Software Development in Startup Companies: A Systematic Mapping Study

Nicolò Paternoster[a], Carmine Giardino[a], Michael Unterkalmsteiner[a], Tony Gorschek[a], Pekka Abrahamsson[b]

[a] *Blekinge Institute of Technology, SE-371 79 Karlskrona, Sweden*
[b] *Free University of Bolzano-Bozen, I-39100 Bolzano-Bozen, Italy*

Abstract

Context: Software startups are newly created companies with no operating history and fast in producing cutting-edge technologies. These companies develop software under highly uncertain conditions, tackling fast-growing markets under severe lack of resources. Therefore, software startups present an unique combination of characteristics which pose several challenges to software development activities. *Objective:* This study aims to structure and analyze the literature on software development in startup companies, determining thereby the potential for technology transfer and identifying software development work practices reported by practitioners and researchers. *Method:* We conducted a systematic mapping study, developing a classification schema, ranking the selected primary studies according their rigor and relevance, and analyzing reported software development work practices in startups. *Results:* A total of 43 primary studies were identified and mapped, synthesizing the available evidence on software development in startups. Only 16 studies are entirely dedicated to software development in startups, of which 10 result in a weak contribution (advice and implications (6); lesson learned (3); tool (1)). Nineteen studies focus on managerial and organizational factors. Moreover, only 9 studies exhibit high scientific rigor and relevance. From the reviewed primary studies, 213 software engineering work practices were extracted, categorized and analyzed. *Conclusion:* This mapping study provides the first systematic exploration of the state-of-art on software startup research. The existing body of knowledge is limited to a few high quality studies. Furthermore, the results indicate that software engineering work practices are chosen opportunistically, adapted and configured to provide value under the constrains imposed by the startup context.

Keywords: Software Development, Startups, Systematic Mapping Study

1. Introduction

A wide body of knowledge has been created in recent years through several empirical studies, investigating how companies leverage software engineering (SE) [1, 2]. However, research on software development activities in newly created companies is scarce. In the past, very few publications have identified, characterized and mapped work practices in software startups [3] and no structured investigation of the area has been performed. Indeed, none of the systematic literature reviews [4] or mapping studies [5] in software engineering (see the tertiary review by Zhang and Babar [6]) address the startup phenomenon.

Understanding how startups take advantage from work practices is essential to support the number of new businesses launched everyday[1]. New software ventures such as *Facebook*, *Linkedin*, *Spotify*, *Pinterest*, *Instagram*, and *Dropbox*, to name a few, are examples of startups that evolved into successful businesses. Startups typically aim to create high-tech and innovative products, and grow by aggressively expanding their business in highly scalable markets.

Despite many successful stories, self-destruction rather than competition drives the majority of startups into failure within two years from their creation [8]. Software startups face intense time-pressure from the market and are exposed to tough competition, operating in a chaotic, rapidly evolving and uncertain context [9, 10]. Choosing the right features to build and adapting quickly to new requests, while being constrained by limited resources, is crucial to the success in this environment [3].

From a software engineering perspective startups are unique, since they develop software in a context where processes can hardly follow a prescriptive methodology [11]. Startups share some characteristics with other contexts such as small companies and web engineering, and present a combination of different factors that make the development environment different from established companies [12]. Therefore, research is needed to support startups' engineering activities, guiding practitioners in taking decisions and avoiding choices that could easily lead business failure [13, 14].

The goal of this paper is to identify and understand the main contributions of the state-of-art towards software engineering in startups. To this end, we perform a systematic mapping study (SMS) [5, 15] aimed at:

- characterizing the state-of-art research on startups
- understanding the context that characterizes startups
- determining the potential for technology transfer of the state-of-art research on startups
- extracting and analyzing software development work practices used in startups

The systematic map consists of 43 primary studies that were identified from an initial set of 1057 papers. Practitioners may take advantage of the 213 identified software engineering work practices, while considering however the studies' respective

[1] According to a recent study, solely in the US *"startups create an average of 3 million new jobs annually"* [7].

rigor and relevance assessments. Furthermore, this first systematic exploration on software startups provides researchers with directions for future work.

The remainder of this paper is structured as follows: Section 2 describes background and related work; Section 3 illustrates the research methodology and discusses validity threats; Section 4 introduces the classification schema, developed from the gathered data; Section 5 presents the results of the mapping study. The state-of-art of software development in startups is discussed in Section 6, whereas in Section 7 the reported software development work practices are analyzed. Section 8 concludes the paper, answering the posed research questions and providing an outlook for future work.

2. Background and Related Work

Modern entrepreneurship, born more than thirty years ago [16], has been boosted by the advent of the consumer Internet markets in the middle of the nineties and culminated with the notorious dot-com bubble burst of 2000 [17]. Today, with the omnipresence of the Internet and mobile devices, we are assisting to an impressive proliferation of software ventures - metaphorically referred as the startup bubble. In fact, easy access to potential markets and low cost of services distribution are appealing conditions for modern entrepreneurs [18]. Inspired by success stories, a large number of software businesses are created every day. However, the great majority of these companies fail within two years from their creation [8].

2.1. Software Startups

An early account for the term *startup* in the SE literature can be found in Carmel [19] who studied in 1994 the time-to-completion in a young package firm. He noticed how these companies were particularly innovative and successful, advocating the need for more research on their software development practices so as to replicate success and try to transfer it to other technology sectors.

Sutton [3] provides a characterization of software startups, defined by the challenges they are faced with:

- little or no operating history - startups have little accumulated experience in development processes and organization management.
- limited resources - startups typically focus on getting the product out, promoting the product and building up strategic alliances.
- multiple influences - pressure from investors, customers, partners and competitors impact the decision-making in a company. Although individually important, overall they might be inconsistent.
- dynamic technologies and markets - newness of software companies often require to develop or operate with disruptive technologies[2] to enter into a high-potential target market.

[2] A new technology that unexpectedly displaces an established technology. It does not rely on incremental improvements to an already established technology, but rather tackles radical technical change and innovation [20].

One of the objectives of this SMS is to understand the context that characterizes startups and to what extent Sutton's definition from 2000 has been adopted or broadened.

2.2. Startup Lifecycle

The lifetime of a startup company, from idea conception to the maturity level, has been identified and reported from different perspectives (e.g. market [37] and innovation [38]). A prominent contribution, from a SE viewpoint, is the model presented by Crowne [8], who synthesized the startup lifecycle in four stages. The startup stage is the time when startups create and refine the idea conception, up to the first sale. This time frame is characterized most from the need to assemble a small executive team with the necessary skills to start to build the product. The stabilization phase begins from the first sale, and it lasts until the product is stable enough to be commissioned to a new customer without causing any overhead on product development. The growth phase begins with a stable product development process and lasts until market size, share and growth rate have been established. Finally, the startup evolves to a mature organization, where the product development becomes robust and predictable with proven processes for new product inventions.

2.3. Software Development in Startups

The implementation of methodologies to structure and control the development activities in startups is a major challenge for engineers [11]. In general, the management of software development is achieved through the introduction of software processes, which can be defined as "the coherent set of policies, organizational structures, technologies, procedures, and artifacts that are needed to conceive, develop, deploy and maintain a software product" [21]. Several models have been introduced to drive software development activities in startups, however without achieving significant benefits [22, 11, 3].

In the startup context, software engineering (SE) faces complex and multifaceted obstacles in understanding how to manage development processes. Startups are creative and flexible in nature and reluctant to introduce process or bureaucratic measures which may hinder their natural attributes [3, 23]. Furthermore, startups have very limited resources and typically wish to use them to support product development instead of establishing processes [11, 24]. Some attempts to tailor lightweight processes to startups reported basic failure of their application [25]. Rejecting the notion of repeatable and controlled processes, startups prominently take advantage of unpredictable, reactive and low-precision engineering practices [3, 26, 27, 28].

Product-oriented practices help startups in having a flexible team, with workflows that leave them the ability to quickly change the direction according to the targeted market [24, 3]. Therefore, many startups focus on team productivity, asserting more control to the employees instead of providing them rigid guidelines [26, 27, 28].

Startups often develop applications to tackle a high-potential target market rather than developing software for a specific client [18, 29]. Issues related to this market type are addressed

in literature by market-driven software development [30]. In this regard, researchers emphasize the importance of time-to-market as a key strategic objective [31, 32]. In fact, startups usually operate in fast-moving and uncertain markets and need to cope with shortage of resources. Other peculiar aspects influencing software development in the market-driven context are related to requirements, which are reported to be often "invented by the software company" [33], "rarely documented" [34], and can be validated only after the product is released to market [35, 36]. Under these circumstances, failure of product launches are largely due to "products not meeting customer needs" [30].

2.4. Related work

The prospects of evidence-based software engineering [39] have motivated researchers to conduct systematic literature reviews and mapping studies. Zhang and Babar [6] report on 148 SLR's and SMS's published between 2004 and 2010. However, none of these reviews nor the reviews conducted up to February 2014[3], investigated software engineering in the context of startups. Nevertheless, there exist reviews that studied software engineering topics pertinent to specific contexts or environments (as opposed to reviews that investigated individual software engineering technologies, e.g. feature location [40] or search-based software testing [41]) that we consider as related work. Small and medium-sized enterprises (SMEs) and startups possibly share some characteristics, such as the low number of employees (fewer than 250 [42]) and limited resources [43, 44]. Hence, reviews that study the literature on SMEs are relevant related work.

Pino et al. [45] studied the adoption of software process improvement approaches in SMEs [46]. They point out that very few of the SMEs that were part of the reviewed studies did achieve one of the pursued certifications, concluding that standard-driven, not tailored improvement initiatives are not suitable for small companies, confirming also Staples' et al. findings [47]. Taticchi et al. [48] observe a similar situation in the area of business performance measures and management (PMM). Their review identifies a lack of PMM models specifically adapted to SMEs, speculating that non-adoption stems from fear of costs and benefits incomprehension.

Thorpe et al. [49] reviewed the literature on using knowledge within SMEs. Managers/entrepreneurs are an important organizational resource in SMEs as they are drivers for creating knowledge. This knowledge is best encoded in organized routines that allow widespread sharing within the firm. The challenge is to provide enough structure, allowing knowledge creation and sharing to scale, without limiting creativity and learning.

Rosenbusch et al. [50] studied the innovation-performance relationship in SMEs by conducting a meta-analysis of 42 empirical studies that cover 21,270 firms. Interesting to our studied context is their finding that innovation has a stronger impact on younger firms than on more established SMEs. Furthermore, evidence suggests that for small and young firms it is more beneficial to conduct internal innovation projects than seeking innovation by collaborating with external partners.

Common to these reviews, looking at different aspects of SMEs, is the recognition that properties of small firms require solutions and technologies that are adapted to that specific context. Similarly, we argue that startups, differing from SMEs in terms of their operating history, outside influences and market dynamism, require software development solutions adapted to their context. This SMS seeks to evaluate, synthesize and present the empirical findings on software development in startups to date and provide an overview of researched topics, findings, strength of evidence, and implications for research and practice.

3. Research methodology

We chose to perform a systematic mapping study (SMS), which is capable of dealing with wide and poorly-defined areas [15, 4]. A systematic literature review (SLR) [4] would have been a less viable option due to the breadth of our overall research question: *What is the state-of-art in literature pertaining to software engineering in startups?*

The review in this paper follows the guidelines developed by Kitchenham and Charters [4] and implements the systematic mapping process proposed by Petersen et al. [15]. Figure 1 illustrates the adopted process, whereas the individual steps are explained in Subsections 3.1- 3.7. Note that rigor and relevance assessment is an extension attributed to Ivarsson and Gorschek [51] and synthesis is based on the constant comparison method proposed by Strauss and Corbin [52].

The SMS procedure was led by the first and second authors, who performed the steps in Figure 1 in a co-located environment, i.e. working together on a single computer screen. Note-taking during screening of papers and keywording alleviated the resolution of conflicts among the reviewers during data extraction and rigor and relevance assessment. If disagreement persisted, an in-depth review of the paper was performed and, if necessary, the third and fourth authors were consulted to take a final decision.

3.1. Definition of Research Questions

The research question driving this mapping study is: *What is the state-of-art in literature pertaining to software engineering in startups?* To answer this question, we state the following sub-questions:

- RQ1 What is the context that characterizes software development in startups?
- RQ2 To what extent does the research on startups provide reliable and transferable results to industry?
- RQ3 What are the reported work practices in association with software engineering in startups?

[3] We performed an automatic search with the search string published by Zhang and Babar [6].

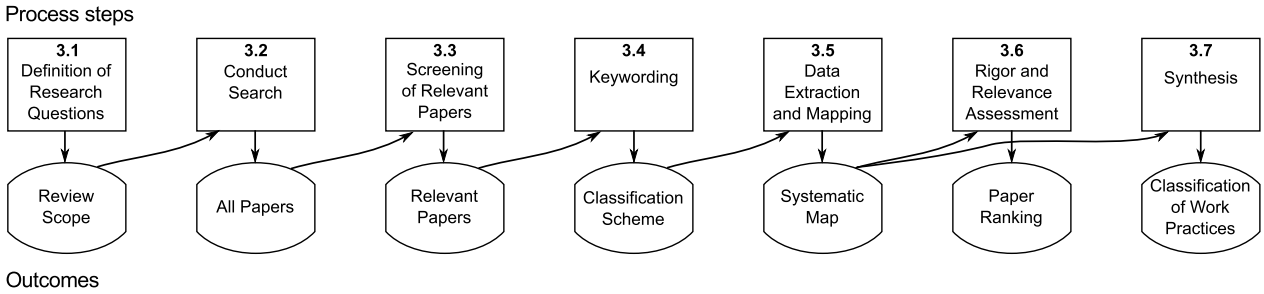

Figure 1: Systematic mapping process (adapted from Petersen et al. [15])

With RQ1 we intend to understand the properties that characterize the nature of software development in startups. Such a characterization illustrates the dimensions by which startups are defined in the state-of-art. With RQ2 we intend to determine the scientific evidence of the results reported in the state-of-art, allowing researchers to identify worthwhile avenues for further work and providing practitioners a tool to navigate within the state-of-art. With RQ3 we intend to identify the software engineering practices applied in startups, providing a basis for determining necessary further research.

3.2. Conduct Search

We identified the primary studies by exercising a search string on scientific databases. The search string is structured according to population, intervention and comparison, as proposed by Kitchenham and Charters [4]. We omitted however the outcome and context facet from the search string structure as our research questions do not warrant a restriction of the results to a particular outcome (e.g. effective/not effective work practices) or context (e.g. startups in a specific product domain).

Table 1 lists the final used keywords. The core concepts, representing population, intervention and comparison, are derived from our research questions. Following Rumsey's guidelines [53], we identified synonymous, related/broader/wider concepts, alternative spelling and part of speech for each core concept. Note that we did not include specific keywords from existing startup definitions (e.g. Sutton [3], discussed in Section 2.1) to the population set of terms as this could have biased the search.

The selected scientific databases on which we performed the search are shown in Table 2, along with the number of publications retrieved from each database (up to December 2013). We selected the databases considering their coverage and use in the domain of software engineering, and their ability to handle advanced queries, following the example of Barney et al. [54].

To increase publication coverage we also used Google Scholar, which indexes a large set of data, both peer and non-peer reviewed. Then, we proceeded to the customization of the search string, adapting the syntax to the particular database[4].

[4]The individual search strings are available in the supplementary material [55].

Table 1: Population, intervention and comparison search string keywords

Core concepts	Terms
Software Startups	software startup*; software start-up*; early-stage firm*; early-stage compan*; high-tech venture*; high-tech start-up*; start-up compan*; startup compan*; lean startup*; lean start-up*; software package start-up*; software package startup*; IT start-up*; IT startup*; software product startup*; software start up*; internet start-up*; internet startup*; web startup*; web start-up*; mobile startup*; mobile start-up*;
Development	develop*; engineer*; model*; construct*; implement*; cod*; creat*; build*;
Strategy	product*; service*; process*; methodolog*; tool*; method*; practice*; artifact*; artefact*; qualit*; ilit*; strateg*; software;

Table 2: Selected databases and retrieved papers

ID	Database	Papers
A	Inspec/Compendex (www.engineeringvillage2.org)	640
B	IEEE Xplore (ieeexplore.ieee.org)	132
C	Scopus (www.scopus.com)	468
D	ISI Web of Science (wokinfo.com)	293
E	ACM Digital Library (dl.acm.org/advsearch.cfm)	78
F	Google Scholar (scholar.google.com)	158
	Total	**1769**

3.3. Screening of Relevant Papers

The criterion that guided the inclusion of a paper was that the study presented a contribution to the body of knowledge on software development in startups. A contribution can be in the form of an experience report, applied engineering practices, development models or lessons learned. We excluded search results that were:

- not peer-reviewed (grey literature, books, presentations, blog posts, etc.)
- not written in English
- clearly obsolete (more than 20 years old)
- related to non-software companies (biotech, manufacturing, electronics, etc.)
- related to established companies (VSE, SME, research spin-offs)

- related to technicalities of startups (algorithms, programming languages, etc.)

For the screening of papers we followed the workflow in Figure 2.

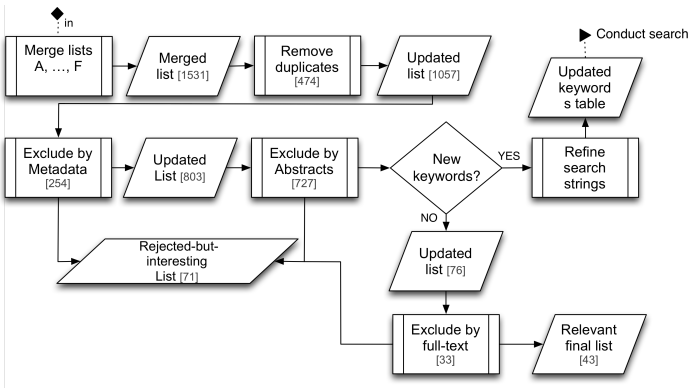

Figure 2: Screening of papers workflow

With the support of a reference management tool [56] we merged the six result lists from the search in the scientific databases. Then, we removed duplicated items in two steps: first we used the reference management tool to automatically detect duplicates based on meta-data (author, publication year and title). Then, we manually deleted instances that were not detected as duplicates by the tool, resulting in 1057 papers.

Then we analyzed the metadata (title, keywords, publication year and type) to identify papers that matched the exclusion criteria, resulting in 803 papers. In a more in-depth review, we analyzed the abstract of each paper, determining whether it matched our inclusion criterion, resulting in 76 papers. As indicated in Figure 2, we improved the search strings while reading the abstracts, adding new keywords identified in retrieved papers and iteratively conducting a new search.

In case of a disagreement among the reviewers or incomplete abstracts we read the entire paper, leading eventually to the final set of 43 primary studies. During the screening process we kept track of the rationale for each exclusion, as shown in Table 3.

Table 3: Rationale for excluded papers

Rationale	Amount
Duplicate	474
Non-software industry	409
Related to software startups but not SE perspective	259
Not software/startup related	122
Related to software but not startups	85
Academic settings	70
Not peer-reviewed	41
Full-text non available	20
Outdated	4
Not in English language	4
Total retrieved	1531
Total excluded	1488
Total included	**43**

3.4. Keywording

The goal of keywording is to efficiently create a classification schema, ensuring that all relevant papers are taken into account [15]. We followed the process illustrated in Figure 3.

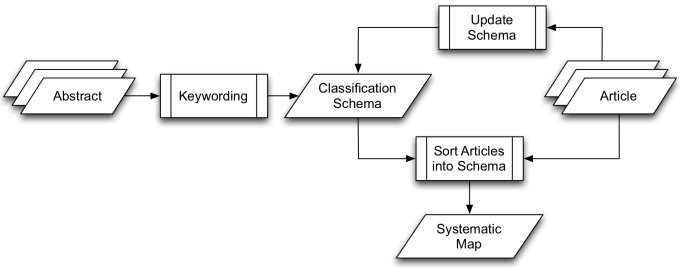

Figure 3: Workflow for classification schema creation (adapted from [15])

The first step consisted in reading the abstracts of the primary studies, assigning them a set of keywords to identify the main contribution area of the paper. Then we combined the keywords forming a high-level set of categories, leading to a rough understanding of the research area represented by the primary studies. By progressively fitting the papers into categories, the schema underwent a refinement process, being continuously updated to account for new data. When performing data extraction and mapping (Subsection 3.5), we annotated the classification with evidence from the respective paper, further refining the schema and sorting. The resulting classification schema is discussed in Section 4 and used in the analysis of the results in Sections 6 and 7.

3.5. Data Extraction and Mapping

After we defined the classification schema, resulting from the keywording process, we proceeded to systematically extract data from the primary studies. For each paper, we filled a spreadsheet, sorting it into the classification schema and extracting the following data, inspired by other similar studies [57, 58]:

- Article title
- First author
- Year of publication
- Synthesis of results (one-line)
- Keywords

We took advantage of the data extraction process to identify an additional relevant aspect which emerged while reading abstracts and the full text: the recurrent patterns of common attributes among startup companies resulted in themes that are reported in Subsection 5.2. Moreover, we screened the bibliography of each paper, identifying other possible relevant studies to our research, adopting the snowballing technique[5] [59].

[5]Note that we didn't identify any additional papers. In case more relevant papers would have been retrieved, a re-iteration of the keywording step would have been necessary (see Subsection 3.4).

3.6. Rigor and relevance assessment

A major challenge of SE is to transfer research results and knowledge to practitioners, showing the findings' validity and concrete advantages [51]. To assess how results are presented in the primary studies, we extended the traditional SMS framework with an additional step, that is, the evaluation of the papers' rigor and relevance (see Figure 1). With this extension we compensate for the SMS' limitation of not assessing the quality of the mapped studies by developing and using a simple ranking function.

We use a systematic and validated model [51] to evaluate the scientific rigor and the industrial relevance of each primary study. The model provides a set of rubrics to measure rigor and relevance, dividing these two factors into different aspects, and quantifying the extent to which each aspect is considered in the study (see Ivarson and Gorschek [60] for an application of the model).

Rigor refers to the precision or exactness of the used research method and how the study is presented. We considered aspects relating to:

- Context - description of development mode, speed, company maturity and any other important aspects where the evaluation is performed.
- Study design - description of the measured variables, treatments, used controls and any other design aspects.
- Validity - description of different types of validity threats.

Relevance refers to the realism of the environment where the study is performed and the degree to which the chosen research method contributes to the potential of transferring the results to practitioners. We considered aspects relating to:

- Subjects - use of subjects who are representative of the intended users of the technology.
- Context - use of settings representative of the intended usage setting.
- Scale - use of a realistic size of the applications.
- Research method - use of a research method that facilitates investigating real situations and relevant for practitioners.

Aspects related to the rigor of the study are scored at three levels: weak (0), medium (0.5) and strong (1). Aspects related to relevance are scored 1 if contributing, 0 otherwise. The detailed rubrics, used to evaluate the studies, can be found in Ivarsson and Gorschek [51]. To obtain the study's final score, we sum the individual scores of the rigor and relevance aspects.

In order to rank the papers, we defined a function incorporating the classification schema, rigor and relevance scores, and two additional factors that characterize the publication type and year. The ranking function provides a rough estimation of the value that a paper provides to practitioners and the research community, giving a stronger weight to recent rigorous journal publications entirely devoted to the topic and presenting empirical results relevant to practitioners. We used tables for converting each factor into an arbitrary numerical value in the range between 0 and 10. The conversion tables used to quantify the internal score of each factor are shown in the supplementary material [55], while the limitations of this approach are discussed in Subsection 3.8.3. The final ranking of the 43 primary studies is discussed in Subsection 6.2.

3.7. Synthesis

In the synthesis we identified the main concepts from each primary study, using the original author's terms in a one line statement. Those main concepts were then organized in tabular form to enable comparisons across studies and translation of findings into higher-order working practices and classification categories. We used the classification categories from Section 4. This process is analogous to the method of constant comparison used in qualitative data analysis [52].

In Section 7 we present the identified work practices, discussing their application in the startup context, their benefits and liabilities, and putting them in perspective with the results of other studies. In summary, synthesis is achieved by:

- Identification of a set of working practices and relative classification categories.
- Documentation of advantages and disadvantages of reported results.
- Elaboration of gaps on the applicability of working practices in startup contexts.

3.8. Threats to validity

We identified potential threats to the validity of the systematic mapping and its results, together with selected mitigation strategies. The structure of this Section follows Unterkalmsteiner et al. [61].

3.8.1. Publication bias

Systematic reviews suffer from the common bias that positive outcomes are more likely to be published than negative ones [4]. This can be observed also in our mapping study which includes few papers on failed startup endeavors and studies, while the success rates of startups is generally rather low. It is unlikely that research is performed only in collaboration with successful startups. Nevertheless, we do not consider this as a major threat as this bias is orthogonal to our study aim, mapping the state-of-art on startup research. Still, this bias takes away some of the possibilities to analyze reported work practices with regard to their performance.

3.8.2. Identification of primary studies

The approach we used to construct the search string (see Subsection 3.2) aimed to be inclusive with respect to the number of retrieved papers, related to software development in startups.

However, a limitation of the current search string lies in the exclusion of the stand-alone terms "startup" and "start-up". These individual terms lead to unmanageable search results (more than 20000 papers) that are mostly irrelevant as they are related to the English phrasal verb "to start up", largely used in many disciplines to indicate the commencing moments of an engine. Therefore, to mitigate the risk of excluding potential relevant primary studies, we constructed a search string containing qualifiers to the term "startup", e.g. "IT startup", and

included synonyms, validating our search strings with the support of librarians specialized in software engineering.

Still, the precision (ratio of retrieved relevant and all retrieved papers [62]) of the used search string is low (43 out of 1057, 4%). However we were not interested in obtaining high precision as much as we aimed to obtain a high recall (ratio of existing relevant papers [62]). The risk of excluding relevant primary studies is further mitigated by the use of multiple databases, which cover the majority of scientific publications in the field.

We were not able to retrieve 20 papers since they were neither available in online catalogs, in the three libraries we consulted, nor on request from the authors. However, this is a minor risk as we had access to their titles, keywords and abstracts, which gave us a good degree of confidence that they were not relevant. Additionally, considering our 4% precision rate, the number of potentially relevant primary studies would be less than 1.

We noticed high inconsistency in the use of the term "startup" by different researchers, even in the same area. For example, Sutton [3] distinguishes startups from established companies by characterizing startups according to their extreme lack of resource, newness and immaturity (in a process sense). On the other hand, Deias et al. [63] define their company as a startup, with more then 150 employees and resources available to certify the quality of their development process. Under these conditions, the attempt to identify a body of knowledge and research scope has been highly challenging. Therefore, we had to identify and analyze multiple and conflicting definitions.

Moreover, several contextual factors, not thoroughly analyzed in this study, can be identified as important. Factors regarding the application domain or the market type could influence the adoption of working practices and processes. However, in this study we compromised details regarding specific context challenges in favor of a general overview of practices reported by primary studies.

Finally, since startups and entrepreneurship in general are appealing for many sectors of the economy, an additional threat lies in the fact that some relevant information can be found in other research areas, such as business innovation and marketing, not considered in this study.

3.8.3. Study selection and data extraction

Threats to study selection and data extraction [57] have been mitigated with an up-front definition of the inclusion/exclusion criteria [4]. The selection of relevant primary studies can be further biased by personal opinions of researchers executing the process. To mitigate this threat, we defined and documented a rigid protocol for the study selection and, by conducting the selection together and dedicating a reasonable amount of time to review conflicts, mutually adjusting each others' biases, as suggested by Kitchenham and Charters [4]. The screening process is threatened by a potential predominance of the opinion of one reviewer over the other, since the first two authors performed the screening process collaboratively at one computer. This threat was mitigated by explicitly recording the exclusion rationale for each paper, requiring clear evidence from the paper

Table 4: Classification schema

(a) Research type facet (adapted from Wieringa [64])

Category	Description
Evaluation Research	The methodology is implemented in practice and an evaluation of it is conducted. That means, it is shown how the research is implemented (solution implementation) and what are the consequences of the implementation in terms of benefits and drawbacks (implementation evaluation). This also includes problems identified in industry.
Solution Proposal	A solution for a problem is proposed. The solution can be either novel or a significant extension of an existing methodology. The potential benefits and the applicability of the solution is shown by a small example or a good line of argumentation.
Philosophical Papers	These papers sketch a new way of looking at existing things by structuring the field in form of a taxonomy or conceptual framework.
Opinion Papers	These papers express the personal opinion of somebody whether a certain technique is good or bad, or how things should have been done. They do not rely on related work and research methodology.
Experience Papers	Experience papers explain what and how something has been done in practice. It has to be the personal experience of the author.

(b) Contribution facet (adapted from Shaw [65])

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

(c) Focus facet

Category	Description
Software development	Engineering activities used to write and maintaining the source code.
Process management	Engineering methods and techniques used to manage the development activities.
Tools and technology	Instruments used to create, debug, maintain and support development activities.
Managerial / Organizational	Aspects that are related to software development, by means of resource management and organizational structure.

(d) Pertinence facet

Category	Description
Full	Entirely related (main focus) to engineering activities in software startups.
Partial	Partially related to engineering activities in software startups. Main research focus related to engineering activities.
Marginal	Marginally related to engineering activities in software startups. Main research focus different from engineering activities.

to support the decision, and supporting the consensus creating process by consulting the history of previously taken decisions.

Another threat is related to researchers personal judgments, which can interfere with the evaluation of rigor and relevance of selected studies. Even though the rigor and relevance model provides guidelines and detailed rubric tables to support objective decisions, the evaluation depends on the reporting quality and not on the intrinsic quality of the study itself [51]. As such, we used the scores only to rank but not to exclude studies from the selection. The ranking itself gives an indication of the study quality, the individual contribution needs however to be qualified by the reader [51].

The validity of the ranking function (see Subsection 3.6) is threatened by the arbitrarily chosen scores for each category and weights for each dimension. To mitigate this threat, we used an automatic spreadsheet to compute the final scores, allowing us to adjust scores and weights, observing the effect of the final ranking in real time. For validating our ranking, we tried to modify scores/weights values several times, and we observed that the final ranking was not significantly altered by numerical adjustments, as long as we kept the ordering of concepts stable.

4. Classification schema

In this section we present the classification schema that is adapted from other existing taxonomies or emerged from the keywording process. The schema consists of four facets:

- Research type: to represent the type of the undertaken study
- Contribution type: to map the different types of the study outcomes
- Focus: to describe the main focus of the research
- Pertinence: to distinguish between studies entirely devoted to engineering activities in startups and the ones that have a broader perspective

The research type facet (Table 4a) is used to distinguish between different types of studies, abstracting from the specific underlying research methodology. The research types were adapted from Wieringa [64].

The contribution facet (Table 4b), similarly to the taxonomy used by Shaw [65], describes the kind of contribution a study provides. Contribution types can be divided into weak (advises and implications, lessons learned, tools and guidelines) and strong (theory, framework/method and model) contributions.

The categories in the focus facet (Table 4c) were obtained by clustering the sets of keywords identified in the keywording process (Subsection 3.4) and abstracting them to four categories. We separated thereby studies concerning software development practices (e.g. writing user stories [13]) from studies focused on higher-level process management (e.g. use Scrum methodology [66]). Furthermore we distinguish between studies focused on specific tools and technologies (e.g. use of post-it notes [67]) and work focused on managerial/organizational aspects in startups (e.g. operate in cross-functional settings [19]).

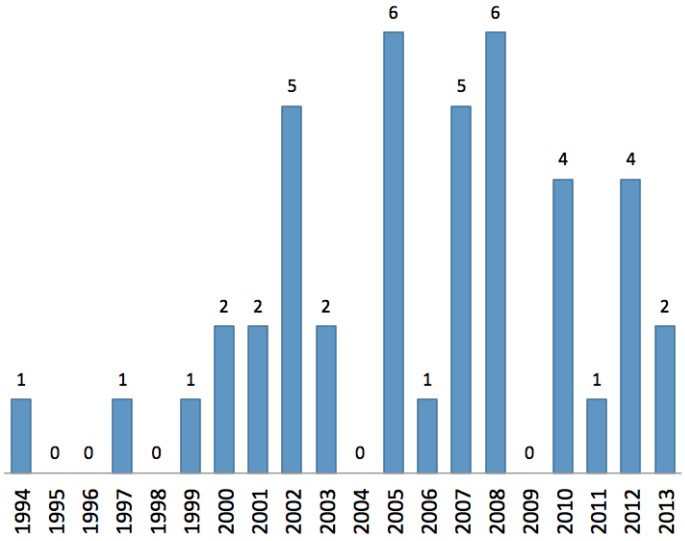

Figure 4: Publication distribution-year

The pertinence facet (Table 4d) distinguishes the levels (full, partial, marginal) on which the study's research focus is directed towards engineering activities in startups.

The classification schema in Table 4 forms the basis for the systematic maps presented and discussed in Section 5.

5. Results

This section presents the results of the systematic mapping study. From an initial sample of 1057 papers, we identified 43 primary studies answering our research questions.

5.1. Startup research categorization

Figure 4 shows the publication years' frequency distribution, from 1994 to 2013.

To characterize the main themes covered within the area of engineering activities in software startups, we used the set of keywords extracted from the study's abstract and author-defined keywords (when available in the metadata). From the 43 primary studies we extracted a total of 346 keywords (125 unique) averaging on about 8 keywords per paper. These formed the basis[6] for the focus and pertinence facet of the classification schema (Section 4). Table 5 applies the classification schema on the primary studies, providing an overview of the field of startup research.

In order to illustrate potential gaps in startup research, we present the systematic map with multi-dimensional bubble charts ("x-y scatter plots with bubbles in categories intersections" [15]), where the size of the bubble is determined by the number of publications corresponding to the x-y coordinates. Differently from other studies (e.g. [97, 98]), each data point is represented by four features. Thus, we created three plots (Figures 5 - 7) to visualize all six possible facets combinations

[6]The raw data is provided in the supplementary material [55].

Table 5: Systematic map overview

1st Author (year)	Research Type	Contribution	Focus	Pertinence
Coleman (2008) [11]	Evaluation Research	Model	Process Management	Full
Kajko (2008) [13]	Evaluation Research	Model	Process Management	Full
Mater (2000) [68]	Evaluation Research	Model	Managerial & Organizational	Partial
Häsel (2010) [69]	Evaluation Research	Model	Managerial & Organizational	Marginal
Hanna (2010) [70]	Evaluation Research	Model	Managerial & Organizational	Marginal
Chorev (2006) [27]	Evaluation Research	Model	Managerial & Organizational	Marginal
Kakati (2003) [28]	Evaluation Research	Model	Managerial & Organizational	Marginal
Kim (2005) [71]	Evaluation Research	Model	Managerial & Organizational	Marginal
Coleman (2007) [72]	Evaluation Research	Theory	Process Management	Full
Coleman (2008) [22]	Evaluation Research	Theory	Process Management	Full
Bosch (2013) [73]	Evaluation Research	Framework & Methods	Process Management	Full
Midler (2008) [74]	Evaluation Research	Framework & Methods	Managerial & Organizational	Partial
Yogendra (2002) [75]	Evaluation Research	Guidelines	Managerial & Organizational	Partial
Yoffie (1999) [76]	Evaluation Research	Guidelines	Managerial & Organizational	Marginal
Camel (1994) [19]	Evaluation Research	Lesson Learned	Software Development	Full
Silva (2005) [67]	Evaluation Research	Lesson Learned	Software Development	Full
Jansen (2008) [77]	Evaluation Research	Lesson Learned	Software Development	Partial
Steenhuis (2008) [78]	Evaluation Research	Lesson Learned	Managerial & Organizational	Marginal
Lai (2010) [79]	Evaluation Research	Lesson Learned	Managerial & Organizational	Marginal
Tingling (2007) [80]	Evaluation Research	Advice & Implications	Software Development	Full
Li (2007) [81]	Evaluation Research	Advice & Implications	Software Development	Marginal
Blank (2013) [29]	Solution Proposal	Framework & Methods	Process Management	Partial
Zettel (2001) [82]	Solution Proposal	Framework & Methods	Software Development	Full
Crowne (2002) [8]	Solution Proposal	Advice & Implications	Software Development	Full
Mirel (2000) [83]	Solution Proposal	Advice & Implications	Managerial & Organizational	Partial
Stanfill (2007) [84]	Solution Proposal	Advice & Implications	Managerial & Organizational	Marginal
Himola (2003) [85]	Solution Proposal	Advice & Implications	Managerial & Organizational	Marginal
Heitlager (2007) [24]	Solution Proposal	Tool	Process Management	Full
Yoo (2012) [86]	Philosophical Paper	Model	Managerial & Organizational	Marginal
Yu (2012) [87]	Philosophical Paper	Guidelines	Managerial & Organizational	Marginal
Fayad (1997) [88]	Philosophical Paper	Advice & Implications	Process Management	Marginal
Bean (2005) [89]	Philosophical Paper	Advice & Implications	Tools & Technology	Marginal
Sutton (2000) [3]	Opinion Paper	Advice & Implications	Process Management	Full
Tanabian (2005) [26]	Opinion Paper	Advice & Implications	Managerial & Organizational	Marginal
Deakins (2005) [90]	Experience Paper	Model	Managerial & Organizational	Partial
Ambler (2002) [91]	Experience Paper	Lesson Learned	Software Development	Full
May (2012) [92]	Experience Paper	Advice & Implications	Software Development	Full
Taipale (2010) [93]	Experience Paper	Advice & Implications	Software Development	Full
Deias (2002) [63]	Experience Paper	Advice & Implications	Software Development	Full
Wood (2005) [94]	Experience Paper	Advice & Implications	Software Development	Partial
Wall (2001) [95]	Experience Paper	Advice & Implications	Software Development	Partial
Kuvinka (2011) [66]	Experience Paper	Advice & Implications	Software Development	Partial
Clark (2012) [96]	Experience Paper	Advice & Implications	Process Management	Marginal

from the classification schema, giving a complete overview of the systematic map and providing means to analyze it.

For example, Figure 5 indicates that 11 studies (26% of the total) are focused on managerial and organizational factors, conducted through an evaluation type research. In the same figure it is possible to observe that 8 studies with managerial and organizational focus contributed to the body of knowledge with a model. However, by looking at Figure 6, one can quickly notice that 6 out of the total 10 models have only a marginal pertinence with engineering activities in software startups.

5.2. Context characteristics of startups

To illustrate how authors use the term "software startup", we systematically extracted themes which characterize the companies in the selected primary studies. We were able to identify 15 main themes, reported in Table 6.

When discussing software startups, 18 authors reported a general lack of human, physical and economical resources (T1). For this reason, startups deeply depend upon external software solutions such as third party APIs, COTS and OSS (T7). Other studies refer to companies which are able to quickly react to changes in the market and technologies (T2), under remarkably uncertain conditions (T4). Some authors indicate that these companies are focused on highly innovative segments of the market (T3), generally working on a single core-product (T9) under extremely high time-pressure (T6). Furthermore, 14 authors write about startups as fast growing companies (T5) designed to rapidly scale-up. Other researches mention a very small founding team (T8), which is often composed by low-experienced people (T10) with a very flat organization structure (T12), where the CEO is sometimes a core developer himself. Finally, other studies agree on the highly risky nature of these businesses (T13), being newly created (T11) and therefore with no or little working history (T15).

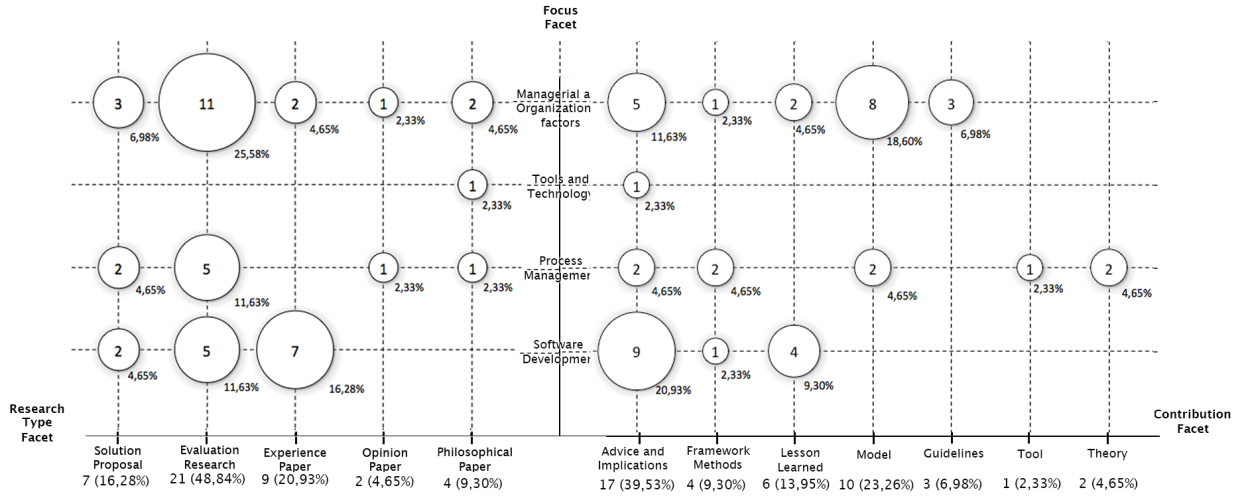

Figure 5: Systematic map - *Focus*, *contribution* and *research type*

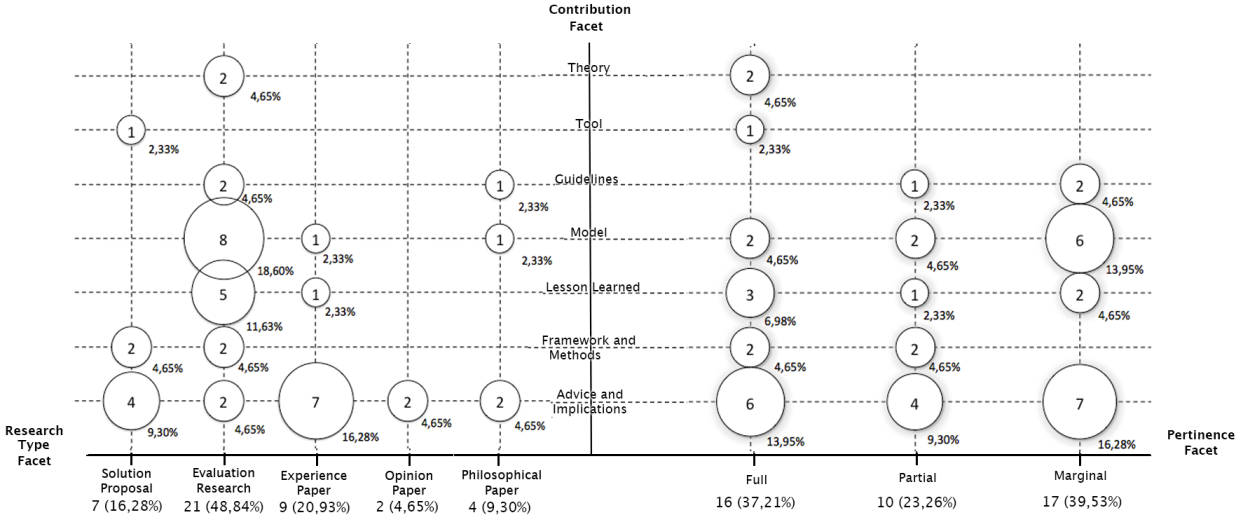

Figure 6: Systematic map - *Contribution*, *pertinence* and *research type*

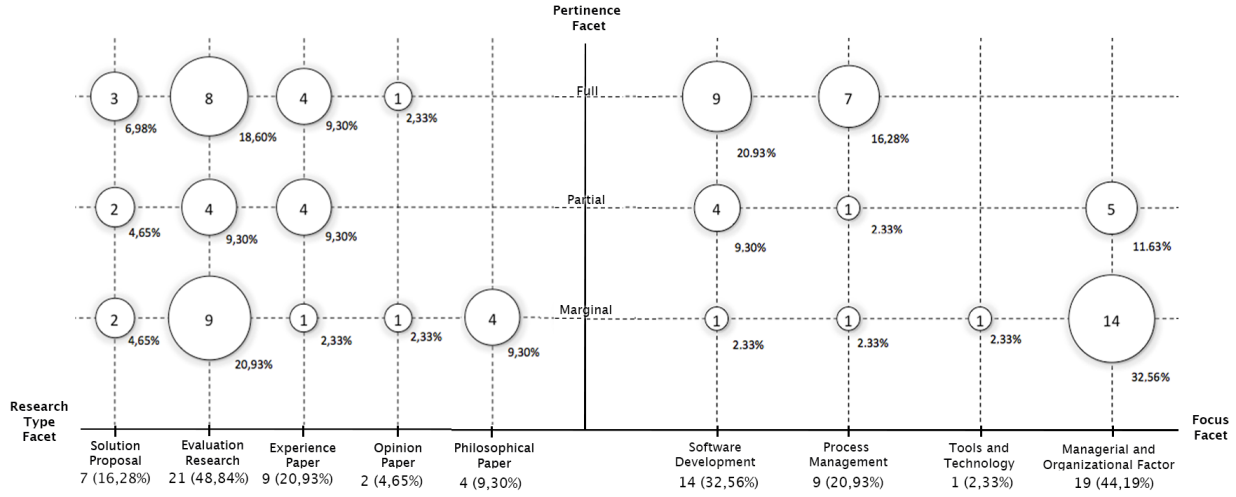

Figure 7: Systematic map - *Pertinence*, *focus* and *research type*

Table 6: Mapping Study - Recurrent themes

ID	Theme	Description	Frequency	Ref.
T1	Lack of resources	Economical, human, and physical resources are extremely limited.	18	[75, 76, 8, 13, 19, 22, 72, 11, 70, 3, 91, 84, 26, 96, 92, 73, 29, 86]
T2	Highly Reactive	Startups are able to quickly react to changes of the underlying market, technologies, and product (compared to more established companies)	17	[82, 13, 19, 22, 11, 80, 3, 88, 91, 66, 63, 67, 96, 86, 29, 73, 92]
T3	Innovation	Given the highly competitive ecosystem, startups need to focus on highly innovative segments of the market.	15	[24, 75, 83, 78, 77, 79, 3, 69, 74, 28, 92, 73, 29, 86, 87]
T4	Uncertainty	Startups deal with a highly uncertain ecosystem under different perspectives: market, product features, competition, people and finance.	14	[24, 71, 22, 72, 11, 88, 74, 26, 85, 87, 86, 29, 73, 92]
T5	Rapidly Evolving	Successful startups aim to grow and scale rapidly.	14	[75, 76, 19, 11, 81, 3, 91, 66, 90, 96, 86, 29, 73, 92]
T6	Time-pressure	The environment often forces startups to release fast and to work under constant pressure (terms sheets, demo days, investors' requests)	13	[82, 19, 11, 80, 3, 90, 85, 68, 96, 86, 29, 73, 92]
T7	Third party dependency	Due to lack of resources, to build their product, startups heavily rely on external solutions: External APIs, Open Source Software, outsourcing, COTS, etc.	10	[76, 94, 95, 77, 79, 70, 3, 91, 92, 73]
T8	Small Team	Startups start with a small numbers of individuals.	9	[82, 76, 8, 13, 3, 27, 26, 92, 96]
T9	One product	Company's activities gravitate around one product/service only.	9	[91, 66, 89, 63, 67, 11, 93, 92, 96]
T10	Low-experienced team	A good part of the development team is formed by people with less than 5 years of experience and often recently graduated students.	8	[19, 22, 72, 91, 28, 92, 73, 86]
T11	New company	The company has been recently created.	7	[8, 76, 19, 27, 96, 29, 73]
T12	Flat organization	Startups are usually founders-centric and everyone in the company has big responsibilities, with no need of high-management.	5	[76, 13, 67, 26, 92]
T13	Higly Risky	The failure rate of startups is extremely high.	5	[24, 13, 19, 26, 29]
T14	Not self-sustained	Especially in the early stage, startups need external funding to sustain their activities (Venture Capitalist, Angel Investments, Personal Funds, etc.).	3	[82, 87, 70]
T15	Little working history	The basis of an organizational culture is not present initially.	3	[76, 91, 92]

5.3. Rigor and relevance

Even though the scientific value of a study is not determined by the publication type, the peer-review process required for publishing a journal article is generally much more rigorous and formal than the procedure to get an article published in a scientific magazine or accepted to a conference [99]. Twenty (47%) of the selected 43 primary studies are journals, while 16 (37%) are published in conference proceedings and 7 (16%) in magazines. Although this feature alone is not enough to represent a direct implication on the quality[7], it can be interpreted as a first indicator of scientific quality. We formally assessed the quality of the primary studies with the rigor and relevance process described in Subsection 3.6, resulting in Figure 8 (the raw data for this figure is available in the supplementary material [55]).

Looking at Figure 8, nine studies (21%) lie in the upper right quadrant, the preferable region, of the chart (rigor $\geq$ 2, relevance $\geq$ 3). Twenty-one studies (49%) exhibit moderate industry relevance (relevance $\geq$ 2), showing however low scientific rigor (rigor $\leq$ 1.5). Ten studies (23%) are located in the lower left quadrant of the chart (rigor $\leq$ 1.5 and relevance $\leq$ 2).

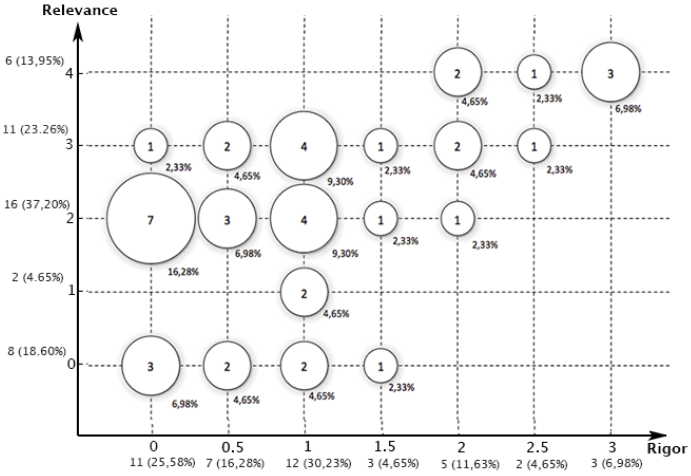

Figure 8: Rigor-relevance overview

6. Analysis of the state-of-art

More than 65% of the 43 identified primary studies have been published in the last ten years (between 2004 and 2013, see Figure 4). Fourteen primary studies, dated prior 2004, discuss software startup related issues. This indicates that the research on startups is still in its infancy, compared to the long-standing history of the SE discipline, and gaining interest in the research community. The yearly distribution of publications attest the novelty of the startup phenomenon, enabled and amplified by the potentially large markets and distribution channels offered by internet and mobile devices [20, 28]. This has opened a set of new challenges that are difficult to address using traditional approaches [3].

By analyzing the three bubble charts (Figure 5 - 7), the fol-

[7]The publication criteria are determined by the specific editor of the journal/magazine or the committee of a conference. There is a vast multitude of excellent quality studies presented in conference proceedings and magazines, and many examples of poor-quality journal articles.

lowing observations can be made on the state-of-art:

- Looking at the pertinence facet in Figure 7 we can observe that only 16 studies (37%) are entirely dedicated to software development in startups; Ten of those produced a weak contribution (advice and implications (6); lesson learned (3); tool (1)).
- Observing the focus facet (Figure 7), it is easy to see that 19 studies (44%) are focused on managerial and organizational factors. None of those 19 studies exhibits a full pertinence to engineering activities in startups.
- The overall studies' contribution types are for the greater part weak: advice and implication, lessons learned, tools and guidelines (27 studies, 63%, Figure 5). Of the 16 remaining studies (37%) which exhibit a strong contribution type (theory, framework/method, model), only 7 focus on what we considered fundamental for our research questions (software development and process management).
- Approximately half of the selected studies were carried out using an evaluation type research (21 studies, 49%, Figure 5), being the only research type which involves a field study. However, we can also observe in the same figure that 11 of these are related to managerial and organizational factors, and only 8 out of 21 have a full pertinence with engineering activities in software startups (Figure 7).
- Looking at the studies which focus on process management and software development (Figure 7), the majority (16 out of 23 studies) has a full pertinence with engineering activities in software startups.

To summarize the systematic map, we can state that Coleman and O'Connor [72, 11, 22], and Kajko-Mattson and Nikitina [13] represent the strongest contributions to the field of startup research, considering strength of contribution type, pertinence to engineering activities and strength of empirical evidence. However, it must be noted that the three publications by Coleman and O'Connor are based on the same dataset originating from 21 companies.

6.1. RQ1 - The context characterizing software development in startups

The results (Table 6) indicate that there is no agreement on a standard definition, specifying the characteristics of a "startup". Different authors provide varying definitions and use the term "startup" referring to varying contexts. This renders any attempt to identify a solid and coherent body of knowledge on startup research very challenging. Looking at Suttons startup characterization [3] from 2000, we can observe that our understanding of the nature of startups expanded to aspects such as innovation, fast growth, time pressure, third party dependency, focus on one product and flat organizational structures (see Table 6).

Defining what makes a software startup unique is an interesting problem. Apparently, the definition is not strictly related to the size of the company. For instance, some authors call "startups" companies with 6 employees [83], whilst others refer to "startups" with more than 300 employees [76, 11]. Uniqueness is not defined through the age of the company alone: some authors studied startups which have been operating for many years [79], while others are more strict and limit the definition to only recently founded companies [13]. Other authors treat "start-up" as a stage of a company [8, 26]. Others claim that startups work on innovative products, without providing an exact definition of "innovation", rendering this characterization less useful (a recent systematic study identified "41 definitions of innovation in 204 selected primary SE studies" [100]).

The most frequent reported themes concern the general lack of resources, high reactiveness and flexibility, innovation, uncertain conditions, time pressure and fast growth. Since the contextual boundaries of startups resulted to be highly blurred, it is the researchers' responsibility who refer to "startups" to explicitly define the features of the company under study (e.g. company age, team size, product type, product development time).

6.2. RQ2 - Transferability of results to industry

Figure 8 illustrates a major weakness of the state-of-art in startup research. Seven primary studies (16% of the total), received an average score for industrial relevance (2) but a low score (0) for scientific rigor. According to the authors of the rigor-relevance model [51], this represents a major threat to the transferability of the results to industry. Even though findings may appear to be somewhat appealing to practitioners (average relevance), low scientific rigor will render knowledge transfer to industry highly unlikely or even dangerous. One of the most important factors contributing to academic results being applied in the industry is the provision of strong scientific evidence [101, 102].

In the remainder of this subsection we extend the analysis of rigor and relevance by integrating factors such as publication type and year, but also the classification schema. We follow the procedure described in Subsection 3.6, computing a score (in the range $[0 - 10]$) for each primary study.

The design of the ranking function considers our research question of identifying software engineering work practices in startups. Hence, the pertinence dimension from the classification schema contributes the most (25%) to the score, followed by rigor and relevance (17.5% each). Age of the publication (15%) is factored in as more recent studies are likely to provide a more relevant context for practitioners. The publication type accounts for 10%. The contribution type, research type and focus account each for 5% of the total score which is the sum of all eight weighed scores. The conversion tables to achieve a normalized score are available in the supplementary material [55].

Table 7 quantifies the body of knowledge on startup research, provided by 43 primary studies analyzed in this systematic mapping study. The ranking gives an indication to what extent we can answer questions targeted at the state-of-art of the software engineering work practices in startups.

Table 7: Ranking of primary studies considering **P**ertinence, **Ri**gor and **Re**levance, Publication **A**ge and **T**ype, **C**ontribution type, **R**esearch type, and **F**ocus

1st Author (year)	**Score**	P	Ri	Re	A	T	C	R	F
Coleman (2007) [72]	**9.70**	2.50	1.75	1.75	1.20	1.00	0.50	0.50	0.50
Coleman (2008) [22]	**9.70**	2.50	1.75	1.75	1.20	1.00	0.50	0.50	0.50
Coleman (2008) [11]	**9.70**	2.50	1.75	1.75	1.20	1.00	0.50	0.50	0.50
Bosh (2013) [73]	**8.15**	1.50	0.88	0.88	1.00	2.50	0.40	0.50	0.50
Kajko (2008) [13]	**8.09**	2.50	0.88	1.31	1.20	0.70	0.50	0.50	0.50
Häsel (2010) [69]	**7.47**	0.75	1.17	1.75	1.50	1.00	0.50	0.50	0.30
Hanna (2010) [70]	**7.47**	0.75	1.17	1.75	1.50	1.00	0.50	0.50	0.30
Blank (2013) [29]	**7.43**	1.50	1.17	1.31	1.00	1.25	0.40	0.30	0.50
May (2012) [92]	**7.11**	1.50	0.58	0.88	0.70	2.50	0.30	0.15	0.50
Deakins(2005) [90]	**6.87**	1.25	1.46	1.31	0.90	1.00	0.50	0.15	0.30
Camel (1994) [19]	**6.61**	1.25	1.46	1.75	0.15	0.70	0.30	0.50	0.50
Silva (2005) [67]	**6.58**	2.50	0.00	0.88	0.90	1.00	0.30	0.50	0.50
Midler (2008) [74]	**6.55**	1.25	0.58	1.31	1.20	1.00	0.40	0.50	0.30
Taipale (2010) [93]	**6.53**	2.50	0.00	0.88	1.50	0.70	0.30	0.15	0.50
Chorev (2006) [27]	**6.43**	0.75	1.17	1.31	0.90	1.00	0.50	0.50	0.30
Zettel (2001) [82]	**6.32**	2.50	0.58	0.44	0.60	1.00	0.40	0.30	0.50
Jansen (2008) [77]	**6.25**	1.25	0.58	1.31	1.20	0.60	0.30	0.50	0.50
Yoo (2012) [86]	**6.24**	1.50	1.17	0.88	1.00	0.75	0.50	0.15	0.30
Sutton (2000) [3]	**6.11**	2.50	0.58	0.88	0.60	0.60	0.30	0.15	0.50
Heitlager (2007) [24]	**6.08**	2.50	0.58	0.00	1.20	0.70	0.30	0.30	0.50
Tingling (2007) [80]	**5.99**	2.50	0.29	0.00	1.20	0.70	0.30	0.50	0.50
Deias (2002) [63]	**5.92**	2.50	0.29	0.88	0.60	0.70	0.30	0.15	0.50
Stanfill (2007) [84]	**5.74**	0.75	0.88	1.31	1.20	0.70	0.30	0.30	0.30
Wood (2005) [94]	**5.70**	1.25	0.29	1.31	0.90	1.00	0.30	0.15	0.50
Clark (2012) [96]	**5.66**	1.50	0.58	0.88	1.00	0.75	0.30	0.15	0.50
Steenhuis (2008) [78]	**5.65**	0.75	0.58	1.31	1.20	0.70	0.30	0.50	0.30
Yogendra (2002) [75]	**5.55**	1.25	0.58	1.31	0.60	0.70	0.30	0.50	0.30
Ambler (2002) [91]	**5.53**	2.50	0.00	0.88	0.60	0.60	0.30	0.15	0.50
Crowne (2002) [8]	**5.48**	2.50	0.58	0.00	0.60	0.70	0.30	0.30	0.50
Mater (2000) [68]	**5.45**	1.25	0.29	1.31	0.60	0.70	0.50	0.50	0.30
Kakati (2003) [28]	**5.41**	0.75	0.58	0.88	0.90	1.00	0.50	0.50	0.30
Kuvinka (2011) [66]	**5.28**	1.25	0.00	0.88	1.50	0.70	0.30	0.15	0.50
Li (2007) [81]	**5.26**	0.75	0.00	1.31	1.20	0.70	0.30	0.50	0.50
Lai (2010) [79]	**5.23**	0.75	0.00	0.88	1.50	1.00	0.30	0.50	0.30
Mirel (2000) [83]	**4.98**	1.25	0.35	0.88	0.60	1.00	0.30	0.30	0.30
Himola (2003) [85]	**4.57**	0.75	0.58	0.44	0.90	1.00	0.30	0.30	0.30
Kim (2005) [71]	**4.53**	0.75	0.88	0.00	0.90	0.70	0.50	0.50	0.30
Wall (2001) [95]	**4.28**	1.25	0.00	0.88	0.60	0.60	0.30	0.15	0.50
Yoffie (1999) [76]	**4.22**	0.75	0.29	0.88	0.60	0.60	0.30	0.50	0.30
Yu (2012) [87]	**3.99**	1.50	0.29	0.00	0.70	0.75	0.30	0.15	0.30
Bean (2005) [89]	**3.50**	0.75	0.00	0.00	0.90	1.00	0.30	0.15	0.40
Tanabian (2005) [26]	**3.10**	0.75	0.00	0.00	0.90	0.70	0.30	0.15	0.30
Fayad (1997) [88]	**2.45**	0.75	0.00	0.00	0.15	0.60	0.30	0.15	0.50

7. RQ3 - Work Practices in startups

We have extracted a total of 213 work practices[8] from the 43 primary studies reviewed in this SMS and subsequently divided them in categories (Table 8), as explained in Subsection 3.7. The categorization of working practices is defined according to the focus facet of the classification schema, presented in Figure 4c. In the remainder of this section, we discuss the identified work practices, pointing out where gaps exist and further research is warranted.

[8]Note that this number does not reflect unique work practices but the total number; a detailed table of work practices is available in the supplementary material [55].

Table 8: Categorization of the identified work practices

Software Development (Subsection 7.2)	90
Managerial/organizational (Subsection 7.3)	70
Process management (Subsection 7.1)	47
Tools and technologies (Subsection 7.4)	6
Sum	213

7.1. Process management practices

Process management represents all the engineering activities used to manage product development in startups. Sutton [3] recognized the need for flexibility to accommodate frequent changes in the development environment, and for reactiveness to obtain timely response in applying methodologies.

Agile methodologies have been considered the most viable processes for software startups, given that Agile methodologies embrace changes rather than avoiding them, allowing development to follow the business strategy [93]. In this context, fast releases with an iterative and incremental approach shorten the lead time from idea conception to production with fast deployment [93]. The benefits of having weekly releases and frequent build cycles, addressing the uncertainty of the market, has been further reported by Blank [29], Tingling [80], Ambler [91] and Silva [67].

A variant to Agile has been the Lean Startup [103, 29], which advocates the identification of the most risky parts of a software business and provide a minimum viable product (MVP) to systematically test and plan modification for a next iteration. In this regard, in order to shorten time-to-market, prototyping is essential [90, 19]. To allow better prototyping activities, evolutionary workflows are needed to implement "soft-coded" solutions in the first phases until the optimal solution is found [90, 3].

Coleman [72] reports that XP is the most used development methodology across startup companies, due to its reduced process costs and low documentation requirements. Also other agile practices are explored, such as Scrumban [66], but not rigorously researched. In any case, flexible in nature, startups' processes don't strictly follow any specific methodology, but opportunistically select practices (e.g. pair-programming [63], backlog [80]). In fact, processes are tailored to the specific features that characterize each development context [92, 72, 11]. For example, Bosch et al. [73] advocate for adjusting the Lean startup methodology to accommodate the development of multiple ideas and integrate them when time for their testing and validation is too long. This concurs with the practice of allocating varying effort for formalizing specifications, design, documentation and testing in tailored development methodologies [91, 82, 19], emphasizing the importance of minimal process management.

Summarizing, process management practices, reported to be useful in startups, are:

- Light-weight methodologies to obtain flexibility in choosing tailored practices, and reactiveness to change the product according to business strategies.

- Fast releases to build a prototype in an evolutionary fashion and quickly learn from the users' feedback to address the uncertainty of the market.

Discussion

The Cynefin framework [104] can be used to explain the orientation of startups towards flexible and reactive development approaches. Within this framework, startups cross the complex and chaotic domains. Those two domains represent the areas where applying rigorous process management to control development activities is not effective, because no matter how much time is spent in analysis, it is not possible to identify all the risks or accurately predict what practices are required to develop a product. Instead, flexible and reactive methods, designed to stimulate customer feedback, increase the number of perspectives and solutions available to decision makers. Moving from complex to chaotic domains, software startups open up new possibilities for creation, generating the condition for innovations. Therefore, any process tailored to the startup context needs at least to allow, but optimally even facilitate movements between complex and chaotic domains that are intrinsic in the innovation generation of startups. In our opinion, this is the main requirement for future attempts of adapting software engineering processes to the startup context.

Developers should have the freedom to choose activities quickly, stop immediately when results are wrong, fix the approach and learn from previous failures. In this regard, in line with the Lean Startup movement, we expect methodologies and techniques tailored from common Agile practices to specific startups' culture and needs, where failing is completely acceptable, even preferred in favor of a faster learning process. However at some point, in preparation for growth, startups need to plan for scalable processes. Similarly to SMEs [49], they need to find a balance between flexibility and repeatability in their organizations' knowledge management and processes.

7.2. Software development practices

We have categorized work practices related to software development as illustrated in Table 9, discussing them individually.

Table 9: Software development practices

Requirements Engineering (Subsection 7.2.1)	21
Design and Architecture (Subsection 7.2.2)	32
Implementation, Maintenance and Deployment (Subsection 7.2.3)	14
Quality Assurance (Subsection 7.2.4)	23
Sum	90

7.2.1. Requirements Engineering practices

Establishing an engineering process for collecting, defining and managing requirements in the startup context is challenging. RE practices are often reduced to some key basic activities [82, 8]. Su-Chan et al. [81] report on efforts in defining the value-proposition that the company aims to achieve at the very first stage of the project.

Initially, as startups often produce software for a growing target market [29, 19], customers and final users are usually not well-known and requirements are therefore market-driven [90] rather than customer-specific. In such a context Mater and Subramanian [68] attest severe difficulties in eliciting and detailing the specifications of both functional and non-functional requirements. Moreover, in unexplored and innovative markets, the already poorly-defined requirements tend to change very rapidly. This makes it hard for the development team to maintain requirements and keep them consistent over time.

Several authors acknowledge the importance of involving the customer/user in the process of eliciting and prioritizing requirements according to their primary needs [92, 90, 74, 8, 95]. However, the market-driven nature of the requirements demands for alternatives. For example, startups can use scenarios in order to be able to identify requirements in the form of user stories [67] and estimate the effort for each story [82]. In scenarios and in similar product-usage simulations, an imaginary customer can be represented by an internal member of the company [80]. An example of a more strict customer-development process [105] that drives the identification of requirements can be found in an experience report by Taipale [93].

Discussion

Polishing requirements that address an unsolicited need is waste. To demonstrate problem/solution fit it is required to discover the real needs of your first customers, testing business speculations only by defining a minimal set of functional requirements. In the future, developing a deep customer collaboration, such as the customer development process [37] will change the requirements elicitation methods, moving towards testing the problem and understand if the solution fits to real needs even before the product goes to the market.

7.2.2. Design and Architecture practices

Deias and Mugheddu [63] observed a general lack of written architecture and design specifications in startups. Development in startups is usually guided by simple principles, trying to avoid architectural constraints that are difficult to overcome as the product and user-base grows. Tinglings [80] results suggest that a not well analyzed architecture causes problems in the long run. However, a good-enough architecture analysis should at least identify the decisions that might cause problems before obtaining product revenue, and which can be fixed at a later point in time, accounting for increased resources after revenue cash starts to flow [92].

One common analysis on determining requirements is the planning game, where developers can arbitrate the cost of desired features and delivered functionalities. However business people can "steer" the process and impact adopted architectural decisions, which could present obstacles for refactoring activities, especially if the software complexity grows [91]. Then the use of design patterns [106], e.g. the model-view-controller [107], can provide advantages to startups which need flexibility in refactoring the product. Moreover, formulating initial architectural strategies with high-level models and code-

reuse from industry standards represents a viable trade-off between big up-front and ad-hoc design [73].

Jansen et al. [77] suggest that startups should take advantage of existing components, reducing thereby time-to-market. Leveraging on frameworks and code-reuse of third party components reinforces the architectural structure of the product and the ability to scale with the company size. As reported by Yoffie [76], scalability represents the most important architectural aspect in startups and should be addressed as soon as possible. Then, startups can benefit from reusing components and shared architectures across projects as well as existing high-level frameworks and modeling practices.

Summarizing, design and architectural practices reported to be useful in startups are:

- The use of well-known frameworks able to provide fast changeability of the product in its refactoring activities.
- The use of existing components, leveraging third party code reinforcing the product ability to scale.

Discussion

Despite the general lack of written architecture specifications in startups, the difficulties presented when the user-base and product complexity grows can be overtaken by a little up-front effort on framework selection and analyzing decisions that might cause problems before obtaining product revenue. When the product evolves, the use of architecture and design to make features modular and independent are crucial to remove or change functionalities. Therefore, employing architectural practices and frameworks that enable easy extension of the design (e.g. pluggable architecture where features can be added and removed as plugins [108]) can better align the product to the uncertainty of the market needs.

7.2.3. Implementation, Maintenance and Deployment practices

Silva and Kon [67] report on positive results from pairing up senior and junior developers together in pair-programming sessions. During these sessions, developers also made use of coding standards to help cross-team code readability and reduce the risks related to the high-rate of developer turnover. In a different study, Tingling [80] attested an initial high resistance to the introduction of coding standards and pair-programming. These practices were then adopted only in later stages, when the complexity of the project required them. Zettel et al. [82] report that the practice of tracking traditional code metrics has been labeled as "obsolete and irrelevant" and that many companies use their ad-hoc methods of assessing processes and metrics. The software team studied by Zettel et al. [82] had a bug-fix process centered around their issue-tracking tool and relied on a release system. Several authors reported on advantages of constant code refactoring: ensuring readability and understandability of the code [82], improving modularity [80] and providing discipline to developers [93]. On the other hand, introducing refactoring may cause problems since developers had no or little experience [63] and they didn't see immediate value in introducing high level abstractions [67]. In the case study described by Ambler [91], an initial lack of refactoring led to the need of re-implementing the whole system after the number of users had grown drastically. Finally some authors reported work practices related to deployment claiming that some software teams deploy manually the code on the infrastructure [67] while others rely on continuous integration tools [93].

Discussion

Startups tend to start the code implementation with an informal style, introducing standards only when the project size requires them. This is in line with the observations made by Thorpe et al. [49] on knowledge management and growth in SMEs. The process is often driven by lightweight ad-hoc systems: bug-tracking, simple code metrics and pair-programming sessions. In this regard, startups in the early stage keep the code base small and simple to develop only what is currently needed, maintaining focus on core functionalities to validate with first customers. As the business goal drives the need of effort in refactoring and implementation, more studies will be needed to align business with execution of concrete development practices in startup contexts (e.g. GQM Strategies [109]).

7.2.4. Quality Assurance practices

Testing software is costly in terms of calendar time and often compromised in startups [19, 82]. Quality assurance, in the broader sense, is largely absent because of the weak methodological management, discussed in Subsection 7.1. The complex task of implementing test practices is further hindered by the lack of team experience [67, 68].

However, usability tests are important to achieve product/market fit [83]. Ongoing customer acceptance ensures that features are provided with tests, which can effectively attest the fitness of the product to the market [80, 92, 29]. Mater and Subramanian [68] suggest to use a small group of early adopters or their proxies as quality assurance fit team. Furthermore, users can be an important means to judge whether the system needs more tests [82]. Outsourcing quality assurance to external software testing experts, handling the independent validation work if resources are not available [68], can also be an alternative.

Summarizing, quality assurance practices, reported to be useful in startups, are:

- The use of ongoing customer acceptance with the use of a focus groups of early adopters, which targets to attest the fitness of the product to the market.
- Outsourcing tests if necessary, to maintain the focus on the development of core functionalities.

Discussion

Even though testing software is costly, acceptance tests are the only practice to validate the effectiveness of the product in uncertain market conditions [29]. Therefore, providing time-efficient means to develop, execute and maintain acceptance tests is the key to improve quality assurance in startups [68]. In our opinion, startups will start making use of different automatic testing strategies, when easily accessible (e.g. create

a test from each fixed bug [110]). Considering startups' frequent changing activities during the validation of the product on the market, UI testing remains not a simple but important task. However, more research is needed to develop practical UI testing approaches that can be commercialized [111].

7.3. Managerial and organizational practices

Flexibility, more than structure, plays an important role in startup companies [85, 96]. Time pressure and lack of resources [86, 87] lead to a loose organizational structure and often lack of traditional project management [19]. To accommodate flexibility of managerial and organizational practices, the empowerment of team members represents the main viable strategy to enhance performance and chances of success [19, 78]. The team needs to be able to absorb and learn from trial and error quickly enough to adapt to new emergent practices [74, 3, 78].

Empowerment allows the team to move rapidly and cutting through the bureaucracy, approval committees and veto cultures [92, 73]. Nevertheless, key performance indicators (e.g. customer attrition, cycle time) and continuous deployment processes can effectively assess the consumers' demand using the least amount of resources possible [103]. However, in building up a startup company, the team needs expertise to counterbalance the lack of resources [27, 22, 92]. Working on innovative products requires creativity, ability to adapt to new roles and to face new challenges everyday [3, 69, 86], working overtime if necessary [19, 26]. Previous experience in similar business domains [76, 87, 86] and a working history in a team, exhibiting characteristics of an entrepreneur (courage, enthusiasm, commitment, leadership), also play a primary role [27, 28, 96] in the skill set of a startup employee.

Nevertheless, the absence of structure might hinder important activities, such as sharing knowledge and team coordination, especially when the company grows [91], as also observed in the context of SMEs [49]. In this case co-location is essential to facilitate informal communication and close interactions between development and business concerns [72, 80]. Effective organization and governance mechanisms need to enable and maintain alignment between business and technology strategies, avoiding waste of resources [75, 82]. Moreover, Crowne suggests to plan organizational objectives in the short-medium term [8], measuring development cycle time to find areas for improvements [93, 13].

Finally, despite Camel [19] reports lack of documentation in startup companies, Kuvinka [66] argues that startups, when approaching the development of long user stories, can take advantage of documentation and sometimes UI design to facilitate their management in the long run, especially when interacting with third parties [93].

Summarizing, managerial and organizational practices reported to be useful in startups are:

- Empowerment of team responsibilities and their ability to influence the final outcomes.
- The use of key performance indicators to assess the consumers' demand.
- Plan of short-medium term objectives, measuring development cycle time to find areas of improvement.

Discussion

More empowerment allows the team to move rapidly with less bureaucracy. Then, the development team plays a key factor to enhance commitment, creativity and ability to adapt to new roles when necessary. In addition, open communication remains crucial for startups to handle engineering activities, understanding the progress, code conflicts and competences. Therefore, new tools and techniques for focusing, exploring and making observations, encouraging comparison and seeking clarification and validation could improve effective verbal communication and lead to less misunderstandings and confusion. Nevertheless, in view of growth, managing transferable knowledge becomes crucial when hiring new personnel. However, keeping it informal but still providing enough structure for knowledge creation is challenging. In this regard, Thorpe et al. [49] suggest to design a flexible "learning architecture", sensitive to entrepreneurial characteristics and specific context of the company, without limiting creativity.

7.4. Tools and technologies

Startups are often established to develop technologically innovative products, which in turn might require cutting-edge development tools and techniques. Technological changes in the IT industry swipe through new network technologies, increasing variety of computing devices, new programming languages, objects and distribution technologies [3].

However, from a managerial perspective, startups still prefer easy-to-implement tools, such as white-boards and real-time tools that are easy to use for handling fast-paced changing information. Sophisticated solutions, such as CASE tools [112], require training and have high implementation and maintenance costs [89, 91].

In general, startups take advantage of those technologies that can quickly change the product and its management [67, 8], avoiding conflicts with business strategic plans. Examples include general-purpose infrastructures, such as configuration management, problem reporting and tracking systems, planning, scheduling and notification systems. Such technologies support the needed activities, accommodating changes when required [3]. To mitigate the lack of resources, startups might take advantage of open source solutions when possible, which also gives access to a large pool of evaluators and evolving contributions [95, 94].

Summarizing, tools and technologies practices reported to be useful in startups are:

- The use of easy-to-implement tools to work with fast-paced changing information.
- The use of open source solutions.

Discussion

Startups can take advantage of using new technologies and development tools without having any legacy or being constrained by previous working experience. However, lack of

experience can also be a disadvantage which could be compensated by a light-weight process to select technologies; this selection could be guided by domain-specific or product specific requirements. For example, if a startup is creating a product that is meant to work with a growing amount and diversity of consumer mobile devices, to create business advantages, the platform should be easy to extend to support new devices [113].

8. Conclusions and future work

Startups are able to produce software products with a strong impact on the market, significantly contributing to the global economy. Software development, being at the core of startups' daily activities, is however not supported by a scientific body of knowledge. The evidence provided by the 43 primary studies is, for the most part, inadequate to understand the underlying phenomenon of software development in startups. To the current date, fourteen years after Sutton assessed that startups have been neglected from process studies [3], the gap has been only partially filled.

This is remarkable, considering startups' proliferation and, at the same time, high failure rate. Hence, to be able to intervene on the software development strategy with scientific and engineering approaches, it is necessary to better understand and characterize the state-of-art in the software startup context.

By means of a systematic mapping study, we provide a classification of the state-of-art, assess rigor and relevance of the identified primary studies, and analyze software development work practices discussed in the surveyed literature.

Looking at the 43 primary studies, 19 (44%) are focused on managerial and organizational factors. Only 16 studies (37%) are entirely dedicated to software development in startups, whereby 10 studies constitute a weak contribution type. Overall, only 4 contributions to the field are entirely dedicated to engineering activities in startups, providing a strong contribution type and conducted through an evidence-based research approach [11, 22, 72, 13]. However, three of these studies are based on the same data, leading to the conclusion that there is a lack of relevant primary studies on software development in the startup context. In the following subsections we provide answers to our initially posed research questions.

8.1. RQ1 What is the context that characterizes software development in startups?

There is no unique definition in literature on what constitutes a startup. The inconsistent use of the term "startup" by different authors and lacking descriptions of context restrains the creation of a coherent body of knowledge on software startups. This also hinders the adoption of results by practitioners as the context in which advancements are applicable is lacking.

The most frequently reported contextual features of startups concern the general lack of resources, high reactiveness and flexibility, intense time-pressure, uncertain conditions and fast growth. Since the contextual boundaries of startups resulted to be highly blurred, we believe it is responsibility of the researchers who refer to the term "startup" to explicitly define the features of the studied companies. In most of primary studies an explicit contextualization has been neglected, affecting the generalizability of their results.

Some of the features characterizing startups are common to other SE contexts: innovative products development, market-driven development, small companies, short time-to-market, web-development. However, the coexistence of all these features poses a new, unique series of challenges.

8.2. RQ2 To what extent does the research on startups provide reliable and transferable results to industry?

The rigor and relevance analysis indicates that only a minority of the studies (9, 21%) representing the state-of-art provide transferable and reliable results to practitioners. Even more concerning is that more than half of the studies (23, 53%) exhibit moderate industry relevance, however at the same time low scientific rigor. This makes the transfer of results to practitioners highly unlikely or even dangerous [51], calling for more rigorous studies in the context of software startups.

8.3. RQ3 What are the reported work practices in association with software engineering in startups?

We extracted and discussed work practices commonly used in startups as reported in the reviewed literature. In terms of process management, agile and more traditional methodologies struggle to get adopted by startups due to an excessive amount of uncertainty and high time-pressure. Light-weight methodologies that allow companies to pick and tailor practices are preferred as they facilitate reactiveness and allow rapid changes in the product. In this sense, processes in startups are evolutionary in nature, and the product is obtained by iterating and updating an early prototype driven by customer feedback.

Software development practices are reported to be adopted only partially and mostly in a late stage of the startup life-cycle. Requirements are market-driven and hardly documented. The architecture and design is often replaced by the use of well-known frameworks that facilitate maintenance and reduce documentation efforts. Ad-hoc code metrics, pair programming sessions, and code refactoring sessions are reported to be valuable practices. Testing is mostly conducted through customer acceptance, focus groups, and sometimes by outsourcing the testing activity.

Managerial and organizational practices are reduced to the essential. With minimal bureaucracy, developers are largely empowered and encouraged to adapt to several roles, creatively impacting on the final products. Given the unpredictability in the startup context, milestones and objectives are focused on the short-medium term and basic key performance indicators are used to track customers' demands.

Startups mainly make use of simple tools to support and trace the knowledge-base and manage the workflow, often opting for open-source solutions that require little or no training and maintenance.

8.4. Implications for practitioners, research and future work

Evidence from the reviewed primary studies indicates that startup companies, struggling to survive and operate in an unpredictable context, can benefit from adopting certain software engineering practices. However, low rigor studies and insufficient context information threaten the adoption of these practices by practitioners.

Performing research on startups is challenging due to the rapidly-changing conditions surrounding the studied environment. Therefore it is crucial to explicitly define the context when studying startups, describing however also study design and validity threats. This strengthens studies, lifting the potential of results transfer to industry.

While the characterization of startups through recurrent themes presented in this paper can serve as a basis, future work is needed to compile a common startup terminology and a set of definitions. That would support the generation of a consistent body of knowledge based on evidence-based research, aiming at supporting activities and decisions of the growing number of startup companies. We are currently investigating early-stage startups that are recently founded and distributed in different geographic areas and market sectors. Following a grounded theory approach, we aim at exploring the state-of-practice in this context, identifying software development strategies engineered by practitioners.

References

[1] B. Kitchenham, P. Brereton, D. Budgen, M. Turner, J. Bailey, S. Linkman, Systematic literature reviews in software engineering – A systematic literature review, Information and Software Technology 51 (1) (2009) 7–15.

[2] F. Q. B. da Silva, A. L. M. Santos, S. Soares, A. C. C. França, C. V. F. Monteiro, F. F. Maciel, Six years of systematic literature reviews in software engineering: An updated tertiary study, Information and Software Technology 53 (9) (2011) 899–913.

[3] S. M. Sutton, The role of process in software start-up, IEEE Software 17 (4) (2000) 33–39.

[4] B. Kitchenham, S. Charters, Guidelines for performing Systematic Literature Reviews in Software Engineering, Tech. Rep. EBSE 2007-001, Keele University and Durham University Joint Report (2007).

[5] D. Budgen, M. Turner, P. Brereton, B. Kitchenham, Using Mapping Studies in Software Engineering, in: Proceedings of the 20th Annual Meeting of the Psychology of Programming Interest Group (PPIG), 2008, pp. 195–204.

[6] H. Zhang, M. A. Babar, Systematic reviews in software engineering: An empirical investigation, Information and Software Technology 55 (7) (2013) 1341–1354.

[7] T. Kane, The Importance of Startups in Job Creation and Job Destruction, Tech. rep., Kauffman Foundation (July 2010).

[8] M. Crowne, Why software product startups fail and what to do about it, in: Proceedings of the International Engineering Management Conference (IEMC), 2002, pp. 338–343.

[9] A. Maccormack, How Internet Companies Build Software, MIT Sloan Management Review 42 (2) (2001) 75–84.

[10] K. M. Eisenhardt, S. L. Brown, Time pacing: competing in markets that won't stand still., Harvard Business Review 76 (2) (1998) 59–69.

[11] G. Coleman, R. O'Connor, An investigation into software development process formation in software start-ups, Journal of Enterprise Information Management 21 (6) (2008) 633–648.

[12] S. Blank, The four steps to the epiphany, Cafepress, 2005.

[13] M. Kajko-Mattsson, N. Nikitina, From Knowing Nothing to Knowing a Little: Experiences Gained from Process Improvement in a Start-Up Company, in: Proceedings of the International Conference on Computer Science and Software Engineering (CSSE), 2008, pp. 617–621.

[14] G. Coleman, An empirical study of software process in practice, in: Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS), 2005, p. 315c.

[15] K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, Systematic Mapping Studies in Software Engineering, in: Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE), 2007, pp. 1–10.

[16] D. Storey, Entrepreneurship and the New Firm, Croom Helm, 1982.

[17] A. B. Perkins, M. C. Perkins, The Internet Bubble: Inside the Overvalued World of High-Tech Stocks – And What You Need to Know to Avoid the Coming Catastrophe, HarperInformation, 1999.

[18] M. Marmer, B. L. Herrmann, E. Dogrultan, R. Berman, C. Eesley, S. Blank, Startup Genome Report Extra: Premature Scaling, Tech. rep., Startup Genome (2011).

[19] E. Carmel, Time-to-completion in software package startups, Proceedings of the 27th Hawaii International Conference on System Sciences (HICSS) (1994) 498–507.

[20] C. M. Christensen, The Innovator's Dilemma, Harvard Business School Press, 1997.

[21] A. Fuggetta, Software process: A roadmap, in: Proceedings of the Conference on The Future of Software Engineering (ICSE), ACM, 2000, pp. 25–34.

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

[23] J. Bach, Microdynamics of process evolution, Computer 31 (1998) 111–113.

[24] I. Heitlager, R. Helms, S. Brinkkemper, A tentative technique for the study and planning of co-evolution in product software startups, in: Proceedings of the 3rd International Workshop on Software Evolvability, 2007, pp. 42–47.

[25] K. Martin, B. Hoffman, An open source approach to developing software in a small organization, IEEE Software 24 (1) (2007) 46–53.

[26] M. Tanabian, Building high-performance team through effective job design for an early stage software start-up, in: Proceedings of the International Engineering Management Conference (IEMC), 2005, pp. 789–792.

[27] S. Chorev, A. R. Anderson, Success in Israeli high-tech start-ups; Critical factors and process, Technovation 26 (2) (2006) 162–174.

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

[29] S. Blank, Why the Lean Start-Up Changes Everything, Harvard Business Review 91 (5) (2013) 64+.

[30] C. Alves, S. Pereira, J. Castro, A Study in Market-Driven Requirements Engineering, Tech. rep., Universidade Federal de Pernambuco (2006).

[31] J. Natt Och Dag, Elicitation and management of user requirements in market-driven software development, Ph.D. thesis, Department of Communication Systems Lund Institute of Technology (2002).

[32] P. Sawyer, I. Sommerville, G. Kotonya, Improving market-driven re processes, in: Proceedings of the International Conference on Product-Focused Software Process Improvement (PROFES), 1999.

[33] C. Potts, Invented requirements and imagined customers: requirements engineering for off-the-shelf software, in: Proceedings of the 2nd International Symposium on Requirements Engineering (RE), 1995, pp. 128–130.

[34] L. Karlsson, Å. G. Dahlstedt, J. Natt Och Dag, B. Regnell, A. Persson, Challenges in market-driven requirements engineering - an industrial interview study, in: Proceedings of the 8th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ), 2002.

[35] A. Dahlstedt, Study of current practices in marketdriven requirements engineering, in: Proceedings of the 3rd Conference for the Promotion of Research in IT, 2003.

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

[37] S. Blank, Embrace failure to start up success., Nature 477 (7363) (2011) 133.

[38] I. Heitlager, S. Jansen, R. Helms, S. Brinkkemper, Understanding the dynamics of product software development using the concept of coevo-

lution, in: Proceedings of the 2nd International Workshop on Software Evolvability, IEEE Computer Society, Washington, DC, USA, 2006, pp. 16–22.

[39] B. A. Kitchenham, T. Dybå, M. Jørgensen, Evidence-based software engineering, in: Proceedings of the 26th International Conference on Software Engineering (ICSE), IEEE Computer Society, Los Alamitos, CA, USA, 2004, pp. 273–281.

[40] B. Dit, M. Revelle, M. Gethers, D. Poshyvanyk, Feature location in source code: a taxonomy and survey, Journal of Software Maintenance and Evolution: Research and Practice 25 (1) (2011) 53–95.

[41] W. Afzal, R. Torkar, R. Feldt, A systematic review of search-based testing for non-functional system properties, Information and Software Technology 51 (6) (2009) 957–976.

[42] Enterprise - SME definition (Feb. 2014).
URL http://www.ec.europa.eu/enterprise/enterprise_policy/sme_definition/sme_user_guide.pdf

[43] E. Kamsties, K. Hörmann, M. Schlich, Requirements engineering in small and medium enterprises, Requirements Engineering 3 (2) (1998) 84–90.

[44] C. Laporte, S. Alexandre, R. O'Connor, A software engineering lifecycle standard for very small enterprises, in: R. O'Connor, N. Baddoo, K. Smolander, R. Messnarz (Eds.), Software Process Improvement, Vol. 16 of Communications in Computer and Information Science, Springer Berlin Heidelberg, 2008, pp. 129–141.

[45] F. J. Pino, F. García, M. Piattini, Software process improvement in small and medium software enterprises: a systematic review, Software Quality Journal 16 (2) (2008) 237–261.

[46] I. Richardson, C. G. von Wangenheim, Guest editors' introduction: Why are small software organizations different?, IEEE Softw. 24 (1) (2007) 18–22.

[47] M. Staples, M. Niazi, R. Jeffery, A. Abrahams, P. Byatt, R. Murphy, An exploratory study of why organizations do not adopt {CMMI}, Journal of Systems and Software 80 (6) (2007) 883 – 895.

[48] P. Taticchi, F. Tonelli, L. Cagnazzo, Performance measurement and management: a literature review and a research agenda, Measuring Business Excellence 14 (1) (2010) 4–18.

[49] R. Thorpe, R. Holt, A. Macpherson, L. Pittaway, Using knowledge within small and medium-sized firms: A systematic review of the evidence, International Journal of Management Reviews 7 (4) (2005) 257–281.

[50] N. Rosenbusch, J. Brinckmann, A. Bausch, Is innovation always beneficial? a meta-analysis of the relationship between innovation and performance in {SMEs}, Journal of Business Venturing 26 (4) (2011) 441 – 457.

[51] M. Ivarsson, T. Gorschek, A method for evaluating rigor and industrial relevance of technology evaluations, Empirical Software Engineering 16 (3) (2010) 365–395.

[52] A. Strauss, J. Corbin, Basics of Qualitative Research, 2nd Edition, SAGE Publications, 1998.

[53] S. Rumsey, How to find information: a guide for researchers, McGraw-Hill, 2008.

[54] S. Barney, K. Petersen, M. Svahnberg, A. Aurum, H. Barney, Software quality trade-offs: A systematic map, Information and Software Technology 54 (7) (2012) 651–662.

[55] N. Paternoster, C. Giardino, M. Unterkalmsteiner, T. Gorschek, Supplementary material to "Software development in startup companies: A systematic mapping study" (2013).
URL http://www.bth.se/com/mun.nsf/pages/startup-sysmap

[56] BibDesk. [Online]. Available: http://bibdesk.sourceforge.net/ (Accessed : Nov. 25, 2013).

[57] T. Dybå, T. Dingsøyr, Empirical studies of agile software development: A systematic review, Information and Software Technology 50 (9-10) (2008) 833–859.

[58] M. Jorgensen, M. Shepperd, A systematic review of software development cost estimation studies, Transactions on Software Engineering 33 (1) (2007) 33–53.

[59] A. Sayers, Tips and tricks in performing a systematic review, The British Journal of General Practice 1 (57) (2007) 542–759.

[60] M. Ivarsson, T. Gorschek, Technology transfer decision support in requirements engineering research: a systematic review of rej, Requirements Engineering 14 (3) (2009) 155–175.

[61] M. Unterkalmsteiner, T. Gorschek, A. Islam, C. K. Cheng, R. Permadi, R. Feldt, Evaluation and measurement of software process improvement – a systematic literature review, Transactions on Software Engineering 38 (2) (2012) 398–424.

[62] T. Saracevic, Evaluation of evaluation in information retrieval, in: Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1995, pp. 138–146.

[63] R. Deias, G. Mugheddu, O. Murru, Introducing xp in a start-up, in: Proceedings 3rd International Conference on eXtreme Programming and Agile Processes in Software Engineering (XP), 2002, pp. 62–65.

[64] R. Wieringa, N. Maiden, N. Mead, C. Rolland, Requirements engineering paper classification and evaluation criteria: a proposal and a discussion, Requirements Engineering 11 (1) (2005) 102–107.

[65] M. Shaw, Writing good software engineering research papers, in: Proceedings of the 25th International Conference on Software Engineering (ICSE), 2003, pp. 726–736.

[66] K. Kuvinka, Scrum and the Single Writer, in: Proceedings of Technical Communication Summit, 2011, pp. 18–19.

[67] A. da Silva, F. Kon, Xp south of the equator: An experience implementing XP in Brazil, Extreme Programming and Agile Processes (2005) 10–18.

[68] J. Mater, B. Subramanian, Solving the software quality management problem in Internet startups, in: Proceedings of the 18th Annual Pacific Northwest Software Quality Conference, 2000, pp. 297–306.

[69] M. Häsel, N. Breugst, T. Kollmann, IT Competence in Internet Founder Teams An Analysis of Preferences and Product Innovativity, Business & Information System Engineering 52 (4) (2010) 210–217.

[70] R. Hanna, T. U. Daim, Information technology acquisition in the service sector, International Journal of Services Sciences 3 (1) (2010) 21–39.

[71] E. Kim, S. Tadisina, Factors impacting customers' initial trust in e-businesses: an empirical study, in: Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS), 2005, pp. 1–10.

[72] G. Coleman, R. O'Connor, Using grounded theory to understand software process improvement: A study of Irish software product companies, Information and Software Technology 49 (6) (2007) 654–667.

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

[74] C. Midler, P. Silberzahn, Managing robust development process for high-tech startups through multi-project learning: The case of two European start-ups, International Journal of Project Management 26 (5) (2008) 479–486.

[75] S. Yogendra, Aligning business and technology strategies: a comparison of established and start-up business contexts, in: Proceedings of the Internal Engineering Management Conference (IEMC), 2002, pp. 2–7.

[76] D. Yoffie, Building a company on Internet time: Lessons from Netscape, California Management Review 4 (3).

[77] S. Jansen, S. Brinkkemper, I. Hunink, Pragmatic and Opportunistic Reuse in Innovative Start-up Companies, IEEE Software (2008) 42–49.

[78] H.-J. Steenhuis, E. de Bruijn, Innovation and technology based economic development: Are there short-cuts?, in: Proceedings of the International Conference on Management of Innovation and Technology (ICMIT), 2008, pp. 837–841.

[79] S.-l. Lai, Chinese Entrepreneurship in the Internet Age: Lessons from Alibaba.com, World Academy of Science, Engineering and Technology 72 (2010) 405–411.

[80] P. Tingling, Extreme programming in action: a longitudinal case study, in: Proceedings of the 12th International Conference on Human-computer Interaction: Interaction Design and Usability (HCI), 2007, pp. 242–251.

[81] S.-C. Li, The role of value proposition and value co-production in new internet startups: How new venture e-businesses achieve competitive advantage, in: Portland International Center for Management of Engineering and Technology (PICMET), 2007, pp. 1126 –1132.

[82] J. Zettel, F. Maurer, J. Münch, L. Wong, LIPE: a lightweight process for e-business startup companies based on extreme programming, in: Proceedings 3rd International Conference on Product Focused Software Process Improvement (PROFES), Springer, 2001, pp. 255–270.

[83] B. Mirel, Product, process, and profit: the politics of usability in a software venture, ACM Journal of Computer Documentation (JCD) 24 (4) (2000) 185–203.
[84] R. Stanfill, T. Astleford, Improving Entrepreneurship Team Performance through Market Feasibility Analysis, Early Identification of Technical Requirements, and Intellectual Property Support, in: Proceedings of the American Society for Engineering Education Annual Conference & Exposition, 2007.
[85] O.-P. Hilmola, P. Helo, L. Ojala, The value of product development lead time in software startup, System Dynamics Review 19 (1) (2003) 75–82.
[86] C. Yoo, D. Yang, H. Kim, E. Heo, Key Value Drivers of Startup Companies in the New Media Industry-The Case of Online Games in Korea, Journal of Media Economics 25 (4) (2012) 244–260.
[87] Y.-W. Yu, Y.-S. Chang, Y.-F. Chen, L.-S. Chu, Entrepreneurial success for high-tech start-ups - case study of taiwan high-tech companies, Palermo, Italy, 2012, pp. 933 – 937.
[88] M. Fayad, Process assessment considered wasteful, Communications of the ACM 40 (11) (1997) 125–128.
[89] L. Bean, D. D. Hott, Wiki: A speedy new tool to manage projects, Journal of Corporate Accounting & Finance 16 (5) (2005) 3–8.
[90] E. Deakins, S. Dillon, A helical model for managing innovative product and service initiatives in volatile commercial environments, International Journal of Project Management 23 (1) (2005) 65–74.
[91] S. Ambler, Lessons in agility from Internet-based development, IEEE Software (April) (2002) 66–73.
[92] B. May, Applying lean startup: An experience report – lean and lean ux by a ux veteran: Lessons learned in creating and launching a complex consumer app, in: Agile Conference (AGILE), 2012, 2012, pp. 141–147.
[93] M. Taipale, Huitale - A story of a Finnish lean startup, in: Lean Enterprise Software and Systems, Vol. 65 of Lecture Notes in Business Information Processing, 2010, pp. 111–114.
[94] D. Wood, Open Source Software Strategies for Venture-Funded Startups, Tech. Rep. TR-MS1287, MIND Laboratory, University of Maryland (2005).
[95] D. Wall, Using open source for a profitable startup, Computer 34 (12) (2001) 158–160.
[96] T. Clark, P.-A. Muller, Exploiting model driven technology: A tale of two startups, Software and Systems Modeling 11 (4) (2012) 481 – 493.
[97] D. Šmite, C. Wohlin, T. Gorschek, R. Feldt, Empirical evidence in global software engineering: a systematic review, Empirical Software Engineering 15 (1) (2009) 91–118.
[98] T. Dybå, V. B. Kampenes, D. I. Sjøberg, A systematic review of statistical power in software engineering experiments, Information and Software Technology 48 (8) (2006) 745–755.
[99] Colin Robson, Real World Research: A Resource for Social Scientists and Practitioner-Researchers, John Wiley and Sons, 2009.
[100] N. Ali, H. Edison, Towards innovation measurement in software industry, Master's thesis, Blekinge Institute of Technology (May 2010).
[101] T. Dybå, B. Kitchenham, M. Jorgensen, Evidence-based software engineering for practitioners, IEEE Software 22 (1) (2005) 58 – 65.
[102] B. Kitchenham, T. Dybå, M. Jorgensen, Evidence-based software engineering, in: Proceedings of the 26th International Conference on Software Engineering (ICSE), 2004, pp. 273–281.
[103] E. Ries, The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses, Crown Business, 2011.
[104] C. F. Kurtz, D. J. Snowden, The new dynamics of strategy: Sense-making in a complex and complicated world, IBM Systems Journal 42 (3) (2003) 462 –483.
[105] D. Adebanjo, Challenges and approaches to customer development in co-located high-tech start-ups, in: Proceedings of the International Conference on Industrial Engineering and Engineering Management (IEEM), 2010, pp. 163–167.
[106] E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design patterns: elements of reusable object-oriented software, 1st Edition, Addison-Wesley, Boston, 1994.
[107] E. Gamma, R. Helm, R. E. Johnson, J. M. Vlissides, Design patterns: Abstraction and reuse of object-oriented design, in: Proceedings of the 7th European Conference on Object-Oriented Programming (ECOOP), 1993, pp. 406–431.
[108] N. Paspallis, R. Rouvoy, P. Barone, G. Papadopoulos, F. Eliassen, A. Mamelli, A pluggable and reconfigurable architecture for a context-aware enabling middleware system, in: Proceedings of the OTM 2008 Confederated International Conferences, 2008, pp. 553–570.
[109] V. R. Basili, M. Lindvall, M. Regardie, C. Seaman, J. Heidrich, J. Munch, D. Rombach, A. Trendowicz, Linking software development and business strategy through measurement, Computer 43 (4) (2010) 57–65.
[110] C. Liu, J. Yang, L. Tan, M. Hafiz, R2fix: Automatically generating bug fixes from bug reports, in: Sixth International Conference on Software Testing, Verification and Validation (ICST), 2013, pp. 282–291.
[111] I. Banerjee, B. Nguyen, V. Garousi, A. Memon, Graphical user interface (gui) testing: Systematic mapping and repository, Information and Software Technology 55 (10) (2013) 1679 – 1694.
[112] D. Kuhn, Selecting and effectively using a computer aided software engineering tool, in: Annual Westinghouse Computer Symposium, 1989.
[113] P. Smutny, Mobile development tools and cross-platform solutions, in: 13th International Carpathian Control Conference (ICCC), 2012, pp. 653–656.