Progression Models in Software Engineering

Yati Ashokkumar Tank
Computer Science and Engineering
Nirma University
Ahmedabad, India
20bce323@nirmauni.ac.in

Mufid Vohra
Computer Science and Engineering
Nirma University
Ahmedabad, India
20bce307@nirmauni.ac.in

Abstract—**In order to find patterns and trends that define technical work in start-ups, we want to gather information on engineering goals, challenges, and approaches in start-ups. Academics may more clearly comprehend how practises connect to objectives and challenges with the use of this kind of data. Future research aimed at creating and resolving those objectives and challenges can then be informed by this information. Additionally, such advancements can assist users in choosing their usage with greater knowledge.**

Keywords—software start-up, software engineering practices, progression mod

I. Introduction

Smaller businesses established for the purpose of producing and marketing software startups.Startups are notorious for their fast development, tiny workforces, investigations into client requests and situations, and a high rate of failure.. However, by utilising cutting-edge technology, threat, and speed, entrepreneurs can still produce software products pretty quickly.

As a corporation develops, the points, objects, and struggles of product engineering carelessly change. Modern engineering approaches provide very little assistance in recognising the changing environment and selecting appropriate practises. A mistake in selecting engineering approaches could result in over- and under of the solution, resulting in wasted request openings.

A lot of other studies look towards engineering techniques in a startup environment, such as conditions engineering, specialised debt, and stoner experience. Numerous research projects, including those by Giardino & Crowne, aim to streamline product engineering methods at start-ups. But none of these studies provides a comprehensive and in-depth examination of the engineering practises that start-ups employ throughout the spectrum of engineering process domains and startup stages. Both Klotins and colleagues and Unterkalmsteiner and colleagues have emphasised the need of understanding the major aim was to integrate at start-ups and providing appropriate assistance to interpreters. With this research, we hope to learn more about how various planning processes are activated and how the techniques employed change during a start-lifespan. up's. Through analysis, we present a model of progress to which design considerations, ie. claims, practices and challenges, can be applied.The model is intended to assist interpreters in making decisions and to identify particular engineering difficulties for further discussion.

II. BACKGROUND

A. *Software Start-ups*

Carmel reported on tiny enterprises developing and marketing breakthrough software products as early as 1994. These startups and small companies utilise informal design methods, have small, motivated staff, and prioritise application speed above product quality. Recent exemplary research by Giardino et al. and Sutton et al. offer an entirely similar description of startups.Startups are known for their high failure rate. Approximately 75-99 percent launch products do not return significant results in the query. A high failure rate can be due to challenges, group issues, difficulty getting support, etc. However, the biggest challenge in software companies, which precedes any application or business challenge, is the ability to make efficient software with minimal understanding of stakeholder requirements and limited cash.

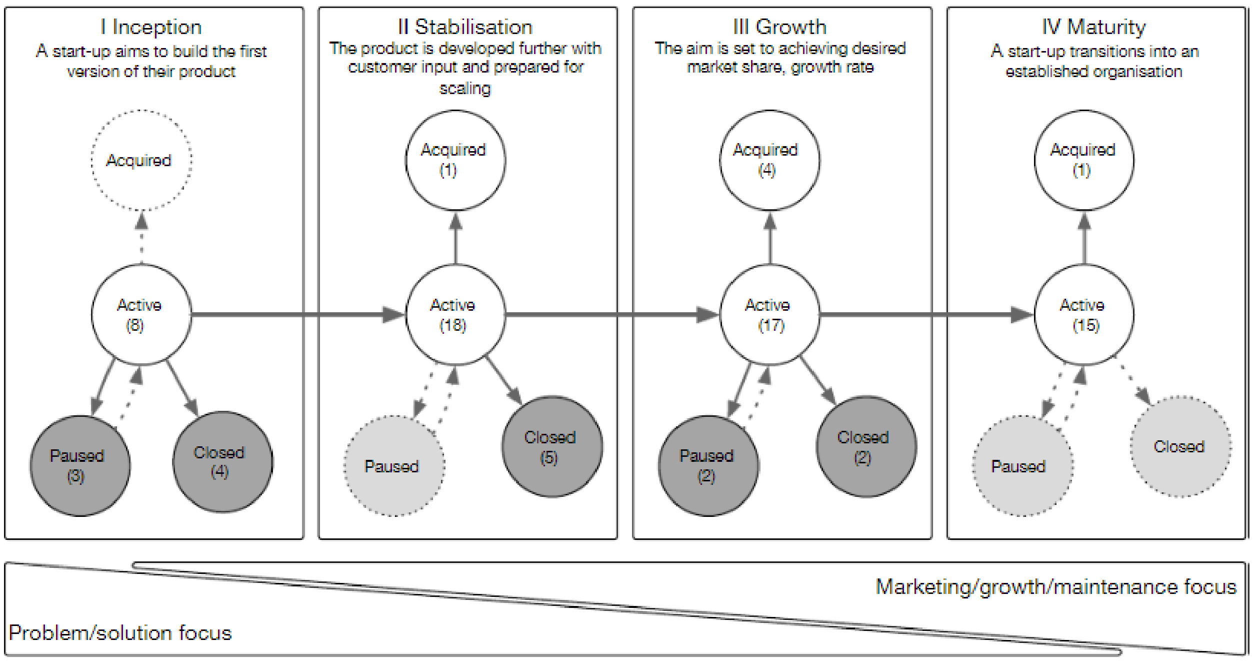
B. *What do we require in terms of software start-ups?*

The release of a methodical assessment of the research in the subject resulted from a broader importance in the study of technology disciplines from the standpoint of software engineering. Unnamed review results were also released in IEEESoftware. The authors of the review emphasise the prospect of launching software as invention tools as well as the paucity of applied study in this area. In comparison to established organisations, the evaluation outlines various contextual problems for software design startups.

Klotins et al. as well as Berg et al. produced two subsequent literature evaluations.The goal is to combine world class launches with SWEBOK's expertise. They conclude that there are many gaps and gaps in the development of startup-specific design practices. however, an essay on the launch of SWEBOK, which is not a specific startup and provides several applicable competencies. For example, question-based condition design and value are based on software design.

C. *Software Engineering practices*

Prior studies have noted that startups typically don't adhere to fashionable engineering techniques and arrive at an ad hoc approach to engineering. A lack of engineering moxie and the company's youth render such an approach to engineering incomplete. Even Nevertheless, there isn't much support from academics for developing engineering approaches that would be appropriate in this situation. The inability of startups to share information contributes to the challenge of practising and studying software engineering in startups.


I Inception
A start-up aims to build the first version of their product
Acquired
Active (8)
Paused (3)
Closed (4)
II Stabilisation
The product is developed further with customer input and prepared for scaling
Acquired (1)
Active (18)
Paused
Closed (5)
III Growth
The aim is set to achieving desired market share, growth rate
Acquired (4)
Active (17)
Paused (2)
Closed (2)
IV Maturity
A start-up transitions into an established organisation
Acquired (1)
Active (15)
Paused
Closed
Marketing/growth/maintenance focus
Problem/solution focus

Since humans are the only ones that possess knowledge of a subject, functioning techniques, and practises, they are lost when a business shuts down and need to be invented over again with each new startup. As a result, building knowledge that includes cutting-edge engineering methodologies for start-ups plays a significant role in releasing exploration. Numerous studies examine product engineering strategies used by startups.Gralha et al. and Melegati et al. examine circumstances bargaining methods in launch-ups as examples. These studies indicate that because it assists in the investigation of the desire occasion and the development of a viable conclusion, conditions engineering is among the most important engineering process domains in start-ups. In a start-up setting, Hokkanen and colleagues provide a framework for managing conceptual user experience design. They contend that in order for a product to satisfy a request, it must meet some minimal functional experience requirements. The frame recognises and values a vibrant user experience.

D. Start-up life cycle models

1) Inception - the time frame between a startup's invention of a product and its initial delivery to a customer. The major objective of this stage is to size and construct the most commercially feasible product while juggling the demands of the client, the resources at hand, and the time constraints.
2) Between the initial product announcement and the point at which it is suitable for expansion is the stabilisation phase. The product should be retired without additional work on the part of the development team at this point.
3) Growth -At this point, the goal is to achieve the required market share and growth. More effort is placed into sales and marketing, and the technical staff must stay up with market developments and new needs.
4) Maturity - it becomes an established organization where the goal is to preserve and establish market share and operations.

These four phases show how startup objects change. In the beginning, the focus is on assuming a relevant issue and identifying a potential outcome. Later, the emphasis will be on improving the launch's operational efficiency and promotion. A launch-up aims to either

a) remain operational and active,

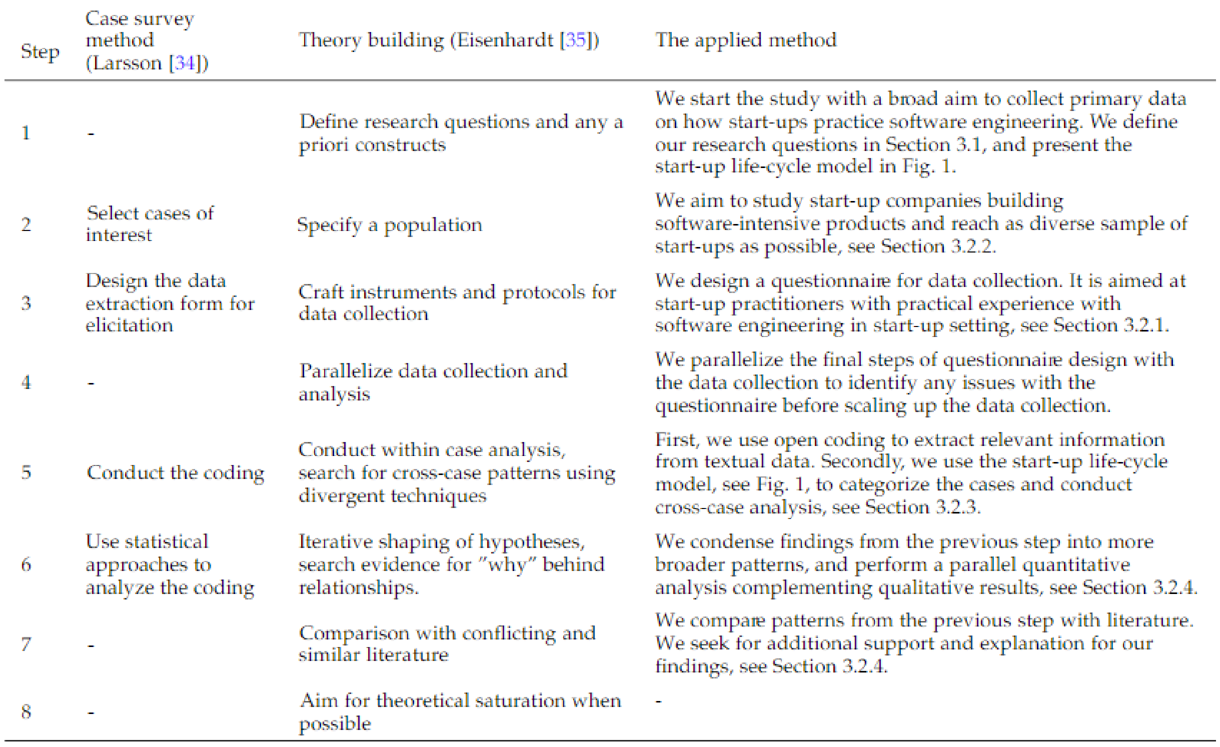
b) go on to the following stage, or

c) at any of the four market stages be acquired by another firm for profit to investors

An organisation may also be shut down at any of these stages.

III. RESEARCH METHODOLOGY

We collect and analyse startup-specific primary data using a case check mechanism. The method is designed to create a balance between thoroughly investigating many examples with conventional case studies and thoroughly doing analytical research on several cases. Case studies compare diverse data sources to provide a lower level of knowledge and internal validity. Surveys are designed to collect data from a wide range of circumstances, and as a result, they are highly universal. The ability to acquire more data than a conventional check and to analyse more instances are the case verification method' key benefits.



Step	Case survey method (Larsson [34])	Theory building (Eisenhardt [35])	The applied method
1	-	Define research questions and any a priori constructs	We start the study with a broad aim to collect primary data on how start-ups practice software engineering. We define our research questions in Section 3.1, and present the start-up life-cycle model in Fig. 1.
2	Select cases of interest	Specify a population	We aim to study start-up companies building software-intensive products and reach as diverse sample of start-ups as possible, see Section 3.2.2.
3	Design the data extraction form for elicitation	Craft instruments and protocols for data collection	We design a questionnaire for data collection. It is aimed at start-up practitioners with practical experience with software engineering in start-up setting, see Section 3.2.1.
4	-	Parallelize data collection and analysis	We parallelize the final steps of questionnaire design with the data collection to identify any issues with the questionnaire before scaling up the data collection.
5	Conduct the coding	Conduct within case analysis, search for cross-case patterns using divergent techniques	First, we use open coding to extract relevant information from textual data. Secondly, we use the start-up life-cycle model, see Fig. 1, to categorize the cases and conduct cross-case analysis, see Section 3.2.3.
6	Use statistical approaches to analyze the coding	Iterative shaping of hypotheses, search evidence for "why" behind relationships.	We condense findings from the previous step into more broader patterns, and perform a parallel quantitative analysis complementing qualitative results, see Section 3.2.4.
7	-	Comparison with conflicting and similar literature	We compare patterns from the previous step with literature. We seek for additional support and explanation for our findings, see Section 3.2.4.
8	-	Aim for theoretical saturation when possible	-

However, the information gathered through a case check is included in the scope of the questionnaire tool. Our studies' validity issues are discussed. The original case check system description advises breaking down the data using statistical and graphical techniques. However, we improve the approach by adding more steps from Eisenhardt's proposition-structure procedure. Eisenhardt offers additional information on transforming a proposition from data despite the compatibility of both styles.

A. *Data Collections and Analysis*

We concentrate data gathering on a form that questions our visitors about their particular start-up circumstance. The information in the survey is drawn from our prior project, the launch-up. A taxonomy of engineering standards, presumptions, and launch-up ambient elements is called a context map.

When creating the survey, we carried out a number of internal and external checks to make sure that all of the questions were significant, simple to comprehend, and that only relevant responses were recorded.

In order to confirm the variety and volume of enquiries, inner author evaluations were done. For the impartial assessments, we consulted with ten members of the Software Start-up Research Network. At first, we requested them to complete the survey and respond to all of the questions in the manner of startup founders. We also created a shared online factory where we could spread their comments and suggestions for improving the survey. The final dataset did not contain their responses. The survey was then tested using four interpreters from different start-ups. Respondents filled out the survey while talking with the initial author of this work during the aviators about the survey's questions, answers, and any potential problems.

IV. TEAM PROCESS AREA

A. *Goals*

Establishing a squad with enough chops and moxie is among the initial steps in a launch-up. Prior research indicates that the squad is a motivator for product design start-ups. Team traits such as unity, cooperation, leadership, and knowledge are regarded as fundamental for good software design. Across our sample, the median platoon size is 4- 8 people and 1- 3 of them focus mainly on software engineering. We notice a tendency of increasing average platoon size throughout the release life-cycle, from one to three in the early phases to four to eight in the stability and growth stages, and more than twenty in the mature stage. According to the comments, in order to form a team, start-ups must solve communication challenges, a lack of domain knowledge, engineering grit, dedication, and product accountability. Statistical analysis of responsibility in the teams shows an association(CramersV = 0.334, $p<0.05$) between unrestricted companies and a lack of responsibility. As a result, establishing a unit that has practical and reciprocating moxie and can function together productively with minimal outflow is important early on in start-ups. By analysing replies from launch-ups at various life-cycle stages, we determined that forming a squad is an issue in the early phases of a launch-up. We observe that start-up sat the beginning and stabilization stages focus on issues associated with building a team. Still, start-ups in the growth and maturity stages focus on challenges related to team operation, Team process area, Inception, and Stabilization phases.

B. Challenges

When we examine the respondents' responses to the question about platoon build, we discovered that a sizable portion, almost 70 out of 84, mention team-related issues. Leadership, teamwork, and platoon grit are the issues. A lack of engineering talent and sphere moxie is cited by startups at various stages of their development. Start-ups are experiencing the most severe shortages at this stage, where only 3 out of every 20 engineers estimate their engineering skills as being on par, and only 1 out of every 6 rate their domain knowledge as being on par. But as the life cycle progresses, we notice a tendency for estimates to increase.

The possibility of survival bias, whereby start-ups that were able to acquire the necessary capabilities were more likely to advance, could be a secondary explanation. We also see that less "claimed" brigades tend to take more time in the beginning stages and frequently "fail" to release the product at all. One startup stated, "We should have searched for additional innovators with experience in producing goods promptly, and don't calculate on commercial " specialists "that produce anything other than completed software." as they reflected on the lessons they had learnt from their platoon. ” According to statistical evidence (Cramer's V = 0.513, p 0.05), engineering chops and sphere knowledge are strongly correlated in the platoon. Engineering expertise is warranted for brigades with lesser sphere knowledge, while brigades with acceptable sphere knowledge are likely to have it in spades. The state of the launch-up is likewise correlated with moxie's position (Cramer0sV = 0.316, p 0.05). Functional start-ups believe that they are far more knowledgeable and skilled than broke or closed businesses. The position of squad chops and expertise is thought to be one of the main success factors in software products as well as the start-up of new companies. Consequently, there can be a negative correlation between the position of platoon chops and start problems. 80, or 33 companies, of the 41 start-ups in growth or maturity stages consider the difficulty in finding specialised knowledge. For instance, recruiting staff with industry-specific experience or executives with a particular set of specialised skills. Startups often think back on how it would have been beneficial to develop similar professional skills earlier. Nevertheless, the responses fail to identify the problem's root. The shortage of experts in a focused and possibly uncharted field may help to explain the lack of experts with sphere-specific experience. Such an explanation emphasises the value of instruction and training for start-ups. However, a crucial explanation could be that experts avoid working with start-ups because, for instance, the company's future is uncertain.

V. QUALITY GOALS AND TESTING PROCESS AREA

A. Goals

According to 32, 27 respondents' responses, product functionality is the characteristic that is mentioned the most frequently throughout all life cycle stages. The alternative and most popular ideal is time-to-request, as shown by 13 launch-ups. Despite the fact that time to request isn't a natural indicator of product quality, it has a significant impact on consumer perceptions of the product. The frequency of other quality claims varies depending on the stage of the life cycle.In the stability, growth, and maturity stages of a quality object, maintainability is continuously documented. Only startups that have reached the mature level report portability. In the stabilisation stage, following functionality and trustability, trustability is the characteristic that is most frequently cited. Shifting quality expectations point to and suggest changing norms and provide additional support for the launch-up life cycle model. We asked launch-ups to investigate how the needed position of quality of product was formed in order for them to reflect on a "decent" level of their quality pretensions.This has to do with defining the MVP's scope and securing non-functional product features. Inaccurately predicting the required quality position could result in bad request perception due to subpar quality or waste by providing excessive quality.

B. Practices

Start-ups apparently use a range of tactics based on internal knowledge, client input, and iterative development to determine their quality standards (P8). Instead of defining any explicit quality objectives, one technique is iteratively finding and enhancing appropriate quality factors. Centred on a single business Instead of building a fixed item, we utilise a continual iterative refinement process, which allows us to assess which areas need improvement on a regular basis, estimate the options, and prioritise them.One launch-up at the beginning stage reported voluntarily aiming As for Being the most basic We search for the solution to this problem that is the simplest in terms of size and complexity. If necessary, this solution can be improved. Can a few hundred drug addicts be supported by that outcome?How many ways may the outcome be improved to encourage more drug users? For locating the required quality location, Regnell et al. develop a quality conditions road mapping model. The approach suggests defining useless, helpful, competitive, and excessive-quality ranges for each of the quality criteria and assigning them to the appropriate quality criteria. Additionally, it uses these ranges to evaluate its own product in comparison to rivals' offerings and identify potential improvements.A model like this could be beneficial in a start-up for determining the necessary costs, the minimal standards for quality that an item must meet in order to provide useful, and possibilities to differentiate among comparable products. This aligns also with internal value that companies often prioritise.

Additional rigorous testing procedures may be beneficial for startups. Effective software testing supports rapid product releases, allowing brigades to reduce the time it takes to demand innovative features and swiftly repeat them.More frequent release cycles are known to increase customer satisfaction and contribute to quickly confirming conditions.

VI. REQUIREMENTS ENGINEERING PROCESS AREA

A. Goals

Finding the needs and limitations placed on a software product is one of the initial steps in any software design. Responses from start-ups at the beginning of their operations concur that product features are mostly designed and built on the knowledge and expertise of the authors. One of the first goals in a launch-up is to dissect created concepts into software conditions and to validate them as a result of constructed conditions. These conditions should be optimised using input from intended guests.As one guru stated "The actual goal of eliciting conditions is to reduce demand for concepts as quickly as feasible without needless problem on meta testing or perpetration.". According to the responses, as soon as the product is introduced in response to the request, start-ups place a greater emphasis on exploiting their visitors as demand sources.Conditions invention becomes less common. Responses, to what extent product conditions are constructed, shift towards disagreement with the statement. Therefore, the tool to swiftly dispel one's own notions is best used in the first stages and before launch-ups have established a review circle and are able to leverage customer feedback for highlight suggestions.

"Any condition that is necessary for proving the growth or value hypothesis is given precedence. Not a high priority is everything that is required to keep the firm running until the number of beyond 100 users. Scoping is a process of removing elements that are not essential from the MVP."

B. Challenges

When the stabilisation and growth responses from active and unrestrained launch-ups are compared, internal funds, which are used by 94 of active startup and 71 of uncontrolled launch-ups, are by far the most frequently employed demand sources. Other sources of demand include comparable items (used by 83 of active launch-ups and 43 of unrestricted launch-ups), as well as request patterns (used by 57 of active launch-ups and 29 of unrestricted launch-ups). However, compared to 91 active launch-ups, only 43 of unconstrained launch-ups have utilised feedback from implicit and live guests. Cramer v = 0.463, p =0.05, indicates that this difference is statistically significant.

A common problem in developing new software products is feature creep, according to reports. It results from late-stage product development changes that were made without the proper analysis. New conditions may be found during development and tone-approved by the platoon, or they may come from outside stakeholders and appear truly pleasant to include in a release. Anyhow, a company should assess the implications and ensure that its high standards of conduct in business are not compromised.

VII. PROJECT MANAGEMENT

A. Goals

The responses demonstrate that start-ups largely seek to utilise outside metrics, such as earnings, number of guests, and customer happiness, to gauge their performance. However, start-ups still use internal criteria as performance indicators, such as platoon effectiveness, timeline compliance, and budget planning. When comparing launches at various life-cycle stages, we find variances. Startups do not report using any criterion to gauge their progress during the early stages of operation, before they have released their product. They still want to leverage outside factors, such the quantity of product drug users, as soon as the product is out.The main success criteria are external at this point and are intended to cover average product abandonment rates. This is known as the stabilisation stage, which occurs shortly after the product is introduced. As start-ups proceed through their life cycles, criteria become more precise, linked to important business benchmarks, and carefully balanced between internal and external factors. In addition to external, request relinquishment criteria, internal platoon performance criteria are discussed and utilised to evaluate startup performance.

B. Challenges

We created a number of models based on the literature to help with early launch-ups. These approaches include defining and verifying the client's requirement before validating a solution to that need, then confirming the viability of the product through small- and large-scale prototypes. Fagerholm suggests a kind comparable technique that focuses on continuous trial. However, more research has to be done to determine how similar approaches are utilised to direct work in early start-ups.

According to a survey of 1000 Finnish start-ups between 2010 and 2013, outside funding is not related to start-up problems. In other words, start-ups with more available funds don't necessarily have better prospects of surviving and succeeding than start-ups with more constrained funds. Additionally, startups without external financing is anticipated to produce additional profits in the long run. Similar findings underline how important scoping, planning, and management are for maximising the use of any amount of money.

VII. CONCLUSION

In this study, we looked at how 84 software start-ups used software engineering to produce products that were cutting-edge and software-ferocious.In order to frame the findings and provide a launch-up life-cycle model, the domains of platoon, conditions engineering, values focuses, quality and testing, architecture and designs, and software development operation process were looked at. Similar architectural design emphasises the goals, challenges, and processes particular to each stage and process area. We have talked about our results in relation to related research and created tasks that interpreters would actually complete in the real world. We conclude that every process area is relevant for start-ups and, in essence, is the same as established businesses. However, managing the development of methods in all the key processes simultaneously with a tight margin for error may be the most important distinction and challenge in doing software engineering at start-ups. However, while it is true that startups are by their very nature risk-taking and that committing crimes during the process is inevitable, there is a difference between falling into suggested traps and ignoring fashionable engineering practices.

References

[1] N. Paternoster, C. Giardino, M. Unterkalmsteiner, T. Gorschek, and P. Abrahamsson, “Software development in startup companies: A systematic mapping study,” Information and Software Technology, vol. 56, no. 10, pp. 1200–1218, oct 2014.

[2] C. Giardino, M. Unterkalmsteiner, N. Paternoster, T. Gorschek, and P. Abrahamsson, “What Do We Know about Software Development in Startups?” IEEE Software, vol. 31, no. 5, pp. 28–32, sep 2014.

[3] C. Gralha, D. Damian, A. I. T. Wasserman, M. Goulao, and J. Araujo, “The evolution of requirements practices in software startups,” pp. 823–833, 2018.

[4] Eriks Klotins, Michael Unterkalmsteiner, Member, IEEE, Panagiota Chatzipetrou, Tony Gorschek, Member, IEEE, Rafael Prikladnicki, Nirnaya Tripathi, and Leandro Bento Pompermaier, “A progression model of software engineering goals, challenges, and practices in start-ups”

[5] M. Crowne, “Why software product startups fail and what to do about it,” in Engineering Management Conference. Cambridge, UK: IEEE, 2002,

[6] G. Carmine, N. Paternoster, M. Unterkalmsteiner, T. Gorschek, and P. Abrahamsson, “Software Development in Startup Companies: The Greenfield Startup Model,” IEEE Transactions on Software Engineering,