An Empirical Investigation on Software Practices in Growth Phase Startups

Cico, Orges[1]; Duc, Anh Nguyen[2]; Jaccheri, Letizia[3]
[1]Institutt for datateknologi og informatikk - Norges teknisk-naturvitenskapelige universitet
[2]Institutt for økonomi og IT - Universitetet i Sørøst-Norge
[3]Institutt for datateknologi og informatikk - Norges teknisk-naturvitenskapelige universitet

Cico, O., Duc, A. N., & Jaccheri, L. (2020). An Empirical Investigation on Software Practices in Growth Phase Startups. I J. Li, L. Jaccheri, T. Dingsøyr & R. Chitchyan (Red.), *Proceedings of the Evaluation and Assessment in Software Engineering* (s. 282-287). https://doi.org/10.1145/3383219.3383249

Accepted version of article in
EASE '20: Proceedings of the Evaluation and Assessment in Software Engineering

Publisher's version: DOI: https://doi.org/10.1145/3383219.3383249

An Empirical Investigation on Software Practices in Growth Phase Startups

Orges Cico
Norwegian University of Science and Technology
Trondheim, Norway
orges.cico@ntnu.no

Anh Nguyen Duc
University of Southeast Norway
Bø i Telemark, Norway
anh.nguyen.duc@usn.no

Letizia Jaccheri
Norwegian University of Science and Technology
Trondheim, Norway
letizia.jaccheri@ntnu.no

ABSTRACT

Context: Software startups are software-intensive early-stage companies with high growth rates. We notice little evidence in the literature concerning engineering practices when startups transition to the growth phase. Aim: Our goal is to evaluate how software startups embrace software engineering practices. Methodology: We conduct a survey guided by semi-structured interviews as an initial step, to be followed by field questionnaires as part of a future exploratory study. We use open coding to identify patterns leading to themes we use to state our hypotheses. To identify our samples, we use purposive sampling. Results: Specifically, we analyze seven startup cases during the first qualitative phase. We obtain five anti-patterns (no-documentation, no-agile, no-code intellectual property protection, cowboy programming, no-automated testing) and corresponding patterns (readable code, ad-hoc project management, private code repositories, paired and individual programming, ad-hoc testing) in adopting software engineering practices. We state 10 corresponding hypotheses we intend to corroborate by surveying a more significant number of software startups. Contribution: This study, throughout its recommendations, provides an initial road map for software startups in the growth phase, allowing future researchers and practitioners to make educated recommendations.

CCS CONCEPTS

• **Software and its engineering → Agile software development**; **Software verification and validation**; **Collaboration in software development**; **Documentation**.

KEYWORDS

Software Startups, Software Engineering Practices, Empirical Investigation, Growth Phase

ACM Reference Format:
Orges Cico, Anh Nguyen Duc, and Letizia Jaccheri. 2020. An Empirical Investigation on Software Practices in Growth Phase Startups. In *Evaluation and Assessment in Software Engineering (EASE 2020), April 15–17, 2020, Trondheim, Norway*. ACM, Trondheim, Norway, 6 pages. https://doi.org/10.1145/3383219.3383249

EASE 2020, April 15–17, 2020, Trondheim, Norway

ACM ISBN 978-1-4503-7731-7/20/04…$15.00
https://doi.org/10.1145/3383219.3383249

1 INTRODUCTION

A startup is commonly defined as newly established companies with small teams, limited resource and aim for rapid scaling business models [3, 8]. At the early stage, the primary goal is to meet a marketplace need by developing a viable business model for products, services, processes, or platforms. The failure rate of startups is commonly high; however, successful startups have had a major impact on the industry [18]. This is particularly true for startups developing software-intensive products, which have shown higher rates of scaling [6], making them stand out.

Adopting software engineering (SE) practices, is becoming even more of an urgent need for many software startups [9, 10, 14]. Empirical evidence on how SE is adopted by software startups is still meager, and the need for empirical evidence is reported from [2, 15]. Compared to previous efforts studying SE practices at different startup phases, the understanding of SE practices at growth phase is very limited. We aim at understanding how software startups already or transitioning in growth phase can adopt SE approaches. As the first step, we formulated the following research question (**RQ**):

RQ: *How do software startups embrace software engineering practices when in growth phase?*

To address the **RQ**, we designed an exploratory study during the first phase of which we conducted semi-structured interviews, with seven Chief Executive Officers (CEOs) and Chief Technical Officers (CTOs) from seven software startups, selected using purposive sampling. We focused on those startups that are almost or have already made a successful transition towards growth phase [1]. The interviews helped us state relevant hypotheses about startups in growth phase to be corroborated by means of field questionnaires.

The rest of this paper is organized as follows: In Section 2, we describe the background. Our research methodology is described in Section 3. The results are presented in Section 4 and discussed in Section 5. Finally, we present in Section 6 conclusions and future work perspectives.

2 BACKGROUND

The failure rate of startups is commonly high; however, successful startups have had a major impact on the industry [18]. Typically, startups operate and evolve in an ecosystem with connections to

[1] Growth phase Startups are well established companies with market revenue being primary source of income

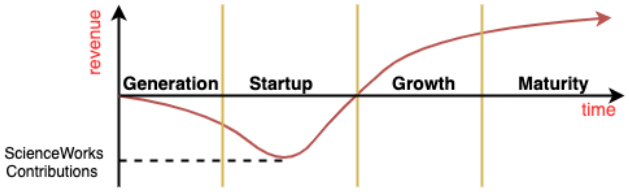

Figure 1: Startup lifecycle phases [12].

various stakeholders, from types of investors to incubators, accelerators and third-party vendors. Startups typically undergo several development phases: Ideation (product or service idea), Concepting (mission and vision), Commitment (team with the initial product), Validation (iteration and testing the initial idea), Scaling (focus on key performance indicators), and Establishment (increasing growth and market potential) [7].

Based on Muzellec et al. [12], the transitions of startups from one stage to another can be characterized under different categories, Figure 1. Finance is one of the most important factors for startup survival. In the early stages, funding is commonly based on self-contributions, in the form of self-investment (by bootstrapping between jobs) or loans (from relatives or friends).

Other funding options in the early stage of startup formation can come from pre-seed or crowdfunding. In later stages, when a Minimum Viable Product (MVP) has been developed and iteration with the market is a must (do-or-die approach), the need for larger funding amounts from venture capitalists (VCs) and angel investors (AIs) becomes obvious. Finally, if the startup has developed a fully operational product or service, then the market, either local or global, decides the startup's growth potential.

The transition of startups is also marked by the methodological evolution from ad-hoc or customized development practices [13] to more principled approaches. A lean startup is a popular methodology among startups at the early and validation stages [17]. It focuses on shortening the product development cycle through iterative product releases, market experimentation, and validation. Meeting the needs of early customers should reduce later risks of large investments' failures. Teams that adopt a lean startup strategy develop a continuously changing MVP [16]. The MVP, comprising the technological Proof of Concept (PoC), helps identify product/service potentials with the startup's limited resources. Nevertheless, if software startups desire to transition towards the growth phase, they must establish themselves in the market by creating valuable products. The growth phase inherits many technological features, benefits, and drawbacks from its successful MVP predecessor, now becoming a professionalized product/service fulfilling a specific market need.

3 RESEARCH METHOD

We aim to understand the perception of software engineering practices in growth-phase software startups. Hence, the research question, which guided our investigation is: *"How do software Startups embrace software engineering practices when in growth phase?"*

To gather and interpret evidence to answer our research question, we devised a mixed-methods approach. First, we conducted semi-structured interviews with seven software startups' CEOs and CTOs. This allowed us to state hypotheses that can be corroborated via surveying a larger number of startups, leading to an exploratory study design. This work presents only the first part of our investigation.

3.1 Case selection

We collected data first from a tech event in USA with the active participation of more than 100 startups (about 75% of the samples). The remaining 25% of the data were collected from Norway. We selected the sample population using the purposive sampling technique. Purposive sampling, is a form of non-probability sampling in which researchers rely on their own judgment when choosing members of the population to participate in their study [20].

3.2 Case Demographics

We present in this section a brief context of the startup companies that participated in our study. We first collected data from the startups' online resources after initial contact (email or face-to-face acquaintance) and then later from CEOs and CTOs. From more than 100 startups during the tech event in USA we identified 12 software startups as potential candidates. However, only five answered our request positively for an interview. We then looked into startups from Norway where the three startups we enquired for an interview only two agreed to participate in our study interview. Demographics of the seven software startups are reported in Table 1. Notably, all the interviewees are co-founders of the startups.

Furthermore, we have summarized each startup context and their present life-cycle phase in Table 2. We established their growth phase based on their (1) customer base and (2) stability in the market during the last year (cf. Section 2).

3.3 Interview design

We performed a pilot study in constructing our interview template, which was used for later data collection in all cases. This allowed us to focus our interview questions in connection to the research question. The interview process took place in two parts. The first part of the survey primarily addressed demographic information about the startup. The second part focused more on a broad context on the software and technological aspects of the startup. This way we detailed the perception of software practices regarding each sample. Dividing the interview into various parts helped us guide the startups in expressing their standpoints, without being biased by our expectations. The approach followed does not compromise the data gathered since, from the start, we planned a semi-structured interview, having little control over the chosen samples. Although all authors collaboratively planned interviews, the execution was performed by only one of them and was, afterward, peer-reviewed again, as further discussed in Section 3.4.

3.4 Data Collection

To answer our research question, based on recommendations from Runeson [19], we collected data from semi-structured, face-to-face interviews. We interviewed eight CTOs/CEOs from seven startups in two countries, all having a high-tech product focus. Each interview lasted 45 to 60 minutes, with 75% of the time dedicated

Table 1: Software Startup sample demographics.

Startup Case Number	Role	Country	Product / Service	Establishment Year	Product Commercialization	Employee Number	Gender Balance	Employee Average Age Range
Startup 1	CTO / CEO	USA	High tech software products	2008	2017	8	50% F / 50% M	20- 30 and 40+
Startup 2	CEO	USA	High tech software intensive and hardware system product	2016	2017	4	100% M	30th
Startup 3	CEO	USA	Software Development	2001	2012	65	40% F / 60% M	30th
Startup 4	CEO	USA	High tech software intensive	2015	2016	7	10% F / 90% M	30th
Startup 5	CTO	USA	High tech software intensive and hardware system product	2012	2017	34	20% F / 80 %M	30th
Startup 6	CTO	Norway	High tech software products	2017	2018	6	15% F / 85% M	30th
Startup 7	CEO	Norway	High tech software products	2012	2012	5	25% F / 75% M	20th

Table 2: Startups' contexts.

Startup Case	High Tech Solution	Project Management Practices	Phase	Number of Customers
Startup 1	Genomic search engine	Semi-Agile practices	Growth Phase	100+
Startup 2	Solves mechanist skilled workers shortage by automating physical component design industry.	No particular practices.	Transitioning to Growth Phase. MVP limitations.	5+
Startup 3	Software development consultancy for high tech products, throughout pair programming practices.	In-house paired programming practices.	Growth Phase	10+
Startup 4	GPU-accelerated software for rapid secondary analysis of next-generation sequencing data.	No particular practices.	Transitioning to Growth Phase	5+
Startup 5	Wind turbine inspection through drone technology.	Semi-Agile practices.	Growth Phase	5+
Startup 6	Optimal wind farm layout services.	Semi-Agile practices.	Growth Phase	20+
Startup 7	Real estate business intelligence.	No particular practices.	Growth Phase	50+

to questions related to startup software practices. All interviews were recorded for later transcription. To facilitate the latter process, we utilized online tools (sonix.ai), delivering an approximate accuracy of around 95% in English transcriptions. We read the transcription accuracy score from reports of the online tool, after each transcription. After some manual update of the text was performed, we shared the transcribed interview text with the startups to acknowledge the obtained results.

The interviews aimed to understand the perception of software engineering practices from startup founders, commonly represented by both CEOs and CTOs, as reported in Table 1. Both these entities represent primary stakeholders with a higher interest in a smooth transition from MVPs to product/services while transitioning in the growth phase.

As explained in Section 3.3, we divided our interview into 1) demographic questions (lasting 10–15 min.) and 2) focused software engineering practice questions (lasting 30–45 min.). Table 3 reports the specific questions for each part.

3.5 Data Analysis

After carefully collecting data to obtain significant evidence that would help us answer our research question, we then used the thematic analysis approach [5], which consisted of identifying recurring patterns and themes within the interview data. The steps to conduct the systematic analysis consisted of the following:

(1) **Reading the transcripts.** This step initially involved quick browsing and correction of the automatically transcribed data from the audio recordings. We made quick notes about first impressions. Later on, we reviewed more carefully the transcribed data by reading judiciously, line by line.
(2) **Coding.** During this step, we focused on choosing and labeling relevant words, phrases or sentences, and even larger text fragments or sections. The labels revealed more about startup SE practices. We primarily looked for repetitive and unexpected answers. We attempted to code as much as possible regarding the software practices. To mitigate bias, the two authors worked separately during this coding process.
(3) **Creating themes.** After gathering all the codes, we decided on the most relevant ones and created different categories, which are also defined as themes. Many initial codes from the previous step were either dropped or merged to create new ones.
(4) **Labeling and connecting themes.** During this step, we decided on which themes were more relevant and defined

Table 3: Interview parts and questions.

Interview Part	Questions
Part 1 Startup Demographics	What is the startup core product/service? When was your startup established? Where is your startup located? What is your role? What type of ecosystem are you presently working in? How many employees do you have at the moment? What is the gender balance in your startup? What is the average employee age? What is your team composition?
Part 2 Software Engineering Practices	What software development practices, tools are you using? Briefly describe how? What are the most important quality attributes (UX, performance, security, reusability) for your current products? What testing practices do you adopt in validating and verifying the quality of your product/service? How much have you invested on testing activities? How do you document your product at different phases of development and testing? How are your documents updated? How do you keep track of changes? How do you protect your technology? How do you assess the artifacts worth protecting?

appropriate names for each of them. Furthermore, we also tried to identify relationships among the themes.

(5) **Drawing/writing the results summary.** After deciding on the theme's importance and hierarchy, we generate a summary of the results. by aggregating the thematic analysis of semi-structured interviews with all the Startups.

To fulfill the five steps, we used the thematic coding tool NVivo 12 [1].

4 RESULTS

During our analysis, we identified five anti-patterns and, correspondingly, five patterns that reflected startups' software practices. Thus, we created 10 major categories/themes and put them in a one-to-one comparison. Last, we stated 10 hypotheses (cf. Section 5.2), useful for further investigation.

4.1 No-Documentation versus Readable Code

In many cases, documentation is deemed cumbersome and useless to the software development process. Developers are reluctant to extensively document their code unless strictly necessary.

One of the interview reports reads:

"...Things are changing so rapidly. Documentation becomes obsolete quickly..." **[Startup 1]**

Yet another interview states the following about the extensive documentation's useless effort:

"I think that a lot of times teams spend a tremendous amount of energy. creating very explicit artifacts with often long documents. And our belief is that the challenge of many documents that exist in software engineering realms is that no one reads them, right? Not even the people who wrote them..." **[Startup 1]**

In software engineering, documentation is regarded as a crucial aspect of the development process, making this attitude an anti-pattern.

Readable code is deemed a more relevant approach, less time consuming and a common practice. Interviews report the following:

"...We expect the code to be highly readable and peer reviewed ... Only critical or complex code is thoroughly documented..." **[Startup 3]**

4.2 No-agile versus Ad-hoc software development paradigms

We found that strictly following agile software development practices is not perceived as being critical or even not the most viable option.

In some cases, the adoption of agile practices is considered hard to implement if software must be integrated with hardware systems. The adoption of agile practices is considered possible only to some extent:

"...So when we are talking about software-hardware projects that can sometimes span six months and agile is a little hard to do in that regard because you need to be thinking six months ahead ... So we stick to like almost a semi Agile... " **[Startup 5]**

Yet another startup makes similar observations:

"...I wouldn't say we are 100 percent you know agile process oriented or I would also say anyone who is 100 percent agile probably missed the point of agile..." **[Startup 1]**

Hence, ad-hoc approaches to project management are preferable to pure agile, eventually demystifying the literature stating that agile is part of the most common practices in software startups. One case reports:

"... Yeah I think the Agile community is very familiar with us and I think we have a lot of respect in general from people in both the lean community and the Agile community. We use very strong visual management system. We do paper based planning and paper based work authorization ..." **[Startup 3]**

Notably, however, the lack of good software practices may lead to risks of project failures in a short time. Very pure ad-hoc practices are noted from cases regarding startups 2, 4, 6, and 7.

4.3 No-code intellectual property protection versus Private code

Many startups agree the intellectual property (IP) protection of code is not deemed possible or even worth pursuing. The main reasons are high legal costs that cannot be covered even from startups at a higher growth rate. Although not strictly an anti-pattern, the lack of code legal protection might be a bad decision from a long-term perspective. We read from some of the interviews' reports:

"...IPs would have meaning if big companies want to invest and protect the software, but we don't have the capacity to do so..." **[Startup 5]**

"...But if you have a very interesting solution or algorithm or something, you have to. I think it is a good idea to protect also from the property rights level ... but I don't think it applies to our case..." **[Startup 6]**

The code is commonly protected via private repositories. In many cases, the secrecy option takes over. We read from the interviews:

"...Our main protection is to keep the source code in private repositories. This has originally been done since the beginning..." **[Startup 5]**

"...So anyone has access to their own product repository. So that's how the basically manage who has access to watch the repository. But in terms of delivery to the customer, everything is executable and encrypted and inside the docket. So everything is containerized. So you don't easily have access to the binary and even the binary is encrypted..." **[Startup 4]**

4.4 Cowboy Programming versus Pair Programming

Cowboy programming is the most-encountered approach. We notice that programming decisions are left to developers, in many cases, the CTO/CEO or additional developer on the team. We read from the interview as follows:

"...Programming was mainly done by myself and our CEO. You don't know that much about the software, so when you looked back at how the tools looked for a year and a half ago, looks terrible. Yeah. But you have learned to become better ourselves..." **[Startup 6]**

Yet another interview observation is:

*"...We use more a cowboy approach of just building it **[The code]** and getting it out there..."* **[Startup 1]**

In many cases, pair-programming is common and helps with software development decision making. The approach helps with better programming practices and the avoidance of coding in the wild. One interview reports the following coding:

"...I think that pairing really helps get to the discipline because there might be a day you come and say oh let's cut a corner let's just move ahead let's not write that test might be too hard but might your poor partner might say you know let's make sure we do it the right way..." **[Startup 3]**

4.5 No-automated Testing versus Ad-hoc Testing

Testing is one of the most challenging software engineering activities that startups face in the growth phase. Automated testing was considered difficult in most cases we investigated; many reports indicate ad-hoc testing from the developers or simply acceptance testing from end users, as follows:

"...I really, really want to do the testing and integration, testing and functional testing, all that stuff. But it takes time...So, okay, going for a quickly clicking through that and making sure that everything works..." **[Startup 6]**

Only two cases have a more structured approach in testing their software systems. Nonetheless, automated unit testing is limited, with at most 60% in one of the cases. The CTO observes as follows:

"... So we do unit testing, integrated testing and kind of sandbox testing. Yeah. Massive entire system testing. Stress testing all that kind of stuff ... we don't do actually a heavy amount of unit testing ... I say we probably have 60/70% coverage ..." **[Startup 5]**

Another case, with a more focused testing approach in development, reports the following:

"...We are doing test driven development so when we are adding new features to the software we are working on writing automated unit tests before they write the code then we fit the code in..." **[Startup 3]**

5 DISCUSSIONS

Findings indicate that many startups in transitioning or already in their growth phase have reached some level of maturity, but they do not fully adopt software engineering practices as recommended by the SWEBOK [4]. During the interviews we noticed that most of the startup co-founders had engineering background and were able to answer our specific questions (cf. Table 3 - Interview part 2).

The expertise of startup founders also determines their adherence to SE practices. Many of the previous efforts [2, 9, 10, 14, 15] have focused on early phase startups, providing little to no evidence related to growth phase software practices. The reasons argued by previous authors were based on the need to help early phase startups avoid failure from bad software engineering practices. However, for a long time this has drawn the attention from gathering empirical evidence about practices adopted at later phases. Growth phase startups and especially software startups, which have been highly lucrative in the past decade, have had major impact on the economy. Thus, supporting startup growth with tailored versions of software engineering best practices is becoming an emergent need. Unfortunately, research and empirical evidence is still at its infant stage.

5.1 Threats to Validity

Based on recommendations from Maxwell [11], we report the following validity threats for our study:

(1) **Descriptive validity**: Although we have tried to gather as much information as possible, we admit that some aspects might not have been able to be recorded. To mitigate this threat to validity, we have used audio to verify the descriptive data back in time and stored the rest of the data electronically. We also confirmed our transcription results with the interviewed samples, to make sure we had correctly interpreted their statements.
(2) **Interpretation validity**: We have carefully kept track of the written perspective of the individuals being researched. This way, we are sure their unique perspective is taken into account, instead of imposing meaning from our point of view. Open-ended questions have been used to allow the participant to elaborate on answers.
(3) **Researcher bias**: We were careful not to put any bias on gender, culture, or professional background. We select cases regardless of their software engineering practices and the main reason for some startups not participating in this study is that they did not have enough time for us. However, we also acknowledge the bias in case selection, as it would be in any multiple case study. Specifically in our case related to interviewing startups in presumably growth phase

(4) **External validity**: Due to the small sample the possibility of generalizing results still remains a threat. However, we have limited the study to hypotheses statement which can be validated by gathering larger data in the future.

5.2 Hypotheses

Conducting first interviews on a small sample in two distinct countries helped us reduce the bias of the obtained results, although fully eliminating it is not possible. Based on these results, we can draw 10 hypotheses, thus completing the first half of our investigation. We intend to corroborate our hypotheses by (1) conducting questionnaires with a larger sample of growth-phase software startups, including the ones that participated in the interview process, and (2) performing triangulation with artifacts analysis of our present findings.

Hypotheses:
H1: *Software startups in the growth phase tend to not document their code, unless strictly necessary.*
H2: *: Software startups tend to write clean, readable code that can be easily interpreted.*
(**cf. Section 4.1**)
H3: *The use of agile practices tend to improve software startups' productivity in the growth phase.*
H4: *The use of ad-hoc project management practices is common in growth-phase software startups.*
(**cf. Section 4.2**)
H5: *Software startups in the growth phase tend to protect their code by storing it in secure private repositories.*
H6: *Software startups do not protect their code through patents, licenses and any other legal means related to IP.*
(**cf. Section 4.3**)
H7: *Software startups in the growth phase adopt programming approaches in the wild contributing to code smells.*
H8: *Pair programming mitigates programming decisions made in the wild and contributes to more readable code development.*
(**cf. Section 4.4**)
H9: *Software startup rely more on automated unit and integrated testing before releasing their product in growth phases than in early phases.*
H10: *Software startups in the growth phase rely only on customer validation and acceptance testing for their product releases.*
(**cf. Section 4.5**)

6 CONCLUSIONS AND FUTURE WORK

We initiated an exploratory study in evaluating software startups in growth-phase software engineering practices. The interviews served in collecting qualitative data that we used in stating our hypotheses. The sample size from the interviews is relatively small, so a larger sample size is required to corroborate our hypotheses. From interview reports, we find that agile practices are seldom adopted. Readable code is regarded as far more important than extensive code documentation. IP protection is deemed impossible, and the best option is keeping the code secret. Programming best practices and testing are the least-explored software engineering approaches. Based on these early findings, we conclude that software startups during the growth phase might still struggle to reach maturity in adopting software engineering best practices. There seems to be a common agreement that software secrecy should be adopted with great care. Indeed, the need for agile and software documentation is not fully appreciated.

However, to fully corroborate our findings, based on the exploratory study approach, we need to complete as part of future work the second phase of our investigation by collecting a larger sample of both qualitative and quantitative data. We also encourage researchers to corroborate the stated hypotheses by performing triangulation with artifacts analysis of our present findings.

7 ACKNOWLEDGEMENT

This work was funded by the Norwegian Research Council under the project IPIT Project Number: 274816.

REFERENCES

[1] Last accessed 16 Aug 2019. Nvivo Homepage. https://www.qsrinternational.com/nvivo/home.
[2] P Abrahamsson, A Nguyen-duc, GH Baltes, K Conboy, D Dennehy, R Sweetman, H Edison, S Shahid, X Wang, J Garbajosa, et al. 2016. Software Startups - A Research Agenda. *e-Informatica Softw. Eng. J* 10, 1 (2016), 1–28.
[3] Vebjørn Berg, Jørgen Birkeland, Anh Nguyen-Duc, Ilias Pappas, and Letizia Jaccheri. 2018. Software Startup Engineering: A Systematic Mapping Study. *Journal of Systems and Software* (2018).
[4] Pierre Bourque, Richard E Fairley, et al. 2014. *Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0.* IEEE Computer Society Press.
[5] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative research in psychology* 3, 2 (2006), 77–101.
[6] Mark V Cannice. 2019. Confidence among Silicon Valley Venture Capitalists Q3 2017–Q4 2018: Trends, Insights, and Tells. *The Journal of Private Equity* 22, 3 (2019), 18–24.
[7] Mark Crowne. 2002. Why software product startups fail and what to do about it. Evolution of software product development in startup companies. In *IEEE International Engineering Management Conference*, Vol. 1. IEEE, 338–343.
[8] Carmine Giardino, Xiaofeng Wang, and Pekka Abrahamsson. 2014. Why early-stage software startups fail: a behavioral framework. In *International Conference of Software Business*. Springer, 27–41.
[9] Eriks Klotins, Michael Unterkalmsteiner, Panagiota Chatzipetrou, Tony Gorschek, Rafael Prikladnicki, Nirnaya Tripathi, and Leandro Pompermaier. 2019. A progression model of software engineering goals, challenges, and practices in start-ups. *IEEE Transactions on Software Engineering* (2019).
[10] Eriks Klotins, Michael Unterkalmsteiner, and Tony Gorschek. 2019. Software engineering in start-up companies: An analysis of 88 experience reports. *Empirical Software Engineering* 24, 1 (2019), 68–102.
[11] Joseph A Maxwell. 2012. *Qualitative research design: An interactive approach*. Vol. 41. Sage publications.
[12] Laurent Muzellec, Sébastien Ronteau, and Mary Lambkin. 2015. Two-sided Internet platforms: A business model lifecycle perspective. *Industrial Marketing Management* 45 (2015), 139–150.
[13] Anh Nguyen-Duc, Xiaofeng Wang, and Pekka Abrahamsson. 2017. What influences the speed of prototyping? an empirical investigation of twenty software startups. In *International Conference on Agile Software Development*. Springer, Cham, 20–36.
[14] Jevgenija Pantiuchina, Marco Mondini, Dron Khanna, Xiaofeng Wang, and Pekka Abrahamsson. 2017. Are software startups applying agile practices? The state of the practice from a large survey. In *International Conference on Agile Software Development*. Springer, Cham, 167–183.
[15] Nicolò Paternoster, Carmine Giardino, Michael Unterkalmsteiner, Tony Gorschek, and Pekka Abrahamsson. 2014. Software development in startup companies: A systematic mapping study. *Information and Software Technology* 56, 10 (2014), 1200–1218.
[16] AL Penenberg. 2011. Eric Lies is a lean startup machine.
[17] Eric Ries. 2011. *The lean startup: how today's entrepreneurs use continuous innovation to create radically successful businesses*.
[18] N Robehmed. 2013. What is a startup? Forbes.
[19] Per Runeson and Martin Höst. 2009. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering* 14, 2 (2009), 131.
[20] Harsh Suri et al. 2011. Purposeful sampling in qualitative research synthesis. *Qualitative research journal* 11, 2 (2011), 63.