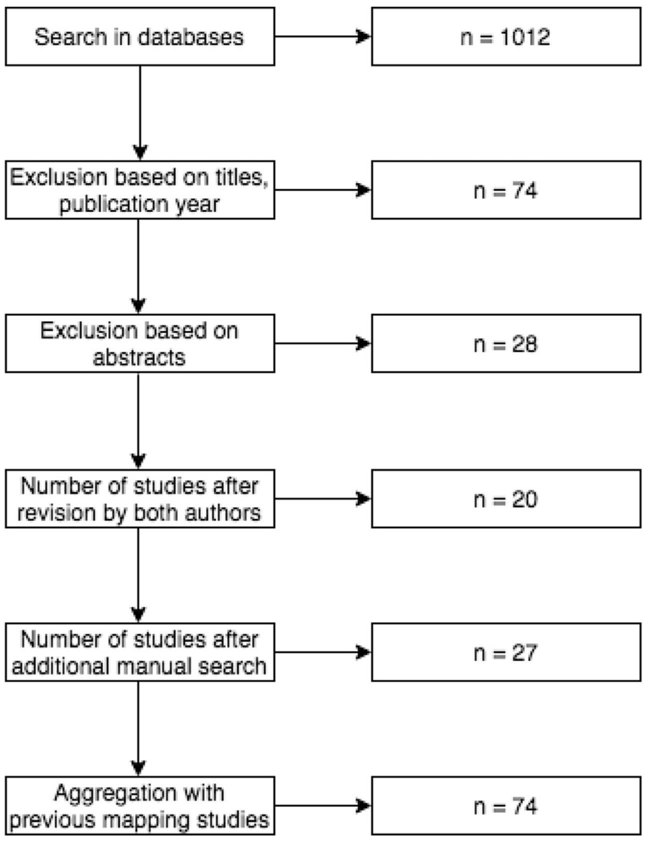
Search in databases
n = 1012
Exclusion based on titles, publication year
n = 74
Exclusion based on abstracts
n = 28
Number of studies after revision by both authors
n = 20
Number of studies after additional manual search
n = 27
Aggregation with previous mapping studies
n = 74

Fig. 2. The study selection process.

independent of the role of authors. We did not find experience reports from entrepreneurs, which might make the sample of papers lean towards the researcher community. To decrease the number of papers into a manageable amount, workshops, and papers based on expert opinion were excluded from the review process.

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

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

3.4. Manual search

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

Table 2
Quality assessment checklist (Nguyen-Duc et al., 2015a).

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

papers, 21 were conference papers, five were journal papers, and one was a book chapter.

3.5. Quality assessment

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

To assess the rigour, credibility, and relevance of the papers, we adopted the quality assessment scheme from Nguyen-Duc et al. (2015a). Quality assessment has been identified as important for performing empirical research in software engineering (Kitchenham, 2004; Ivarsson and Gorschek, 2011). Table 2 illustrates 10 quality evaluation criteria. For each criterion the papers met, they got a score of 1, and otherwise 0. This means that the maximum score a paper could get was 10. A score of 0–3 was regarded as low rigour, 4–6 medium rigour, and 7–10 high rigour. The complete quality assessment can be found in Table B.10.

3.6. Data extraction and synthesis

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

The software engineering book of knowledge (SWEBOK) was created to provide a consistent view of software engineering, and to set the boundary of software engineering with respect to other disciplines (Bourque and Fairley, 2014). SWEBOK contains 15 knowledge areas that characterize the practice of software engineering. The focal point of the paper is to propose research directions based on the knowledge

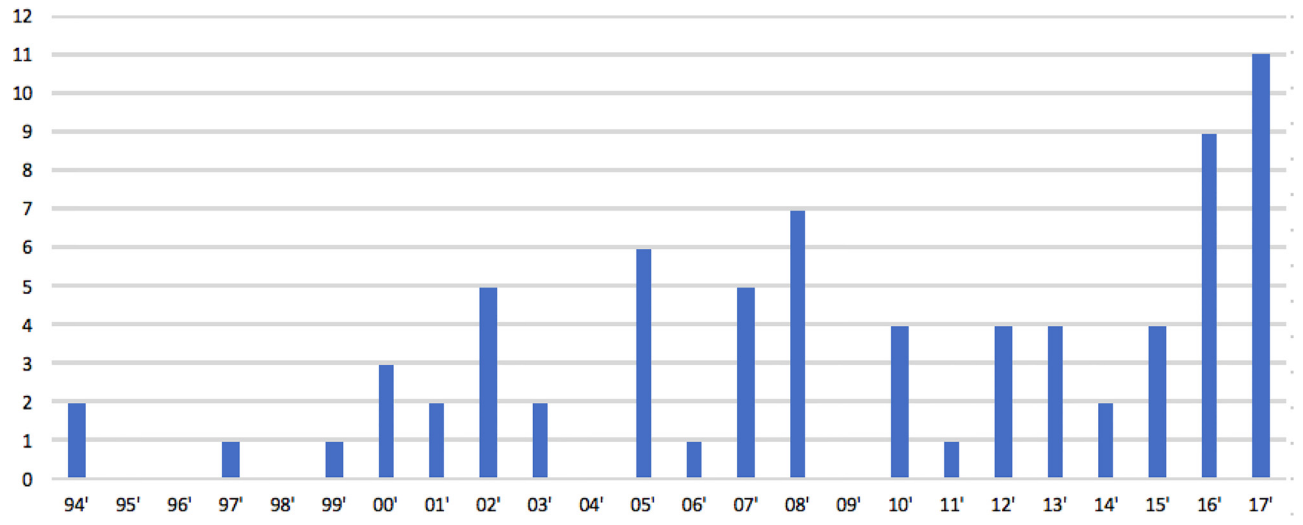

Fig. 3. Publication frequency, 1994–2017.

areas following the work done by existing literature. Since the two major mapping studies in the area follow different approaches, that is SWEBOK (Klotins et al., 2015) and focus facets (Paternoster et al., 2014), in the present study we focus on KAs, as this can allow the reader to better comprehend how the two different approaches are connected, thus offering a more holistic understanding of the current status in software startup engineering research. Assigning each paper into the specific knowledge areas was done by the first and second author. Two researchers read the titles, keywords, abstracts, and the body of each paper, before evaluating the papers' conformance with each specific knowledge area's description or subareas.

3.7. *Threats to validity*

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

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

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

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

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

4. Results

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

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

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

4.1.1. *Publication frequency*

Fig. 3 shows the number of studies published in relation to software startup engineering between 1994–2017, constituting a total of 74 published papers. We observe that the publication frequency of papers between 2013–2017 is higher than for any period before 2013. From 1994–2013, the highest number of primary papers within a single year was 7 (2008). In comparison, 2016 and 2017 constituted 9 and 11 papers respectively. The pertinence of the papers published between 2013–2017 was generally higher than what was found for the period 1994–2013. 85 percent of the papers from 2013 to 2017 had high pertinence, meaning that they were entirely dedicated to software engineering activities in startups. The remaining four papers were focusing on activities of small software companies, and so set to partial pertinence (Table A.7). Although their focus was not entirely dedicated to startups, some of them (Laporte et al., 2014) performed empirical studies on startups, referring to them as "very small entities" or "small software companies". In the period 1994–2013, 57 percent of the papers had high pertinence, while 20 percent had partial pertinence.

Klotins et al. (2015) only includes 4 unique primary papers (Klotins et al., 2018; Kautz, 2000; Jansen et al., 2008; Shakir and Nørbjerg, 2013), as the remaining 10 papers were included among the 43 primary

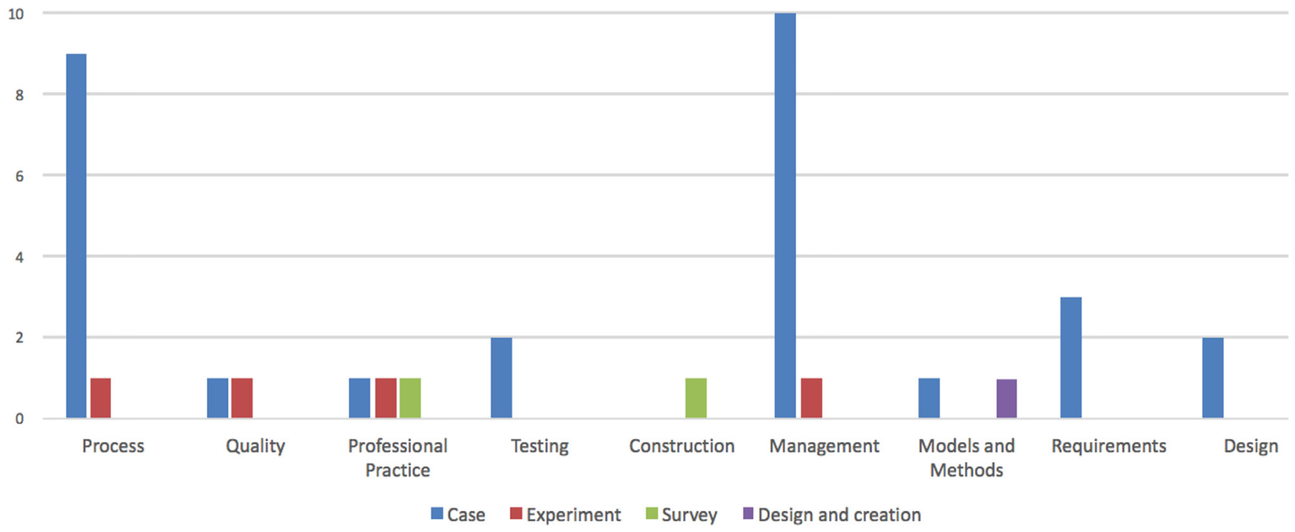

Fig. 4. Empirical evidence, 2013–2017.

papers in Paternoster et al. (2014). The 4 papers are from 1994, 2000, 2008, and 2013.

4.1.2. *Knowledge areas*

The 27 new primary studies were classified into the knowledge areas of SWEBOK (Bourque and Fairley, 2014). The categories were developed by the software community as a baseline for the body of knowledge within software engineering. Fig. 4 illustrates what knowledge areas that have received most attention the last five years, and to what extent empirical studies have been undertaken. The figure shows which research methods that have been used to address each of the knowledge areas. Only the papers providing empirical evidence (22 papers) were included in the figure, covering a total of 9 knowledge areas. Some of the papers covered one or more knowledge areas (e.g., Nguyen-Duc et al., 2015b).

The assessed research methods followed guidelines from Oates (2005), and include (1) survey, (2) design and creation, (3) experiment, (4) case study, (5) action research, and (6) ethnography (Table A.8). Case study was the most frequently used empirical research method (81 percent), followed by experiments (10 percent), surveys (6 percent), and design and creation (3 percent). Action research and ethnography were not used as research methods in any of the primary studies.

Fig. 5 illustrates contribution types (as applied in Paternoster et al. (2014), originally suggested by Shaw (2003)) within each each knowledge area between 2013–2017. The 9 different knowledge areas are represented a total of 49 times through 7 different contribution types. These include (1) model, (2) theory, (3) framework, (4) guidelines, (5) lessons learned, (6) advice, and (7) tools (Table A.9). Lessons learned is the most frequently used contribution type (43 percent), followed by advice (25 percent), model (12 percent), theory (10 percent), framework (5 percent), guidelines (5 percent), and tools (3 percent).

Fig. 6 presents the number of papers that cover the different knowledge areas in our study (red columns) and Klotins et al. (2015) (blue columns). The total number of primary papers in Klotins et al. (2015) was 14. Both mapping studies include papers that cover more than one knowledge area.

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

From Fig. 6, we see that there is a significant change in the research direction for the last five years. Between 1994–2013 “software design” and “software requirements” are the most represented knowledge areas,

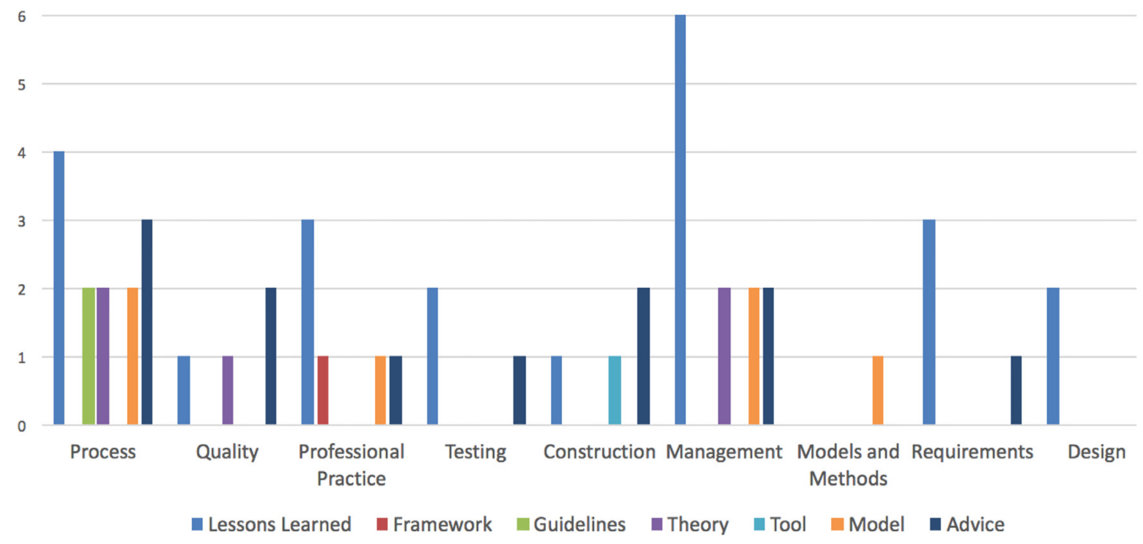

Fig. 5. Contribution types, 2013–2017.

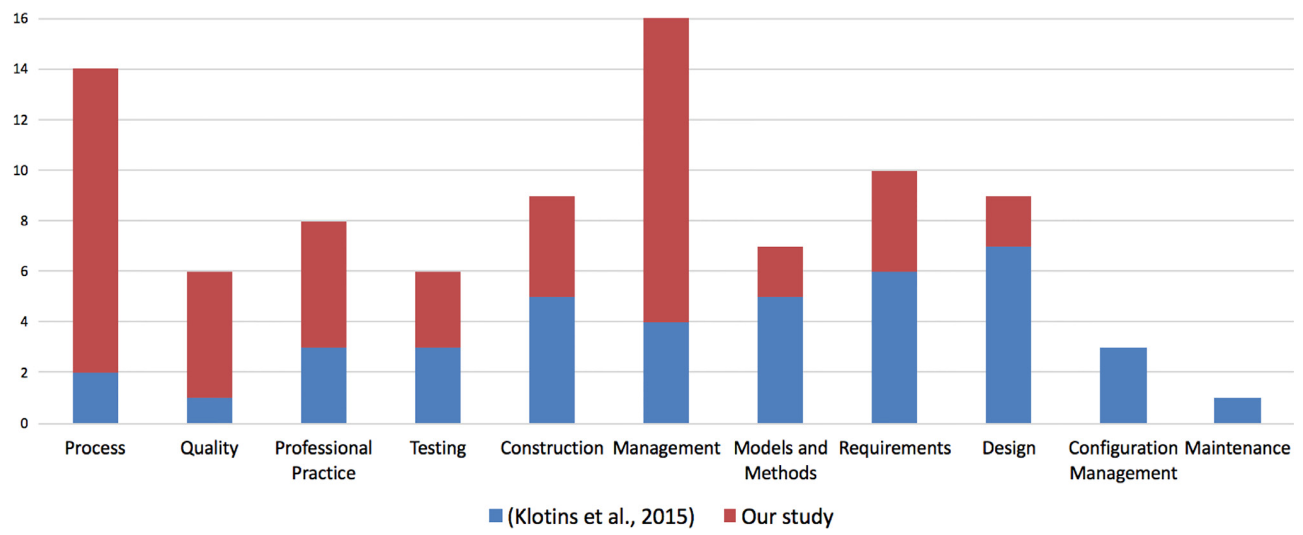

Fig. 6. Knowledge area coverage, 1994–2017.

whereas “software engineering process” and “software management” have received significant attention from the community between 2013–2017. “Software configuration management” and “software maintenance” are only covered between 1994–2013.

Paternoster et al. (2014) did not present any results in relation to the SWEBOK knowledge areas. However, they provided the contribution type of each primary study. Fig. 7 shows the contribution types of primary papers between 1994–2017, separating the periods before and after 2013. The most frequently provided contribution types between 1994–2013 were advice and model, while lessons learned was most represented between 2013–2017. The least frequently used ones combined from both studies were framework, guidelines, and tools.

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

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

4.2.1. Rigour of primary studies 2013–2017

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

Fig. 8 shows the degree of rigour within each knowledge area between 2013–2017. The figure is based on the quality assessment (Table B.10). The x-axis represents the knowledge areas, while the y-axis represents the rigour. Only one paper received low rigour score (Edison et al., 2015), as it didn’t provide enough details about the data analysis and included no assessment of the validity of the results. However, as only the papers providing empirical evidence were assessed, it is possible that more papers would receive low rigour as well. In general, the papers received high rigour score, indicating that the quality of research was high.

4.2.2. Rigour of primary studies 1994–2013

Fig. 9 shows the rigour of the primary studies from Klotins et al. (2015), and which research type each constituted. The paper did not specify how they calculated the rigour of each paper. The

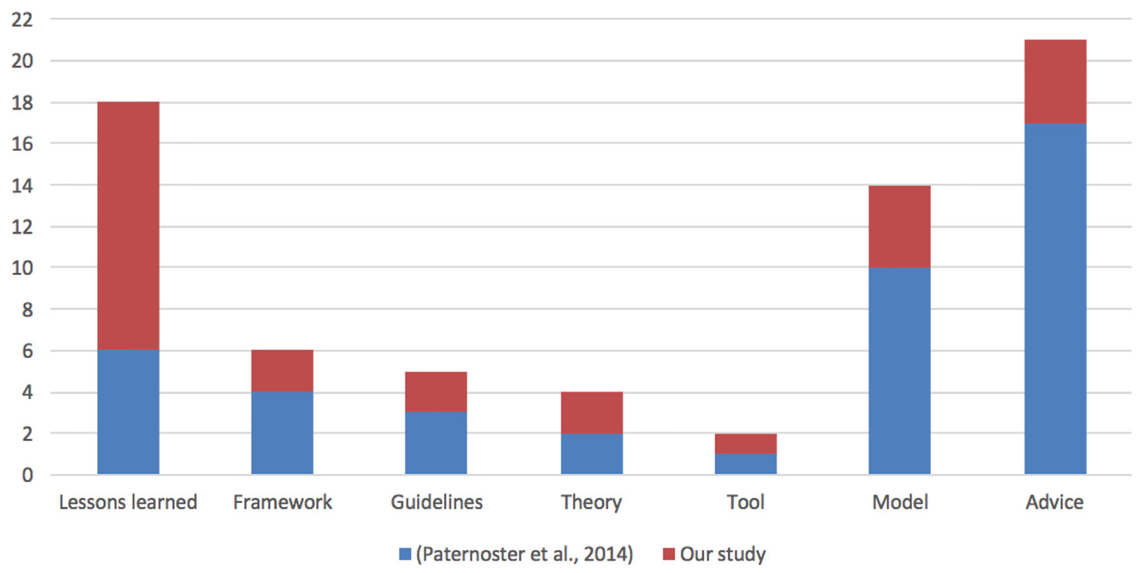

Fig. 7. Contribution types 1994–2017.

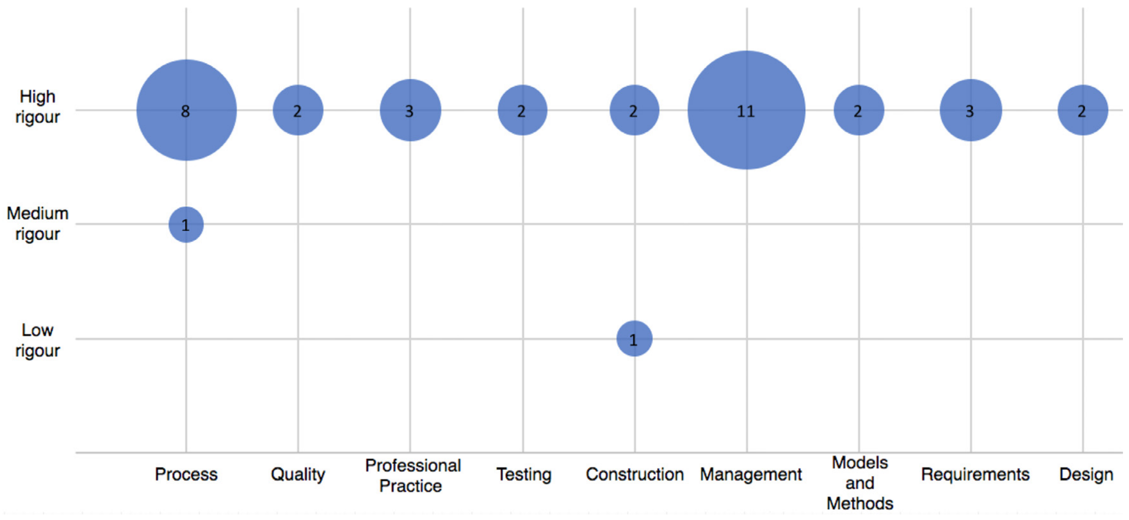

Fig. 8. Rigour of each covered knowledge area, 2013–2017.

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.

Fig. 10 illustrates the rigour of the contribution types provided by each of the primary papers in Paternoster et al. (2014). The x-axis represents the contribution types, and the y-axis represents the rigour. The division of rigour score is based on Table A.7 in the study. Papers that got a total score above 7 received high rigour, between 4 and 7 received medium rigour, while less than 4 received low rigour. 70 percent of the papers in Fig. 10 received a medium score, while 21 got a high score.

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

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

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

4.3.1. Thematic concepts and term frequency

To illustrate how researchers use different definitions and thematic concepts in their characterizations of startups, we extracted the thematic concepts (i.e., referred to as recurring themes in Paternoster et al., 2014) between 1994–2017. Paternoster et al. (2014)

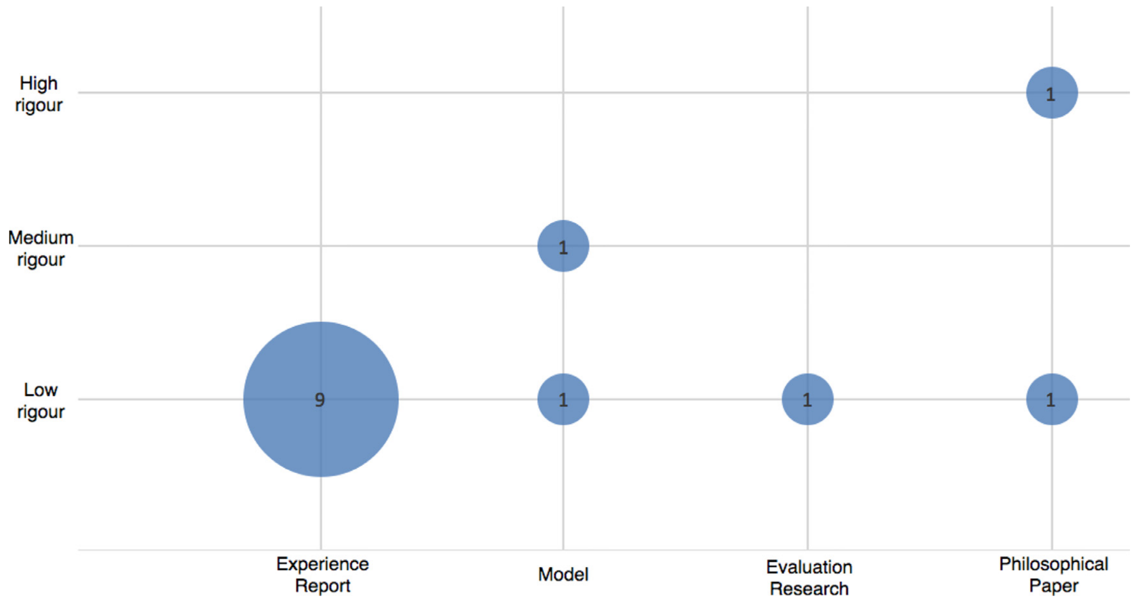

Fig. 9. Rigour and research type (Klotins et al., 2015).

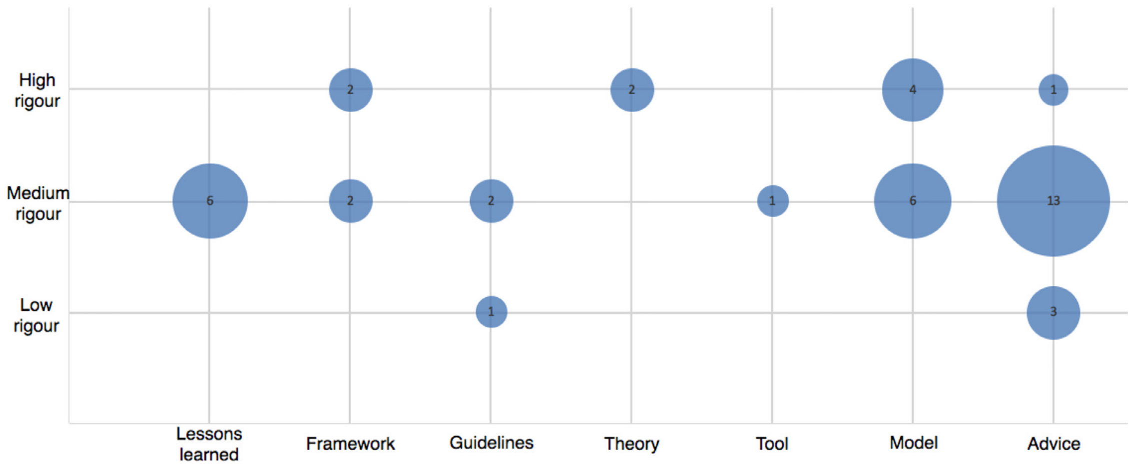

Fig. 10. Rigour and contribution type (Paternoster et al., 2014).

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

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

Table 4 shows the number of primary papers from 1994 to 2017 using the specified terms for "startup company". In situations where the title did not use any of the terms, the abstract was revised. Several papers (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. From the table, it can be observed that the use of terms

Table 3
Thematic concepts, 1994–2017.

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

Table 4
Term frequencies, 1994–2017.

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

for startup companies has changed. The most significant finding is that the term "startup" is more frequently used now than before. 75 percent of the studies from 2013 to 2017 used the term "startup", compared to 48 percent in 1994–2013. 15 percent used the term "start-up" in the period 2013–2017, while 48 percent used the term "start-up" between 1994–2013. The inconsistent use of terms is one of the main challenges for developing a coherent body of knowledge within software startup engineering. Even though 40 studies used the term "startup", the context for which they were used was not the same, or the study context was poorly described.

4.3.2. *Incubated companies*

Fig. 11 shows the percentage of the newly selected empirical papers that have performed research in the context of incubators. That is, mentioning incubators or presenting research on startups that are part of incubator environments. As illustrated, 91 percent of the papers focused on startups outside of incubator context or did not mention this in their description. Two papers focused on incubated startups (Pompermaier et al., 2017; Souza et al., 2017).

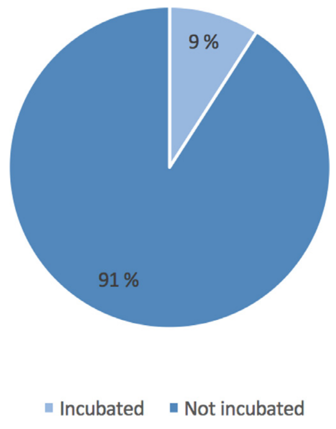

Fig. 11. Percentage of incubated companies, 2013–2017.

Table 5
Contextual descriptions, 2013–2017.

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

4.3.3. *Contextual descriptions*

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

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

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

- The number of startups under investigation is in the range from 1 to 20 startups. The most frequently used number of startups was found to be 3–5.
- The number of employees is usually in the range of 2–25, depending on the lifecycle stage of the company. At early stages, the number of employees tends to be equal to the number of founders, which seems to be in the range of 2–6. At later stages, more employees are needed. For the scaling phase, most companies have 10–20 employees.
- It is usual that researchers specify the product orientation of the startups (e.g., B2B/B2C).
- The age of the investigated companies is usually in the range 1 month to 3 years. Some papers investigated VSEs, one of them 18 years active (Sánchez-Gordón and O’Connor, 2016). Companies beyond three years of age tend to be past the scaling phase.
- Startups use different software development methods. The most usual methods found were agile, scrum, or ad-hoc.
- No more than two papers mentioned whether the investigated companies had received any funding, and if so what kind of funding they had received.
- Even if some of the investigated startups develop products with mixed software and hardware parts, no paper focused on their specific challenges or practices.
- A bootstrap startup is a company that started out without initial funding and resources (Duc and Abrahamsson, 2017).
- “VSEs” and “high growth firms” can in the related studies be

Table 6
Summary of results.

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

regarded as startup companies.

- In relation to the startup stages presented in Section 2.3: (1) Concept and pre startup stage are similar to stage 1. (2) Implementation, functional, and startup stages are similar to stage 2. Commercial and scaling stages are similar to stage 3. (4) Mature stage can be either stage 3 or 4.

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

5. Discussion

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

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

5.1.1. *SWEBOK knowledge areas*

Software engineering process. The need for the software development process to be adapted to a projects 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 (Marks et al., 2017). The situational context consists 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). The situational factors in the reference framework explain the need for startups' own software development processes, and why strictly following the agile methodology is often outside the scope of small startups (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).

Software engineering professional practice. Software engineers need to 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 by 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. It is about the application of knowledge in the engineering discipline, allowing engineers to develop and to maintain software more efficiently and effectively, and help practitioners to adopt professional software engineering principles. As to this, more work should be undertaken to identify the engineering foundations of software developers in startups. Most prior research has focused on the needs of established companies. A possible research area could be to investigate which engineering practices graduates and other engineers should possess if they are to work in a software startup, and how universities and other educational institutions can facilitate learning and other services to support the specific needs of practitioners that are to work in software startups.

Software Engineering Management. Software engineering management concerns about a wide range of different areas, including planning, measuring, coordinating, and reporting activities to support systematic software development and maintenance (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), and 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.

The managerial part of software engineering has been identified by Duc and Abrahamsson (2017), exploring the outsourcing relationship in software startups. They concluded that outsourcing is a feasible option for early-stage startups. The authors are underway to provide a guideline with best practices for outsourcing in startups.

Other papers have focused on principles from Lean Startup, especially the role of pivoting in software startups. This includes why startups pivot (Bajwa et al., 2017), and which pivot 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 can model and measure risks. This further relates to which tools and techniques they should utilize to preserve agility and speed in dynamic environments of high uncertainty. Lean Startup offers entrepreneurs a method to handle such environments, but more empirical evidence is needed to understand how software startups apply this theory in practice so that researchers can develop tailored models and frameworks to reduce business and technical risks.

Software quality. A frequent issue in terms of software quality for startups is technical debt. The development of minimum viable products, and releasing the product as fast as possible, often require the development team to take shortcuts and workarounds. Steve McConnell showed that technical debt can be divided into intentional and unintentional debt McConnell. Shortcuts can lead to the accumulation of intentional technical debt, while unintentional technical debt can happen when business model experimentation is leaved out (Yli-Huumo et al., 2015). No matter how good the idea may seem, not validating the idea with customers could lead to the development of unnecessary features.

Not focusing on technical debt will have consequences for the product quality, while constantly changing and improving the business model will be necessary to stay competitive (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 emphasizes the need for managers to communicate the learning goals of the product precisely so that developers can adjust the quality accordingly. Not finding the correct match between learning goals and quality will often lead to technical debt, waste, or missed learning.

To help startups focus on technical debt, one estimation method is proposed based on Visual Thinking (Chicote, 2017). The technique is based on "duck taping" each part of the code that is developed or fixed in a messy way, to keep an overview of what might cause quality issues in the future. Measuring technical debt is hard, and as the author also concludes, the method needs more empirical evidence as to whether it actually is capable of solving issues related to technical debt.

Software Construction. There exists a wide range of various software tools to speed up the development processes in software startups. However, as Edison et al. (2015) suggest, there does not exist a clear understanding of how entrepreneurs can use the different tools efficiently to meet their specific needs. As to this, the paper describes the outline of a system that provides a software tool portal that supports and recommends which tools to use in the construction of software products and services. The portal can be directly connected to the research agenda (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 use 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.

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 to their specific contexts. These should be based on Lean Startup and agile principles (Paternoster et al., 2014). Researchers are encouraged to identify what engineering methods and models that are used today, and whether they work in a startup context (Unterkalmsteiner et al., 2016).

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 do not adopt traditional development methodologies. Instead, rapid prototyping and continuous experimentation are in focus, hence engineering practices are adapted to each startup's specific context. These models provide development objectives that software engineers in startups can use, as well as guidelines for future research aiming at improving the current state-of-the-art.

We see an existing need to validate the software engineering models adapted to the startup context. This includes areas like technical debt management for particular contexts, and how new models from academia and industry can be applied in the startup context (Giardino et al., 2016).

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

Pompermaier et al. (2017) found that testing is critical to startups' success. However, in the construction phase of the first version of the system, technical teams did not use any software testing techniques. This changed in the following phases, where 75 percent of the technical teams used software testing techniques. The most common testing techniques were unit tests (37 percent), pilot clients (25 percent), functional tests (25 percent), and specialist testers (13 percent).

Due to the importance of testing, startups should apply testing techniques at a more consistent and detailed level to enhance the quality and professionalism of their development processes. Apart from the results presented by Pompermaier et al. (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).

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

Future research should investigate a larger number of software startups to identify a greater amount of elicitation techniques and to provide stronger evidence of the findings in Rafiq et al. (2017). More research should be undertaken to identify negotiation, specification, and validation techniques. In addition, research is necessary to identify requirements engineering for different lifecycle stages to help startups in specific situational contexts identify appropriate requirements engineering techniques.

Software design. The role of MVPs in software startups has been addressed by Nguyen-Duc and Abrahamsson (2016). They suggest that MVPs are effective tools for requirements elicitation, and for bridging knowledge gaps between entrepreneurs, investors, and software developers - emphasizing that MVPs can serve as a multiple facet product. A research topic requiring more work is how software prototype practices can be applied in an agile development context, and how startups can benefit from adopting open source software in prototyping.

The speed of prototyping has been addressed by Nguyen-Duc et al. (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 make it important to define practices and processes to support decision-making in prototyping. While throwaway prototypes are used mainly for specification and experiments, evolutionary prototypes provide a basis for complete systems, usually developed with extensive reuse. Customer feedback is an essential part of business experimentation and is mainly done through prototyping. More work is required to identify what kinds of learning different prototypes provide and to identify effective prototyping and development patterns among software startups.

5.1.2. Startup research 1994–2017

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

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

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

Areas with numerous contributions include software engineering process, software engineering management, software construction, software design, and software requirements. Management was suggested by Klotins et al. (2015) as a potential area for future work. Recently, several papers have contributed to important managerial aspects like pivoting, experimentation, and the role of prototypes to define and assess business and development scope. It is clear that the startup context requires fast and effective decision-making, both at a managerial and technical level. Software requirements engineering is important to manage in order to minimize time and costs and avoid feature creeps related to prototyping and business experimentation.

Another area not sufficiently covered between 1994–2013 was software engineering process. This is in contrast to the last five years, where process has received most contributions. Klotins et al. (2015) argue that software engineering process becomes relevant for the maturity phase when product development is more robust and processes more predictable. As to this, the software process knowledge area is more relevant for SMEs. In our study, however, we have regarded process as relevant for early-stage development as well, illustrating the different interpretation among researchers.

The publication frequency of primary papers between 1994–2017 indicates that increasingly more papers are published. No other year is more represented than 2017, which indicates that there is an increased focus on research within the field. This can be seen as a direct response to the research agendas (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 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 work is undertaken to address startups' unique needs.

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

5.1.3. Identification of research gaps

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

Paternoster et al. (2014) expected more studies to contribute to the adoption of agile practices in startups. In particular, they recommended the need for future studies to provide techniques for aligning business goals of software startups with the execution of specific development practices. Another area suggested for future research was the development of customer collaboration processes for requirement elicitation, allowing for testing the problem before releasing the product to market. Lastly, they identified the need for improved verbal communication with the introduction of new tools and techniques to enhance knowledge transfer in startups. These research gaps have only partly been addressed the last five years. Towards agile methodologies and techniques tailored to the startup context Pantiuchina et al. (2017) provide a better understanding of the current adoption of agile practices in software startups. The study indicates that speed-related agile practices are more frequently used than quality-related practices. Comparable findings have been presented by Yau and Murphy (2013), stating that a rigorous agile methodology intended for established companies may not be applicable to the startup environment. In relation to customer development Chanin et al. (2017) present the results from applying a customer development process to three startups, indicating that the process can improve the requirements process. Others have also contributed to addressing customer development and requirements elicitation (Rafiq et al., 2017; Nguyen-Duc and Abrahamsson, 2016; Bajwa et al., 2016). We could not identify research directly related to team communication, documentation, or knowledge transfer.

Klotins et al. (2015) stated that there is an insufficient understanding of quality requirements role in software startups, and that maintenance of product integrity in startups is yet to be explored. This is especially relevant due to the evolutionary approach and restricted resources of startups. Similar to Paternoster et al. (2014) and Klotins et al. (2015) highlight the need for addressing the relation between software technical decisions and organizational business goals, and a better understanding for human capabilities in startups. Comparable to the need for customer development processes presented by Paternoster et al. (2014) and Klotins et al. (2015) identified the need to further investigate the role of scope in software startups. Discovering the right scope can greatly improve development speed, by identifying the necessary features and effort. Since 2013 empirical research has been undertaken to explore state-of-the-practice in testing activities of software startups (Pompermaier et al., 2017), and the accumulation of technical debt (Yli-Huumo et al., 2015; Terho et al., 2016; Chicote, 2017).

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

5.1.4. Future classification

Unterkalmsteiner et al. (2016) identified more than 70 research questions in different areas supporting activities of software startups. The researchers contributing to the paper are all part of a network (The Software Startup Research Network) of researchers that have created eighteen research track descriptions to ease the presentation and discussion of the research agenda. These eighteen research tracks were grouped into six themes based on similarities.

Classifying the newly selected primary studies according to the SWEBOK knowledge areas resulted in 9 out of 15 of the areas being addressed. This indicates that some of the knowledge areas might not be related to startups, while some are of big interest. The same pattern was discovered in Klotins et al. (2015), which used SWEBOK for lack of a better alternative. The low coverage is most certainly because most of the research in the field of software engineering is undertaken in relation to established companies, from which the SWEBOK knowledge areas are developed.

For future mappings, it would be sensible to categorize the papers into the newly established research themes (Unterkalmsteiner et al., 2016). These are better suited for startup research and can help guide researchers in providing knowledge to the specific areas that are most important for the challenges faced by startups. Do notice that number 7 and 8 require more evidence as to whether they can be related to software startup engineering: (1) Supporting startup engineering activities, (2) Startup evolution models and patterns, (3) Human aspects in software startups, (4) Applying startup concepts in non-startup environments, (5) Startup ecosystem and innovation hubs, (6) Methodologies and theories for startup research, (7) Marketing, (8) Economics and business development.

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

Paternoster et al. (2014) provided a mapping of the research within software development in startups for the period 1994–2013, including 43 primary studies. Each study provided empirical evidence as this was a quality criterion of the mapping. Overall, only 4 of these papers were found to be (1) contributions entirely dedicated to engineering activities in startups, (2) providing a strong contribution type, and (3) conducted through an evidence-based research approach (Coleman and O'Connor, 2008a; 2008b; 2007; Kajko-Mattsson and Nikitina, 2008).

The mapping study Klotins et al. (2015) found that (1) most of their primary studies did not compare and analyze data from more than one case, and (2) most studies had low rigour, making it difficult to compare results. As to this, they emphasized the need for more empirical research to provide stronger evidence and enable results to be generalized to all software startups. More specifically, the paper identified a lack of studies related to requirements processes, the "developer's dilemma" (as discussed in Section 5.1.1), software architecture, and software engineering processes.

Fig. 4 shows the areas that have received most contributions in terms of empirical research between 2013–2017. These are software engineering process, software engineering management, and software engineering professional practice. On the contrary, five knowledge areas received less than five scientific contributions, arguing the need for more research, even within areas that have gotten attention from the research community. This is also in line with the research agenda's addressed need for further empirical evidence (Unterkalmsteiner et al., 2016).

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

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

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

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

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

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

Many of the newly selected primary studies did not explain the startups under investigation sufficiently. From the 22 papers providing empirical evidence, only 14 of these provided a sufficient amount of descriptions (Table 5). Interestingly, only two papers mentioned incubators as part of the startup context. Without a unique definition in literature, the importance of precise descriptions becomes even bigger, especially for transferring results from one environment to another (Klotins et al., 2015).

“Team size” received little focus. A startup with 5 employees have different needs and challenges from a startup with more than 150 employees (e.g., communication needs) (Deias et al., 2002). Even though team size will affect engineering practices to a large degree, too few of the primary papers presented the team size of the startups investigated. More research is needed to understand how software engineering practices change according to team size, and to what extent team size should be part of a unique definition of startups. We found that the usual number of employees in investigated startups was 2–25. The number depends on their respective lifecycle stage or the age of the company. A startup usually starts with 2–6 founders, but as the business scales, more employees are required. This will, in turn, affect the startup’s need for software processes. As to this, we emphasize that researchers must be aware of which describing concepts that are relevant for the startups they are investigating, and that they specify this in their work. More consistent focus on the situational context is a vital step towards a more coherent body of knowledge.

The research track of Unterkalmsteiner et al. (2016) aims at developing a software startup context model that would allow a coherent characterization of software startups. Since there is no agreement on a standard definition, it is challenging to provide coherent contributions to the research area.

6. Conclusion

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

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

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

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

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

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

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

Appendix A

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

Table A.7
Classification schema.

ID	Research method	Contribution type	Knowledge area	Pertinence	Term for startup	Incubator context	Publisher
Yau and Murphy (2013)	Case study	Lessons learned	Process	Full	Startup	No	Penn University
Laporte et al. (2014)	Experiment	Lessons learned	Professional practice, management, quality	Partial	Start-up	No	QUATIC
Eloranta (2014)		Framework	Professional practice	Full	Start-up	No	ACM
Laporte et al. (2015)	Experiment	Guidelines	Process	Full	Start-up	No	ENASE
Yli-Huumo et al. (2015)	Case study	Theory	Process, management, quality	Partial	Startup	No	Springer
Edison et al. (2015)	Survey	Tool	Construction	Full	Startup	No	Springer
Nguyen-Duc et al. (2015b)	Case study	Model	Process, management	Full	Startup	No	ACM
Sánchez-Gordón and O'Connor (2016)	Case study	Lessons learned	Process	Full	Very small company	No	Springer
Wasserman (2016)		Guidelines	Process	Partial	Startup	No	Springer
Unterkalmsteiner et al. (2016)		Advice	Process, management, professional practice, construction, requirements, quality, testing	Full	Startup	No	EISEJ
Terho et al. (2016)		Advice	Quality, construction	Full	Startup	No	Springer
Laporte and O'Connor (2016)	Case study	Lessons learned	Process, management	Partial	Very small enterprise	No	IEEE
Giardino et al. (2016)	Design and creation	Model	Models and methods	Full	Startup	No	IEEE
Bajwa et al. (2016)	Case study	Lessons learned	Management, requirements	Full	Startup	No	Springer
Nguyen-Duc and Abrahamsson (2016)	Case study	Lessons learned	Management, design, construction	Full	Startup	No	Springer
Nguyen-Duc et al. (2016)	Case study	Model	Methods and models, management	Full	Startup	No	IEEE
Duc and Abrahamsson (2017)	Case study	Advice	Management, process	Full	Startup	No	EASE
Bajwa et al. (2017)	Case study	Lessons learned	Management, testing	Full	Startup	No	Springer
Nguven-Duc et al. (2017)	Case study	Theory	Management, process	Full	Startup	No	Springer
Nguyen-Duc et al. (2017)	Case study	Lessons learned	Management, design	Full	Startup	No	Springer
Pompermaier et al. (2017)	Case study	Lessons learned	Process, testing	Full	Startup	Yes	KSI Graduate School
Pantiuchina et al. (2017)	Survey	Lessons learned	Professional practice	Full	Startup	No	Springer
Rafiq et al. (2017)	Case study	Lessons learned	Requirements	Full	Startup	No	IEEE
Souza et al. (2017)	Case study	Model	Professional practice	Full	Startup	Yes	IEEE
Chicote (2017)		Framework	Quality	Full	Startup	No	IEEE
Chanin et al. (2017)	Case study	Lessons learned	Requirements	Full	Startup	No	IEEE
Marks et al. (2017)	Case study	Advice	Process	Full	Start-up	No	Springer

Research methods

Table A.8
Research methods (Oates, 2005).

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

Contribution types

Table A.9
Contribution types (Paternoster et al., 2014).

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

Appendix B

Table B.10
Quality assessment 2013–2017, based on Table 3 in Nguyen-Duc et al. (2015a).

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

Table B.11
Explanation of recurring themes, based on Table 6 in Paternoster et al. (2014).

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

References

Alvarez, S.A., Barney, J.B., 2007. Discovery and creation: alternative theories of entrepreneurial action. Strat. Entrepreneurship J. 1 (1–2), 11–26.
Alves, C., Pereira, S., Castro, J., 2006. A Study in Market-driven Requirements Engineering, Anais do WER06 - Workshop em Engenharia de Requisitos, Rio de Janeiro, RJ, Brasil, Julho 13-14, 2006.
Bajwa, S.S., Wang, X., Duc, A.N., Abrahamsson, P., 2016. How do software startups pivot? empirical results from a multiple case study. Proceedings of the International Conference of Software Business. Springer, pp. 169–176.
Bajwa, S.S., Wang, X., Duc, A.N., Abrahamsson, P., 2017. "Failures" to be celebrated: an analysis of major pivots of software startups. Empir. Softw. Eng. 22 (5), 2373–2408.
Blank, S., 2013a. The Four Steps to the Epiphany: Successful Strategies for Products that Win. BookBaby.
Blank, S., 2013b. Why the lean start-up changes everything. Harv. Bus. Rev. 91 (5), 63–72.
Bosch, J., Olsson, H.H., Björk, J., Ljungblad, J., 2013. The early stage software startup development model: a framework for operationalizing lean principles in software startups. Lean Enterprise Software and Systems. Springer, pp. 1–15.
Bourque, P., Fairley, R.E., 2014. Guide to the Software Engineering Body of Knowledge (SWEBOK (r)): Version 3.0. IEEE Computer Society Press.
Carmel, E., 1994. Time-to-completion in software package startups. Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences.
Chanin, R., Pompermaier, L., Fraga, K., Sales, A., Prikladnicki, R., 2017. Applying customer development for software requirements in a startup development program. Proceedings of the 1st International Workshop on Software Engineering for Startups. IEEE Press, pp. 2–5.
Chicote, M., 2017. Startups and technical debt: Managing technical debt with visual thinking. Proceedings of the IEEE/ACM 1st International Workshop on Software Engineering for Startups (SoftStart). IEEE Computer Society, pp. 10–11. http://dx.doi.org/10.1109/SoftStart.2017.6.
Chorev, S., Anderson, A.R., 2006. Success in israeli high-tech start-ups; critical factors and process. Technovation 26 (2), 162–174.
Clarke, P., O'Connor, R.V., 2012. The situational factors that affect the software development process: towards a comprehensive reference framework. Inf. Softw. Technol. 54 (5), 433–447.
Coleman, G., O'Connor, R.V., 2008a. An investigation into software development process formation in software start-ups. J. Enterprise Inf. Manag. 21 (6), 633–648. http://dx.doi.org/10.1108/17410390810911221.
Coleman, G., O'Connor, R., 2007. Using grounded theory to understand software process improvement: a study of irish software product companies. Inf. Softw. Technol. 49 (6), 654–667.
Coleman, G., O'Connor, R., 2008b. Investigating software process in practice: a grounded theory perspective. J. Syst. Softw. 81 (5), 772–784.
Crowne, M., 2002. Why software product startups fail and what to do about it. evolution of software product development in startup companies. Proceedings of the IEEE International Engineering Management Conference (IEMC). IEEE, pp. 338–343.
Dahlstedt, A., 2003. Study of current practices in market-driven requirements engineering. Proceedings of the Third Conference for the Promotion of Research in IT at New Universities and University Colleges in Sweden.
Deias, R., Mugheddu, G., Murru, O., 2002. Introducing xp in a start-up. Proceedings 3rd International Conference on eXtreme Programming and Agile Processes in Software Engineering (XP). pp. 62–65.
Duc, A.N., Abrahamsson, P., 2017. Exploring the outsourcing relationship in software startups: a multiple case study. Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering. ACM, pp. 134–143.
Edison, H., Khanna, D., Bajwa, S.S., Brancaleoni, V., Bellettati, L.U., 2015. Towards a software tool portal to support startup process. Proceedings of the 16th International Conference on Product-Focused Software Process Improvement, PROFES. Springer Verlag, pp. 577–583. http://dx.doi.org/10.1007/978-3-319-26844-6_43.
Eloranta, V.-P., 2014. Towards a pattern language for software start-ups. Proceedings of the 19th European Conference on Pattern Languages of Programs, EuroPLoP. Association for Computing Machineryhttp://dx.doi.org/10.1145/2721956.2721965.
Giardino, C., Bajwa, S.S., Wang, X., Abrahamsson, P., 2015. Key challenges in early-stage software startups. Proceedings of the International Conference on Agile Software Development. Springer, pp. 52–63.
Giardino, C., Paternoster, N., Unterkalmsteiner, M., Gorschek, T., Abrahamsson, P., 2016. Software development in startup companies: the greenfield startup model. IEEE Trans. Softw. Eng. 42 (6), 585–604. http://dx.doi.org/10.1109/TSE.2015.2509970.
Giardino, C., Unterkalmsteiner, M., Paternoster, N., Gorschek, T., Abrahamsson, P., 2014. What do we know about software development in startups? IEEE Software 31 (5), 28–32. http://dx.doi.org/10.1109/MS.2014.129.
Häsel, M., Kollmann, T., Breugst, N., 2010. It competence in internet founder teams. Bus. Inf. Syst. Eng. 2 (4), 209–217.
ISO, 2017. https://www.iso.org/home.html. Access date: 2017-11-12.
Ivarsson, M., Gorschek, T., 2011. A method for evaluating rigor and industrial relevance of technology evaluations. Emp. Softw. Eng. 16 (3), 365–395.
Jansen, S., Brinkkemper, S., Hunink, I., Demir, C., 2008. Pragmatic and opportunistic reuse in innovative start-up companies. IEEE Softw. 25 (6).
Kajko-Mattsson, M., Nikitina, N., 2008. From knowing nothing to knowing a little: Experiences gained from process improvement in a start-up company. Proceedings of the International Conference on Computer Science and Software Engineering, CSSE. IEEE Computer Society, pp. 617–621. http://dx.doi.org/10.1109/CSSE.2008.1370.
Kakati, M., 2003. Success criteria in high-tech new ventures. Technovation 23 (5), 447–457.
Karlsson, L., Dahlstedt, och Dag, J.N., Regnell, B., Persson, A., 2002. Challenges in market-driven requirements engineering-an industrial interview study. Proceedings of the Eighth International Workshop on Requirements Engineering: Foundation for Software Quality.
Kautz, K., 2000. Improvement in very small enterprisese: does it pay off. Softw. Process Improv. Pr 226 (1988), 209–226.
Keil, M., Carmel, E., 1995. Customer-developer links in software development. Commun ACM 38 (5), 33–44.
Kitchenham, B., 2004. Procedures for performing systematic reviews. Keele UK Keele Univ. 33 (2004), 1–26.
Klotins, E., Unterkalmsteiner, M., Gorschek, T., 2015. Software engineering knowledge areas in startup companies: a mapping study. vol. 210 of lecture notes in business information processing. pp. 245–257. http://dx.doi.org/10.1007/978-3-319-19593-3_22.
Klotins, E., Unterkalmsteiner, M., Gorschek, T., 2018. Software engineering in start-up companies: an analysis of 88 experience reports. Emp. Softw. Eng. 23, 1–35.
Kuhrmann, M., Münch, J., Richardson, I., Rausch, A., Zhang, H., 2016. Managing Software Process Evolution: Traditional, Agile and Beyond-How to Handle Process Change. Springer.
Kuvinka, K., 2011. Scrum and the single writer. Proceedings of Technical Communication Summit. pp. 18–19.
Lai, S.-l., 2010. Chinese entrepreneurship in the internet age: lessons from alibaba. com. World Acad. Sci. Eng. Technol. 72, 405–411.
Laporte, C.Y., O'Connor, R.V., 2016. Implementing process improvement in very small enterprises with iso/iec 29110: a multiple case study analysis. Proceedings of the 10th International Conference on the Quality of Information and Communications Technology (QUATIC). IEEE, pp. 125–130.
Laporte, C.Y., O'Connor, R.V., Ieee, 2014. Systems and software engineering standards for very small entities: implementation and initial results. Proceedings of the 9th International Conference on the Quality of Information and Communications Technology (QUATIC). pp. 38–47. http://dx.doi.org/10.1109/quatic.2014.12.
Laporte, C.Y., O'Connor, R.V., Paucar, L.H.G., 2015. Software engineering standards and guides for very small entities: Implementation in two start-ups. In: Evaluation of Novel Approaches to Software Engineering (ENASE), 2015 International Conference on. IEEE. pp. 5–15.
Marks, G., O'Connor, R.V., Clarke, P.M., 2017. The impact of situational context on the software development process - a case study of a highly innovative start-up organization. In: International Conference on Software Process Improvement and Capability Determination. Springer, Cham. 770. pp. 455–466. http://dx.doi.org/10.1007/978-3-319-67383-7_33.
Marmer, M., Herrmann, B.L., Dogrultan, E., Berman, R., Eesley, C., Blank, S., 2011. Startup Genome Report Extra: Premature Scaling. Startup Genome. 10.
Maurya, A., 2012. Running Lean: Iterate from Plan a to a Plan that Works. O'Reilly Media, Inc.
McConnell, S. 2007. Technical Debt. 10x software development. Blog]. Available at: http://blogs.construx.com/blogs/stevemcc/archive/2007/11/01/technical-debt-2.aspx. Access date: 2018-05-28.
Nguyen-Duc, A., Dahle, Y., Steinert, M., Abrahamsson, P., 2017. Towards understanding startup product development as effectual entrepreneurial behaviors. Proceedings of the International Conference on Product-Focused Software Process Improvement. Springer, pp. 265–279.
Nguyen-Duc, A., Abrahamsson, P., 2016. Minimum viable product or multiple facet product? the role of mvp in software startups. Proceedings of the International Conference on Agile Software Development. Springer, pp. 118–130.
Nguyen-Duc, A., Cruzes, D.S., Conradi, R., 2015a. The impact of global dispersion on coordination, team performance and software quality–a systematic literature review. Inf. Softw. Technol. 57, 277–294.
Nguyen-Duc, A., Seppanen, P., Abrahamsson, P., 2015b. Hunter-gatherer cycle: A conceptual model of the evolution of software startups. Proceedings of the International Conference on Software and Systems Process, ICSSP. Association for Computing Machinery, pp. 199–203. http://dx.doi.org/10.1145/2785592.2795368.
Nguyen-Duc, A., Shah, S.M.A., Ambrahamsson, P., 2016. Towards an early stage software startups evolution model. Proceedings of the 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). IEEE, pp. 120–127.
Nguyen-Duc, A., Wang, X., Abrahamsson, P., 2017. What influences the speed of prototyping? an empirical investigation of twenty software startups. Proceedings of the International Conference on Agile Software Development. Springer, pp. 20–36.
Oates, B.J., 2005. Researching Information Systems and Computing. Sage.
Pantiuchina, J., Mondini, M., Khanna, D., Wang, X., Abrahamsson, P., 2017. Are software startups applying agile practices? the state of the practice from a large survey. Proceedings of the 18th International Conference on Agile Software Development, XP. Springer Verlag, pp. 167–183. http://dx.doi.org/10.1007/978-3-319-57633-6_11.
Paternoster, N., Giardino, C., Unterkalmsteiner, M., Gorschek, T., Abrahamsson, P., 2014. Software development in startup companies: a systematic mapping study. Inf. Softw. Technol. 56 (10). 1200–18. 10.1016/j.infsof.2014.04.014.
Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M., 2008. Systematic mapping studies in software engineering. Proceedings of the EASE. 8. pp. 68–77.
Pompermaier, L., Chanin, R., Sales, A., Fraga, K., Prikladnicki, R., 2017. An empirical study on software engineering and software startups: findings from cases in an innovation ecosystem. Proceedings of the 29th International Conference on Software Engineering and Knowledge Engineering, SEKE. Knowledge Systems Institute Graduate School, pp. 48–51. http://dx.doi.org/10.18293/SEKE2017-115.
Rafiq, U., Bajwa, S.S., Xiaofeng, W., Lunesu, I., 2017. Requirements elicitation techniques applied in software startups. Proceedings of the 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA). IEEE Computer Society.

141–4. 10.1109/SEAA.2017.73.
Ries, E., 2011. The Lean Startup: How Today's Entrepreneurs use Contestant Innovation to Create Radically Successful Businesses. Crown Books.
Sarasvathy, S.D., 2001. Causation and effectuation: toward a theoretical shift from economic inevitability to entrepreneurial contingency. Acad. Manag. Rev. 26 (2), 243–263.
Shakir, S., Nørbjerg, J., 2013. It project management in very small software companies: a case of pakistan. Proceedings of the Americas Conference on Information Systems. pp. 1–8.
Shaw, M., 2003. Writing good software engineering research papers. Proceedings of the 25th International Conference on Software Engineering. IEEE, pp. 726–736.
Shull, F., Singer, J., Sjøberg, D.I., 2007. Guide to Advanced Empirical Software Engineering. Springer.
Souza, R., Malta, K., Almeida, E.S.D., 2017. Software engineering in startups: A single embedded case study. Proceedings of the 1st IEEE/ACM International Workshop on Software Engineering for Startups, SoftStart. Institute of Electrical and Electronics Engineers Inc., pp. 17–23. http://dx.doi.org/10.1109/SoftStart.2017.2.
Stanfill, R., Astleford, T., 2007. Improving entrepreneurship team performance through market feasibility analysis, early identification of technical requirements, and intellectual property support. Proceedings of the American Society for Engineering Education Annual Conference & Exposition.
Sutton Jr, S.M., 2000. Role of process in a software start-up. IEEE Softw. 17 (4), 33–39. http://dx.doi.org/10.1109/52.854066.
Sánchez-Gordón, M.-L., O'Connor, R.V., 2016. Understanding the gap between software process practices and actual practice in very small companies. Softw. Q. J. 24 (3), 549–570.
Tanabian, M., ZahirAzami, B., 2005. Building high-performance team through effective job design for an early stage software start-up. Proceedings of the IEEE International Engineering Management Conference. IEEE, pp. 789–792.
Terho, H., Suonsyrja, S., Systa, K., 2016. The developers dilemma: Perfect product development or fast business validation? Proceedings of the 17th International Conference on Product-Focused Software Process Improvement, PROFES. Springer Verlag, pp. 571–579. http://dx.doi.org/10.1007/978-3-319-49094-6_42.
Tripathi, N., Annanpera, E., Oivo, M., Liukkunen, K., 2016. Exploring processes in small software companies: a systematic review. In: International Conference on Software Process Improvement and Capability Determination. Springer, Cham. pp. 150–165. http://dx.doi.org/10.1007/978-3-319-38980-6_12.
Unterkalmsteiner, M., Abrahamsson, P., Wang, X.F., Anh, N.D., Shah, S., Bajwa, S.S., Baltes, G.H., Conboy, K., Cullina, E., Dennehy, D., Edison, H., Fernandez-Sanchez, C., Garbajosa, J., Gorschek, T., Klotins, E., Hokkanen, L., Kon, F., Lunesu, I., Marchesi, M., Morgan, L., Oivo, M., Selig, C., Seppanen, P., Sweetman, R., Tyrvainen, P., Ungerer, C., Yague, A., 2016. Software startups - a research agenda. E Inf. Softw. Eng. J. 10 (1), 89–123. http://dx.doi.org/10.5277/e-Inf160105.
Wasserman, A.I., 2016. Low Ceremony Processes for Short Lifecycle Projects. Managing Software Process Evolution. Springer, pp. 1–13.
Wohlin, C., 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering. ACM, pp. 38.
Womack, J.P., Jones, D.T., Roos, D., 1990. Machine that changed the world. Simon and Schuster.
Yau, A., Murphy, C., 2013. Is a rigorous agile methodology the best development strategy for small scale tech startups?.
Yli-Huumo, J., Rissanen, T., Maglyas, A., Smolander, K., Sainio, L.-M., 2015. The relationship between business model experimentation and technical debt. Proceedings of the 6th International Conference on Software Business, ICSOB. Springer Verlag, pp. 17–29. http://dx.doi.org/10.1007/978-3-319-19593-3_2.
Yoffie, D.B., Cusumano, M.A., 1999. Building a company on internet time: lessons from netscape. Calif. Manage Rev. 41 (3), 8–28.
Zhou, X., Jin, Y., Zhang, H., Li, S., Huang, X., 2016. A map of threats to validity of systematic literature reviews in software engineering. Proceedings of the 23rd Asia-Pacific Software Engineering Conference (APSEC). IEEE, pp. 153–160.

Vebjørn Berg M.Sc. in Computer Science at NTNU: Norwegian University of Science and Technology.

Jørgen Birkeland M.Sc. in Computer Science at NTNU: Norwegian University of Science and Technology.

Anh Nguyen-Duc is a Associate Professor at the Department of Business and IT, University of Southeast Norway. His research interests include Empirical Software Engineering, Data Mining, Software Startups Research and Cybersecurity.

Ilias O. Pappas is a Marie Skłodowska-Curie Fellow at the Department of Computer Science, NTNU, Norway. His teaching and research activities focus on the areas of social innovation and entrepreneurship, strategic university–industry R&D partnerships, and internet marketing and information technology adoption. He has been a Guest Editor for the journals *Information & Management* and *Information Systems and e-Business Management*. He has published articles in peer-reviewed journals and conferences including *Journal of Business Research, European Journal of Marketing, Computers in Human Behavior, Psychology & Marketing, and Information & Management*. He is also a recipient of an ERCIM fellowship.

Letizia Jaccheri (Ph.D. from Politecnico di Torino, Italy) is Professor at the Department of Computer and Information Science, NTNU, Norway. Jaccheri's research is on: software engineering; entertainment computing; computational creativity; ICT-enabled social innovation. She has published more than 100 papers in International conferences and journals. She has been teaching courses in software engineering at various levels since 1994. She has supervised PhD students, post doctoral students and acted as opponent for national and international defences.