Architecting for Scale: The Case for Systematic Software Reuse in Managing Technical Debt in Start-ups

Mercy Njima
University of Antwerp
Antwerp, Belgium
mercy.njima@uantwerpen.be

ABSTRACT

Most start-ups aspire to become non-start-ups someday. One would argue then that architecting for scale means doing it right the first time. However, start-ups usually start with a single idea in search of market fit. Taking time to design and implement a scalable architecture is time that is not spent responding to customer demands and is usually not a priority. This short-term vision may lead start-ups to accumulate technical debt. This work is geared towards understanding how start-ups find a viable trade-off between customer demands and long term goals. We conduct expert interviews at start-ups and scale-ups with the hypothesis that reusing software for the management of technical debt would allow a start-up to quickly respond to the demanding needs of the market in the long run.

CCS CONCEPTS

• **Software and its engineering → Software design tradeoffs**; **Software product lines**; **Maintaining software**; *Software usability*; *Empirical software validation*; • **Social and professional topics** → *Reengineering*.

KEYWORDS

Start-ups, Software Reuse, Technical Debt, Exploratory Study

ACM Reference Format:
Mercy Njima. 2019. Architecting for Scale: The Case for Systematic Software Reuse in Managing Technical Debt in Start-ups. In *European Conference on Software Architecture (ECSA), September 9–13, 2019, Paris, France.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3344948.3344950

1 INTRODUCTION

Software start-ups are small companies created to develop and market an innovative and software-intensive product with the aim to benefit from economies of scale [17]. Start-ups typically develop early software versions to test and validate emerging ideas to avoid wasteful implementation of complicated software which may be unsuccessful in the market [31]. They aim to spend as little time as possible on activities that have an uncertain contribution to customer value [15]. Under these conditions, often the extra effort required to design and implement software with optimal design is considered unaffordable and a potential waste of time and effort [28].

Start-ups often make sub-optimal design decisions to allow them to get to market quickly and this strategy allows the start-up to put their product in users hands faster, get feedback, and improve. They build non-traditional business architectures by taking the easy path to find a product-market fit and thus accumulate large amounts of technical debt. Technical debt refers to the accumulated backlog of software development needed because developers favour a quick solution over a more complete solution, usually to reduce the overall implementation time [11]. These shortcuts and workarounds are often made by start-ups in their effort to bring their product to market as quickly as possible and to validate their ideas about their target market and the problem they are trying to solve [10, 29]. When the developed product or service is successful, then start-up focuses on maintaining and modifying the software to meet the user needs by adding new features. Scaling-up the system may become an obstacle, which will prevent the company from gaining new customers [30]. As the number of features increases so does the complexity of their development and maintenance, hence systematic reuse and effective technical management become necessary.

2 PREVIOUS AND RELATED WORK

Most research on technical debt has been focused on mature software teams say in large organizations [20]. While technical debt is as important to software start-ups as it is to mature companies, the kind of decisions to make and the consequences of making the wrong decisions are not the same, justifying research on technical debt in software start-ups [30]. In 2016, Unterkalmsteiner et al. published a research agenda focusing on software engineering in start-ups, identifying research questions in the areas of supporting start-up engineering activities, start-up evolution models and patterns and methodologies and theories for start-up research [30]. The authors answer several research questions related to technical debt that help clarify the role of design decisions in software development in start-ups.

Gralha et al. present a study that characterises the evolution of requirement practices in software start-ups along six interrelated dimensions including technical debt and product quality [11]. Chicote introduces a technique for managing technical debt based on visual thinking [4]. The technique addresses the problem of knowing how much debt is in place and how it is affecting the development cycle. This work proposes a cognitively light weight approach to managing technical debt. Yli-Hummo et al. investigated one middle-size Finnish software company with two independent

ECSA, September 9–13, 2019, Paris, France

ACM ISBN 978-1-4503-7142-1/19/09…$15.00
https://doi.org/10.1145/3344948.3344950

product lines to understand the sources and effects of technical debt [32]. Besker et al. present the results of an exploratory study and provide a set of recommendations and a first strategy which can be used by software startups to support their decisions related to the accumulation and refactoring of technical debt [2]. Klotins et al. report on how technical debt is estimated and how start-ups use technical debt as a leverage [16]. They found that technical debt peaks at the growth phase.

In our exploratory study at two start-ups we found that technical debt peaks at the stabilization and growth phases. In the stabilization phase, a start-up gradually finds its way of working and the amount of uncertainty decreases. In the growth phase, the start-up goes through a series of changes and needs to restructure itself to support growth in all areas: business, personnel, the scope of the product, etc [5, 7]. At this stage the company should work on how the systems can evolve later. When the products evolve, features need to be added, removed, turned into modules or separate products [6]. This requires attention to the architecture and implementation. Software product lines encourage an organization to reuse existing assets and capabilities rather than repeatedly developing them for new systems. Therefore, they use architectures, designs and implementation techniques that make features modular and independent so that they are easy to reuse, remove, turn into modules or separate products. We propose and assess the viability of adopting systematic reuse by migrating single products to product lines as an approach to manage technical debt [23].

There exists a significant body of knowledge on the transitioning to software product line for large companies. Moreover, some initial investigations have been conducted on small and medium sized companies and almost none for start-ups.

Heymans et al. present the project PLENTY whose aim was to adapt software product lines engineering for small and medium sized companies which can not support huge investment and risks [14]. They list some specific problems associated with the adoption of software product lines in small and medium companies and propose suggestions to prevent these problems.

Gacek et al. present the experience of a small company in Germany as they adopted and used software product lines to help their business grow successfully into a new market area. The case study reports on the history of the company, how it came to adopt product lines as its prominent development strategy, the role of the key individuals involved, and how they met and overcame various technical challenges [8].

Hanssen et al. presented a case study of a software product company that had successfully integrated practices from software product line engineering and agile software development [12]. They show how practices from the two fields support the company's strategic and tactical ambitions. The findings from this study are relevant to software product companies seeking ways to balance agility and product management.

Hetrick et al. presented their experience of incremental transitioning to software product line engineering to bypass the typical up-front adoption barrier of high costs [13]. The company made a small initial investment and were able to start a chain reaction in which the tactical and strategic incremental returns quickly outpaced the incremental investments, making the transition pay for itself.

Bastos et al. reported on a multi-method study, where results from a mapping study, industrial case study and also expert opinion survey were considered to investigate software product line engineering adoption in the context of small and medium sized enterprises. The study documented evidence observed during the transition from single-system development to a software product line engineering approach [1].

In the "Software Variant building" project, Knauber et al. transfered product line concepts to six small- and medium-sized enterprises by applying the Product Line Software Engineering method, PuLSE [18]. They report on the experience and lessons learned from 24 months of work including first results within the companies.

Nazar et al. used an ethnographic approach at a small company in China to investigate how well the software product line approach would work for a small organization [22]. They addressed the challenges faced by the company and suggested potential improvements to ensure that the company reaped the benefits of software product line engineering.

Ghanam et al. provide a comprehensive taxonomy of the challenges faced when a medium-scale organization decided to adopt software platforms in order to achieve high levels of reuse and improve quality [9]. They report on the need to investigate the issues and challenges organizations face when transitioning to a software platform strategy. The work reveals how new trends in software engineering (i.e. agile methods, distributed development and flat management structures) inter played with the chosen platform strategy.

From the overview of the related work, there has not been work done to propose migration to software product line engineering as a suitable approach for managing technical debt.

3 WORK PLAN

The goal of this work is to support start-ups and scale-ups in finding a viable trade-off between customer demands and long term goals through three interlinked work packages. Table 1 below summarizes the objectives of each work package.

Due to the nascent nature of software start-up research area, exploratory cases studies are suitable approaches to utilize [30]. One the main reasons is that the analytical research paradigm is not sufficient for investigating such issues [1]. While a large gap between research and practice presents many opportunities, it also brings many potential pitfalls. Without a mature body of theoretical knowledge to use as a guiding light, systematic research becomes somewhat more difficult.

3.1 Work Package 1: Software Product Lines

In order to evaluate the start-ups potential for software product line adoption, we conducted in-depth semi-structured interviews to identify the strategies applied in relation to reuse and in general the perception towards software product line engineering practices. We followed the guidelines presented in [25, 26]

We targeted software start-ups that develop a software intensive product or service and are adopting the latest technology trends to create a competitive advantage in the market. In the first stage of

GOAL: Support start-ups and scale-ups in finding a viable trade-off between customer demands and long term goals.	
Work package 1 - Product Lines	A survey of the impediments for adopting product-lines in start-ups.
Work package 2 - Technical Debt	A survey of the state-of-the-practice of dealing with technical debt in start-ups
Work package 3 - Decision Framework	A framework that supports decision making as a start-up evolves. Will provide a list of indicators assessing current performance, areas for improvement and opportunities for growth

Table 1: Research goal and objectives

the study, we established contact with individuals in the two start-ups to request their participation in this study. To form a broad view of a start-up and its processes, we interviewed individuals from different teams and at different ranks / roles from two start-ups based in two different countries.

We created an interview guide based on the software reuse concepts. All the interviews were face-to-face interviews and followed an open-ended interview style. At the beginning of each interview, we started with a short introduction to familiarize the interviewees with the scope of the research. The introduction carried more depth than the introductory e-mails, and the interviewees had a possibility to ask for more details. The interviews were two hours long. All the interviews were recorded with a digital voice recorder, so that any of the information collected would be captured and only correct information would be used when analyzing the results.

This study was conducted in light of the following research questions:

(1) *To what extent is the migration to software product line engineering useful in practice for start-ups?*
(2) *What are the major challenges that would be faced while adopting software product lines in a start-up?*
(3) *How do we evaluate a start-ups readiness for the adoption of software product line engineering approaches?*

To analyse the results of the interviews, we started by reading notes and transcribing the recordings of the interviews. We developed and analyzed the ideas arising from the concepts in the collected data. Further, we used our knowledge and experience in the software product line domain to further strengthen our analysis. We thoroughly checked the start-ups' websites in order to extract useful information and to verify whether we had missed anything. In the end, we compared our findings to the existing literature in order to identify similarities and differences. The results were published at the 13th International Workshop on Variability Modelling of Software-Intensive Systems, (VAMOS 2019) [23].

3.2 Work Package 2: Technical Debt

The goal of this work package is to understand how software start-ups reason about and manage technical debt. In addition, we are interested in the factors that impact technical debt and the potential benefits and challenges of technical debt. We, therefore, aim at answering the following research questions:

(1) *What factors influence the accumulation and management of technical debt in software start-ups?*
(2) *What are the challenges and benefits of technical debt for software start-ups?*
(3) *How do start-ups manage technical debt and when does that become a priority?*

To expand our study and deal with limitations related to our small sample of start-ups investigated, we are conducting expert interviews at another set of start-ups. Given the lack of sufficient and relevant research addressing software product line engineering and technical debt in start-ups, we will conduct expert interviews which are useful research approach to uncover knowledge [3]. The expert interview is a method of qualitative empirical research designed to explore expert knowledge [21]. In this case, start-up founders, product managers and engineers are the experts given their knowledge and experiences which result from the actions, responsibilities, obligations of their specific functions within a start-up.

3.3 Work Package 3: Decision Framework

Due to their dynamic nature, software start-ups must constantly make crucial decisions on whether to change direction or stay on the chosen course [30]. These decisions are known as pivot or persevere in the terms of Lean Startup [24]. Our decision framework will be a support tool to recommend software engineering practices that promote an optimal balance between short term demands and long term goals.

The output from the expert interviews will be transcribed in full and analyzed by means of grounded theory [19, 27]. The goal of grounded theory is to generate theory rather than test or validate existing theory. Using the theories and concepts arising we will build a decision framework for start-ups to help answer the following questions:

(1) *When is the optimal time for a start-up to manage technical debt?*
(2) *What software engineering techniques are useful during different phases of a start-up's life-cycle?*
(3) *To what extent are software engineering goals and business goals connected in customer value creation?*
(4) *What potential opportunities for growth exist in developing products with a long term view?*

4 EXPECTED RESULTS

Over time, we hope that our research results will enable the technical founders of future start-ups to improve the processes they use to

create their products, bringing tried and tested software engineering disciplines to their product development without significant financial or schedule overhead.

We hope that the doctoral symposium will create an opportunity for us to receive feedback on the effectiveness of our research methods, give suggestions for the questions to expand our interview guide and references to related work on developing out decision framework.

ACKNOWLEDGMENTS

This work is supported by Flanders Make vzw, the strategic research centre for the manufacturing industry.

REFERENCES

[1] Jonatas Ferreira Bastos, Paulo Anselmo da Mota Silveira Neto, Pdraig OLeary, Eduardo Santana de Almeida, and Silvio Romero de Lemos Meira. 2017. Software Product Lines Adoption in Small Organizations. *J. Syst. Softw.* 131, C (Sept. 2017), 112–128. https://doi.org/10.1016/j.jss.2017.05.052

[2] T. Besker, A. Martini, R. Edirisooriya Lokuge, K. Blincoe, and J. Bosch. 2018. Embracing Technical Debt, from a Startup Company Perspective. In *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 415–425. https://doi.org/10.1109/ICSME.2018.00051

[3] Alexander Bogner and Wolfgang Menz. 2009. *The Theory-Generating Expert Interview: Epistemological Interest, Forms of Knowledge, Interaction.* Palgrave Macmillan UK, London, 43–80. https://doi.org/10.1057/9780230244276_3

[4] Marcos Chicote. 2017. Startups and Technical Debt: Managing Technical Debt with Visual Thinking. In *Proceedings of the 1st International Workshop on Software Engineering for Startups (SoftStart '17)*. IEEE Press, Piscataway, NJ, USA, 10–11. https://doi.org/10.1109/SoftStart.2017...6

[5] M. Crowne. 2002. Why software product startups fail and what to do about it. Evolution of software product development in startup companies. In *Engineering Management Conference, 2002. IEMC '02. 2002 IEEE International.* 338–343 vol.1.

[6] Anuradha Dande, Veli-Pekka Eloranta, Antti-Jussi Kovalainen, Timo Lehtonen, Marko Leppänen, Taru Salmimaa, Mahbubul Sayeed, Matti Vuori, Claude Rubattel, Wolfgang Weck, et al. 2014. Software Startup Patterns-An Empirical Study. *Tampereen teknillinen yliopisto. Tietotekniikan laitos. Raportti-Tampere University of Technology. Department of Pervasive Computing. Report; 4* (2014).

[7] Veli-Pekka Eloranta. 2014. Towards a Pattern Language for Software Start-ups. In *Proceedings of the 19th European Conference on Pattern Languages of Programs (EuroPLoP '14)*. ACM, New York, NY, USA, Article 24, 11 pages.

[8] Cristina Gacek, Peter Knauber, Klaus Schmid, and Paul Clements. 2001. Successful Software Product Line Develop ment in a Small Organization. (01 2001).

[9] Yaser Ghanam, Frank Maurer, and Pekka Abrahamsson. 2012. Making the leap to a software platform strategy: Issues and challenges. *Information and Software Technology* 54, 9 (2012), 968 – 984. https://doi.org/10.1016/j.infsof.2012.03.005

[10] C. Giardino, N. Paternoster, M. Unterkalmsteiner, T. Gorschek, and P. Abrahamsson. 2016. Software Development in Startup Companies: The Greenfield Startup Model. *IEEE Transactions on Software Engineering* 42, 6 (June 2016), 585–604. https://doi.org/10.1109/TSE.2015.2509970

[11] Catarina Gralha, Daniela Damian, Anthony I. (Tony) Wasserman, Miguel Goulão, and João Araújo. 2018. The evolution of requirements practices in software startups. 823–833. https://doi.org/10.1145/3180155.3180158

[12] Geir K. Hanssen and Tor E. Fígri. 2008. Process Fusion: An Industrial Case Study on Agile Software Product Line Engineering. *J. Syst. Softw.* 81, 6 (June 2008), 843–854. https://doi.org/10.1016/j.jss.2007.10.025

[13] William A. Hetrick, Charles W. Krueger, and Joseph G. Moore. 2006. Incremental Return on Incremental Investment: Engenio's Transition to Software Product Line Practice. In *Companion to the 21st ACM SIGPLAN Symposium on Object-oriented Programming Systems, Languages, and Applications (OOPSLA '06)*. ACM, New York, NY, USA, 798–804. https://doi.org/10.1145/1176617.1176726

[14] Patrick Heymans and Jean-Christophe Trigaux. 2004. Software Product Lines : State of the art.

[15] Eriks Klotins. 2018. Software start-ups through an empirical lens: are start-ups snowflakes?. In *Proceedings of the International Workshop on Software-intensive Business: Start-ups, Ecosystems and Platforms (SiBW 2018), Espoo, Finland, December 3, 2018.* 1–14. http://ceur-ws.org/Vol-2305/paper01.pdf

[16] Eriks Klotins, Michael Unterkalmsteiner, Panagiota Chatzipetrou, Tony Gorschek, Rafael Prikladnicki, Nirnaya Tripathi, and Leandro Bento Pompermaier. 2018. Exploration of Technical Debt in Start-ups. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP '18)*. ACM, New York, NY, USA, 75–84. https://doi.org/10.1145/3183519.3183539

[17] Eriks Klotins, Michael Unterkalmsteiner, Panagiota Chatzipetrou, Tony Gorschek, Rafael Prikladniki, Nirnaya Tripathi, and Leandro Pompermaier. 2019. A progression model of software engineering goals, challenges, and practices in start-ups. *IEEE Transactions on Software Engineering* PP (02 2019), 1–1. https://doi.org/10.1109/TSE.2019.2900213

[18] Peter Knauber, Dirk Muthig, Klaus Schmid, and Tanya Widen. 2000. Applying Product Line Concepts in Small and Medium-Sized Companies. *IEEE Software* 17, 5 (2000), 88–95.

[19] Anselm L. Strauss and Juliet M. Corbin. 1997. *Grounded Theory in Practice.* Vol. 28. 296 pages. https://doi.org/10.2307/2655359

[20] Zengyang Li, Paris Avgeriou, and Peng Liang. 2015. A Systematic Mapping Study on Technical Debt and Its Management. *J. Syst. Softw.* 101, C (March 2015), 193–220. https://doi.org/10.1016/j.jss.2014.12.027

[21] Michael Meuser and Ulrike Nagel. 2009. *The Expert Interview and Changes in Knowledge Production.* Palgrave Macmillan UK, London, 17–42. https://doi.org/10.1057/9780230244276_2

[22] Najam Nazar and T M. J. Rakotomahefa. 2016. Analysis of a Small Company for Software Product Line Adoption — An Industrial Case Study. *International Journal of Computer Theory and Engineering* 8 (08 2016), 313–322. https://doi.org/10.7763/IJCTE.2016.V8.1064

[23] Mercy Njima and Serge Demeyer. 2019. An Exploratory Study on Migrating Single-Products towards Product Lines in Startup Contexts. In *Proceedings of the 13th International Workshop on Variability Modelling of Software-Intensive Systems, VAMOS 2019, Leuven, Belgium, February 06-08, 2019.* 10:1–10:6. https://doi.org/10.1145/3302333.3302347

[24] Erich Ries. 2011. *The lean startup : how today's entrepreneurs use continuous innovation to create radically successful businesses.* Crown Business, New York.

[25] Per Runeson and Martin Höst. 2009. Guidelines for Conducting and Reporting Case Study Research in Software Engineering. *Empirical Softw. Engg.* 14, 2 (April 2009), 131–164. https://doi.org/10.1007/s10664-008-9102-8

[26] Forrest Shull, Janice Singer, and Dag I.K. Sjøberg. 2007. *Guide to Advanced Empirical Software Engineering.* Springer-Verlag, Berlin, Heidelberg.

[27] Klaas-Jan Stol, Paul Ralph, and Brian Fitzgerald. 2016. Grounded Theory in Software Engineering Research: A Critical Review and Guidelines. In *Proceedings of the 38th International Conference on Software Engineering (ICSE '16)*. ACM, New York, NY, USA, 120–131. https://doi.org/10.1145/2884781.2884833

[28] Henri Terho, Sampo Suonsyrjä, and Kari Systä. 2016. The Developers Dilemma: Perfect Product Development or Fast Business Validation?. In *Product-Focused Software Process Improvement*, Pekka Abrahamsson, Andreas Jedlitschka, Anh Nguyen Duc, Michael Felderer, Sousuke Amasaki, and Tommi Mikkonen (Eds.). Springer International Publishing, Cham, 571–579.

[29] Edith Tom, Aybüke Aurum, and Richard Vidgen. 2013. An exploration of technical debt. *Journal of Systems and Software* 86 (06 2013), 1498–1516. https://doi.org/10.1016/j.jss.2012.12.052

[30] Michael Unterkalmsteiner, Pekka Abrahamsson, Xiaofeng Wang, Anh Nguyen-Duc, Syed M. Ali Shah, Sohaib Shahid Bajwa, Guido H. Baltes, Kieran Conboy, Eoin Cullina, Denis Dennehy, Henry Edison, Carlos Fernández-Sánchez, Juan Garbajosa, Tony Gorschek, Eriks Klotins, Laura Hokkanen, Fabio Kon, Ilaria Lunesu, Michele Marchesi, Lorraine Morgan, Markku Oivo, Christoph Selig, Pertti Seppänen, Roger Sweetman, Pasi Tyrväinen, Christina Ungerer, and Agustín Yagüe. 2016. Software Startups - A Research Agenda. *e-Informatica* 10, 1 (2016), 89–124. https://doi.org/10.5277/e-Inf160105

[31] Muhammad Waseem and Naveed Ikram. 2016. Architecting Activities Evolution and Emergence in Agile Software Development: An Empirical Investigation - Initial Research Proposal. In *Agile Processes, in Software Engineering, and Extreme Programming - 17th International Conference, XP 2016, Edinburgh, UK, May 24-27, 2016, Proceedings.* 326–332. https://doi.org/10.1007/978-3-319-33515-5_35

[32] Jesse Yli-Huumo, Andrey Maglyas, and Kari Smolander. 2014. *The Sources and Approaches to Management of Technical Debt: A Case Study of Two Product Lines in a Middle-Size Finnish Software Company.* Springer International Publishing, Cham, 93–107. https://doi.org/10.1007/978-3-319-13835-0_7