IET Software

Review Article

IET Journals
The Institution of Engineering and Technology

Software development methodologies and practices in start-ups

ISSN 1751-8806
Received on 12th July 2018
Revised 23rd July 2019
Accepted on 9th September 2019
E-First on 7th November 2019
doi: 10.1049/iet-sen.2018.5270
www.ietdl.org

Esubalew Workineh Tegegne[1], Pertti Seppänen[1], Muhammad Ovais Ahmad[1,2,3] ✉

[1]*M3S Research Unit, University of Oulu, P.O. Box 90570, Oulu, Finland*
[2]*Department of Mathematics and Computer Science, Karlstad University, Karlstad, Sweden*
[3]*Faculty of Electronics, Telecommunications and Informatics, Gdansk University of Technology, Gdansk, Poland*
✉ *E-mail: Ovais.Ahmad@oulu.fi*

Abstract: Software start-ups are aiming to develop cutting-edge software products under highly uncertain conditions, overcoming fast-growing markets under multiple influences. This study aims to identify and analyse the existing scientific literature regarding software development methodologies and practices in software start-ups published between January 2006 and December 2017 using the systematic mapping study. The results identified 37 relevant primary studies out of 1982 papers. To validate the results from the mapping study, an empirical study was based on the research data collected from 14 real-life software start-ups located in Finland, Italy and Norway. The result shows that Agile and Lean start-up methodologies are the most commonly used in software start-ups due to the flexible nature and easy tailoring. A total of 144 software development work practices are extracted from the primary studies. This study contributes to the research in several ways: (i) provides state of the art regarding software development methods and practices in software start-up contexts, (ii) reports commonly used methods along with its benefits identified in primary studies and (iii) identifies opportunities for future software start-up research.

1 Introduction

In recent years, there is a significant increase in the number of software start-ups around the globe. Start-ups are new phenomena in the technology market, creating new businesses and jobs [1]. Technological advancements have brought an increased demand for new software and services. According to Klotins *et al.* [2], such advancement makes to establish new software companies and exploit the technology. Software start-ups are playing an important role in addressing the increasing demand for new software products and services [2]. Many success stories surrounding the software start-ups (i.e. Facebook, Instagram, Uber, Airbnb etc.) contribute to the popularity of start-up phenomenon. These success stories are used as a driving fuel in the software start-up environment. However, despite these success stories, not all start-up companies succeed [3–5]. According to Giardino *et al.* [6], 60% of the total start-ups in the world do not survive the first 5 years from their creation. This has led researchers to study various factors contributing to succeeding or failing of software start-ups [4, 5, 7]. For example, a high failure rate among start-ups is due to lack of resources, immaturity, multiple influences and dynamic technologies. On the other hand, there are assumptions that inadequate software development practices might be a significant contributing factor for the high failure rates in start-ups [2]. Start-up companies develop the software in a situation where resources are very limited and practices are not specified by a prescriptive methodology [7].

Despite the popularity of the start-up phenomenon, the existing literature on software development activities in start-ups is scarce. Few studies have identified, characterised and mapped software development methods and practices in software start-ups [2, 7–9]. According to Klotins *et al.* [2], the existing knowledge about software engineering practices is limited among the start-ups and the existing body of knowledge is insufficient to support engineering aspects in start-ups. The existing published studies [2, 7] on start-ups have limitations, as shown in Table 1.

The mapping study conducted by Paternoster *et al.* [7] has identified, categorised and analysed 213 software development work practices used in software start-up companies. Klotins *et al.* [2] identified 54 software engineering practices in start-up companies, which are mapped to the Software Engineering Body of Knowledge (SWEBOK) knowledge areas [10]. However, both studies [2, 7] have focused on development practices but did not make a deeper analysis of development methodologies.

More recently, two literature reviews, Berg *et al.* [11] and Klotins *et al.* [12], are published in the same vein. The Berg *et al.* [11] mapping study observes how the software start-up research has evolved compared to the software engineering knowledge areas presented in SWEBOK. Berg *et al.* [11] primary studies were 74, obtained from three databases naming Scopus, ISI Web of Science

Table 1 Comparison of last 6 years (2014–2019) literature reviews on software start-ups

Comparison context	Paternoster *et al.* [7]	Klotins *et al.* [2]	Berg *et al.* [11]	Klotins *et al.* [12]	This study (2019)
purpose	extract and analyse software development practices used in start-ups	identify and categorise software engineering practices utilised in start-ups and mapped to SWEBOK	identifying change in focus of research area and thematic concepts operating start-up research	identify key knowledge areas and opportunities for further research	identify and analyse software development methods and practices used in start-ups
years included	1994–2013	1994–2014	1994–2017	2001–2015	2006–2017
no. of primary studies	43	14	74	88	37
sources	scientific literature	scientific literature	scientific literature	experience reports	scientific literature

and Engineering Village Compendex. The study focus was to map the software start-up literature in the light of SWEBOK, the context of start-ups and inferring applicability of empirical findings as well as to identify the research gaps. Whereas Klotins *et al.* [12] investigated on how the software engineering is applied in a start-up context with a focus to identify key knowledge areas and opportunities for further research. The data source for Klotins *et al.* [12] was an experience report from CB Insights website, which is 'global network of executives and startups…' [13, 14]. A total of 88 experienced company reports were analysed to identify various software engineering knowledge areas. Both reviews [11, 12] provide a high level of categorisation of the start-up process and practices dominantly in the light of SWEBOK knowledge areas. Further, 5 years ago, two studies [2, 7] were conducted; however, the start-up environment is fast changing and is dynamic, which needs up-to-date research results. This study aims at filling the gap in the research with a clear focus on development methods, processes and practices in start-ups. Different from the previous mapping and systematic literature reviews [2, 7, 11, 12], this study will be performed as a complementing study conducted by Paternoster *et al.* [7].

This study investigates the software development methods and practices used in start-ups. The study provides state of art regarding software start-ups, synthesis of adopted development methods, its use benefits and provides direction for future research.

The paper is structured as follows: Section 2 describes the background and related work with this study; Section 3 discusses the systematic mapping study (SMS) process (e.g. planning, conducting, reporting) followed to perform the review and the empirical study. The results are presented in Section 4. Section 5 discusses the answers to the research questions; and Section 6 provides a conclusion to the study and future directions.

2 Background

Start-ups are described as a human institution created to build a new product under conditions of extreme uncertainty [15]. Blank [3] defines a start-up as a temporary organisation that creates high-tech innovative products and has no accumulated experience in the development activities. A start-up is a company that is challenged by immaturity, severe resource limitations, multiple influences dynamic technologies and markets [8].

The term 'Software Startup' or 'software package startups' was first introduced by Carmel [16]. Software start-ups are young package firms that create new niche products under a short period of time [16]. Since 1994 various other definitions for the term 'Software Startups' are emerged in the field [8, 9, 17]. Coleman and O'Connor [9] describe software start-ups as unique companies that develop software through various processes and without a prescribed methodology. On the other hand, Sutton [8] characterised the software start-ups by the challenge they face as:

- little or no operating history – software start-ups have little-accumulated experience in development processes and organisation management;
- young or immature compared to more established and mature companies;
- *limited resources*: scarcity in finance, time, human resources and market; typically focus on delivering and promoting the product, and building up customers;
- *multiple influences*: under pressure from investors, customers, partners and competitors in decision-making process;
- *dynamic technologies and markets*: newness of software companies often require to develop or operate with disruptive technologies to enter into a high-potential target market.

Therefore, many researchers shared the above characterisation of software start-ups [7]. From a business model perspective, there are two kinds of software start-ups: product-oriented and project-oriented start-ups. The product-oriented software start-ups develop market and sell their own product, whereas the project-oriented software start-ups do not develop their own product, instead they serve their customers by building products for them [18]. In earlier times, the development of software as a service or a product started as a chaotic activity often referred to as 'code and fix'. Software was written with a little planning, and the design of the product was set by many short-term decisions [19]. As an alternative to this approach, a methodology was introduced. *A development methodology imposes a disciplined process upon software development with the aim of making software development more predictable and more efficient* [19]. A software development methodology is a set of activities, practices or processes that an organisation uses to develop and maintain the software and the associated products (e.g. project plans, design documents, code, test cases and user manuals) [20]. Chapman [21] defined the software development methodology as a documented collection of policies, process disciplines and procedures exercised by a development team or an organisation to practice software engineering activities and software development practices that a methodology defines to be used in the process [22].

It is important to note that software development methodology does not simply imply the development of software code, it covers the entire process involved in developing and maintaining a software product [21]. However, the process is not a rigid instruction on how to develop software. Rather, it is a flexible approach that enables the software team or companies to do the work by choosing the appropriate set of work actions and tasks [20]. As a result, software companies are able to deliver a software product in a timely manner and with sufficient product quality to satisfy their customers. In the past, there had been a number of different methodologies designed to help companies manage their software development activities. However, no single approach has achieved a generalised acceptance, as there is a number of other contextual and situational factors that influence the choice of methodologies and processes [23].

The start-ups develop innovative software and services without strictly following any specific development methodology or practices [7]. However, the existing studies highlighted that various development methods and practices are partially used in software start-ups, such as Agile methods, Lean start-up [2, 7, 11, 15, 22]. According to Abrahamsson [22] the Agile methods emphasise on the relationship and commonality of software developers and the human role reflected in the contracts, as opposed to institutionalised processes and development tools. Agile methods also manifest itself in close team and working environment relationships, as well as other procedures boosting the team spirit [22]. Scrum is a common example of the Agile methodology. The Agile method is used as a project management approach in the context of this publication. Lean start-up method is a product development process, which combines the business-driven hypothesis experimentation and iterative product releases [15]. According to Ries [15], the Lean start-up method enables to build a product iteratively based on the needs of early customers that could lead to reduced market risks such as expensive product launches and failures.

In general, development methods in start-ups have been neglected in the studies or are less familiar [6–8]. In 2015, Klotins *et al.* [2] conducted a study to identify and categorise software engineering knowledge areas utilised in start-ups to map out the state of art. Klotins *et al.* [2] identified 54 software development practices from 14 primary studies and categorised them into 11 main knowledge areas from SWEBOK. The study highlighted that existing research does not provide reliable support for software engineering in any phase of a start-up life cycle. Klotins *et al.* [12] extend their previous study [2] by conducting a multi-vocal exploratory study of 88 start-up experience reports. The study shows *that most engineering challenges in start-ups stem from inadequacies in requirements engineering* and stress that *more research on adaptation of established practices, and validation of new start-up specific practices is needed* [12].

Recently, Berg *et al.* [11] conducted an extensive mapping study on software start-up engineering covering the literature from three databases (i.e. Scopus, ISI Web of Science and Engineering Village Compendex) from 1994 to 2017. The study provides a comprehensive view of software start-ups for software engineering researchers and proposes that future research should focus on start-

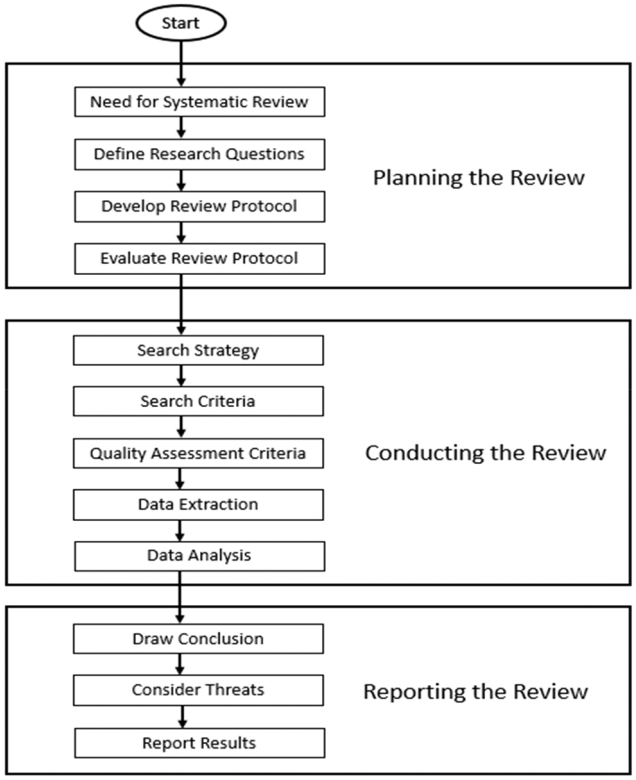

Fig. 1 *Systematic mapping process*

Table 2 Search strings

Criteria	Core concept	Related words
population	software start-up	start-up, software start-up, software start-up, high-tech venture, high-tech start-up, IT start-up, lean start-up
intervention	software development	software development, engineering, implementation, build
comparison	practice	practice, methodology, method, process, strategy, approach, model

up evolution models and human aspects. Paternoster *et al.* [7] provide the state of the art and evidence on software development practices used in start-ups. The study claims that software *development work practices are chosen opportunistically, practiced in a way to provide value under constrains imposed by the startup context* [7]. However, the study reported that there is lack of highlight quality literature and proposed investigation on identifying the software development strategies engineered by practitioners.

The existing studies cover a wide range of areas related to development practices in start-ups [2, 6–8, 11, 12]. Additionally, there is limited fresh knowledge regarding development methodologies in the start-up world. However, little coverage is given in investigating and identifying software development methodologies used by start-ups. This research gap is clearly highlighted by Paternoster *et al.* [7]. This knowledge gap motivated this study. We conducted systematic mapping study along with empirical research with a focus on software development methodologies and practices in the start-up context.

3 Research methodology

3.1 Systematic mapping study

We followed the established SMS guidelines and procedures of Kitchenham and Charters [24]. The SMS process is illustrated in Fig. 1, which consists of three main phases: planning (four steps), conducting (five steps) and documenting (three steps). In the following section, we will discuss these phases in detail.

3.1.1 Planning the review: The planning phase comprises four steps. First, motivation for conducting this SMS is to provide the start of the art related to software development methods and practices used in the context of start-ups. Further, to collect such methods that benefit in the software start-up context. The following research questions drive this SMS:

- RQ1. What is currently known about the software development methodologies and practices in software start-ups?
- RQ2. What are the claimed benefits of software development methods in the software start-up literature?
- RQ3. What are the software development methods and practices used in software start-ups?

RQ1 is designed with the intent to identify empirical evidence regarding software development methodologies in start-ups in existing scientific literature. This will help researchers and practitioners to shed light on the use of development methodologies in software context and open an avenue for future research. Then, explore on what research methods are used in software start-up papers and their quality. Further, RQ1 focused to identify software development practices in software start-ups.

Those studies were eligible for inclusion, which present empirical data on software development practices in software start-ups. The inclusion criteria used were:

- The study should be written in English and published between January 2006 and December 2017.
- The study directly answers one or more of the research questions of this study.
- The study should have a clear focus on software start-ups.
- The study should be available in full text.
- The study can be in the form of systematic mapping study, experience report, applied engineering practices, development models or lessons learned.

Studies were excluded from this SMS if their focus was specifically not on development methods and practices in software start-ups or if they did not provide academic rigour or industry relevance:

- duplicate articles;
- studies not clearly focused on software start-ups nor development methods and practices in software start-ups (non-software companies: biotech, manufacturing etc.);
- studies related to technicalities of start-ups (funding, algorithms, programming languages etc.);
- not peer-reviewed scientific papers (i.e. books, blog posts, presentations etc.).

3.1.2 Conduct the review: Pilot search is an important part of conducting SMS [24], which provides an overall idea about available literature. We use Google Scholar for pilot search because it yields diverse and unbiased literature. The initial keywords, *startups, software, development*, are used in various combinations. The pilot helped to identify that the existing studies use various synonyms for software start-ups i.e. software start-up, high-tech venture, high-tech start-up, IT startup, Lean startup etc. The SMS search strings are structured according to guidelines proposed by Kitchenham and Charters [24] (see Table 2).

Search strings are applied to all the selected data sources. The example of search strings used is as follows:

(‘early-stage firm’ OR ‘early-stage company’ OR ‘high-tech venture’ OR ‘high-tech ventures’ OR ‘start-up company’ OR ‘start-up companies’ OR ‘high-tech start-up’ OR ‘high-tech start-ups’ OR ‘high-tech startups’ OR ‘high-tech startup’ OR ‘startup company’ OR ‘startup companies’ OR ‘software startup’ OR ‘lean startup’ OR ‘lean start-up’ OR ‘lean startups’ OR ‘software startups’ OR ‘IT start-ups’ OR ‘IT start-up’ OR ‘IT startup’ OR ‘IT startups’ OR ‘software start-up’ OR ‘software start-ups’ OR ‘software product startup’ OR ‘software product startups’ OR ‘software start up’ OR ‘web startup’ OR ‘web start-up’ OR ‘web

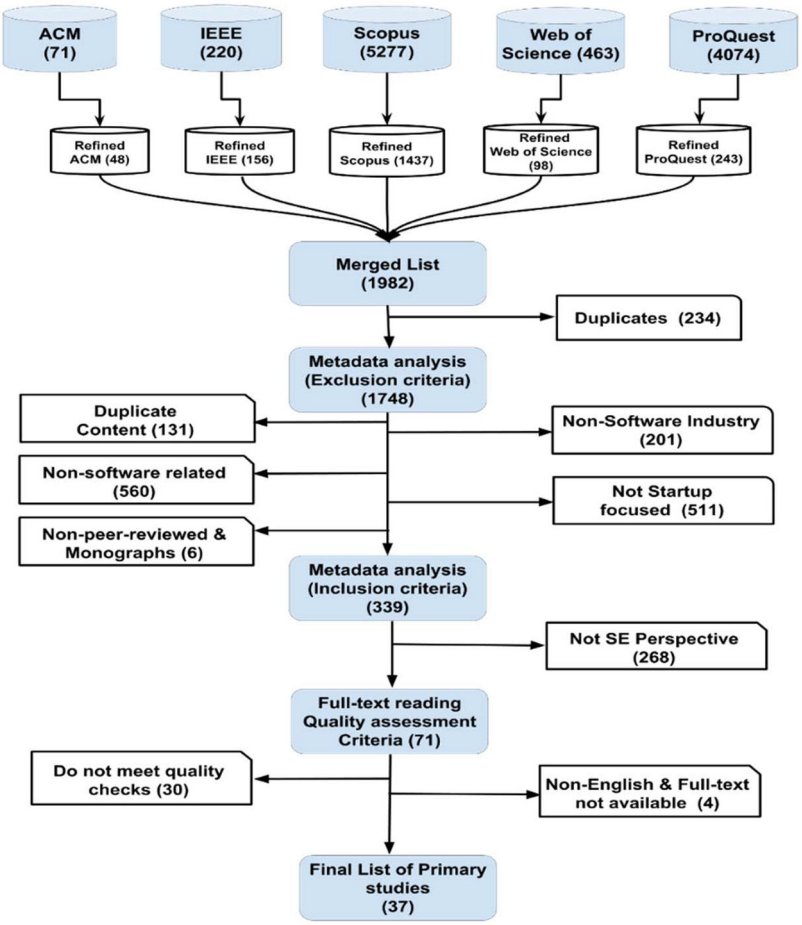

Fig. 2 *Identification of primary studies*

startups' OR 'web start-ups' OR 'internet startup' OR 'internet start-up' OR 'internet startups' OR 'internet start-ups' OR 'mobile startup' OR 'mobile start-up' OR 'mobile startups' OR 'mobile start-ups') AND (develop* OR engineer* OR implement* OR build*) AND (software OR process OR model OR methodolog* OR method OR practice OR strategy).

3.1.3 Identification of primary studies: Five databases were selected to collect relevant papers: *IEEE, Scopus, Web of Science, ACM digital library and ProQuest*. These major electronic databases cover software engineering domain and the ability to handle advanced queries [7]. Each database required specific search string to retrieve an initial list of studies. The advanced functionality of each database helps to focus on specific fields such as software engineering and computer science. The records are imported into Google Spreadsheet. The basic input includes meta-data such as (i) title, (ii) author, (iii) year, (iv) publication type and (v) abstract. To screen the retrieved papers, we follow the best practices proposed by Kitchenham and Charters [24] which is illustrated in Fig. 2.

The search strings in each database were used to search each paper title, keywords and abstracts. At the database level, the papers were screened with the advanced features of each database (i.e. screening by year, language and scope of software engineering domain). This step results in a total of 1982 studies. All the studies were imported to Refworks (http://refworks.com) with the aim to remove duplicate elements. In total, 234 studies were removed as duplicates from the list. Then the results were transferred to Google spreadsheet to study inclusion and exclusion criteria. This step indicated that many publications focused on areas that were outside the scope of this study. Our focus is on software start-ups and their software development methods and practices. There were many start-up studies which focused on a variety of different fields such as agricultural, hardware, electronics etc.

This screening yielded 71 studies for further analysis. The remaining papers were checked in detail for whether they qualify the inclusion criteria. The papers that met the inclusion criteria were transferred for quality assessment. By reading the full-text of the paper, 41 primary studies were selected. The two researchers have disagreements on the quality of four papers. These papers were jointly discussed and decided to exclude owing to the quality problem. Finally, 37 primary studies were obtained.

3.1.4 Quality assessment: The 37 primary studies were assessed on the ten-factor quality criteria proposed by Dybå and Dingsøyr [25]. A binary grade was used for all the primary studies quality assessment, '1' represents 'yes' and '0' represents 'no'. Two researchers independently checked the quality assessment of primary studies:

- Was this a research paper? (or was it merely 'lessons learned' report based on expert opinion?)
- Was there a clear statement of the aims of the research?
- Was there an adequate description of the context in which the research was carried out?
- Was the research design appropriate to address the aims of the research?
- Was the recruitment strategy appropriate to the aims of the research?
- Was there a control group with which to compare treatments?
- Was the data collected in a way that addressed the research issue?
- Was the data analysis sufficiently rigorous?
- Had the relationship between researcher and participants been considered to be an adequate degree?
- Was there a clear statement of findings?
- Was the study of value for research or practice?

Table 3 Interview schema

ID	Information	Topic	Note
initial phase			
IQ01	Do you remember the time (month-year) when you had the initial idea?	pivot	—
IQ02	What was your initial idea?	pivot	—
IQ03	Who were your targeted customers?	start-up model, pivot	—
IQ04	How did you find out about these customers?	start-up model, pivot	—
IQ05	What was the initial team composition?	competency	—
IQ06	How did you come up with the list of features for the first prototype?	start-up model, pivot, testing	—
IQ07	When did you have the first prototype?	competency, pivot, testing	—
IQ08	How did you make the first prototype?	start-up model, competency, pivot, testing	—
IQ09	How did you test it?	testing, competency	—
IQ10	How much effort was spent on testing?	testing, competency	—
IQ11	What are the important quality attributes?	testing	—
IQ12	What were the special competencies needed to make this prototype?	competency	—
significant pivots			
IQ13	How many changes were made up till now?	pivot	—
IQ14	Is the initial idea and the current product same? (use product, marketing, financial, team dimensions)	pivot	—
IQ15	When did you make the last changes in business, market, and product features?	pivot	—
IQ16	What do you think is the significant pivot? Explain.	pivot	—
evolution			
IQ19	Time when you had the first launching	pivot, start-up model	—
IQ20	What is the current size of your customer base?	pivot	—
IQ21	What is the current team composition?	competency	—
IQ22	How had the company grown since it was established? (organic or buying subcontracting)	competency, start-up model	—
IQ23	What employees are currently working in the company?	competency	—
IQ24	Why are these people preferred?	competency	—
IQ25	How do you perform testing now?	testing	—
IQ26	How much effort is spent on testing?	testing	—
IQ27	What is an important quality attribute now?	testing	—
future plan			
IQ28	Assuming growth, which kind of people you wish to reach?	competency	—
IQ29	How will the situation change in the next 3–5 years?	general	—

3.1.5 Data extraction and analysis: The data extraction was started by reading full-text analysis of primary papers. One author extracted the data and other authors facilitated and improved the data extraction process. In the data extraction process, the relevant information related to software development methodologies and practices in start-ups was identified from each primary study, using the original author's terms. Then recording the results in a tabular form to enable comparisons and analysis based on different groupings was made. In summary, the data synthesis was achieved by: (i) identification of software development methodologies used by start-ups from each study and (ii) documentation of a set of reported development practices in start-ups from each primary study.

3.1.6 Data extraction and analysis: The data extraction was started by reading a full-text analysis of primary papers. One author extracted the data and other authors facilitated and improved the data extraction process. In the data extraction process, the relevant information related to software development methodologies and practices in start-ups was identified from each primary study, using the original author's terms. Then recording the results in a tabular form to enable comparisons and analysis based on different groupings was made. In summary, the data synthesis was achieved by:

- identification of software development methodologies used by start-ups from each study;
- documentation of a set of reported development practices in start-ups from each primary study.

3.2 Empirical study

An empirical study was conducted as a comparison to the findings of the literature study. The empirical study followed the guidelines presented by Runeson and Höst [26] for conducting the case study research in software engineering. Interviews were opted for the research data gathering due to practical difficulties tied to other data gathering methods proposed by Runeson and Höst [26]. The empirical study was based on the research data collected from 14 real-life software start-ups located in Finland, Italy and Norway. The data was gathered using interviews with a broad interview schema to cover different viewpoints on software development in start-ups (see Table 3).

3.2.1 Study design overview: The interviews were semi-structured, as presented by Runeson and Höst [26]. The semi-structured interviews with a broad schema allowed the interviewees to focus on areas that were especially relevant to their start-ups. The key informant technique [27] was utilised by selecting the interviewees among the founders or other key persons of the case start-ups, who were assumed to be able to answer the questions addressing software start-ups in a broad manner. In two cases where the founders did not have software development skills or experiences of their own additional interviews were conducted with software experts.

3.2.2 Transcribing interviews, analysis and synthesis procedure: A professional transcription company transcribed the interview recordings to textual documents. We analysed the empirical data qualitatively by conducting a thematic synthesis as defined by Cruzes *et al*. [28] and Cruzes and Dybå [28, 29] by

Table 4 Case startups

Company case summary	Customers	Interviewee(s)	Status
company A Bolzano Italy	B2C[a], internet users	founder	discontinued due to difficulties in customer discovery
company B Trondheim Norway	B2C, B2B[b], internet users, event organisers	founder and software expert	first product on markets
company C Trondheim Norway	B2C, B2B, mobile users, emergency call centres	founder and software expert	first product on markets
company F Oulu Finland	B2B, logistics, manufacturing industries	chief technical officer of the host company (an internal start-up)	prototype series under customer testing
company G Helsinki-area Finland	B2B, Finnair	founder	established business
company H Helsinki-area Finland	B2B, mobile device vendors	founder	first prototypes under customer testing
company I Oulu Finland	B2B, fisheries, biology researchers, underwater construction companies	chief technical officer	established business
company J Oulu Finland		founder	established business
company K Oulu Finland		founder	established business

[a]B2C: business-to-customers.

[b]B2B: business-to-business.

Table 5 Themes and codes identified in the research data

Methods by the book	Company-specific methods
Lean start-up	methods based on personal skills
test-driven development	unsystematic methods
Kanban	own processes
customer cooperation	own methods
continuous integration Agile	informal Agile

utilising both inductive and deductive coding. The coding started with a set of initial codes and new codes were defined along for new topics that emerge from the research data. We utilised the NVivo 11 tool for conducting the thematic synthesis. In the data analysis, it turned out that the search data relevant for software development methods was available from nine start-ups, while in five cases the detailed data on methods was missing. Thus, the analysis is based on nine case start-ups, which are listed in Table 4.

Coding the data: The initial codes were selected based on the reading of the transcribed interviews for deductive coding. The amount of initial codes was small, containing only *Lean start-up, Agile development and own (company-specific) development methods*. While conducting the coding, seven new codes were identified: continuous integration, customer cooperation, Kanban, test-driven development, own processes, unsystematic development methods and methods based on personal skills of the key persons. It is worth noting that the following phenomena were identified during the coding process, affecting the interpretation of the findings:

- The distinction between process-specific and pure-method-specific topics was not clearly definable. Instead, the process-type views and methods views were used in a mixed manner by the interviewees. An example of the former is Kanban, having the main focus on project management, while the test-driven development can be seen as a pure development method, not being dependent on any specific process model. As processes and methods are many times interlinked and the interviewees addressed both in a mixed ways, both were included in this study.
- Although the key concepts of the Lean start-up are generic, not specific for software start-ups, the interviewees referred Lean start-up as a software development method, thus giving us reasoning to include it in the study.
- Differences between own methods and processes, methods based on personal skills, and unsystematic development methods were identified in the research data, and thus they were classified under separate codes. The differences are more closely described in the results section.

Creating themes out of codes: While studying the above-mentioned codes and the research data gathered under them, we could identify a division line that divided the very heterogeneous set of software development methods of our research data into two clearly separate approaches: software development methods by the book and company-specific software development methods. The division gave us reasoning to define two themes according to the two approaches, methods by the book and company-specific methods.

While we were able to classify most of the codes to one approach only, Agile methods turned out to be deployed in the case start-ups both by the book and by company-specific ways. Thus, we divided the Agile development code into two, Agile by the book and informal Agile. The results of the thematic synthesis are shown in Table 5.

Creating a model of themes and codes: The next step of our study was to create a model describing the utilisation of software development methods in our case start-ups, as recommended by Cruzes and Dybå [29]. The model was created by drawing a distribution chart that shows the identified themes and codes with bubbles representing the share of each identified code. The amounts of NVivo11 references under the codes were used to

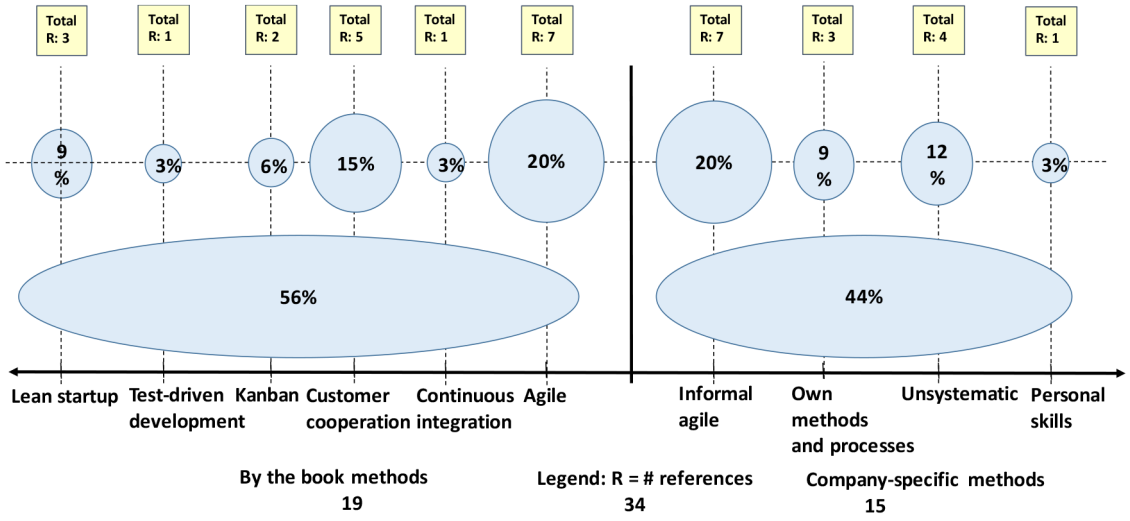

Fig. 3 *Distribution of software development methods identified from the research data*

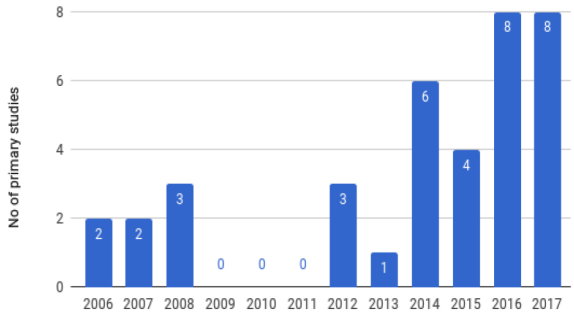

Fig. 4 *Publication distribution by year*

define the sizes of bubbles, and both absolute numbers of references and their percentages compared to the total amount of references were shown in the chart. The distribution chart is presented in Fig. 3. Due to a close relationship of own processes and own methods, those two codes are drawn as a common bubble in the chart.

4 Threats to validity

Validity threats are major factors that influence the accuracy of research negatively. To evaluate and mitigate threats to validity, we follow guidelines provided by Wohlin *et al.* [30].

Construct validity relates to obtaining the right measures for the concept being studied [26, 30, 31]. In qualitative studies it referred to the interview questions interpretation and understanding. To reduce this threat, all the SMS data collection process is exhibited in Fig. 2 and interview questions are pre-tested. At the beginning of our interview, the researchers explain the study goal and key terminologies in order to avoid misunderstanding. Additionally, two authors acted as external reviewers to validate the research protocol. Further, there is a possibility of bias in the nature of software start-up papers which are subjective in nature. However, due to a large set of data from primary studies and diverse interviewees' population this threat is minimised. To provide strength to internal validity, a small number of data can be analysed using the grounded theory [12].

Internal validity relates to causal relationships and ensuring that it is not a result of a factor that was not measured or the researcher had no control over. This study does not consider the internal validity threat because we are not aiming for statistical causal relationship on software development methods and process in software start-ups. However, its internal validity is also linked to researcher biasness during the analysis and coding process. During the analysis the researcher without his/her knowledge might ignore a factor which is unknown and mix up the studied relationship with known factors [12]. To minimise this threat, the SMS data is complemented with the qualitative data, where the analysis is conducted by at least two researchers. The results are discussed and compared with the state of the art.

Conclusion validity relates to the bias of researchers in the interpretation of that data. The investigator biasness is difficult to eliminate completely. However, we took the following actions to reduce its intensity: Two authors were involved in the analysis of the primary papers; a full record of all retrieved papers is maintained to show how 37 primary studies were identified. These primary studies were read and conclusions were drawn by at least two authors. This same technique was applied to data collection, analysis and reporting the results of interviews.

External validity relates to the extent to which the study results are generalisable [30, 31]. To mitigate external validity, this study rigorously followed the guidelines provided by Petersen *et al.* [31] and Dybå and Dingsøyr [25]. For instance, the data was collected from 14 software start-ups from three European countries (Finland, Italy and Norway). Threats to study selection in SMS are handled with the help of inclusion and exclusion criteria. Additionally, the different contextual factors, such as type of product, competitive landscape etc., also bring the bias. However, following the tradition, we take into consideration the dimensions of Macmillan *et al.* to facilitate a broader reasoning related to the reasons that obstruct the triumph of software start-ups [1, 32]. We argue that these different contextual factors provide us rich findings. Nevertheless, the results should be considered with caution, because our SMS and participated companies in the present study cannot be assumed to represent software start-up companies in general. The number of participating companies can be considered as one limitation of this study. However, the start-up companies provide valuable and rich information. The interviewees provide similar information and insight to development methods and practices in software start-ups. This means not much new information was added in later interviews. This is an indication of information saturation sufficiency so no more interviewees were needed. Hence, this threat is considered to be under control.

5 Results

5.1 Results of systematic mapping study

This section presents the results of the analysis of 37 primary studies selected from five databases. Section 5.1 provides an overview of primary studies related to (i) publication year, (ii) research method adopted, (iii), quality assessment of primary studies and (iv) research focus. Section 5.2 presents the state-of-the-art analysis of development methodologies in software start-ups and Section 5.3 discusses development practices used in start-ups, according to the research question.

5.1.1 Overview of primary studies: The growing trend of the publications is exhibited in Fig. 4, which covers the annual number of studies published on software start-ups between 2006 and 2017.

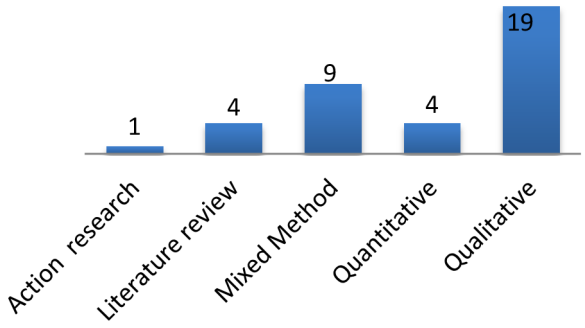

Fig. 5 *Research methods used in primary studies*

Table 6 Primary studies and their research focus

Study ID	Research focus
[9, 35–46]	process management
[2, 4, 6, 7, 18, 20, 34, 47–61]	software development
[62, 63]	managerial/organisational

Table 7 Development method used in software start-ups

Method	Frequency	Percentage	Primary studies
Lean start-ups	14	36	[18, 34, 36, 39, 45, 47–51, 53, 57, 61, 63]
Agile in general	10	26	[20, 37, 39, 41, 43, 44, 55, 56, 59, 60]
ad-hoc management	4	10	[40, 50] [46, 60]
prototyping	2	5	[4, 35]
XP	2	5	[9, 58]
model-driven development	1	2.5	[38]
waterfall	1	2.5	[9]
RUP	1	2.5	[9]
distributed scrum	1	2.5	[50]
hyper-agility	1	2.5	[54]
iterative/incremental	1	2.5	[20]
tailored Agile	1	2.5	[50]

There were only seven studies (19%) published prior to 2012, whereas 30 studies (81%) were published in 2012 and later. Therefore, it can be argued that there is a growing interest regarding software start-ups in recent years.

Our inclusion criteria did not yield any publication between 2009 and 2011. However, such observations are aligned with Paternoster [7] because no primary studies fully focused on software development methods and process in those years. The exception is that one study [33] was relevant to this SMS but did not pass quality assessment criteria.

The primary studies used diverse research methods and techniques as shown in Fig. 5. Qualitative method using interviews is the predominant (51%) data collection technique adopted in primary studies. In qualitative studies, the author of one study [34] considered the method used as 'startup stories' due to small amount of data collected.

Nine primary studies (24%) used mixed method approach (i.e. survey and literature review, server logs and interviews, mapping study and interviews). Only one paper adopted design science approach and constructed an artefact to evaluate developer's work in software start-ups. The remaining 12 primary studies are quantitative ($n = 4$), systematic mapping ($n = 4$) and one study comes under action research category. Table 6 shows the research focus of primary studies, which is categorised into three groups i.e. software development, process management and managerial/organisational. The classification of research focus is based on the definitions and categories adopted from Paternoster *et al.* [7].

Table 8 Lean use benefits in software start-ups

Lean benefits	Primary studies
creating an MVP for quick learning in product development	[34–36, 47, 49–51] [57, 61]
customer development by 'getting out of the building'	[36, 47, 49, 51] [34, 53]
directly validating the business with the real people	[47, 50, 51] [45, 53]
validating the business type and growth hypotheses through build >measure > learn feedback loop	[35, 47, 49] [57, 61]
reduce waste by eliminating unnecessary development	[36, 47, 49] [34, 53]
possibility of pivoting if the product is unsuccessful	[34, 37, 57] [45]
help to create functioning products based on customers' needs	[36, 47, 49, 50]
quick feedback from customers	[36, 49, 53, 57]
continuous testing and integration	[34, 61]
help in searching for and validating a viable, sustainable and repeatable business model	[47, 51] [45, 57]

Software development is the *engineering activities used to write and maintaining the source code* [7]. More than half of the primary studies (60%) focus on software development activities in start-ups. For instance, the study in [49] introduced a strategy to release the product as quickly as possible, speeding up the development through low-precision engineering activities. Further, Tingling and Saeed [58] focus on the adoption of Agile methodology and extreme programming (XP) principles as a solution for problems related to traditional software developments in start-ups.

Process management means *engineering methods and techniques used to manage the development activities* [7]. The 35% of the primary studies focus on the investigation of process management that is used in software start-ups. Such as the study in [36] investigates the challenges of Lean start-up approach and customer development in building products and businesses in start-ups. Accordingly, the study in [9] characterised the experiences of start-up companies in developing processes to support their software development activity.

Managerial/organisational concern with *aspects that are related to software development, by means of resource management and organizational structure* [7]. Only 5% of the primary studies focus on managerial and organisational aspects of software development. For example, Chorev and Anderson [62] identify the critical factors for the success of start-up companies in Israel. We can observe that the majority of primary studies (35 out of 37 studies) are related to software development and process management in start-ups, which we considered to be fundamental for our research questions (software development methodologies and practices). This indicates that the selected primary studies have a strong contribution and high relevance to answer the research questions.

5.1.2 Development methodologies in software start-ups: The analysis of primary studies identified 12 different software development methodologies used by software start-ups (see Table 7). The result indicates that Lean start-up is 36% and Agile, in general, is 26% widely in start-up companies. Other methodologies include prototyping, model-driven development, waterfall, rational unified process (RUP), and *ad-hoc* methodology.

Two studies [42, 62] have not specified any software development method, but included as primary studies because they reported development practices. Some primary studies reported multiple start-up experiences in one paper and some start-ups used multiple development methodologies. Our unit of analysis is development methodology studies.

Lean in start-ups and its benefits are reported in 16 out of 37 primary studies. The use of Lean in software start-ups yields a number of benefits which are reported in Table 8. Lean start-up

[15] supports the identification of the most risky parts of a software business and produces a minimum viable product (MVP) to systematically test and plan the modified version of next iteration as well as empower development team [7, 49].

Promising principles of customer development are the first motivation factor for using the Lean start-up. Batova *et al*. [36] define the customer discovery as a process in which business owners develop hypotheses about their business models and then validate or invalidate those hypotheses by researching potential customers. This supports the idea of customer development which aims to come up with new viable products and services under conditions of extreme uncertainty. It involves the production of a functioning prototype guided by customer feedback [36, 49]. With this tangible prototype, it helps to approach anticipated customers. Customer discovery leads to the validation of the product idea or prototype and, if proven, the creation of the product [34, 53]. Therefore, this results in a strong customer development and relationship [49].

Waste reduction is important in production processes and making these processes more efficient by eliminating unnecessary research and development [49]. Before developing products or even prototypes, start-ups should find out whether they have actual customers and understand what these customers want [36]. As a result, start-ups avoid wasting time in building unwanted functionality and preventing exhaustion of financial, human and time resources for the start-up companies [36, 47, 53].

Build-measure-learn is the principle of Lean start-ups [15]. At early stages of a start-up, the prototype-centric development approaches are commonly practiced by case start-ups under [35] to validate the business idea. This is supported by development of throw-away or rapid prototypes by visualising the business idea on a paper or software. Start-ups at the early stage apply fast cycles of 'build and fix' when necessary to act quickly and decisively enough to get the first response from the market [49]. Developing a set of suitable functionality gives the chance for developers to present a prototype to a potential customer and get productive feedbacks.

5.1.3 Agile in start-ups and various obtained in primary studies: The use of Agile in general and more specific methods are identified i.e. distributed scrum [50], XP [9, 58], hyper-agility [54], iterative and incremental [20] and tailored Agile [50]. The Agile is described as a preferred methodology for start-ups for its benefits (see Table 9).

Start-ups are attracted to the Agile methodology with the promise of shorter development schedules and greater delivery flexibility and support to their business goals for small software companies [41]. Taylor *et al.* [56] indicated that using Agile methods in start-up companies improves the management of the development process and customer relationships. These development methods have been considered the most viable due to a number of benefits i.e. embrace changes rather than avoiding them, allowing development to follow the start-ups' business strategy, shortens the lead time from idea conception to production with fast deployment [39, 50, 59].

Table 9 Agile use benefits in software start-ups

Agile benefits	Primary studies
fast releases using iterative and incremental approach	[20, 39, 43, 50] [59, 60, 63]
shorter development time	[20, 41, 43, 50] [39, 44]
response to change	[37, 39, 43, 59]
delivery flexibility	[39, 41, 43, 56]
customer collaboration and frequent customer feedback	[20, 37, 59, 60]
self-organising collaboration between cross-functional teams	[43, 44, 56]
customer satisfaction/relationship	[20, 56, 59]
team motivation/enthusiastic development teams	[43, 44, 56]
continuous delivery	[20, 48, 56]
easiness to manage small teams	[44, 56]

Evolutionary prototyping or prototype-centric development methodology was adopted by case start-ups in primary studies [4, 35]. According to Giardino *et al*. [4], start-ups prefer to build an initial quick prototype and afterwards build the real product after validating the product in the market. Throw-away prototypes are rapidly constructed on paper/software to visualise the business idea whereas evolutionary prototypes are construction of the launching product through a detail and gradual planning [4]. Such prototype development helps to obtain fast user responses and quick product validation, followed by building a functioning prototype and iterate it over time.

XP importance in software start-ups is also visible in primary studies [9, 58]. Coleman and O'Connor [9] state that XP is flexible and developer-centred development method with the advocacy of self-empowered teams and shared ownership. It is also associated with an 'embrace and empower' style of management in software start-ups. Tingling and Saeed [58] described that XP has a generative set of guidelines that consist of 12 interrelated principles. Some of these principles include 40 h work week, coding standards, collective code ownership, continuous integration, pair programming, refactoring etc. The extent of adopting the principles of XP in start-ups is affected by temporal conditions and institutional maturity of the start-ups, and both management and developer cultures are also important determinants [58].

5.1.4 Software development practices in start-ups: From 37 primary studies, a total of 144 work practices were extracted, which are clustered into three categories: development practices, process management practices and managerial or organisational practices.

Software development practices are defined as *engineering activities used to write and maintaining the source code* [7]. Common development practices among software start-ups include prototyping and experimenting via existing components, continuous value delivery, use of easy-to-implement tools to facilitate product development [6], develop only what is needed for now [18], goal-driven rapid development of technology rather than process directed [54], use of open source solutions, use of key performance indicators to assess the consumer's demand [7] and pair programming [58]. Prototyping is the most common development practice among start-ups to focus on implementing a limited number of suitable functionalities (MVP) and to launch the product on the market as quickly as possible [6, 35, 48–52, 57]. However, certain practices such as refactoring and test-driven development may not be considered to be viable practices for software start-ups, especially at the very early stage [39]. This might be due to a constant pressure of time-to-market demand which brings low priority for product quality.

Process management practices in start-ups include customer-centric development, use customer feedback [35, 36], daily Scrum stand-up meetings, Kanban board [47], process tailoring [9, 35, 56], performing incremental process improvement actions, adopting short and light software process improvement and release management process model [41], pivoting practices [34, 35], and incremental and iterative establishment of software processes [42]. Software process improvement actions and process tailoring practices are the most commonly used process management practices explored in this review. It is argued that adoption of release management process helps to improve the workflow of a start-up company [40]. Coleman and O'Connor [9] explain that start-up companies tailored a process model, by dropping some of the practices and adding the new ones which suit their environment. Such tailoring process is an influential factor for success in start-ups [45].

Managerial or organisational practices include aspects that are related to software development, by means of resource management and organisational structure. These include core team expertise, and diversified team management, careful selection of personnel [62], managerial experience [9], building small omni-

functional teams with no functional boundaries [54], focusing on team building [34], team empowerment [7].

In general, development methodologies have their own defined principles and practices, but they share common values that can be achieved by each practice. For a start-up, it is difficult to choose a proper practice from a pool of different practices. Identification of a set of development practices is essential for start-ups to help process tailoring. Since start-ups are flexible in selection of practices from different methodologies, it can be an input for practitioners to better apply methodologies in a tailored way. Sometimes, development practices are interrelated between different methodologies. As a result, the selection of preferred practices fits the needs of a software start-up. The selection of a suitable development practice helps software start-ups to maximise the benefit of adopting a new development methodology [59].

5.2 Results of interviews

In the empirical study, we identified 11 software development methods divided into two high-level approaches, methods by the book and company-specific methods (see Fig. 3). The results indicate that the two high-level approaches have an almost identical distribution, methods by the book having a slightly bigger distribution, 19 references compared to 15 references of the company-specific methods (56 versus 44%). One finding of the empirical study is the dominance of Agile methods among the case start-ups, being in line with the findings of the literature study. Up to 40% of the identified references showed deployment of Agile methods. Notable is, however, that the start-ups do not necessarily follow Agile methods as they are formally defined, but tend to define company-specific implementations of them. In our research data the codes of Agile methods by the book and in a company-specific way had equal distributions, 20% of the total references.

In a relatively small group of case start-ups, the other formally deployed methods, Lean start-up, test-driven development, Kanban, continuous integration and customer cooperation, collected only fairly small amounts of references each. The most frequently utilised method was customer cooperation. It was classified as a separate code, although it is a cornerstone of the Lean start-up. The reason is that it appeared in the research data also as a generic method, independent of the Lean start-up. As mentioned above, the implementation of Agile methods dominated among company-specific methods. The three other codes collected together only a slightly bigger amount of references, 24 versus 20% of the Agile methods.

Compared to the methods deployed by the book, the company-specific methods are hard to classify at a more detailed level. The classification shown in Table 5 and used when creating the chart of Fig. 3 was based on the details identified in the research data. The classification can be interpreted as a case-specific one, reflecting the findings of this study as well the prior studies on start-ups [2, 7]. However, it should not be a basis of any generic conclusions. A close look on the references under the category of company-specific methods, including the company contexts and the broader research data of the related companies, revealed that there were besides unsystematic *ad hoc* ways of work a set of practices and methods that did not fall in the category of any established methodology. However, they were justified and utilised in a systematic way. Further, in that category, cases arose where the justification of utilising certain practices was the key persons' earlier experiences and personal skills. Due to the specific weight put on those cases by the interviewees, we separated those cases under a code of their own.

An interesting detail is that unsystematic methods gained the biggest amount of references after informal Agile methods in the category of company-specific methods. It indicates that in our group of start-ups there were cases where the initial development team was both unable to take any formal method into use and unable to derive systematic methods and practices from the earlier experience and skills of the individual team members. Taking a look on Fig. 3, we can conclude that group case start-ups have only a small minority of references, 12%, fell under the code of unsystematic methods. The other identified methods were utilised by the book, were implemented to fit the companies' situations, or were based on the earlier experiences and skills of the key persons. The findings give, thus, reasoning to the conclusion that in our group of case start-ups the value of systematic approaches in organising the software development work was recognised, even though not always implemented along the formally defined ways.

6 Quality of primary papers

To check the quality of the 37 primary studies, we assessed each study using the ten-factor framework proposed by Dybå and Dingsøyr [25]. The accumulation of this quality assessment is presented in Table 10. The research aim of all primary studies was ranked '1' as it was clearly elaborated along with the context and how the study was conducted.

The 37 primary studies in the mapping study provide a clear research aim and described the context in which research is conducted. Nevertheless, the research design of some primary studies was not sufficiently discussed, as four primary studies were literature reviews, where sampling was not applicable. The primary studies adequately elaborated data collection and analysis. Additionally, findings are presented appropriately in primary studies. Our analysis shows that 18 out of 37 primary studies (49%) got full points from the quality assessment.

7 Discussion

In this paper, we have applied a systematic mapping method to analyse the 37 primary papers, to provide the state-of-art knowledge of the methodologies and practice of software development in start-up companies. The literature findings are further investigated in 14 software start-ups in three European countries to elaborate how start-ups choose methodologies to manage their development processes.

The SMS shows that an increasing number of publications can be an indicator of an increasing interest in software start-up research. The majority of primary studies uses qualitative research (15 studies) and mixed research methods (9 studies). The mixed methods are used in various combinations such as survey and interview, systematic mapping study, interviews etc. These methods provide rich data; however in the context of our review primary studies, maturity is not explicit (i.e. when and what software development method and practices was initially adopted and how frequently they were used). In terms of research focus, primary studies were categorised [7] into three groups i.e. software development, process management and managerial/organisational.

The major portion of primary studies was 'software development' ($n = 22$) and 'process management' ($n = 13$), where focus is on software development-related activities. Almost all primary studies claimed to take advantages of 'Lean start-up' and 'Agile methodologies', only 2% used the waterfall model. These results are aligned with previous studies [2, 6, 7].

In software start-up context, there is no clear guidance about the selection and adaption of development methods, processes and practices. The primary studies show that software start-ups opportunistically select a method and customise according to their needs. In early stages, mixed development approaches and practices due to a rapidly changing environment were applied. One reason is that start-ups are more chaotic and unpredictable, and it is difficult to accurately predict the risks and the required practices to develop a product. The project managers prefer lightweight methodologies due to flexibility in order to adopt tailored practices for software development fast response according to business strategies. Yan and Murphy [64] also highlighted that start-ups need to define their own development processes based on their needs, requirements and situation. These findings are in line with existing software start-up literature [7, 9, 37, 40, 64].

Applying strictly the principles of a specific methodology is difficult at early stage of a start-up [37]. Coleman and O'Connor [9] explain that process tailoring is made by dropping some of the practices of a specific methodology and adding some new practices from other methodologies which reflected their own particular needs. Such tailoring helps start-ups to select a process that is suitable for the company's business strategy (i.e. [9, 37]). It is

evident that start-up companies pick development practices that suit their start-up situation and most of the process is tailored. The flexible and reactive methodologies favour changes in making decisions, development processes and learn from failures are most suitable in the start-up companies. Learning from failures is an important factor in choosing and adopting effectively a development methodology in a start-up company. Developers should have the freedom to choose development practices quickly and change or stop immediately when the process goes wrong, fix the approach and learn from previous failures. Further, empowerment of team members is considered a productive managerial/organisational mechanism for better responsibilities and improved expertise.

In software start-ups, small collocated teams have effective communication and collaboration as well as solving problems quickly [35]. Pair programming is one of the Agile practice where two teammates work in pair. The primary studies support that pair programming and small teams are good to fix code defects and complex problems [58].

Customer support plays an important role in successful adoption of development methodologies in software start-ups and speedy product deliveries. An active customer participant in providing requirements and updates frequently enables swift adoption of software methodology. Taylor *et al.* [56] highlighted that sometimes the customer provides initial requirements and remains practically uninvolved until the end of project deadline. In such a scenario, start-ups face a high level of uncertainty and may risk of change of a product feature(s), which could cost more time and resources. A motivated customer who provides support throughout the development process enables the smooth process adaption and helps to deliver MVP [56, 64]. However, the focus is on speedy delivery of the product to compare the quality attributes. Start-ups usually exercise speed-related practices (such as short iterations, iteration planning and release planning) than quality-related practices (such as unit testing, refactoring, test-driven development). It seems that quality concerns have low priority than speed-related practices especially at their early stages [39, 64]. Daily stand-up meeting is a well-known Agile practice that is frequently used in software start-ups to facilitate communication. Since most start-ups have very small size of teams, informal communication happens frequently and daily stand-up meetings are suitable to manage communication among team members. However, Lean start-up and Agile processes do not suit for every problem domain in start-up cases. The development processes in start-ups are evolutionary in nature, and the product is obtained by iterating and updating an early prototype following the customer feedback [7].

The primary studies reported a number of benefits of using the Agile and Lean software start-ups. For example, start-ups are capable of creating fast and flexible minimum variable product, direct validation of the business with the real people and learn quickly due to short feedback loop. Agile helps to enable customer

Table 10 Quality assessment of primary papers

Primary studies	Research	Aim	Context	Design	Sampling	Control	Data collection	Reflexivity	Finding	Value	Total score (10)
[35]	1	1	1	1	1	1	1	1	1	1	10
[47]	1	1	1	0	0	0	1	0	1	1	6
[36]	1	1	1	0	1	1	0	1	1	1	8
[48]	1	1	1	0	0	0	0	1	1	1	6
[37]	1	1	1	1	1	1	1	1	1	1	10
[49]	1	1	1	1	1	1	1	1	1	1	10
[62]	1	1	1	1	1	1	1	1	1	1	10
[38]	1	1	1	0	0	1	0	0	1	1	6
[9]	1	1	1	1	1	1	1	1	1	1	10
[50]	1	1	1	1	1	1	1	1	1	1	10
[18]	1	1	1	0	0	0	1	0	1	1	6
[39]	1	1	1	1	1	1	1	1	1	1	10
[6]	1	1	1	1	1	1	1	1	1	1	10
[4]	1	1	1	1	1	1	1	1	1	1	10
[51]	1	1	1	0	0	0	1	1	1	1	7
[40]	1	1	1	0	0	0	1	0	1	1	6
[2]	1	1	1	1	1	1	1	1	1	1	10
[52]	1	1	1	1	1	1	1	1	1	1	10
[53]	1	1	1	0	0	0	1	0	1	1	6
[54]	1	1	1	1	1	1	1	1	1	1	10
[41]	1	1	1	1	0	1	0	1	1	1	8
[34]	1	1	1	1	0	0	1	1	1	1	8
[7]	1	1	1	1	1	1	1	1	1	1	10
[55]	1	1	1	0	1	0	1	1	1	1	8
[20]	1	1	1	1	1	1	1	1	1	1	10
[56]	1	1	1	0	0	1	0	1	1	1	7
[57]	1	1	1	0	0	1	1	1	1	1	8
[58]	1	1	1	0	1	1	1	1	1	1	9
[42]	1	1	1	0	1	1	1	1	1	1	9
[43]	1	1	1	0	0	0	0	1	1	1	6
[63]	1	1	1	1	1	1	1	1	1	1	10
[44]	1	1	1	0	1	1	1	0	1	1	8
[59]	1	1	1	1	1	1	1	1	1	1	10
[60]	1	1	1	1	1	1	1	1	1	1	10
[45]	1	1	1	1	1	1	1	1	1	1	10
[61]	1	1	1	0	1	1	0	1	1	1	8
[46]	1	1	1	0	1	0	1	0	1	1	7

collaboration and frequent feedback, and the team members become more self-organised and manage tasks easily. However, a number of challenges also highlighted in primary studies such as people behaviour, lack of owners or/and project manager in using light weighted methods and practices.

The findings of SMS and our empirical study showed that software start-ups commonly use Agile methods and practices. Start-ups do not necessarily follow the principles of each methodology as they are formally defined, tailored to company-specific implementations of them. It is also evident that, early phases of software start-ups, they are dominant by non-development activities. Most of them are activities related to business and customer discovery with not much focus on software development process. They focus on idea refinement, team building and funding activities.

In a nutshell, software start-ups are using mixed or tailored methodologies. There is no 'complete solution' methodology for software start-ups. The tailored method from Agile and Lean start-ups is considered to be a better option to fit with the culture and needs of start-up companies. Other studies highlighted similar results that start-ups often replace engineering processes by light-weight *ad-hoc* processes [6, 11]. Nguyen-Duc *et al.* [34] proposed a model to help start-ups in all phases of the company ranging innovative ideas to commercial products. However, the model requires empirical evidence to generalise the results [11, 34].

8 Conclusion

Software start-ups have significant contribution in today's global economy. Software development activities are one of the main daily activities in software start-ups. In this study, we analyse development methodologies and practices adopted in software start-ups using the mix method approach (i.e. systematic mapping literature review and interviews). The SMS identified in total 1982 articles in which 37 papers were selected as primary studies published between 2006 and December 2017. These primary studies were analysed with respect to (i) frequency of publication by year, (ii) research method, (iii) adopted methods and practices in software start-ups and its benefits and (iv) quality of primary studies.

The results of our SMS show that there is an increasing number of academic researches on software start-ups. The majority of academic literature is based on a single case study approach using interviews. There is a growing trend that start-ups are paying more attention to software development processes compared to managerial or organisational structure. The software start-ups are dominantly using the Agile and Lean start-up development methodologies due to a number of reasons, such as flexibility in the process, adaptability to a frequently changing environment, makes ease communication with the customer and fast delivery of product to customers. Nevertheless, the SMS also shows that start-ups do not strictly follow methodology principles due to limited resources, time pressure and people behaviour. The structured adoption of any development methodology is challenging in the context of software start-ups. This results in a solution called tailoring methodology *ad-hoc*-based adaption.

Start-ups preferred to tailor methodology which matches to their situation by dropping some of the features in the existing methodology and adding new practices from other methods. An important factor in the adoption of software development methods and practices is the experience of the owners or/and project managers, the maturity level of the start-ups, team size, availability of resources. However, still there is a scarcity of studies on software development practices. The potential of software development practices is not fully exploited by start-ups or not reported in scientific literature. Klotins *et al*. [2] explain that the existing literature on engineering practices in start-ups is not transferable to practitioners.

In general, software development methodologies in start-ups are informal, tailored and very light weighted methods. There is no 'complete solution' methodology for software start-ups. The selection and adoption of appropriate methodology is a difficult process and requires a deep analysis of the company itself, the stage of the company, team size and behaviour and the type and stage of the product concerned. There is no longitudinal study on the use of development methods and adoption. Therefore, we do not have information about their success, failure nor their current status in the market.

Future research requires to strengthen the results of this study. Due to the scarcity of studies and inadequate level of reporting rigour in the primary studies, we believe longitudinal studies with high rigour research are needed. It is recommended to evaluate the effectiveness and suitability of development methods and practices in a software start-up. Start-ups change their development methods and practices frequently due to a rapidly changing environment. This hints for a research topic to study the software methodologies and practices which are suitable for different start-up situations. Practices that are identified in this study are the commonly used practices in software start-ups, and do not reflect whether they are the good or bad practices. For that reason, it is necessary to evaluate the efficiency of each practice on the start-ups and categorise them as 'good' or 'bad' practices in the start-up context. In software start-ups, there is a high research gap in identifying a suitable development methodology for a specific start-up context. Based on the results of this study, there is no suitable methodology for all kinds of start-ups. A model proposed by Nguyen-Duc *et al.* [34] for start-ups is available but lacking the empirical evidence. Future studies are needed in the formulation of a new development methodology which could be beneficial or more suitable in the software start-up context.

9 Acknowledgments

The authors would like to thank the editor and anonymous reviewers for their suggestions on earlier versions of this paper. The literature data collection from databases and demographic analysis is only conducted by the first author. The qualitative data is collected by the second author. Finally, the whole paper is concluded by the second and third authors.

10 References

[1] Giardino, C., Bajwa, S.S., Wang, X., *et al.*: 'Key challenges in early-stage software startups'. Int. Conf. on Agile Software Development, Helsinki, Finland, May 2015, pp. 52–63

[2] Klotins, E., Unterkalmsteiner, M., Gorschek, T.: 'Software engineering knowledge areas in startup companies: a mapping study'. Int. Conf. on Software Business, Braga, Portugal, June 2015, pp. 245–257

[3] Blank, S.: '*The four steps to the epiphany: successful strategies for products that win*' (K&S Ranch, Pescadero, CA, USA, 2006)

[4] Giardino, C., Wang, X., Abrahamsson, P.: 'Why early-stage software startups fail: a behavioral framework'. Int. Conf. of Software Business, Paphos, Cyprus, June 2014, pp. 27–41

[5] Blank, S.: 'Embrace failure to startup success', *Nature*, 2011, **477**, (7363), pp. 133–133

[6] Giardino, C., Unterkalmsteiner, M., Paternoster, N., *et al.*: 'What do we know about software development in startups?', *IEEE Softw.*, 2014, **31**, (5), pp. 28–32

[7] Paternoster, N., Giardino, C., Unterkalmsteiner, M., *et al.*: 'Software development in startup companies: a systematic mapping study', *Inf. Softw. Technol.*, 2014, **56**, (10), pp. 1200–1218

[8] Sutton, S.M.: 'The role of process in software start-up', *IEEE Softw.*, 2000, **17**, (4), pp. 33–39

[9] Coleman, G., O'Connor, R.V.: 'An investigation into software development process formation in software start-ups', *J. Enterp. Inf. Manag.*, 2008, **21**, (6), pp. 633–648

[10] IEEE Computer Society: '*Guide to the software engineering body of knowledge (SWEBOK-2004 version)*' (IEEE Computer Society, Los Alamitos, CA, USA, 2004)

[11] Berg, V., Birkeland, J., Nguyen-Duc, A., *et al.*: 'Software startup engineering: a systematic mapping study', *J. Syst. Softw.*, 2018, **144**, pp. 255–274

[12] Klotins, E., Unterkalmsteiner, M., Gorschek, T.: 'Software engineering in start-up companies: an analysis of 88 experience reports', *Empir. Softw. Eng.*, 2019, **24**, (1), pp. 68–102

[13] cbinsights.com, 2015. Available at www.cbinsights.com/blog/startup-failure-post-mortem/

[14] Lethbridge, T.C., Sim, S.E., Singer, J.: 'Studying software engineers: data collection techniques for software field studies', *Empir. Softw. Eng.*, 2005, **10**, (3), pp. 311–341

[15] Ries, E.: '*The lean startup: how today's entrepreneurs use continuous innovation to create radically successful businesses*' (Crown Books, New York, NY, USA, 2011)

[16] Carmel, E.: 'Time-to-completion in software package startups'. Proc. 27th Hawaii Int. Conf. on System Sciences (HICSS), Wailea, HI, USA, January 1994, pp. 498–507

[17] Unterkalmsteiner, M., Abrahamsson, P., Wang, X., *et al.*: 'Software startups – a research agenda', *e-Informatica Softw. Eng. J.*, 2016, **10**, (1), pp. 89–123
[18] Eloranta, V.P.: 'Towards a pattern language for software start-ups'. 19th European Conf. on Pattern Languages of Programs, Isee, Germany, July 2104, pp. 1–11
[19] Awad, M.A.: '*A comparison between agile and traditional software development methodologies*' (University of Western Australia, Perth, WA, Australia, 2005)
[20] Sánchez-Gordón, M.L., O'Connor, R.V.: 'Understanding the gap between software process practices and actual practice in very small companies', *Softw. Qual. J.*, 2016, **24**, (3), pp. 549–570
[21] Chapman, J.R.: 'Software development methodology', 2004. Available at http://www.hyperthot.com/pm_sdm.html, retrieved 15 November 2018
[22] Abrahamsson, P., Salo, O., Ronkainen, J., *et al.*: 'Agile software development methods: review and analysis'. arXiv preprint arXiv:1709.08439, 2017, pp. 1–32
[23] Clarke, P., O'Connor, R.V.: 'The situational factors that affect the software development process: towards a comprehensive reference framework', *Inf. Softw. Technol.*, 2012, **54**, (5), pp. 433–447
[24] Kitchenham, B., Charters, B.: 'Systematic literature reviews in software engineering'. Tech. Rep. EBSE, Keele University and Durham University, Joint Report, Staffordshire, UK, 2007
[25] Dybå, T., Dingsøyr, T.: 'Empirical studies of agile software development: a systematic review', *Inf. Softw. Technol.*, 2008, **50**, (9-10), pp. 833–859
[26] Runeson, P., Höst, M.: 'Guidelines for conducting and reporting case study research in software engineering', *Empir. Softw. Eng.*, 2009, **14**, (2), pp. 131–164
[27] Marshall, M.N.: 'The key informant technique', *Fam. Pract.*, 1996, **13**, (1), pp. 92–97
[28] Cruzes, D.S., Dybå, T., Runeson, P., *et al.*: 'Case studies synthesis: a thematic, cross-case, and narrative synthesis worked example', *Empir. Softw. Eng.*, 2015, **20**, (6), pp. 1634–1665
[29] Cruzes, D.S., Dybå, T.: 'Recommended steps for thematic synthesis in software engineering'. Int. Symp. on Empirical Software Engineering and Measurement, 2011, pp. 275–284
[30] Wohlin, C., Runeson, P., Höst, M., *et al.*: '*Experimentation in software engineering*' (Springer Science & Business Media, Berlin & Heidelberg, Germany, 2012)
[31] Petersen, K., Vakkalanka, S., Kuzniarz, L.: 'Guidelines for conducting systematic mapping studies in software engineering: an update', *Inf. Softw. Technol.*, 2015, **64**, pp. 1–18
[32] Hui, A.: 'Lean change: enabling agile transformation through lean startup, kotter and kanban: an experience report'. Agile Conf., Nashville, TN, USA, August 2013, pp. 169–174
[33] Taipale, M.: 'Huitale – a story of a Finnish lean startup', in '*Lean enterprise software and systems*' (Springer, Berlin, Heidelberg, 2010), pp. 111–114
[34] Nguyen-Duc, A., Seppänen, P., Abrahamsson, P.: 'Hunter-gatherer cycle: a conceptual model of the evolution of software startups'. Int. Conf. on Software and System Process, Tallinn, Estonia, August 2015, pp. 199–203
[35] Nguyen-Duc, A., Shah, S.M.A., Ambrahamsson, P.: 'Towards an early stage software startups evolution model'. 42th Euromicro Conf. Software Engineering and Advanced Applications, Limassol, Cyprus, 31 August - 1 September 2016, pp. 120–127
[36] Batova, T., Clark, D., Card, D.: 'Challenges of lean customer discovery as invention'. Int. Professional Communication Conf., 2016, pp. 1–5
[37] Björk, J., Ljungblad, J., Bosch, J.: 'Lean product development in early-stage startups'. IW-LCSP' 2013, Potsdam, Germany, June 2013, pp. 19–32
[38] Clark, T., Muller, P.A.: 'Exploiting model driven technology: a tale of two startups', *Softw. Syst. Model.*, 2012, **11**, (4), pp. 481–493
[39] Pantiuchina, J., Mondini, M., Khanna, D., *et al.*: 'Are software startups applying agile practices? The state of the practice from a large survey'. Int. Conf. on Agile Software Development, Cologne, Germany, May 2017, pp. 167–183
[40] Kajko-Mattsson, M., Nikitina, N.: 'From knowing nothing to knowing a little: experiences gained from process improvement in a start-up company'. Int. Conf. on Computer Science and Software Engineering, Wuhan, China, December 2008, pp. 617–621
[41] Mc Caffery, F., Taylor, P.S., Coleman, G.: 'Adept: a unified assessment method for small software companies', *IEEE Softw.*, 2007, **24**, (1), pp. 24–31
[42] Wangenheim, C.G.V., Weber, S., Hauck, J.C.R., *et al.*: 'Experiences on establishing software processes in small companies', *Inf. Softw. Technol.*, 2006, **48**, (9), pp. 890–900
[43] Wu, H.Y., Callaghan, V.: 'From imagination to innovation: a creative development process', in Novais, P., Konomi, S. (Eds.): '*Intelligent Environments 2016*' (IOS Press, Amsterdam, Netherlands, 2016)
[44] Souza, R., Malta, K., De Almeida, E.S.: 'Software engineering in startups: a single embedded case study'. 1st Int. Workshop on Software Engineering for Startups, Buenos Aires, Argentina, May 2017, pp. 17–23
[45] Bajwa, S.S., Wang, X., Duc, A.N., *et al.*: ''Failures' to be celebrated: an analysis of major pivots of software startups', *Empir. Softw. Eng.*, 2017, **22**, (5), pp. 2373–2408
[46] Seppänen, P., Tripathi, N., Oivo, M., *et al.*: 'How are product ideas validated?'. Int. Conf. on Software Business, Essen, Germany, June 2017, pp. 3–17
[47] May, B.: 'Applying lean startup: an experience report-lean & lean UX by a UX veteran: lessons learned in creating & launching a complex consumer app'. Agile Conf., Dallas, TX, USA, August 2012, pp. 141–147
[48] Berrocal, J., Garcia-Alonso, J., Murillo, J.M.: 'The fall and rise of NimBees'. 42th Euromicro Conf. Software Engineering and Advanced Applications, Limassol, Cyprus, 31 August - 2 September 2016
[49] Giardino, C., Paternoster, N., Unterkalmsteiner, M., *et al.*: 'Software development in startup companies: the greenfield startup model', *IEEE TSE*, 2016, **42**, (6), pp. 585–604
[50] Duc, A.N., Abrahamsson, P.: 'Minimum viable product or multiple facet product? The role of MVP in software startups'. Int. Conf. on Agile Software Development, Edinburgh, UK, May 2016, pp. 118–130
[51] Hokkanen, L., Leppänen, M.: 'Three patterns for user involvement in startups'. European Conf. on Pattern Languages of Programs, 2015, p. 51
[52] Lenarduzzi, V., Taibi, D.: 'Mvp explained: a systematic mapping study on the definitions of minimal viable product'. 42th Euromicro Conf. on Software Engineering and Advanced Applications, Limassol, Cyprus, 31 August - 2 September 2016, pp. 112–119
[53] Leppänen, M.: 'Patterns for starting up a software startup company'. European Conf. on Pattern Languages of Programs Irsee, Germany, July 2014, pp. 1–7
[54] Marion, T., Dunlap, D., Friar, J.: 'Instilling the entrepreneurial spirit in your R&D team: what large firms can learn from successful startups?', *IEEE Trans. Eng. Manag.*, 2012, **59**, (2), pp. 323–337
[55] Pompermaier, L., Chanin, R., Sales, A., *et al.*: 'An empirical study on software engineering and software startups: findings from cases in an innovation ecosystem'. The 29th International Conference on Software Engineering and Knowledge Engineering, Pittsburgh, PA, USA, July 2017
[56] Taylor, P.S., Greer, D., Coleman, G., *et al.*: 'Preparing small software companies for tailored agile method adoption: minimally intrusive risk assessment', *Softw. Process, Improv. Pract.*, 2008, **13**, (5), pp. 421–437
[57] Terho, H., Suonsyrjä, S., Jaaksi, A., *et al.*: 'Lean startup meets software product lines: survival of the fittest or letting products bloom?'. SPLST, Tampere, Finland, October 2015, pp. 134–148
[58] Tingling, P., Saeed, A.: 'Extreme programming in action: a longitudinal case study'. Conf. on Human-Computer Interaction, Beijing, China, July 2007, pp. 242–251
[59] Al-Sakkaf, A.M., Hashim, N.L., Omar, M.: 'Using hierarchical cluster analysis to generate clusters of agile practices', *J. Telecommun. Electron. Comput. Eng. (JTEC)*, 2017, **9**, (1–2), pp. 53–56
[60] Nguyen-Duc, A., Wang, X., Abrahamsson, P.: 'What influences the speed of prototyping? An empirical investigation of twenty software startups'. Agile Software Development Conf., Cologne, Germany, May 2017, pp. 20–36
[61] Gutbrod, M., Münch, J., Tichy, M.: 'How do software startups approach experimentation? Empirical results from a qualitative interview study'. Int. Conf. on Product-Focused Software Process Improvement, Innsbruck, Austria, November 2017, pp. 297–304
[62] Chorev, S., Anderson, A.R.: 'Success in Israeli high-tech start-ups; critical factors and process', *Technovation*, 2006, **26**, (2), pp. 162–174
[63] Bicen, P., Johnson, W.H.A.: 'How do firms innovate with limited resources in turbulent markets?', *Innov., Manag. Policy Pract.*, 2014, **16**, (3), pp. 430–444
[64] Yau, A., Murphy, C.: 'Is a rigorous agile methodology the best development strategy for small scale tech startups?', 2013, available at https://repository.upenn.edu/cis_reports/980/