TAMPEREEN TEKNILLINEN YLIOPISTO

HENRI TERHO

ASSESSING PRODUCT AND FEATURE VIABILITY USING ITERATION

Master of Science Thesis

Master of Science Thesis
Tampere, 6.5.2015

Examiner: Tommi Mikkonen
Supervisor: Stefan Baggström

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Degree Programme in computer science and engineering
TERHO HENRI: June 2015
Asessing Product and Feature Viability Using Iteration
Master of Science Thesis, 46 pages
Examiner: Professor Tommi Mikkonen
Supervisor: Stefan Baggström M.Sc (Tech.)
Keywords: Lean Startup, minimum viable product, agile software development, startup

Managing a startup using methods proven to work in established enterprises does not typically work as expected, as the startup operates under a totally different environment as a large enterprise. While enterprises create and develop products based on a known business model, a startup operates under extreme uncertainties and changing conditions. These different conditions have led to the development of new tools and practices for startups with little resources to develop new products.

This thesis investigates the use of the Lean Startup method using two different startups, Movendos and Taplia, as case studies. It is said that Lean Startup can be used in startup environments where the company is working under uncertain conditions to quickly learn about its business space. The methodology focuses on fast iteration of product and feature concepts based on minimum viable products. This quick iteration and delivery of product to clients produces immediate feedback from the clients and allows the startup to develop their business practices.

The case study showed that the two companies had implemented a slightly different subset, but both utilized the MVP concepts. It was found that the lean startup concepts seemed to allow the companies to quickly iterate their products towards a viable one. However both companies had problems analyzing when a minimum viable product was actually viable. Testing a full product with a subset of features sometimes led to situations where the MVP did not really emulate the full product. Another problem was the resource intensiveness of the testing and iteration. Both companies had to make compromises and not use all the recommended tools in the Lean Startup method.

The Lean Startup seems to be a decent framework for setting up a software business as its iterative nature fits well with the agile software development. Both companies under investigation reported that they had benefited from the lean startup ideas. As such the lean startup method should not be taken as total method for developing software, but as a toolkit from which to pick tools that suit your software product.

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan tutkinto-ohjelma
TERHO HENRI : Kesäkuu 2015
Tuotteen ja ominaisuuksien toimivuuden arviointi käyttäen iteraatiota
Diplomityö, 46 sivua
Tarkastaja: Professori Tommi Mikkonen
Ohjaaja: Diplomi-insinööri Stefan Baggström
Asiasanat: Lean startup, minimituote, ketterä ohjelmistokehitys, startup-yritys, asiakaslähtöinen ohjelmistokehitys

Startup-yritysten johtaminen isoissa yrityksissä vakiintuneilla käytännöillä ei yleensä toimi oletetulla tavalla. Tämä johtuu startup-yrityksen ja vakiintuneen yrityksen aivan erilaisesta tietoisuudesta omasta toimintaympäristöstään. Kun isot yritykset toistavat jo tiedossa olevaa liiketoimintamallia, niin startup-yritykset yrittävät vielä löytää itselleen toimivan liiketoimintamallin. Startup-yritysten tietous omasta toimintaympäristöstään on myös heikko. Tähän erilaiseen ympäristöön tarvitaan omat toimintatapansa ja työkalunsa, joista yksi on Lean Startup metodi.

Tämä diplomityö tutkii Lean Startup metodin käyttöä kahden yrityksen, Taplia Oy:n ja Movendos Oy:n tapaustutkimusten kautta. Lean Startup metodi on ketterästä ohjelmistokehityksestä ja asiakaslähtöisestä suunnittelusta kehitetty yrityksen kehitysmetodi. Se koostuu useista eri tekniikoista, joiden tarkoitus on tuottaa nopein iteraatioin uusia tuotteita, joiden perusteella yrityksen olettamia sen toimintaympäristöstä varmennetaan tai kumotaan. Yksi keskeinen tekniikka on nopea iteraatio minimituotteiden avulla.

Taustatutkimuksessa huomattiin, että molemmat yritykset ovat toteuttaneet hieman erilaisen osuuden Lean Startup metodin tekniikoista, mutta molemmat käyttivät minimituotteita. Molemmissa yrityksissä oli havaittavissa, että minimituotteet ovat hyvä tapa testata yrityksen toimintamallia ja tuotteita nopeasti. Näin varmennetaan, että alkupään vähäisiä resursseja ei käytetä vääriin asioihin. Käytännössä molemmilla yrityksillä oli kuitenkin ongelmia rajata minimituotteiden suuruutta ja testata isompaa lopullista tuotetta rajatummalla minimituotteella, koska se ei kuvaa lopullista tuotetta kokonaan. Myös minimituotteen eriävä hinnoittelu varsinaiseen tuotteeseen verrattuna tuotti ongelmia. Usean minimituotteen jatkuva testaaminen ja tuottaminen oli myös hyvin resurssi-intensiivistä. Tästä johtuen molemmat yritykset tekivät kompromisseja Lean Startup toteutuksissaan.

Tapaustutkimusten pohjalta Lean Startup näyttäisi toimivan hyvänä toimintamallina ohjelmistoalan startup-yrityksille. Molemmat yritykset kokivat, että olivat hyötyneet Lean Startup metodin käytöstä ja että sen iteratiivinen tekniikka sopii hyvin yhteen ketterän ohjelmistokehityksen kanssa. He kuitenkin kokivat, että kaikkien sen suosittelemien teknologioiden toteuttaminen oli liian työlästä. Näin ollen Lean Startup metodia pitäisi ajatella kokoelmana työkaluja, joista valita omaan ohjelmistokehitysmalliin sopivimmat työkalut.

PREFACE

This masters thesis is based partly on my work in the two startups Movendos and Taplia. I would like to thank the personnel of both companies, Esa Kaarna and Sampo Suonsyrjä from Taplia and Arto Leppisaari, Stefan Baggström, Hannu Nieminen, Hannu Mikkola and Carlos Perez Blancom from Movendos for my time in both companies. For proofreading this work I thank Teemu Terho, Merja Terho, Arno Pammo and Sampo Suonsyrjä. I would also like to thank Tommi Mikkonen who is the examiner of this thesis. I would also like to thank all of the persons who helped to stitch me back together after my skydiving accident in the summer of 2014 and made it possible for me to finish this thesis.

Henri Terho
Tampere 20.5.2015

CONTENTS

1 INTRODUCTION

Managing a startup using methods proven to work in established enterprises does not typically work as expected, as the startup operates under a totally different environment as a large enterprise. While enterprises create and develop products based on a know business model, a startup operates under extreme uncertainties and changing conditions. These different conditions have led to the development of new tools and practices for startups with little resources to develop new products. One example of this is the Lean Startup and its minimum viable product model for product development.[1]

The goal of this thesis is to investigate the validity of the Lean Startup and its iterative software development method as a way to develop software into a commercial product in a startup environment. The research has been executed as a case study of two startups claiming to be using parts of the lean methodology in their software development.[1]

We investigate if the additional job of developing multiple minimum viable products, later abreviated as MVP, actually produces waste as the amount of different software being developed increases or does it allow the company to quickly test new concepts without too much development time used. The Lean Startup uses the concept of validated learning to give value to these MVP projects. Therefore we will also investigate if the concept of validated learning is useful for software development.

We investigate the above by establishing what methodologies are used in Lean Startup to develop software and compare these theoretical models for software development into the actual practice in two companies, Taplia[2] and Movendos[3]. Based on the data collected from the companies on how they develop software in a lean way, we evaluate the validity of the Lean Startup method to create software.

This thesis is focused on the software development side of the Lean Startup. The business side, while inherently entwined with software development, is given only a lighter focus. The main focus is on the early startup phase, not the product acceleration and management after a successful start of the company.

We will also briefly go through the basics of agile development, but do not expand towards these too much, as there is a large amount of literature covering these subjects. The business model iteration is also only skimmed upon in the case studies and left out of the final thesis.

In Chapter 2 we go through the Lean Startup method. The origins of the Lean Startup movement are explored. We also go through how Lean Startup method has developed from the roots that were laid out by lean and customer centric development methods. We also go through the Lean Startup software and business development methods and how iteration and the MVP is used thorough the development process.

In Chapter 3 we focus on the aspect of hypothesis validation in Lean Startup. We study iteration and the build-measure-learn loop in practice. We also expand the thinking into the three different levels of development in startups and then talk about how the minimum viable product offers us a practical tool to do this iteration with both the products and features.

In Chapter 4 we go through two case studies of lean practices in two different companies. First we go through the case of Movendos, a web service startup developing coaching solutions for the web. Then we go through Taplia, a company which develops work hour logging software as a service on the web.

In Chapter 5 we consider how Lean Startup methods have been used in practice in the two companies. We analyze how the Lean Startup has affected software and product development in these companies and whether or not it gave the companies a competitive edge in the software development.

In Chapter 6 we summarize the results and offer a conclusion to the research question of the feasibility of the Lean Startup as a software development method in Finnish startups.

2 LEAN STARTUP

This chapter is devoted to explain the new Lean Startup approach to business, which is being adopted around the world. The core ideas of the Lean Startup method and how it links with software development are presented. The Lean Startup model in itself needs not be limited to the software industry, but it links with it really well. The idea is based on the realizations of Eric Ries, who understood that established businesses and startups need totally different methods for creating new products in their environment. The approach is closely related to customer centered design and agile software development and which are explained in this chapter. The basic idea of the Lean Startup is to develop different measurement tools based on learning for startup companies who really do not know their market that well. [1]

2.1 Origins of Lean Startup

The Lean Startup method was developed by Eric Ries based on his two failed startup experiences with Catalyst Recruiting and There Inc. Catalyst Recruiting was a “Dot-com boom” time startup that failed because Catalyst did not really know what the customers wanted and focused too much on the initial product release and marketing. The second was There Inc; a Silicon Valley startup with millions of dollars and five years spent on stealth R&D. There Inc;s product failed, because they were unable to gain more popularity after the first early adopters. [4][5]

After these experiences with software companies, Ries was determined to do things differently with his next startup, IMVU. IMVU is a company whose product is a 3D avatar chat program. At IMVU Eric Ries and cofounder Steve Blank were both ready to test new practices in software development and determined to make everything wrong when compared to the traditional model for software development. Thus the Lean Startup methodology evolved through the ideas that the business side of a software startup should also be managed with the same types of methodology as programming and engineering. [1]

The Lean Startup method also incorporated some ideas of lean manufacturing, which was pioneered by the Japanese auto manufactures in the 1980's. This idea that you should only focus on creation of customer value as fast and with as minimal waste as possible. An additional side of lean manufacturing is immediate quality control. When a quality defect is detected, the whole production line can be stopped to determine the reason for the problem immediately. The Lean Startup method imitates these same attributes, but in the context of software development and startups.[6]

The Lean Startup methodology tries to eliminate waste in the production phase of software and add user feedback as the quality control to enable startups to create satisfactory initial products on much more limited resources. In comparison to large companies, startups generally have low funding and limited resources. By focusing only on the essential in software development and validating that you really are concentrating on the essentials through customer feedback, the model has potential to reduce the amount of waste in startups. Customer feedback is usually measured through performance metrics from the system, A/B testing, key metrics and prototyping.[1]

The Lean Startup method is not limited to low resource startups however. Lean Startup is being adopted by many larger companies to use in their research and development departments when creating new products. The idea to build and test new business avenues for big companies effectively is tempting. To achieve this, many companies want to create internal startups. These internal startups are given resources to work independently from the rest of the company and effectively work like a company within a company. [1]

The Lean Startup method has been mainly received positively in the industry, with big companies adopting it quickly and deploying their own internal startups. It has also garnered an active community around it with many new startups advocating for it. Some criticism on the methods weak points has also started to surface, showing situations where the lean methodology and its terms do not work as intended[7][8][9].[1]

2.2 The Lean Startup method

Eric Ries summarizes the Lean Startup method into five core principles in his book “The Lean Startup” adressed in the following subsections. All of these are geared for an entrepreneur or a startup that tries to innovate a radically different product in a situation where they still do not have an extensive knowledge of the market for the product. These core principles are overlapping, but support each other in their message.[1]

2.2.1 Entrepreneurs are everywhere

Typically a startup company is pictured as fresh graduates with limited funding. Actually, as explained by Ries, there are entrepreneurs everywhere in the corporate world. A startup is any organization or group of people who try to create new products or services under conditions of extreme uncertainty. The specific definition for a startup according to Ries is: ”A startup is a human institution designed to create a new product or service under extreme uncertainty.”[1]

A product development team in a large corporation can easily be a startup, as it is a human institution, which tries to learn about a new business space. The idea of operating under extreme uncertainty is important. If the business model which is going to be executed upon is known and its success lies only on the execution, that company fulfilling that business plan cannot be said to be a startup as it is not operating under uncertainty, but according to an already known business plan. [1]

Many companies have started to deploy Lean Startup in their product development processes. For example Intuit, Americas largest producer of finance, tax and accounting tools, created a small team which operated outside the corporate structure to create new products. This team succeeded in producing a new product called SnapTax, which was highly successful. The product was developed in a customer centric way, with a core idea of automatically collecting data to fulfill a tax return. Based on conversations with potential customers, an minimum viable product which allowed them to use their cell phones to take a picture of their forms and fill the data that way was quickly tested and deployed. [1][10]

2.2.2 Entrepreneurship is management

A startup is not only its product, it is an institution. Thus it requires new kind of management to work under the aforementioned conditions of extreme uncertainty. The old managerial techniques are designed for corporations, not really for small startups. These small companies need lean management tactics. [11][12]

Startups are normally filled with people who do not know much about being an enterprise, and usually resort to copying the classic enterprise structure to their company as stated by Steve Blank. This leads to the people in the company to adopt standard roles like chief technical officer and chief marketing officer. This creates an environment where the roles of larger companies are integrated into the small company. This also creates an environment where the company tries to use classic managerial techniques to run a startup company. This is not an optimal strategy for small companies, as traditional business plans presume no trial and no errors, which form the core of the

Lean Startup. Also the classical exact budgets are unnecessary and might even be hampering. [1][11][12]

A new leaner management structure should be established, which places emphasis more on hypothesis testing and iteration, not just on execution. Also the classic idea of a revenue forecasting and assuming higher profits in the future leads to premature scaling on the product, and also leads to situations where the company should spiral always upwards. These models do not promote a lean thinking, where one should pause or slow down to understand your customers and products better to make the necessary product adjustments. Normally these problems lead to premature scaling of the company, which always expects future profits to grow. When the profits do not grow, management usually switches to management by crisis. In this mode, problems are solved when they appear, but if you would have stopped and analyzed the situation this death spiral might have been averted according to Steve Blank and Bob Dorf.[11]

To create a new business strategy for startups, Ries offers the Lean Startup strategy as a new way to manage companies under uncertainty. By measuring the effect of our work in fulfilling customer needs, we can assess the validity of our business plan. In the context of Lean Startup, this means that the company must have a basic vision, which is used to steer the company and its core values. The company has an initial strategy, which is tested with a minimum viable product. If the validated learning from these tests show that your business plan is not fulfilling the customer needs, or is resulting in an unsustainable business model, you should change your whole business strategy based on the data gained from the experiments. This is shown in Figure 2.1. This model decouples the execution of the plan from its value, as executing a bad strategy well does not increase its value.[11]

2.2.3 Validated learning

Startups do not exist just to create a product and money. According to Eric Ries, they exist to learn how to create a successful business model that can be validated through scientific methods and used to feed the engine of growth of choice. To achieve this, one must have tools and measures to make sure one is progressing towards an successful business model, and not going any other way. Validated learning is a way to produce valuable data from your initial business experiments with a plan, not just create a product and see what happens. Validated learning expects that the experiment must be based on an empirical and measurable evidence, with a hypothesis formed beforehand, which can then be either validated or falsified based on observations made from the real world. [13][1]

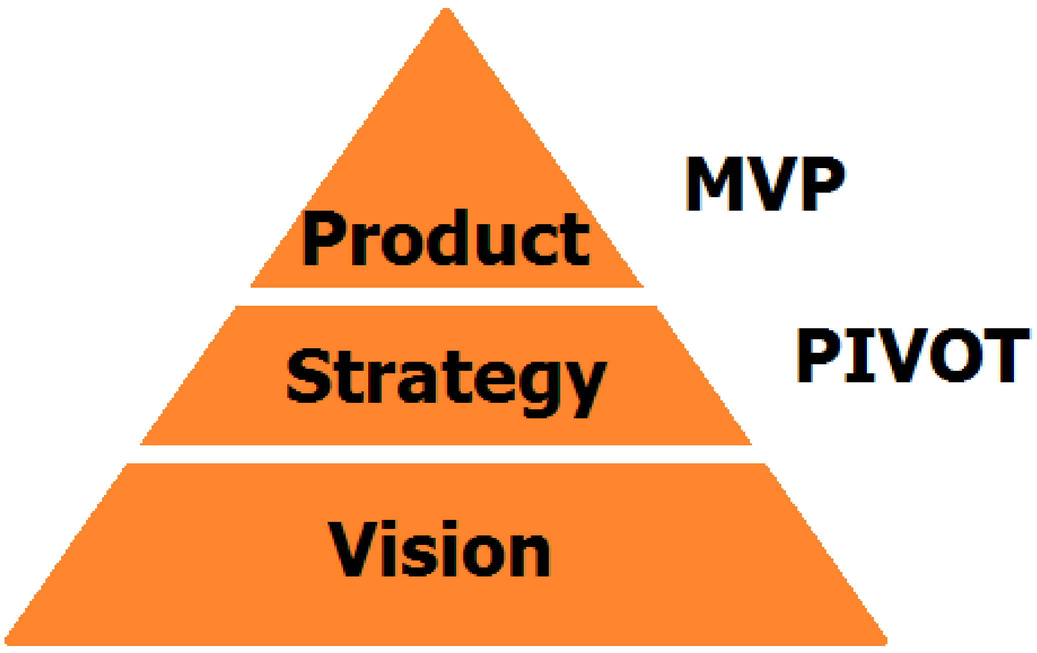


Figure 2.1: Relationship of product, strategy and vision

Unfortunately, in the modern corporate world, learning is usually seen as an excuse for failed execution. The idea behind validated learning is that you now have validated the learning you have achieved through documented tests and hypotheses. The lessons learned from these can be then fed to innovation accounting to help you to make sense of the business space you occupy. This information can then help you create the right products to test your new hypotheses about your possible customers, the so called minimum viable products.[11]

Minimum viable products are smallest possible product iterations, which are used to measure customer behavior. Customer behavior in minimum viable products can be used to test new products, or just new minimum viable features. As an example, minimum viable features can be tested with A/B testing, where the new feature is only provided to half the users, and the other users are provided with the old version. Thus we can see the impact of the new feature immediately and compare it to the old version. This continuous testing of features in the fastest way possible with real customers is one of the key ideas in validated learning.[1][13]

2.2.4 Build-Measure-Learn

One of the core ideas of the Lean Startup, and the core of its iterative nature is the build-measure-learn cycle. According to Eric Ries, the core idea of a startup is to transform new ideas into products. Also for startups, information is much more valuable than money in the starting phase. When this is broken down into phases, you can identify phases. First, an idea is turned into a product by the startup. When customers interact with the product, they generate feedback. This is typically both qualitative and quantitative. Then based on feedback, the startup learns more about their business space and the performance of their products. This is illustrated in the Figure 2.2. [1][11][14]

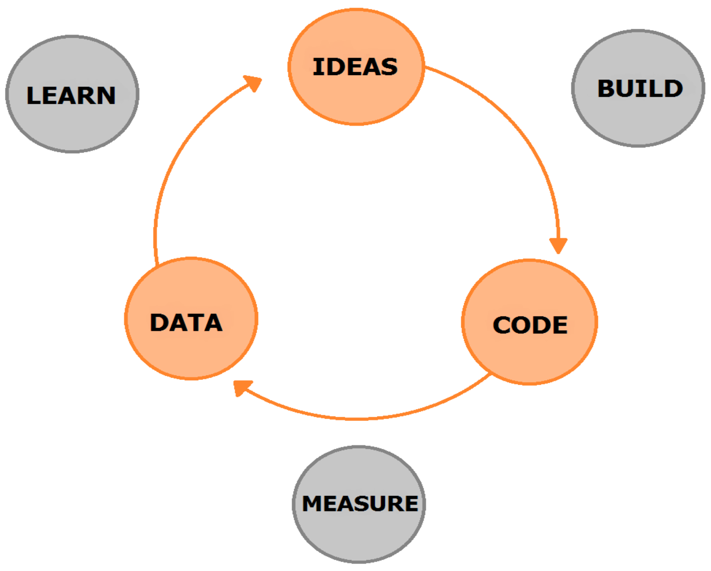


Figure 2.2: build-measure-learn loop

The first state one enters the loop is 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 Lean Startup provides no way to test it beforehand. The problem should be assessed with customer interviews or other methods for initial validity. This is then fed to the build-measure-learn as an original assumption and iterated upon to reach a valid product hypothesis.

The *build phase* is the transition from ideas state to the code state. During this phase a version of the product is being built, based on initial ideas. Typically this product is a minimum viable product, meaning a product that only contains the minimum amount of features to make it viable with the minimum amount of work. The product should also mimic full product in the aspects of monetization model. The goal is to maximize the amount of iterations through the build-measure-learn loop. The product should also be built with analytics integrated, to enable the collection of feedback from the customers.

In the *code state*, you have a ready product version of your program. The code is not totally ready, but it is a current iteration. This is an minimum viable product version. The software should also include analytics code to enable the collection of data about customer behavior, or there can even be two versions to enable A/B testing of the product.

The *measure phase* is where you deploy your product to your customers. The product is used by real customers and data about their behaviour inside the product is collected by the analytics code in the program. The type of data collected should be such, that it can be used to create validated learning.

The *data state* is where you analyze the data collected from your product. The purpose here is to decide whether product development efforts have led to progress. The data collected from the product should be something that can be acted upon, Innovation accounting produces tools that allow judging of the data collected here. In this phase it should be assessed whether your learning milestones have been fulfilled.

After this comes *learning phase*. In this phase you compare the data collected to your original product hypothesis. The data can be used to see if the customer behavior matched your expectations and if the changes made to this iteration improved the software, for example if the new landing page increased the amount of subscriptions. If the hypothesis was successful, a new hypothesis should be formulated to further improve the software, but if it was not successful, a pivot should be considered.

The pivot is a decision if the company should stick to the same business plan based on the data collected from customers. In essence, does our product hypothesis match what we observed from the tests? If our hypothesis did not match, the company should change its strategy and form a new product hypothesis, which is then tested with an additional rotation of the build-measure-learn cycle. Here we can see that failure is an integral part of the iterative lean process where even a typical failure of getting the product hypothesis wrong, can lead to valuable data. Ries states that the more iterations of the build-measure-learn cycle you can do with your initial funding, the better your learning about the market is and thus your product can be steered towards success. [1] [11]

2.2.5 Innovation accounting

Innovation accounting gives startups tools to measure their progress, with the focus placed on learning, not on execution. Innovation accounting gives tools to measure progress in the company, a guideline on how to set up milestones and how to prioritize work. Innovation is fed validated learning and innovation accounting is how to make decisions and plan new hypotheses based on the data and learning gained.

One important distinction should be made for the data that is used for justification. The metrics that are used to judge the next phase should be actionable metrics, not just positive vanity metrics. Vanity metrics are dangerous in the sense that they are any metrics that paint a positive picture of your product, but not a metric that was chosen as a way to falsify your own hypothesis. These vanity metrics are typically used to uphold a success theater, where you can always find something positive if you measure everything and therefore claim your success.[6][11]

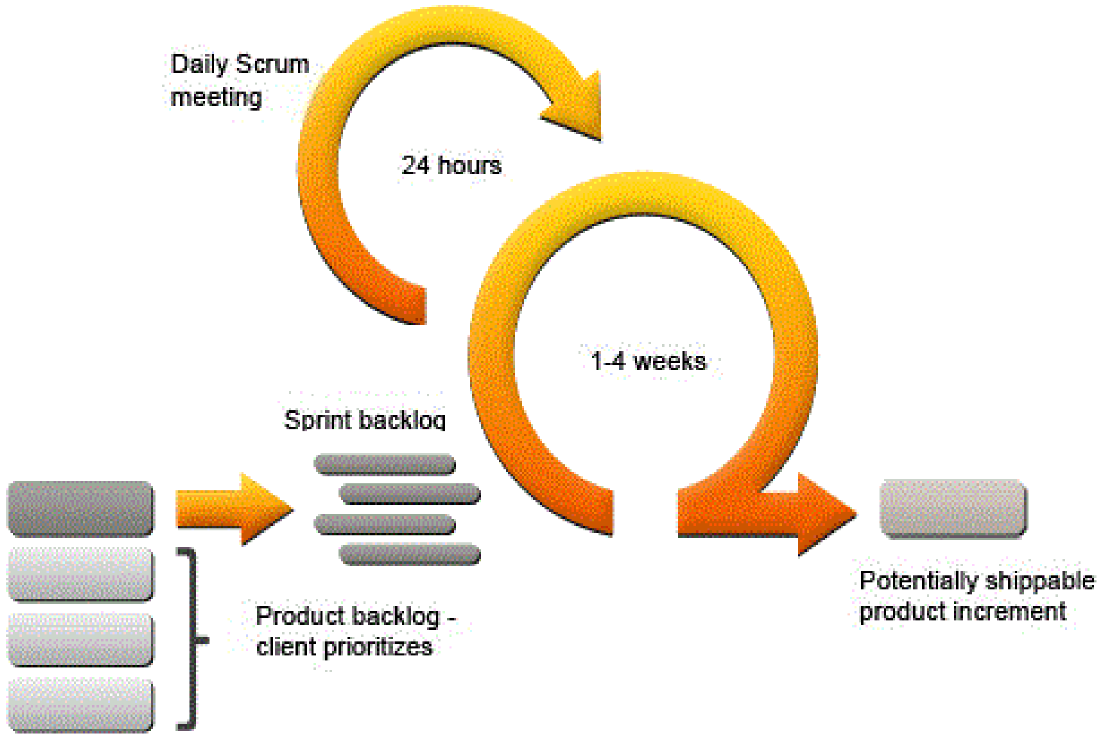
One of the tools outlined as an analysis tool in the Lean Startup is root cause analysis, specifically the five whys variant. This is based on the original ideas of Sakichi Toyoda in the toyota industries corporation. The five whys technique is meant to find the root cause behind a longer cause and effect chain. This allows you to find the root cause for your problem. The root cause analysis should not be used as five blames, but assume that if people have failed, it is the process that failed.[1][15]

2.3 Agile software production

The Lean Startup method is defined to create an environment of cost-effective software production by building a minimal product focused on customer needs and testing the product-market fit through this. The iterative nature of agile software development matches the iterative structure of the Lean Startup. Another document that emphasizes agile principles is the lean manifesto. The manifesto lists the core tenets of agile programming as compared to the waterfall model as: 1)individuals and interactions over processes and tools, 2) working software over comprehensive documentation, 3) customer collaboration over contract negotiation and 4) responding to change over following a plan [16]

Even though it could be interpreted as such, agile software production is not anti-methodology. Multiple different iterative software development methodologies have been developed around agile, for example Extreme Programming(XP) and Scrum. The companies covered in this thesis use a variation of Scrum. Scrum is an iterative and agile methodology. Its focus is on a flexible iterative development in an environment, where the customers can change their mind on the software requirements. This was outlined for the first time in the 'new new product development game' by Hirotaka Takeuchi and Ikujiro Nonaka.[17][18]

Scrum decomposes the work into small manageable items, which are collected into a backlog of all work to be done. These are then allocated into sprints. Sprints are iteration cycles, which implement a part of the backlog, called a sprint backlog. A sprint is a timeboxed segment, typically 1-4 weeks in length, after which a working part of software based on the items in the spring backlog has been created. The idea is to always have a potentially shippable product increment after a sprint, which can be demoed to the client. This cycle is shown in Figure 2.3. After a successful sprint, the cycle is repeated again to produce a yet another potentially shippable product increment.


Daily Scrum meeting
24 hours
1-4 weeks
Sprint backlog
Product backlog - client prioritizes
Potentially shippable product increment
Figure 2.3: Scrum cycles

Scrum also organizes the software development team into diferent roles. These are the scrum master, product owner, team and stakeholders. A scrum master is a person whose job is to facilitate the team and is responsible that the team is capable of delivering product goals or deliverables. He is responsible for maintaining scrum processes within the team. A product owner is responsible that the team delivers those product goals or deliverables. He is responsible for organizing the product backlog and prioritizing work to be done on the product. The team refers to the group of people doing actual implementation work, for example coder, testers and user experience designers. Stakeholders are all the other project stakeholders, clients and if the company is large enough its senior management.

One aspect of Scrum is the focus on inter team communication to improve the quality of software created. According to the Agile Manifesto, the most efficient method of conveying information between the team is face to face conversation. The Scrum process includes multiple meetings to this effect. The contents of any sprint is chosen in the sprint planning meeting, where the list of features to be done in the next sprint is frozen. Only the backlog can be changed by the client requirements during a sprint, so the sprint segment stays unchanged for its duration. During the sprint there are daily meetings, as seen in Figure 2.3, to keep everyone up to date on the state of the project and to quickly surface any impediments or problems with work items. The last is the sprint end meeting, which is used to asses what was done and what went well during the sprint. This is to demonstrate the completed work that was done and also learn more about the software development process. These lessons are then used to improve the next sprint. This usually leads to the Scrum process changing over time as the teams adapt the software process based on their findings.[19]

3 HYPOTHESIS VALIDATION

This chapter is about validating the chosen business hypothesis using the tools of the Lean Startup method. In the following, we go through all of them, but our main focus is on the minimum viable product approach and later two case studies of MVP usage. We limit the analysis of business hypothesis iteration to be outside of the scope of this paper and only skim it. The focus is more on product-market fit iteration and feature iteration.

3.1 Business hypothesis driven experimentation

The core tenets behind the Lean Startup in all phases of product development are based on the build-measure-learn loop shown in previous chapters. The core idea is that iteration and validation are the most important functions of the Lean Startup. At first product hypotheses are tested and iterated and ranked against each other. Then after a potential hypothesis has been found it is tested using a minimum viable product to see if the hypothesis matches with reality. Finally, when a valid product has been found, the products features are iterated to produce better value. This is illustrated in Figure 3.1.

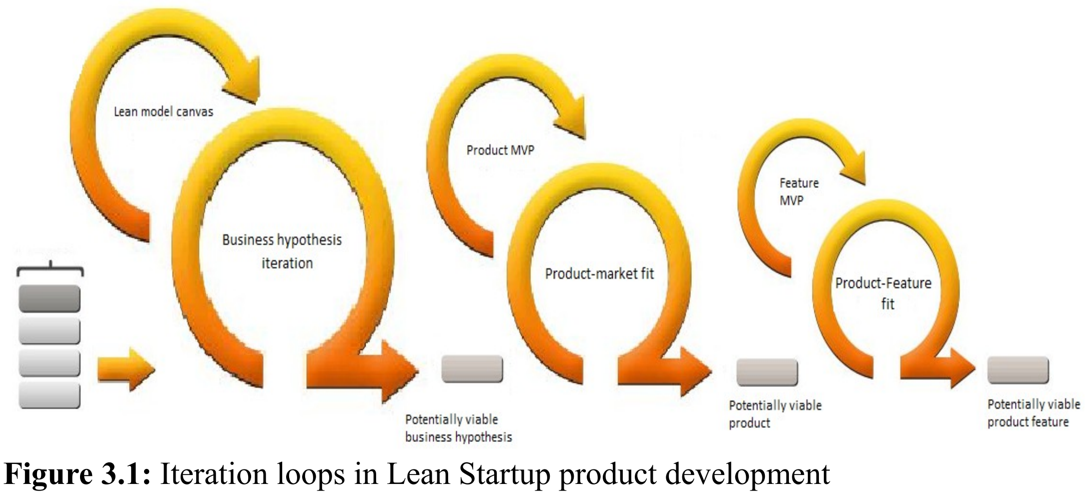

Lean model canvas
Business hypothesis iteration
Potentially viable business hypothesis
Product MVP
Product-market fit
Potentially viable product
Feature MVP
Product-Feature fit
Potentially viable product feature

Figure 3.1: Iteration loops in Lean Startup product development

3.1.1 Business-hypothesis iteration

The first phase of product development in a startup company is choosing the right business hypothesis. This is the first loop in Figure 3.1. In this phase the company is actively looking for a problem to solve. Initially the startup company has no data at all on the validity of different business models. This can be solved by finding a customer need that the company wants to solve and iterating on problems in this area. One method of validating these ideas is to do initial surveys and face to face interviews with potential customers. Other way to analyze the business aspects in a lean way is to formulate initial lean business model canvases of these hypotheses. Based on these initial findings, the most promising candidate can be chosen to be implemented as a product. This product is then used to validate the business hypothesis. If the hypothesis proves to be false, an another idea can be tested, a pivot happens to a different idea. [1][11]

Another important goal to be achieved here is the identification of early adopters. Early adopters are people, who are willing to look past the small errors in your product and see the potential in it. They typically fill the missing parts with their imagination and are the first to test out new technologies. Early adopters are important first testers of the product as using them, the product and its viability can be tested early without having a ready product.

The core idea behind the iterative approach is to test the hypothesis as early as possible. Before this validation, the choice of business hypothesis can be considered to be a leap of faith. This is the typical situation in the start up company field nowadays, that an idea is chosen because it sounds feasible to the founders and they think they know the answer. But with lean methodology the idea is to steer your idea towards the real customer problems, where your own ideas work as the starting point. The testing and probing around the hypothesis provides the company with knowledge about the business space they have chosen, allowing them to base their software decision choices on more concrete data and minimize wasted efforts on products. This idea that the whole product can be considered as an feature that can be cut of, if it seems unviable after testing is just a scaled up version of the agile software development methods implemented on a product feature scale in many lean software companies.

3.1.2 Product-market fit

The next stage of iteration is the iteration of the Product-market fit. In this stage a concrete product based on your business hypothesis established in the last loop is made. The main idea of this part is to find out if the business hypothesis can be validated and really matches the customer need. Also the goal of the lean methodology is to achieve this goal with minimal waste of resources, because typically start ups have a limited amount of resources available to them. This makes the efficient usage of resources in a startup environment even more critical. The tool to achieve this from the Lean Startup toolkit is the minimum viable product.

3.1.3 Feature-product fit

The final loop in Figure 3.1 is about iterating the features of your product when a suitable enough product-market fit has been found. This phase closely resembles agile development. Feature-product fit is a zoomed version of the product-market fit, where suitable features for an established base product are iterated upon. The main goal of this phase is to accelerate the found business model to maturity as soon as possible. The idea is the same as in product market fit, iterate through the build-measure-learn loop as fast as possible, testing which features are deemed worthwhile and make your business go forwards. An added feature is compared to the old version using split testing or similar methods and the data about user behavior is collected. Based on the user behavior data and feedback, it can be assessed if the new features is a wanted one, which produces more value to the customer.

3.2 MVP as a tool to find product-market fit and feature-product-fit

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 least amount of development time. The minimum viable product is a version of the software that will be tested with some identified early adopters that are willing to evaluate your product in its current state.

3.2.1 Minimum viable product structure

The minimum viable product should contain the features that produce your software solutions unique value proposition and little else except logging and metrics integration. The idea is to cut out all non-essential features and leave just the core features of your application. No fancy graphics, just the core that is still viable. This problem of selecting what is viable in an MVP also makes you focus only on the essential part of software development. The amount of features in an MVP should be optimized to allow for maximum customer feedback and resonance against as small a feature set as possible. This is illustrated below in Figure 3.2, where the amount of functionality and resonance with early adopters is graphed. This is an adapted version of the featuritis curve.[20][21]

As illustrated by the Figure 3.2, the minimum viable product is not always the simplest possible product if the problem we are trying to solve is not simple. Rather, the MVP should solve the core problems or jobs that the customer wants to get done. In here customer resonance means the amount of feedback you get from your user base.[1]

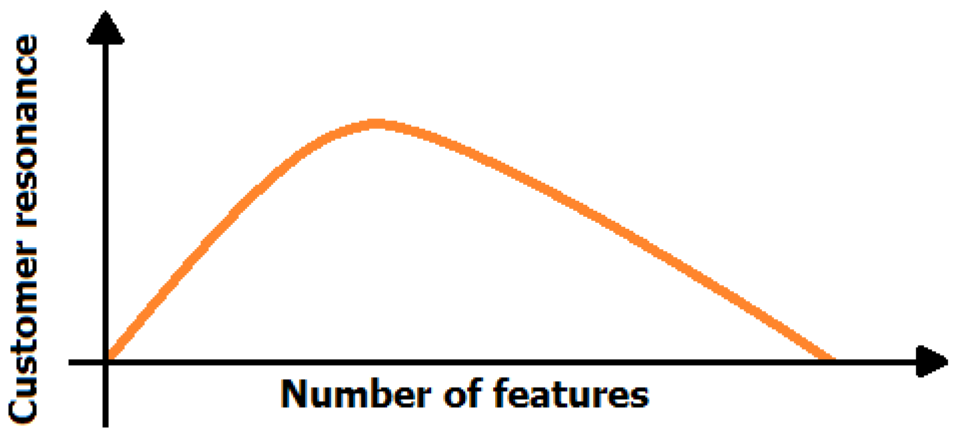


Figure 3.2: MVP feature curve

Also the MVP product should also match your product in its monetization model. This provides straight feedback for your value hypothesis and amount of work required to solve the problem. Essentially: Is this problem worth solving when compared to the amount of work needed to solve it? If the solution takes too much resources and software development as compared to the revenue gained from it, it is clearly not worth it to devote software development time to solve the problem.[1][11]

3.2.2 Typical MVP structures

Next, we dive a bit into the development of a minimum viable product and explain some common types of MVP's seen in software development. We also consider how they should measure customer behavior in the system to produce the maximal amount of validated learning.

When compared to the waterfall software development and prototyping, the MVP is a different kind of prototype. Usually in software development, a prototype is built after the product idea has been set in stone and the prototype could be considered a beta/alpha test to iron out bugs in the product. Also typically a standard prototype is made in such a way that it uses the same technologies as the full product, and it has been built to scale from the ground up.

MVP takes a different approach. Usually MVPs are implemented using different technologies than the final product, and they are not built to scale. They are mock up versions of the actual idea built upon smoke and mirrors to make sure that no money is spent on building a product no one wants. From this we can see that the driving factors in an MVP are testing the product-market fit, gaining information about the market-space and customers and doing all this with minimal amount of resources. Totally different goals than in a typical software prototype.[22]

Some common types of MVP are listed in the Table 3.1. This is a loosely categorized set of different MVP building strategies grouped by their inner workings. As can be seen, some do not actually do any software automation at all at this point, but the work is actually done by humans in the back end.

Table 3.1: MVP stereotypes

MVP type	Core idea
Smoke test	Not even an actual product. Just a mock up landing page, video or a mailing list to ask people to sign up.
Concierge	Provide the service to a single or small group of customers in person without any software.
Wizard of Oz	Fake main functionality behind the application with human resources.
MVP not to scale	Only include core functionality of your application and strip anything else.

Smoke test MVPs are versions that involve very little actual product development. They essentially make a customer believe you have a product coming up that accomplishes the features shown. One smoke test type MVP is the Dropbox video. The video was an edited video clip that showed how Dropbox would work even before a single line of code had been written. Using this video and a mailing list they confirmed their market and secured funding for actually developing the product.[23][24]

Concierge MVP also takes a different route to test the core business hypothesis. This applies especially to service oriented businesses. The service the company is trying to achieve through an cloud service or an application is first tested with a couple of customers in a face to face situation, where an actual person actually does the work and provides the service to customers. This also provides quick insight if the actual concept to be realized with software is valid.

Wizard of Oz is a hybrid application where some of the core functionality of the application is done by humans behind the curtain. For example a cloud based company that specializes in finding the closest shoe shop around you would actually manually search from Google and provide results in the application. This type of MVP can be best utilized for example when a core component of the software is difficult to solve in code, but easy for humans to solve.

The standard MVP not to scale is just a version of your product done fast. This usually means that not the same software solutions are used that you plan to use in your final application. For example HTML5 and document databases can be quickly used to mock up an webservice, and time is not initially spent to produce a perfectly secure, scalable website. Also no time is spent on building a service that could actually scale with a large amount of users as this is not necessary in the early stage when you are not even sure if anyone is going to use it.[25]

The most important aspect in a good MVP is that it produces data about customer behavior so your software and product development decisions can be based on data. This means that your MVP should include enough analytics tools and logging functionality to produce meaningful data of how your customers actually use your software. Some industry standards have been adopted by the Lean Startup and a couple of new ones have been added.

Before implementing logging in your application you should formulate a hypothesis on what you want the metrics to answer as otherwise you could fall victim to vanity metrics. The idea is that too much data helps you find data that partially match your expectations and provide a false picture of your applications performance. By formulating a hypothesis in the type of : "After two weeks we have 20% retention rate" , you provide question before the data.

To measure the performance of your MVP you have to measure it in some meaningful way. The metrics should be actionable in such a way that from them you can see what in your product caused these changes and act accordingly. These are the same hypotheses that were introduced in the last subsection. According to Ries, good values to measure the performance for web services are:[1] [26]

- **Acquisition**: The number of people who visit your landing page.
- **Activation**: Measures the amount of people your web service can activate to become repeat visitors.
- **Retention**: The amount of customers who visit frequently.
- **Referral**: A user who was referred to the site by someone already using it.
- **Revenue**: How much money your web service is producing.

To measure the change in these parameters, you should have the same data from your previous MVP versions. One way of testing two different changes simultaneously is called A/B testing, where the clients of your MVP are divided into two groups and shown a different version of the same product. The alteration between these two versions is the new feature you are developing. This provides two sets of data, which can be compared to see if the new feature really is effective in what it was planned to do and what is the real customer reaction to it.[27]

Collecting real world customer behavior data with your minimum viable products provides you with more information about your business plan and its viability. This enables you to decide whether you should pivot your business plan and repeat the lean build-measure-learn feedback loop.[1][27]

4 CASE MOVENDOS/TAPLIA

The goal of this chapter is to compare and contrast the behaviour of two companies that have used Lean Startup methods in their business. The two companies were selected because the author was working in both of the companies, and their situation is sufficiently different to warrant a comparison between them. The idea is to investigate the effects that the Lean Startup method has had on the business model hypothesis validation in these companies, and how does the business model enable the company to form a growing and profitable business. The investigation was done interviewing the CTOs of both companies, analyzing version control data and contrasting this data with the work experience the author has from both companies.

4.1 Movendos

Movendos is a software startup focusing on creating effective tools for health and wellness coaching, an eight person small software company located at Tampere University of Technology. The main product of the company is the Movendos health coaching platform. The goal is to create an online cloud tool to help coaches to keep better track of their trainees and thus enable cost savings and better service through efficiency improvements.[3][28]

4.1.1 Background

Multiple original business models were created by brainstorming and listing the most viable ones. The original ideas were then withered down to a list of handful potential business models, which were then subjected to a first round of validation by assessing them internally using the business model canvas. The remaining business models hypotheses were validated through trying to find prospective competitors that had basically already validated the business model and also by talking to the prospective customers and finding out if they had a need for such a service. Thus customers were immediately included in the process of choosing the right business model. The consideration of multiple software business hypotheses at the same time can be thought of as a first scan of the market, looking for a niche that could be filled.

After the first round of validation Movendos saw a possibility in the wellness sector. From the initial business hypotheses, the ones in the wellness sector got the best response from clients. Many private trainers in the personal health sector used tools like Heiaheia to track or manage their coachees, but these tools do not help them really to automate away some tedious managerial parts of their jobs. Moreover email was the most common tool used to communicate with the clients, which led to a situtation where most of the client data was actually stored in the email inbox in a non-convenient way. The coaches said a tool combining good ways to track the coachees progress with communication tools was needed.

In the private coaching and wellness business, one of the companies identified, called Corusfit[29] was identified as having a problem of tracking patient exercise outside their own private facilities. Corusfit is a wellness sector company specializing in the rehabilitation and prevention of cardiac arrest. After talks with Corusfit personnel it was identified as a potential early adopter and a pilot agreement with Movendos and Corusfit was created to develop a product MVP of the remote coaching service. This service would be in the shape of a mobile application that participants in the Corusfit training could use to log their spinning interval training and heartrate remotely. The finding of a such early adopter allowed Movendos to actually start focusing on product development.

After the problem fit, the company proceeded to the next phase, where the question is to find the product-market fit of your business hypothesis. The hypothesis was that there is a need for a remote coaching tool that can be used to track the progress of trainees remotely. The first MVP for Corusfit case was decided to be developed with new HTML5 as a hybrid application for Android phones and a relatively new database/backend solution called couchDB. These technologies were selected because they allow the quick prototyping of software using just web technologies that can be then easily ported to other devices. The android platform was chosen because this goes according to the lean methodologies presented earlier that in the beginning you should try to maximize your learning, by minimizing work and not yet building to scale. The product was developed using lean principles in the summer of 2012.[1][25][30][31]

4.1.2 Software development methodology

This subsection summarizes the style of software development and how lean start up thinking has affected it. In Movendos the software development workflow is based on agile principles, mostly Scrum.[17] Software development is split into variable length sprints, which are based on agreements with the customer about release dates and the features that are to be implemented in the next sprint. The sprint ideology is mainly used for the company inside the software development. UX development is done in parallel with software development, but according to Movendos, the UX development cycle does not work in the same way. It was felt that UX development should not be included in the sprints same sprints as software development. At Movendos the CTO plays the role of the Scrum master, Product owner and holds the meetings and organizes the tickets. There are two weekly meetings, which replace the daily meetings, as it was felt that the daily meetings were too frequent, and instead two weekly meetings were implemented.

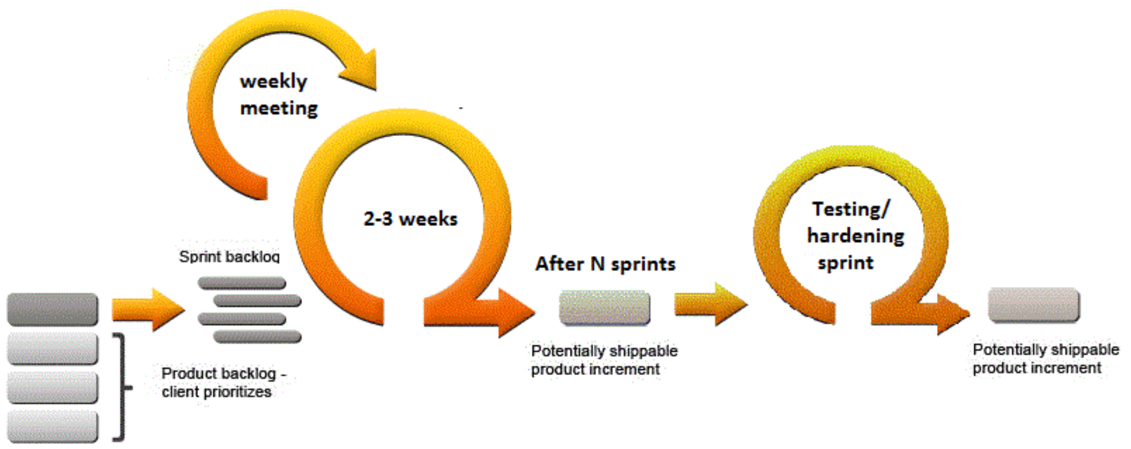

weekly meeting
2-3 weeks
Sprint backlog
After N sprints
Testing/ hardening sprint
Product backlog - client prioritizes
Potentially shippable product increment
Potentially shippable product increment

Figure 4.1: Movendos Software development work flow

The software versions are not also shipped directly to the client after a sprint, but there are dedicated hardening sprints before shipping the product to the client. This way the work is split into smaller deliverables. There were typically three month plans into the future with locked dates for releases that guided the development process. These release dates were preceded by a hardening and testing sprint, which were only focused on the testing of the release version. This development tactic is illustrated in Figure 4.1. The project owner typically creates a development plan for the next 3 months, which guided the more agile feature development.

Before all this, Movendos has an UX workflow, in which the UX person in the company analyzes the results from client tests and prioritizes and designs features for the software development team to create. These features are then also included in the backlog for the sprint. Thus the three month plan and the more agile UX development feeds the software development cycle. This also sometimes results in items in the backlog that ask for a removal of already implemented features. This customer analysis is done concurrently with the agile software development.

4.1.3 Executing Lean Startup

Movendos is a typical tehcnology sector startup which has managed to secure initial funding and is developing software. At the start Movendos had decided to do their software development in a customer centered and agile way and the Lean Startup way was seen as a solution. The chosen area to do software was a in the wellness sector. The software development style in movendos is a variant of scrum, but without daily meetings.

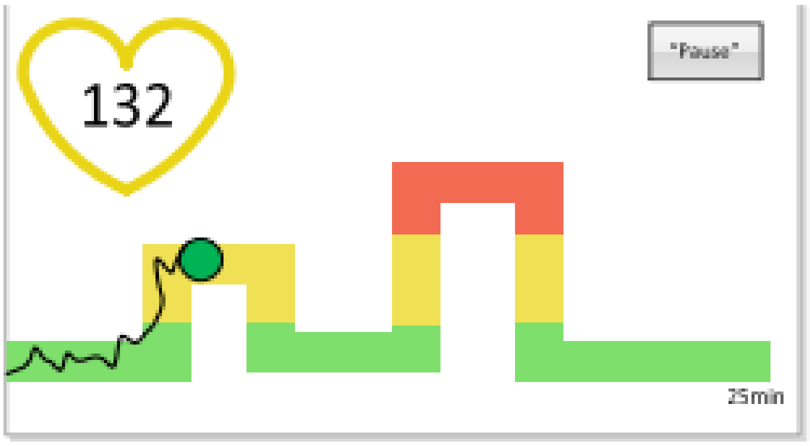
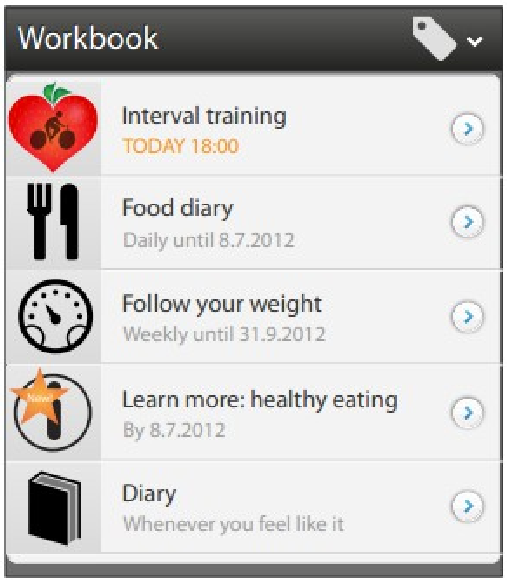


Figure 4.1: CorusFit application

The first product version developed was a cardio training tool for a company called Corusfit. The product was an Android hybrid application. HTML 5 was chosen for the UI development to help release the software on multiple platforms without redoing too much code. The original application used native bluetooth to communicate with heartrate belt to draw a graph of heartrate. Some gamification elements were included into the software, such as a heartrate "track" for the user to navigate through. These first versions were tested with Corusfit and it seemed that they had a need for such software, which would help their customers do home training. A screenshot of the software can be seen in Figure 4.1. In this pilot most of the server side reporting from the backend were mostly faked using a Wizard of oz type implementation. Also many security settings were faked using multiple servers and databases to quickly create isolated environments for different users, even though heavy security was not yet implemented.

The first MVP version of the product was incomplete and did not yet have any web portal features. The first MVP was used to asses that there indeed was a need for software that helps people train at home. After this initial test, the next MVP version of the software was chosen to enlarge the software to include a web portal for a coach to track the training of the customers.

The second MVP version was chosen to be designed as an Android hybrid application with a database and web interface in the cloud to enable the synchronization of exercise data and to enable the coach to track the progress of the users. The web interface and database were implemented with CouchDB. CouchDB was chosen as a database/server solution for the application because of the advertised synchronization features between the android client and web server. Also the simple document database structure and the ability to serve web pages directly from the database enabled quick development of the next MVP. It was known that there would be scalability problems for these technologies, but these were overlooked at this point.



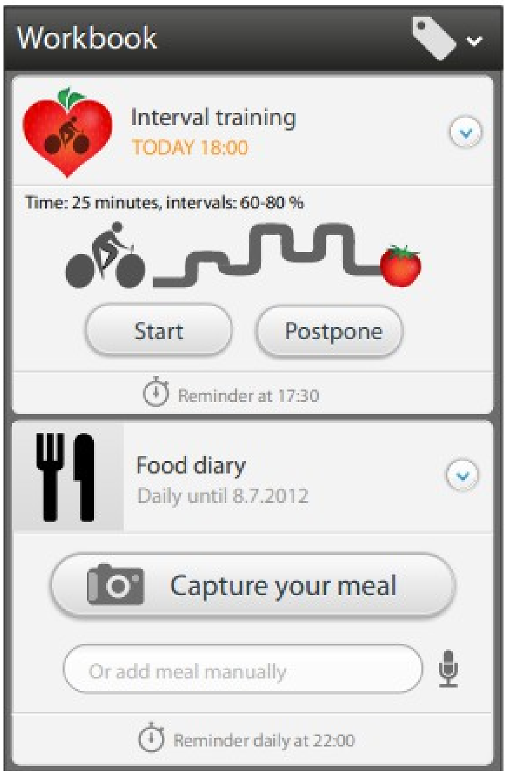
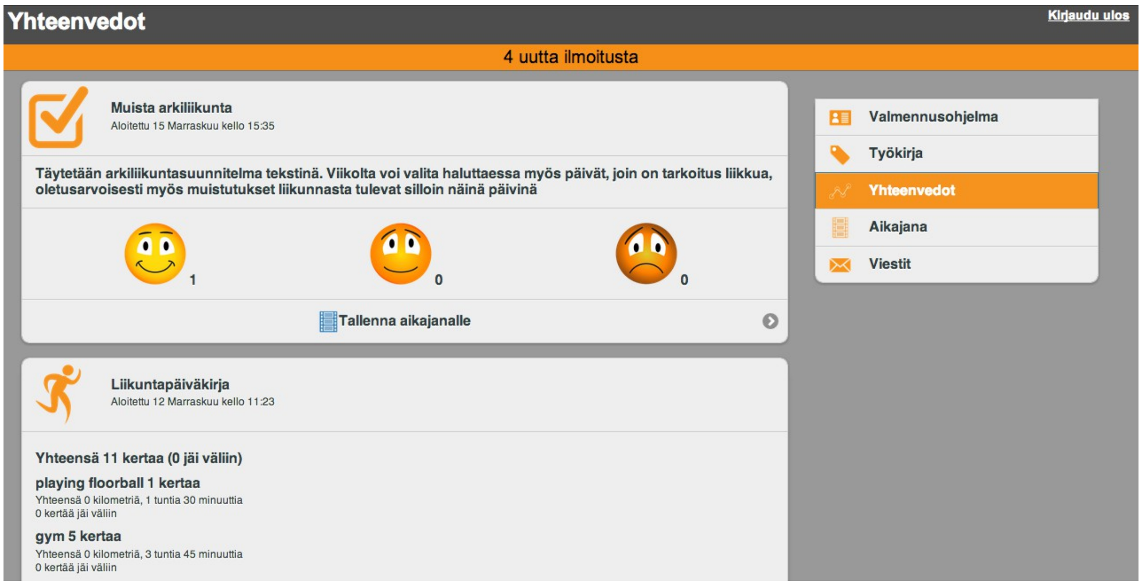


Figure 4.2: Incito Android software

The next version also had a wider set of training tasks and it was expanded so that it could be tested with a wider customer base. Screenshots of the application can be seen in Figure 4.2. This version included a Facebook style workbook. Also a rudimentary server backend for the coaches was implemented, seen in Figure 4.3. This version was piloted with Lassila&Tikanoja[33] and feedback of the application was collected from the testing group. This was done in the form of a questionnaire and phone call consultations.

Based on the experiences, the application was decided to beredesigned as the Android platform into a web application that can be run with any browser on any mobilephone or PC. The last MVP had shown that many people lacked Android phones with wanted operating system versions or used a different brand of mobilephone. There also were some problems with the actual platform and tools used to create the software that were causing software development problems. The software used for the backend server was not performing as wanted and was determined to be a bottleneck going on forwards. The core business model on the application was however deemed to be confirmed on the pilot, most of the problems identified were to do with the application, not the core idea.



Muista arkiliikunta
Aloitettu 15 Marraskuu kello 15:35

Täytetään arkiliikuntasuunnitelma tekstinä. Viikolta voi valita haluttaessa myös päivät, join on tarkoitus liikkua, oletusarvoisesti myös muistutukset liikunnasta tulevat silloin näinä päivinä

Liikuntapäiväkirja
Aloitettu 12 Marraskuu kello 11:23

Yhteensä 11 kertaa (0 jäi väliin)
playing floorball 1 kertaa
Yhteensä 0 kilometriä, 1 tuntia 30 minuuttia
0 kertää jäi väliin
gym 5 kertaa
Yhteensä 0 kilometriä, 3 tuntia 45 minuuttia
0 kertää jäi väliin

Figure 4.3: Incito web interview

The success in testing of the previous version was seen as a sign that the business could be a success. The next version of the software was to be done with heavier investment on scalability. Technologies for the next version of the product were chosen with the help of an outside software contractor Vincit, who have experience on deploying web based services, As the technologies chosen for the next iteration of the software were a Java spring back end solution[32][34], with a single page application style HTML5 page, with backbone functioning as the front end base. This software was to be run on the amazon elastic beanstalk for scalability. The pivot away from just Android platform to the web application based platform can be considered a zoom out pivot, where the original business idea was expanded from the MVP Android ecosystem to be accessible with all smart phones and devices. Also at this point the original data collection facilities were removed from the application. These depended on platform specific code on the mobile devices and were also shown not to be as crucial to the original business case. The application was pivoted to be a more diary / training log based, where the user inputs all the data, for example how many kilometers he jogged. Also the communication between coaches and clients inside the application was improved upon, so no external communications channels were necessary. The development of this version was done with a larger developer team with additional workforce from Vincit and the inhouse developers. This was done in the Movendos agile style outlined before. This version of the software has been sold to a couple of clients and continues to operate as the core product of Movendos. Screenshots of the application can be seen in Figure 4.4.

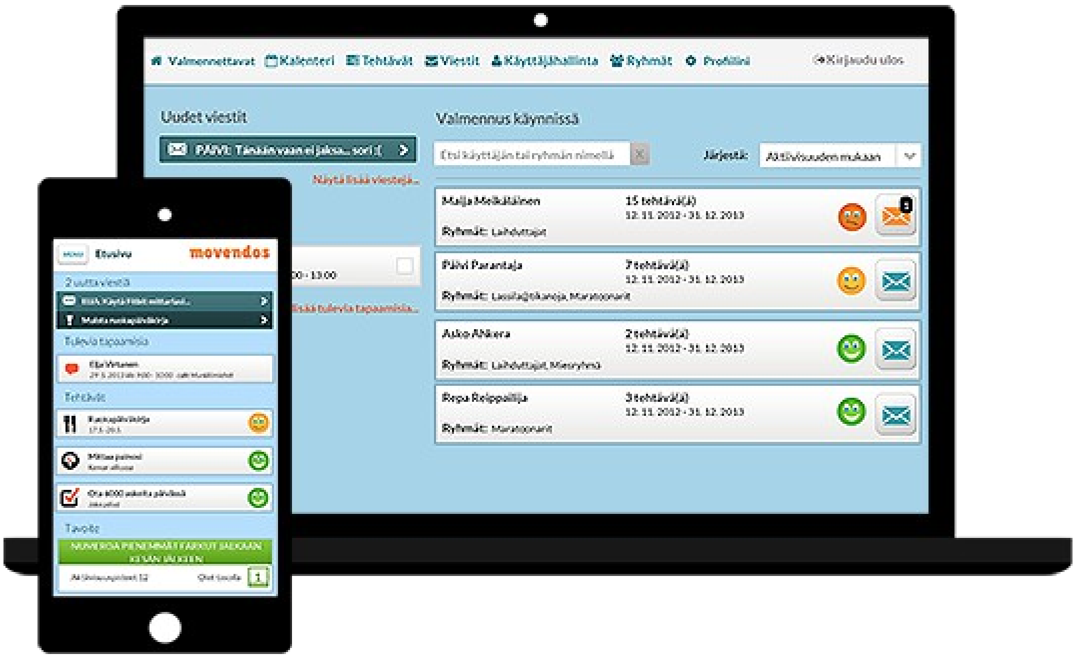


Figure 4.4: Movendos web software

4.1.4 Evaluation of Lean Startup

The workflow in the software development side and product development side were influenced by the Lean Startup method. In the beginning, many concepts were tested with paper prototypes and subjected to a lean canvas analysis. When a potential market was found, a possible client was contacted and the first MVP done in close collaboration with the client. This inclusion of the client from the start helped confirm that the product actually served a need. In hindsight, only using one client and their localized needs had the risk of producing a product for a local maxima, which means that only the one client or few clients actually had the same need.

The above problem was avoided by trying to generalize the product and selling the original MVP as a part of the new incito mobile workout/workbook application. This reduced the corus fit cardio tool to just one feature of the whole. Other features like gym repetition trackers and food diaries were tested with this product. This was in a way MVP feature testing. The modular structure of the incito application and the fluidity of the backend allowed Movendos to quickly test out different “product concepts” within the one product. This testing allowed the company to steer the software development from data of how the different components were used.

Figure 4.5 shows at an abstract level how the development time and resources were split between the different product versions over time. There were three distinct products which were Corusfit, Incito and Movendos web. The iterations can be clearly seen. There was some overlap between versions as the development for the next version was started in conjunction with the previous version. [28]

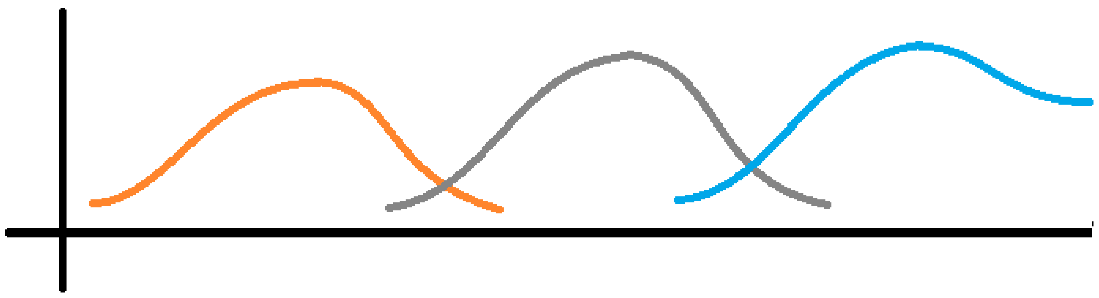


Figure 4.5: Movendos perceived work graph

Because of the limits of the MVP implementation, the software solutions used in the first MVPs were starting to become a hindrance and as these were core software development choices, the Incito software had to be completely redone. Also because this sofware was originally built for logging heartrate it used a large amount Android platform specific code. These features were cut from the later versions and thus work used on these parts was wasted.

The new version therefore had nearly zero code reuse, and the MVP was labeled as a more of a throw away prototype. This resulted in a large amount of wasted work effort as the key components were redone. But this also allowed a pivot into a much more scalable software solution stack. Also the original amount of work used in the original android MVP was minimized through the process as the the possibility of wasting that code was kept in mind. The opinion of the CTO of Movendos was that the original transition from Corusfit to Incito system saved code, but the Incito MVP was allowed to grow too large before it was cut. It was aknowledged that the technical solutions in the Incito MVP were not scalable. For example the database selection of using CouchDB was known to have problems in larger deployments, but as the product was envisioned to be a quick minimum viable product, it was not deemed to be a problem in the beginning. As can be seen from the Table 4.1 The code reuse to the last version was practically zero and all the features had to be redone. The code reuse statistics were based on an approximation by the CTO of Movendos. To support this, the amount of code in each version were checked from the version control systems of Movendos. The percentage tries to take into account the boilerplate code for the different platforms, such as Android and Java Spring. This code The question of how much work was saved by doing MVP minimum versions on the android first is a question that is not easily answered without actually trying to do the Incito system again from scratch without cutting corners in development and choosing faster to implement methods. Also as the goals of the different products were not the same as they were developed at different points in time and the company had different opinions about the business space and the product they should develop for it.

Table 4.1: code reuse between versions

Product	Based on	% of code reuse(approx.)
CorusFit MVP 1	-	-
Incito MVP 2	CorusFit MVP 1	50*
Movendos web	Incito MVP 2	0

On the other hand the validated learning milestones from these products was achieved in the opinion of the CTO of Movendos. From every version Movendos learned more about the business space they operate in. On the first CorusFit MVP the goal was to check if the mobile logging was a working concept for software. This goal was achieved in less than three months. The second Incito system was a test for the additional features of logging weight, food diary and whether these increase the perceived value of the software. This was achieved through the Lassila&Tikanoja pilot. Also the facebook like timeline was noted to produce no value and was removed. This allowed Movendos to move on to the Movendos web with confidence.

As a startup company the environment of extreme uncertainty from before MVP 1 has changed and it is the opinion of Movendos that they know much more about their business space now than in the beginning. The development of multiple versions let them zoom out the business concept and test its viability in phases. This allowed for adaptation of the original plans based on what was learned from the previous versions. Overall the opinion of Movendos towards Lean Startup method was positive.

One aspect that was not tested in the MVP solutions was pricing. The Lean Startup advocates for immediately charging for your product, even though it is just a minimum viable product. This also validates the monetization side of the product. This was not done on the first two MVP products and thus the viability of the monetization strategy for the Movendos product was still in a state of uncertainty.[28]

4.2 Taplia

Taplia is a small software company founded by three Tampere University of Technology students in their free time. The company started of from a need perceived by one of the employees in his previous job working as an electrical installer.

4.2.1 Background

The core business of Taplia is to create simple and lightweight software that can be used to automate the logging of work hours. The core idea was to make the tracking more efficient by removing paper shuffling and manual entering of all the hour slips into a payment system. Also customization on a per client basis is done for an additional price. The company is a small part-time three man operation and was started without any additional funding or angel investors. 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 upon. Some business cases have been tested and validated through lean ideologies, but it still remains to be seen if the current business model is the final one.

The top problems the company's product is trying to solve are time and money spent on regular time slip collection and payments. Time is saved when the hour logging is done automatically and time is saved when the system automatically calculates the totals for the wage department. Also no money and time is wasted collecting typical paper payment slips from different building sites in the construction industry.[2].

The target segment of the company is the construction industry. The unique value proposition of Taplia is ease of use. Taplia prioritize ease of use over features to produce easy to use systems. In the solution box Taplia has put the easy to use mobile design, automatic data analysis and simple feature set for a small price. The sales channels being utilized are currently industry contacts, viral sales and direct marketing. The revenue model in Taplia's product is centered upon SaaS concepts, offering the basic product version with a monthly license to its users. Another revenue model is to fund new feature development through customers that want those features by offering customization as an option to the customers. This enables testing new features with customers money.

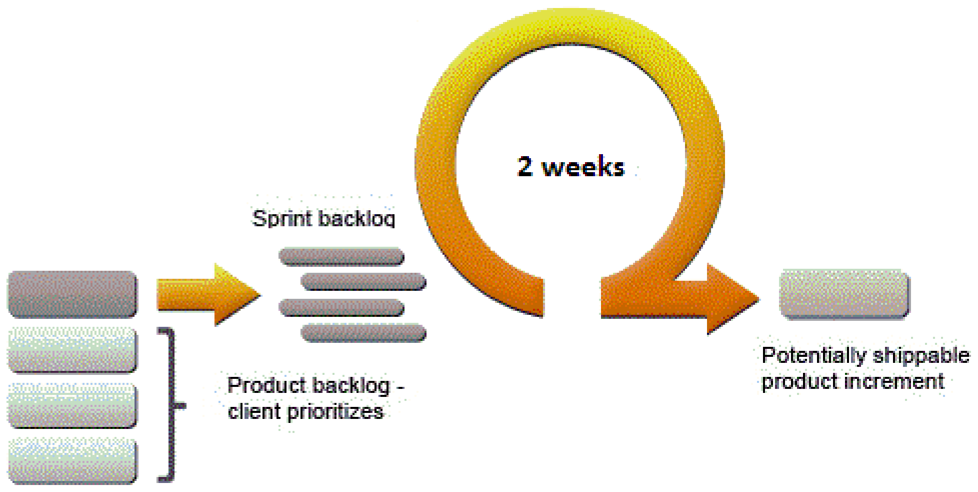
In contrast to Movendos, Taplia found its original hypothesis from a problem one of the founders had experienced first hand and has seen the problem being faced in many different companies in the building industry. The building industry has a problem, that logging and storing of working hour information is still done by paper in many companies. The companies usually employ secretaries whose only job is to calculate and collect the working hour slips and pay that is to be paid to employees. This seemed like a place to use software to make the automated collection of hour slips a reality and to automatically produce total working hours for employees that could be used to pay the correct amount of salary. This problem was verified by talking to multiple companies in the industry and a business plan was formulated for the company as a part of Tampere University of Technology course. The base business hypothesis is also already validated by large companies like SAP[35], but their focus is on larger companies. The small companies that don't have much resources are a niche market that is still open for competition.

After formulating the business plan a couple of companies were interviewed to find companies that are early adopters and were willing to pilot test the system. A company called Sähköansio[36] was one of the companies interviewed and identified as the most potential early adopter as they were eager to start developing a system for their needs. A piloting deal was made with the company to produce a pilot system where employees can fill in their hours by using a mobile application and the system outputs a spreadsheet for the secretary to inspect.

4.2.2 Software development methodology

The software development style being used in Taplia is a one variant of agile software development, mostly based on scrum. The main software development is split into variable length sprints, which are based on agreements with the customer and the features that are to be implemented in the next sprint. The tasks under the sprint are chosen after a meeting with the client after the last sprint. The tasks are not then split up immediately for the different coders, but picked from the sprint backlog by the individual developers directly. Because of the small size of the company and the developing team, no dedicated daily sprint meetings are held. After a sprint the increment is shipped to the client and put into usage immediately.

The introduction of Lean Startup principles into this software development practice does not actually change the agile development too much directly, mainly it changes the scope of the requirements. Also the idea in scrum is that the customers can change their minds on what they want, and the implementation priorities can change. This is illustrated in Figure 4.6.


2 weeks
Sprint backlog
Product backlog - client prioritizes
Potentially shippable product increment
Figure 4.6: Taplia software development work flow

This integrates well with the Lean Startup where the requirements can change because of new insights about the customer behaviour achieved through analyzing the usage of the last product increment. In this way the product development cycle is not changed by the introduction of lean, only the contents of the backlog and their justifications change.

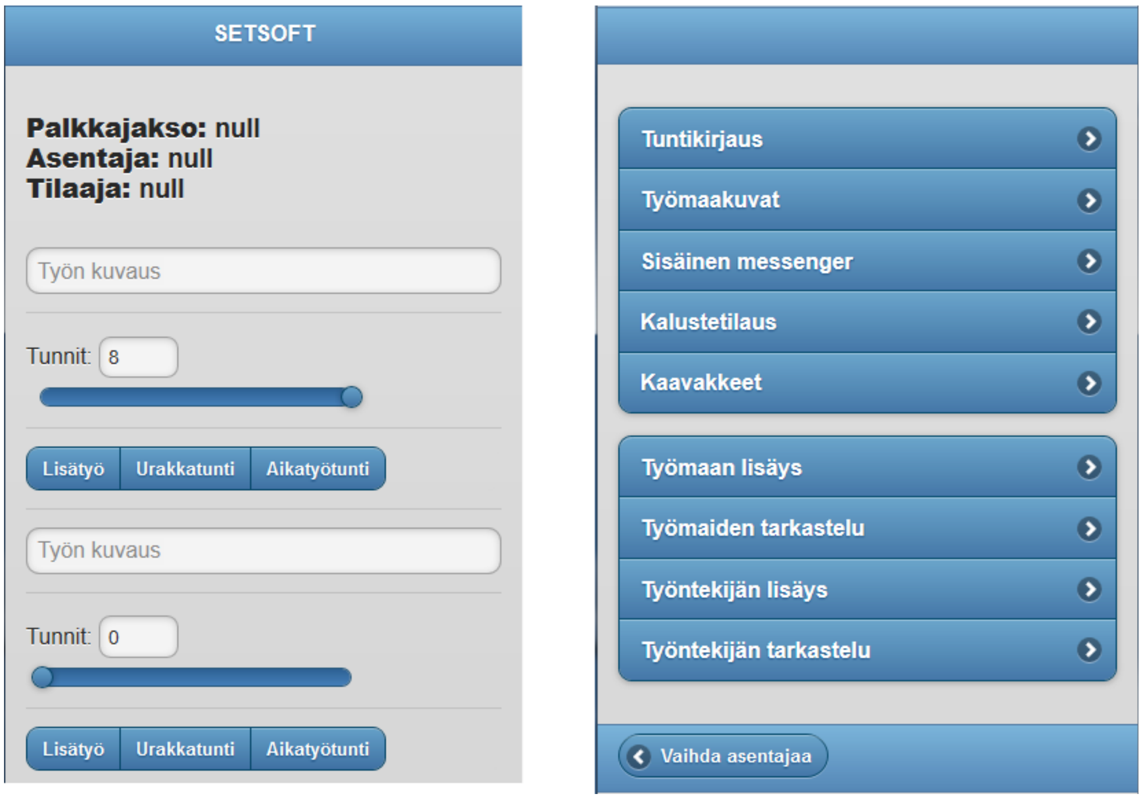
Some adjustments were done by lean because of the minimum viable product ideology. Most of the technologies chosen for implementation were chosen because they enabled quick implementation in favor of scaling and features. The first products were done with web technologies for quick deployment and because of their usage in MVP's, minimum amount of work.

4.2.3 Executing Lean Startup

Taplia is a prime example of a company with really limited resources operating in an unknown business space under conditions of extreme uncertainty. At the company the lean model was adopted to cope with the realities of not knowing hard facts about the business space and because of the limited amount of resources available for the company.

The original business-hypothesis of building a simple hour logging software for small construction companies was decided to be tested with a minimum viable product. The scope of the product was narrowed to just the core of logging working hours using a mobile device.

The first product of Taplia was a software as a service(SaaS) based hour logging portal, which enabled the users to log their working hours in a web portal. Web technologies were chosen because they allowed the company to quickly deploy an application with their existing skillset. HTML5 based web application was something the developers had experience with, so the costs of learning new programming languages and frameworks was reduced. Jquery mobile was used to create a webpage that worked efficiently with mobile handsets and also with the desktop. CouchDB was used as a service, because it allowed to serve the web pages directly from the database, without a dedicated server. This allowed for a quick deployment of the first version in under a month. The product was done in a customer centric way with Sähköansio as a piloting company. [37]


Palkkajakso: null
Asentaja: null
Tilaaja: null

Tunnit: 8

Tunnit: 0

Figure 4.7: Sähköansio MVP software

As can be seen from Figure 4.7, the MVP did not focus on aesthetics or usability issues, it just delivered a simple tool for logging working hours that can be used with a mobile device. The ui elements used here were done with premade libraries such as jQuery mobile. The MVP only focused on finding out if providing a tool for easily logging working hours is a feasible business model.

At the same time as the Sähköansio product was being developed, Taplia got a contact from a Finnish gym company called Fressi[38]. The company was also on the lookout for an hour logging system, but using physical devices and RFID cards for authentication. This was also taken into development, by adapting the same Sähköansio code to now log in with card from the reader, not usernames.

At this point Taplia switched to a more classical agile development, where the requirements from the two clients were assessed and a backlog of features was created for the next sprint for the two products. The Fressi MVP version and the Sähköansio version would be done as separate copies from the original MVP to speed up development. This software rebranding can be seen in Figure 4.8.

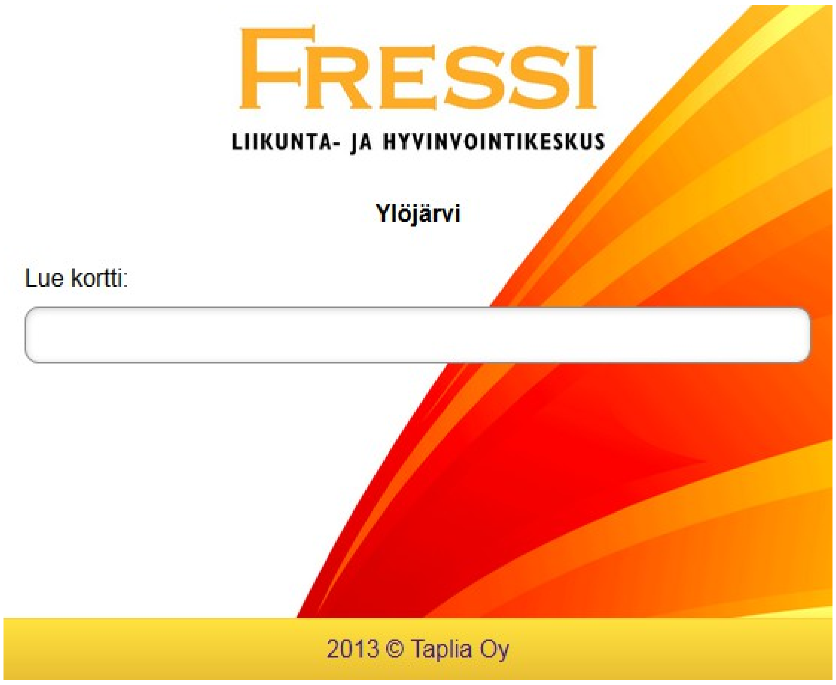
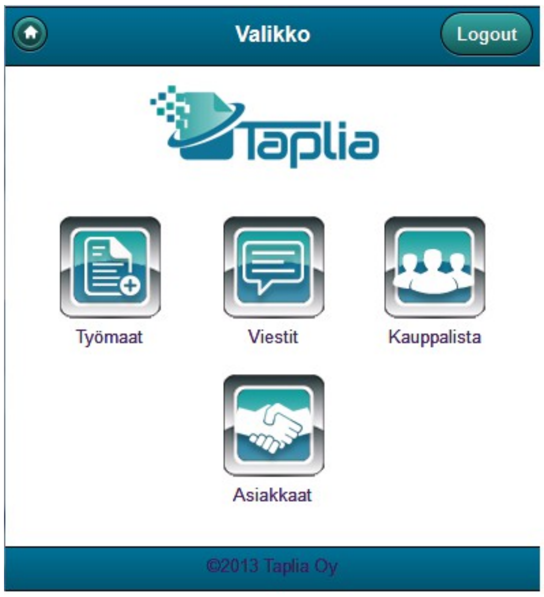
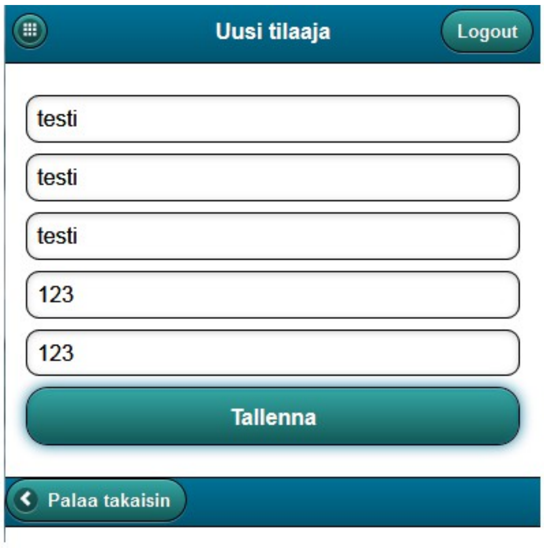


Figure 4.8: Fressi MVP product

The two products were developed independently of each other, and resulted in Taplia having to manage two differing projects in their development. According to interviews with the company CEO and CTO, this was a problem as the company only has 3 members, 2 full time, so the limiting factor for the company really is the workforce. The division of how to split the workload of 3 persons between two differing MVP products to be tested was seen as a major hindrance at this point.





testi
testi
testi
123
123

Figure 4.9: Taplia sähköansio product

New MVP iterations of both were done and the features were added to both, some in mimimum viable feature style, where no actual automation was done in the background but the work was delegated to humans. At this point Taplia wanted to know if their business model of selling a customizable hour logging software was viable with other companies. The second Sähköansio minimum viable product was shown to a third company, called Betoco, which also agreed to buy the software from Taplia with some customizations. This is illustrated in Figure 4.9.

This resulted in a situation where Taplia now had three differing product versions and only limited amount of resources available. Because of the MVP way the three products were done, with minimal amount of work, the codelines had drifted further apart from each other and most of the code was not scalable. At this time it was decided that the three codelines should be merged and additional effort to be made to create a single codeline, which would have all the features that the products need. In addition, extra work would be done to ensure that the code would be of good quality. So at the point that minimum viable product iteration had enabled the company to build three successfully working products with quick coding. The transition phase to a more easily maintainable code base needed more resources, but because Taplia failed to secure external funding and because the MVP pricing structure for the client was not finalized, the resource bottlenecks were seen as the next big problem, caused partly by lean ideology.

4.2.4 Evaluation of Lean Startup

The use of lean techniques in Taplia is centered around the usage of MVPs to test new products and if they match the customer need. During interviews with the board of Taplia it was deemed that the key limiting factor in running Lean Startup MVPs and testing was the amount of resources the company has. The three man team could not effectively create minimum viable products that would be viable. Also one problem was that shutting down MVP products which were already sold to customer was quite difficult.

When the amount of work spent on the different versions of software were examined, the following perceived amount of work graph was created which can be seen in Figure 4.10. This shows how the work was split between the different product versions that the company had. The two Sähköansio versions are shown in blue, Fressi in orange and Betoco in green. As can be seen the focus of the team switched between the versions but there was always some overlapping with the projects.

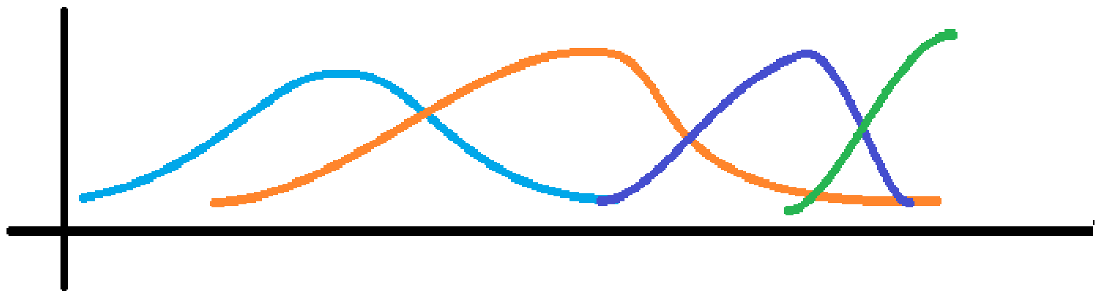


Figure 4.10: Taplia perceived work graph

The three man team had multiple versions of the application to maintain which generated problems when deploying new versions of the products to test new features. This also led to many features being omitted and some of them produced in a Wizard of oz way. For example PDF logging was at first done by hand from the database by Taplia members. This way of developing and testing features also added to the workload of the small team.

The MVP ideology, where the first versions were not built to scale or with proper solutions and software techniques, actually led to situations where a significant part of the MVP code had to be remade to actually perform as the clients demanded. The code reuse between the projects was evaluated and is shown in Table 4.2. This is based on the approximation of Taplia CTO and the commit histories from the version control system of Taplia. For Taplia, a technology change only happened between the Sähköansio MVP 1 and Fressi MVP 1. Subsequent products were done using the same technologies. Most of the HTML content was easily portable to the next version.

Table 4.2: Taplia code reuse between versions

Product	Based on	% of code reuse(approx.)
Sähköansio MVP 1	-	-
Fressi MVP 1	Sähköansio MVP 1	40
Sähköansio Product 1	Sähköansio MVP 1	80
Betoco Product 1	Sähköansio Product 1	50

Taplia identified that one main problem with the Lean Startup method was that it was extremely difficult to judge what is actually a minimum viable product. For example in the Sähköansio hour logging software there were many features. These features ranged from inputting the data easily, reviewing the data, authorizing the data and exporting the data from the system. The question of what features of the product actually make it viable and what parts can be left out was a constant stepping stone and no good solutions were arrived upon.But the usability side of these features was considered as the main selling point for Taplia. How can you measure when the usability of a feature is viable? The Sähköansio product currently has over 50 features, so can it still be considered an minimum viable product?

Also the recurring problem seemed to be the limit on manpower. No A/B testing for new features was done because there was actually no one to do them. Continuous integration was planned, but there was no time to actually implement it at any point, because of the demand for multiple different versions from the clients. In the interview it was also said that the Lean Startup MVP ideology behaved in an unexpected way for the company in the sense that both of the experiments succeeded, but the product could not easily be integrated into each other because of the shortcuts taken in the development phase to get them out quickly.

Other factor that Taplia was concerned was that the product had actually been tested with just two early adopters and even they both had slightly differing requirements. Taplias business model was changed to reflect this need to customize the product to each customers differing needs, and the customization work was billed from the customer. Taplia still does not know, if their core is generic enough. The business model that would require minor customizations to the core software per clients as opposed to creating the software from scratch is still to be tested.[40]

5 DISCUSSION

In this chapter we go through the opinions and experiences of both case study companies and form a general view of the Lean Startup method and its usage in a startup environment. Some key points of its usage and relation to software development are discussed here.

5.1 Iteration and the build-measure-learn cycle

The fast iteration and customer centric development was seen as the main positive factor in both companies. Both companies felt that the iteration and pivoting helped them find a product-market fit and both of the companies had a sellable product, that had paying customers. In the case of Movendos, they have two main clients for their software, and Taplia has three different clients. It was felt that the Lean Startup provided tools and practices that can be used to maximize learning while minimizing the amount of resources used.

In the two companies presented, the iteration in the two companies was mostly minimum viable product iteration, not feature iteration. This led to major changes between versions and the learning between the versions was not as strong as it could have been if the changes between versions would have been smaller so that the change in usage of the software could be more directly linked to certain changes in the software. But as the purpose of minimum viable product iteration and minimum feature iteration is different, you should choose the right tool for the job needed. These tools allowed to test either the whole product on larger scale or smaller feature implementations. It can be said that the minimum viable product can be used to test the product, the minimum viable feature can then be used to accelerate the development of a found potential product. These ideas integrate well into agile development, where the minimum viable features can be implemented in one sprint.

5.2 Minimum viable product development

The MVP model of quickly creating small products to test if the company is solving a problem worth solving with its software was perceived as a good tool. The minimization of work to test a hypothesis allowed both companies to quickly produce a working base model product that could be given to clients for testing. This resulted in immediate feedback from the client about the features currently in the software. According to the companies this allowed them to not commit to a certain path before the business logic of the software had been verified.

The amount of code reuse in the MVP models was sometimes seen as a problem, but as the minimum viable products had not been done using quality coding, this technical debt when reused might have caused problems on the long run. The implementation choices for a minimum viable product are different from an actual software point of view. Most of the features are not actually done, or are faked using the different MVP strategies. The implementation favors tools that help you to quickly create test software, not engineer a scalable software solution. For example the original Movendos Incito system used a different database for every user to create a secure environment. This creates a large overhead per user, but when only used to test a small amount of users this can be ignored.

Based on the comments of both companies, it would seem that the transition from the minimum product to an actual product is the most difficult point in the Lean Startup method. The different requirements of quickly implementing a product and implementing a product properly result in differing requirements and choices in software development, which typically are not in line with each other. Minimum viable product coding usually eschews software life cycle demands such as Maintainability and good coding practices to ship code faster to test hypotheses faster. These usually lead to decisions that favor quick deployment, not long term maintainability. Products that are used by clients for a long time typically require better life cycle management to reduce the load on the developers further down the road. The minimum viable product development style is opposed to this and if an minimum viable product stays in production for a long time, these problems start to manifest. This can be already seen in products of Taplia, where some hastily done versions are still in use by clients. Also the same problems started to manifest in the products of Movendos, where one product version was in development for a long time. The quick database solutions chosen in the beginning started to create problems in a larger scale deployment as they were not designed to it and many additional work hours were sunk into creating custom solutions to problems caused by the choice of a database.

5.3 Resource intensiveness

The main problems that both companies reported were the resource intensiveness of iteration and continuous testing and the MVP. As both companies are startups the main bottleneck is the resources they have. Extensive A/B testing and repeated experiments described in the Lean Startup seem to be a better fit to large companies, or venture capital funded startups, that have extra resources available. Both companies also had problems securing additional funding when a potential proper product was found. Neither of the companies did A/B testing on actively as they felt that they did not have the time to create A/B testing sites and no ready frameworks for this which could be used to easily integrate A/B testing into the products. The companies felt that the amount of coding required to do proper A/B testing did not yet provide enough learning compared to other methods, like feature MVPs that it was warranted.

5.4 Product validity and learning

The concept of validated learning and hypotheses was seen as a strong point in both companies. When you define what you should learn from this version of the software beforehand you solidify the goals of that minimum viable product version. Also it enabled learning goals to be better monitored and learning did not devolve into an explanation for a failed product prototype that it commonly is. Some problems arose on the area of how to define the learning goals effectively. This sometimes resulted in more learning that was in retrospect deemed sufficient. The validation for the amount of learning in the learning goals has to be better thought out to keep the size of the learning goals and the work that goes into them as reasonable.

Another problem seen in both companies was how to measure when a minimum viable product was actually viable and could be used to measure how the core ideas of the actual product would work. This problem led to one of the Movendos products, the incito system to grow really large. This resulted in a large amount of throw away code. The question of how to validate validity is left for the personnel in the company to decide and sometimes it can be difficult to asses even in retrospect. Also it should be noted that the choice of a hypothesis for the minimum viable product affects the scope of the MVP. For example in the Movendos product, large segments of implementation were changed between MVP models and this resulted in a system, where it was difficult to say what effects all the changes had. To control this the hypotheses should be kept limited in scope and the changes between version should be clearly thought and minimized so that causal relations between the different changes and user behaviour can be more easily mapped.

These questions also point us towards an another problem, what are the main features in the minimum viable product? The main problem in the validity of a minimum viable product is that it tries to emulate a fully featured software with limited features. This-causes a a situation where the success of the minimum viable product is dependent on the fact that the producers can succesfully identify the core features of their software for the client. But as the normal expected environment for creating a minimum viable product is a startup, which as stated operates under uncertainty of their market space and thus their clients, this judgement is mostly based on experience in other areas and guesswork.

5.5 Product pricing

Also one important factor that affects the product is its pricing. If the pricing strategy is not part of the minimum product, it does not really reflect how the product is perceived for the client. This is something that both Taplia and Movendos had noticed, as offering nearly free pilots of a software to clients does not really emulate a product with a price and the transition into a paid product was difficult. Taplia had some problems with Sähköansio in transitioning from the MVP phase to a paid product phase, as the client was at first receiving piloting discounts. For the new price the software was not deemed to be as interesting and crucial for the company as when priced with the lower MVP piloting prices. The same problems were also noticed in Movendos, where they could not raise the price of the usage of their software, even though they added features which were perceived to add customer value from the data collected on software usage.

6 CONCLUSION

The purpose of this thesis was to analyze the validity of the Lean Startup and its iterative software development method as a way to develop software into a commercial product in a startup environment. The study was done based on a case study of two companies, which were interviewed and their version control data inspected. Also the experiences of the author in both companies were used to contrast this data. The backbone for the analysis was the version control data and release cycles of both companies for their products. The interviews were also used to learn more of the context of both startups. This made it easier to compare and contrast the Lean Startup usage in both companies.

In conclusion the case studies of the Lean Startup approaches in two Finnish startup companies found that both companies felt that the Lean Startup tools were useful and helped them get their software business running faster. The learning gained from the MVP methodology combined with customer centered agile software strategy seems to be a good way to start a startup with limited resources and knowledge of their business space. MVP's can quickly be used to check the core business idea for validity and avoid an early wrong choice. When complemented with iteration on product features, companies can quickly iterate towards a program that produces value to the customer.

The planning of the MVP should be done with care as to limit its scope and so that it answers the wanted hypothesis questions. The problems seen in both companies on the scope of the MVP could be seen as implementing it with more traditional software engineering tools and methods, which caused the projects to bloat. Also as neither of the products were simple one page application or simple mobile applications that can be quickly iterated, as the movendos sports tracking system is a quite large product. More product iteration can be seen on the Taplia side, but a different problem arose, which is how to phase out the now unneeded MVP versions, but which are used by paying customers? This caused Taplia much extra work.

A negative factor about the Lean Startup is its resource intensiveness in the Finnish software environment. Both companies felt that they could not afford to do full A/B testing or an MVP feature for everything that they wanted to test. The lack of easy tooling also made this step more expensive. This resource intensiveness is also mentioned when Snaptax reported that they run over 500 new tests every year. This would seem like that the full Lean Startup approach is quite resource intensive and it is better wielded by larger companies exploring a new business space. The effect Lean Startup has on the learning speed of a startup is its most significant aspect, this is applicable to both smaller and larger startups.

One thing to note in both companies was the extensive use of premade frameworks and toolkits to use as many premade components as possible. It could be even argued that the frameworks and libraries, such as jQuery, Backbone and Ruby on rails which enable faster development by supplying premade components made the faster development cycle possible. If similar toolkits would be developed for other aspects of the Lean Startup, the problems points such as the resource intensiveness of multiple MVP iteration and A/B testing could be alleviated.

This intensiveness points towards the fact that there seems to be to a concept of minimum size startup. There is a minimum size for a startup that is needed to run the whole Lean Startup cycle and all its features. The need to do multiple iterations on the product, collect data between different versions and analyze the data is really demanding work for resource deprived startups and this does not seem to be achievable to full effect in the Finnish sub five person startups. As such the Lean Startup method should not be taken as total method for developing software, but rather as a toolkit from which to pick tools that suit your software development product.

7 BIBLIOGRAPHY

[1] Eric Ries. *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses* , Crown Business, 2011

[2] Taplia. Available: http://www.taplia.com . Referenced: 13.5.2015.

[3] Movendos. Available: http://www.movendos.com. Referenced: 13.5.2015.

[4] Venture Capital. *Eric Ries, author of "The Lean Startup".* Available: http://www.youtube.com/watch?v=EMysvIXmbl0 . Referenced: 13.5.2015.

[5] Reuters. *"Lean Startup" evangelist Eric Ries is just getting started.* Available: http://blogs.reuters.com/small-business/2011/05/26/lean-startup-evangelist-eric-ries-is-just-getting-started/ . Referenced: 13.5.2015.

[6] Taiichi Ohno. *Toyota Production System: Beyond Large-Scale Production*. Productivity Press, 1988

[7] Layla Foord. *How We Built a Lean Startup Inside a 200 Person Company.* Available: http://inside.envato.com/how-we-built-a-lean-startup-inside-a-200-person-company/ . Referenced: 13.5.2015.

[8] Andres Klinger. *Why Lean Startup sucks for startups.* Available: http://klinger.io/post/69794653694/why-lean-startup-sucks-for-startups?utm_source=buffer&utm_campaign=Buffer&utm_content=buffer9ddee&utm_medium=linkedin#!. Referenced: 13.5.2015.

[9] Michael Sharkey. *6 things wrong with the 'Lean Startup' model (and what to do about it).* Available: http://venturebeat.com/2013/10/16/lean-startups-boo/ . Referenced: 13.5.2015.

[10] Technology review. *Intuit's big refresh.* Available: http://www.technologyreview.com/news/423642/intuits-big-refresh . Referenced: 13.5.2015.

[11] Steve Blank,Bob Dorf. *The Startup Owner's Manual.* K&S Ranch 2012

[12] Harvard Business review. *Is Entrepreneurship a Management Science.* Available: http://blogs.hbr.org/2010/01/is-entrepreneurship-a-manageme/ . Referenced: 13.5.2015.

[13] Ash Maurya. *Running lean.* O'Reilly Media. 2012

[14] Ash Maurya. *Achieving Flow in a Lean Startup.* Available: http://practicetrumpstheory.com/2009/12/achieving-flow-in-a-lean-startup/ . Referenced: 13.5.2015.

[15] Olivier Serrat. *The Five Whys Technique.* Available: http://www.adb.org/sites/default/files/pub/2009/the-five-whys-technique.pdf . Referenced: 13.5.2015.

[16] Kent Beck, Mike Beedle, Are van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas. *Manifesto for Agile Software Development.* Available: http://www.agilemanifesto.org/ . Referenced: 13.5.2015.

[17] Hirotaka Takeuchi, Ikujiro Nonaka. *The new new product development game.* Available: http://hbr.org/1986/01/the-new-new-product-development-game/ar/1 . Referenced: 13.5.2015.

[18] Kent Beck. *Extreme Programming Explained: Embrace Change*. Addison-Wesley, 2004

[19] Dean Leffingwell. *Scaling software agility: Best practices for large enterprises.* Addison-Wesley Professional, 2007.

[20] Kathy Sierra. *Featuritis vs. the Happy User Peak.* Available: http://headrush.typepad.com/creating_passionate_users/2005/06/featuritis_vs_t.html . Referenced: 13.5.2015.

[21] Zuly Gonzalez. *Why the hurry to launch an MVP.* Available: http://businessofsoftware.org/2012/06/why-the-hurry-to-launch-an-mvp/ . Referenced: 13.5.2015.

[22] Winston W. Royce. *Managing the development of large software systems.* Available:http://leadinganswers.typepad.com/leading_answers/files/original_waterfall_paper_winston_royce.pdf . Referenced: 13.5.2015.

[23] Dropbox. Available: http://www.dropbox.com . Referenced: 13.5.2015.

[24] Dropbox. *DropBox Demo*. Available: https://www.youtube.com/watch?v=7QmCUDHpNzE . Referenced: 13.5.2015.

[25] Henri Terho. *Using NoSQL database solutions in mobile healhcare applications.* (Bachelor's Thesis), Tampere university of technology, 2013.

[26] Dave McClure. *Startup Metrics For Pirates.* Available: http://500hats.typepad.com/500blogs/2007/09/startup-metrics.html . Referenced: 13.5.2015.

[27] Ville Kautto. *Mittaaminen ja validoitu oppiminen verkkopalvelun kehityksessä*, (Master's Thesis). Tampere university of technology 2012.

[28] Stefan Baggström (2013, November) Personal Interview.

[29] Corusfit. Available: http://www.corusfit.com/ . Referenced: 13.5.2015.

[30] W3C. *HTML5.* Available: http://www.w3.org/TR/html5/ . Referenced: 13.5.2015.

[31] Apache Software foundation. *CouchDB.* Available: http://couchdb.apache.org/ . Referenced: 13.5.2015.

[32] Pivotal Software. *Spring Framework.* Available: https://spring.io/ . Referenced: 13.5.2015.

[33] Lassila&Tikanoja. Available: http://www.lassila-tikanoja.fi/ . Referenced: 13.5.2015.

[34] Vincit. Available: http://www.vincit.fi/ . Referenced: 13.5.2015.

[35] SAP AG. Available: http://www.sap.com/index.html . Referenced: 13.5.2015.

[36] Sähköansio. Available: http://www.sahkoansio.fi/ . Referenced: 13.5.2015.

[37] The jQuery Project. Available: http://jquerymobile.com/ . Referenced: 13.5.2015.

[38] Fressi. Available: http://www.fressi.fi/ . Referenced: 13.5.2015.

[39] Betoco. Available: http://www.betoco.fi/ . Referenced: 13.5.2015.

[40] Sampo Suonsyrjä (2014, January) Personal interview.