Analysis of Lean Startup Metrics: Literature Review and Empirical Findings

Teppo Pennanen

Helsinki October 21, 2015

MSc Thesis

UNIVERSITY OF HELSINKI

Department of Computer Science
This thesis is a study on Lean Startup metrics. It attempts to answer what is measured and how in Lean Startup, how it differs from other software measurement and what information needs today's startups have. This study has literature study and an empirical survey using real start-ups.

This study explains how the software measurement has changed over the years and what kind of metrics Lean Startup suggests to be used and why. It shows differences in measurement use between traditional start-ups and Lean Startups. This study suggest reasons and motivations to use measurement in start-ups and gives examples when not to.

In the scope of this study a survey with questionnaires and interviews was conducted. It showed distinctly different attitudes towards measurement between traditional start-up entrepreneurs and those who like to call themselves Lean Startup entrepreneurs.

Measurement in Lean Startup is not an end in itself, but a useful tool for gaining feedback for the gut-feelings of an entrepreneur. Metrics, when meaningful and correct, communicate the focus within a start-up and will objectively evaluate the business' success.

ACM Computing Classification System (CCS):
D.2 [Software Engineering],
D.2.8 [Metrics],
D.2.9 [Management]
Contents

1 Introduction

2 Software Development History in the light of Lean Startup and Software Measurement

2.1 The Late 1940s ................................................................. 2
  2.1.1 Toyota Production System .......................................... 2
  2.1.2 Deming cycle ............................................................ 5

2.2 The 1950s and 1960s ....................................................... 5
  2.2.1 The start of software development industry .................... 5
  2.2.2 Measuring the programmers ....................................... 5
  2.2.3 Optimizing software for hardware ............................... 6
  2.2.4 The birth of software development process .................... 7
  2.2.5 Measurement of Quality ........................................... 8
  2.2.6 Structured programming ........................................... 9

2.3 The 1970s Oil Crisis and Lean Manufacturing ....................... 9
  2.3.1 Waterfall ............................................................... 9
  2.3.2 A complexity measure ............................................. 10
  2.3.3 The definition of quality .......................................... 10

2.4 "Doing things right" era of 1980’s ..................................... 11
  2.4.1 Constructive Cost Model ........................................... 11
  2.4.2 Spiral Model .......................................................... 11
  2.4.3 Scrum ................................................................. 12
  2.4.4 Open source .......................................................... 12
  2.4.5 Six Sigma ............................................................. 12
  2.4.6 Capability Maturity Model ....................................... 13

2.5 "Doing the Right Things" era of the 1990’s ............................. 13
  2.5.1 Dynamic Systems Development Method ........................... 13
4.1.3 Measuring product .............................................. 38
4.1.4 Measuring business ............................................. 39
4.1.5 User Involvement .............................................. 39
4.2 Lean Startup Metrics ............................................. 40
   4.2.1 Empathy stage .............................................. 40
   4.2.2 Stickiness stage .............................................. 41
   4.2.3 Virality stage ................................................. 42
   4.2.4 Revenue stage ............................................... 43
   4.2.5 Scale stage .................................................. 44
5 Survey Results .................................................... 44
   5.1 Questionnaire .................................................. 44
   5.2 Interviews ..................................................... 49
      5.2.1 Case one: A small marketing company expanding their business to other areas .............................................. 49
      5.2.2 Case two: Startup attempts to build a communication device ................................. 50
      5.2.3 Case three: An entrepreneur creates a start-up to build a software to solve own problems ............................................. 52
      5.2.4 Case four: A start-up based on value of trust .................................................. 53
      5.2.5 Case five: A training software for music lovers ................................................. 54
      5.2.6 Case six: Startup tries to expand abroad ............................................. 55
      5.2.7 Case seven: Two start-ups with a feasible business model, one abandons and one pivots ............................................. 56
   5.3 Startup metrics .................................................. 56
      5.3.1 Lean Startup vs traditional entrepreneurs ............................................. 57
      5.3.2 Revenue ..................................................... 57
   5.4 Usage of Lean Startup Metrics ................................ 57
      5.4.1 Verification versus intuition ............................................. 58
      5.4.2 Metrics as means of communication ............................................. 59
5.4.3 Ease of measurement ................................. 59
5.4.4 Funnel metrics ........................................ 60
5.4.5 User behaviour ........................................ 60
5.4.6 Exploratory measurement ............................. 60
5.4.7 Revenue .................................................. 61
5.4.8 Scaling needs ............................................. 61

6 Analysis ....................................................... 61
   6.1 Limitations ............................................... 62
   6.2 Conclusions ............................................... 63
   6.3 Summary of Contributions ............................... 63
   6.4 Future Research ......................................... 63

7 Summary ...................................................... 64

References .................................................. 65

Appendices

1 List of terms
1 Introduction

Lean Startup is an umbrella term for a set of principles and methodologies for finding a successful repeatable business model in an uncertain environment. It encourages entrepreneurs to validate different aspects of their business plan by creating hypotheses, putting them through tests, measuring the results, analysing the results and making conclusions based on the analysis. Lean Startup suggests different goals and attributes for measuring the business model depending on the stage of the start-up.

Lean Startup has gained a lot of hype, followers, authors and contributors after its introduction by Eric Ries in 2011. As it’s a relatively new field, formal scientific research has not yet been made on the subject. The Lean Startup authors base their claims partly on personal experience and life experience. In literature, the start-ups that contribute their success to Lean Startup methodologies, are usually hand picked and selected as are the examples to the contradictory. The reasons for the failures are contributed to not following Lean Startup principles. Stories about failed start-ups that tried to implement Lean Startup methodologies are less common in literature, but such cases emerge in different blogs in the internet.

This study focuses on the metrics and measurements in Lean Startup. What kind of metrics do Lean Startup entrepreneurs prefer and why? To answer these questions, we must first take a look at history of software measurement and the history of Lean Startup. The origins of both can be traced back to the middle of 20th century. Lean Startup history begins in Japan with Taiichi Ohno’s Toyota Production System (TPS) [Ohno78] and software measurement begins in United States with the birth of software industry.

Chapter 2 narrates this history as a result of a literature study. The focus of the measurement history is on the measurement goals, rather than specific metrics and measures. The Lean Startup principles and methodologies are described in more detail in chapter 2.7. The literature study answers the RQ1 and RQ2.

Chapter 3 describes the process of how the literature study, the survey and the interviews were conducted. The surveys evolved through several iterations and tried to attract responders from several sources. Interviews were held with several start-ups, with different situations, stages, problems and outcomes. The survey and interviews answer the RQ3 and RQ4. The survey results are discussed in chapter 5.

Chapters 4 and 5 list the research questions and answers them. There are four
research questions. RQ1: "What kind of software measures have been and how the goal of measuring has changed over the years?", RQ2: "What kind metrics are suggested to be used with Lean Startup and what do they measure?", RQ3: "How are the metrics in Lean Startup being used?" and RQ4: "How should the metrics in Lean Startup be used?".

Chapter 6 contains the analyses of the study. A lot of what’s included in Lean Startup, has been around for decades. Understanding the customers and the market is important to success of a start-up. From one perspective this continuance to focusing on minimizing waste caused by uncertain and rapidly changing circumstances, with the focus on measuring the user behaviour. When compared to traditional start-up, Lean Startup provides tools and methods to complement entrepreneur’s insight and wisdom and to some extent make up for poor intuition. Lean Startup provides more focus with clearer instructions and less work.

2 Software Development History in the light of Lean Startup and Software Measurement

This chapter reviews the history of software measuring, reviews the history of Lean Startup and describes the main principles of Lean Startup.

After the list of terms, the review starts from post-war Japanese auto industry where the foundation of lean manufacturing was started. The review then progresses to start of software development and software processes, and how the software measurements and software measurement goals evolved through the decades. In the nineties customer development process and agile development paved way to Lean Startup. Lean Startup is described in more detail in chapter 2.7

2.1 The Late 1940s

2.1.1 Toyota Production System

Shortly after the World War II, as the rebuilding had begun and money was scarce, the Japanese auto-mobile industry was struggling. It had a relatively small domestic market and it couldn’t compete with American mass produced cars. The cost of building cars in Japan was significantly more than making them in United States, where they were mass produced. Kiichiro Toyoda, president of Toyoda Mo-
tor Company, and Taiichi Ohno, Toyota’s leading manufacturing engineer, visited Ford’s mass production plants in United States and were convinced that setting up such factories to improve the productivity was not a viable option for Toyota Motors, due to smaller domestic market and Japan’s more limited land area. Some elements, however, could be scaled down to their needs. [Ohno78, Duqu07]

At the same time, Ohno learned that the average American worker was nine times more productive than his Japanese counterpart, and not just in automobile industry. He tried to understand why productivity of a Japanese worker was so low. This suggested to Ohno, that Japanese workers were wasting something. Ohno started to improve the productivity by identifying waste and removing it. He began to measure and optimize the time-line from the order to delivering the product. If something in the process did not add to the value of the product to the customer, it was considered waste and eliminated. [Ohno78]

"All we are doing is looking at the time-line from the moment the customer gives us an order to the point when we collect the cash. And we are reducing that time line by removing the non-value-added wastes." - Taiichi Ohno [Ohno78]

Ohno defined seven types of waste (muda) [Ohno78]:

- **Overproduction.** All cars that are manufactured, but not sold, are considered waste. Therefore no cars would be made unless they were ordered first. This is handled by the pull production Kanban cards.

- **Waiting.** When the product is not being worked with or is in transport, it’s waiting. This waiting time is considered waste.

- **Transportation.** Transportation takes time and may cause damage to the product being transported. The product lines are arranged so that the transportation of the product from one location to another is minimized.

- **Processing.** The how and why things are done, is written down and taught to workers. All work is done exactly the same way so the processes are continuously tested and measured. Workers are expected to suggest improvements. The most promising of these suggestions are then tested and the best practices are taken into use. When a problem occurs, the product line is stopped. The problem is fixed. It’s made sure, it doesn’t happen again and cause more waste in delays and extra work.
- **Inventory.** There is no excess inventory, but everything is produced Just-In-Time (JIT). Only the amount that is needed, is produced, and only when they are needed.

- **Movement.** The movement around the product is minimized. If as a result of changes the worker is idle, it is still considered waste. The workers should be multi-talented and be able to do other work if they otherwise would be idle.

- **Defects:** The product is being tested in every step. Defective products are not passed from one step to the next step. The next step in the production line inspects the previous deliverable. The testing is made so easy that it won’t add much extra time.

Doctor Shigeo Shingo worked, as a consultant to Toyota, to improve the Toyota Production System (TPS). He developed and implemented Single-Minute Exchange of Die (SMED) to minimize the time wasted on changing the product line to work on a different product. He was convinced that to implement SMED successfully, the workers had to "know-how" to do their work and "know-why" things are done. A worker has to understand why things are done in order to cope with changing situations. [Shin89]

TPS didn’t raise any flags until 1970 oil crisis. Prior to that that nothing was written down in fear of losing advantage. Over the decades TPS had introduced many improvements, such as Kanban, JIT and SMED. As important as these improvements were for Toyota, merely copying the processes or using the tools did not give Toyota’s competitors the expected boost. In their work Steven Spear and H. Kent Bowen concluded that it’s not the tools that are important, but a set of four rules [Spea99]:

- **Instructions** how people work are detailed. The work is taught and done in the same way so the processes are tested and measured continuously. The process improvements are suggested by the workers. These improvements are tested and implemented, if they improve the whole.

- **The connections** between people are to be standardised and direct. These is no ambiguity about who provides, what, when and for whom.

- **The production line** is constructed in such way, that the product being developed should have only one path it follows. Product does not flow to the
next available person, but to a specific person. If the next person is not available, it is considered a problem with the product-line, which may have to be redesigned.

- The workers are taught to improve the process using scientific method. The workers are taught and expected to be able to formulate a solution into a hypothesis, test it and use the results as a foundation for their suggestions. This is the only way to bring changes to activities, connections and pathways.

2.1.2 Deming cycle

In 1951, at a Japanese Union of Scientists and Engineers, Edwards Deming refined the Shewhart’s Cycle (specification - production - inspection), which at the time presented a dynamic scientific process of acquiring knowledge. He added a market research and iteration through re-design (design - make it - put it on the market, test it through market research, re-design the product, in the light of consumer reactions) into the process to get feedback. This was later renamed as the Plan-Do-Check-Act (PDCA) cycle. [Moen06]

2.2 The 1950s and 1960s

2.2.1 The start of software development industry

In 1951 there were only ten computer installations in the United States [Wein68]. The computers were programmed by engineers. Although there were no standards how to create software, the programming was already somewhat structured, due to the compartmentalized mindset of the engineers creating the software. The programming work was slow and costly. In symposium of advanced programming methods held in 1956 one of the main concerns was developing higher level languages, with the aim of reducing the cost of one instruction line from $50 to $1[Beni87].

2.2.2 Measuring the programmers

By the end of the sixties the number of computer installations increased approximately to 50,000 to 70,000 installations because of new commercially produced computers. By that time there was estimated a shortage of 50,000 to 90,000 programmers [Wein68].
In this new industry area, there were no common standards according to which programs were developed. Programming was considered craftsmanship and no two programmers were likely to create a similar solution to a given problem. Nor were there any common tools to develop with, apart from what the individual programmers made for themselves [Kay69]. The success of software project was largely the result of the capabilities of the programmer. Therefore it was thought to be necessary to measure and analyse the aptitude of a person for the task of programming.

First such test was the Programmer Aptitude Test (PAT) by IBM in 1959. It tested different attributes such as reasoning, logical ability, attention to detail and accuracy with series completion, figure classification and arithmetic reasoning tasks [Mazl80]. Some of the other tests were APTitude Test for Programmer Personnel (ATPP), Strong Vocational Interest Blank (SVIB). Wonderlic Personnel Test and the test of Primary Mental Abilities (PMA) [Maye68]. In 1962 the Special Interest Group for Computing Personnel Research (SIG/CPR) was formed and one of it’s major issues was to study to what degree do these kind of tests assist management in recruiting trainees and experienced programmers.

According to a survey, made to computer managers in 1967, the most widespread problems of the industry were personnel recruiting, evaluation, and job assignment [Gott67]. A study made in 1968 suggested that, while the most widely used tests were able to predict their how well they performed in training, none were able to predict job performance. The study also questioned the concept of general aptitude tests and suggested tailored solutions for each company as the job profiles varied [Maye68]. Another study concluded that people at college level with differing disciplines do not require competing courses to learn programming [Mazl80].

### 2.2.3 Optimizing software for hardware

The computers in the sixties had harsh limitations in both speed and capacity. To improve a program, its performance, size and resource usage was being measured. Optimizing the program usually caused the code to be filled with hacks, that would make the program logic more complex. This made testing more difficult and applications less robust. [Boeh70, Beni87]
2.2.4 The birth of software development process

In 1955 the development of SAGE airspace monitoring system was started in MIT Lincoln Laboratory. In that project the software development scale increased significantly. Hundreds of programmers working together for over a decade required more disciplined approach to software development than before. One of the used methods in that project was a software development process that consist of the following parts: dividing software into smaller components, defining interfaces, requirements, machine and operational specifications and integrated testing of hardware and software in operational environment [Hell66]. The project had a schedule and measured the milestone slippage, i.e. by how much the milestones were missed [Beni87].

In the proceedings of 1968 of The Association for Computing Machinery (ACM) National Conference George F. Weinwurm argued that to reach the full potential, the information processing needs formalized procedures and standards, so that reasonably trained people can do what now only the exceptional programmers could do. He called out for ideas, where such management methodology could come from. In the same conference Dick H Brandon suggested that the industry needed to be specified and redefined as the functions and definitions were insufficient and conflicting with current practices [Wein68].

"We do not really know how to select programmers, and we tend to select those with some undesirable characteristics... Typically, they work for a manager who is ineffective because he has been given neither proper management training nor basic tools and disciplines with which to work; whose functions have not been defined, and whose process of communication with the system analyst or user is generally confused. Finally, all this takes place within a technology which changes so rapidly that it is almost impossible to get a fix on the functions and the method by which the work is supposed to take place, before it changes" - Dick H. Brandon [Kay69]

In response at Spring Joint Computer Conference in 1969 Ronald H Kay summarized that, given certain prerequisites a computer program project can be managed like any traditional project. These prerequisites were: assessment of what is possible, organization, project plan (Analysis, Specification, Design, Implementation, Integration, Testing, Publication and Maintenance), specification, evaluation of the
end product, evaluation of the programmers, software developing tools and higher level languages.

To evaluate programmer performance metrics such as: time required to complete an assignment, tightness of code, quality of documentation, running time, storage requirements, and computer time required for debugging, were used. The adequacy of tools and programming languages were measured by their usage. If a tool or a language was preferred by many programmers, it was assumed to be adequate. The data was used to select the best employees and tools. The project plan was measured for accuracy; planned versus realized. The progress was measured giving visibility to the project. The end product was measured for correctness. The tools and methods to do that were found inadequate and the testing was possible only after the implementation was done. Mean time between hardware failure was a typical metric of the era. [Kay69]

2.2.5 Measurement of Quality

"User should be able to specify precisely how good a product he wishes to buy, such things as how easy the program should be to run production with and how easily it can be modified. Rarely can the user even discuss these factors, much less specify the extent of their importance to him." - R J Rubey and R D Hartwick. [Rube68]

In the end of the sixties the software testing was often black-box testing using quality assurance systems devised for hardware. A program was good, if it performed its functionalities well. In 1968 Raymond J. Rubey and R. Dean Hartwick published a paper, in which they brought up the topic of measuring quality of programs. The paper was based on a study of quality in space-borne software, created for the Air Force Space and Missile Systems Organization. The paper suggests several attributes for software, which would then be quantitatively measurable. From the measures, a metric describing software's quality is then derived. The quality attributes were: correctness of mathematical calculations, correctness of logic in program, level of interference between program entities, optimization of computation time, optimization of memory usage, intelligibility of the program, modifiability, learnability and usability. The metrics were based on quantitative measures, some of which were subjective estimations [Rube68].
2.2.6 Structured programming

In 1969 E.W. Dijkstra formally argued that as computers are getting more complex and increase in capacity, comprehensive black box testing is very laborious. Therefore structuring programs to smaller components is mandatory to maintain some level of control over program correctness [Dijk69, Dijk70]. The goal of structured programming is get confidence in the programs correctness, without resorting to testing with all possible inputs. The software is measured for size and program structure is abstracted. These structural abstractions can then be compared and the abstractions can be used in planning the test cases and measuring test coverage.

2.3 The 1970s Oil Crisis and Lean Manufacturing

The 1970s started with two big oil crises. The increase in oil price caused problems for auto industry, because the cars had so far been mainly large in size and petrol consumption. With the high oil price and oncoming recession, the demand for high consuming cars decreased. Toyota motor company, however, did not suffer as much as the rest of the car manufacturing industry. Because Toyota had abandoned the idea of mass production, they didn’t have large supplies of cars no-one cared to buy (Kanban, JIT), and because it could shift focus more rapidly (SMED) according to the changes in demand in the market. This is when the western world noticed Toyota and its different approach to making cars. The car-manufacturers of the west became interested in TPS.

Programs at the start of the 1970s were structured, some steps in defining and measuring software quality had been made and the software development process was becoming a necessity.

2.3.1 Waterfall

In 1970 Winston W Royce summarized his experiences in large software projects and argued that all software development has two essential steps: analysis and implementation. He then continued to add more stages and created a model for software development. According to the model, the software was to be designed first, the design was to be documented, a prototype was to be made, testing was to be planned, controlled and monitored, and the customer has to commit to the process and be involved all through the process. He separated the programmers
from the testers, called for code reviews and path coverage in testing [Royc70]. The model emphasized designing and documenting in advance and nominates testing as the biggest risk in software development. It measures the amount and quality of documentation with the presumption that, if the documentation is poor, the outcome will be poor. In testing and in addition to peer reviews, path coverage is measured and documented. Errors found in programs are used as an indicator of how ready the software is. Customer involvement is encouraged and customer is invited to review the program design, any and all critical software component designs and the final product. This model was subsequently named as the Waterfall model [Bell76].

2.3.2 A complexity measure

In 1976 Thomas J. McCabe, Sr. introduced Cyclomatic complexity measure to help analyse code controls structures. These calculations, can be automatised and the result is a number telling the complexity of the source code. This value can then be used to assess the complexity of the source code, to find modules that are difficult to test and maintain. The complexity measure can point areas where there room for simplifying and it can be used to verify test coverage [Mcca76]. Although Martin Shepperd criticises the science behind the complexity measure [Shep88] the goal of measuring has been to reduce the cost of programming and maintenance.

2.3.3 The definition of quality

In 1977 James A. McCall introduced the General Electric model as GE model for describing software quality. It contained eleven software quality factors. These were updated by Evans and Marciniak in 1987 and Deutsch and Willis in 1988 to a list of twelve software quality factors: correctness, reliability, efficiency, integrity, usability, maintainability, flexibility, testability, portability, re-usability, interoperability, verifiability, expandability, safety, manageability and survivability. Each of these have been assigned their own sub-factors and measures and metrics can assigned to each of these sub-factors. The measures are numerous and each was chosen to be ratio of actual occurrences to possible number occurrences [Mcca77, Mcca78, Gali04].
2.4 "Doing things right" era of 1980's

2.4.1 Constructive Cost Model

1981 Barry Boehm, the director of Software Research and Technology at TRW, introduced his software engineering principles for dealing with the problem of making decisions with limited resources. This was known as Constructive Cost Model (COCOMO) prediction technique. He recognizes the unpredictability of software engineering projects and argues how main economic analysis techniques fall short. He emphasized information buying, with developing rapid prototypes of the high risk components, as the most effective way to reduce the overall risk and to get a better view of the probability of building successful software. Boehm focuses on estimating the cost of building the software and suggests using combination of several existing estimation techniques (algorithmic models, expert judgement, analogy, top down and bottom up), comparing the results and investigating the differences. He argues that the more defined the problem is, the more accurate the estimations will be [Boeh81]. The estimations in COCOMO were first based on Delivered Source Instructions (DSI), then later to Source Lines of Code (SLOC). The algorithms use these numbers with qualitative information, and historical quantitative data that is derived from numerous projects. Other such prediction techniques are Mermaid (1990) and Case based reasoning (1998).

2.4.2 Spiral Model

Following his work with COCOMO (chapter 2.4.1) Boehm introduced in 1986 the spiral model as his improvement on the waterfall model [Boeh88]. In spiral model the software is developed in small iterations, each iteration consisting of evaluating, developing, planning and committing to the next phase. The spiral starts with creating a hypothesis, where the operational mission is assumed to be improved by means of software development. If, at any point of the spiral, the hypothesis fails, the spiral is broken. Each iteration accumulates more information in regards to the previous iteration and prototyping is included in each iteration. First iteration is feasibility study, next couple of iterations are documentation of different levels, next iterations are software development and the iteration ends with installation of the new or modified software.

In spiral model the risks in developing the software are evaluated. A cost effective strategy is developed for resolving the risks. In each iteration there are questions
that need answers. One of the main questions is whether to continue to planning the next iteration or abandon.

This may involve prototyping, simulation, benchmarking, reference checking, administering user questionnaires, analytic modeling, or combinations of these and other risk-resolution techniques.

2.4.3 Scrum

In 1986 Hirotaka Takeuchi and Ikujiro Nonaka took a slightly different perspective to software development. They made an analogy of software market to rugby field, where the situation changes rapidly. They introduced self-organizing highly autonomous project teams and named them scrums. In these scrum teams, the capability to handle the continuously changing environment is built into the teams and the way of work. Team members were hand-picked with the focus of building a functional team. Team is subtly managed by changing the team members. In order to respond quickly to changing situations, the team members work closely together, communicate with each other and have versatile skills. The team members are encouraged to get in touch with the customers and listen to what they have to say about the product or feature. Their performance is monitored in checkpoints and the product they are creating is evaluated [Take86].

2.4.4 Open source

1986 Margaret H Hamilton reasoned that totally error free software may be possible in theory, but in practice will be difficult to achieve. Even if most of the faults can be eradicated, the users error creating actions, whether intentional or unintentional, cannot be taken into consideration when developing the software. She suggest two methods for reducing errors: building software from reliable components and using static analysers to analyse the source code [Hami86].

2.4.5 Six Sigma

In 1986 at Motorola Bill Smith developed a set of tools for improving product or service process quality [Hahn00, Pepp10]. This was developed further at IBM. The focus was on customer satisfaction, which according to Six Sigma, was the result of translating customer needs and requirements to measurable characteristics,
designing and developing the system based on those, verifying the results and ensuring their continuous monitoring. This process, mainly developed for reducing defect rates, usually has five steps: define, measure, analyse, improve and control (DMAIC). First the different aspects of the problem are defined. Then set of measures to test critical-to-quality characteristics are developed and tested by measuring the baseline. One of the root causes for problem is selected for removal. The change is implemented and tested. The key indicators are monitored and it's controlled that they stay within acceptable limits. Implementing Six Sigma is top down process, which ensures that the management is committed. The continuance of this, according to commitment is the result of access to statistical data, availability of analysing tools and effective communication [Hahn00]. The goal of Six Sigma is to optimize processes or product to decrease the amount of defects per million opportunities (DPMO) [Lind03].

2.4.6 Capability Maturity Model

In 1973 Richard L Nolan created a model by applying stage theory into introducing computer resource to an organization. The stage theory presumes that "elements in systems move through a pattern of distinct stages over a period of time and that these stages can be described". One of the two guidelines for stage theories is that characteristics of each stage should be distinct and empirically testable [Nola73]. Based on that, in 1988 Watts Humphrey published Capability Maturity Model (CMM) to predict project's capability to meet its goals. Requirements for each level are set in stones and it is monitored that they are fulfilled [Hump88].

CMM defines five levels of maturity: initial, repeatable, defined, managed and optimizing. Each of these levels is given a set of key performance indicators (KPI). These KPIs use metrics like error density, productivity, percentage of rework, cycle time, schedule fidelity, error detection effectiveness and return on investment (ROI) [Fent97].

2.5 "Doing the Right Things" era of the 1990's

2.5.1 Dynamic Systems Development Method

In 1994 a number of end-user organizations in United Kingdom, software-tool vendors and academics interested in rapid application development formed a group
called Dynamic Systems Development Method (DSDM) consortium. Their goal was "to develop and evolve continuously a public-domain method for rapid application development". They recognized the users involvement in software development as something that resulted in better product, mainly because if the user is involved, the software can be modified to better match the user requirements (whether implicit or explicit) [Mcca00].

The consortium published its Rapid Application Development (RAD) standard, in which the business requirements take precedence over the quality of system's operational characteristics, and what the product does is preferred over how the product was made. After assuring that a project fills the RAD project requirements, a business study is made and a prototype plan is outlined. Prototype is developed in co-operation with the users. Product development is timeboxed and iterative. Testing is integrated into all stages. Small teams with both developers and users are used and formal documentation is not required [Mill95].

DSDM recommends collecting the following metrics for each timebox and prototype: business functions delivered, effort expended, elapsed time and size of the prototype.

2.5.2 Customer Development

"Startups don’t fail because they don’t have the product; they fail because they lack customers and a profitable business model" - S. Blank

In the mid-nineties Steve Blank developed his customer development methodology, which encourages entrepreneurs to try to validate their hypotheses about the product and the market rather than base their ideas on faith that "if they build the product, the customers will come". Blank speaks against premature scaling and emphasises low money burn rate before understanding the market and customers [Blan13, Blan12].

The process has four steps: Customer Discovery, Customer Validation, Customer Creation and Company Building. Before the process is started, a commitment has to be received from all participants in the start-up. After each step a "pivot or continue" decision has to be made.

The first step, Customer Discovery, validates that a product solves the problem for a group of people. In this step, the problem and product hypotheses are created and tested, by famously "getting out of the building" and asking directly from people. The purpose is to listen to the people and try to determine if the problem
is real and who would be the people for whom the problem is real. If the problem hypothesis is validated, then the solution hypothesis is created and put to test. For this the entrepreneur goes to the potential customers and tries to listen and verify, if the solution fixes the problem for the potential customers. The market type, competition, distribution and the pricing hypotheses are also created in this step. If all the hypotheses have been validated according to the best current knowledge, the start-up proceeds to the next step.

The purpose of Customer Validation step is to verify that the market for the product is big enough to build a viable business on. The step starts by acquiring few so-called earlyvangelists, who are enthusiastic enough to purchase a solution that is not yet fully developed. The product is further developed according to their feedback. The distribution channels are tested when the product is delivered to the earlyvangelists and instructions, of to whom and how to sell, are developed based on the feedback of selling to earlyvangelists. Product positioning is decided. The hypotheses and instructions are validated by analysing the feedback from the customers and talking to the experts of the field. If all validated, the start-up continues to the next step.

In Customer Creation step the business attempts to scale and cross the customer gap between the earlyvangelists and the main market. This is done by verifying market type with questionnaires, positioning the company and the product in that market with the help of internal and external audits, planning for advertisement, launching the product, and creating demand for the product. Measures are defined to measure the success of launch and demand creation activities. The decision to move to the fourth and final step, is based on these numbers.

The Company Building step focuses on the change from start-up to a mission centric organization and culture. The metrics that are measured, are the sales numbers to verify that the cross to main market has been successful.

2.5.3 Personal Software Process

In 1995 Watts Humphrey introduced the personal software process as means for a developer to improve her software development process. As in CMM there are five levels: initial, repeatable, defined, managed and optimizing. The developer gathers different data from his development work. Pieces of this data is analysed, for getting the level of different attributes and finding the places where to improve. The measurement continues though the process to follow the progress, adding new mea-
sures to gather with every new level achieved. The process measures development
time, source code size and code quality related metrics. They are used to assess
baseline, show progress in skill development and teach the developer to make better
estimations [Hump00, Hump05].

2.6 "Building the right things" era in 2000’s

2.6.1 Agile Manifesto

In August 2001 the Agile Software Development Alliance published "The Manifesto
for Agile Software Development" as a result of a meeting in which the alliance was
formed. This alliance defined characteristics of an agile process as [Beck01]:

- Individuals and interactions over Processes and tools
- Working software over Comprehensive documentation
- Customer collaboration over Contract negotiation
- Responding to change over Following a plan

The Agile Manifesto emphasizes customer, rapid delivery in cycles of one to four
weeks, working and useful software, embracing changing requirements, close and
daily cooperation between business people and developers, motivated and trust-
worthy individuals, face-to-face conversations, sustainable development at constant
pace, attention to technical excellence, good design, simplicity, self-organizing teams,
and regular adaptation to changing circumstances. The progress is measured mainly
by working software. To implement these, agile provides several light software de-
development methods to use, such as scrum, crystal clear, extreme programming,
feature-driven development and adaptive software development.

In the scrum method a representative of the customer (Product Owner) evaluates the
new features by their business value, by giving them value points. The development
team divides features into tasks and evaluates the effort for each task, by giving them
story points. In planning poker, or scrum poker, team members estimate the effort
for each task and reveal their estimations at the same time. The estimations are
debated and consensus is reached about the estimations. While the different tasks
effort levels have different comparable values, Agile methodology advises against
using any real-life units in the estimations.
When the tasks have effort estimations, the product owner prioritises the features of the product. Only as many of the features are taken into one timeboxed iteration (sprint) as the team can handle. If the cost for the features that are left out, are too cumbersome to implement compared to their value, the product owner can say that the project is finished.

The development team implements the features in a sprint and makes a new delivery, with new functional features, for the customer to test and take into use. The progress is estimated by amount of work left, by calculating the effort estimations of all the unfinished tasks (Burndown). The team can measured how much work has been done in sprint, by counting the effort estimations of the finished tasks. This information is used in estimating how much work can be done in the next sprint (Velocity).

The scrum teams work in the same space with the customer for the communication to be effective. They have daily progress meetings where possible problems are flagged. All that has been learned during a sprint is used in retrospective-meeting to improve the way of work.

2.6.2 Design for Six Sigma

Design For Six Sigma is closely related to Six Sigma (chapter 2.4.5, which is often called DMAIC or Six Sigma Improvement to mark the difference). Where Six Sigma attempts to improve existing processes and products, DFSS focuses on "designing it right the first time". DFSS has four phases: "Identify requirements", "Characterize the design", "Optimize the design" and "Verify the design" (ICOV). In fist phase (Identify requirements), the customer needs and want are gathered into list, from which they are converted into functional and measurable requirements. The requirements are prioritized and metrics and acceptable performance levels are set. In the second phase (Characterize the design) the customer requirements are converted into functional requirements and design entities that fulfil those requirements are characterized. Those design entities are evaluated and one concept will be chosen. In the third phase (Optimize the design) the optimized design with all functionalities is released. In the fourth phase (Verify the design), the design is release is in a piloting environment and verified that it fulfils the requirements. Finally after that the full customer roll-out and handover is executed. [Yang03].
2.6.3 Lean Software Development

In 2003 Mary and Tom Poppendieck took the TPS’s lean principles (chapter 2.1.1) and compared them to agile methods. They argued that applying lean principles to software developing principles instead of practices, would lead to process and quality improvements. This resulted in seven principles for software development: eliminate waste, build quality in, create knowledge, defer commitment, deliver fast, respect people and optimize the whole. They named it Lean Software Development [Popp07, Popp12].

Poppendiecks claim that the value of the software is in the eye of the customer. Understanding the customer is a necessity and it can rarely be measured, without interaction from the customer through deployment. It can be assessed easily by asking if the customer would recommend the service to others. Anything that doesn’t add to that value, or how to deliver that value more effectively, is waste. Lean development prefers fewer system level metrics, that guide the behaviour of the lower levels. It names cycle time, financial results and customer satisfaction as the best metrics [Popp07].

Lean software development takes the wastes from TPS (chapter 2.1.1) and renames them for software development as: partially done work, extra features, relearning, handoffs, task switching, delays and defects.

2.7 Lean Startup

A start-up is a company, a partnership or temporary organization, designed to search for a repeatable and scalable business model, in an uncertain environment, before running out of resources. Lean Startup utilizes lessons from TPS (chapter 2.1.1), Customer Development (chapter 2.5.2) and agile methodologies (chapter 2.6.1). It focuses on doing things iteratively, in small batches, constantly validating, continuously learning and eliminating waste. It does this to try to match the production according to customer demand [Ries11].

In Lean Startup, entrepreneurs try to learn from customers, not only by asking qualitative question, but also with experimenting with users. By running experiments on their users, the entrepreneurs are able to ask questions users can’t, or won’t, answer truthfully, if asked directly.
2.7.1 Build-Measure-Learn -cycle

A famous bon mot asserts that opinions are like arse-holes, in that everyone has one. There is great wisdom in this... but I would add that opinions differ significantly from arse-holes, in that yours should be constantly and thoroughly examined." - Tim Minchin

When working in an uncertain environment or ecosystem, acting on unfounded opinions and assumptions can prove to be devastating. To minimize the risk resulting from faith or guesswork, Lean Startup thrives on validated learning. The validated learning is a continuous process and it's achieved by implementing one of the most fundamental principles of Lean Startup; the "build-measure-learn" -cycle. In the beginning, an important unsolved problem is studied. For finding the root cause for the problem, Lean Startup suggests utilizing the "Five Whys"-method [Ries11]. When the root problem is selected and defined, a hypothesis about how the problem is solved is created. The hypothesis should be such, that it can be disproved with relative ease. A test is planned and measures, that monitor the effects of the change, are selected. Then the test is executed and the measured. The measurement data is analysed for correlation and causality. New decisions are made according to this validated learning. If the hypothesis was accepted, a new problem is selected. If the hypothesis was rejected, a new hypothesis is created. This iteration is continuous throughout the lifespan of the start-up.

2.7.2 Metrics and measures

A good metric and measure should be so simple that anyone can understand it and its meaning. This requires that understandable units are used, and that the values mean something. If everyone has consensus about what the metrics mean, it is easier to discuss what actions should be taken in response to the results. A Comparable measures are good, as they put the results in context. Ratios or rates are good measures, as they are understandable, comparable and they are easier to act on than measures of other scales. [Ries11, Crol13].

The data from the measurement should be analysed and made available. The data should be shown in such format, that makes sense in the context. Cohorts should be used, as they usually give great insight to the data. For example showing the rate of new users and abandoning users in cohorts, can show if the leaving users are
the long time users or new users, who quickly abandon the product. These analyses call for different actions.

As with all scientific research, the data should be available for those, who wish to verify the results against the data, in case of misunderstanding or misinformation. The measuring serves no purpose if the results are wholly or partially misinterpreted, manipulated or hidden. In such case, the measurement loses its purpose and strength, and becomes a political tool.

The results from the test should be verifiable by other means than the executed test. This is needed to eliminate possible errors or misconducts in the measuring process. If testing by other means brings in similar results, the odds, that the result are valid, increase.

Vanity metrics are the opposite of actionable metrics. They are the kind of metrics, that cannot be acted on, but tend to make people feel good or take unwarranted pride in the result. Vanity metrics don’t tell what caused the change or what the change means. They usually go up and right, as they can only increase over time. Metrics, like number of clicks or page visits, fall under this description [Cro13, Ries11].

In addition to these reporting metrics, which tell us "things we know we don’t know", Lean Startup raises the awareness of exploratory metrics, or to put it another way, finding out things "we don’t know we don’t know". These metrics are important, especially in the beginning of the start-up. Partly because there are many unknowns, but mainly because these are where the unfair advantages usually lie. By measuring many things and trying to analyse data, the entrepreneur may notice patterns that may result in significant discoveries. As, by definition, no-one usually asks the questions, that would normally get these answers, it is unlikely that others have found this out. Now "an unknown unknown" has become "a known unknown" and can be verified with reporting metrics, if needed.

2.7.3 Marketing funnel

The Lean Startup marketing funnel consists of typical steps, that users take on their way to becoming customer. The steps are: acquisition, activation, retention, referral and revenue (AARRR). The funnel metrics, which measure activity in the different parts of the funnel, are named after the steps and are more commonly known as "Startup Metrics For Pirates" or "Pirate Metrics". The exact measure, which make up the metrics, depend on the solution [Cro13, 1].
Acquisition metrics measures how well users discover the product. Acquisition rate tells how well the start-up’s channels to customer work, and how well the product attracts users. To measure the acquisition rate, information about the product is spread through different channels using either viral or paid engines of growth. A percentage of the people who have become aware of the product check the product out. The acquisition rate can be measured for example by counting visits to the product landing page or product web page. This information more is useful when shown as rates in cohorts by different channel, so the different channels can be compared and effects in individual channels can be isolated.

Tracking paid advertising services, like Google Adwords and Bing Ads, provide measurement data for comparing the channels. This data can be analysed using different services, such as Google Analytics, Kiss Metrics or Mixpanel.

When people visit landing page, their behaviour can be measured and analysed. The most interesting thing to measure is the point where they either become active users or leave the page. Finding out the reasons for people not starting to use the service is both hard to get and valuable to know. In some cases may be possible to notice, when the visitor is moving away from the landing or product page and then ask why they did not sign up, but even then, getting an answer is not guaranteed.

The activation metrics measure how great a first experience the users have, using the product. Activation rate measures the ratio of the people who register or start to use the product, compared to the people who have checked the product out. It can be measured by, for example, rate of new registrations or rate of trial subscriptions.

Studying user data in cohorts makes calculating the average user lifespan possible. When users register (submitting email address for example), it’s easier to get feedback from the users, as the start-up has now means to contact the user. They users can be sent gentle reminders and asked for feedback or, in case they have stopped using the product, why they are abandoning the product.

Retention metrics measure whether and how often the users come back to the service. Retention rate is the ratio of users actively using the service, compared to all users. It is calculated by measuring the amount of users at the start of a time period, end of a time period, and the amount of new users gained during the period. Showing the measures in cohorts is useful to gain knowledge, who the leaving people are. Whether they are long time users or the recently signed up users, can suggest a very different ways to react to the numbers. The returning users behaviour can be monitored to find out what is compelling about the product, and to optimize the
product.

Referral metrics measure how well the users tell others about the product. Referral rate is the ratio of the users who refer the solution to others, compared to the users who actively use the product. The viral coefficient tells how many new users each user brings in on average. If the number is less than one, it means that referrals of each new user bring in less than one new user, so the number of new users and referrals will become zero over time. If the number is more than one, it means that each new user brings in more than one new user so the growth is exponential. If the virality is inherent, the entrepreneur can calculate the number of actions, that add to referral (such as "shares" or file transmits). The efficiency of those actions can then be analysed when checked against rate of new users. If the virality is not inherent, then the new users acquired by other means, such as paid advertising or SEO, should be excluded before calculating the rate of new users who are contributed to referral.

Revenue metrics tell how well the start-up is making money. Revenue rate measures the ratio of users who, end up paying for the product. By knowing how much average customer brings revenue in a period, what is the customer acquisition cost (CAC), and how long an average customer stays, customer lifetime value (CLV) and the Time to Payback (TTP) can be calculated. Multiplying CLV with amount of customers, the amount of revenue can be estimated.

All of the pirate metrics should be followed all the time, but the actions should be taken based on only some of them. Essentially the revenue will be the main concern, but there may not be need to spend effort on optimizing how to monetize the product, if the acquisition and activation metrics are low. If the activation metrics are down, the product is not getting attention. If the activation metrics are down while the acquisition metrics are high, finding out why the users don’t start using the product may be the most important. If the retention metrics are down, the reasons why users are abandoning the service should be discovered. Increasing the referral metrics may result in increased acquisition, activation, and retention metrics. If incoming revenue is low regardless of other metrics, other means to monetize should investigated and experimented with.

2.7.4 One-Metric-That-Matters

Lean Startup encourages the entrepreneurs to gather lots of data, focus on few, but have only one metric which tells how the business is doing. That metric is called
the One-Metric-That-Matters (OMTM). The OMTM answers the most important question that the entrepreneur has at a moment. When the most important question from the riskiest areas has been found, the metric should be easy find out [Crol13]. OMTM needs to have a clear goal, some border values that the value of the metric is compared to. Which metric is the OMTM, changes as the start-up situation changes. The OMTM focuses the company and encourages doing experimentations, that will improve the OMTM [Crol13].

2.7.5 Hypotheses

A hypothesis, in Lean Startup, should be a clear statement that can be clearly falsified. A falsifiable hypothesis states that a certain repeatable action will result in a measurable expected outcome. The hypothesis should contain the metrics and measures, that are followed, and instructions how the test is executed. The hypothesis should have minimum success criteria by which to define whether the hypothesis is verified. [Ries11, Crol13, Maur12]

Hypotheses can be ranked with importance and ease of testing. The importance of a hypothesis comes from how risky the subject is. Lean Startup suggest testing the riskiest hypotheses first, using ingenuity to find cheapest way to test it.

Testing a hypothesis should be timeboxed. Minimal amount of changes should be tested at the same time, to make it easier to tell what the individual impact of each change is. Testing only one change at a time may occasionally be impossible because, as the tests should run for some time to take effect, there may not be time to test hypotheses one change at the time.

2.7.6 Minimum Viable Product

To make the problem and the solution more concrete to customers and to avoid basing decisions solely on qualitative data, a Minimum Viable Product (MVP) is created and delivered to users. MVP has the minimum number of features needed to test the solution hypotheses and start learning. A typical way to implement a MVP is to take an existing similar solution and tweak it. Another way is to replace some of the functionality that would be difficult or laborious to implement with manual labour [Ries11].

Learning through MVP requires feedback. The feedback may be monitoring key
activities in the product usage, or if the MVP is used in an interview situation, the feedback may be received from answers to questions asked face to face.

Steve Blank and Ash Maurya state that the MVP should have price attached to it. This is to ensure that the problem is such that people are willing to pay for it. If the MVP is free, the entrepreneur must consider the fact that no-one will pay for the product and has to prepare for other means of monetizing, such as built-in purchases or selling ad-space [Blan13, Maur12].

2.7.7 Split testing

In Split testing, or A/B testing, the new and the old functionality (or two alternate new implementations) is tested in parallel. A planned improvement is introduced into the system, while the old is still in place. A small part of the users use the new implementation and the rest use the old implementation. The same measurements are taken for both and the results are compared. If the new functionality performs better than the old one, the new functionality is taken into use system-wide. Otherwise system is reverted to the old implementation and a new improvement is selected for implementation [Ries11, Crol13].

It's needless to measure what percentage of the people, who have become aware of the product, but don't visit the landing page. More useful is to split test the ads, to see which ads get more traction. What is hard, but important, to find out is why people, who have become aware of the product, don't visit. One way to get some insight, is to send invites to people via email and if they don't visit the product page, ask them why.

2.7.8 Lean Canvas

Lean canvas is an one page business model [Crol13, 2, 3], based on Business Model Canvas [Blan13, Blan12, Oste10]. Lean Canvas has nine fields: "problem", "customer segment", "unique value proposition (UVP)", "solution", "channels", "revenue streams", "cost structure", "key activities" and "unfair advantage". The canvas is divided into two parts: product and market. The "customer segment"-field and the "problem"-field describe who the customers or user are and what their top three problems are. If there are several customer segments, a new lean canvas should be created for all of them as the values of the fields may vary. The "unique value proposition"-field is a short description or slogan that explains what makes
the product worth buying. The "solution"-field lists top-three features that show that MVP is demonstrating UVP. The "Channels"-field lists the free and paid channels, that are used to reach the users. The "Revenue streams"-field identifies the revenue model, estimations and limits. The "Cost structure"-field lists the fixed and variable costs. The "Key activities"-field lists which user activities map to retention or revenue. The "Unfair advantage"-field describes why the start-up idea cannot be copied or stolen by another start-up or company, or why the start-up cannot be outperformed in an open market [4].

Lean Canvas is a living document which is updated through start-up evolution according to the validated learning. In the beginning something should be attempted to be written in all fields, but some fields may have vague answers or no answers at all. Each empty field is a good indicator of an area that is risky and needs to be studied further. The fields have a rough order, in which to fill them. After each Lean Startup stage, certain fields should have somewhat definitive answers.

2.7.9 Lean Startup Stages

Which questions to ask and which measures to measure, depends on the stage of the start-up. Lean Startup stages have different definitions, all of which follow the same pattern. Eric Ries speaks of "problem validation", "solution validation", "MVP building", three engines of growth: "sticky engine", "virality engine" and "paid engine", and "inorganic growth" [Ries11, Crol13]. Ash Maurya describes Three steps: "Problem/Solution Fit", "Product/Market Fit" and "Scale" [Maur12]. Croll defines five stages: "empathy", "stickiness", "virality", "revenue" and "scale" [Crol13].

The definition by Croll is used when answering RQ2 in the empirical study (chapter 4.2). Therefore it is used as a basis for explaining the start-up stages.

The empathy stage is where the problem, solution hypotheses and, to some degree, market hypotheses are verified. The "problem and solution fit" is made by finding potential users and asking them about the problem and the solution or, to some extent, monitoring how they use the MVP.

The problem validation tries to verify that the problem is painful enough, enough people care about the problem and that people are already trying to solve the problem. The data gathered is mostly qualitative and the result of asking people face to face [Ries11, Crol13, Maur12]. The entrepreneur asks questions like "Do
consumers recognize that they have the problem you are trying to solve?", "If there was a solution, would they buy it?", "Would they buy it from us?" and "Can we build a solution for that problem?" [Ries11].

The solution validation tries to verify that the solution being developed will solve the problem for the users in such manner that they are willing to pay for it. The solution is made more clear, by developing the MVP as well as interviewing and monitoring how the users use it.

Validating the market at this point is usually done by setting a price for the product and asking users if they would pay that price. The price can then be adjusted according to the feedback.

Typical metrics to follow are the attention, acquisition and activation measures. In empathy stage the problem and the customer segment parts of the lean canvas should be updated.

In stickiness stages the product/market fit is made. At this point there is a MVP that is being used by small amount of enthusiastic users, who may even have paid for the MVP.

Ries lists three engines of growth that can be used to increase the amount of users. First of them, the sticky engine of growth, focuses on building a product that people keeping existing customers and. It measures the retention rate, optimizing the product in a way that users return to the service. Main measure is the churn rate: a percentage of the users abandoning the service over time. By getting the churn rate low, will increase the customer lifetime value, providing that the cost of customer acquisition stays the same.

The metrics followed in the stickiness stage, are activation and retention metrics. In this stage the unique value proposition and the solution parts of the lean canvas should be updated.

The virality stage is where the knowledge of the product is spread by the users who are using the product. The second of Ries’ engines of growth, the viral engine of growth, focuses on attracting new users with either word-of-mouth or inherent virality. Virality through word-of-mouth, where people are talking about the product and recommending it, is very hard to come by, as it requires building a product so good, that users are passionate enough about it, to spread the word. Or it requires great skills in search engine optimization (SEO), to improve visibility of a landing page or product in search engines.
Inherent virality is easier to control, as it is built in the system. In inherent virality, using the product, advertises the product. For example every time someone sends someone else files through a file transferring services, the service advertises itself to the recipient.

The funnel metrics that are in focus in virality stage are acquisition rate and referral rate. In this stage the channels part of the lean canvas should be updated.

In the revenue stage the entrepreneur has to validate that business makes money. Metrics such as revenue per customer (RPC), customer acquiring cost (CAC) customer lifetime value (CLV). The customer segments that are making purchases should be analysed to understand who they are. Means to extract revenue from users the price should be tried out and what effect the price has on the funnel metrics should be checked. Churn rate tells when the price is becoming too high.

The paid engine of growth also focuses on attracting new users, but does it through advertising which costs money. Lean Startup urges the entrepreneur to not spending a lot of money on advertising before there are grounds for it. In revenue stage the start-up should be making money and more money can be put to customer acquisition. When the CLV is known, the CAC can be checked against it. If CAC is more than CLV, the start-up will run out of money eventually.

The focus of the viral and paid engines is on the rate of new acquisitions. In revenue stage the revenue stream and the cost structure parts in lean canvas should be updated.

In scale stage the entrepreneur takes first serious look at the competition and the start-ups place among them. The start-up is monitoring a lot of things that are helpful in this stage and one thing to monitor is, as the business scales, the costs of different functions don't scale more. Scaling makes sense if it brings in more revenue as the volumes increase.

In scale stage the (formal) channels and the unfair advantage parts of lean canvas should be updated.

All the engines of growth will eventually run out. The viral engine and the paid engine affect the amount of new users and exponential growth is not possible forever. When these start running out, the numbers will go down indicting none or negative growth. The sticky engine may have constant churn rate, but as it is a percentage of the total number of the users, the number of leaving users increases as the number of total users increase. If the rate of new users in time period is constant, the number
of leaving users will at some point in time match the number of new users and there will be no more growth.

2.7.10 Critique for Lean Startup

Although on the surface Lean Startup seems legitimate and seems to make sense at the cursory glance, the apparent novelty of Lean Startup raises questions whether it is really something totally new and whether there is any foundation to the claims it is usually associated with.

Many blogs offer criticism for Lean Startup. Some of them provide good insight and add to the discussion, after all, Lean Startup is something that still evolves and is refined by the community. Some of these criticisms are misunderstandings and some are bitter whines.

Nick Penning gives a list of claims about Lean Startup that should be taken with a grain of salt [5]. He makes the notion that Lean Startup methods have not been proven scientifically, mainly because there isn’t yet enough body of data to study. He also raises the question about the lean part of the Lean Startups. Lean is about eliminating waste and much of Lean Startup is about eliminating the waste of building something nobody wants.

The fact that about 75% to 80% percent of start-ups fail in the first two years has been outed in many Lean Startup introductions. This is mainly because the failing start-ups are building something nobody wants. There is no evidence that says that with Lean Startup, this rate will go down. It may just be that that the reasons for failing may differ. The start-up may lack the resources or ingenuity to pivot to success, or the start-up may be outperformed by competition that takes the initial idea and makes it a success so efficiently that the "first in market" start-up is overthrown.

John Finneran lists start-up lessons learned in his blog [6]. He raises the problem of funding. Customers who are willing to buy a barely ready product are few and far apart. The business angels can’t be expected to fund any learning experiments an entrepreneur has in mind. Whether this is true, the point of validating and gathering data is to learn what the customers want and need in order to create such a product, that receives a positive commercial reaction from customer or to convince possible venture capitalists and business angels to invest in their start-up.

Finneran also raises the question of why spend on tests to verify things you should
already know. It’s a valid question and Lean Startup cannot be taken too literally. In practice, a line has to be drawn to select which assumptions to verify and which not to. As the person making that decision is the entrepreneur whose judgement has been initially challenged. For a starting entrepreneur, or a new start-up it is advisable to verify more hypotheses to gain better understanding of the market situation and the customer segment. Once the understanding is more solid, and the entrepreneur has proved to have valid judgements, there may be grounds for alleviating the need for some verifications.

Finneran’s blog also makes it necessary to remind everyone that Lean Startup is about minimizing waste, not just minimizing the one waste of building the product no-one wants. The other wastes, such as low quality and faults, still need to be eliminated.

The interview in chapter 5.2.7, brought up the idea that pivoting may end up with entrepreneurs not wanting to build the product users want.

### 2.8 Measurement Goals

The table 1 summarizes the targets of measurement over the years and in Lean Startup. The column "Domain" explains a topic in the software development history. The column "Measurement Target" defines whether the measuring focuses on software development resources, processes or the product, or the market. The column "Goal" summarizes the goal of the measurement activity in the topic.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Measurement Target</th>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programmer</td>
<td>Resource</td>
<td>To assess, evaluate and select the best to ensure project success</td>
</tr>
<tr>
<td>Hardw. opt. (2.2.3)</td>
<td>Resource</td>
<td>Optimize the hardware usage</td>
</tr>
<tr>
<td>SW development process (2.2.4)</td>
<td>Resource</td>
<td>to choose best resources and tools for projects</td>
</tr>
<tr>
<td>SW development process (2.2.4)</td>
<td>Process</td>
<td>To estimate SW project cost and to be able to manage a SW project</td>
</tr>
<tr>
<td>SW development process (2.2.4)</td>
<td>Product</td>
<td>The correctness of the end product</td>
</tr>
<tr>
<td>Method</td>
<td>Type</td>
<td>Resource/Process</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Software quality (2.3.3)</td>
<td></td>
<td>Resource</td>
</tr>
<tr>
<td>Software quality (2.3.3)</td>
<td></td>
<td>Product</td>
</tr>
<tr>
<td>Structured programming (2.2.6)</td>
<td></td>
<td>Resource</td>
</tr>
<tr>
<td>Waterfall (2.3.1)</td>
<td></td>
<td>Process</td>
</tr>
<tr>
<td>Complexity Measure (2.3.2)</td>
<td></td>
<td>Resource</td>
</tr>
<tr>
<td>The definition of quality (2.3.3)</td>
<td></td>
<td>Resource</td>
</tr>
<tr>
<td>The definition of quality (2.3.3)</td>
<td></td>
<td>Product</td>
</tr>
<tr>
<td>COCOMO (2.4.1)</td>
<td></td>
<td>Process</td>
</tr>
<tr>
<td>Spiral Model (2.4.2)</td>
<td></td>
<td>Resource</td>
</tr>
<tr>
<td>Spiral Model (2.4.2)</td>
<td></td>
<td>Process</td>
</tr>
<tr>
<td>Spiral Model (2.4.2)</td>
<td></td>
<td>Product</td>
</tr>
<tr>
<td>Open source (2.4.4)</td>
<td></td>
<td>Resource</td>
</tr>
<tr>
<td>Open source (2.4.4)</td>
<td></td>
<td>Product</td>
</tr>
<tr>
<td>Scrum (2.4.3)</td>
<td></td>
<td>Resource</td>
</tr>
<tr>
<td>Scrum (2.4.3)</td>
<td></td>
<td>Product</td>
</tr>
<tr>
<td>Six sigma (2.4.5)</td>
<td></td>
<td>Process</td>
</tr>
<tr>
<td>Method</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Six sigma (2.4.5)</td>
<td>Product</td>
<td>Evaluate and measure the requirements and improve the product</td>
</tr>
<tr>
<td>CMM (2.4.6)</td>
<td>Process</td>
<td>Find out the CMM level and what to do to gain next level</td>
</tr>
<tr>
<td>DSDM (2.5.1)</td>
<td>Resource</td>
<td>Source code size for each prototype</td>
</tr>
<tr>
<td>DSDM (2.5.1)</td>
<td>Process</td>
<td>Effort and time expended for each prototype, prototype size, effort expanded and function-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>alities in each timebox</td>
</tr>
<tr>
<td>DSDM (2.5.1)</td>
<td>Product</td>
<td>Functionality delivered in each prototype</td>
</tr>
<tr>
<td>Customer development (2.5.2)</td>
<td>Product</td>
<td>Problem/Solution fit, Product/Market fit</td>
</tr>
<tr>
<td>Customer development (2.5.2)</td>
<td>Market</td>
<td>Qualitative measures of the customers and the market from users, customers, internal and external experts</td>
</tr>
<tr>
<td>PSP (2.5.3)</td>
<td>Resource</td>
<td>Source code quality, productivity to evaluate and improve developer skills</td>
</tr>
<tr>
<td>PSP (2.5.3)</td>
<td>Process</td>
<td>Effort estimations and realizations to evaluate and improve estimation skills</td>
</tr>
<tr>
<td>PSP (2.5.3)</td>
<td>Product</td>
<td>Number of faults to improve developer skills</td>
</tr>
<tr>
<td>Agile (2.6.1)</td>
<td>Process</td>
<td>Improvement ideas to improve the way of work, task effort estimations to plan work, velocity to assess teams capability, work left to follow the progress, task effort estimations and business value to prioritise work</td>
</tr>
<tr>
<td>Agile (2.6.1)</td>
<td>Product</td>
<td>Feedback to be able to react to customer wants and needs</td>
</tr>
<tr>
<td>Design for Six Sigma (2.6.2)</td>
<td>Product</td>
<td>Requirement metrics to measure the product compliance to customer requirements</td>
</tr>
<tr>
<td>Design for Six Sigma (2.6.2)</td>
<td>Process</td>
<td>Requirement metrics to measure the process compliance to customer requirements</td>
</tr>
<tr>
<td>Lean Software Development (2.6.3)</td>
<td>Process</td>
<td>Time from order to delivery, to expose wastes in processes</td>
</tr>
<tr>
<td>Lean Software Development (2.6.3)</td>
<td>Product</td>
<td>Realization of business case to measure ROI, Customer satisfaction.</td>
</tr>
<tr>
<td>Lean Startup (2.7)</td>
<td>Product</td>
<td>Problem/solution fit, product/market fit</td>
</tr>
</tbody>
</table>
Funnel metrics to learn about the customers and the market, user behaviour to find out who the users are

Table 1: Measurement Targets and Goals

In his article [Fent99] Fenton describes the gap between the academic world and the business world when it comes to metrics. He makes a note that mainly the source code size metric (LOC) and fault metrics have been successfully adopted into practical use. There seems to be little motivation to take different metrics into use, as their usefulness has not been proven. The predictive qualities of software quality in metrics usually correlates with LOC. The motivation in companies to implement measuring systems is mainly from outside pressure such as achieving a certain CMM level, required by customers.

3 Research Design

3.1 Literature Study

The software measurement study was performed by finding and reading books and articles on software measurement, software quality and software processes. Starting point for the history was selected to be the birth of software industry around the middle of the 20th century. Research was done to find out what kind of questions plagued the industry at which point in time, how were the answer to those questions discovered, and how they affected the goal of measurement. Soon after the sixties the industry started to disperse, so to maintain some structure in this thesis, only some more fundamental points of more contemporary technical advancements were included, emphasizing those of interest from the point of view of Lean Startup.

The history of Lean Startup was researched from the sources, mentioned in Lean Startup literature and tracing those backwards. This took us back to the post-war Japan in middle of the 20th century and the birth of TPS. Other contributing concepts found on the way were agile methods and customer development.

The literature study of Lean Startup (chapter 2.7) was done by reading the main books about the subject [Ries11, CroI13, Maur12] and researching articles of mentioning Lean Startup. Lean Startup as a concept is only a few years old, so some
material is found only from the internet. Some important blogs and slide-shares were studied to find more hands-on information and criticism on the subject.

3.2 Survey

To find out how things are being done in real world, a survey, with a questionnaire and series of interviews, was done.

The research objective for the survey was to find out, how different start-ups, in different stages, gather and use measurement data. The target audience for the survey was active start-ups as well as pending and discontinued start-ups. The plan was to interview a handful start-ups and to attract dozens of responders for the questionnaire.

The questionnaire was to be sanity checked by professionals and tested with, at least, one start-up. The distribution of the questionnaire was to be done in iterations, learning from each iteration.

3.2.1 Questionnaire

The Questionnaire tried to shed light on the question "what kind of information need do start-ups have in which stage?". To that end it was necessary to first find out the approximate stage the start-up was in, and then what kind of information needs they thought they had.

The survey was divided into two sections; Demography and Measurement. The first section gathered background information from the start-up and the entrepreneur. The topics asked in the section were: familiarity with Lean Startup principles, if it's the first start-up, amount people working in the start-up, what business the start-up is in, what is their intended customer segment, how is the start-up funded, how often have they pivoted and why and how the start-up was doing in their opinion.

The demography section also asked the entrepreneur to tell which claims apply to their start-up. This information was used to estimate the stage the start-up was in. The stage was approximated with following logic: In empathy stage the entrepreneurs are pondering the validity of the problem and the solution. They do not have MVP or users using the MVP. In stickiness stage start-up has a MVP and users and are trying to get feedback from them. They are improving the product not yet actively trying to get new customers or users. In virality stage acquiring
new users and customer is in the focus. the entrepreneurs are interested in different market sections. They may be not be getting enough revenue to sustain the business. In revenue stage the entrepreneurs are interested in pricing issues and revenue models and how the customers find the product valuable. In scale phase the entrepreneur are financially comfortable and interested in expanding the business by multiplying the business model.

In short this roughly translates as:

<table>
<thead>
<tr>
<th>Lack of MVP</th>
<th>Empathy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lack of Users</td>
<td>Stickiness</td>
</tr>
<tr>
<td>Lack of Customers and Revenue</td>
<td>Virality</td>
</tr>
<tr>
<td>Lack of financial success</td>
<td>Revenue</td>
</tr>
<tr>
<td>Otherwise</td>
<td>Scale</td>
</tr>
</tbody>
</table>

The measurement section focused on the information needs of the start-up and how they are met.

First question listed questions deemed important by Lean Startup literature. The entrepreneurs were asked to select the ones that are the most important to them at the moment. If none of the question fitted, a text box to give a free-form answer was given.

Second question asked, what kinds of decisions can answering those questions allow the entrepreneur to make. Third question asked, the entrepreneurs to list the most important metrics they have, and what makes them important. Fourth question asked, what is the start-up’s One-Metric-That-Matters and why.

An optional questions asked the entrepreneur to list important metrics the have previously measured and used, and why they were important. These questions were asked to find out, if start-ups in the later stages consider similar measures important in hindsight, as start-ups, in the earlier stages, consider important at the moment.

Another optional question asked the entrepreneurs to list metrics they would like to use and measure, and why they think they are important. This question attempted to gather those metrics, which entrepreneurs think are important, but are not able to measure at the moment.

The information needs of the start-up was asked by listing questions that according to Lean Startup should be important, and asking the responders to choose which they think are the most relevant questions to their start-up. If none of the questions
were appropriate, a free-form answer box was provided. The survey asked also what actions finding answers to these questions allows them to make.

3.2.2 Interviews

In the course of this study, at least one entrepreneur would be contacted for being interviewed. The interviews were to be held in informal situations and recorded for later reference. The interview would follow the structure of the questionnaire.

4 Literature Review Results

This chapter answers the research questions RQ1 and RQ2. They are answered based on the literature study documented in chapter 2.

4.1 Goals of Software measurement

The first research question of this thesis is "What kind of software measures there have been and how the goal of measuring has changed over the years?".

In 1984 Basili described in his paper a three step process for discovering measures that are more useful for the people for whom they are gathered. It starts with creating a list of goals of data collection. The goals fall into two categories; the ones that are relevant for the particular methodology for which claims are made, and the ones that are more general in nature and apply to other methodologies. For each goal, a set of questions is devised in such way that answering the questions would shed light if the goal is met. For each question a set of measures are chosen in order to gain enough information to answer the questions adequately. [Basi84]

Fenton divides software measures into three different classes: processes, products and resources. The entities that are the targets of the measuring have internal and external attributes. Internal attributes can be measured by examining solely the entity itself. External attributes are the ones that can be measured only through their relationships to their environments. He also divides the uses of measurements in two categories: assessment and prediction. In order to characterize and evaluate an entity, a model needs to be made with attributes and characteristics that can be measured. Based on the model, improvements can be worked out and their impact can be predicted in order to select the best ways to improve. When the
improvements are implemented, their impact can be measured and verified [Fent97, Fent94, Park96].

4.1.1 Measuring resources

In the 1950s, the industry was immature and lacked definition. The measurement focused on resources. The hardware had already clear and measurable attributes, but software was more ambiguous. The managers didn’t understand the industry or programming. The programmers were the first software resource being measured (chapter 2.2.1). Programming was slow and costly. The hardware resources were scarce.

The optimization of usage of the resources was the next focus. Programmers were selected based on testing. Software was optimized for size, memory consumption and CPU consumption. There was little need to use higher level languages, unless the compilers generated as efficient assembler as developing by hand with assembler.

By the end of the sixties the software projects and the programs had grown so much, that they had become impossible to test reliably and structural programming was a necessity (chapter 2.2.6). The complexity of source code was defined and measured (chapter 2.3.2). Some software quality attributes and qualitative measures were introduced in order to give a piece of software quality assessment (chapter 2.3.3). The measured attributes were such as correctness, optimized for resource use, intelligibility and modifiability.

in the 70s and 80s the software resource metrics were used mainly to help characterize the different quality attributes, such as maintainability, testability and re-usability. The metrics were used to help in estimating and predicting software project efforts. In the mid 1990s the software resource metrics were used to help software developer analyse and develop their own skills (chapter 2.5.3).

Much of the open source code, libraries, more powerful higher level languages and increased performance and capacity of hardware have replaced the need for code tweaking to structured and disciplined development where software consists of pre-made customizable components or libraries. The measuring is done automatically and the metrics are used by different tools. the source code size has kept its popularity, because it is a simple, understandable metric, that is easy to gather and has some explanatory power.
4.1.2 Measuring processes

In the 1960s the topic of software development process became important, because the scale of the software projects grew. In waterfall (chapter 2.3.1) effort estimations come from the SLOC and subjective estimations of the developers’ skills. The project work was divided into parts and schedules were set. The schedule was measured and milestone slippage was followed, to verify that the project stayed on it’s course and if it didn’t, corrective actions could be taken.

In COCOMO (chapter 2.4.1) and the spiral model (chapter 2.4.2) rapid prototypes with short iterations are used to gain knowledge in a software development project. The estimations are made from using different methods (such as resource metrics, expert opinions and historical quantitative data) and the estimation is a synthesis of all the data. The main driving point was to gain as much information with the limited resources that the project team has.

With CMM (chapters 2.4.6, and 2.4.6) the measuring and improving software processes was introduced. In the first stages after the initial stage, the processes are designed and written down. In the last two stages, measurements are used to evaluate and to improve the processes.

Six Sigma (chapter 2.4.5) in the 1980s was developed for improving project quality. In SixSigma measures are defined for the process under development and a baseline is taken. The improvement is designed and implemented and the measurements are taken again and compared to the baseline to see if the improvements were successfully implemented.

DSDM (chapter 2.5.1) focuses on bringing valuable functionality to the customers and measures implemented functionality and development effort per prototype and timebox. The developers work closely together with the customers to get feedback and understand the customer.

The agile software development (chapter 2.6.1 recognized the same uncertainty as of software development as Barry Boehm in twenty years before (chapters 2.4.1 and 2.4.2). In the beginning it’s not completely know, what the end product will look like, but the process guides the development toward it in small iterations. The effort estimations are made together based on whatever methods the team members wish to use. The team productivity is in each iteration and the predictions are based on measurement data from previous iteration. The progress is followed by measuring how much work is left. The business decisions are based on value estimations of
each feature and their effort estimations. The process development happens between iterations by selecting those tools and processes that bring improvements to the way of work.

Lean Software Development (chapter 2.6.3) mentions the cycle time as one of the most important metrics. The aim is to find out how long does it take to repeatedly and reliably respond to a customer request. When the cycle is established, its optimisation can be started.

The shift in process measurement has changed from predicting the project effort and schedule to finding out what to build and optimizing the time to do that. All the time the projects gather more information about their ways of work and the problem domain. The more information is gained, the more informed decisions can be done, and decisions are fixed in as late as possible.

4.1.3 Measuring product

In the 1950s and 1960s the software product was though as a black box, something was given to it as input and something came out. These input and output pairs needed to match the defined and implicit product requirements in order to be said it worked correctly. Another aspect that was measured, regarding the product, was the cost of development (chapter 2.2.1).

Measuring the software quality at the end of 1960s (chapter 2.2.5) brought new external metrics to software products such as usability and learnability. The measures were based on subjective data made quantitative. The structured programming (chapter 2.2.6) allowed for measuring decision coverage and path coverage when designing tests.

With the waterfall project model and spiral model the product quality, in regards of end customer, was improved, by getting requirements from users or customers, and involving users in the development process, prior to the final release.

The 1980s came with the concept that if the software is developed using good software processes and practises, the quality will be good. The quality was followed with measures and processes, that were meant to ensure that the end product would meet it’s quality requirements.

In the scrum team (chapter 2.4.3), the team members were hand picked by using the intuition of the superiors. The project team was monitored through checkpoints. The teams were autonomous and team was managed by changing the team members.
The teams were given a task, and their performance was analysed, by evaluating how well the product solved the problem given in the task.

In the naughties, DFSS took the improving of processes of the Six Sigma and transferred into product development (chapter 2.6.2). In DFSS the requirements of the product were analysed and quantitative measurements were named and measured for the product.

In Lean Software Development (chapter 2.6.3 the customer satisfaction is one of the most important metrics and it’s assesses through customer feedback through customer interaction.

4.1.4 Measuring business

In 1990s Steve Blank argued that it doesn’t matter how fantastic features it has, or how well it is made, if no-one is interested in the product, There’s no need to make it.

In customer development (chapter 2.5.2), building a repeatable profitable business model is what matters. The product is one aspect and it’s developed according to feedback from a few customers. The product is used in order to understand market and the customer segment. Customer development metrics are mostly qualitative data, feedback from users, customers and experts of different fields, but it does have some quantitative metrics, especially in the as the start-up moves to later stages.

Lean development lists financial success as one of it’s most important metric. It is the result of the two other important metrics: the efficiency of the development and delivery processes and the customer satisfaction.

Lean Startup brings a lot of quantitative measurements into software development. The main purpose of these metrics, is to learn from the users, customers and market about the users, customers and market. By experimenting and taking measures, entrepreneur tries first to find a problem/solution fit, then product/market fit and finally try to scale the business. These results help in making decisions with the product and all other aspects in the business model.

4.1.5 User Involvement

Since 1968, when Rubey and Hartwick called for time, when user would have a say in what kind of program was being developed, the involvement and importance of user has increased. With waterfall process (chapter 2.3.1), the developers or customers
were able to create requirements based on what they thought the users need. The software was developed based on how those requirements were interpreted. The user was involved in the beginning and in the end of the project. In spiral model (chapter 2.4.2), feedback from the customer was requested in different milestones and demo applications were used to get information. The user's involvement during the software process was in small bursts. In agile development (chapter 2.6.1, to ensure customer satisfaction, the customer became a part of the daily routines of a scrum team and worked in the same place. The users periodically received working software to give feedback on and the product owner was always available to answer any questions the team might have. In Lean Startup (chapter 2.7) the customer and the market is constantly tested and measured and actions are taken based on those findings. The users are involved either knowingly or their behaviour is analysed unknowst to them.

As the importance of understanding the user increased, the level of involvement with the user increased. This has increased the mental stress with the user. With Lean Startup the mental stress on the users can be decreased, because the users' behaviour can also be monitored without too much extra involvement from them. This also allows the entrepreneurs to get answers, no user could give them.

4.2 Lean Startup Metrics

The second research question of this thesis is "What kind metrics are suggested to be used with Lean Startup and what do they measure?".

A typical Lean Startup entrepreneur is assumed to have the capability of measuring numerous things. He's also expected to know which ones to focus on and select the one measure that describes how the start-up is doing. "Gather many, focus on few" is what is often said in regards of Lean Startup measuring.

4.2.1 Empathy stage

In empathy stage the goal is to identify a problem which people care about. They are willing to pay for the solution, if that the solution solves problem in a way, that is feasible. This requires "getting out of the building", talking to people, asking them qualitative questions and trying to figure out, what the customers need, instead of what they tell they need. To this end the entrepreneur should pay more attention to listening to users or early adopters than telling them their visions. Important
questions to answer are: "does anyone care?", "is my definition of the problem correct?", "does my solution solve the problem?", "how are people currently trying to solve the problem?" and "would people pay for my solution?"

"Getting out of the building" doesn't always require actual physical act of getting out of a building. In addition to asking early adopters directly over a coffee, the entrepreneur can, for example, create a fake landing page of a non-existent product and try to attract traffic to the page. If the page gets little or no attention, then it's a good indicator that the problem is not important for others and pivoting might be in order. To take things further, there could be a possibility to indicate attempt to purchase for example in the form of a non-functioning purchase button on the page. Pressing the button would indicate that a purchase would have been made. If the page attracts no purchase attempts, then it should be checked, whether the wording on the page, or the design of the page, should be improved. In case of a purchase attempt, a notification, that would not leave the customer disappointed due to the missing product, should be given. Each of the purchase attempts validates the problem/solution fit more [7].

In empathy stage the acquisition and activation metrics are gathered and analysed to verify that the problem/solution provides answers to someone's problems and to measure the amount of interest in the solution and willingness to pay for it.

4.2.2 Stickiness stage

In stickiness stage, the focus is on getting the users to come back and continue to use the product. To get to the stickiness stage, the MVP should be up and running. With users actually using the product, the entrepreneur should be able to gather metrics and analyse, who the people are who are using the product, and how they are using it. This may give the entrepreneur valuable data regarding why things happen and how the product could be improved. It could suggest new customer segments to target.

The registration usually requires giving an email, which can then be used to directly ask questions from the user to explain their behaviour. If the user is not using the product, a mail can be sent to ask, whether there is a problem. If a certain pattern in the users use of the product is detected, an opinion on a certain new feature can be asked and so on. The metrics that are followed, are the activation and retention metrics.
The goal of testing is to validate, that the users like to use the product repeatedly. It has to be decided on product level, which actions mean that the user has been activated. It has to be something, where user shows more interest than just visiting the page, usually registering or downloading and installing a software.

It’s usually not enough, that the users use the service just once. A successful business model requires, that the customers use the service repeatedly. Measuring the retention requires gathering data from how the user uses the product, for example how often they use it and for how long in one go. This paints a fairly good picture, of how interesting the product is to users.

The user behaviour data can be used to optimize the product’s usability, by focusing on the most used features that lack the most. With split testing the impact of each improvement can be measured and analysed. Improving the products usability, improves the customer satisfaction and increase stickiness and referral metrics.

4.2.3 Virality stage

In virality stage, the focus is on crossing from the small group of early evangelists, to customers in the main market segment. The product has been optimized for a small group of enthusiastic people. Now it’s time for optimizing the channels to get large numbers of new customers, and see how the mass market sees the product.

When trying to get new main market users to repeatedly use the product, the entrepreneur is faced with similar problems as with stickiness stage. It’s safer to avoid making assumptions, about how the mass market behaves, based solely on findings with the small group of early evangelists, with whom the start-up has worked so far. The acquisition and referral metrics are followed to see any changes.

Although, some of the problems are the same as previously, at this point the channels to new customers are being developed. The paid engine of growth is one of the three engines that work for product virality. Entrepreneur can pay for advertisement and see which of them and in what volume bring in users. The acquisition rates should be shown in cohorts by channels, in order to find out the best channels and marketing strategies and optimize the impact of any changes in each of the channels. Investing heavily on advertising is not advised, until in revenue stage the CLV metric is known and limits for CAC can be calculated.

The second engine of growth, the viral engine of growth, gets new users by word of mouth. Users referring the product to other users. If the referral is inherent or
implemented in the product, the effect of referrals is easier to measure. When user refers the product to another potential user through the product, the referee becomes known to the product, and can be later checked if the referral has been successful. If the referral happens outside the product, i.e. word-of-mouth, its effect can be approximated by reducing the impact of paid engine of growth and the inherent referrals from the total ratio of new users. What ever remains, can at some level be attributed to word-of-mouth.

The third engine of growth, the sticky engine, increases the amount of new users by the logic that, if the users use the product and keep using it, they probably like it and find it useful. Therefore they are more likely to refer it to other people. With this in mind, the referral metrics also tell how liked the product is. One aspect about the referral metric is that, users refer the product to someone who they think should use it. Therefore by analysing who referees are, may open new ideas about new customer segments.

As previously, the behaviour of the users should be monitored to find out how the product is being used.

The channels used to attract users should be shown in cohorts to optimize the use of channels. The product should be optimized so that the activation and retention rates are as high enough. Unless the virality is inherent, a system to enforce referrals should be implemented so that the product virality can be measured and possibly controlled.

4.2.4 Revenue stage

The revenue stage is about proving the business. In this stage the different ways to monetize the users using the product are thought of.

To measure only how much money the start-up is making is not advised. Better metrics are revenue per customer. This ratio can be used to estimate CLV. When CAC is known, ROI can be calculated.

It may be that the users who use the service are not willing to pay for the service. In such case it’s good to ask questions like "what would the users do if service was removed or required paying?" or "who cares enough about the user’s problem that is willing to pay?". 
4.2.5 Scale stage

In scale stage the entrepreneur turns the focus to the competition. The start-up now needs to find it’s place in the market and start increasing volume. Lean Startup considers scaling a bit outside of scope as the start-up is becoming a more business-like company. In scaling stage, all the measures that have been gathered previously are still useful to gather. Entrepreneur needs to check that scaling the individual activities doesn’t increase the cost beyond a healthy company.

5 Survey Results

This chapter discusses the results of the survey and answers the research questions RQ3 and RQ4. They are answered based on the information gathered through the survey described in chapters 3.2.2 and the results documented in chapters 5.1 and 5.2.

5.1 Questionnaire

The first version of the questionnaire was tested with two start-ups, one lean and one more traditional, in a face-to-face meeting. After changes from those sessions were made, comments from a lean start-up expert was asked for the updated version. The questionnaire was then released to a small group of known start-ups. When none of them replied, it was easy to ask for reasons, which were mostly that they had forgotten and some were not sure they understood the questionnaire. As a result great simplifications were made to the questionnaire, and the next iteration would ask for volunteers to fill the questionnaire. This allowed the questionnaire web application could keep anonymously track of the volunteers, and send reminders accordingly. The questionnaire also had a bonus question to find out responders’ motivations for answering the questionnaire, in order to attract more responders in the future iterations.

One of the planned channels was used to get volunteers. As the number of volunteers was greatly below what was expected, it was thought that, all-in-all, it’s easier for people to answer questionnaires, if there is a link in the message, rather that sending an email and receiving an invitation to that questionnaire. Two different versions of messages were written and sent to two different channels to see if the wording
made great difference. Again the amount of responders was minimal. Once the rest of the channels became unusable, due to either the contact person not responding to mails or calls, the contact person changing without naming a successor or due to technical problems, which prohibited sending messages through the channel, the questionnaire was closed.

Only the request through development community Protomo [8] and hands-on seed investor Polte [9] resulted in any responders. The number of responders in the survey was not big (n=7), and the findings cannot be statistically meaningful in any level. However, there were curious patterns, that could encourage further study.

Below is a table describing all the start-ups that answered the surveys. The responders were quite heterogeneous which provided a little broader view of what could be possible.

<table>
<thead>
<tr>
<th>Name</th>
<th>Business type</th>
<th>Funding</th>
<th>Stage</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trad-1</td>
<td>media sites, create content and advertise for revenue</td>
<td>B2B</td>
<td>Own Money</td>
<td>Empathy 7 - Badly</td>
</tr>
<tr>
<td>Trad-2</td>
<td>Own</td>
<td>B2B</td>
<td>Own Money</td>
<td>Empathy 4 - Could be better</td>
</tr>
<tr>
<td>Lean-1</td>
<td>Mobile application with in-app purchases</td>
<td>B2C</td>
<td>Own money, government</td>
<td>Stickiness 5 - Struggling</td>
</tr>
<tr>
<td>Lean-4</td>
<td>Software as a service, mobile application with in-app purchases, media sites, create content and advertise for revenue, platform for user generated content, two-sided marketplace where buyers and sellers can come together.</td>
<td>(B2C), B2G</td>
<td>Own Money</td>
<td>Stickiness 7 - Badly</td>
</tr>
<tr>
<td>Trad-3</td>
<td>Software as a service</td>
<td>B2B, B2G</td>
<td>Own Money</td>
<td>Stickiness 4 - Could be better</td>
</tr>
<tr>
<td>Trad-4</td>
<td>Software as a service</td>
<td>B2B</td>
<td>Own Money</td>
<td>Stickiness 3 - Ok</td>
</tr>
<tr>
<td>Name</td>
<td>Questions</td>
<td>Goal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trad-1 (Empathy)</td>
<td>&quot;What's compelling about the product?&quot;</td>
<td>The incentive for the customer to get involved.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trad-2 (Empathy)</td>
<td>&quot;What is the problem you are solving?&quot;, &quot;Who is the competition?&quot;, &quot;What is the most profitable pricing model?&quot;, &quot;What's compelling about the product?&quot;</td>
<td>No answer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lean-1 (Stickiness)</td>
<td>&quot;Do customers make it through your activation flow?&quot;, &quot;How do users become aware of your product?&quot;</td>
<td>No answer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lean-4 (Stickiness)</td>
<td>&quot;Is our customer segment a viable one?&quot;, &quot;Will users pay for the solution?&quot;, &quot;What price will the customers bear?&quot;, &quot;What's compelling about the product?&quot;, &quot;Do users make it through my activation flow and become active users?&quot;, &quot;What are the usability hot spots?&quot;, &quot;Does our Minimum Viable Product demonstrate and deliver on our Unique Value Proposition?&quot;, &quot;How do users become aware of our product?&quot;</td>
<td>Allows us to generate cash flow</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trad-3 (Stickiness)</td>
<td>&quot;Who is the competition?&quot;, &quot;How do you find early adopters?&quot;, &quot;Will customers pay for the solution?&quot;, &quot;What price will they bear?&quot;</td>
<td>To find out who controls the money</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 4: Startup questions and measurement goals

<table>
<thead>
<tr>
<th>Name</th>
<th>Measures</th>
<th>OMTM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trad-1</td>
<td>The quantity interest, the time-span interest and the workability of the interaction.</td>
<td>The amount of customers and the amount of the involvement of each other.</td>
</tr>
<tr>
<td>Lean-1</td>
<td>No answer</td>
<td>No answer</td>
</tr>
<tr>
<td>Lean-2</td>
<td>User feedback, facebook user profile information, mobile-app analytics, registration rate</td>
<td>The percentage of users that start chatting with other users.</td>
</tr>
</tbody>
</table>
Lean-3 (Stickiness) | Cash Flow, users, conversion-rate, retention. | Start-ups live or die on cash flow. 
---|---|---
Trad-3 (Stickiness) | what stuff is being used and what are the changes during time | how many users 
Trad-4 (Stickiness) | Sales and sales agent numbers | Amount of licences sold 
Lean-3 (Scale) | Most important one, hands down: "User attends first activity - yes/no" Other: "User engages with emails/push notifications", "User goes through the app introduction", "User visits certain screens" and "User attends first activity - yes/no". | That's the only way for the user to experience the UVP (same thing for Uber, you only get it once you get a ride once). Once it happens, the user gets hooked and their wallet increases significantly. 
Lean-2 (Revenue) | amount of downloads, amount of runs, amount of users returning to the application, amount of tunes / takes / clips created, amount of IAPs purchased, amount of takes shared, | sum of revenue sum of revenue, it is the ultimate goal and derives from the other metrics. 

Table 5: Startup important metrics and measures

What is being measured differs significantly between traditional and Lean Startups. When asked, the Lean Startups listed numerous exact measures some of which corresponded with their estimated stage. The traditional start-ups had vague or ambiguous answers, or no answers at all. Two commented on the difficulty of the questions and one didn't answer.

Surprisingly, regardless of the approximated state of the start-up, cash flow was often named to be one of the most important metrics. As stated by one answer "Startups live or die on cash flow". Four out of the five start-ups funded solely by own money, were wondering about revenue. None of start-ups funded by other than
own money, listed revenue related questions as the most important.

5.2 Interviews

After the lack of success with the questionnaire, the importance of the interviews was heightened. Seven people were interviewed during this study in regards of eight start-ups.

Some of the interviewees were familiar with Lean Startup methodologies and some weren’t. Only one of them claimed actually to be a Lean Startup (chapter 5.2.5). The interviews took about half an hour per start-up. For some start-ups (chapters 5.2.2 and 5.2.3), several people were interviewed.

As the data is not used to prove anything, and in order to get maximum amount of information, second hand information was also accepted (chapter 5.2.7).

The interviews followed the same line as the survey. First the entrepreneurs explained the history and the status of the start-up. Then questions about measurement and information needs were asked.

5.2.1 Case one: A small marketing company expanding their business to other areas

A small marketing company in one of the largest cities in Finland, was trying to expand their business by copying a successful business model from another company located in a smaller city. The business model had a customer segment, that would want to advertise and another customer segment, that provided the location for the ads. They had all technological and hardware problems solved by using partners. By the time I became aware of the start-up, they had estimated that the new business would start generate revenue in a matters of months. This was the first start-up of one of the entrepreneurs, the second for others.

One of the partners had became the salesperson, trying to accumulate customers, who would want to advertise and customers, who would provide space for the ads. In the beginning the advertisers showed some interest, but no-one was interested in providing places for the ads. What they measured, was the number of sales visits, whether the customer said yes, no or maybe at the end of those visits, and the amount of money coming in and going out.

The problem and the solution were validated in another market, in a small town
where there was no competition. As the customers, who provide the ad space, are fundamental part of the product, they had no minimum viable product. They had done market research and were aware of the competition. They had thought they could enter the market as a low cost entrant. They weren’t interested in learning, what it would take for the ad space providers to give ad space. They just offered free ad time and if that wasn’t enough, then part of the profit from the advertises.

A few months later the start-up ran out of money and stopped. In post mortem interview, they admitted that the market was different because many of the potential customers were already using similar solutions from other companies. The most promising customers ended up preferring a company, that had been in the business for a long time, to a new company. Also they claimed that they entered the business only for the possibility of easy earnings and weren’t therefore interested in pivoting. They were also reluctant to test any of the hypotheses, because they were afraid someone might steal the business idea from them.

In analysis, the company was in empathy state. They hadn’t verified the problem, solution and market hypotheses, as they had a market situation different from the more successful start-up’s business model. As they had no place where to put the ads, they had no product. They were not interested in finding out what it would take to get people to give them that ad space. Instead they were focusing on trying to get space providers and advertisers on their own terms.

5.2.2 Case two: Startup attempts to build a communication device

Both entrepreneurs in this start-up have history of running a successful software business. The project was started before Lean Startup existed. The start-up was funded mostly by grants and one third with own money.

Based on their own experiences and hobbies, the entrepreneurs in the start-up had a need of a underwater communication device. The entrepreneurs studied the market and found several solutions, none of which satisfied their underwater communication needs. They listed a few top problems they would like to see solved. They talked with diving professionals to find an existing solution and, at the same time, tried to assess the interest in such product. They learned that no such product existed.

They studied the market and decided to make a communication device rather than dive computer, because dive computers are very competitive market and has greater risk of lawsuit, in case of battery discharge. They didn’t do a formal market re-
search, but they claim the feedback was very positive, based on their gut feeling of discussions with a a few dozen experienced divers and diving center employees, guides and trainers. There was also interest in piloting the product amongst the potential customers. They had an idea of a price range, which they iterated it a couple of times.

They applied for a grant and received one from government (ELY) to do a design thesis on the keyboard design, which they had identified as the biggest product risk. The keyboard worked well, but soon after that, they hit another technological problem, which was more difficult to solve. It was decided that the product is possible in theory, but not with current information. The project was put on hold for a year when, by chance, they got help from external experts in the field, who did research and found that a problem similar to theirs, had been solved in two cases in the world. They applied again for a grant and received one. After more research, they got their hands on piece of hardware to study and ended up with a physical proof of concept, that the problem is solvable. To protect their solution, they applied for patent. The project had thus far cost approximately 35 000 euros and to make it into a real product, they estimated would cost a couple of hundred thousand more. They decided that building the device would not so easy, that they could just buy the components and create a Kickstarter project, so they wanted a partner with whom to finish the product. No such partner has been found.

In analysis, the start-up is still in empathy state. They have in their opinion a working technology demo, but they have not been able to prove that the problem is real enough for the venture capitalists to invest money. As they currently lack the funds to convert the demo to working product, they are in a kind of limbo. It has been often argued that business angels and vendor capitalist do not wish to invest in someone’s learning process, so start-ups should be able to prove that there is a problem and solution fit as well as product market fit before trying to find funding.

I asked one of the potential partners in-officially about their underwater communication device needs. They had a need, but are not willing to start from the scratch. If this is true, it suggests that, either they are not aware of the demo, they are not as trusting in regards of the demo or the entrepreneurs have failed in trying to present their case. This suggests a different analysis, that the potential partner is not convinced, that the solution is technologically ready. The start-up is still in Empathy stage.
5.2.3 Case three: An entrepreneur creates a start-up to build a software to solve own problems

The entrepreneur in this is a dentist, who has years of experience in running his own dental clinic. The start-up is funded with own money. The entrepreneur is not familiar with Lean Startup methodologies.

The original problem was that the entrepreneur wanted to cut excessive costs regarding inventory management. He calculated, that one nurse spent two workdays a month updating inventory and making orders, and still they ended up having problems of running out of supplies in time of need, and having excess supplies that would expire before they could be used. He studied the market and found out, that the existing solutions were implemented from the logistics point of view and that they would not be a solution to the problem.

The solution was to build a inventory management software, that would be extremely easy to use. The design was done from the user point of view. If the inventory changes were difficult or cumbersome to make, no-one would do them, and the product would be rendered useless. First the entrepreneur tried to figure out how the users would like to use the product. The first idea was to identify products with bar code reader. They made a small test and found the approach useless. Next idea was to use tablet computers and focus on optimizing the search function. During the development the nurses, working in the entrepreneur’s clinic, have been involved in the process. The development has been done in small iterations.

The product has evolved to a kind of JIT implementation, where the amount of local storage is kept small, by ordering material only when they are about to run out. The software helps user create ready orders to different suppliers, based on what is currently in the storage.

The first product was sold in 2014 November and development work continued with that customer. They monitor how often the product is used. If they detect that customer is not using the product, they will contact them and ask if they have problems with the software. In 2015 they got new customers and are planning to automate the database initialization process, which currently is tedious and manual, and needs to be done by trained staff.

The entrepreneur is thinking of monitoring how the material moves. He thinks this information could in the future be useful for suppliers. Based on the information, suppliers could, for example, make offers to a customer. The entrepreneur is also
monitoring the coverage of how many dental clinics, out of all dental clinics in the country, are his customers. When this percentage exceeds a certain number, he thinks the start-up will be ready for next big change.

The product has a landing page which is monitored with google analytics, but the data is not followed constantly.

In analysis the start-up is in stickiness stage. They have a MVP and customers, and are paying attention to retention, while improving the product and trying to get new customers. To get the viral engine started they are removing technical obstacles, that are in the way. It seems that they are following Lean Startup methodologies, even if they are unaware of them. The entrepreneur explained this, that since it’s his own money, he has to do it this way. If he had no problem with the funding, the start-up would look very different, more professional company-like.

### 5.2.4 Case four: A start-up based on value of trust

This start-up is the only one of the interviews that isn’t a company. An entrepreneur, with his own consulting company, has a strong sense of duty towards his customers and friends. He likes to help the customer in any way he can and occasionally has to turn down offers, because they were out his areas of expertise or interest. At the same time he had professional friends and comrades, with their own consulting companies.

He created a network with those comrades, he trusted professionally and who had knowledge, especially in the fields where he lacked. Initially, the network was to bounce ideas, ask professional opinions and suggestions, and in the same spirit return the favour. In practise they discovered, that they could also offer their individual customers wider range of services through the coalition.

They measure how much work is referred this way but are currently focusing on setting the price right. They are keen on getting new recruits, but prefer ones that already have customer contacts.

In analysis, they are in empathy stage. They have a MVP but the customers come and go and they haven’t been able to validate that the problem is something that could one day turn into profitable business.
5.2.5 Case five: A training software for music lovers

The entrepreneurs in this start-up are familiar with Lean Startup methodologies and considers the company a Lean Startup. They are funded by business angels and are not self-financed. The start-up was interviewed twice, with six months in between.

The idea for this start-up came from own needs. When a musician is performing, it’s hard to analyse own performance, without it affecting the performance. Other problem was, that performing with a group is always easier, but it’s not possible, if the group has no common place or time to practise. The entrepreneurs thought of creating software to help separate analysis from the performance.

They interviewed people in order to find out, whether the problem is real for others. They checked the market for similar solutions and found none. They currently have a working product and have sold licenses. According to customer data, the product is used continuously, and the entrepreneurs know how the product is being used. They know how their customers are segmented and they also have detected other customer segments, that appear to be interested.

They have analysed different advertising channels and measure their impact on new users. They can see correlation between events and changes in user acquisition rates, but cannot be sure about the causality. They also measure activation and abandonment measures rates, and study them in cohorts. They are able to measure some referrals (those made from the application). They have tens of thousands users (including trial users) and they have tried different licensing models in order to get the revenue up.

They monitor application installation dates and dates when the application is used. They are currently unable find out why they are losing customers, because registration is optional and they do not wish to spam users. Still they consider this a very important question.

They gather software statistics from Github and user statistics from Mixpanel. They have developed their own application for analysing some of the data, because Mixpanel is expensive when user amount exceeds thousands. They get customer satisfaction information from reviews in App Store, direct feedback and email. They had a Web-form but got no feedback from there. Their goal is to understand the customer needs and therefore ultimately increase profits.

The start-up gathers a lot of different kind of data and they make decisions based on data in different ways. In some cases they have preset limits, which are acted upon,
when they are exceeded. In some cases they look at data, make assumptions, think about how to improve, and put the tasks in the backlog. They base their decision to implementing new features on cost of feature, customer behaviour and customer feedback.

Their One-Metric-That-Matters is revenue, although they admit that it’s difficult to find causality between different actions and the revenue. They have implemented the application on different platforms and are focusing the development on the most profitable ones.

In analysis this start-up is in revenue state. Their focus should be in converting customers into revenue. They recently changed how their trial version can be used in order to improve activation rate.

In the follow-up interview, the situation had not changed much. The new revenue model didn’t improve the situation much and they are still not self-sufficient. They have studied different customer segments and believe they have found a new one. They are planning to enter that market by introducing a new feature, but are currently making ends meet by doing consulting work, unrelated to their product. They are also considering giving up on their principles of not spamming the registered users, in order to find out why the abandonment rate is high.

5.2.6 Case six: Startup tries to expand abroad

The entrepreneur in this start-up is considered technically competent. The person interviewed was an employee.

The idea for this start-up came from recognizing a potential problem. The market had some existing solutions, but this start-up aimed at doing customization faster than the competition. They have a MVP that is being developed in co-operation with the customers. They have domestic customer and are thinking of expanding to international markets. They are working full time doing their customizations and are not able to develop their product back-end or optimize their tasks at desired rate. They would like to hire more developers, but need more customers before they can do that. They are funded with own money and with the money coming from the customizing tasks.

In analysis, the start-up is in stickiness stage and they are not ready for virality stage. As each customer bring more work, they need to automate some of their processes to avoid hiring excessive amounts of developers.
What was interesting in this case, was that when asked about metrics, the interviewee was not certain what metrics and measures were gathered and used. This created slight worry and mistrust in the employee toward the company’s vision and the management.

5.2.7 Case seven: Two start-ups with a feasible business model, one abandons and one pivots

These cases are based on interview of a consult to the other start-up and an entrepreneur in the other.

The first start-up had an idea of a product and had created a minimum viable product, that was being used by customers. The MVP had its problems and wasn’t as pretty as it could be, but was appraised by their customers. The entrepreneurs had narrowed down a price for the product and they calculated how many customers they could have with their product. They noticed, that nationally their product would not be profitable, so they considered making their product for international market. They came to a conclusion, that this required a lot of work they were not very keen on doing, so they abandoned the product.

The second start-up had similarly created a business model, that would be profitable. They had a customer, which was a leader in the world in its field. Basing their knowledge on what they had learned from the business segment, they started to speculate how many customers they could sell their product to. They discovered that with the current concept, they would have about twelve potential customers in the world. They pivoted their product to make it more simple and general and doubled their revenue. At the point of the decision to pivot, they had had their one metric set on the margin of revenue.

What this interview suggested, is that start-ups may be abandoned for more interesting start-up idea or pivoted for more profitable model. It’s not uncommon for companies that do research to have more ideas than what they can execute and some have given their business ideas to other interested parties free of charge.

5.3 Startup metrics

The third research question of this thesis is "How are the metrics in Lean Startup being used?". The question is a bit ambiguous when considering the RQ4. To find
out whether a start-up is a Lean Startup, one aspect would be to estimate how they use measures and metrics. If only those start-ups were considered that use the same metrics and measures as a Lean Startup should use, this question doesn’t differ from the RQ4 in any way. The question is more meaningful if reworded as "How are the metrics in start-ups being used?".

5.3.1 Lean Startup vs traditional entrepreneurs

First aspect that came across strongly, with entrepreneurs following Lean Startup principles, was their love of measures. They had questions that they wanted answered, and were gathering data to find those answers. The funnel metrics were mandatory but some of them also gathered all kinds of data, in the off chance that they might discovers something they didn’t know to look for.

Meanwhile the more traditional entrepreneurs tended to rely on their experience, knowledge and intuition. Some of them more successfully than others. Where the inexperienced entrepreneur in interview in chapter 5.2.1 was at loss at what to do, the more experienced entrepreneur in interview in chapter 5.2.3 seemed to be making all the right decisions.

5.3.2 Revenue

The revenue stage, where turning the product into constant revenue in order to create a sustaining business model, is rather late in the process. This combined with the fact that the entrepreneur can’t truly trust the word of potential customers when asked in empathy stage whether they would purchase the product, may result in difficulties in turning users into profit. Lean Startup discourages excessive spending before the business model has been verified so the amount of funding used until this stage is inherently low, as there usually is little or no income and the nest egg has to last long enough. The start-up interviewed in chapter 5.2.5 has gone through several funding periods, have met their targets which validate their funding, but still struggle to find enough paying customers.

5.4 Usage of Lean Startup Metrics

The fourth research question in this thesis is "How should the metrics in Lean Startup be used?"."
5.4.1 Verification versus intuition

Lean Startup stands for two main things, making decisions based on validated learning and removing waste.

Using measures and metrics in decision making is nothing new. Experienced and skilled entrepreneurs do this naturally. What Lean Startup offers in this perspective, is a model for new entrepreneurs to compensate their lack experience with systematic routines and channel their enthusiasm more productively. For the experienced entrepreneurs, Lean Startup offers methods with which to validate their insights, if they so desire, and to expand and optimize their business model through exploratory measuring.

The other aspect, eliminating waste, is also something that numerous entrepreneurs tend to do naturally, especially when money is scarce. In case where a hypothesis is considered self-evident by the entrepreneur, laborious verification could be considered waste. To differentiate a good intuitive gut feel and a blatant attempt of self-deception, requires objective investigation, which can be achieved in many ways. One of which is to execute a test and measure the result. This suggests a dilemma with Lean Startup. It attempts to verify the gut feeling of an entrepreneur, to minimize the waste of making bad decision based on incorrect assumptions. What if following Lean Startup principles is doing, is creating waste of verifying things entrepreneur intuitively already knew to be true. Of course, in the long run, such verifications may prove to be helpful in convincing others about the validity of the business model.

Much of the Lean Startup is about gathering measures. Metrics are gathered and analysed to validate a hypothesis. However, there may be reasons to do things differently. In some of the examples in Lean Startup literature, the authors revisit failed cases with hindsight, that they should have done market surveys and never implemented the feature. Let's assume a start-up has a choice between implementing a Halloween related feature starting right now, or doing a market study, that would risk the meeting the deadline for the implementation. If they choose to do market study early to leave room for implementation, it could be made during a time, when no-one is interested in Halloween. The study could show that there is no need for the feature, when in reality it could have been a killer feature. Everything can not be verified within the given resources, and the need for gut feeling of an entrepreneur can never be totally eliminated.
Build-measure-learn cycle can help entrepreneur get confidence in one’s intuition, or teach to not trust them, depending on the situation. If the gut feeling of the entrepreneur intuition has been proven right time after time, it’s possible intuition is used more often, instead of formally testing all hypotheses. For vendor capitalists and business angels having proven the gut feeling to be correct is, of course, very convincing. It is then more likely to result in receiving funding for the start-up when, for example, scaling.

One of the most painful situations have been when start-up has had no clue how get answers. If they run a small ad campaign for their product, which results in no hits on their landing page, where could the problem be? More money to advertisement? Improve the advertisement text? Use different channels? Refine the problem hypothesis? Refine the solution hypothesis? Find new channels? The answer in Lean Startup literature usually is "whatever it takes" and that ultimately means going back and asking the users and customers directly.

5.4.2 Metrics as means of communication

The metrics, when measured properly, analysed correctly, presented intelligibly and being auditable, is a reliable way to communicate. At best, for the employees the metrics tell what the start-up is focusing on and why, how the start-up is doing and what the vision is.

For the venture capitalists and business angels, the measurement data provides solid information, upon which to make the decision whether to investigate.

5.4.3 Ease of measurement

For an entrepreneur to start measuring and analysing measurements, he needs to be motivated to measure and ingenuity to come up with ways to measure that aren’t too cumbersome.

Lean Startup requires that entrepreneurs use metrics that are useful and understandable. When the the entrepreneurs understands the benefit of using certain metrics, they may be more motivated to use them. Unfortunately the entrepreneurs ego may be in the way of measuring. Because he’s convinced he knows enough, he sees no reason for measuring.

The history of software measurements have demonstrated that, over time, gather-
ing useful quantitative measures will be automated and integrated into everyday tools. With Lean Startup some of measurement gathering and analysing services are available online. There are libraries that can be used to gather solution specific measurements and to present them in different forms for analysis.

Implementing complex measurement gathering and analysing systems is feasible, when the number of users is big. If there are just few known users, simple asking might be more feasible or analysing the results by hand.

5.4.4 Funnel metrics

Some of the Lean Startup tools, like the funnel metrics, provide easy ways to produce solid information on which to make decision. If these tools can be integrated into use with minimal effort there is no reason why they should not be incorporated.

For the entrepreneur the funnel metrics provide a view inside her business model to find out places that need improvement. For business angels and venture capitalists it provides solid data to make decisions, whether to invest.

5.4.5 User behaviour

How the users use the product, is also important in Lean Startup. This is done by mapping different interesting actions user might take and then storing the information when a user attempts to do such an act. Analysing the reasons behind different funnel metrics, require knowledge of the user behaviour. The user behaviour can also shed some light into possible problem areas in service or application. As suggested in the interview in chapter 5.2.3, the customer behaviour can be valuable information to third parties.

5.4.6 Exploratory measurement

The lean Startup entrepreneurs measure as much as they can with ease. All the data, if analysed will add to their understanding of the domain. Finding patterns in the data may occasionally give entrepreneurs ideas to identify other unsatisfied needs, which would be basis for pivots or a solution for a completely different problem. In this, the measurement is basis for learning and understanding, but it depends on the characteristics of the particular entrepreneur, whether she will identify the patterns and understand their meaning.
5.4.7 Revenue

The time, how long a start-up has, is dictated by money. Money is the final metric, and understandably the most important in the end. It will dictate whether the start-up lives to see another day or dies off. But to have the bottom line as the metric, to which to react could be fatal, as it may already be too late to react, when the bottom line goes down. To get to control revenue, smaller steps need to be taken. Revenue should be examined in cohorts and as ratios. Using the funnel metrics in product/market fit will tell the entrepreneur which parts require improvement the most. Improving those will eventually result in increased revenue.

5.4.8 Scaling needs

If all goes well, the start-up will see the need for scaling into a large business. The scaling stage is costly and usually requires external funding. The scaling also means, that the start-up is becoming more company like, and companies can no longer be run by enthusiastic entrepreneurs as small start-ups can. Implementing measuring system at this point would be problematic as everything would have to be done quickly and from scratch. If the start-up attributes have thus far been measured rigorously with little or no overhead, those processes are already in place and automated. Also it’s easier to find funding for the start-up that can prove their business model with actionable, accessible and auditable metrics.

6 Analysis

In software development the resources, processes and products have been constantly measured. When a new idea is innovated, metrics are attached to it. These metrics are used to monitor important aspects of the innovation. First the metrics set a base line and the model, which is based on the innovation, is used to improve the innovation. The improved innovation is measured to see if it has indeed been an improvement.

There is some gap between the academic world and the business world when it comes to measurement. The motivation to measure things is often the result of something else than pure need. For example if a client requires a company to have a certain certificate, the company must be able to measure certain metrics they probably wouldn’t measure otherwise. Where the academic world needs to measure things to
validate certain claims, the business world doesn’t have this requirement. Whether they are doing the right things and making the right decisions, are not dependant on whether they measure certain things.

The Lean Startup makes measurement easier and more understandable. Because one of the purposes of Lean Startup is to minimize waste, complex and laborious measurement systems are not an option. As the metrics need to make sense, they must have a meaning and must therefore be understandable. The metrics suggested by literature, give big help to certain types of problems. In Lean Startup the metrics and measurement is seen as fundamental part of entrepreneurship.

How the entrepreneurs use metrics, varies according to the whether they claim to be Lean Startup entrepreneurs. The Lean Startup entrepreneurs believe in measuring. They keenly try to figure out different aspects of the customers and the market. They are measuring several things and focus on things they consider important. In short, they are trying to learn and understand the market and are doing that by validating their assumptions with measurement. The traditional entrepreneurs do the same, but instead of formally measuring and analysing data they make assumptions based on how they experience the situation and how they think the causality works.

The metrics in Lean Startup are used to learn and focus the work in the start-up. Through the measurement feedback loop an entrepreneurs are faced with their errors or valid assumptions, which in both cases contributes to their learning. The selection of important metrics communicate to the people in the start-up, what is thought to be important in the start-up. By defining the metrics and assessing the limits and the actions to be taken beforehand, the entrepreneur gets objective help to steer the start-up.

No matter how much the Lean Startup literature suggests metrics, measures, lines in the sand and actions to be taken, at the end of the day it is the entrepreneur who selects those metrics, defines the actions, sets the limits. The start-up is as good as the entrepreneur making the judgements, so the most important thing that constant measuring brings to the entrepreneur, is the understanding.

6.1 Limitations

Due to the failure to attract responders to the survey sets limit to what conclusions can be made from the data available. From large sampling size it would have been possible to find, how does measurement predict for example start-up success or if
the experience of the entrepreneur has more weight. This kind of conclusions cannot be made from this study.

The main source of new information in this study is the interviews and the insights had from that. The interviews tell different stories how things have happened in real life. This in itself is not enough to validate any conclusions, but only give indications to what might be possible. These kinds of indications are good starting points for further studying.

6.2 Conclusions

1. The goals of measurement in software (RQ1) are described in sections 2 and 4.1.
2. The Lean Startup metrics and measures (RQ2) are described in sections 2.7 and 4.2.
3. How the metrics are used in today’s start-ups (RQ3) are described in sections 5.3, 5.
4. How the metrics and measurements should be used according to Lean Startup (RQ4) is described in sections 2.7 and 5.4.

6.3 Summary of Contributions

1. Narrated through the history of software measurement from the point of view of Lean Startup (chapters 2.1, 2.2, 2.3, 2.4, 2.5 and 2.6.
2. Describes metrics used in Lean Startups (chapters 2.7.2, 2.7.3, 2.7.4, 2.7.9 and 4.2).
3. Takes a look into metrics used in today’s start-ups (chapters 5.3, 5.1 and 5.2).
4. Describes different uses for metrics in Lean Startup (chapter 5.4).

6.4 Future Research

This chapter describes several questions that have risen as a result of this study, but fall outside its scope.

1. What are the entrepreneur attributes that affect most in start-up failure? Is it the experience, ingenuity, relentlessness or other attribute commonly assigned
to successful entrepreneurs? Or is it the fact that they are using Lean Startup methodologies? Would the ones, whose start-up would fail, quit faster if using Lean Startup?

2. Lean Startup takes a very conservative look into what could be done. If the metrics don't approve a decision, an entrepreneurs has to decide whether to trust the metrics. Are there any examples of hit products that would not have made it, if they had used Lean Startup methodologies?

3. When does the market become so saturated with Lean Startups, that some other advantage needs to be found? What will those advantages be? Do those differ from the advantages before Lean Startup?

4. Are Lean Startups more successful than other start-ups? A more meaningful questionnaire with bigger sampling could be made. The correlation and causality of use of metrics, between experienced entrepreneurs and lean start-up entrepreneurs, could be measured. Which variables best explain the success of a start-up?

7 Summary

This thesis takes a look into history and describes the history of software measurement and Lean Startup. The software development history goes through several phases. First is the definition of software industry, where a new industry field is attempted to understand by defining its resources. Next the processes how the resources are turned into products are defined and measured. The goal of defining and improving the resources and processes is to get a better product, which is measured more intensively with definition of software quality and increased customer and user involvement.

This study explains the principles of Lean Startup, such as build-measure-cycle, that is in the heart of the continuous learning in Lean Startup. It explains what kind of measurement is done, what kind of metrics are used in Lean Startup and how they are selected. The study explains MVP and split testing as tools to measure with. Some noteworthy thoughts are expressed from critique gathered from certain blogs. The answer to RQ1 describes how the resources, processes and products are measured over the years and how the measurement of the business model has increased significantly in importance after customer development. Also the role of user in software development and software measurement is discussed under this chapter.
The answer to RQ2 explains what different metrics, mainly funnel metrics, according to literature, should be used with Lean Startup in different stages.

The answer to RQ3 shows the result from survey and suggest how the traditional start-ups differ in attitude and practises, from Lean Startups in measurement. It also addresses the importance of revenue in start-up.

The answer to RQ4 suggests different reasons for using measurement in start-ups, such as communication. It describes benefits from measuring user behaviour and exploratory measurement, and it tells how revenue should be measured. The relationship between validation through measurement and intuition from the point of view of elimination of waste, is also addressed.

The research design for the study is also described. The research consists of the literature study and a survey, with questionnaire and interviews. The survey results and the interviews are shown and summarised.

**References**


Boeh81 Barry Boehm *Software engineering economics.* 1981.


Hami86 Margaret H. Hamilton *Zero-defect software: The elusive goal: It is theoretically possible but difficult to achieve; logic and interface errors are most common, but errors in user intent may also occur.* Spectrum, IEEE 23.3 : 47-53, 1986.


Hump00  Watts S Humphrey, *The personal software process (PSP)*. 2000.


9 Polte. Hands on seed investor, http://reaktor.vc/
## Appendix 1. List of terms

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measures and metrics</td>
<td>Metrics and measurements have overlapping meanings. Metric is more abstract standard of measurement, one that evaluates a complex process or system, while measure is more concrete. Measures are bases for metrics. For example the Lean Startup acquisition metric can be found out in one application by measuring rate of registrations in another by rate of downloads.</td>
</tr>
<tr>
<td>Software measurements</td>
<td>The software measurement targets can be divided into three categories: software resources, software processes and software products. The first measures the attributes of the entities, which are used to create software. Entities such as developers, compilers, computer languages or tools. The second metrics measure the attributes of the way of working or how, from the resources, a product is created. The last measures the attributes of the product, that is built.</td>
</tr>
<tr>
<td>Lean Startup</td>
<td>There have been discussions on the correct way to spell the term Lean Startup. In this thesis the format Lean Startup is chosen because the source material mostly use startup instead of start-up. Also the capital initials are chosen to distinguish between the concept Lean Startup form start-ups that have the attribute of being lean. In this thesis the term Lean Startup is an umbrella term for the methodologies and for the start-ups which claim to use Lean Startup methodologies. When the term makes no difference whether it is a traditional start-up or Lean Startup, the term start-up is used.</td>
</tr>
<tr>
<td>Users and customers</td>
<td>Users and customers can have some overlap. In this thesis users are the ones who use the product and customers are the ones that provide revenue. These groups can be exactly or partially the same or completely exclusive. Facebook has users who create content and interact with each other. The customers are, for example, the advertisers who pay to for advertisement.</td>
</tr>
</tbody>
</table>