Towards Effective Technical Debt Decision Making in Software Startups: Early-Stage

Abdullah Aldaeej
Department of Information Systems
University of Maryland Baltimore County
Baltimore, MD, United States
aldaeej1@umbc.edu

DOI: 10.1145/3356773.3356793
http://doi.acm.org/10.1145/3356773.3356793

ABSTRACT

Context: Technical Debt (TD) is a metaphor used to describe outstanding software maintenance tasks or shortcuts made in the software development to achieve short-term benefits (i.e. time to market), but negatively impact the software quality in the long term. TD is quite common in a software startup, which is characterized as a young company with low resources and a small client base, aiming to accelerate time to market. Decisions related to TD can be critical for startup success. *Objective*: I aim to understand the relationship between TD decisions and the success or failure of software startups, and explore the best practices related to TD decisions that would better contribute to the startup success. *Method*: I plan to apply multiple retrospective case studies in different software startups that succeed or failed to pass the startup period and become a mature organization. Semi structured interviews will be used to collect data from the team who was involved in the software development in the startup era. *Contribution*: The outcome of this study will help software founders/entrepreneurs to make effective TD decisions during the startup timeframe; that can better contribute to the startup success and decrease the risk of the startup failure.

Categories and Subject Descriptors

K.6.3 [**Management of Computing and Information Systems**]: Software Management – *software maintenance.*

Keywords

Technical debt, Decision making, Software startups.

1. INTRODUCTION

A startup is a new company initiated by founders or entrepreneurs to seek profitable and scalable business model. It refers to the process of creating a new business, starting from an idea until a well-established company. The software startup is a startup that involves software development activities during the startup evolution cycle. In this type of startups, software is considered as a major component and a prerequisite to create the business, where the business model depends on the software. The software startups would result in different forms of companies with different business models.

Startups are an important driver for economic growth [1]. The revolution of advanced technologies has created a new market opportunity and various business models. As a result, the number of software startups has increased significantly in the past decade. Despite the presence of a large number of software startups, 90 percent of them fail due to self-destruction rather than competition [2],[3]. These failures come from different factors such as financial, marketing, or team organization. Some examples are insufficient funding to operate the activities, failure to find the appropriate product-market fit, and failure to build an effective team [4]. In addition to these factors, there are some unique challenges related to software development that can lead to startup failure [3]. These challenges increase the attention toward investigating software engineering activities in the startup context; and one of these activities is TD management [1].

TD is an essential concept in the context of software startup. In this context, the development speed is vital in order to quickly experiment with the software product and respond to customer feedback. In addition, software startups usually suffer from lack of human resources and experience. Thus, software startup is highly vulnerable to large TD accumulation. Quality Assurance (QA) activities and testing are mostly skipped, and testing is only conducted from the perspective of customer acceptance [1]. Although accumulating TD can bring some short-term benefits to startups (i.e. faster development to receive customer feedback quickly), the presence of TD can negatively affect software quality in the long term. At some points in the software startup evolution, the software product scales up significantly in terms of the number of new features and the size of the user base. Some of the software development issues in startups (such as team member shortage, lack of experience, high dependency in third party platform, and lack of well-established process) can lead to startup failure [3]. These issues are also part of the main causes that lead to accumulating TD [5].

The main goal of this dissertation is to understand the effect of TD decisions on the software startup success; and explore best practices related to TD decisions that can effectively contribute to the startup success. In general, startup success means the successful transformation from a startup to a mature company (i.e. achieving the startup short-term goals in terms of the required net-profit, market-share or growth rate, etc.). Each software startup might have different measures of success. The research questions are formulated as follows:

RQ1: How TD decisions affect software startup success?

RQ1 Rational: the purpose of this question is to investigate the impact of TD decisions on the startup success. The answer to this question will shed the light on the decision consequences, and how these consequences led to either success or failure. It will also provide insight about the relationship between TD decision and startup success.

RQ2: In which ways can TD decisions effectively be made during the evolution of software startups?

RQ2 Rational: I aim to investigate how TD decisions should be made in a way that better contributes to the startup success. The answer to this question will identify the best practices related to TD decisions. For example, the type of TD that needs to be accumulated and when. What is the best time to repay TD, and the common types of TD that should be repaid during the startup evolution period.

2. BACKGROUND AND RELATED WORK

In this section, I will provide some background about TD decision making. Then, I will discuss the software startup, its life cycle, and some related works about TD in software startups.

2.1 TD Decision Making

TD literature provides different approaches to support TD decision making. One approach supports TD decisions via quantifying the value of TD by estimating TD principal and interest [6],[7],[8]. TD principal is the estimated effort to fix TD, whereas TD interest reflects the extra cost that results from the presence of TD. In this approach, the decision can be made based on the proportion of TD interest to the principal. Another approach focuses on visualizing TD using some risk assessment methods [9]. This approach emphasizes on analyzing the impact and assessing the severity of TD. It supports TD decisions by prioritizing TD based on severity. Another approach supports TD decision through using some optimization methods [10]. In all these approaches, TD decisions are supported primarily by providing TD information, but without examining the comprehensive view of the decision-making process [11].

2.2 Software Startup Definition

There is no consensus on the definition of software startup. Some researchers consider software startups as organizations that develop cutting edge software product [12],[13]. The software startups are viewed as a product-oriented where the product is developed with little or no operating history, aiming to rapidly scale their business. Sutton [14] characterizes software startups as a young and inexperienced organization that work with very limited resources, immaturity, and dynamic technologies and markets. Additionally, Coleman and O'Connor [15] define software startup as unique organizations that develop software without a prescriptive methodology. Despite the lack of consensus, the previous definitions of software startups share some common characteristics as follows:

- Extreme uncertainty
- Young and inexperienced organization
- Scarce resources
- Little to no operating history
- Little to no customer base
- Seeking sustainable and scalable business model

In contrast to mature software companies, a software startup is a temporary organization that seeks a scalable and profitable business model [16]. Software startups have some unique characteristics as compared to other software development environments. In the startup context, the focus is more on what product to develop rather than how to develop the product. This involves high uncertainty about the software requirements. The development teams search for a suitable product that can achieve a sustainable and scalable business model. In addition, software startups operate with no customers and operating history, following a market-driven approach that intensively focuses on collecting customer feedback and attaining customer needs.

In this dissertation, my definition of software startup will adopt the common characteristics of the software startups. It will also include any startups where the software development is the major activity; that the startups' business model depends heavily on the developed software. This definition will include any company that generate money from the developed software. Examples of such company are Social network applications (i.e. Facebook, Snapchat), E-commerce applications (i.e. Amazon, Groupon), Digital marketing applications, Game applications, etc.

2.3 The Evolution of Software Startups

Crowne [3] depicts the evolution of product development in software startups into four stages: *Start-up, Stabilization, Growth, Maturity*. Each stage has some critical development issues that can lead to the company failure. The startup stage reflects the timeframe from identifying the product idea until the first release. This stage is characterized as developing the business concept with no well-defined requirements. During the startup stage, the focus is to develop a simplest form of the product idea which is called Minimum Viable Product (MVP). So, the product is developed as quickly and simply as possible. Also, this stage considers the use of inexperienced developers to minimize the development cost.

The second stage is 'Stabilization' which refers to the time between the first release (MVP) until the product is ready to be commercialized without causing any overhead on the product development. This stage is characterized as proving the business concept. During this stage, the first release is validated with customers and the customer feedback is collected. The product goes through a series of experimentations until finding the good market fit. Some development issues might arise during this stage such as high volume of new feature requests and complex defects that could be revealed due to the MVP shortcuts.

The third stage 'Growth' begins when the product is ready to be commercialized until the market share and growth rates have been established. During this stage, the company pays more attention to growing its market share and attaining new customers. The software product grows in terms of functionality and user base. Some development issues might occur due to team shortage, high dependency on third-party platform, and no well-established process [3].

The last stage 'Maturity' is attained when the product achieves the required market share and growth rate. At this point, the startup is successfully transformed to a mature company. The mature stage is characterized as having a well-established team and all necessary processes to support the product development.

2.4 Technical Debt in Software Startups

Software startups are highly prone to TD due to their unique characteristics. Time to market is vital in this context. The software is developed with a fast evolutionary delivery approach in order to validate the product in the market as soon as possible. With the high uncertainly in this context, the quick release is important to collect customer feedback early and find the market niche for the product. The rapid evolution and high uncertainty are the key characteristics for software startups [13]. These characteristics are among the main organizational factors that influence the accumulation of TD [17].

Giardino et al. [18] investigated the software development strategy employed by startups using the grounded theory approach. Based on the empirical findings from 13 startup cases, they created an abstract model called 'Greenfield Startup Model (GSM)' that depicts the main themes characterized by software development in startups. Sped-up development is found as the core theme which illustrates the importance for startups to release the product as quickly as possible. Also, the study found a causal relationship between sped-up development and TD accumulation, i.e. that the requirement to develop faster influences the startups to incur TD as an investment, whose repayment may never come due. The study finds that the negative impacts of TD in startups would be on morale, productivity, and product quality.

Yli-Huumo et al. [19] study the relationship between business model experimentation and TD, in order to explore if conducting the business model experimentation impacts the amount of TD incurred. The business model experimentation is a technique used to validate assumptions made about a product with real customers, before creating the actual product. The result shows that the relationship between business model experimentation and the occurrence of TD has a U-shaped curve, that the use of experimentation reduces the amount of TD. But focusing too much on the experimentation decreases the effort on TD repayment which can have negative consequences on the product quality.

In addition, software startups have immature teams, which may influence the accumulation of TD. Klotins et al. [20] explore the antecedent associated with TD in startups. They found that team size and experience of the start-up is a leading precedent for accumulating TD. The development team in startups is typically very small, who handle multiple diverse roles ranging from software engineering, to marketing and sales [18]. So, the startup team works under extreme time pressure, which may lead to incurring TD. Moreover, startups usually hire young and inexperienced developers because of the limited resources. The lack of knowledge and experience of the development team is one of the main causes for TD accumulation [5], [21], [17].

Software startups usually grow at a faster rate once the market niche is found. During the growth stage, the startups scale up in terms of the team size, the number of clients or users, and the number of features. TD becomes more severe in this startup evolution stage [20]. With the presence of TD, scaling up the software product becomes a barrier that might prevent the startups from delivering new features faster, and gaining new clients. Thus, it is important at this startup stage to consider TD repayment before adding new features [20].

Gralha et al. [22] use the grounded theory approach to study 16 startups with an emphasis on the requirements evolution. They characterize the evolution of requirements practices in software startups into six main dimensions, and one of them is TD. They identified three phases regarding TD decisions as well as the trigger points that cause the transition from one phase to the next. In the first phase, TD is acknowledged and accepted. Then, when the size of the team and the number of features is increased, it causes the transition to the next phase, which is tracking and recording TD. After that, when the customer retention rate decreases, or the amount of negative feedback increases, it causes startups to move to the third phase, which is manage and control TD. The results of this study support TD decisions in startups by identifying the conditions where incurring or repaying TD is important. However, this study focused primarily on TD from the requirements perspective.

The aforementioned studies reveal that software startups have many contextual factors that increase the accumulation of TD. According to the research agenda by Unterkalmsteiner et al. [1], there is a need to explore TD management and decision making in the startup context. There is a lack of strategies that support dealing with TD during the startup timeframe. This dissertation will address this gap by exploring TD in the startup context in order to support effective TD decision making in this context.

3. METHODOLOGY

In order to achieve the research goal, I plan to conduct multiple retrospective case studies for different software startups that are established within the past five years. The retrospective studies will be based on in-depth analysis of major TD decisions made during the software startup evolution; starting from the development of the initial version (MVP) until the startup either success or fail. I decide to use the retrospective approach because software startups take an average of 3 years to be a mature company [23], which is beyond the PhD dissertation timeframe. The use of retrospective case study methodology will allow me to quickly explore the entire startup evolution period that occurred in the past.

3.1 Case Selection

The selection of the cases will be based on convenience sample [24], selecting the cases that I know their founders or CEOs. I will also consult some researchers who previously interviewed software startup teams. In the initial step, I will contact the founders/ CEOs whom I know. The purpose of this initial contact is to assess their willingness to participant in the study. Also, to verify the possibility of reaching other team members who involve in the startup software development. Furthermore, the initial contact will verify the time when the software startup development began. Since the period of software startups is an average of 3 years [23], the population of this study will be the software startup that are established within 3 – 5 years. This time bound is also adopted from [17]. It is important for the selected case to spend at least three years in order to better understand whether the startup succeed or failed. Also, it is better to select the startups that succeed or failed recently (not long time ago) to facilitate recalling past startup experience. So, using five as a maximum year age of the startup can be a reasonable criterion to satisfy that. However, this maximum can be subject to variation depends on the case availability. Finally, the initial contact will help me to refer to other cases (snowballing) that can be suitable for the study.

Since most of the software startups that I know have web or mobile based product, the software startup cases in this dissertation will be more specific to web/mobile application. Although this selection procedure brings an issue in the external validity, it will provide more confidence results for a specific software startup context (web/mobile app).

After completing the initial step, the final selection of the cases will be based on the following criteria:

- The case age should be at least three years, since the start of developing the initial software product (MVP).
- There should be a possibility to interview at least three persons per case.

Additional 'optional' criteria would be:

- The interviewees in a case should have different roles.
- The case age should not be far longer than 5 years.
- There should be a possibility to access the startup project documents (i.e. issue tracking, communication tool, meeting minutes, etc.).

Once the cases are selected, an invitation message will be sent to a founder or CEO in each case. The invitation message will include the purpose of the study, a brief definition of TD and its related decisions, and the consent form. The brief TD definition adopts the Dagstuhl's definition [25] that TD is *" a collection of design or implementation constructs that are expedient in the short term but set up a technical context that can make future changes more costly or impossible. It presents a liability whose impact is limited to internal system qualities, primarily maintainability and evolvability."* This definition is further illustrated utilizing McConnell's [26] definition that *"TD is a metaphor used to represent outstanding tasks that could not be implemented in the present, but it may be implemented later with additional cost. Example of outstanding tasks would be violations of architecture/code design best practices or pending maintenance tasks (such as bug-fixing, testing, document update, etc.)."* The founder or CEO in each case will be asked to forward the invitation message to their team members who involved in the startup development.

In addition, our TD definition includes some decision scenarios that relate to TD decisions as listed below. The first two scenarios are related to the decision to accumulate TD, whereas the last two scenarios are associated with the decision to payoff TD. These examples help clarify what is meant by a "TD decision":

- Purposefully make a design or implementation shortcut that does not fulfill the quality standard.
- Purposefully delay a maintenance task (i.e. delay the fix of an identified bug, delay the fix of a design rule violation, delay the testing of a software change, delay document update, …).
- Implement an outstanding maintenance task that was delayed before.
- Perform an internal improvement of the code understandability or maintainability (i.e. code refactor or redesign).

3.2 Data Collection

The data will be collected using semi-structured interviews. The interviews will be either in-person or online, depends on the geographic location of the participants, and it is expected to take about 60 minutes. The data collection will be performed in two rounds in order to facilitate top-down analysis of TD decisions in each case. In the first round, I will interview the founder or CEO in each of the selected cases. The first-round interviews will help to understand the major TD decisions made throughout the startup timeline. The founders are the best person that oversee the entire startup stages; and are the main decision makers in the startup. Also, the founders usually know the success criteria related to their startups. So, the major TD decisions identified in this round can be better related to the startup success.

The second-round interviews will target other team members (i.e. developers) who involve in the startup development. The major TD decisions identified in the first round will be used to further collect additional perspective about the major TD decisions. Participants will be asked to discuss only those major TD decisions that they involved in, since developers might join the startup at different time. For example, the developers might be hired in the earlier stage (when developing the MVP), or they might join the team during the MVP experimentation or during the MVP expansion and grow. In addition, participants will be given an opportunity to identify and discuss other TD decisions that are not identified in the first round.

The interview guide will apply the Critical Decision Method (CDM) [27], in order to better simulate the discussion of the previous TD decisions made. The CDM is a retrospective semi-structured interview that employs a set of probes about an incident or task that required subjective judgment. It is used as a knowledge engineering approach to elicit expert knowledge and decision strategies for decisions that rely heavily on experience and gut feeling. Using CDM format, the participants will be asked to describe some TD decisions during the startup era. Then, for each decision, some probe questions will be asked which are guided to answer the research questions.

During the interviews, I will first briefly explain to the participants the definition of TD and its related decisions to confirm that we have the same understanding of the TD concept before proceeding the interview. At the beginning of the interview, I will ask some demographic questions related to the case context which include general information about the context (i.e. case domain, programming languages used), the project characteristics at the decision time (i.e. size of the team, team distribution), and participant characteristics (i.e. education background, experience, roles in the project).

After the demographic questions, the interview questions will slightly vary based on the interview round, In the first-round interviews, the participants will be asked an open question that allow them to tell their startup story since the beginning of the startup idea. The discussion will follow a chronological order based on three main startup stages [3]; 1) developing the MVP 2) experimenting the MVP, expanding and growing the MVP. Then, I will narrow down the discussion to major TD decisions made during the startup evolution stages. On the other hand, participants in the second-round interviews will be asked when and how long they join the startup. Then, the discussion will be limited to the timeframe of their involvement in the startup. The participants will be asked about those major TD decisions (identified in the first-round) that they involved in them. In addition, they will be asked to identify and discuss additional TD decisions that they involved in them.

In both interview rounds, the discussion about TD decisions will be in a sequence chronological order, starting with the first TD decision then moving forward. For each decision, participants will be asked about the reason and the consequences of the decision, and how these consequences contributed to the success or failure of the startup. In the last part of the interviews, the participants will be asked about their opinions about the decisions made; and how they should have been made to better contribute to the startup success.

3.3 Data Analysis

The data collected from the interviews will be audio recorded, transcribed, and then analyzed using the NVivo tool[1]. The analysis will follow deductive and inductive approaches. Using a top-down process for the data coding, the core (top) categories will be identified based on the research questions as follows: *decision impact and decision best practices*. First, the interview transcripts will be used to extract quotes and chunks, which will be assigned to their corresponding core categories. Then, I will perform open coding on the extracted chunks; after that, axial coding will be used to organize the data into meaningful groups within each core category. Finally, I will use constant-comparison and cross-cases analysis to generate the conclusions that would address the research questions.

In order to provide better insight in the analysis, I will use the coding scheme (Table 1) to extract additional context information about TD decisions. For each TD decision, I will code the type of TD and the type of decision as well as the startup evolution stage where the decision is made. The type of TD will be coded based on the classification in [28] which classified TD into different types (i.e. architecture debt, test debt, defect debt, etc.). The TD decisions will be coded as either accumulating TD, or repaying TD, or both; based on which TD decision scenario (that were given to the participant in the invitation message) are associated with the decision described. Also, the startup evolution stages in [3] (i.e. startup, stabilization, and growth) will be leveraged to code the stage when the TD decisions are made.

Table 1. Coding scheme for TD decisions

TD decision Dimension	Code
TD type	Based on the TD type classification in [28]
TD decision type	Based on the TD decision scenarios given to participants
Startup stage	Based on the classification of startup evolution stages in [3]

3.4 Evaluation Plan

One of the selected cases will be used, at the end, to evaluate the study result. Given that, all the selected cases (except one) will be used as exploratory cases to answer the research questions and propose TD decision making best practices. After that, the last case will be conducted to evaluate the study results. The effective way to evaluate the result would be to apply the proposed best practice on an actual software startup, and then verify whether it improves the success rates of the startup. However, this process needs a long time (about 2-3 years) in order to observe the results. An alternative approach would be to retrospectively evaluate the result on either a succeed or failed startup. In this approach, the evaluation case will resemble the process of the previous exploratory cases. However, the last part of the interview (about how the decisions should be) will be approached differently. I will show the generated best practices to the participants; and then ask them whether applying it would have better contributed to their startup success.

3.5 Timeline and Milestones

Figure 1 shows the timeline plan for the study with milestones. Level 1 depicts the high-level overview of the study timeline. Initially, I am currently conducting an empirical study in TD decision making. This study is designed to explore a comprehensive view of the decision-making process related to TD. The preliminary findings from this study indicate one important factor, named 'market condition' that is a core precedent of other factors that influence TD decisions. One of the market conditions revealed in the study was software startups. This motivates me to search for TD in the startup context. Then, I found that this context is more important to be explored in terms of TD decision making; since it incorporates many contextual factors that lead to TD accumulation. As a result, I have revised the study design to focus on TD decision making in the software startup context. The new study design is planned to be presented in IDoESE (Sep 2019) to receive valuable feedback from ESE researchers. After that, the study design will be updated, considering the received feedback. Then, the final proposal is planned to be defended by Dec 2019.

The implementation of the study is planned to begin in Jan 2020. The data collection and analysis phase would take 6-8 months. The level 2 in

[1] https://www.qsrinternational.com/nvivo/home

Figure 1 further illustrates the data collection and analysis. In this phase, I will select the study cases (as described previously in section 3.1). One of these cases will be used later for the evaluation. So, the exploratory cases will be used to generate the study result. After that, the evaluation case will be conducted to validate the research results.

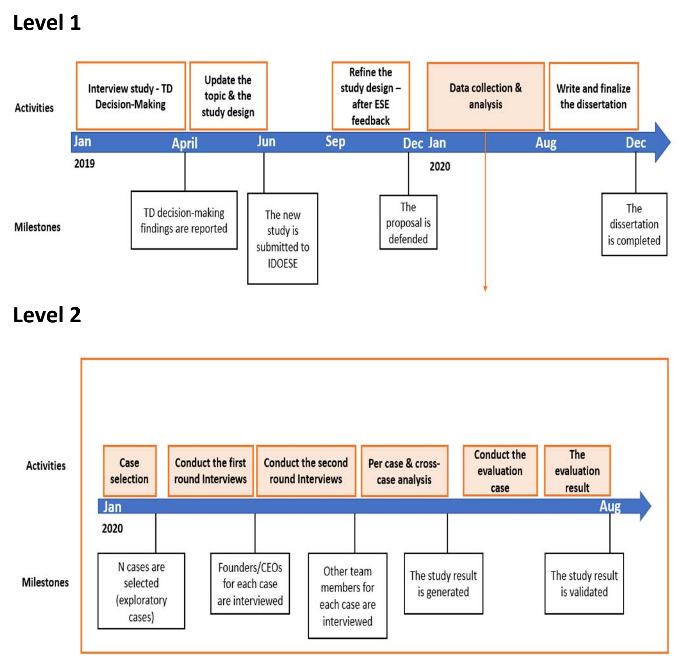
Level 1
Activities
Interview study - TD Decision-Making
Update the topic & the study design
Refine the study design – after ESE feedback
Data collection & analysis
Write and finalize the dissertation
Jan 2019
April
Jun
Sep
Dec
Jan 2020
Aug
Dec
Milestones
TD decision-making findings are reported
The new study is submitted to IDOESE
The proposal is defended
The dissertation is completed
Level 2
Activities
Case selection
Conduct the first round Interviews
Conduct the second round Interviews
Per case & cross-case analysis
Conduct the evaluation case
The evaluation result
Jan 2020
Aug
Milestones
N cases are selected (exploratory cases)
Founders/CEOs for each case are interviewed
Other team members for each case are interviewed
The study result is generated
The study result is validated

Figure 1. The study timeline plan.

3.6 Possible Threats to Validity

One of the threats will be related to external validity. Since the study sample is expected to be small, the result cannot be generalized to all software startups. Despite this issue, the selected cases would have a very specific context in the software startup which is web and mobile application. Specifying the cases to this type of software startup would provide more confident result related to this particular context.

Other issues are related to internal and construct validity. First, the result evaluation is based on the participants' opinions. Due to time constraint, the study result will be evaluated retrospectively by providing a qualitative evidence from interviews. Second, there will be a lack of number of participants (around 3 participants per case). This is inevitable since the team size of software startup is small (around 2-5 individuals). Unlike well-establish companies, software startups operate with a very small team. Thus, the number of participants in each case is expected to be low. To ensure adequate number of participants in each case, at least three participants needed in order to select any case. I may consider using an incentive approach to recruit participants to improve the participant rate.

In addition, the study depends solely on the interview as the data collection method. This might affect the reliability of the collected data. However, I will mitigate this issue by interviewing participants from different startups and different roles. Also, I will try to access the project documents (if possible) to support the interview data. The case that grant access to its documents will be given high priority in the selection process.

Finally, the understandability of the TD concept can threaten the validity of the result; since this concept might be interpreted differently by different participants. To mitigate this issue, I will send a brief TD definition and its related decisions as part of the invitation message. In addition, I will verbally explain TD and some scenarios related to TD decisions at the beginning of the interview. This would help ensure a common understanding of the TD concept.

4. PROGRESS OF THE STUDY

This dissertation study is a result of insights I am gaining from my current interview study. The purpose of the interview study is to explore how TD decisions are made and identify ways to improve TD decision-making practices. So far, I have interviewed seven software practitioners from seven different organizations during January – April 2019. The preliminary results show some factors that influence TD decisions. One of these factors led me to narrow down the research topic to a specific context (software startups). After an ad-hoc literature search about software startups and technical debt, I found an interesting research direction where TD decision making needs to be explored.

Although I made progress from conducting the initial interview study, shifting the focus to the software startup context entails some changes in the research questions and the study design. In this dissertation, I focus more on the TD decision outcomes and how it could impact the startup success. In addition, the study design is changed to be retrospective software startup cases. Instead of centralizing the interview discussion around one TD decision, this dissertation addresses a series of TD decisions throughout the startup life cycle. As a result of these major changes, I would consider this proposal an early stage proposal. The progress made would be the ad-hoc literature review on software startup which help me to refine the study design.

5. THE EXPECTED CONTRIBUTIONS

The outcome of this study is expected to help software founders/entrepreneurs when making TD decisions during the startup period. The expected contribution can be in a form of TD decision best practices that would guide the software founders/entrepreneurs toward effective TD decisions that better contribute to startup success.

6. REFERENCES

[1] M. Unterkalmsteiner *et al.*, "Software Startups – A Research Agenda," *e-Informatica Software Engineering Journal*, vol. Vol. 10, no. nr 1, 2016.

[2] M. Marmer, B. L. Herrmann, E. Dogrultan, R. Berman, C. Eesley, and S. Blank, "Startup genome report extra: Premature scaling," vol. 10, pp. 1–56, 2011.

[3] M. Crowne, "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*, 2002, vol. 1, pp. 338–343 vol.1.

[4] C. Giardino, S. S. Bajwa, X. Wang, and P. Abrahamsson, "Key Challenges in Early-Stage Software Startups," in *Agile Processes in Software Engineering and Extreme Programming*, 2015, pp. 52–63.

[5] N. Rios, R. O. Spínola, M. Mendonça, and C. Seaman, "The Most Common Causes and Effects of Technical Debt: First Results from a Global Family of Industrial Surveys," in *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, New York, NY, USA, 2018, pp. 39:1–39:10.

[6] C. Seaman and Y. Guo, "Measuring and Monitoring Technical Debt," in *Advances in Computers*, vol. 82, 2011, p. 22.

[7] K. Schmid, "A Formal Approach to Technical Debt Decision Making," in *Proceedings of the 9th International ACM Sigsoft Conference on Quality of Software Architectures*, New York, NY, USA, 2013, pp. 153–162.

[8] A. Chatzigeorgiou, A. Ampatzoglou, A. Ampatzoglou, and T. Amanatidis, "Estimating the breaking point for technical debt," in *2015 IEEE 7th International Workshop on Managing Technical Debt (MTD)*, 2015, pp. 53–56.

[9] P. S. M. dos Santos, A. Varella, C. R. Dantas, and D. B. Borges, "Visualizing and Managing Technical Debt in Agile Development: An Experience Report," in *Agile Processes in Software Engineering and Extreme Programming*, 2013, pp. 121–134.

[10] N. Ramasubbu and C. F. Kemerer, "Managing Technical Debt in Enterprise Software Packages," *IEEE Transactions on Software Engineering*, vol. 40, no. 8, pp. 758–772, Aug. 2014.

[11] C. Becker, R. Chitchyan, S. Betz, and C. McCord, “Trade-off Decisions Across Time in Technical Debt Management: A Systematic Literature Review,” in *Proceedings of the 2018 International Conference on Technical Debt*, Gothenburg, Sweden, 2018, pp. 85–94.

[12] O.-P. Hilmola, P. Helo, and L. Ojala, “The value of product development lead time in software startup,” *System Dynamics Review*, vol. 19, no. 1, pp. 75–82, 2003.

[13] 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.

[14] S. M. Sutton, “The role of process in software start-up,” *IEEE Software*, vol. 17, no. 4, pp. 33–39, Jul. 2000.

[15] G. Coleman and R. V. O’Connor, “An investigation into software development process formation in software start□ups,” *Journal of Ent Info Management*, vol. 21, no. 6, pp. 633–648, Oct. 2008.

[16] V. Berg, J. Birkeland, A. Nguyen-Duc, I. O. Pappas, and L. Jaccheri, “Software startup engineering: A systematic mapping study,” *Journal of Systems and Software*, vol. 144, pp. 255–274, Oct. 2018.

[17] T. Besker, A. Martini, R. E. Lokuge, K. Blincoe, and J. Bosch, “Embracing Technical Debt, from a Startup Company Perspective,” in *International Conference on Software Maintenance and Evolution*, Madrid, Spain, 2018, p. 12.

[18] C. Giardino, N. Paternoster, M. Unterkalmsteiner, T. Gorschek, and P. Abrahamsson, “Software Development in Startup Companies: The Greenfield Startup Model,” *IEEE Transactions on Software Engineering*, vol. 42, no. 6, pp. 585–604, Jun. 2016.

[19] J. Yli-Huumo, T. Rissanen, A. Maglyas, K. Smolander, and L.-M. Sainio, “The Relationship Between Business Model Experimentation and Technical Debt,” in *Software Business*, 2015, pp. 17–29.

[20] E. Klotins *et al.*, “Exploration of Technical Debt in Start-ups,” in *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*, 2017, pp. 75–84.

[21] J. Yli-Huumo, A. Maglyas, and K. Smolander, “The Sources and Approaches to Management of Technical Debt: A Case Study of Two Product Lines in a Middle-Size Finnish Software Company,” in *Product-Focused Software Process Improvement*, 2014, pp. 93–107.

[22] C. Gralha, D. Damian, A. I. (Tony) Wasserman, M. Goulão, and J. Araújo, “The Evolution of Requirements Practices in Software Startups,” in *Proceedings of the 40th International Conference on Software Engineering*, New York, NY, USA, 2018, pp. 823–833.

[23] V. Berg, J. Birkeland, A. Nguyen-Duc, I. O. Pappas, and L. Jaccheri, “Software startup engineering: A systematic mapping study,” *Journal of Systems and Software*, vol. 144, pp. 255–274, Oct. 2018.

[24] P. Runeson, M. Host, A. Rainer, and B. Regnell, *Case Study Research In Software Engineering*. Hoboken, New Jersey: John Wiley & Sons, Inc., 2012.

[25] P. Avgeriou, P. Kruchten, I. Ozkaya, and C. Seaman, “Managing Technical Debt in Software Engineering (Dagstuhl Seminar 16162),” *Dagstuhl Reports*, vol. 6, no. 4, pp. 110--138, 2016.

[26] S. McConnell, “Technical debt,” *10x Software Development Blog. URL = https://www.construx.com/resources/whitepaper-managing-technical-debt/*, 2007. .

[27] G. A. Klein, R. Calderwood, and D. MacGregor, “Critical decision method for eliciting knowledge,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 3, pp. 462–472, May 1989.

[28] N. Rios, M. G. de Mendonça Neto, and R. O. Spínola, “A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners,” *Information and Software Technology*, vol. 102, pp. 117–145, Oct. 2018.