Value-Based Technical Debt Management: An Exploratory Case Study in Start-Ups and Scale-Ups

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

Serge Demeyer
University of Antwerp and Flanders Make
Antwerp, Belgium
serge.demeyer@uantwerpen.be

ABSTRACT

Software start-ups face fierce competition in the market forcing them to release their products quickly and often under tough time constraints. To meet their deadlines, start-ups take shortcuts in software development leading to the accumulation of technical debt. They are able to put their product in users hands faster, get feedback, and improve at the expense of quality issues in the long run. As a start-up evolves through inception, stabilization and growth this debt will have to be managed. Technical debt management and software product line engineering techniques have some similar benefits of increased productivity and reduced time-to-market. Our aim is to check whether software product line engineering can be a candidate technique for start-ups to employ in managing technical debt as a response to their life-cycle phase goals and challenges. We conducted expert interviews with nine start-up professionals to identify the strategies applied in relation to technical debt management and software product lines engineering and other software engineering practices in start-ups. By analyzing the responses from the interviews we found that depending on the life-cycle phase of the start-up software product line engineering proved effective in managing technical debt and helped the start-ups to advance through the life-cycle phases.

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, Scale-ups, Technical Debt, Software Reuse, Exploratory Study, Software Product Lines

ACM Reference Format:
Mercy Njima and Serge Demeyer. 2019. Value-Based Technical Debt Management: An Exploratory Case Study in Start-Ups and Scale-Ups. In *Proceedings of the 2nd ACM SIGSOFT International Workshop on Software-Intensive Business: Start-ups, Platforms and Ecosystems (IWSiB '19), August 26, 2019, Tallinn, Estonia.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3340481.3342739

1 INTRODUCTION

Software start-ups are freshly created companies with no operating history and mainly oriented towards developing high-tech and innovative products, aiming to grow their business in highly scalable markets [34]. Start-ups search for product-market fit by building fast, testing and iterating until they find a solution. In the process of releasing a product, the development team knows that it has released software with flaws that must be fixed at some point in the future [10]. Balancing the choice of releasing poor-quality software early or high-quality software late is challenging. This leads start-ups to an awkward situation where they have to decide what quality is acceptable and what compromises in the development process they have to take [40, 42]. This is the basis of how start-ups accumulate technical debt. Technical debt is a metaphor reflecting technical compromises that can yield short-term benefit but may hurt the long-term maintainability and evolvability of a software system [9, 30]. A start-up may also accrue technical debt as a result of inexperienced developers who gain knowledge and experience during development [39].

The aims, objectives, and challenges of product engineering change quickly through the different phases of a start-up's life-cycle [8]. A miscalculation in choosing engineering practices could lead to over or under-engineering of the product, and contribute to wasted resources and missed market opportunities [18, 24].

Figure 1 presents the stages of a start-up's life-cycle identified as inception phase, stabilization phase and growth phase [8, 13]. The inception phase is the period between product conception and the first sale where the start-up is searching for the problem-solution fit. The stabilization phase begins when the first customer takes delivery of the product. Although the amount of uncertainty decreases, unresolved issues from the inception phase will have a growing impact on the company, and become harder to deal with. 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. The road map for future product development becomes clear but existing issues can significantly affect the product development capability of the company [8]. During the stabilization phase most start-ups are considered scale-ups.

When the products evolve, features need to be added, removed, turned into modules or separate products [11]. As the evolving product continually changes its complexity, reflecting deteriorating structure, increases unless work is done to maintain or reduce it [29]. Herein lies the opportunity to adopt software product line engineering. It has been suggested that start-ups can not afford the upfront costs of adopting software product line engineering nor the tedious planning that run counter to start-ups' lean culture [14]. There is evidence that the majority of small, especially very small software organizations, are not adopting existing software engineering standards as they perceive them as being orientated towards large organizations and studies have shown that small firms' negative perceptions of process model standards are primarily driven by negative views of cost, documentation and bureaucracy [28, 37]. Time to market and economy of production are two of software product line engineering best assets, and both are many times more critical for a start-up [17]. Start-ups report on attempts to leverage on cutting-edge technologies with the aim to gain a competitive advantage such as faster time-to-market or ease in creating additional features [41].

With this study, we aim to understand how start-ups manage their technical debt and whether software product line engineering would be a useful approach. We use in-depth semi-structured interviews and expert interviews to collect and analyze primary data on engineering practices in six start-up cases. The rest of the paper is organized as follows: In Section 2, background information and related work is provided. In Section 3, the details of the study are discussed. In Section 4, we discuss our findings. Finally, we present the conclusions in Section 5.

2 BACKGROUND AND RELATED WORK

Most research on technical debt has been focused on mature software teams say in large organizations [30]. 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 [41]. 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 [41]. 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 [19]. Chicote introduces a technique for managing technical debt based on visual thinking [6]. 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 product lines to understand the sources and effects of technical debt [42]. 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 start-ups to support their decisions related to the accumulation and refactoring of technical debt [3]. Klotins et al. report on how technical debt is estimated and how start-ups use technical debt as a leverage [23]. They found that technical debt peaks at the growth phase.

In the stabilization and growth phases a start-up 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 [11]. 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 [26]. 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 [33].

There exists a significant body of knowledge on the transitioning to software product line engineering 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 [22]. 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 [14].

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 [20]. 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 [21]. 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 [2].

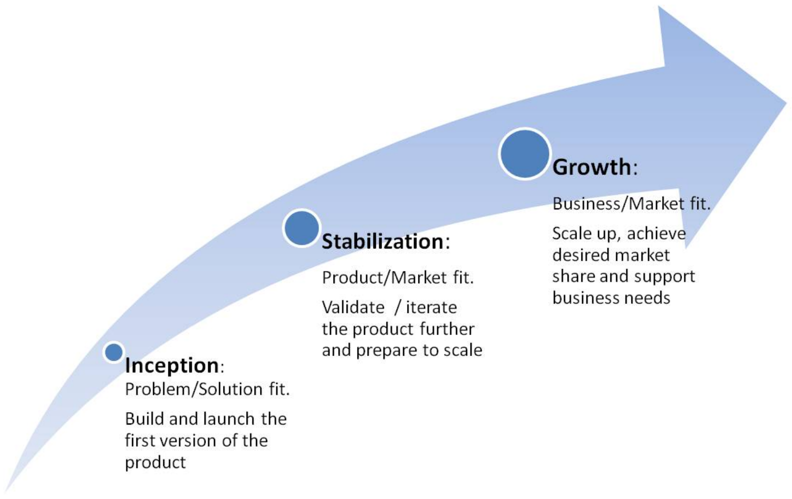
Inception:
Problem/Solution fit.
Build and launch the first version of the product
Stabilization:
Product/Market fit.
Validate / iterate the product further and prepare to scale
Growth:
Business/Market fit.
Scale up, achieve desired market share and support business needs

Figure 1: Start-up life-cycle phases based on Crowne [8].

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 [25]. 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 [32]. 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 [16]. 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.

Boehm et al. present some of the major elements of a value-based software engineering agenda comprising of the value considerations to be integrated into the existing and emerging software engineering principles and practices in a company [5]. Since transitioning to software product line engineering techniques promises increased value and reduction of costs for companies, it may also prove to be a value-based technical debt management approach.

From the overview of the related work, we deduce that it remains largely unknown to what extent product line engineering is a suitable approach for managing technical debt in start-ups. There have been a few success stories, nevertheless the heavy weight approach associated with product lines is seen as a major obstacle. Hence the need for a qualitative study providing insights in barriers, best practices, and experiences.

3 RESEARCH DESIGN

In order to evaluate the start-ups potential for software product line adoption, we conducted in-depth semi-structured interviews and expert interviews to identify the strategies applied in relation to technical debt management, software reuse and in general the perception towards software product line engineering practices. We followed the guidelines presented in [36, 38]

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 the study, we established contact with individuals in the start-ups to request their participation in this study. Specifically, we sent personalized emails to the individuals describing the scope and objectives of the study. We asked for interviewees with experience in software development and product management processes. All of the participants in this study were volunteers, and no compensation of any form was offered or paid. We informed the participants that the assessment being conducted was part of a Ph.D. research project and that neither the identity of an individual nor of the start-up would be disclosed in the resulting thesis or in any research publications. In order to protect the privacy of the start-ups, they will be referred to using letters. To form a broad view of a start-up and its processes, we interviewed individuals from different teams and at different ranks / roles, see Table 1.

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. 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. After the interviews, the recordings were transcribed.

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?*

Table 1: Study Participants

Role	Company	Country	Segment	Life-cycle Phase
Senior Product Manager Developer	A	Belgium	Data / Telecom	Growth
CEO / Lead Developer Advisor (Business and Technology)	B	Netherlands	Automation / Services	Stabilization
Head of Software Development Software Development Manager	C	Belgium	Health	Stabilization
Co-founder & Chief Product Officer	D	Belgium	Blockchain	Stabilization
Software Architect	E	Belgium	Event Planning / Services	Growth
Founder & CEO	F	Belgium	Education	Stabilization

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

The events in the context of the case study are captured by the perception of the researchers based on observations, interviews and experience. For the analysis, 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 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.

4 FINDINGS AND DISCUSSION

As can be seen in Table 1, all the start-ups in our study have evolved in their life-cycles and are in the stabilization and growth phases. This means that as they are preparing to scale up, they were already facing the unresolved development and technical debt issues accrued during the inception phase. We found that all the start-ups accumulated technical debt during the inception phase and it peaks and should be dealt with during the stabilization and growth phases. We list our findings on the factors that led to the accumulation of technical debt and the related challenges:

(1) **Skills shortage**: The level of engineering skills has a significant effect on the amount of technical debt accumulated and leads to delays in development and poor quality features. This was the case for all of our start-ups at one point or other. For instance, start-up F utilised interns at some point who could not create good enough code. Their work was either not included in the product when it was released to the market or had to be heavily refactored. Start-up B on the other hand, has a social enterprise arm that works with the regional government to employ autistic developers most of whom are self taught. Their work requires considerable peer review and need for additional training.
(2) **Pivoting**: Start-ups may pivot in order to target new markets and user groups. In the Lean Startup approach, Ries defines a pivot as a strategic change, designed to test a fundamental hypothesis about a product, business model or engine of growth [35]. The process of pivoting unearths unresolved development issues that would not have been discovered otherwise. Start-up A for instance, acquired new funding and the investors required a re-brand and pivot. The whole product become a single feature of a much larger product, mainly because the original product was insufficient to address new customer needs. This is called a zoom-out pivot [1]. It lead to a huge refactoring initiative and removal of technical debt along the way in order to increase their scalability.
(3) **Diverse product requirements became unmanageable**: As a software product grows and the customer requirements increase, it naturally becomes more difficult to maintain the products and to deal with technical debt. This was the case at start-up E.
(4) **Development team instability**: Overall team size and the number of people being hired and leaving a start-up amplifies the accumulation of technical debt. In start-up F there was a divide between developers who joined the company early and those who were recruited later. In the process, some developers quit and one of the co-founders left the start-up. A proper hand over process of their workload was not followed and the features that they were working on were not complete.
(5) **Service provision**: Technical sales, product integration and customer support to a new customer base could be a source of new challenges for the product development team especially in coping with a flow of requests for unanticipated features. The team focuses on creating great customer experiences by expediting the delivery of a piece of functionality or a project

which later needs to be refactored. Start-up F has a small cross-functional team and struggles the most among our start-ups in balancing service provision and development.

(6) **Undefined processes**: When there are no defined processes to introduce new products and respond to customer requests, the ad-hoc responses contribute to the accumulation of technical debt. The development team applies Clone and Own techniques to re-purpose the older code base and other times they build the new product features from scratch. During the inception phase all our start-ups had to deal with this dilemma.

(7) **Transition periods**: Sometimes founders hire an external company to build a proof of concept. This happens in situations where the start-up founders do not have a software engineering background. Start-up C built their initial product at a consultancy company and when it was ready the code base was transferred to the start-up. At the same time, the start-up was hiring new developers to take over the task which was very challenging and in the process they accumulated large amounts of technical debt. The new developers had little time to get acquainted with the product before they had to deliver it to a new customers.

Using "clone and own" approaches as the primary indicator for readiness for software product lines we found that all of our start-ups are reusing software artifacts arbitrarily. Clone and own is a common and ad-hoc practice that consists in reusing artifacts of an existing product variant then modifying it to add and/or remove some functionalities in order to derive a new product variant [15, 27]. It is perceived as a favorable reuse approach when a company has to address its target market needs by releasing a new product that is similar, yet not identical, to existing ones [12].

Software product line engineering on the other hand promotes strategic, well-managed software reuse with the aim producing a family of systems rather than individual systems [7]. Start-up A and E who are in the growth phase were already exploring the use of software product line engineering techniques to manage technical debt by exploiting commonalities within their code base when refactoring. They reported that they already had similar products that were not developed from a common platform or as a family of products. Nevertheless, the start-ups in the stabilization phase were hesitant to commit to exploring software product line engineering as they could not see short-term return on investment.

There are three widely recognized ways to adopt a software product line engineering approach: reactive, proactive and incremental. None of them is the best approach in all cases, it depends on the context in which a start-up operates [4, 31]. We observed these approaches in our cases as well.

In the proactive approach the reusable assets are developed prior to any new product development. Start-up A is employing the proactive approach. They have expanded their reach to different markets outside Belgium and the customer requirements were quite similar. The developers were already implementing a common, managed set of features to satisfy the specific needs of any new market segment.

In the reactive approach the reusable assets are harvested from products after they are built and deployed. Start-up E is employing the reactive approach as they found that the product expectations from the customers were very high. They had made promises of new product features to customers but did not deliver as timescales were not met. In addition, the task of maintaining their product variants is increasingly difficult and expensive. They are exploring what avenues exist for them to lower the initial cost of transitioning to software product line engineering.

The incremental approach is a compromise between the proactive and reactive approaches and builds the set of assets in scheduled increments. Start-up C though in the stabilization phase is set to employ the incremental approach. There's demand for a number of new product variants from the market and from the management within the start-up.

Multiple preconditions must be satisfied if an software product line engineering approach is to be profitable, but the most fundamental of these is that the market for the products be predicted accurately [20]. In addition, product management is more complicated due to complex dependencies between product variants and the scope of the platform especially in cases where agile practices are employed.

5 CONCLUSION

In this paper, we investigated how six software start-ups accumulate and manage technical debt and whether software product line engineering approaches can be used to better manage the debt, maintain the product and improve the time-to-market. We conclude that software reuse approaches are relevant for start-ups for maintenance and evolution of products. As seen from our study though, start-ups in the growth phase stand to benefit most from software product line engineering approaches by lowering the overall complexity of their products, increasing the scalability of the product portfolios and enabling them to make the transition to maturity with less time, cost and effort.

ACKNOWLEDGEMENTS

The authors would like to thank all the practitioners who found time and motivation to share their experiences. This work is supported by Flanders Make vzw, the strategic research centre for the manufacturing industry.

REFERENCES

[1] Sohaib Shahid Bajwa, Xiaofeng Wang, Anh Nguven Duc, and Pekka Abrahamsson. 2016. *How Do Software Startups Pivot? Empirical Results from a Multiple Case Study.* Springer International Publishing, Cham, 169–176. https://doi.org/10.1007/978-3-319-40515-5_14

[2] 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

[3] 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

[4] Günter Böckle, Jesús Bermejo Muñoz, Peter Knauber, Charles W. Krueger, Julio Cesar Sampaio do Prado Leite, Frank van der Linden, Linda Northrop, Michael Stark, and David M. Weiss. 2002. Adopting and Institutionalizing a Product Line Culture. In *Software Product Lines*, Gary J. Chastek (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 49–59.

[5] Barry Boehm. 2003. Value-based Software Engineering. *SIGSOFT Softw. Eng. Notes* 28, 2 (March 2003), 4–. https://doi.org/10.1145/638750.638776

[6] 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

[7] Paul C. Clements and Linda Northrop. 2001. *Software Product Lines: Practices and Patterns*. Addison-Wesley.

[8] 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.

[9] Ward Cunningham. 1992. The WyCash Portfolio Management System. In *Addendum to the Proceedings on Object-oriented Programming Systems, Languages, and Applications (Addendum) (OOPSLA '92)*. ACM, New York, NY, USA, 29–30. https://doi.org/10.1145/157709.157715

[10] Bill Curtis, Jay Sappidi, and Alexandra Szynkarski. 2012. Estimating the Size, Cost, and Types of Technical Debt. In *Proceedings of the Third International Workshop on Managing Technical Debt (MTD '12)*. IEEE Press, Piscataway, NJ, USA, 49–53. http://dl.acm.org/citation.cfm?id=2666036.2666045

[11] 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).

[12] Y. Dubinsky, J. Rubin, T. Berger, S. Duszynski, M. Becker, and K. Czarnecki. 2013. An Exploratory Study of Cloning in Industrial Software Product Lines. In *2013 17th European Conference on Software Maintenance and Reengineering*. 25–34. https://doi.org/10.1109/CSMR.2013.13

[13] 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.

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

[15] E. Ghabach, M. Blay-Fornarino, F. E. Khoury, and B. Baz. 2018. Clone-and-Own software product derivation based on developer preferences and cost estimation. In *2018 12th International Conference on Research Challenges in Information Science (RCIS)*. 1–6. https://doi.org/10.1109/RCIS.2018.8406682

[16] 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

[17] 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

[18] C. Giardino, M. Unterkalmsteiner, N. Paternoster, T. Gorschek, and P. Abrahamsson. 2014. What Do We Know about Software Development in Startups? *IEEE Software* 31, 5 (Sept 2014), 28–32. https://doi.org/10.1109/MS.2014.129

[19] 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

[20] 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

[21] 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

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

[23] 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

[24] 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 startups. *IEEE Transactions on Software Engineering* PP (02 2019), 1–1. https://doi.org/10.1109/TSE.2019.2900213

[25] 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.

[26] R. Krut and S. Cohen. 2008. Service-Oriented Architectures and Software Product Lines - Putting Both Together. In *2008 12th International Software Product Line Conference*. 383–383. https://doi.org/10.1109/SPLC.2008.71

[27] Raúl Lapeña, Manuel Ballarin, and Carlos Cetina. 2016. Towards Clone-and-own Support: Locating Relevant Methods in Legacy Products. In *Proceedings of the 20th International Systems and Software Product Line Conference (SPLC '16)*. ACM, New York, NY, USA, 194–203. https://doi.org/10.1145/2934466.2934485

[28] Claude Y. Laporte, Simon Alexandre, and Rory V. O'Connor. 2008. A Software Engineering Lifecycle Standard for Very Small Enterprises. In *Software Process Improvement*, Rory V. O'Connor, Nathan Baddoo, Kari Smolander, and Richard Messnarz (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 129–141.

[29] M. M. Lehman. 1980. Programs, life cycles, and laws of software evolution. *Proc. IEEE* 68, 9 (Sept 1980), 1060–1076. https://doi.org/10.1109/PROC.1980.11805

[30] 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

[31] John McGregor. 2008. Agile Software Product Lines, Deconstructed. *Journal of Object Technology* 7 (2008), 7–19.

[32] 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

[33] 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

[34] Nicolo Paternoster, Carmine Giardino, Michael Unterkalmsteiner, Tony Gorschek, and Pekka Abrahamsson. 2014. Software development in startup companies: A systematic mapping study. 56, 10 (2014), 1200–1218.

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

[36] 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

[37] Mary Sánchez-Gordón, Ricardo Colomo-Palacios, Antonio de Amescua, and Rory O'Connor. 2016. *The Route to Software Process Improvement in Small- and Medium-Sized Enterprises*. 109–136. https://doi.org/10.1007/978-3-319-31545-4-7

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

[39] Marek Grzegorz Stochel, Mariusz R. Wawrowski, and Magdalena Rabiej. 2012. Value-Based Technical Debt Model and Its Application. In *ICSEA 2012*.

[40] Henri Terho, Sampo Suonsyrjä, and Kari Systä. 2016. *The Developers Dilemma: Perfect Product Development or Fast Business Validation?* Springer International Publishing, Cham, 571–579. https://doi.org/10.1007/978-3-319-49094-6_42

[41] 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

[42] 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