Lean Startup Meets Software Product Lines: Survival of the Fittest or Letting Products Bloom?

Henri Terho[1], Sampo Suonsyrjä[1], Ari Jaaksi[2], Tommi Mikkonen[1], Rick Kazman[3], and Hong-Mei Chen[3]

[1] Tampere university of technology, Korkeakoulunkatu 1, FI-33720 Tampere, Finland
henri.terho@tut.fi, sampo.suonsyrja@tut.fi, tommi.mikkonen@tut.fi
[2] Idean Enterprises Ltd., Hämeenkatu 18, FI-33200 Tampere, Finland,
ari.jaaksi@linux.com
[3] University of Hawaii, Honolulu, HI, USA
kazman@hawaii.edu,hmchen@hawaii.edu

Abstract. Typical management strategies proven to work in already established businesses do not work as expected in startups. Startups do not yet have a business model and product that they could focus on, but are still looking for a working business model. Lean Startup is a method for startup management that focuses on quick iteration and on fast learning to find an iterable business model. As a method, Lean Startup is still quite novel. It does not have much scientific literature written about it, but it is used by startups. The two case study companies were both positive about Lean Startup and felt that the method had given them a helpful approach.

Keywords: lean startup, iteration, case study

1 Introduction

New product development and business model evolution are critical competencies for any company. Their value is intensified, however, in the case of startups, where the entire business model can be unclear or at least remain uncertain and untested. To find a fast-track to profitability, a startup needs to streamline and speed up the two vital processes – finding new markets and developing novel products. This requires highly optimized techniques and methods for the management of products [4].

An emerging choice for such a management method is Lean Startup [19, 5]. As the name suggests, the method has emerged from the niche of small companies that are starting up their businesses. Such companies form an interesting field of study, as they are seeking to validate their ideas and products as quickly as possible, but, at the same time, efficient execution of such processes is vital. Lean Startup defines a process for validation, where companies build, measure and learn by creating Minimum Viable Products (MVP) [19]. In the process of

getting to a finalized product, companies might go through a number of different MVPs. Depending on the company, these MVPs might share characteristics, build on common tools and technologies, or aim at the same market.

Efficient, rapid development of new products has long been a desire for most software organizations, and there have been other attempts to manage families of related products, such as Software Product Lines (SPL). Similarly to Lean Startup, SPLs promise increased productivity and reduced time-to-market [24], in the context of SPLs achieved by reusing common core assets for building families of related software products. Although the initial development of reusable software and a common product-line architecture requires additional up-front effort, this effort will later be more than compensated over time. For example, maintenance and evolution of the different products in the SPL can be centrally planned and coherently staged.

As Lean Startup and Software Product Lines have some similar goals, we have decided to analyze their commonalities and differences. After we first consider the background and relationships of these two concepts, we provide an empirical study, where we investigate how two Finnish software startups have implemented Lean Startup in their software development. In particular, we analyze how multiple MVPs developed by these companies compare to an SPL, and whether we could use the established body of knowledge regarding SPLs to analyze Lean Startup's MVP development practices to avoid some of its known risks.

The main research questions we have formulated for this paper are listed as follows:

RQ1: What parallels can be drawn between SPLs and Lean Startup?
RQ2: How did the case companies use the Lean Startup method in their software development?
RQ3: What kind of similarities are there between the outcomes experienced by the case companies and can we relate these to the similarities (and differences) between SPLs and Lean Startup?

The rest of this paper is structured as follows. In Section 2 we go through the theory of SPLs and Lean Startup. We also place particular focus on the MVP aspect of the Lean Startup approach. In Section 3 we go through our research approach and case study companies. In Section 4 we go through the case study results. In Section 6 we provide an overview of lessons we have learned in the process. In section 5 we take a look at the validity and reliability of the study. Finally, in Section 7 we summarize the paper by drawing final conclusions.

2 Background

2.1 Software Product Lines

A Software Product Line (SPL) is a systematic way to share a common set of core assets used in a series of related products, targeted for a certain market or

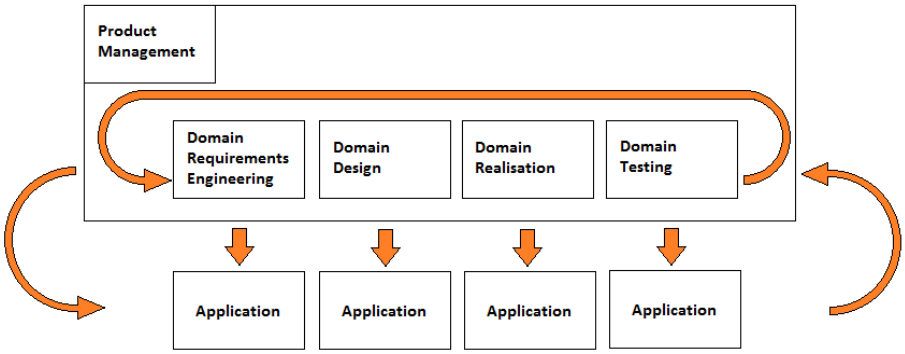

Fig. 1. Software Product Lines

for a specific mission [6]. In technical terms, the creation of an SPL culminates in the creation of an infrastructure that allows rapid, organized production of similar software systems [9]. Such an approach has proven to be efficient for both architecture and component-level reuse [24]. Moreover, an SPL can be seen to include a broad field of different subjects ranging from business to architecture and from processes to organizations [12].

A major goal of an SPLs is strategic reuse: managers, analysts, architects, developers, and testers can avoid performing the same activities over and over again by reusing existing (parameterized, tailorable) assets. SPLs have been regarded as a methodology for developing software products and software-intensive systems in short time and with higher quality [21]. This also allows companies to produce products that closely related to each other with lower cost and higher quality (for example, reduced rates of defects) [11].

There are two principal ways to develop an SPL, and they balance their risks and the aforementioned benefits somewhat differently:

- A proactive reuse-based approach may be adopted in cases where the risk of developing possibly useless assets is accepted. As the shared assets of an SPL are developed before they are used in products, an up-front investment is obviously required. This requires added work in software asset management and introduces the risk of increased time to market for the first products [10]. This approach is typically taken in more mature domains, where the company already has experience in creating similar products and has the expectation of creating many such products in the future.
- A reactive approach—where assets are created and made general on an as-needed basis, and in an evolutionary fashion—can significantly reduce this up-front cost, but at the same time it requires closer coordination within the SPL project [24]. Moreover, this can also lead to a shorter time to market but at the expense of greater levels of re-work and waste.

Typically SPL product development is divided into two distinct software development processes: the domain and application development processes. The domain process focuses on creating shared, reusable software artifacts for the different applications created in the applications process. The applications build on the domain assets and add functionality, as needed, that differentiates the applications one from another. This is outlined in Figure 1. The nature of the software artifacts in the different application instances is constantly evaluated, and if recurring artifacts are discovered in those applications they are integrated into domain engineering. The split between domain and application development requires domain expertise from the software developers and architects who evaluate what should be part of the domain. [11]

Despite all the advantages of using an SPL, there are some risks and issues related to them as well. For example, [14] describes several challenges, especially for smaller companies. Smaller companies usually have more limited resources for creating holistic product lines and have difficulties in affording full-scale platform development and maintenance. Moreover, their environments are often highly volatile, which increases the risk that the costs of creating the product line exceeds the benefits. In addition, the difficulty of using objective methods without historical data leaves these companies relying on their personal opinions as to whether a given reuse strategy is actually cost-effective.

2.2 Lean Startup

Lean Startup is a model created by Eric Ries and Steve Blank for the development of startups [19] [5], building on the idea that the goal of a startup is to transform new ideas into products. An iterative cycle of building, measuring, and learning is proposed. First, an idea is turned into a piece of software. When customers interact with the produced software, they generate feedback, typically both qualitative and quantitative. Based on this feedback, the startup learns more about their business space, the performance of their designs, and hence the acceptance of their products. In the practical implementation of these cycles, each cycle is typically linked with its own MVP, which is used to test the hypothesis of the current cycle.

The iterative cycle is run as follows (Figure 2). A company initially enters the loop in the ideas state. This means that one has an assumption: a hypothesis of a business plan that is being refined into the first product. This first assumption is called a leap-of-faith assumption, which is based on data outside of the build-measure-learn cycle. The original leap-of-faith assumption is one of the most critical points, but the Lean Startup method provides no way to test it beforehand. The problem should be assessed with customer interviews or other methods for determining initial feasibility. This is then fed into the build-measure-learn as an initial assumption and iterated upon to reach a valid product hypothesis.

The build phase is the transition from the ideas state to the code state. During this phase a version of the product is built, based on the most recent ideas. This product is an MVP, meaning a product that only contains the minimum

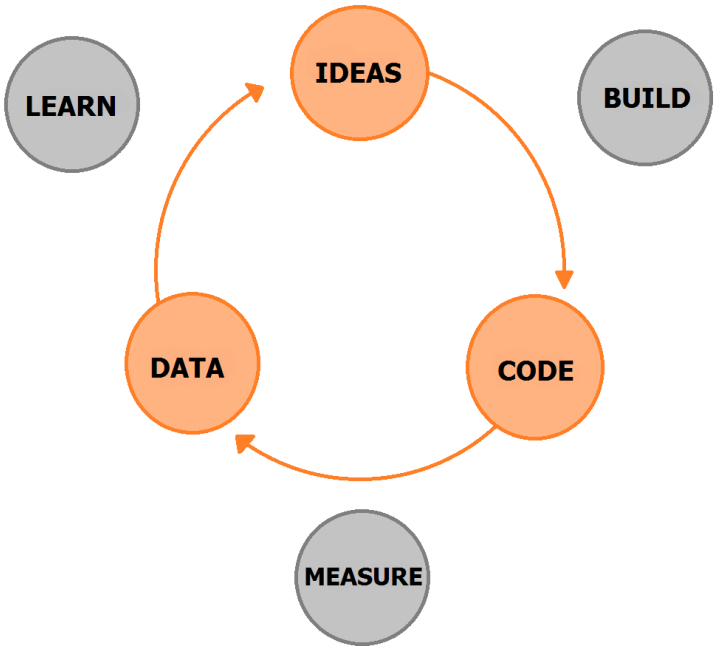

Fig. 2. Build-measure-learn loop

amount of features to make it viable with the minimum amount of work. The goal is to maximize the learning gained from multiple iterations through the build-measure-learn loop.

In the code state, startups have a ready product version of their program. The code is not totally ready—for example, it may not be robust and may not cover a broad range of exceptional conditions—but it fulfills the goals to test the current hypothesis. This is a minimum viable product version. The software should also include analytics code to enable the collection of data about customer behavior, or there may even be two versions of the MVP, to enable A/B testing of the product.

The measure phase is where the product is deployed to the customers. The product is used by real customers and data about their behaviour using the product is collected by the analytics code in the program.

The data state is where the startup has collected data from the usage of the MVP. The purpose of this data is to decide whether product development efforts have led to progress. The data collected from the product should be sufficiently concrete and sufficiently connected to the desired customer outcomes so that it can be immediately acted upon.

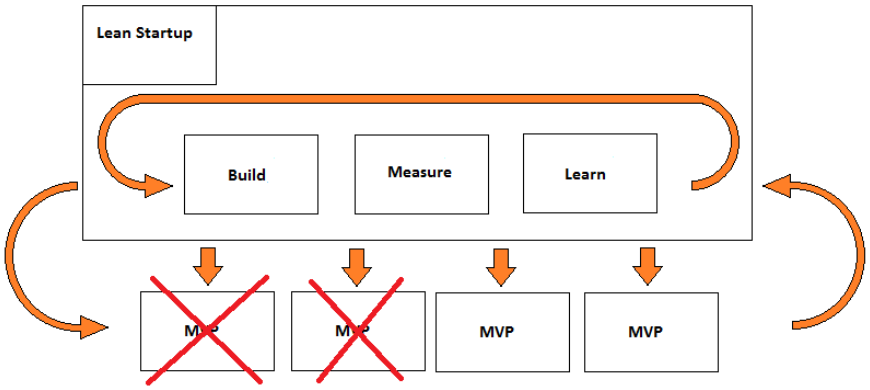

Fig. 3. Lean Startup process

After this comes the learning phase. In this phase the startup compares the data collected to their original product hypothesis and assesses whether learning milestones from the hypotheses have been fulfilled. The data can be used to see if the customer behavior matched the startup's expectations and if the changes made in this iteration have improved the software. For example, one might learn whether the new landing page has increased the amount of subscriptions. If the hypothesis was successful, a new hypothesis should be formulated to further improve the software, ending up back in the ideas phase.

In Lean Startup software development, multiple MVPs are created to help find the optimal business model for the company and to find information about the business space the company operates in. Typically multiple MVP versions are done during the lifecycle of the company, as multiple turns of the build-measure-learn loop are iterated. This iterative process is outlined in Figure 3. These MVPs and its subset minimum viable features(MVF) are then used to iterate the different aspects of the startup. During this iterative process multiple MVPs and MVFs are created and evaluated, and further MVPs and MVFs can be built upon the older versions with increasing efficiency.

An MVP might not be used just for finding the optimal business model for the company. The minimum viable product is defined as a version of the product that enables a full turn of the build-measure-learn loop with minimum amount of effort and the least amount of development time. It should contain the features that realize the software solution's unique value proposition and little else, except for logging and metrics integration. The idea is to cut out all non-essential features and leave just the core features of your application. In the same way a minimum viable feature (MVF) is a single software feature with just its basic aspects implemented. [13]

The minimum viable product is not always the simplest possible product if the problem the startup is trying to solve is not simple. Rather, the MVP should solve the core problems or jobs that the customer wants to get done.

Regarding actual software development, Lean Startup does not define any particular model. Instead, various approaches, including Scrum [22, 23], Lean software development [17, 18], and Extreme Programming (XP) [3, 2], are seen to be applicable. The main characteristics desired of the development model are the ability to provide cost-efficient designs that can be experimented with and their rapid development. Moreover, one can consider development approaches (e.g. [15, 16, 8, 7]) that build on experimentation as derivatives of the same mindset.

2.3 Synthesis

Software product lines have been developed to allow a company to produce multiple similar variants of its core product. This product line approach enables the company to push products out to the market faster and more efficiently than before, once the common assets and common architecture have been defined, by reusing shared domain assets for each product version.

Similarly, Lean Startup produces MVPs as fast and efficiently as possible to the market to gain information about the business space of the company and to iterate the company's core product towards a better one. In a sense, these iterations with MVPs eventually produce similar kind of a product line with specific variations to each product that answer the current hypotheses of the company.

The first few MVPs that a startup produces with Lean Startup might differ considerably from each other. This is due to the fact that the startups have limited domain knowledge and are working mainly on the basis of their leap-of-faith assumptions. After validating these first assumptions through MVP iterations, their MVPs should be more and more focused and resemble each other increasingly later on. This is because the company should first learn the things that are the most vital to their business, such as their business model, targeted markets, and the problem they are trying to solve for their customers. This avoids larger modifications in the later stages of product life.

After validating these hypotheses about their initial business space with MVPs, new MVPs should be developed to assist in learning ever smaller and more focused things. This again allows the increase in the reuse of software components from the previous MVPs. For example, if the software company focuses their product on a specific domain, such as web applications, the product line to produce multiple web MVPs could be constructed based on the existing knowledge of what is common in web applications and on top of this create different MVPs. The production (line) of such focused and similar MVPs could be seen as an "Iterative Innovation Engine" - a platform for a core product on which the startup builds its experiments to validate new hypotheses.

Thus, both Software product lines and the Lean Startup method promise increased productivity and faster time to market. Even though they have a different originating context, one from the startup domain and one from the corporate domain, they both are processes to produce multiple similar products. Whereas Lean Startup talks about MVPs, or MVFs, these end up resembling

similar software artifacts as the different applications in reactive software product lines. One key difference, however, is that the SPL approach plans to produce similar products in parallel, whereas the MVP approach plans to produce them sequentially. Thus the up-front costs and time-to-market of MVPs is lower, but the total lifecycle costs may be higher.

3 Case Study

3.1 Research Approach

The goal of this study is to investigate how our case startup companies used Lean Startup for their software development in practice. In particular, we address the effect of using MVPs in the creation of new businesses and business models, and the consequences of executing such a process.

This study was executed by performing semi-structured interviews and based on the knowledge of the authors from working in the two case companies. In general, a case study approach is particularly useful in situations where the phenomenon and context are difficult to separate [25, 20]. These semi-structured interviews also allowed for more in-depth discussions, which in turn made it possible to gather more information from the processes of the companies than pre-selected questions would have enabled.

3.2 Case Study Companies

Movendos Movendos is a new software startup focusing on creating effective tools for health and wellness coaching. The main product of the company is the Movendos health coaching platform. The first author of the paper has worked in this company. The goal of the company at the time of writing is to create an online cloud tool to help coaches to keep better track of their trainees and clients and thus enable cost savings for the health service provider.

The company has used the Lean Startup product development and multiple MVPs to iterate its core product to its current state through multiple iterations on it. The original business idea has evolved through the different versions from personal training data tracking, remote heart rate tracking to its current form as a remote training tracking tool.

Taplia Taplia is a software company creating simple and lightweight web applications that can be used to automate the logging of work hours. The products are targeted for field service engineers and other professionals working on time-based assignments. The first and the second author work in this company. Taplia aims at making work time tracking more efficient by removing paper shuffling and manual entry of time slips into a payment system. The company offers product customization on a per-client basis for an additional fee.

The company is still in its early stages of trying to find a valid business plan on which to iterate and produce a profitable business. A few business cases have

been developed with Lean Startup, but it still remains to be seen if the current business model is the final one.

3.3 Software Development in Case Companies

First, we investigated what kind of processes the case companies used for their software development. It turned out that both case study companies followed the iterative development model laid out by the Lean Startup. Although both companies dropped some parts of the Lean Startup methodology, both used the build-measure-learn based iterative method with MVPs. Taplia has produced four MVP products, Movendos implemented three during the time investigated in the study. The MVPs were used to test new product concepts or refine existing ones to guide the company decision making.

Secondly, we compared the different MVP versions developed by both companies and identified some common parts and patterns. The first MVP for Movendos was a test of a product concept for mobile heart rate and gym log tracking on a mobile platform. The second MVP was a more generalized version intended for tracking all exercises and food diary data with a mobile platform. The third product was expanded to include more focus on the coach using the system and transferred to the web platform. In total Movendos created three MVPs in the web application domain.

For Taplia, the first MVP done was a work-hour-logging web application for the mobile environment. The second was an hour-logging and planning application for gym employee hour tracking. The third was a work-logging and customer work order tracking tool for constructions companies. The fourth was an expanded gym employee hour planning and tracking software in the cloud. In total Taplia created four MVPs in the web application domain.

In both companies, the MVPs were based on the same core web technologies, which were also used in different MVPs. Although Lean Startup promotes the possibility to make radical pivots if an MVP is unsuccessful, both of the companies were highly web software oriented from the beginning and thus neither of the companies changed their initial decision on producing web software. Therefore, the later MVPs in the companies were generally developed faster as there was considerable opportunity to reuse gained knowledge of these technologies. For example, parts of the UI components were reused between the different MVP versions. However, in some cases technology switches were made and the core technologies were updated or changed. All in all, the case companies felt that the Lean Startup allowed the companies to develop their MVPs efficiently and learn rapidly from customer feedback.

Finally, we looked into the situations where the case companies ended up by following Lean Startup. Despite the advantages of reuse options, both companies also found out difficulties in their implementations of Lean Startup. In both of them, the termination of completed MVPs turned out to be difficult, and consequently the companies ended up with multiple products, each with a small user base each in use at the same time. By the book, however, MVPs are

primarily intended for learning purposes, and therefore only the fittest of the MVPs should survive (Figure 4).

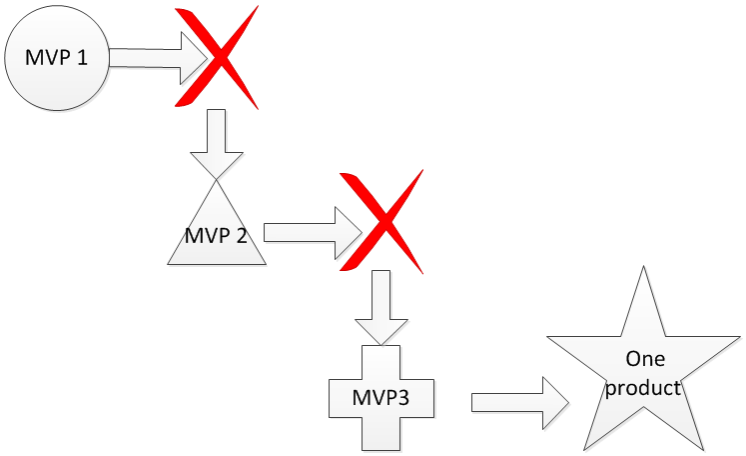

Fig. 4. Survival of the fittest MVPs

In our view, the closing of previous MVPs and freeing their resources could allow companies to make more extensive pivots, and therefore end up with a stronger final product. However, the situations our case companies ended up with resembled an involuntary (and sub-optimal) product line depicted in Figure 5.

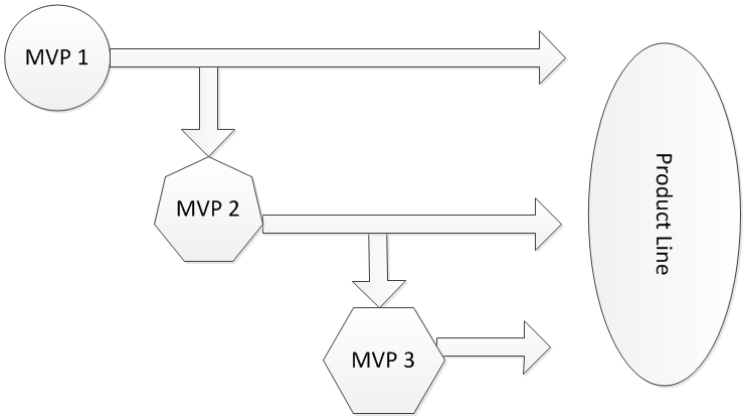

Fig. 5. Involuntary product line from MVPs

Both companies faced the same major challenges in the latter phases of the MVP development. When an MVP was developed and introduced to the customer, it became very difficult to stop developing it further, even if the customer feedback was not very good. Taplia customers even wanted to continue using older MVPs that were created for early prototyping and testing. Thus, shutting down the MVPs was not possible without risking the unhappiness and potential loss of these customers. This led to a situation where Taplia had to support multiple versions of practically the same MVPs at the same time but in practice mostly separately from each other. Movendos also had similar situations, where multiple versions of an MVP were in development at the same time. There was an overlap between finishing one MVP and starting to design another one. The first MVP was also in use with a customer at the same time as the development of the second one was progressing.

In these cases some individual features were beneficial to the customer and even though the MVP itself was not successful enough for further development. The customer, and therefore also the business, relied on those few individual features. If the MVP had been discontinued, the customer would have been unhappy, and the companies would have lost significant amount of business they had been able to attract. This led to the situation, where both companies decided to further maintain unsuccessful MVPs, despite the fact that they tied valuable development resources to relatively unprofitable applications when considering the full potential of companies.

4 Discussion

SPLs are typically seen as means to reuse existing assets. They require the separation of a common core from which different products can be efficiently derived over their lifetime. In MVPs, on the other hand, reuse is mostly opportunistic, and the emphasis is in the speed of development and meeting a minimum set of key requirements. Thus, reuse with MVPs only takes place, if a new MVP happens to be similar to the previous ones. Moreover, variability can only be seen in hindsight, by comparing a longitudinal set of features included in designs. For example, in the studied cases, the different MVPs used the same web technologies and even the same UI components. In this sense, both of the cases produced their MVPs by sharing some important core assets, which could be interpreted as a realization of an SPL, if executed properly. In a true SPL, however, the variability is created and evolves over time as well, but in a planned, managed, and controlled fashion.

Keeping multiple MVPs alive simultaneously led to resourcing problems in Movendos and Taplia. Every product was allowed to bloom. We propose that this kind of a situation can be seen as the creation of an unhealthy, accidental SPL, where the scope of the SPL is not managed—that is, where each MVP and their features were all maintained as core assets forever. This inevitably leads to a situation of "death by 1000 cuts", where the excessively broad scope bleeds the companies of resources. To summarize, while it is true that a Lean Startup

must be focused primarily on the short-term, and an SPL is focused more on the medium and long-term, the short-term focus of the Lean Startup does not abdicate the company from the responsibility of doing cost-benefit analyses [1].

Additionally, we propose that resource spending in the above situation steers the MVPs to be more similar with each other, since more pressure tends to emerge towards reuse than towards closing more MVPs. However, the technical basis of the MVP-driven product line cannot become as solid as an SPL, because the ad-hoc reuse practices will not produce as sustainable and well-thought set of assets. Still, the eventual situations inside the case companies studied in this paper resemble the outcomes of using SPL method, with their multiple MVPs alive and kicking. Thus, the main difference between the SPL and MVP is in how the eventual outcome is understood and valued. In SPLs the outcome is a basis for new products, worth being developed further, and adequately maintained. In MVP, on the other hand, the outcome is often something to be thrown away and used only as a means to learn and understand the customer needs.

To study these similarities between SPLs and Lean Startup's MVP development even further, we also compared the two methods and the risks they involve. We found that creating MVPs introduces similar risks to the organization as developing an SPL. These are discussed in more detail in the following.

Developing useless assets. The risk of developing possibly useless assets can be seen as the leap of faith assumptions taken with the first MVP versions. Also improper validation of the build-measure-learn cycle could lead to creating a new useless MVP version if the assumptions which upon it is based are false. In SPLs, by contrast, similar risk is associated with support for products that will never be produced—in other words, getting the SPL scope wrong.

Reuse related risks. The need for close coordination of reusing assets in an SPL can be seen similar as making unnecessary changes to an MVP, for example switching technologies, even though the switch is useless. This requires larger development effort than the simplest MVP production, and if the technology switch is not directly associated to an MVP hypothesis, it is a wasted effort. The same coordination is needed in SPLs where decisions on when to reuse assets or create new ones are also crucial.

Resource limitations. The limited resources in a startup environment and MVP development are directly related to the way SPLs are used to optimize resource usage by reusing components between different MVP versions. Startup companies have higly limited resources and resource management is critical. SPL theory could be used to guide these decisions in a startup environment.

Volatility and uncertainty risks. Highly volatile and uncertain environments are typical for startup companies that do not yet know their business space fully and are still exploring for the right product. The same problems of exceeding benefits while developing a reusable platform in SPL can be seen as the same risk, as creating reusable MVP components. We cannot clearly say what part should be reused without the use of historical data that recent startups do not have. The uncertainty is also tied to the lack of objective methods for evaluating component reuse in SPLs. The same problem surfaces in MVP development

when the company has to create MVP metrics and hypotheses to evaluate the changes made in the different MVP versions. At the same time component reuse and what to change to the next MVP should be analyzed. This might be higly variable, based on the results of the test MVP, and because one cannot be sure if the MVP test will be a success before executing the tests, planning component reuse becomes extremely difficult.

5 Validity and Reliability

Internal validity is a critical examination of whether the experimental treatment makes a difference; that is, whether the independent variable actually causes the changes seen in the dependent variables being examined [25]. Given that this was a relatively small case study, there was no control group, so it was impossible to run this study as a classic controlled experiment. Hence we can make no inference of causality, but can only observe relations that may merit further study.

External validity is a critical examination of whether the results of the study are generalizable [25]. The external validity of the case studies presented here is hampered by the fact that it is based on just two case study companies. These companies volunteered their data and as such they do not represent a random sample of software startups in Finland. Also as both companies are from Finland this study represents the behaviour of companies in the Finish operating environment.

6 Lessons Learned

The most important lesson learned from this study is that both Lean Startup and Software Product Lines require long-term strategies to be successful. A focus on gaining short-term benefits by first creating products rapidly and then iterating them quickly results in the achievement of some characteristics of both approaches, but not really in the benefits promised by neither of the approaches. In practice, however, startups—if they are operating on severely limited budgets—will likely choose short-term gains over following either of the approaches by the book simply because there are daily operations and practicalities to take care of. In this case they may experience the worst of both worlds: little strategic advantage is gained, because the assets created can not be broadly reused, and reduced agility over time as the software needing to be maintained grows in size and complexity, while losing its conceptual integrity due to rapid small modifications. The results indicate that it is reasonable to advocate closing MVPs without transfering any code forward in the development.

In the best possible world, combining Lean Startup and SPLs can result in an SPL that is actually the MVP for the company that is being built. However, this requires 1) carefully analyzing and controlling the scope of SPL, and 2) the ability to rapidly validate the business value of the technical construct. However, since creating a successful SPL requires considerable investment, aiming at

such an MVP will be even more complicated and costly than simply creating a more traditional product, which would also in most cases better correspond to Lean Startup ideals. Therefore, while displaying some promise, combining the two approaches is not a straightforward recipe for automatic success. Instead, a successful execution of Lean Startup and SPL creation requires careful thought, strategic planning, and good insights into future trajectories for the MVP. However, we do acknowledge that the MVPs can be created with a faster cycle using common core components, provided by an SPL.

Regarding the literature review performed for this paper, we found that while SPLs have been extensively studied, there are fewer reports regarding the use of the Lean Startup methodology in practice. Consequently, we believe that the experiences reported in this paper help in understanding how Lean Startups work in practice, as well as the risks associated with the approach. However, further research is required to gain more conclusive data on benefits and pitfalls of this combination.

7 Conclusions

Software Product Lines and Lean Startup, via its MVP model, promise increased productivity and reduced time to market. In this paper, we are reporting findings from our case study based on observing two companies and investigating their style of Lean Startup software development, where SPLs were formed by the differing MVP versions. These SPLs closely resemble classic SPLs – the handling of MVP versions, their component reuse, customer adherence, and lifecycle management are similar to SPLs, albeit on a faster time scale. However, the creation of SPLs was at least partially accidental due to mishandling MVP lifecycles, which is the root cause for the creation of an involuntary SPL. This multitude of products then consumes resources that could benefit companies more when invested in a more focused fashion.

Finally, there are numerous directions for future research. To begin with, in this paper we are reporting experiences from Finland only, and therefore extending the research to cover other countries is an obvious possibility. In addition, studying the different funding models of startups and their role in the resulting product and company strategy is also a subject for future research.

References

1. Asundi, J., Kazman, R., Klein, M.: Using economic considerations to choose among architecture design alternatives. Tech. rep., DTIC Document (2001)
2. Beck, K.: Embracing change with extreme programming. Computer 32(10), 70–77 (1999)
3. Beck, K.: Extreme programming explained: embrace change. Addison-Wesley Professional (2000)
4. Blank, S.: The four steps to the epiphany. K&S Ranch (2013)
5. Blank, S., Dorf, B.: The startup owner's manual. K&S; Ranch (2012)

6. Clements, P.C., Jones, L.G., Northrop, L.M., McGregor, J.D.: Project management in a software product line organization. Software, IEEE 22(5), 54–62 (2005)
7. Fagerholm, F., Guinea, A.S., Mäenpää, H., Münch, J.: Building blocks for continuous experimentation. In: Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering. pp. 26–35. ACM (2014)
8. Feitelson, D.G., Frachtenberg, E., Beck, K.L.: Development and deployment at Facebook. IEEE Internet Computing 17(4), 8–17 (2013)
9. Gamez, N., Fuentes, L.: Software product line evolution with cardinality-based feature models. In: Top Productivity through Software Reuse, pp. 102–118. Springer (2011)
10. Jaaksi, A.: Developing mobile browsers in a product line. IEEE software 19(4), 73–80 (2002)
11. van der Linden, Pohl, B.: Software product line engineering: Foundations, Principles and Techniques. Springer (2005)
12. van der Linden, F.J., Schmid, K., Rommes, E.: Software product lines in action: the best industrial practice in product line engineering. Springer Science & Business Media (2007)
13. Maurya, A.: Running lean: iterate from plan A to a plan that works. " O'Reilly Media, Inc." (2012)
14. Nobauer, M., Seyff, N., Groher, I., Dhungana, D.: A lightweight approach for product line scoping. In: Software Engineering and Advanced Applications (SEAA), 2012 38th EUROMICRO Conference on. pp. 105–108. IEEE (2012)
15. Olsson, H.H., Alahyari, H., Bosch, J.: Climbing the" stairway to heaven"–a mulitiple-case study exploring barriers in the transition from agile development towards continuous deployment of software. In: Software Engineering and Advanced Applications (SEAA), 2012 38th EUROMICRO Conference on. pp. 392–399. IEEE (2012)
16. Olsson, H.H., Bosch, J., Alahyari, H.: Towards r&d as innovation experiment systems: A framework for moving beyond agile software development. In: Proceedings of the IASTED. pp. 798–805 (2013)
17. Poppendieck, M.: Lean software development. In: Companion to the proceedings of the 29th International Conference on Software Engineering. pp. 165–166. IEEE Computer Society (2007)
18. Poppendieck, M., Poppendieck, T.: Lean software development: an agile toolkit. Addison-Wesley Professional (2003)
19. Ries, E.: The Lean Startup. Penguin New York (2011)
20. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empirical software engineering 14(2), 131–164 (2009)
21. Santos, W.B., de Almeida, E.S., de L Meira, S.R.: Tirt: A traceability information retrieval tool for software product lines projects. In: Software Engineering and Advanced Applications (SEAA), 2012 38th EUROMICRO Conference on. pp. 93–100. IEEE (2012)
22. Schwaber, K.: Scrum development process. In: Business Object Design and Implementation, pp. 117–134. Springer (1997)
23. Schwaber, K., Sutherland, J.: The scrum guide. Scrum Alliance (2011)
24. Wu, Y., Peng, X., Zhao, W.: Architecture evolution in software product line: an industrial case study. In: Top Productivity through Software Reuse, pp. 135–150. Springer (2011)
25. Yin, R.K.: Case study research: Design and methods. Sage publications (1994)