Ville Ravaska

THE ESSENCE OF SOFTWARE STARTUP : AN EMPIRICAL STUDY ON THE APPLICATION OF ESSENCE FRAMEWORK

UNIVERSITY OF JYVÄSKYLÄ
FACULTY OF INFORMATION TECHNOLOGY
2020

ABSTRACT

Ravaska, Ville
Name of the publication: The Essence of Software Startup : an Empirical Study on the Application of Essence Framework
Jyväskylä: University of Jyväskylä, 2020, 84 pp.
Information Systems, Master's Thesis
Supervisors: Pekka, Abrahamsson; Kai-Kristian Kemell

Software startups are still a scarcely studied subject even though success stories of the companies like Facebook and Twitter have boosted the popularity of new software startup companies. The impact of successful startups for the economy is massive by creation of new jobs and wealth. The Essence framework is relatively new approach for modelling software development in the companies. It has not been extensively tested in the unique case of software startups. Software startup practices have been studied and acknowledged in the academia but further research is still needed to comprehensively understand how these companies work and conduct their work.

This thesis studied software startup practices regarding software development and other vital activities performed by these companies. The Essence framework was also introduced to software startups' working practices. The basis of the thesis was the list of software startup practices derived from previous studies that was categorized following the alphas of the Essence framework. The empirical study was conducted as a multiple case study with thirteen different cases.

Three new alphas (funding, marketing and business model) were suggested to the Essence in order to revise the framework to support software startups whole endeavour. Thirteen additional practices were also found from the empirical material. The Essence was found to support software startups software development efforts but lacking the business aspects that cannot be separated from the software development in small new companies working with limited resources.

Keywords: startup, software startup, essence, practice, software development practice, software startup practice

FIGURES

TABLES

TABLE OF CONTENTS

1 INTRODUCTION

Everyday many new software startups are born and success stories like Twitter and Facebook only boost their popularity. However other side of these success stories is that most of the startups fail during their first two years (Crowne, 2002). Increasing the percentage of successful startups could have huge positive impact in the economy by creating new jobs and wealth (Kane, 2010).

Software startup practices are a scarcely studied subject while general software development methods and practices have been subject to extensive studies. Majority of these studies concentrate on established and mature companies rather than startups working with limited resources under immense pressure. (Giardino et al., 2016.)

Software Engineering Method and Theory (SEMAT) community has created a framework called the Essence. It described a common ground for all software development endeavours (Submitters, 2012). This framework was selected as a foundation of this thesis because of its two usages, its extensibility and its comprehensive approach to software endeavour management (Jacobson et al., 2012). These two usages allow the kernel to be used to describe software startup practices from numerous different sources under one model.

1.1 Motivation

Klotins et al. (2015) and Paternoster et al. (2014) have argued that mainstream software research lacks studies focusing on startups. For example Klotins et al. (2015) indicated that only 28 of 62 knowledge areas of IEEE Computer Society's Guide to the Software Engineering Body of Knowledge were covered in studies focusing on startups in 2015 (Bourque & Fairley, 2014).

Unterkalmsteiner et al. (2016) have expressed in their software startup research agenda that research on this topic already provides snapshots of startups' software development practices but more comprehensive studies are needed. Startups have been subject to studies of numerous fields, for example business,

software development methods and user experience. Still academia has not reached unanimous conclusions for any major topics related to startups starting from the definition of startup. More studies of startup related topics are needed to provide comprehensive empirical knowledge of this phenomenon. (Unterkalmsteiner et al., 2016.)

Recent software startup studies have been centering on the core knowledge of software development and software engineering challenges in startups (Unterkalmsteiner et al., 2016). A network of researchers interested in startup phenomenon created a mind map of different research areas regarding startups, which need to be studied more comprehensively, in December 2015. These research tracks were grouped into six clusters (Unterkalmsteiner et al., 2016.):

- Supporting Startup Engineering Activities
- Startup Evolution Models and Patterns
- Human Aspects in Software Startups
- Applying Startup Concepts in Non-Startup Environments
- Startup Ecosystems and Innovation Hubs
- Methodologies and Theories for Startup Research

There are also other research areas that authors find interesting and may be added to research agenda in the future. For example Marketing and Business and Economic Development are directions that will most likely be relevant for startups' performance. Other research areas might also rise while the knowledge of the phenomenon increases. (Unterkalmsteiner et al., 2016.)

1.2 Research Problem

Purpose of this thesis is to study essential universal practices of software startup companies. The practices are discovered through empirical research. From the found practices those that are present and requisite for every software startup form the essence of software startups. The results will be presented following SEMAT Essence kernel framework. The Essence kernel so far presents the essence of software development but it does not focus on uniqueness of startups thus revising it considering startups' universal aspects will provide valuable practical tool for startups.

Research question of this thesis is: *How software startups' working practices fit under the alphas of the Essence framework?* The main question is divided into sub-questions:

1. How to define a software startup?
2. What practices are universal for all software startups?

3. What aspects of the essence of software development are not universal for all software startups?

Sub-question 1 is answered through literature review while sub-questions 2 and 3 are answered through literature review and empirical findings of this thesis study.

1.3 Scope of the Research

Since there are massive amount of startups found around the globe every year this thesis was focused only to those that produce software or whose product includes software as a major part alongside the physical product or a service. This thesis was also focused to early stage startups working in Finland and Nordic countries. This was chosen since most of the recent software startup studies focus on startups that have already managed to succeed in the market. Focusing on small early stage software startups gives a fresh aspect to startup research.

Software development is a widely studied subject in the academia and immense amount of different theories, frameworks and models have been developed to describe different software development practices and methods. In this thesis the framework on top of which the study is based was chosen to be the Essence. This framework was chosen due to its ability to describe different practices under one comprehensive framework.

Data for this study was collected through thematic interviews and later analysed following qualitative methods. These interviews were expanded by interviews obtained from another study of software startups practices that were analysed together with the original interviews. More precisely thematic synthesis was used to analyse the data. Study included thirteen early stage software startups as studied cases. Eleven of them were active startups and two were recently discontinued.

1.4 Structure of the Thesis

This thesis has eight chapters: Introduction, Software Startup, Essence Framework, Theoretical Framework, Research Design and Methodology, Empirical Results, Discussion and Conclusion.

Chapter 1 presents an introduction to the topic and reasons for this study. Research problem and Scope of the study are also discussed in this chapter.

Chapter 2 discusses the definition of software startup used in this thesis. This chapter also includes software startup practices from the literature.

Chapter 3 introduces the Essence framework and discusses its usage, criticism and other competing frameworks.

Chapter 4 presents the theoretical framework that is used as bases of the empirical study of this thesis. It is based on the Essence framework presented in the chapter 3 and software startup practices presented in the chapter 2.

Chapter 5 introduces empirical design and methodology of this thesis and presents how empirical study was conducted.

Chapter 6 presents empirical findings and summarizes the key findings as primary empirical conclusions (PEC).

Chapter 7 discusses about the PECs presented in the previous chapter. In this chapter the findings are compared with literature.

Chapter 8 concludes this thesis. In this chapter the research question is answered, limitations of the study are discussed and potential future research aspects are presented.

2 SOFTWARE STARTUP AND STARTUP PRACTICES

This section concentrates on what is known of software startups in the literature. In the first subsection software startup in this context is defined. In the second subsection startup life cycles are introduced. In the third subsection the concept of practice is defined and in the last subsection and its subsections the software startup practices are presented.

In the literature defining a startup has not been simple task and several different definitions can be found. These definitions focus on totally different characteristics that make company a startup (Paternoster et al., 2014). Especially in the grey literature and in informal situations inside startup community the definition of startup varies a lot. Some informal definitions focus on number of employees, some focus on length of a company's history while some focus on set of characteristics of a company (Robehmed, 2013).

Most of the startups fail during the first years of their existence. Quite often failure comes from lacking ability to pivot when situation changes. It has been widely acknowledged that startups, whose one main feature is working with uncertainty and rapidly changing new markets, need to be agile in their endeavours. Blank & Dorf (2012) developed a Lean Startup method to address this issue and it has been widely accepted in startup community. Of course most of the startups do not follow any textbook methods rigorously but combine and tailor methods to suit their endeavour. (Paternoster et al., 2014.)

2.1 Definition of Software Startup

Software startups are new companies working under pressure and limited resources creating new innovations. Innovations can be for example new products or new way to do something existent or bringing some existing product to a new market from other context. Startup is a temporary phase in company's life cycle that ends when company either fails or evolves into mature company

or entrepreneurs perform an exit by sale. (Wennberg et al., 2010; DeTienne et al., 2012.)

Ries (2011) defines startup as a human institution designed to develop a new product or service under extreme uncertainty. Blank & Dorf (2012) use similar definition that it is a temporary organization with no or limited operating history creating high-tech innovative products. These definitions differentiate startups from established companies that have more resources and already a mature market. Blank (2007) identifies startups from small businesses by their intention to grow and develop a scalable business model which differs from other small enterprises.

A looped evolution path is proposed Ries' *The Lean Startup* (2011). It consists of the steps of idea, build, product, measure, data, and learn. This model indicates the non-linear progress that is typical for software startups by defining a restart, pivoting, and the loop from learning to a refined idea. The Lean Startup has been widely acknowledged and used by academia and startups. Research evidence suggests that development activities in startups need to be tailored to fit unique situations of these companies in order to support flexibility and reactiveness of development workflow. (Giardino, C., Unterkalmsteiner, M., Paternoster, N., Gorschek, T., & Abrahamsson, P. 2014.)

Startups usually use different metrics to measure their progress than mature companies due to uniqueness of the situation, for example lack of resources demands metrics for how long can development be done with the current funding. Financial progress metrics that are used in mature companies such as income statement, balance sheet and cash flow are not convenient for startups. Usually they are adopted because they are traditional metrics. In startups these metrics are not tracking progress against a goal to find a repeatable and scalable business model.

Valuable metrics for startups are: cash burn rate and number of months' worth cash left in the bank. Amount of time until the company reaches cash-flow break-even is also widely used metric for startups. Customer acquisition cost is useful metric as well as an estimated per-user cost and any incremental costs when new user is added. Startups should also consider the costs of selling through the channel: payments to app stores, marketplace sites like Amazon.com, or related sites referring customers to you. Useful question to ask in startups are: What are the average selling price? What is total achievable revenue and number of customers a year? And how long or how often will customers spend? (Blank & Dorf, 2012)

Startups also should use non-financial metrics like: Have the customer problem and product features been validated? Does minimum feature set resonate with customers? Who in fact is the customer, and have initial customer-related hypotheses on likes of value proposition, customer segments, and channels been validated through face-to-face customer interaction? (Blank & Dorf, 2012)

In this thesis software startups are defined following the characterization created by Paternoster et al. (2014). This definition was chosen because it's the

most widely used in resent startup research. They extend the startup definition created by Sutton (2000). Sutton's definition focuses on the characteristics: scarce resources, multiple influences, little or no operating history and dynamic technologies and markets. Paternoster et al. (2014) expanded this definition by adding the characteristics: creating innovative products, fast growth, flat organization structures, focusing on one product, third party dependency and time pressure. Therefore not all newly founded companies are startups but those working under high-uncertainty and rapidly evolving. Startups may be found on various domains. Software startup is a company following characteristics created by Paternoster et al. (2014) whose product is software system or combination of software system and physical product or a service.

Software startup characteristics:

- Scarce resources
- Multiple influences
- Little or no operating history
- Dynamic technologies and markets
- Creating innovative products
- Fast growth
- Flat organization structures
- Focusing on one product
- Third party dependency
- Time pressure
- Software as a product or a major part of it

2.2 Startup Life Cycle Stages

There are multiple ways to describe life cycles of startups. These different definitions are presented in this sub-section and the definition used in this thesis is presented in the end of the sub-section.

Crowne (2002) represents startup's life-cycle as a three-plus-one stage model including startup phase, stabilization phase, growth phase and maturity. The other way to present evolutionary stages of an early state startup is to divide them to two: the exploration stage and the validation stage. (Giardino et al., 2016) The stages are further divided into four dimensions: market, product, team and business. These dimensions were originally proposed by MacMillan et al. (1987).

In their study Klotins et al. (2018a) have used startup milestones to point out certain stages of software startups. They have used a model with three phases. Phase one is: *build the first version of the product*. Phase two is: *attract customer interest to the product*. Phase three is: *grow into new markets*. As can be seen these phases are highly connected to the marketing and business growth of a company.

According to Wang et al. (2016) startups life cycle can be divided into different stages. They consider startups' life cycle two-dimensional having learning stages and product development stages. Learning stages are based on Ries (2011) Lean Startup methodology which argues that startups exits to "learn how to build a sustainable business" rather than to "make stuff". Wang et al. (2016) divided the learning process into four stages according to the customer development process. Learning process stages are: *defining or observing a problem*, *evaluating the problem*, *defining a solution*, *and evaluating the solution*. These stages are not linear process but startups go through multiple build-measure-learn loops to find a sustainable business model.

Concurrently with business model learning process startups go through product development process. Product development process stages are: *concept*, *in development*, *working prototype*, *functional product with limited users*, *functional product with high growth*, and *mature product*. According to Blank (2007) to succeed startups need to keep these two processes synchronized and worked simultaneously.

In this thesis both learning and product development process stages (Wang et al., 2016.) are used to define startup life cycles. Startup life cycle stages are presented in a figure bellow. Processes are presented as linear progress in the figure but in practice they are iterated multiple times.

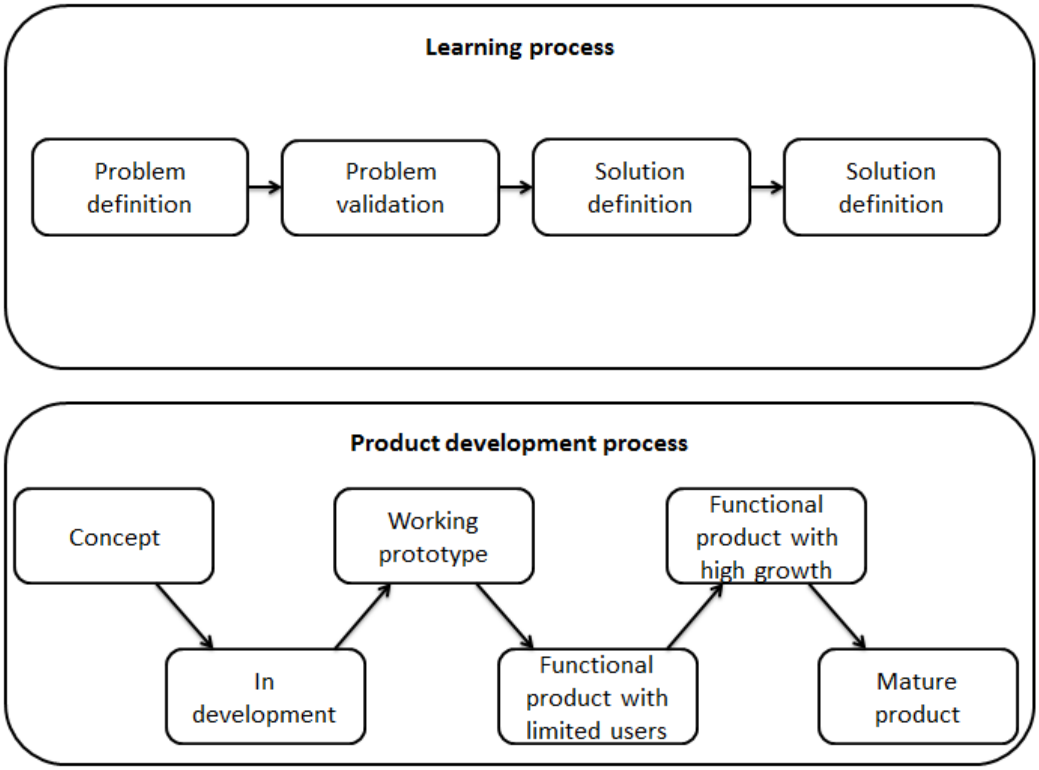

FIGURE 1 Startup Life Cycles (Wang et al., 2016)

2.3 Definition of Practice

In this sub-section the term practice is defined. Definition is widely based on Tolvanen's (1998) definition but also influenced and supported by other definitions of the term practice. Tolvanen was selected as a basis since he is one of the few researchers who have defined term practice in information systems context and this thesis focuses on software startups working practices on their software development endeavours.

Tolvanen (1998) defines information system development as "a change process taken with respect to object systems in a set of environments by a development group using tools and an organized collection of techniques collectively referred to as a method to achieve or maintain some objectives". He describes technique as a set of steps and rules that define how a representation of information system is derived and handled using conceptual structure and related notation. By a tool he means a computer-based application supporting the use of a technique.

Oxford dictionary defines practice as "The actual application or use of an idea, belief, or method, as opposed to theories relating to it or the customary, habitual, or expected procedure or way of doing of something." (Oxford dictionary) This implies that practice is a method or technique brought to actual practical use. Bourdieu's (1973) definition of practices is similar to Oxford Dictionary's: "the recognizable patterned actions in which both individuals and groups engage". Even though these are not information system science specific definitions they are on line with Tolvanen's (1998) definition of methods and techniques. Practice is widely used construct in information system science but not comprehensively defined one. It is usually referred to something practical that is done repetitively to result in something desired goal.

Jacobson et al. (2019) suggest that a set of practices form a method. They point out that huge variety (over 100 000) software development methods can be identified in the world but only a few hundred reusable practices. They treat practice as a mini-method guiding one specific aspect of the team's work. Practice in this context could be for example *create software architecture* which will not alone produce working software but is important part of the process. Therefore a team needs a set of these mini practices. Tolvanen (1998) uses the term technique to describe these components of methods. Relation between methodology, method, practice, technique and tool are illustrated in the figure below.

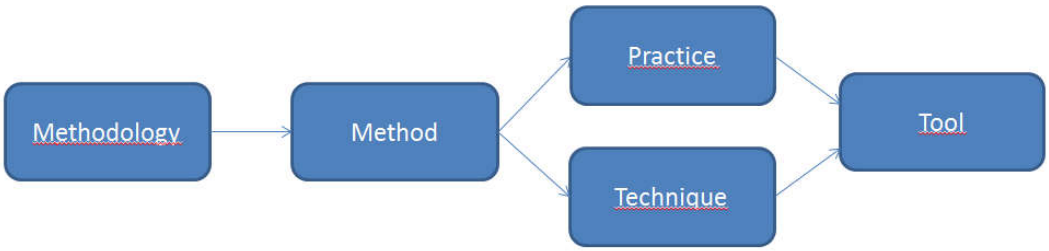

FIGURE 2 Relations between Terms

In this thesis practice is defined as an established way of working used by startup's team. Compounded with Tolvanen (1998) definition practice is a component of a method that is less abstract and more practical than technique and may require use of tools. A method may include a set of practices. In this thesis practices are not limited to software development practices but startups are addressed more comprehensively including organizational practices concerning funding and customer relationships for example. This arises from the startups' characteristics of a small team working with limited resources under high pressure. It is more beneficial to study the entity of startup rather than a small aspect of their endeavour because usually startup's core team needs to be working with software development and business aspects concurrently.

2.4 Software Startup Practices

Practices used by the software startups are described in this sub-section. The basis of the practices utilized in this thesis is the list of practices found by Dande et al. (2014). First other current literature of the subject is introduced. Later in the subsection the list of practices by Dande et al. (2014) is provided with the additional information of what other literature supports a practice.

Usually startups do not follow any software development method strictly. Instead they combine practices from different methods that suit their needs at the moment or use ad hoc working practices. (Melegati et al., 2016.) Starting point for startup practices considered in this report is Dande et al. (2014) study. They observed 63 practices by studying startups from Finland and Switzerland. Not all of these practices are focused solely on software development but they provided also practices considering customers and business goals.

Kamulegeya et al. (2017) have studied some of the practices found by Dande et al. (2014) in Ugandan startup context. Their findings indicate that startup practices observed in Finland and Switzerland apply widely in Uganda also. They suggest that most of the startup practices are universal while some of them might vary in different geographical areas or cultures.

In their study of software engineering anti-patterns in startups Klotins et al. (2018a) have identified some software startup practices. They focused on why software startups fail so often and identified potential challenges for these companies. They had studied reports of 88 software startups to search scenarios for potential failures. Their key findings include overscoping, failure to establish a feedback loop early enough, difficulties to scale product and focusing on more engineering rather than better engineering. They have found that many challenges that appear to be market or business related are in fact rooted partially or entirely to the software engineering. Klotins et al. (2018b) have also created a new map for categorizing software startup practices that differs from the one used by Dande et al. (2014).

Giardino et al. (2016) conducted a study to develop the Greenfield Startup Model that explains software development in early state startups. They focused on the aspects of structure, planning and control of software projects in startups based on 13 cases. In the study they found practices used by early state startups that supplement and confirm the founding of Dande et al. (2014) study.

In the systematic mapping study Paternoster et al. (2014) discovered total of 213 software development practices used in startups. They categorized practices in 4 topics: software development, managerial, process management and Tools and technologies. They discovered that startups rely on light-weight methods to pick and tailor practices that suit their situation in rapidly changing environment. For example minimum viable product (MVP) has been seen suitable for startups to fast develop a light-weight product to test and plan modifications according to the customer's needs. They didn't list all of the found practices in their paper but the practices discussed in their paper mostly support the findings of the study conducted by Dande et al. (2014).

Klotins et al. (2019) have studied software startup practices in the context of how software engineering is applied in startup context in their multi-vocal exploratory study of 88 startups experience reports. They used different categories of practices than Dande et al. (2014) as they based their research on the SWEBOK (Software Engineering Body of Knowledge) and business aspects. They found that the most frequently reported software engineering aspects were requirements engineering, software design and quality, and the most frequent business aspects were vision and strategy development. One of their main findings was that process of identifying product value proposition works as a bridge between marketing and engineering aspects of a product.

None of the considered studies had any major contradictions with the other studies. Only practices get venture capital and push your product (Dande et al., 2014; Giardino et al., 2016; Kamulegeya et al., 2017.) and fund it yourself (Dande et al., 2014; Kamulegeya et al., 2017; Yu et al., 2012) are in contradiction since they suggest completely different approach for funding. Even so they are not exclusive practices since they both might be right choice for startups in different stage of company's growth.

Rafiq et al. (2017) focused on requirements elicitation techniques of startups in their study. They also pointed out that MVP and learning from cus-

tomer's feedback are crucial for startups as well as Bajwa et al. (2017) study. All of the studies were unanimous that startups need to consider customer's needs and form good relations with their customers. These same focus areas are presented in the practices found by Dande et al. (2014).

Bajwa et al. (2017) among others propose that startups need to be prepared to change their focus since they are working with high uncertainty with innovative products. They suggest that failures should be allowed and learning from mistakes should guide the future of the endeavour. Dande et al. (2014) propose that on way to achieve agile changes in focus is to let anyone in development team to release or stop release. This results in faster releases and faster feedback.

Swenson et al. (2014) have approached subject of startup practices from marketing point of view. They suggest following practices to be necessity for a startup company: *Unleash the Power of Product Advocates, Empower Early-Adopter Customers, Land an Anchor Customer, Work with Benefactors* and *Build an Advisory Board*.

Understanding competence needs is seen as crucial factor for the startups' success. Creation of the first product requires substantial knowledge and skills from startup's founder team. Seppänen et al. (2017) found that capabilities needed in startups can be divided into three categories regarding the way they are brought to team. These categories are capabilities brought by founders, capabilities of hired or subcontracted team members and capabilities developed through learning.

Hallen, Bingham and Cohen (2014) compared startups graduated from accelerator programs in the USA with startups that didn't participate in any accelerator program. They found that startups participating in top accelerator programs reached their key milestones, such as raising venture capital or gaining customer traction, faster. However participation in many of the programs was not helpful or could even be harmful for startup's development.

The formal definition of a start-up accelerator, first offered by Cohen (2013) and Cohen and Hochberg (2014), is a fixed-term, cohort-based program, including mentorship and educational components, that culminates in a public pitch event. Other somewhat similar early stage startup groups or programs or other aids are referred as startup hubs and incubators.

Winston-Smith and Hannigan (2015) suggest that participating in top accelerator programs can be even more beneficial than getting angel funding. They found that comparable startups graduated from top accelerator program received their next round of financing sooner than ones that raised angel funding from leading angel investment groups instead.

Pitching can be used as a tool for imagination and design rather than only a sales tool. Pitching to other startups, customers, investors, mentors etc. helps startups to iterate their idea and business model according to the feedback from the stakeholders. It may also lead to funding or customer acquisition as a by-product.

Software startup practices found by Dande et al. (2014) are presented in the table 1 below with additional notes of other studies supporting the same

practice. The practices are categorized according to the original paper. Descriptions presented in the table are shortened version of ones presented in the original paper. The last column in the table is marking other studies that support a certain practice.

It's notable that these supporting studies don't necessarily use same terms or names of the practice but they have presented the same meaning in their own words and terminology. The list is not meant to be comprehensive listing of all software startup practices but the basis of startup practices considered by this study. This list will be later elaborated and extended by the findings of the empirical part of this study.

TABLE 1 Software Startup Practices

Identifier	Practice	Category	Description	Supporting literature
P1	Focus your product	Goals	Focus on the most potential customer segment. Be prepared to change the focus	Dande et al., 2014; Paternoster et al., 2014; Deakins & Dillon, 2005; Bajwa et al., 2017; Klotins et al., 2018a.
P2	Find your value proposition and stick to it on all levels	Goals	Create a valid value proposition. Discuss with experts from strategic and operational levels at customer's organization.	Dande et al., 2014; Giardino et al., 2016; Kamulegeya et al., 2017; Giardino et al., 2014; Cho et al., 2017; Klotins et al., 2019.
P3	Present the product as facilitating rather than competing to the competitors	Goals	Develop a product that can co-operate rather than compete with competitors	Dande et al., 2014; Assyne, 2017;
P4	Focus on goals, whys	Goals	Find real motivations behind customers' wishes by asking why they want something rather than just what they want. This way you understand customers' needs deeper and can address them in other situations as well.	Dande et al., 2014; Klotins et al.,2019.
P5	Use proven UX methods	Goals	Use proven UX development methods from the beginning. Validate ideas quickly by using prototypes.	Dande et al., 2014; Paternoster et al., 2014; Giardino et al., 2014; Klotins et al.,2019
P6	Do something spectacular	Goals	Create WOW effects and feelings to the customer to stand out in the competition.	Dande et al., 2014; Cho et al., 2017;
P7	Have a single product, no per customer variants	Goals	Have a modular and flexible single product rather than multiple per customer variants.	Dande et al., 2014; Paternoster et al., 2014
P8	Restrict the number of platforms that your product works on	Goals	Make business decisions on what platforms you want to support. Focus on the most important ones. For example the most used browsers and operating systems.	Dande et al., 2014
P9	Use enabling specifications	Goals	Enable specification to guide work efficiently. Let team work independently without constant intervene from the owner or customer.	Dande et al., 2014; Klotins et al.,2019
P10	Design and conduct experiments to find out about user preferences	Goals	Use experiments and communication with user to determinate in which directions product should be developed.	Dande et al., 2014; Paternoster et al., 2014; Deakins & Dillon, 2005; Rafiq et al., 2017; Björk et al., 2013; Klotins et al.,2019; Klotins et al., 2018a.
P11	Use tools to collect data about user behaviour	Goals	Use data to acknowledge user behaviour and choose best marketing channels.	Dande et al., 2014; Klotins et al.,2019; Klotins et al., 2018a.

P12	Make your idea into a product	Goals	Turn your ideas into products rather than projects. Projects are not easily scaled.	Dande et al., 2014; Giardino et al., 2016
P13	Outsource your growth	Culture	Use outsourcing to keep your focus on the product.	Dande et al., 2014
P14	Anyone can release and stop release	Culture	Allow anyone to make a release or stop it. Fast releases allow quick feedback from users.	Dande et al., 2014; Paternoster et al., 2014; Giardino et al., 2016
P15	Create the development culture before processes	Culture	In the beginning develop a culture that supports what you want to be. Processes are likely to change as company evolves so focus first on building the culture that fits your goals and future processes.	Dande et al., 2014
P16	Get venture capital and push your product	Culture	Try to get your product profitable fast with venture capital rather than develop it slowly in silence with low resources.	Dande et al., 2014; Giardino et al., 2016; Kamulegeya et al., 2017;
P17	Fund it yourself	Culture	Getting funding with proof of concept is not easy. Fund first yourself and get investment later.	Dande et al., 2014; Kamulegeya et al., 2017; Yu et al., 2012; Cho et al., 2017;
P18	Validate that your product sells	Culture	Validate your idea before starting development or try to get a few customers before you start developing.	Dande et al., 2014; Giardino et al., 2016; Klotins et al.,2019
P19	Focus early on those people who will give you income in the long run	Customer	Try to get your business model running from the start, even in small scale. Focus on paying customers to ensure that the company is profitable.	Dande et al., 2014; Klotins et al.,2019; Klotins et al., 2018a.
P20	Form deep relations with first customers to really understand their needs	Customer	To understand the customers and the business develop as deep relations as you can with the first customers.	Dande et al., 2014; Paternoster et al., 2014; Giardino et al., 2016; Giardino et al., 2014; Björk et al., 2013; Cho et al., 2017; Klotins et al.,2019
P21	Use planning tools that really show value provided to customer	Customer	Choose tools that allow mapping the value customer gets from what is done and planned.	Dande et al., 2014
P22	Start locally grow globally	Customer	Target local customers in the beginning but make all decisions considering the global growth.	Dande et al., 2014
P23	Adapt your release cycles to the culture of your users	Customer	Depending on your customers choose how fast releases are and how much can be changed at once.	Dande et al., 2014; Klotins et al.,2019;
P24	Keep customer communications simple and natural	Customer	A startup needs quick and good feedback from customers for development decisions. Try to encourage direct contacts by email or through integrated feedback mechanisms.	Dande et al., 2014; Kamulegeya et al., 2017; Giardino et al., 2014; da Rosa et al., 2017;
P25	Help customers create a great showcase for you with support	Customer	The first customers can provide a visible showcase to attract other customers.	Dande et al., 2014
P26	Flat organization	Organization	In flat organization people are committed to a common good and communications are easy as they don't require intermediates.	Dande et al., 2014; Giardino et al., 2016; Klotins et al.,2019
P27	Consider career expectations of good people	Organization	Keep team happy by offering opportunities to build up their skills. They can raise their market value as an insurance for the case that startup fails.	Dande et al., 2014
P28	Don't grow in personnel	Organization	If you don't need more resources or competence don't grow in personnel.	Dande et al., 2014
P29	Bind key people	Organization	Most important people should be shareholders, partners or founders because critical information is easily lost.	Dande et al., 2014; Kamulegeya et al., 2017;
P30	Form partnerships and bonds with other startups	Competence	Focus on developing your product and on your core business. For other issues find partnering startup. Startups are usually keen to co-operate.	Dande et al., 2014; Assyne, 2017; Tripathi et al., 2017; Chesbrough, 2003;
P31	Make your own strength as a “brand”	Competence	All startups should have exceptional skills or product. Turn this strength into a brand in the	Dande et al., 2014

			market.	
P32	Showing alternatives is the highest proof of expertise	Competence	Finding different alternatives for a solution is expertise. Explore alternatives to find a good solution.	Dande et al., 2014
P33	In the development of customer solutions, find a unique value proposition in your way of acting	Competence	Find the way of acting that differs from your competitors. For example super-fast or people centric.	Dande et al., 2014
P34	Follow communities	Competence	Everyone should follow communities to know what is happening and to find new values for customer.	Dande et al., 2014
P35	Share ideas and get more back	Competence	Sharing ideas will help you get valuable feedback.	Dande et al., 2014; Chesbrough, 2003;
P36	Small co-located teams	Team	Small teams with scarce resources need good communication to survive. Speaking in the same room is the most effective way to communicate.	Dande et al., 2014; Giardino et al., 2016; Kamulegeya et al., 2017;
P37	Have multi-skilled developers	Team	Startups have usually small teams, yet there are lots of different things to do. Multi-skilled developers are needed to address all the issues in startup without growing in personnel.	Dande et al., 2014; Paternoster et al., 2014; Giardino et al., 2016; Kamulegeya et al., 2017; Giardino et al., 2014;
P38	Keep teams stable in growth mode	Team	While growing as a company try to keep teams and individual roles stable.	Dande et al., 2014
P39	Let teams self-select	Team	Teams should be allowed to self-organize.	Dande et al., 2014
P40	Sharing competence in team	Team	In team everyone has slightly different expertise. Since startups need skilled developers sharing competence inside the team is necessary.	Dande et al., 2014; Paternoster et al., 2014; Giardino et al., 2016; Seppänen et al., 2017;
P41	Start with a competence focus and expand as needed	Team	In the beginning focus on specific competence with a small group of people. Expand team and competences later.	Dande et al., 2014; Kamulegeya et al., 2017; Seppänen et al., 2017;
P42	Start with small and experienced team and expand as needed	Team	Start with small and experienced team that has efficient ways to communicate. Anticipating all needed skills beforehand is hard.	Dande et al., 2014; Kamulegeya et al., 2017; Seppänen et al., 2017; Klotins et al.,2019
P43	Have different processes for different goals	Process	Choose different practices for different tasks if needed.	Dande et al., 2014
P44	Tailored gates and done criteria	Process	Process phases leading to something being done or assessed or accepted should reflect the overall process and business.	Dande et al., 2014; Klotins et al.,2019
P45	Time process improvements right	Process	Improve and change processes only when it is absolutely needed. At some point of the growth startup might need to change its preliminary processes.	Dande et al., 2014; Paternoster et al., 2014; Kamulegeya et al., 2017
P46	Find the overall development approach that fits your company and its business	Process	Find the best approach for your business. Don't follow latest trends if it's not best fit for you.	Dande et al., 2014
P47	Tailor common agile practices for your culture and needs	Process	Most textbook practices are highly general. Tailor them to fit your needs and culture.	Dande et al., 2014; Paternoster et al., 2014; Klotins et al., 2018b.
P48	Fail fast, stop and fix	Process	Allow developers to do things quickly and freely and stop if something goes wrong. They will then fix the problem and process in the team.	Dande et al., 2014; Paternoster et al., 2014; Giardino et al., 2016; Bajwa et al., 2017;
P49	Move fast and break things	Process	Prefer culture with fast development and where failing is acceptable.	Dande et al., 2014; Giardino et al., 2016; Bajwa et al., 2017;
P50	Forget Software Engineering	Process	Software development may be ad hoc and unorganized if it us good enough with the physical product.	Dande et al., 2014; Klotins et al., 2018b.
P51	Anything goes in product planning	Design	Startup needs to figure out new features, system concepts and new projects.	Dande et al., 2014
P52	To minimize problems with changes and variations,	Design	Develop a validated and focused concept to minimize risks with changes. Be still ready to	Dande et al., 2014

	develop a very focused concept		do changes if needed.	
P53	Develop only what is needed now	Design	Be efficient by developing only what is needed now.	Dande et al., 2014; Paternoster et al., 2014; Klotins et al.,2019
P54	Make features easy to remove	Design	Use techniques and architecture that make features easy to remove if needed.	Dande et al., 2014; Giardino et al., 2016
P55	Use extendable product architecture	Design	Use architecture and techniques that allow to extent design easily.	Dande et al., 2014; Paternoster et al., 2014
P56	Only use reliable metrics	Testing	Use reliable metrics to validate things. Wrong metrics might do harm for validation.	Dande et al., 2014; Giardino et al., 2016
P57	Bughunt	Testing	During fast development of new features arrange days for bughunt. Make bughunt fun occasion when everyone is searching for bugs.	Dande et al., 2014
P58	Test APIs automatically, UIs manually	Testing	APIs can be tested by tools that are easy to find and cheap. Test UI manually in the beginning.	Dande et al., 2014
P59	Use generic, non-proprietary technologies	Technology	Use platform independent technologies to avoid re-implementing features.	Dande et al., 2014; Paternoster et al., 2014; Giardino et al., 2016
P60	Create a solid platform	Technology	Keep scaling in mind while developing a platform.	Dande et al., 2014; Paternoster et al., 2014
P61	Choose scalable technologies	Technology	Favour development techniques that scale easily.	Dande et al., 2014; Paternoster et al., 2014; Giardino et al., 2016; Wall, 2001; Giardino et al., 2014;
P62	Use the most efficient programming languages and platforms	Technology	With a small team choose the most efficient programming languages and development platforms.	Dande et al., 2014; Paternoster et al., 2014; Giardino et al., 2016; Wall, 2001;
P63	Start with familiar technologies and processes	Technology	Save the time of learning new technologies and processes by using those that team is familiar with.	Dande et al., 2014

3 ESSENCE FRAMEWORK

Software Engineering Methods and Theory's (SEMAT) Essence framework is introduced in this section of the thesis. The background of SEMAT is provided first followed by more detailed definition of the Essence framework's kernel. The use cases of the Essence are provided in the sub-section 3.3 and 3.4. Criticism of the framework and SEMAT is provided in the sub-section 3.5 and competing frameworks are introduced in the sub-section 3.6.

3.1 Background

The Essence was introduced by a group of people who founded SEMAT community. They felt that it is time to fundamentally change how people see software engineering methods. Goal was to redefine software engineering as a rigorous discipline according to the problems they identified for the current state of the subject. (Jacobson et al., 2012) They wrote The SEMAT Call for Action Statement (Jacobson, Meyer & Soley, 2009) to describe the current problems and issues they wish to improve with their work. The SEMAT call for action gained wide support with academia and industry while the number of supporters and signatories of SEMAT are still growing. Some critique has been raised after the SEMAT Call for Action, addressed in the subsection 3.5.

SEMAT community identified the specific problems in software engineering as following (Jacobson et al., 2009):

- *The prevalence of fads more typical of fashion industry than of an engineering discipline.*
- *The lack of a sound, widely accepted theoretical basis.*
- *The huge number of methods and method variants, with differences little understood and artificially magnified.*
- *The lack of credible experimental evaluation and validation.*
- *The split between industry practice and academic research.*

They also described the key principles for the process of redefining software engineering based on a solid theory, proven principles, and best practices that:

- *Includes a kernel of widely-agreed elements, extensible for specific uses.*
- *Addresses both technology and people issues.*
- *Is supported by industry, academia, researchers and users.*
- *Supports extension in the face of changing requirements and technology.*

SEMAT community's answer to redefined software engineering is Essence. It is a framework providing the common ground for the creation, use and improvement of software development methods. The Essence consists of two parts, kernel and language. The Essence allows people to examine the essential parts of their current and future methods and practices for comparison, evaluation, tailoring and use of these methods. It also allows teams to continuously evaluate the progress and health of the software development endeavours. (Jacobson et al., 2012; Submitters, 2012.)

The kernel is the core of the Essence describing the commonalities of software development practices. It consists of the fundamental elements that are prevalent in all software development endeavours including for example team, requirements and stakeholders. Those elements have states that represent progress and health of the element. The kernel allows practitioners to compare different methods and make considered decisions about practices they use. (Submitters, 2012; Striewe, 2012.)

The language defines abstract syntax, dynamic semantics, graphic syntax, and textual syntax to describe the kernel. The language allows composing two or more practices to form a new practice. The kernel and the language together form the Essence. (Submitters, 2012.)

This theory was chosen to be a foundation of this thesis because of its universal functionality. Software development has numerous different methods and the Essence does not bind you with just one or a few of them (Dwolatzky, 2012). The Essence rather allows comparing different methods and their suitability to the issue at hand. It also enables mixing two or more methods to suit specific development situation. (Jacobson et al., 2012)

3.2 The Kernel

Object Management Group (OMG) standard (Submitters, 2012) defines the kernel as a simple stripped-down set of definitions of the essence of software development methods and practices. The kernel focuses on defining the common ground for the software development methods that allows comparing and tailoring of different methods. One benefit of the kernel is that it allows people to start software development endeavours with a minimal method adding practic-

es as needed in the future. In this section the kernel is introduced as it is defined in the OMG standard. (Submitters, 2012.)

The kernel is organized into three areas of concern that focus on a specific aspect of software development. These areas are: *customer, solution and endeavour*. The customer area of concern includes everything to do with the actual use and exploitation of the software system to be produced. The solution area of concern includes everything to do the specification and development of software system. The endeavour area of concern includes everything to do with the team, and the way that they approach their work.

Each area of concern contains a small number of alphas, activity spaces and competencies. Since the Essence is independent from other software development methods, that can expand it, it does not include any other elements that only make sense within the context of a specific method. (Submitters, 2012; Jacobson et al., 2019; Jacobson et al., 2012.)

The alphas represent the fundamental things to work with in software development. The alphas describe things that are needed to manage, produce and use in the process of developing, maintaining and supporting software. This means they are necessary for assessing he progress and health of a software endeavour. The alphas also work as an anchor for additional sub-alphas required by the software development methods used by the team. (Submitters, 2012; Jacobson et al., 2019; Jacobson et al., 2012.)

The activity spaces represent the essential things to do in software development. They describe the challenges that team faces while developing, maintaining and supporting software systems. They also describe the things that team needs to do to meet its objects. (Submitters, 2012; Jacobson et al., 2012.)

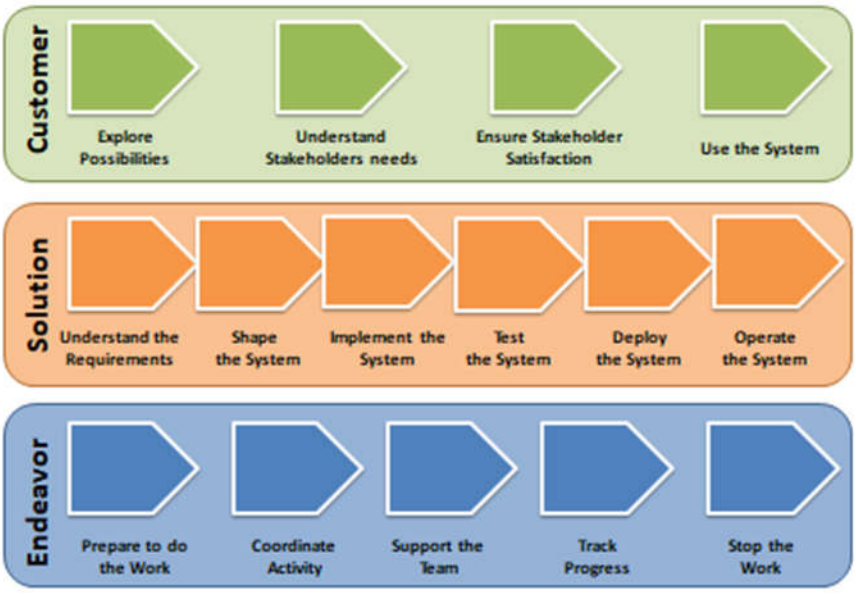

FIGURE 3 Activity Spaces (Submitters, 2012)

Competencies represent the capabilities required for software development endeavours. The competencies complete the alphas and the activity spaces by introducing the necessary capabilities to success in the endeavour carried out by following the alphas and the activity spaces. (Submitters, 2012; Jacobson et al., 2012.)

FIGURE 4 Competencies (Submitters, 2012)

The alphas in the kernel describe the key concepts in the software development. Each of the alphas has a small set of pre-defined states. The states are used to monitor progress and health of that alpha. This means that teams can use the states to analyse and deal with the risks and challenges for the alpha. Each state has also a set of pre-defined checklists that are used to define the state of the alpha.

States are not one-way linear progression but a tool for continuous review of the progress of that alpha. Depending on the chosen practices iteration through the states can be done multiple times. Rather than seeing the alphas as physical partitioning or abstract work products they should be considered as indicators of the most important monitored things in the software development endeavour. For example team members can be both part of the team alpha and part of the stakeholders alpha. (Submitters, 2012; Ng & Huang, 2013; Jacobson et al., 2012.)

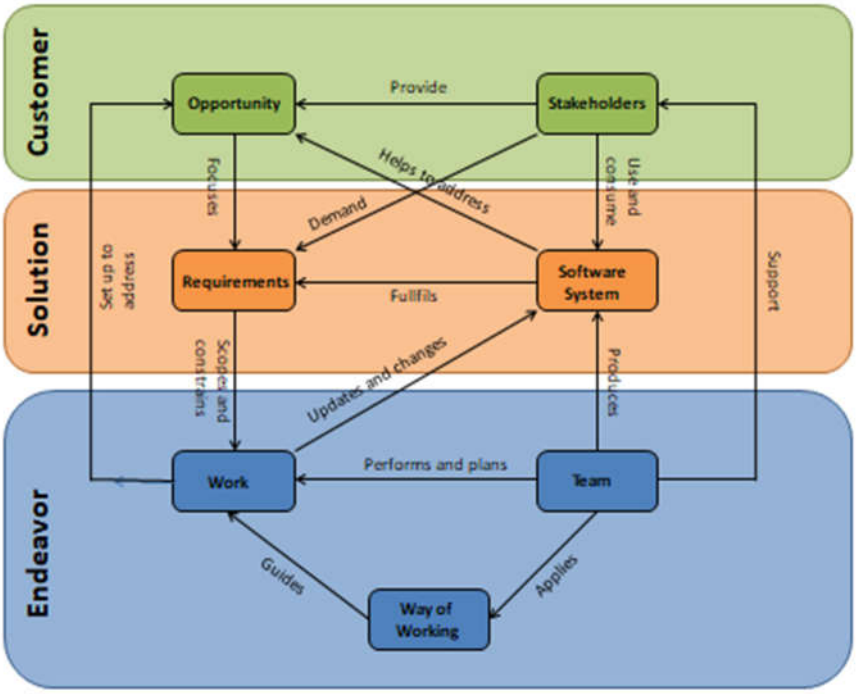

FIGURE 5 Alphas (Submitters, 2012)

3.3 Essence in Practice

The Essence is designed as an abstract model of the most important monitored things related to software system development. This means while it has many purposes for monitoring the health and progress of software development endeavours it can be too abstract to actually conduct the software development work. For this reason it is designed to be used in conjunction with other selected methods and practices. For example while every software development endeavour has requirement items types of those items may vary in different projects. Some projects might use user stories while some user cases or both of them. (Jacobson et al., 2012; Submitters, 2012.)

The first use of the Essence is the project management. It allows teams to monitor their work and plan next steps regardless of chosen methods and practices as a whole. With the Essence a team can plan their actions with all of the most important aspects of development process simultaneously. By evaluating the process of all the alphas at the same time the team can choose next steps so that all of the alphas are balanced. This allows teams to progress at the same rate in all of the most important aspects of their endeavour without overlooking anything. With the Essence teams can continuously add and remove practices if endeavour situation requires it or if they come up with better fitting practices while working on a project. For example in everyday life the Essence can be used to run an iteration and entire development endeavour from idea to product. (Jacobson et al., 2012.)

The second use of the Essence is to present different software development methods and practices. Its abstract nature allows adding practices from development methods and mixing different methods. With the kernel practices are presented as distinct modular units that can be used or not or changed at any given moment. Traditionally methods are seen as indistinguishable package of practices that works only as a whole. (Jacobson et al., 2012. Jacobson et al., 2019)

Jacobson (Jacobson et al., 2019) points out that developers typically don't have time or interest to study detailed methods or practices. They learn the essentials to get team working as fast as possible rather than first learning "all" the things about subject. Jacobson (Jacobson et al., 2019) has identified that about 5% what experts know about the subject forms the essentials. In some cases team needs to focus on more than just the essentials, so they have made the Essence modular and applicable for different levels of details. See section 3.4 for more information about the modification of the Essence.

The third use case for the Essence is teaching. The Essence has been adopted by some universities to teach new software engineering students basic principles of software development endeavour. For example Gil et al. (2014) have had good preliminary experiences after adopting the Essence alongside agile methods to teach software development for bachelor students. Huang and Ng (2014) have suggested that the Essence could also be used in the software engineering research as a common framework for reporting empirical findings.

3.4 Essentializing Practices

The Essence kernel and language can be used to make different practices explicit. This process of describing practices using the Essence framework is referred as essentialization. (Jacobson et al., 2019) The idea of essentialization is to make practices from different sources compatible and also to help introducing new practices to team. Practices used by team can originate from various sources with various ways to describe them or even practices that are not described previously. This means that combining them is difficult and understanding them can be hard for a new team member. The way essentialization supports these efforts is a coherent way to describe all the practices used by the team using consistent language and symbols. Essentialization also helps researchers to describe practices for different sources explicitly in consistent way. (Jacobson et al., 2019)

Making a practice explicit and reusable through the essentialization process means ensuring that practice is a proven practice and the elements described are essential elements only. Usually teams need many practices that are designed to be used together in so called practice architecture. Even though practices can be seen as separate elements they are not usually independent. Thus the practice architecture is useful to describe the relation of practices. The practice architecture can be constructed for example in a layered form with

more generic practices at the bottom and more specialized practices at the top. (Jacobson et al., 2019)

The core idea of essentialization process is to take chosen practice and gather the essential elements of that practice describing them with the kernel and the Essence language. The essential elements of a practice are things that need to be done (activity), abilities and capabilities needed to conduct the practice (competency), the tangible things produced using that practice (work product) and things that need to be monitored regarding their progression (alpha).

Some practices also include other essential elements (patterns). In the Essence context pattern refers to other essential elements of a practice that are not activities, competencies, work products or alphas. Pattern can be for example a role in a team or checkpoints that synchronize alpha state progression.

In a figure below is shown an example of simple essentialization of SCRUM practice added to the Essence. In the figure scrum team, product owner and scrum master roles are presented as patterns. Sprint planning, daily scrum, sprint review, and sprint retrospective are activities. Product backlog, sprint backlog, and increment are work products and sprint and product backlog Item are alphas. (Jacobson et al., 2019; Elvesæter, 2013.)

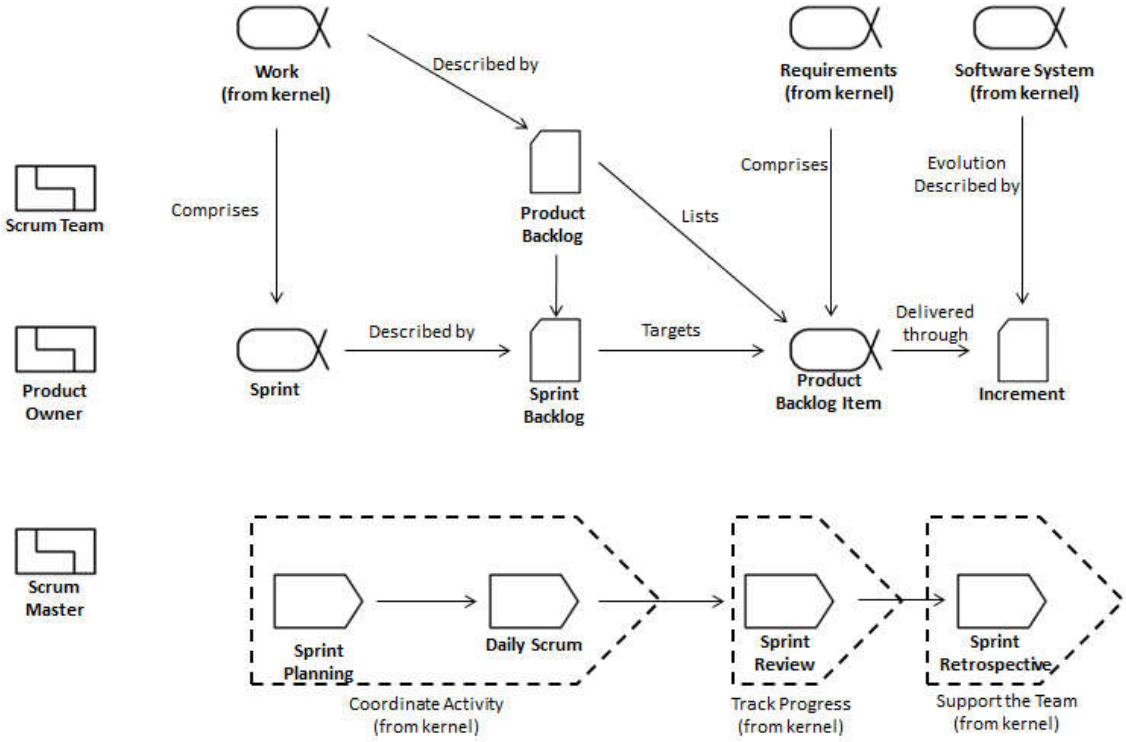

FIGURE 6 Scrum Essentialization (Jacobson et al., 2019.)

As presented in figure above, a particular method can be plugged into the Essence to extend the framework according to the needs of the team. The added alphas or sub-alphas also have their own states like alpha elements in the kernel. For practical use of the kernel the states of added alphas and sub-alphas are

necessary to monitor the progress of added practices. The language is designed to support definition of different views that are suitable for different practitioners. For example some teams might prefer using the checkpoints of the alpha states to guide the work or some might decide to use list of activities to do for monitoring progress from one state to another. (Striewe et al., 2012.)

Evensen et al. (2018) have developed a tool called Essencery which allows professionals and academics to present different software development practices and methods using the Essence language's graphical syntax. This tool can also support software startup practices by introducing them in an established manner following the Essence language. The tool can be used to add informal software startup practices to the agile methods and modify those methods to meet the reality of startup's endeavour. Combining of different formal or informal practices can be done using the tool. In their study Evensen et al. (2018) have argued that Essence lacks practical adoption and they intended to create a tool that can be used for essentialization of practices and methods. Their study suggests that the tool Essencery is easy to learn and use and can be utilized for practical use of the framework.

3.5 Criticism

SEMAT has numerous signatories and supporters for their process to refine software engineering. However it has also raised some criticism in academic community. For example one of the initiators of the agile movement Alistair Cockburn (2010) has been criticizing SEMAT initiative not to be able to produce what they pursue. Most of the critique for SEMAT has been published in grey literature rather than in academic sources and they have been done during the initiative state of the process. The Essence framework is relatively new which might be a factor why only a little criticism has been raised after its publication. This subsection points out some critique SEMAT initiative and the Essence framework have generated.

Smolander and Päivärinta (2013) have proposed a model for theorizing development practices called Coat Hanger. They also point some critique for the Essence and propose some complementary elements to the Essence. They don't suggest their model to be direct rival for the Essence since they aim for a different goal. They use concepts of technical rationality and reflection in action (Schön, 2017.) to compare different mindsets behind the Essence and the Coat Hanger.

Smolander and Päivärinta (2013) argue that the goal of theorizing for SEMAT is to improve practices by evaluating and choosing best practices for the given context. In contradiction their own goal is to refine understanding and theories through learning and reflection. They criticize the Essence for having its ontology fixed and based on either previous experiences, conventional views or a best guess. They suggest that ontological difference with the Coat Hanger allows more freedom to theory creation.

The Coat Hanger does not make specific assumptions on the ontology of professional software development practices. They criticize the Essence of its one-time view and argue that theorizing software engineering should be seen as continuous dialect process between experience and theory. According to Smolander and Päivärinta (2013) SEMAT has focused on standardization angle where concepts are based on consensus. They claim that SEMAT doesn't take into account that theories should evolve. Even though the authors criticize the Essence they see it as a legitimate initiative and their main critique deals with the concepts of theories and theorizing.

The critique of the Essence raised by Smolander and Päivärinta (2013) includes a claim that the purposes of the Work alpha and the Way-of-working alphas are unclear and should be better defined. They also suggest that the alphas describing the solution and endeavour should be checked again against the literature concerning contextual issues having an impact on software engineering success. They suggest additions to the Essence in form of new alphas to describe rationale, deviations from intended practices, actual impacts, and lessons learned. They also address that the Essence should stay reflective for future theoretical advancements and that especially in organizational theory related to software development. Park et al. (n.d.) have answered to this critique that the Essence is expected to be stable but not static as it will continue to evolve while going through real applications in the field and through feedback.

Cockburn (2010) has pointed out some criticism of SEMAT initiative. His critique doesn't focus on the Essence framework but the early stages of its development and ideas behind the process. His main points of critique are:

1. *The call-for-action was inflammatory, poorly researched and logically broken; it uses the very hype-for-fashion language that it decries, it is internally contradictory, the problems it deplores it cannot fix, and its proposed solutions do not address the problems named. It sets a direction in tone and content that does not do the topic justice. It is a red herring, intended to generate support through appeal-to-authority, hype, and ambition.*
2. *They are unlikely to discuss either engineering or engineering theory – a more accurate name for the initiative would be the Meta-Process-Kernel initiative.*
3. *Whatever they produce is unlikely to affect topics that matter to the industry.*

He argues that even though Jacobson and Meyer (2009) see software development methodology as fashion or politics being prevalent with fads they are correct but unable to change that due to nature of human beings and our tendency of being attracted to new popular trends. He even suggests that SEMAT is doing this same thing with their initiative even though they criticize software engineering of it. In his opinion SEMAT call-for-action is factually and logically broken and he suggests that people who are not really versed in the field should look for critique and alternatives also. He also thinks that initiative

should be named Meta-Method-Kernel initiative to reflect its actual goal rather than engineering theory.

Fowler (2010) widely agrees with Cockburn's (2010) critique on the Essence framework and SEMAT initiative. His main point of critique is that since people are the central element of software engineering and they are inherently non-linear and unpredictable effort to produce a software meta-method-kernel is doomed. This is especially because people are not predictable agents that can be described with tractable mathematics. Zalewski (2013) has reviewed a book "The essence of software engineering : applying the SEMAT kernel" (Jacobson et al., 2013.) his focus is on book review but he also points out that the Essence is not focusing enough on the aspect of design in software development, or at least not in the case of this book.

Aranda (2009) agrees with Cockburn's (2010) and Fowler's (2010) critique on the Essence framework and SEMAT initiative. He adds to the critique that software development has little to do with most established engineering disciplines and disagrees with initiative's aim to pull software engineering discipline closer to other engineering disciplines. He also points out that fads are not necessarily bad thing for discipline since many fads have turned out to be good ideas that have evolved to mainstream practices. For example object orientation was seen as a fad but it's now widely accepted and used. In Aranda's opinion SEMAT is right that software engineering lacks a sound and widely accepted theoretical basis but he disagrees that it is possible to create universal methods for software development since different settings in software development will always require different methods. For example developing an operating system is not the same as developing videogames. He also agrees with SEMAT initiative that there is too much split between industry practices and academic research in this field.

Ibargüengoitia and Oktaba (2014) have studied the Essence in context of teaching and inexperienced practitioners. They argue that the Essence works well as a tool for software industry for measuring the progress of projects with experienced teams. On contrary they suggest the activity spaces are too general to serve as a guide of what to do for inexperienced practitioners and students while they don't necessarily know how to get from one alpha state to another.

3.6 Competing Frameworks

Software and Systems Process Engineering Metamodel (SPEM) is another proposal for methods and practices presentation that addresses roughly the same issues than the Essence. Its aim is to provide a common framework for represent practices, methods and software development models. OMG standard (2008) describes SPEM as "a process engineering meta-model as well as conceptual framework, which can provide the necessary concepts for modelling, documenting, presenting, managing, interchanging, and enacting development methods and processes. An implementation of this meta-model would be tar-

geted at process engineers, project leads, project and program managers who are responsible for maintaining and implementing processes for their development organizations or individual projects."

SPEM's downside is that it is more complex and it doesn't provide the support for monitoring projects progress as the kernel. Even though SPEM is older alternative than the Essence it has not been widely recognized or adopted. It focuses on organizations with separate group of individuals that are in charge of maintaining processes. Its main focus is on process engineers, project leaders and managers. (González-Pérez et al., 2014; Striewe et al., 2012.) In this thesis the Essence has been chosen as underlying framework because of these issues and the fact that the Essence supports enactment better while also being easier to learn and use in practice.

Smolander and Päivärintä (2013) have created Coat Hanger framework for theorizing development practices. The main difference between the Essence and the Coat Hanger is that the Coat Hanger focuses more on learning and reflection rather than providing common ground for all software development endeavours. It doesn't provide any tangible way to describe practices used by an organization but it focuses on how new theories should be created in the academia. This is the reason why the Essence fits better for the purpose of this thesis.

4 THEORETICAL FRAMEWORK

This section provides summary of theoretical framework of this thesis. This theoretical framework was used as a basis of interviews for empirical data collection (chapter 5). Theoretical framework is based on the literature review presented in the sections 2 and 3. Theoretical framework was later used as a basis for empirical part of this thesis. First the alphas from the Essence kernel are introduced. These will be used as categories for the studied practices. Later the practices found by Dande et al. (2014) will be re-categorized by replacing original categories with the alphas.

The Essence provides a framework describing the ubiquitous parts of software development. It can also be used as a tool for presenting different software development practices and methods using same syntax and language. Third usage of the Essence is evaluation of software development progress in all of the most important aspects of endeavour. (Submitters, 2012.)

The Essence isn't only suitable framework to categorize practices used by software startups as for example Klotins et al. (2019) have conducted a study on software startup practices using categorization from SWEBOK knowledge areas. The Essence was chosen as framework for its multiple usages for academia and professionals and since it hasn't been widely tested in the startup environment. Klotins et al. (2018b) have also formed a startup context map to categorize startup practices under different goals and environmental factors.

For this thesis the central part of the Essence is the kernel. The kernel was chosen as a framework because it presents the most important sectors of software development called alphas (Submitters, 2012). These alphas present the areas of startup's activities that are significant for this study. Under the alphas can be included different practices a certain team uses for their development endeavour (Submitters, 2012). The following figure 7 is the basis of theoretical framework and empirical study of this thesis.

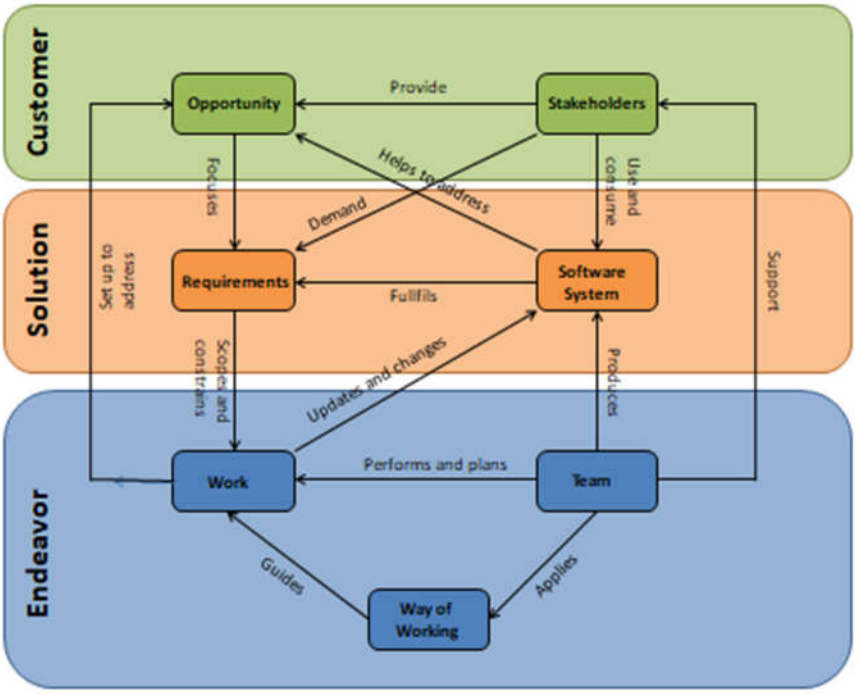

FIGURE 7 Framework

These alphas presented in this section and following subsections provide the theoretical framework that guides the empirical study of this thesis. They work as a basis for interviews conducted to gain empirical material. Results of the empirical study are also presented and categorized following the kernel and its alphas. In the following subsections the alphas are described and software startup practices found by Dande et al. (2014) are re-categorised under the alphas. Those practices not distinctly fitting under an alpha are presented in the subsection 4.8.

4.1 Opportunity

Opportunity alpha is described as "The set of circumstances that makes it appropriate to develop or change a software system." by Submitters (2012). The opportunity deals with the reasons to develop or change software system. It also includes the teams understanding of the needs stakeholders have. These needs should guide the requirements and provide the justification of the development endeavour. (Jacobson et al., 2012; Submitters, 2012.) In the table 2 the practices focusing on the aspects that are associated with the opportunity alpha are listed.

TABLE 2 Opportunity Related Practices

Identifier	Practice	Description	Alpha
P1	Focus your product	Focus on the most potential customer segment. Be prepared to change the focus	Opportunity
P2	Find your value proposition and stick to it on all levels	Create a valid value proposition. Discuss with experts from strategic and operational levels at customer's organization.	Opportunity
P4	Focus on goals, whys	Find real motivations behind customers' wishes by asking why they want something rather than just what they want. This way you understand customers' needs deeper and can address them in other situations as well.	Opportunity
P18	Validate that your product sells	Validate your idea before starting development or try to get a few customers before you start developing.	Opportunity
P20	Form deep relations with first customers to really understand their needs	To understand the customers and the business develop as deep relations as you can with the first customers.	Opportunity
P33	In the development of customer solutions, find a unique value proposition in your way of acting	Find the way of acting that differs from your competitors. For example super-fast or people centric.	Opportunity
P34	Follow communities	Everyone should follow communities to know what is happening and to find new values for customer.	Opportunity

4.2 Stakeholders

Stakeholders alpha embodies the people and organizations affecting or affected by a software system. They can be groups of people or a single person who's somehow connected to a software system that is developed. One part of the stakeholder alpha is always a customer or a user of the software system. It also includes people who will make software system development possible. Stakeholders provide the opportunity by supporting the team and ensuring that acceptable software system is being developed. (Jacobson et al., 2012; Submitters, 2012.) The practices concerning the stakeholders alpha are listed in the table 3 bellow.

TABLE 3 Stakeholders Related Practices

Identifier	Practice	Description	Alpha
P24	Keep customer communications simple and natural	A startup needs quick and good feedback from customers for development decisions. Try to encourage direct contacts by email or through integrated feedback mechanisms.	Stakeholders
P32	Showing alternatives is the highest proof of expertise	Finding different alternatives for a solution is expertise. Explore alternatives to find a good solution.	Stakeholders
P35	Share ideas and get more back	Sharing ideas will help you get valuable feedback.	Stakeholders

4.3 Requirements

Requirements alpha includes the features software system has to have to address the opportunity and satisfy stakeholders. Requirements of the software system need to be addressed to share the knowledge among the team what

software must do to respond to the opportunity. Requirements are what drive the development and testing of a new system. (Jacobson et al., 2012; Submitters, 2012.) The following table 4 presents the practices related to the requirements alpha.

TABLE 4 Requirements Related Practices

Identifier	Practice	Description	Alpha
P3	Present the product as facilitating rather than competing to the competitors	Develop a product that can co-operate rather than compete with competitors	Requirements
P5	Use proven UX methods	Use proven UX development methods from the beginning. Validate ideas quickly by using prototypes.	Requirements
P10	Design and conduct experiments to find out about user preferences	Use experiments and communication with user to determinate in which directions product should be developed.	Requirements
P21	Use planning tools that really show value provided to customer	Choose tools that allow mapping the value customer gets from what is done and planned.	Requirements
P51	Anything goes in product planning	Startup needs to figure out new features, system concepts and new projects.	Requirements
P52	To minimize problems with changes and variations, develop a very focused concept	Develop a validated and focused concept to minimize risks with changes. Be still ready to do changes if needed.	Requirements
P53	Develop only what is needed now	Be efficient by developing only what is needed now.	Requirements

4.4 Software system

The alpha called software system considers a system that is being developed or changed. It includes software, hardware and data used to create a software system. It is the product the team is working on and it can be also used to describe a part of a bigger software, hardware, business or social solution. (Jacobson et al., 2012; Submitters, 2012.) The practices associated with the software system alpha are presented in the table 5.

TABLE 5 Software System Related Practices

Identifier	Practice	Description	Alpha
P7	Have a single product, no per customer variants	Have a modular and flexible single product rather than multiple per customer variants.	Software System
P8	Restrict the number of platforms that your product works on	Make business decisions on what platforms you want to support. Focus on the most important ones. For example the most used browsers and operating systems.	Software System
P14	Anyone can release and stop release	Allow anyone to make a release or stop it. Fast releases allow quick feedback from users.	Software System
P23	Adapt your release cycles to the culture of your users	Depending on your customers choose how fast releases are and how much can be changed at once.	Software System
P54	Make features easy to remove	Use techniques and architecture that make features easy to remove if needed.	Software System
P55	Use extendable product architecture	Use architecture and techniques that allow to extent design easily.	Software System
P57	Bughunt	During fast development of new features arrange days for bughunt. Make bughunt fun occasion when everyone is searching for bugs.	Software System
P58	Test APIs automatically, UIs manually	APIs can be tested by tools that are easy to find and cheap. Test UI manually in the beginning.	Software System
P59	Use generic, non-proprietary technologies	Use platform independent technologies to avoid re-implementing features.	Software System
P60	Create a solid platform	Keep scaling in mind while developing a platform.	Software System

4.5 Work

Work alpha describes all the mental and physical work done to develop or change a software system. It is everything that the team needs to do to achieve the goals of endeavour matching the requirements and addressing the opportunity presented by stakeholders. (Jacobson et al., 2012; Submitters, 2012.) The practices concerning the work alpha are presented in the following table 6.

TABLE 6 Work Related Practices

Identifier	Practice	Description	Alpha
P44	Tailored gates and done criteria	Process phases leading to something being done or assessed or accepted should reflect the overall process and business.	Work
P48	Fail fast, stop and fix	Allow developers to do things quickly and freely and stop if something goes wrong. They will then fix the problem and process in the team.	Work
P62	Use the most efficient programming languages and platforms	With a small team choose the most efficient programming languages and development platforms.	Work

4.6 Team

Team alpha is a group of people who are actively part of the development endeavour. They can work with development itself or maintenance, delivery, or

support of a software system for example. In bigger projects multiple teams might be working together to achieve the goal of a working software system. Team plans and performs all the work needed for the endeavour. (Jacobson et al., 2012; Submitters, 2012.) The following table 7 introduces the practices concerning the team alpha.

TABLE 7 Team Related Practices

Identifier	Practice	Description	Alpha
P26	Flat organization	In flat organization people are committed to a common good and communications are easy as they don't require intermediates.	Team
P27	Consider career expectations of good people	Keep team happy by offering opportunities to build up their skills. They can raise their market value as an insurance for the case that startup fails.	Team
P28	Don't grow in personnel	If you don't need more resources or competence don't grow in personnel.	Team
P29	Bind key people	Most important people should be shareholders, partners or founders because critical information is easily lost.	Team
P36	Small co-located teams	Small teams with scarce resources need good communication to survive. Speaking in the same room is the most effective way to communicate.	Team
P37	Have multi-skilled developers	Startups have usually small teams, yet there are lots of different things to do. Multi-skilled developers are needed to address all the issues in startup without growing in personnel.	Team
P38	Keep teams stable in growth mode	While growing as a company try to keep teams and individual roles stable.	Team
P40	Sharing competence in team	In team everyone has slightly different expertise. Since startups need skilled developers sharing competence inside the team is necessary.	Team
P41	Start with a competence focus and expand as needed	In the beginning focus on specific competence with a small group of people. Expand team and competences later.	Team
P42	Start with small and experienced team and expand as needed	Start with small and experienced team that has efficient ways to communicate. Anticipating all needed skills beforehand is hard.	Team

4.7 Way of Working

Way-of-working is the alpha that describes working practices that guide the work. It is closely connected to work alpha since it includes all the tools and practices that guide work itself. Way of working evolves while team performs work and learns new better ways to conduct the work. (Jacobson et al., 2012; Submitters, 2012.)In the following table 8 the practices associated with the way of working alpha are introduced.

TABLE 8 Way of Working Related Practices

Identifier	Practice	Description	Alpha
P9	Use enabling specifications	Enable specification to guide work efficiently. Let team work independently without constant intervene from the owner or customer.	Way of Working
P15	Create the development culture before processes	In the beginning develop a culture that supports what you want to be. Processes are likely to change as company evolves so focus first on building the culture that fits your goals and future processes.	Way of Working
P39	Let teams self-select	Teams should be allowed to self-organize.	Way of Working
P43	Have different processes for different goals	Choose different practices for different tasks if needed.	Way of Working
P45	Time process improvements right	Improve and change processes only when it is absolutely needed. At some point of the growth startup might need to change its preliminary processes.	Way of Working
P46	Find the overall development approach that fits your company and its business	Find the best approach for your business. Don't follow latest trends if it's not best fit for you.	Way of Working
P47	Tailor common agile practices for your culture and needs	Most textbook practices are highly general. Tailor them to fit your needs and culture.	Way of Working
P49	Move fast and break things	Prefer culture with fast development and where failing is acceptable.	Way of Working
P50	Forget Software Engineering	Software development may be ad hoc and unorganized if it us good enough with the physical product.	Way of Working
P61	Choose scalable technologies	Favour development techniques that scale easily.	Way of Working
P63	Start with familiar technologies and processes	Save the time of learning new technologies and processes by using those that team is familiar with.	Way of Working

4.8 Other Practices

In this subsection the practices found by Dande et al. (2014) that do not fit distinctly under any alpha from the Essence are presented. The practices are displayed in the table 9. Most of the practices that do not fit distinctively under any of the alphas from the Essence kernel deal with business related issues such as branding, marketing or business model development. It is natural that these do not find clear category in the Essence framework since the framework was designed considering software development and practices found by Dande et al. are studied from startup companies that develop software systems as their product or part of it and also have to design their business and perform business related tasks simultaneously. Klotins et al. (2019) have also discovered in their study that software startup practices are not limited to the software engineering practices but rather joint together with business aspects.

TABLE 9 Other Practices

Identifier	Practice	Description	Area related to
P6	Do something spectacular	Create WOW effects and feelings to the customer to stand out in the competition.	Marketing
P11	Use tools to collect data about user behaviour	Use data to acknowledge user behaviour and choose best marketing channels.	Business metrics
P12	Make your idea into a product	Turn your ideas into products rather than projects. Projects are not easily scaled.	Business process
P13	Outsource your growth	Use outsourcing to keep your focus on the product.	Business process
P16	Get venture capital and push your product	Try to get your product profitable fast with venture capital rather than develop it slowly in silence with low resources.	Funding
P17	Fund it yourself	Getting funding with proof of concept is not easy. Fund first yourself and get investment later.	Funding
P19	Focus early on those people who will give you income in the long run	Try to get your business model running from the start, even in small scale. Focus on paying customers to ensure that the company is profitable.	Business / Revenue streams
P22	Start locally grow globally	Target local customers in the beginning but make all decisions considering the global growth.	Business / Marketing
P25	Help customers create a great showcase for you with support	The first customers can provide a visible showcase to attract other customers.	Marketing
P30	Form partnerships and bonds with other startups	Focus on developing your product and on your core business. For other issues find partnering startup. Startups are usually keen to co-operate.	Business process
P31	Make your own strength as a “brand”	All startups should have exceptional skills or product. Turn this strength into a brand in the market.	Marketing
P56	Only use reliable metrics	Use reliable metrics to validate things. Wrong metrics might do harm for validation.	Business metrics

5 RESEARCH DESIGN AND METHODOLOGY

This section describes the research methodologies and the data collection methods used in this thesis. This section also describes how research process was conducted in practice. The purpose of the empirical section of the thesis is to validate and improve theoretical framework. Considering the essence of researching software startups, a qualitative research approach in form of a multiple case-study has been chosen as the method for the empirical research.

5.1 Research Methodology and Data Collection

The empirical section of the thesis consists of thirteen case studies. Case studies can include only quantitative data, only qualitative data or both of them (Yin, 2013). A qualitative method was chosen in order to understand the practices software startups adopt in their endeavours. The study aims to create understanding of practices used by software startups which can be further tested with quantitative studies to generate solid knowledge in the future. Qualitative analysis can be used to find patterns or often reappearing themes inside the field of interest (Caracelli and Greene, 1993).

In qualitative research data collection can be done through interviews, archival research or observational techniques. Written data sources may include published or unpublished documents, reports, memos, letters and so forth. Meyers and Newman (2006) suggest interviews as a sufficient way of conducting qualitative research. Interviews can be structured, semi-structured or unstructured.

Structured interviews typically have a rigid order and form of the questions without any room for improvisation. Thematic and other semi-structured interviews have a set of themes that are considered in all of the interviews but order and form of the questions are less rigid leaving room for more detailed questions if needed. Unstructured interviews have no script while it is conduct-

ed in a conversational manner. Group interviews include two or more people interviewed at once in a structured or unstructured approach.

Thematic interviews were chosen as a data collection method of this thesis because its suitability for the context. Thematic interviews work well for situation where researcher wants to know *why* and *how* some phenomenon happens. The purpose of this thesis is to find out how software startups work with their endeavours. Thematic interview also secures that all the necessary topics are addressed during the interview.

Structured interview would not work in this context since most startups don't follow any information system development methods but develop their software ad hoc and they tend to use different terminology than academia. Thematic interview leaves room for discussion and lets the interviewee describe his or hers own experiences without strict script. This will provide more comprehensive understanding of practices used in case startups.

Interviews were recorded and transcribed. Recording was be done by recording applications of mobile phone and laptop. Dual recording was used to ensure at least one working recording of the interview. Recordings were transcribed as soon as possible after the interview. Themes of interviews were derived from theoretical framework of this study. Chosen themes were alphas from the Essence framework. Frame of the interviews can be found from appendix 2.

Themes for interviews in this research were the alphas from the Essence kernel: Opportunity, Stakeholders, Requirements, Software Systems, Team, Work and Way of working. These themes are introduced in section 4. Interviews were conducted during between April 2018 and September 2018. Interviews were 35-85 minutes long.

Five interviews were conducted by the researcher himself and eight were obtained from the earlier study conducted by different team of researchers. These eight interviews were conducted in the similar manner. Thematic interview process was used and they were transcribed shortly after the interviews. These interviews were also 35-90 minutes long.

5.2 Participants

Runeson et al. (2008) suggest that the number of cases and criteria of case selection should be defined early on in the process. The criteria for these cases in research process were defined as following:

1. The case is a software startup. See section 2.1 for definition.
2. The unit of analysis is someone who has extensive understanding about the practices used in a startup, such as a founder or CEO.
3. Availability to the researcher.

Sampling was done by selecting cases available to the researcher that match sampling criteria above. Patton (2002) has introduced eight different sampling

strategies for case study. Strategy applied in this study is a mix of typical case sampling and sampling by convenience according to the Patton's descriptions. Selected cases were typical early stage software startups available for the researcher.

Holistic approach for this study was chosen considering the nature of software startups and underlying phenomenon of the study. Also no logical subunits can be identified from the cases. In the holistic approach of case study only one unit of analysis is studied from the case compared to embedded approach where multiple units of analysis are considered inside a case. (Yin, 2013.)

Myers and Newman (2007) suggest that so-called elite bias could be one of the problems for qualitative interviews. This means interviewing only certain high status people inside a case therefore failing to understand the broader situation. This bias is easily avoided while studying startups since startups usually have a small team and flat organization (Paternoster et al., 2014.) which naturally eliminates the elite bias. The multiple-case study will consist of several separate holistic cases and the cases should be conducted in consistent manner in order to obtain ecological validity.

Usually in multiple-case study adding of new cases stops when theoretical saturation is reached. Theoretical saturation means the point at which incremental learning is minimal while researcher has seen observed phenomena before. (Pare, 2004.) In this thesis cases were chosen at the same time and planned to extend to new cases if needed after the primary five cases. Eight already conducted and transcribed interviews were added after the primary five interviews. Whit these additional interviews saturation was reached.

Cases studied were early stage startups from Finland. The interviews were conducted in finish which was the language of all of the participants and interviewer himself for the minimum risk of misunderstanding between interviewee and interviewer. Eight external interviews were conducted in English since some of the interviewers and interviewees were not Finnish persons.

Interviews with startups 1, 4 and 5 were done with one person of the startup attending the interview. Interview with startups 3 had two persons from the startup participating and with startup 2 three people participated in the interview. All of the participants were either CEOs or founders of the startup. The external interviews were conducted with only one participant from every startup.

The external empirical material originally included interviews of eleven startups. Some of the companies didn't match the definition of software startup used in this theses and were excluded from the analysis resulting in the eight additional interviews. For example these startups were producing only physical products or they were new companies founded by engineers who made customer projects as consulting and didn't have any own software they were developing or selling. All of the studied cases are presented in the following table 10.

TABLE 10 Studied Cases

Case	Company size (Persons)	Company domain
#1	6	Software / Physical product
#2	5	Software
#3	3	Software / Physical product
#4	5	Software
#5	7	Software / Consulting
#6	3	Software / Physical product
#7	8	Software
#8	12	Software
#9	6	Software
#10	5	Software
#11	85	Software / Physical product
#12	5	Software / Physical product
#13	6	Software

As can be seen two of the startups had already been discontinued and one of the companies had matured. These companies focused on their interviews on startup phases of the company rather than their current situation.

5.3 Data Analysis

The goal of qualitative data analysis is to develop an understanding or interpretation of the phenomenon. This is done by an iterative process starting by developing an initial understanding of the setting. The next step is testing and modifying the initial understanding through cycles of additional data collection and analysis until an adequately coherent interpretation is reached. (Kaplan & Maxwell, 2005.) Coding and categorizing strategy were used for analysing the transcripts of the interviews in this thesis.

Data analysis in qualitative research can be done through various approaches. However qualitative data analysis always uses some basic principles: extract data, data coding and forming an interpretation. There is not one universal right or superior technique to execute data analysis. Kvale (1996) has introduced different techniques for data analysis. Data analysis starts by reading the text. Reading cannot be passive but it must be done to comprehend. (Dey, 2003.) One way to conduct a data analysis is through thematic synthesis (Cruzes et al., 2011). They suggest 5 steps for data analysis: extract data, code data, translate code into themes, create a model of higher-order themes and at last assess the trustworthiness of the synthesis.

Data collection, data analysis, interpretation, and even research design are linked and dependable on each other. Kaplan and Maxwell (2005) have identified four basic techniques of qualitative data analysis: coding, analytical memos, displays, and contextual and narrative analysis. They can be used separately or

in combination to identify themes, develop categories, and explore similarities and differences in the data.

Coding in qualitative research is done by selecting particular segments of data and sorting these into categories that facilitate insight, comparison, and the development of theory. Most of the categories are developed inductively by the researcher during the analysis while some categories might rise from the existing theory or prior knowledge. Some categories might also be taken from the language and conceptual structure of the people studied. The most important aspect of qualitative coding is that it is grounded in the data. (Kaplan & Maxwell, 2005.)

Researchers can use memos as a tool alongside direct field notes and transcripts. The most important use of memos is to get ideas down on a paper. Maxwell and Kaplan (2005) suggest memos to be an important analysis technique since they help researchers to convert their perceptions and thoughts into a visible form allowing reflection and further processing.

The analysis of the empirical material in this thesis was conducted following the thematic synthesis (Cruzes et al., 2011). The material was first transcribed. The external interviews were transcribed by the original researchers. Second the material was read through thoroughly and some notes were made from the material. After this step the interviews were coded and the codes were categorized under different themes. The original themes were then fitted to the theoretical framework of this thesis and those themes that didn't fit under the categories of the framework were analysed by creating additions to the original theoretical framework.

5.4 Research Validity and Reliability

In qualitative research researcher is the instrument for collecting and analysing data. This means that with the same research question different researcher may collect different data or interpret the same data differently and all researcher's biases, interests, perceptions, observations and knowledge play a role in the research. As a result qualitative research is always to some extent subjective. To address these issues researchers should acknowledge their role as research instruments by making it an explicit part of data collection, analysis, and reporting. (Kaplan & Maxwell, 2005.)

In qualitative research it is important to include reports of researchers' background to participants and research community for evaluation of potential influence in research results. Researchers must also recognize how their personal experiences and theoretical tendency influence their choice of evaluation questions, data, and interpretation. As a result researchers have to constantly doubt every observation and every interpretation to help avoid being blinded or misdirected by how they influence the study themselves. (Kaplan & Maxwell, 2005.)

Kaplan and Maxwell (2005) suggest that reliability is usually weaker in qualitative that in quantitative studies but validity is stronger because of flexibility and individual judgment. These issues mean closer attention to meaning, context, and processes while making researchers less likely to ask the wrong questions or overlook or exclude important data (Kirk & Miller, 1986).

Kaplan and Maxwell (2005) have identified strategies to ensure validity in qualitative research. They discussed five following strategies:

- Collecting rich data
- Paying attention to puzzles
- Triangulation
- Feedback or member checking
- Searching for discrepant evidence and negative cases

Rich data is data that is detailed and varied enough for providing a comprehensive description of what is going on. Collection of rich data provides a basis for developing theories rather than only supporting researchers' own prejudices and expectations. (Kaplan & Maxwell, 2005.)

Paying attention to puzzles deals with the underlying assumption of qualitative research that things make sense. Usually things make sense only to the people involved. If researcher cannot understand how situation makes sense he or she has not yet achieved an adequate interpretation. This may result from not having collected enough data or approaching the problem from the wrong angle or theoretical framework. As a result researcher must focus on resolving puzzles to develop a valid interpretation. (Kaplan & Maxwell, 2005.)

In qualitative research data is typically collected from a set of individuals and settings. Multiple sources and methods affect the robustness of results by allowing findings to be strengthened by cross-validation or in other words triangulation. If different kinds of data from different sources are found congruent the results have stronger credibility than basing only on one method and source. (Kaplan & Maxwell, 2005; Jick, 1979; Yin, 2013;)

Feedback and member checking mean systematically gathering feedback about conclusions made by researcher from others who are familiar with the setting. Kaplan and Maxwell (2005) consider this to be the most important way of ruling out possible misinterpretations. This is also an important way to identify researcher's biases. (Kaplan & Maxwell, 2005.)

Identifying and analysing discrepant data and negative cases plays important part in validity testing in qualitative research. Researchers might feel strong pressure to ignore data that does not support prior theories or conclusions they are making. But examining of both discrepant and supporting data is important for the research to point out possible defects. Good solution is to include reporting of the discrepant evidence and allowing readers to form their own conclusions. (Kaplan & Maxwell, 2005.)

5.5 Research Ethics

All researched companies will be informed about their rights to be anonymous and possibility to read through the thesis before publication. Interviewees have also a right to quit the interview anytime they wish. The research is conducted following the guidelines of The Finnish Advisory Board on Research Integrity (Varantola et al., 2013).

6 EMPIRICAL RESULTS

In this section results of the study are presented based on the empirical material. The results are categorized following the theoretical framework of this study. Most important findings are presented as primary empirical conclusions (PEC). PECs are further discussed in the section 7. Empirical conclusions (EC) are also used as a tool to point at certain empirical findings from the study's material that form primary empirical conclusions.

The results are presented following the empirical framework. Categories according to which the results are divided are alphas from the Essence framework. Which case has stated certain information has been introduced as a startup or case following the identifying number of said case. Case startups and identifying numbers are presented in the table 10 in the section 5.2. It is notable that startups interviewed didn't discuss all of the listed practices so there might be more supporting or conflicting startups.

Thirteen new practices were found from the empirical material that were absent from the original list from Dande et al. (2014). These added practices were those that at least two different cases clearly stated. Some of the startups also described other new practices. These were excluded from the study if they were only address by one startup. These new practices are presented in the following table 11.

TABLE 11 New Practices

Identifier	Practice	Description	Alpha
P64	Study subjects that support startup	Studying while working on a startup gains competence in the team without growing in personnel.	Team
P65	Attend startup events	Startup events provide opportunity for feedback from experts and allows you to meet potential investors	N/A
P66	Create an MVP in the beginning	MVP helps you to focus on the most important features in the beginning	Requirements
P67	Test features with customers	Testing features with real customers gets you the best feedback	Requirements
P68	Get advisors	Experienced professionals or investors can help startup to grow in advisor or mentor role	Stakeholders
P69	Use efficient tools to plan your business model	Business model canvas, pitch deck etc. help you to focus your business idea and are easy to change if needed	N/A
P70	Test different tools	Start with tools team is familiar with and test different ones to find those that work the best for you	Way of Working
P71	Conduct market research	Research the markets and competitors to focus your idea and to find your unique value proposition.	N/A
P72	Have frequent meetings with whole team	Use meetings to organize and plan your work at least once a week	Way of Working
P73	Don't have strict roles	Let the team co-operate in all of the tasks	way of working
P74	Create prototype	Create prototype to validate your product or features	Requirements
P75	Use efficient communication tools	Use tools that allow natural communication inside the team when not working in the same space.	Way of Working
P76	Prioritize features	Choose which features are needed now and plan others for future releases.	Requirements

In the following subsections the practices found by Dande et al. (2014) are presented with additional information of which cases studied in this thesis support or conflict with those practices. Also the practices presented above are added to the lists.

PEC 1: The list of practices found by Dande et al. (2014) is not comprehensive enough. Startups have more working practices than presented in their study.

6.1 Opportunity

The Opportunity alpha deals with the circumstances that make software development or changes appropriate and the team understanding of the stakeholder's needs. There were no cases conflicting with any of the practices considering the opportunity alpha. No new practices were derived from the cases considering the opportunity alpha neither. The practices of opportunity alpha are presented in the table 12 below.

TABLE 12 Results of Practices Related to Opportunity

Identifier	Practice	Description	Alpha	Case supporting	Case conflicting
P1	Focus your product	Focus on the most potential customer segment. Be prepared to change the focus	Opportunity	1,2,6,7,8,9,11,12,13	
P2	Find your value proposition and stick to it on all levels	Create a valid value proposition. Discuss with experts from strategic and operational levels at customer's organization.	Opportunity	9,13	
P4	Focus on goals, whys	Find real motivations behind customers' wishes by asking why they want something rather than just what they want. This way you understand customers' needs deeper and can address them in other situations as well.	Opportunity	9	
P18	Validate that your product sells	Validate your idea before starting development or try to get a few customers before you start developing.	Opportunity	1,2,4,5,7,8,11	
P20	Form deep relations with first customers to really understand their needs	To understand the customers and the business develop as deep relations as you can with the first customers.	Opportunity	1,6,9,11,13	
P33	In the development of customer solutions, find a unique value proposition in your way of acting	Find the way of acting that differs from your competitors. For example super-fast or people centric.	Opportunity	1,2,3,5,6,8,9	
P34	Follow communities	Everyone should follow communities to know what is happening and to find new values for customer.	Opportunity	1,2	

Practice P1 was especially supported practice in this category alongside P18, P20 and P33. These practices deal with focusing the product to suit a certain customer segment and validating that the focus and the product are set right. Startup 13 discussed practice P13 as following.

> We have a second customer and they are operating in a completely different business but still they are really similar. Because our main target in those customers, it's on their technical services side. On their maintenance, repair and overhaul departments. Even if they are maintaining and operating different things the maintenance processes, they are similar and have similar needs. This kind of maintenance services are what we can improve. (Case #13)

Practice P33 deals with the idea to create something unique that is missing from the market or to have a company that creates something in a unique way. Seven of the startups studied discussed this idea and thought their product or company is doing this. This was not confirmed by any further analysis of their competitors by researcher as startup's own insight of this was sufficient for this study. Even if their insight was proven wrong their effort to create something unique for the market means that they follow this practice. Startup 2 discussed this effort to find unique value proposition as following:

> Yes there was some competitors making similar products but they were still a bit different. We had a completely new point of view on these kinds of services. Most of the

> competitors we found were really in general. They offered something for everyone and we had focused on this only. We were also only ones in the market that but companies and customers to the same platform. All of existing services were only for consumers. (Case #2)

Seven of the studied startups discussed how important it is to form deep relationships with the first customers in their opinion. This idea of involving customers to validate and improve the business idea and product lines with the practice P18 found by Dande et al. (2014). Startup 1 discussed their relationship with the first customer as following:

> We had talks with some customers before we started the development and they gave us green light that this is something they want to use. We also got some ideas to which kind of companies could use our product and got some new ideas. We found a couple customers with whom we could test the product as soon as we had one. (Case #1)

PEC 2: The most important practices regarding the opportunity alpha are related to the customer relationship and focus of the product.

Interestingly it seems that startups subject to this study hadn't followed the practice P4 or didn't mention that aspect in the interviews. It seems that they were focused on what their customers want or need and not so much on the aspect of why people want something. It seems that startups ware basing their product to suit a certain need or want of the customers but weren't so eager on finding the deeper meanings of these wants to enhance their product or service according to the deeper needs of customers. They used some tools and market research to validate customers wants. These tools or techniques are further described in the section 6.3.

6.2 Stakeholders

Stakeholders provide the opportunity for startup to produce software and build business. The most notable stakeholders for usual case of a startup are investors and customers. Investors provide resources for startup to keep up their work and customers are the goal for whose needs startup's product aims to fulfil. This means that together they provide the opportunity for a startup. Additional practice concerning stakeholders was added after the analysis of empirical material. This new practice P68 concerns stakeholder group called advisors. These advisors were experienced entrepreneurs or professionals who helped startup to achieve their goal by mentoring the team. In some cases advisors were also startups investors.

None of the cases studied had practices conflicting with Dande et al.'s (2014) practices regarding the stakeholders alpha. Neither did they support a lot of those practices except the newly added P68. Only startups 1 and 2 backed up

practice P35 and startup 6 backed up practice P24. Practices from this category and cases supporting them are presented in the table 13.

TABLE 13 Results of Practices Related to Stakeholders

Identifier	Practice	Description	Alpha	Case supporting	Case conflicting
P24	Keep customer communications simple and natural	A startup needs quick and good feedback from customers for development decisions. Try to encourage direct contacts by email or through integrated feedback mechanisms.	Stakeholders	6	
P32	Showing alternatives is the highest proof of expertise	Finding different alternatives for a solution is expertise. Explore alternatives to find a good solution.	Stakeholders		
P35	Share ideas and get more back	Sharing ideas will help you get valuable feedback.	Stakeholders	1,2	
P68	Get advisors	Experienced professionals or investors can help startup to grow in advisor or mentor role	Stakeholders	1,4,5,6,8,9	

PEC 3: Startups usually use advisors or mentors to guide their work in the early stages. This is one of the major stakeholder groups along the customers and investors.

The practice P35 concerns startups sharing their ideas. This sharing was done and discussed by startups 1 and 2. They had shared their ideas in the startup events or communities with other startup teams and coaches to get new ideas for their work. They didn't indicate that they had been afraid of others stealing their ideas but rather find it useful to get feedback from outside of the relatively small team. Startup 2 discussed their idea sharing as following:

> We went to pitching competition and shared our ideas with investors there and got a lot of feedback. We also had working spaces in Jyväskylän Yritystehdas where we attended events with our idea and talked with investors and such. From them we got validation that this is idea is working. (Case #2)

6.3 Requirements

Requirements scope the work. They guide what team does and what the software system should include. In the category of requirements alpha most of the practices were well supported. For example practice P52 was confirmed by startups 1, 2, 3, 4, 5, 6, 7, 12 and 13. Only practice that had conflict with the results was P3. P3 was described by Dande et al. (2014) that startups should present their product as facilitating rather than competing to the competitors. Startups 1, 2 and 6 stated in the interviews that they had been focusing on how to compete with other products on the market. None of the cases studied discussed about the opportunity to facilitate products of competitors. It's notable that this practice is highly dependent on a company's product and not so much

on a company's stage. For example startup 6 stated their intensions on the market as following.

> It is business to business market. It's by no means consumer product. What our idea in this business was to make it cheaper. Make it simply cheaper and technically better than the existing competitor to get the market. (Case #6)

All of the practices concerning requirements alpha are listed in the table 14 below. Four new practices were added to this category after the interviews (P66, P67, P74 and P76). Practice P74 was added as a new practice even though Dande et al.'s (2014) practice P5 also suggests creating the prototype. Dande et al. focus on UX in their practice involving the prototype. They suggest that prototype should be used to test UX. After the analysis of empirical material it was clear that startups use prototype also for testing other features than just UX. This was the reason for the new practice to be added to the list.

Practices P52 and P74 were also particularly well supported practices. It seems that startups are focusing highly on their core product and validating that it's working and what their customers want. This results in a very focused concept of business and product. It seems that main tool to validate the business idea and product as well as its features is a prototype. Prototypes were widely used to test the preliminary product and its different features with pilot or potential customers. Startup 3 discussed this practice as following.

> We made a proof of concept kind of prototype. To test which kinds of technologies could be used for this. We found quite many and tested them. We also chose some features we thought would be good and made a very basic version of it to see how it's working and how the data flows and such. Basically we tested that it can be done and what features we need and now we just have to focus on creating a first real version of it. (Case #3)

TABLE 14 Results of Practices Related to Requirements

Identifier	Practice	Description	Alpha	Case supporting	Case conflicting
P3	Present the product as facilitating rather than competing to the competitors	Develop a product that can co-operate rather than compete with competitors	Requirements		1,2,6
P5	Use proven UX methods	Use proven UX development methods from the beginning. Validate ideas quickly by using prototypes.	Requirements	12	
P10	Design and conduct experiments to find out about user preferences	Use experiments and communication with user to determinate in which directions product should be developed.	Requirements	1,2,4,6,9,12,13	
P21	Use planning tools that really show value provided to customer	Choose tools that allow mapping the value customer gets from what is done and planned.	Requirements	2	
P51	Anything goes in product planning	Startup needs to figure out new features, system concepts and new projects.	Requirements	1,2,11	
P52	To minimize problems with changes and variations, develop a very focused concept	Develop a validated and focused concept to minimize risks with changes. Be still ready to do changes if needed.	Requirements	1,2,3,4,5,6,7,12,13	
P53	Develop only what is needed now	Be efficient by developing only what is needed now.	Requirements	1,2,3,12	
P66	Create an MVP in the beginning	MVP helps you to focus on the most important features in the beginning	Requirements	1,2,4,13	
P67	Test features with customers	Testing features with real customers gets you the best feedback	Requirements	1,3,4,5,6,7,8,9,11	
P74	Create prototype	Create prototype to validate your product or features	Requirements	1,2,3,4,5,6,9,12	
P76	Prioritize features	Choose which features are needed now and plan others for future releases.	Requirements	1,2,3,9,11	

Newly added practice P76 describes the practice of prioritizing product features. This is closely connected to the practice P51 which suggest that startup team should always be open for new ideas for their product but as seen in the practice P52 concept should be focused. This leads to the need of prioritizing features in order to keep the concept focused and team's focus on most important features in the beginning. Startup 2 discussed this process of prioritizing the features as following:

> Basically we had thought what kind of functionalities we will create. After that we sat down and gave each functionality a priority and made a list arranged by those priorities and made groups of functionalities to form logic entities. Then we started working on those groups in order to make most important ones in the beginning and so on. (Case #2)

PEC 4: Software startup's most used practices regarding the requirements are focused on prioritizing the product features that are the most needed now and focusing the concept. These both are tested and validated.

6.4 Software system

Software system category addresses the product a startup is selling to its customers. It can be either a software system or a mix of a software system and physical product or another service. Practices regarding the software system alpha were somewhat divided. Practice P7 had eight startups supporting the practice and two cases against it. P7 indicates that a startup should have a single product rather than per customer variants. This is something that is also domain dependent and that might cause some startups to oppose it if their product just cannot be as general as mediocre startup's. None of the other practices were conflicting by any of the startups studied.

Practices P23, P54 and P57 weren't discussed by startups in the interviews. This doesn't necessarily mean that they are absent in the startups' endeavours but just not mentioned in the interview. Interviewed persons from the startups didn't have the main responsibility of programming which might affect that they didn't provide information on such things as release cycles. All the practices concerning the software system alpha are presented in the table 15 below. No new practices were added to this category.

TABLE 15 Results of Practices Related to Software System

Identifier	Practice	Description	Alpha	Case supporting	Case conflicting
P7	Have a single product, no per customer variants	Have a modular and flexible single product rather than multiple per customer variants.	Software System	1,2,3,5,7,8, 11,12	6,13
P8	Restrict the number of platforms that your product works on	Make business decisions on what platforms you want to support. Focus on the most important ones. For example the most used browsers and operating systems.	Software System	1,2,3,4,7,12	
P14	Anyone can release and stop release	Allow anyone to make a release or stop it. Fast releases allow quick feedback from users.	Software System	2	
P23	Adapt your release cycles to the culture of your users	Depending on your customers choose how fast releases are and how much can be changed at once.	Software System		
P54	Make features easy to remove	Use techniques and architecture that make features easy to remove if needed.	Software System		
P55	Use extendable product architecture	Use architecture and techniques that allow to extent design easily.	Software System	1,2,3,9,11	
P57	Bughunt	During fast development of new features arrange days for bughunt. Make bughunt fun occasion when everyone is searching for bugs.	Software System		
P58	Test APIs automatically, UIs manually	APIs can be tested by tools that are easy to find and cheap. Test UI manually in the beginning.	Software System	2,13	
P59	Use generic, non-proprietary technologies	Use platform independent technologies to avoid re-implementing features.	Software System	2,7	
P60	Create a solid platform	Keep scaling in mind while developing a platform.	Software System	3,8,9,11	

The most endorsed practices concerning the software system alpha were P7, P8, P55 and P60. P7 suggests that a startup should have a single product with pos-

sibility to modify it slightly. Most of the startups wee following this idea as startup 11 discussed their opinion

> It was IOT product. The hardware is a bit expensive but then the software configuration should be so easy that anyone can do it. This way we have one hardware we developed for this particular use case but it can be used to all the different use cases just by configuring it differently. (Case #11)

The practice P8 discusses the potential of focusing on a single platform for the product in the market entry phase. Originally Dande et al. (2014) thought that this should be chosen according to which platform is the most used in the market. Some of the startups made their choice also according to which platform is easiest or cheapest for them to start with.

P60 is closely connected to the practices P7, P8, P55 and P59. It suggests that the product should be developed with scaling in mind. These other practices also implicate the same or guide the product towards scalable result. P55 implies that product architecture should be scalable and P60 confirms that with the aspect of platform.

6.5 Work

The Essence describes work as software system development. As startup's unique characteristic is that a small team does both the software development and business model development a work alpha in startup circumstances should include work towards healthy business model and revenue streams or an exit for owners. With software startups software development and business model development cannot be separated. This simultaneous work on product and business model was noticed during the analysis of empirical material.

Work alpha related practices were least found from the Dande et al.'s (2014) study compared to other alphas alongside the stakeholders alpha. These two were also the least backed up by the results of this study. None of the startups conflicted with any of these practices. All of the practices were slightly supported by the interviews but not as clearly as some of the other categories' practices. There were no new practices that emerged from the empirical material that fit under this category.

Practices concerning the work alpha are presented in the following table 16. As can be seen from the table these practices were not well supported by this study. It's notable that most of the persons interviewed didn't have main responsible on programming in their startup which can affect that they didn't talk a so much about these practices or were unable to answer in the detail when asked. This means that these practices might have be found more from the startups if the interviewed person had been more programming oriented person.

TABLE 16 Results of Practices Related to Work

Identifier	Practice	Description	Alpha	Case supporting	Case conflicting
P44	Tailored gates and done criteria	Process phases leading to something being done or assessed or accepted should reflect the overall process and business.	Work	8	
P48	Fail fast, stop and fix	Allow developers to do things quickly and freely and stop if something goes wrong. They will then fix the problem and process in the team.	Work	1	
P62	Use the most efficient programming languages and platforms	With a small team choose the most efficient programming languages and development platforms.	Work	2,3,7	

The most supported practice in this category was P62 that was found from the work of three studied startups. They described efficient as efficient for themselves as chosen languages and development platforms were known to them before and fastest for them to start working with. They didn't discuss overall efficiency of different programming languages in general. For example startup 7 discussed how they chose development platforms as following

> We just made a website, standard website of course. Completely coded by ourselves because boys didn't believe anything else but what they can code. So we coded a standard HTML, CSS, PHP platform and we were in a super hurry. (Case #7)

6.6 Team

Team is the group of people who work on a startup. They can be founders, owners or paid workers of the startup. From the table 17 can be seen team sizes of all the startups subject to this study. As can be seen startup 11 had already 85 people working for them. They had already grown out of the startup phase which is the reason for so high number of workers. In the interview they discussed their work in the early stages but didn't mention the original team size in numbers or when the team had had its biggest growth. One new practice (P64) was added from the empirical material to this category. Startups 1, 2, 3, 4, 8 and 9 discussed the practice P64 as their team members or some of them were students while working on a startup and had benefitted from their studies.

Interviews supported well all of the practices categorized to the team alpha despite a few conflicting interviews. Startup's original teams form from were different basis depending on where and by whom the idea of the startup comes from. It's quite natural that all of the practices cannot suit every startup since the teams experience for example might not always be optimal if the idea comes from the student group for example. This might also effect on how fast and on what basis team growth is based on.

Practice P41 was one of the practices that were highly supported. It was clear from the empirical material that startups need to focus their recruits on certain competences they are lacking. Also P26 was clearly supported practice

which suggests having a flat organizational structure. For example startup 1 stated following:

> Even though I was the founder and I had 51% of the shares and I had the official decision making power we made all the decisions with democracy so that we follow a shared vision and nobody starts to make solo decisions or bossing around. We had active conversations and we made sure that we don't have too strict roles. (Case #1)

As seen from the quote startups feel that flat organization suits small and cooperative teams well. Some of the other startups also indicated that their organizational structure is quite flat but didn't clearly state that or describe structure in detail.

TABLE 17 Results of Practices Related to Team

Identifier	Practice	Description	Alpha	Case supporting	Case conflicting
P26	Flat organization	In flat organization people are committed to a common good and communications are easy as they don't require intermediates.	Team	1,2,3,5,9	
P27	Consider career expectations of good people	Keep team happy by offering opportunities to build up their skills. They can raise their market value as an insurance for the case that startup fails.	Team	4,9	
P28	Don't grow in personnel	If you don't need more resources or competence don't grow in personnel.	Team	1,2,3,12	
P29	Bind key people	Most important people should be shareholders, partners or founders because critical information is easily lost.	Team	2,3,6,7	
P36	Small co-located teams	Small teams with scarce resources need good communication to survive. Speaking in the same room is the most effective way to communicate.	Team	1,2,3,4,5,6	12
P37	Have multi-skilled developers	Startups have usually small teams, yet there are lots of different things to do. Multi-skilled developers are needed to address all the issues in startup without growing in personnel.	Team	1,2,3,12	
P38	Keep teams stable in growth mode	While growing as a company try to keep teams and individual roles stable.	Team	1,2,3,4,6,7, 13	9
P40	Sharing competence in team	In team everyone has slightly different expertise. Since startups need skilled developers sharing competence inside the team is necessary.	Team	4,5	
P41	Start with a competence focus and expand as needed	In the beginning focus on specific competence with a small group of people. Expand team and competences later.	Team	1,2,3,4,6,8, 9,13	
P42	Start with small and experienced team and expand as needed	Start with small and experienced team that has efficient ways to communicate. Anticipating all needed skills beforehand is hard.	Team	1,2,3,4,7,8, 12,13	1,2,3
P64	Study subjects that support startup	Studying while working on a startup gains competence in the team without growing in personnel.	Team	1,2,3,4,8,9	

The practice P42 was conflicted by the interviews of startups 1, 2 and 3. This doesn't mean that they didn't want to start with a small and experienced team but they rather started the startup by group of friends who were students at the moment. All of them described their biggest challenge as their inexperience in the field. They and a few other studied startups were fixing the lack of compe-

tence by studying in the university subjects that are helpful for their endeavour. This simultaneously studying and working on a startup was found in so many cases that a new practice was derived from that.

Seven of the studied startups indicated that they wanted to keep the team stable in the early stages of the startup and recruited new members only when absolutely necessary. This practice was also noticed by Dande et al. (2014) in their study and presented in this thesis as the practice P38. One startup was doing the opposite as they changed the team many times during the first two years of the company.

The practice p36 implies that startups work the most efficiently with a small co-located team. Six of the studied startups indicated the same and a couple discussed during the interview that a chance to work in the same space could have helped them succeed. Only one startup discussed how they had managed to create an efficiently working environment with the team that only met in the same room occasionally.

PEC 5: Flat and self-organizing teams are the most used organization structure by software startups.

6.7 Way of Working

Way of working alpha in the Essence includes practices and tools a team uses to support their work. It also includes processes how a team improves their working practices. Basic tools used in the cases were quite similar. They included instant messaging software such as Slack and Whatsapp for daily communication and Google Drive for file sharing for example.

The practices concerning the way of working alpha had some dispersion. Only one case conflicted with one practice but some of the practices were not clearly supported either. Still most of the practices had clear support from the empirical material. Four new practices were also found from this category (P70, P72, P73 and P75). The practices from this category are presented in the table 18.

TABLE 18 Results of Practices Related to Way of Working

Identifier	Practice	Description	Alpha	Case supporting	Case conflicting
P9	Use enabling specifications	Enable specification to guide work efficiently. Let team work independently without constant intervene from the owner or customer.	Way of Working	1,2,3	
P15	Create the development culture before processes	In the beginning develop a culture that supports what you want to be. Processes are likely to change as company evolves so focus first on building the culture that fits your goals and future processes.	Way of Working	1,8,11	
P39	Let teams self-select	Team should be allowed to self-organize.	Way of Working	1,2,3,5,8	
P43	Have different processes for different goals	Choose different practices for different tasks if needed.	Way of Working		
P45	Time process improvements right	Improve and change processes only when it is absolutely needed. At some point of the growth startup might need to change its preliminary processes.	Way of Working	3	
P46	Find the overall development approach that fits your company and its business	Find the best approach for your business. Don't follow latest trends if it's not best fit for you.	Way of Working		
P47	Tailor common agile practices for your culture and needs	Most textbook practices are highly general. Tailor them to fit your needs and culture.	Way of Working	1,2,3,4,6,7,8, 13	
P49	Move fast and break things	Prefer culture with fast development and where failing is acceptable.	Way of Working	4,7	
P50	Forget Software Engineering	Software development may be ad hoc and unorganized if it us good enough with the physical product.	Way of Working	1	
P61	Choose scalable technologies	Favour development techniques that scale easily.	Way of Working	2,3,9,11	
P63	Start with familiar technologies and processes	Save the time of learning new technologies and processes by using those that team is familiar with.	Way of Working	1,2,3,7	
P70	Test different tools	Start with tools team is familiar with and test different ones to find those that work the best for you	Way of Working	1,3	
P72	Have frequent meetings with whole team	Use meetings to organize and plan your work at least once a week	Way of Working	1,2,3,4,5,8,12	
P73	Don't have strict roles	Let the team co-operate in all of the tasks	way of working	1,2,3	9
P75	Use efficient communication tools	Use tools that allow natural communication inside the team when not working in the same space.	Way of Working	2,3,5	

The most supported practice concerning way of working alpha was P47 which suggest tailoring common agile methods to suit a particular startup's work. Most of the cases had chosen some agile methods as basis of their work but didn't apply them in the textbook way but rather tailoring them to suit their team and work. For example the startup 6 discussed this as following.

> We are sitting in the same room and we are, having a kind of agile, we have had a kind of an agile process which is very practice-oriented in such a way that it fits to that, I would say that we have done plenty of agile principles. We have followed plenty of agile principles without much formalism. We divide the work in such a way like a typical scrum team is doing. You are doing this and I am doing this and then we check how it works. We have a kind of continuous integration and continuous testing. It is not automatic testing is not automatic because of the embedded na-

> ture of the product it is difficult to make any automatic software testing on digital signal processing. Then we have done refactoring and even some parts programming and such stuff. But we haven't had let's say that the paradigm has been kind of agile, but without formalism. (Case #6)

The practice P72 was also well supported. It was added after the analysis of the empirical material. It suggests that frequent meetings with the whole team help to achieve startups goals. The most used schedule for meetings was weekly meetings. Also the programming part of the team usually had own daily meetings.
The practice P39 was also well supported in this category. It is closely connected to the practice P26 as flat organization and self-organizing team seems to be linked since in the flat organization there isn't usually supervisor to guide the work or assign tasks. The startup 2 discussed their views of self-organizing team as following.

> So we had weekly meetings every Monday and in those meetings we thought what should be done next and who wants to focus on what and so on. Like he said the Monday meeting was kind of round up meeting. Then in between the meetings everyone did what was agreed as they wanted and then we met again to check the progress. (Case #2)

PEC 6: Startups tend to tailor common agile methods to form their own working practices.

6.8 Other Practices

All of the startup practices used as basis of this thesis or ones that were derived from the empirical material don't fit into existing alphas from the Essence kernel. Most of the practices that didn't fit for any of the alphas dealt with clearly business aspects. They considered such things as marketing, business model development or funding. The Essence was developed as a tool for the software system development which causes this problem in the software startups case since they can't only develop the software system but also business model in the close co-operation. These practices that don't fit under any of the existing alphas are presented in the following table 19.

TABLE 19 Results of Other Practices

Identifier	Practice	Description	Case supporting	Case conflicting
P6	Do something spectacular	Create WOW effects and feelings to the customer to stand out in the competition.		
P11	Use tools to collect data about user behaviour	Use data to acknowledge user behaviour and choose best marketing channels.	1,2,7	
P12	Make your idea into a product	Turn your ideas into products rather than projects. Projects are not easily scaled.	1,2,3,4,5,6,7,8, 12,13	11
P13	Outsource your growth	Use outsourcing to keep your focus on the product.	5,9,11,12,13	3
P16	Get venture capital and push your product	Try to get your product profitable fast with venture capital rather than develop it slowly in silence with low resources.	1,2,4,5,8,9	3
P17	Fund it yourself	Getting funding with proof of concept is not easy. Fund first yourself and get investment later.	1,2,3,7,9	
P19	Focus early on those people who will give you income in the long run	Try to get your business model running from the start, even in small scale. Focus on paying customers to ensure that the company is profitable.	5,6,7,8,11,13	
P22	Start locally grow globally	Target local customers in the beginning but make all decisions considering the global growth.	1,2,3,6,7,8,9,13	
P25	Help customers create a great showcase for you with support	The first customers can provide a visible showcase to attract other customers.	1,6,8,9	
P30	Form partnerships and bonds with other startups	Focus on developing your product and on your core business. For other issues find partnering startup. Startups are usually keen to co-operate.	1,3,4,5,13	
P31	Make your own strength as a "brand"	All startups should have exceptional skills or product. Turn this strength into a brand in the market.	8	
P56	Only use reliable metrics	Use reliable metrics to validate things. Wrong metrics might do harm for validation.	5,6,7	
P65	Attend startup events	Startup events provide opportunity for feedback from experts and allows you to meet potential investors	1,2,3,4,8	
P69	Use efficient tools to plan your business model	Business model canvas, pitch deck etc. help you to focus your business idea and are easy to change if needed	1,2,3	
P71	Conduct market research	Research the markets and competitors to focus your idea and to find your unique value proposition.	1,2,6,12	

The practices P6, P11, P25, P31 and P71 concern marketing activities. They imply practices startups could use for their marketing efforts and to find their place in the market. For example P25 describes the idea to gain first few customers as pilot customers with whom to form deep relationships and who can be used as reference customers in the marketing. They also help to test and validate ideas if the relationships are good and the customer is involved in the development. For example startup 6 discussed this following

> We have two kinds of customers. We have customers who had understanding of that phenomenon and we have customers who were unhappy and were not willing to cooperate. Now we have got the first order of that unfriendly customer too because we were able to show that customer we were piloting with. We could make them believe we can make it with that pilot. (Case #6)

EC: There is no current alpha that can be used to monitor the progress of marketing activities.

The practices P16 and P17 consider funding of the startup. They are essentially opposites of each other but don't necessarily mean that startup should choose one of them since they might be justified in the same startup in different states. Only startup 3 was marked as conflicting case since they stated that they didn't want to search for the external funding in the point they were. Even so they kept the option open that they might search investors later.

EC: There is no current alpha that can be used to monitor the progress of funding.

Others practices in this category dealt with overall business planning and business model development. For example practice P19 discusses that s startup should get their business running as soon as possible even in the small scale. This will help securing funding and validating that the idea works in the market. P13 suggest that sometimes outsourcing of some sector can help to focus on the core of the startup. For example this was done to outsource the production of physical product and focus on the software that the product uses.

EC: There is no current alpha that can be used to monitor the progress of business model development.

PEC 7: Current alphas don't take business aspects into consideration well enough to cover software startups' work.

6.9 Summary of Results

Dande et al.'s (2014) startup practices are widely supported by the results of this study. Some conflicts were found but they seem to be minor ones. Some new practices were also added due to incompleteness of the original list of the practices. Other new practices also emerged from the interviews but were excluded from this analysis since only one case startup discussed them. It also seems that the Essence kernel supports software startups work well but some modification might be needed. These recommended modifications are further discussed in the chapter 7.

The most conflicting practices were found under the team alpha where three practices had conflicting evidence from the empirical material. The most supported alpha was opportunity where all of the practices under that category were supported by at least one startup and none had conflicting evidence from the empiric material.

7 DISCUSSION

In this section the results based on the analysis will be discussed. The discussion is done through primary empirical conclusions founded in the section 6 and its subsections. Primary empirical conclusions (PECs) are listed below:

TABLE 20 Primary empirical conclusions

PEC	Description
#1	The list of practices found by Dande et al. (2014) is not comprehensive enough. Startups have a lot more working practices.
#2	The most important practices regarding the opportunity alpha are related to the customer relationship and focus of the product.
#3	Startups usually use advisors or mentors to guide their work in the early stages. This is one of the major stakeholder groups along the customers and investors.
#4	Software startup's most used practices regarding the requirements are focused on prioritizing the product features that are the most needed now and focusing the concept. These both are tested and validated.
#5	Flat and self-organizing teams are the most suitable structure for software startups.
#6	Startups tend to tailor common agile methods to form their own working practices.
#7	Current alphas don't take business aspects into consideration well enough to cover software startups' work.

7.1 Theoretical Implications

In this subsection the theoretical implications of this thesis are presented. The theoretical framework of this thesis is based on literature concerning the Es-

sence framework and software startup practices. For example Paternoster et al. (2014) have already studied software startup practices but this study goes to more detailed level of practices as they focused on higher level groups of practices on their results. Klotins et al. (2019) have also studied software startup practices following different categorization.

The basis for the software startup practices was Dande et al. (2014) study of software startup practices. These practices were validated by the other literature concerning the same issue. The theoretical framework was developed combining those practices with the Essence framework and categorizing the practices under the Essence alphas. Unterkalmsteiner et al. (2016) have created a research agenda regarding software startup studies. This thesis aimed to address software startups' practices in general level of their work not focusing solely on their software development work or business activities which expanded this study to concern multiple sections of their research agenda.

The opportunity alpha takes some business aspects into consideration since it focuses on the aspects that make software development endeavour feasible (Submitters, 2012). It was clear after the analysis of the empirical material that even though some business aspects can be seen in the opportunity alpha it is not comprehensive enough to describe startups work. It is understandable since the Essence framework was designed as a tool for software development projects and startups are more than just that. It seems that some new alphas need to be added to the Essence for it to support whole endeavour of software startup companies as stated in PEC 7.

Findings of the study conducted by Klotins et al. (2019) support this suggestion. They have concluded in their study that software startups software engineering and business development practices are tightly connected. Most of the Essence kernel's alphas seem to suit software startups' work satisfyingly as it is. This seems to originate from the fact that software development is the key aspect of the endeavours of these startups. In their study Klotins et al. (2018a) have pointed that most cases where software startups fail due to issues that first seem to concern business aspects originate in fact to software engineering processes. This indicates that business and software development practices of software startups should be presented in the same framework since they are not separable.

Alphas that should be added to the Essence are funding, marketing and business model. These are things that all startups have to address somehow. For example funding can be searched from investors outside the team or the startup could be funded by its founders or by the combination of these. No matter which approach is chosen it's necessary to monitor that source of funding has been found, funding is secured and the funding is adequate for them to enter the market with profitable business. Marketing and sales in the other hand are the main work by company to acquire its customers and if not considered in the company there's change for the startup to become a project that develops a prototype or a product but not profitable business. Business model should be developed and measured in deep co-operation with the actual product to make

sure that revenue streams are gained and startup provides income for its founders and employees.

The importance of the funding for startups has already been acknowledged in the literature (Chang, 2004). This aspect seems to be missing from the Essence framework and should be addressed in order to support the whole endeavour of software startup. Even though stakeholder alpha is concerning stakeholders that provide the opportunity for software endeavour and funding for the startup is provided by these stakeholders i.e. investors it doesn't take into consideration the funding itself but the relationships with the people who provides the funding. The search for investors and securing the funding is such major part of startups early stages that adding a new alpha considering funding should be considered. Submitters (2012) stated that the alphas are the most important aspects of software development endeavour whose progress should always be tracked. Funding is one of these things in a software startup case as it could be broken into states that can be progressed. For example potential investors found, talks with investors, primary funding secured, sufficient funds acquired for startup to run on own revenue streams. This process doesn't fit into the sates of Stakeholder alpha as they deal with more generalized states of identification and involving of stakeholders.

The importance of the Marketing is widely studied subject in the academia. For example Weerawardena (2003) has pointed out its importance for the companies' success. The startups are not exception on this as Crowne (2002) has stated. In the Essence framework opportunity addresses the situation that makes software development feasible (Submitters, 2012). This way marketing activities might be seen to fit under that category but it's reasonable to argue that since the opportunity alpha considers the circumstances that provide opportunity for software development the marketing activities should be extracted from the opportunity alpha and presented under their own alpha because marketing doesn't only provide opportunity but it also is vital part of making the product successful in the market. Klotins et al. (2018a) have pointed that difficulty to find potential customers and convert them into paying customers is one of the biggest challenges for software startups. Even superior products can be hard to sell if they are not marketed correctly and the market segment is chosen incorrect. Marketing is also closely linked up with business model development. Sales and marketing activities could be tied together under this new alpha.

Business model is seen as one of the most important aspects of early stage startups in the current literature (Lueg, Malinauskaite, & Marinova, 2014). Many startups design their preliminary business model using tools like business model canvas or lean canvas. As the Essence framework was designed to cover all the ubiquitous aspects of software development these kinds of business aspects are missing from it. In order to cover the ubiquitous aspects of software startups' work the business model development should be added as a new alpha. The business model and the business decisions on startup affect the requirements and opportunity as they deal with the issues such as will the

product be profitable for the company when it's on the market and what kind of technologies should be used to make profit with the product. It also effects on the marketing since the decisions like whether to start locally and grow later or to go straight to global markets change the way the marketing is done. The business model is also presented to the potential investors while searching the funding for the startup. This is so critical and broad part of the startups' work that it should be monitored and addressed as its own, not part of other existing alphas.

The list of the startup practices found by Dande et al. (2014) was supported well by the empirical material of this thesis but still thirteen new practices were added because of the minor limitations of the original list as stated in PEC 1. The most supported categories of practices were opportunity, requirements, team and way of working. Software system category was also quite well supported alongside with the practices that didn't fit into any existing alphas. From the thirteen new practices four fitted into the requirements alpha, one under the stakeholders, one to the team, four under the way of working alpha while three were considering business aspects that didn't fit under the existing Essence framework's alphas.

Overall the Essence framework seems to support the software startups' practices well. With additional alphas of marketing, funding and business model startup practices found by Dande et al. (2014) fit into the Essence framework. Some of the practices might be case-by-case practices since they are not ubiquitous on software startup but may differ by basis of startup and the fact of which kind of product or service they are offering. For example team related practices seem to differ with startups whose preliminary teams are found from different backgrounds. It was notable that most of the practices also fitted to the Ries (2011) lean startup method as most of the cases described their work to focus on only the necessary things at the moment and they tended to use the process of doing fast, testing, learning and improving.

7.2 Practical Implications

Based on the empirical material some practical suggestions can also be made for software startups. Software startups should have advisors as advisor board or by involving more experienced people from the field, for example their funders, to discuss their business model and work (PEC 3). This way they get new ideas, validate their ideas by more experienced view and get help for their work in general. Advisors also can be helpful while networking with customers and other startups.

Flat organization and self-organizing team seem to be effective way to construct the initial team for the startup stages of the company as stated in PEC 5. In many cases the startup is formed by the people who also work elsewhere or study while performing the initial activities of the startup such as developing a prototype and validating the idea. This leads to the fact that self-organizing

teams that are not strictly divided into different roles is effective as people's allocation of time for the startup might change. Initial team members are also usually co-partners so flat organization might result in more committed team.

The empirical material of this thesis suggests that focusing the product features to what is needed at the given moment and focusing the product to serve the most important customer segment helps the startup to succeed, stated in PEC 4. The startup works with the limited resources which means that they cannot do everything they want simultaneously so keeping tight focus on the most important things for the moment they can succeed on those and change or expand the focus later. Koltins et al. (2018a) have also pointed out that over-scoping the product or MVP is one of the most frequent reasons for failure of software startup. They suggest that scope of the product, especially in the early stage, should be chosen carefully to avoid adding too much features in the initial versions of product.

Usually startups have to start their work by funding from the founders or with small funding sources in the beginning. With this preliminary funding they can design their product and business and sometimes develop a prototype of the product. From the empirical material it is clear that securing investments from outside the team are usually the best way to make sure that the team has needed resources to develop profitable product and business model.

Forming deep relationship with the first customers is suggested by the empirical material. This will help startups to validate their product features with actual customers rather than just testing them inside the team (PEC 2). First customers can also be involved in the marketing activities if they agree to be references for the company which helps convincing others to buy the product or the service.

As PEC 6 stated it's notable that software startups tend to use common agile methods as the basis of their working methods but they tailor those methods to better suit their work. For example some of the interviewed startups told that they had implemented daily meetings from scrum but had left other parts of that method unimplemented because they didn't find those relevant for their current work.

Even though the Essence is a framework discussing the ubiquitous parts of software development it has also practical use. Especially after the suggested new alphas this framework could be used as a tool for software startups to monitor their progress in the most important areas of their work. Even without suggested modification the Essence could be used as a tool to monitor the progress of the software development and for documenting the working practices in a simple manner.

8 CONCLUSION

This thesis studied what is the essence of software startup. The list of software startup practices was used as a tool to point out the most crucial areas of software startups' endeavour to find the ubiquitous areas that always have to be addressed and progress measured. In the theoretical framework the practices were categorized under the Essence framework alphas to see how well they fit under those areas that are always present in the software development according to the Essence framework. Later thirteen new practices were added after the analysis of the empirical material and three new alphas (marketing, funding and business model) were suggested to be added to the Essence framework for it to support the whole endeavour of software startups.

The study was conducted following the qualitative case study method. The participants for the interviews were founders or CEOs of software startups. Five of the interviews were conducted by the researcher himself and eight of them were acquired from the previous s1tudy concerning early stage startups pivots and prototype development. These eight interviews were conducted and transcribed by the researchers of the original study.

The data collection method along with the research methods were presented in the section 5. The empirical results were presented in the section 6 and further discussed in the section 7 with the practical and theoretical implications of this study. In this section 8 the thesis is concluded. First the research question is answered in the subsection 8.1 and the limitations and future research possibilities are presented in the subsections 8.2 and 8.3.

8.1 Answer to Research Questions

This thesis aimed to answer the question: *How software startups' working practices fit under the alphas of the Essence framework?* The main question is divided into sub-questions:

1. How to define a software startup?

2. What practices are universal for all software startups?

3. What aspects of the essence of software development are not universal for all software startups?

The first sub-question was answered through the literature. Software startups were defined following the characterization created by Paternoster et al. (2014). This was chosen since it's the most widely used in resent startup research. They extend the startup definition created by Sutton (2000). Sutton's definition focuses on the characteristics: scarce resources, multiple influences, little or no operating history and dynamic technologies and markets. Paternoster et al. (2014) expanded this definition by adding the characteristics: creating innovative products, fast growth, flat organization structures, focusing on one product, third party dependency and time pressure. Therefore not all newly founded companies are startups but those fitting under this characterizing. Software startup is a company following characteristics created by Paternoster et al. (2014) whose product is software system or combination of software system and physical product or a service.

The answer for the second sub-question started with the list of startup practices found by Dande et al. (2014) that was later extended by thirteen practices emerging from the empirical material of this study. These practices are listed in the table ?? in the section ??.

According to this thesis the answer for the sub-question three is that all of the aspects of the Essence framework are universal for all software startups. The Essence framework describes universal aspects of software development and since the major part of software startups is software development these same universal aspects apply to the software startups also. The work practices differ between software startups and more mature software companies but the universal areas that need to be addressed and measured are the same.

The answer to the research question is that the essence of software startup can be defined using the Essence framework and adding three new alphas that are universal for all software startups and need to be addressed and whose progress should be measured. These alphas that need to be added to capture the essence of software startup are funding, marketing and business model. All the existing alphas are universal for all software startups but they lack the business aspects that are also universal for all software startups. These universal business related aspects can be fitted under the alphas marketing, funding and business model.

8.2 Limitations

When discussing the limitations of this study there are some features that need to be considered. All of the startups subject to this study where from the Finland or Norway. This raises the question if the practices studied in this thesis are indeed practices that startups use or rather practices that small companies

tend to use in the Nordic countries. Kamulegeya et al. (2017) study indicates that this risk is minimal since they have already tested the practices found by Dande et al. (2014) in the Ugandan startup context.

Another noticeable thing is that eight of the cases studied had been interviewed and transcribed by another research and they hadn't used the exact same interview themes since the original study had been focusing on the early stage startups' pivots and prototype development. The risk of missing crucial information because the interviews were conducted for another study is minimal since the original interviews themes were highly related to the ones of this study and the interviews had been conducted following the same thematic interview procedures.

8.3 Future Research

This study considers software startup's practices. The basis of these practices was the study by Dande et al. (2014) to which thirteen new practices was added. These practices could be further studied and the list should be expanded since there are many more working practices used by software startups and these could be studied to see which ones are the most beneficial in the practice. Practices should also be tested in the different geographical and cultural environment.

This study proposes three new alphas for the Essence framework to create a new framework for software startups' work. This opens an opportunity to test these suggestions and develop alternative changes for the framework for it to work in this context. The interactions between the new alphas introduced in this study and the existing alphas should be tested and formalized by another study.

The practices studied in this thesis and new practices emerging from future studies could also be mapped following categorization of Klotins et al. (2018b) startup context map. They have created a new categorization for software startups' practices that differ from the one used in this thesis. The model is relatively new and could be further tested by categorizing new practices following their map.

Unterkalmsteiner et al. (2016) have suggested a research agenda for software startup studies that includes areas that should be further studied. This thesis studied software startup practices that fit under multiple sections of their agenda. Future studies regarding the software startup practices could address practices more precisely on certain sector of that agenda to generate deeper knowledge on particular aspect of software startups' work.

REFERENCES

Aranda, J. (2009). Against SEMAT. Accessed 19 July 2013. Available: http://catenary.wordpress.com/2009/11/29/against-semat/.

Assyne, N. (2017, May). Collaborative-startup (co-startup): the role of communities of practices. In Proceedings of the 1st International Workshop on Software Engineering for Startups (pp. 6-9). IEEE Press.

Bajwa, S. S., Wang, X., Duc, A. N., & Abrahamsson, P. (2017). "Failures" to be celebrated: an analysis of major pivots of software startups. Empirical Software Engineering, 22(5), 2373-2408.

Björk, J., Ljungblad, J., & Bosch, J. (2013, June). Lean Product Development in Early Stage Startups. In IW-LCSP@ ICSOB (pp. 19-32).

Blank, S. (2007). The four steps to the epiphany: successful strategies for products that win. BookBaby.

Blank, S., & Dorf, B. (2012). The Startup Owners Manual, vol. 1. K&S Ranch Inc.

Bourdieu, P. (1973). The three forms of theoretical knowledge. Information (International Social Science Council), 12(1), 53-80.

Bourque, P., & Fairley, R. E. (2014). Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0. IEEE Computer Society Press.

Chang, S. J. (2004). Venture capital financing, strategic alliances, and the initial public offerings of Internet startups. Journal of Business Venturing, 19(5), 721-741.

Cho, S. T., Chomina-Chavez, A., & Bronowitz, J. (2017, June). A map of technology entrepreneurship: Aha to Exit. In Technology & Engineering Management Conference (TEMSCON), 2017 IEEE (pp. 148-154). IEEE.

Cockburn, A. (2010). A Detailed Critique of the SEMAT Initiative. Humans and Technology Technical Report HaT TR 201002. Accessed 19 July 2013. Available: http://alistair.cockburn.us/A+Detailed+Critique+of+the+SEMAT+Initiative.

Crowne, M. (2002). Why software product startups fail and what to do about it. Evolution of software product development in startup companies. In Engineering Management Conference, 2002. IEMC'02. 2002 IEEE International (Vol. 1, pp. 338-343). IEEE.

Cruzes, D. S., & Dyba, T. (2011, September). Recommended steps for thematic synthesis in software engineering. In Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on (pp. 275-284). IEEE.

da Rosa, S. C., Schreiber, D., Schmidt, S., & Junior, N. K. (2017). MANAGEMENT PRACTICES THAT COMBINE VALUE COCREATION AND USER EXPERIENCE An Analysis of the Nubank Startup in the Brazilian Market. Revista de Gestão, Finanças e Contabilidade, 7(2), 22.

Dande, A., Eloranta, V. P., Kovalainen, A. J., Lehtonen, T., Leppänen, M., Salmimaa, T., ... & Koskimies, K. (2014). Software startup patterns-an empirical study. Tampereen teknillinen yliopisto. Tietotekniikan laitos. Raportti-Tampere University of Technology. Department of Pervasive Computing. Report; 4.

DeTienne, D. R., & Cardon, M. S. (2012). Impact of founder experience on exit intentions. Small Business Economics, 38(4), 351-374.

Dey, I. (2003). Qualitative data analysis: A user friendly guide for social scientists. Routledge.

Dwolatzky, B. (2012, May). Re-founding software engineering practice-The SEMAT initiative. In Software Engineering Colloquium (SE), 2012 4th (pp. 1-3). IEEE.

Elvesæter, B. (2013, June). Extending the Kernel with Practices to Create Methods. In Essence Workshop. Berlin, Germany.

Evensen, A., Kemell, K. K., Wang, X., Risku, J., & Abrahamsson, P. (2018). Essencery-A Tool for Essentializing Software Engineering Practices. arXiv preprint arXiv:1808.02723.

Fowler, M. (2010). Semat. Accessed 19 July 2013. Available: http://martinfowler.com/bliki/Semat.html.

Giardino, C., Paternoster, N., Unterkalmsteiner, M., Gorschek, T., & Abrahamsson, P. (2016). Software development in startup companies: The greenfield startup model. IEEE Transactions on Software Engineering, 42(6), 585-604.

Giardino, C., Unterkalmsteiner, M., Paternoster, N., Gorschek, T., & Abrahamsson, P. (2014). What do we know about software development in startups?. IEEE software, 31(5), 28-32.

Giardino, C., Wang, X., & Abrahamsson, P. (2014, June). Why early-stage software startups fail: a behavioral framework. In International Conference of Software Business (pp. 27-41). Springer, Cham.

Gil, D. C., Serrato, J. H., & Melo, J. A. (2014). On the use of the SEMAT kernel within a software engineering course. METHODS, MODELING, AND TEACHING, VOL. 3, 77.

González-Pérez, M. E., Zapata-Jaramillo, C. M., & González-Palacio, L. (2014). Toward a standardized representation of RUP best practices of project management in the SEMAT kernel. METHODS, MODELING, AND TEACHING, VOL. 3, 47.

Hallen, B. L., Bingham, C. B., & Cohen, S. (2014, January). Do accelerators accelerate? A study of venture accelerators as a path to success?. In Academy of management proceedings (Vol. 2014, No. 1, p. 12955). Academy of Management.

Huang, S., & Ng, P. W. (2014). Essence as a Framework for Conducting Empirical Studies. METHODS, MODELING, AND TEACHING, VOL. 3, 9.

Ibargüengoitia, G., & Oktaba, H. (2014). Identifying the scope of Software Engineering for Beginners course using ESSENCE. METHODS, MODELING, AND TEACHING, VOL. 3, 67.

Jacobson, I., & Meyer, B. (2009). Methods need theory. Dr. Dobb's Journal.

Jacobson, I., Meyer, B., & Soley, R. (2009). Call for action: The SEMAT initiative. Dr. Dobb's Journal, 10.

Jacobson, I., Ng, P. W., McMahon, P. E., & Goedicke, M. (2019). The Essentials of Modern Software Engineering: Free the Practices from the Method Prisons!. Morgan & Claypool.

Jacobson, I., Ng, P. W., McMahon, P. E., Spence, I., & Lidman, S. (2013). The essence of software Engineering: applying the SEMAT kernel. Addison-Wesley.

Jacobson, I., Ng, P. W., McMahon, P., Spence, I., & Lidman, S. (2012). The essence of software engineering: the SEMAT kernel. Queue, 10(10), 40.

Jacobson, I., Ng, P. W., McMahon, P., Spence, I., & Lidman, S. (2012). The essence of software engineering: the SEMAT kernel. Queue, 10(10), 40.

Jacobson, I., Spence, I., & Ng, P. W. (2013). Agile and SEMAT: perfect partners. Communications of the ACM, 56(11), 53-59.

Jick, T. D. (1979). Mixing qualitative and quantitative methods: Triangulation in action. Administrative science quarterly, 24(4), 602-611.

Kamulegeya, G., Hebig, R., Hammouda, I., Chaudron, M., & Mugwanya, R. (2017, August). Exploring the Applicability of Software Startup Patterns in the Ugandan Context. In Software Engineering and Advanced Applications (SEAA), 2017 43rd Euromicro Conference on (pp. 116-124). IEEE.

Kane, T. J. (2010). The importance of startups in job creation and job destruction. Available at SSRN 1646934.

Kaplan, B., & Maxwell, J. A. (2005). Qualitative research methods for evaluating computer information systems. In Evaluating the organizational impact of healthcare information systems (pp. 30-55). Springer, New York, NY.

Kirk, J., & Miller, M. L. (1986). Reliability and validity in qualitative research. Sage.

Klotins, E., Unterkalmsteiner, M., & Gorschek, T. (2015, June). Software engineering knowledge areas in startup companies: a mapping study. In International Conference of Software Business (pp. 245-257). Springer, Cham.

Klotins, E., Unterkalmsteiner, M., & Gorschek, T. (2018a). Software Engineering Antipatterns in start-ups. IEEE Software, 36(2), 118-126.

Klotins, E., Unterkalmsteiner, M., & Gorschek, T. (2018b). Software-intensive product engineering in start-ups: a taxonomy. IEEE Software, 35(4), 44-52.

Klotins, E., Unterkalmsteiner, M., & Gorschek, T. (2019). Software engineering in start-up companies: An analysis of 88 experience reports. Empirical Software Engineering, 24(1), 68-102.

Lueg, R., Malinauskaite, L., & Marinova, I. (2014). The vital role of business processes for a business model: the case of a startup company. Problems and Perspectives in Management, (12, Iss. 4 (contin.)), 213-220.

MacMillan, I. C., Zemann, L., & Subbanarasimha, P. N. (1987). Criteria distinguishing successful from unsuccessful ventures in the venture screening process. Journal of business venturing, 2(2), 123-137.

Melegati, J., Goldman, A., & Paulo, S. (2016). Requirements Engineering in Software Startups: a Grounded Theory approach. 2nd Int. Work. Softw. Startups, Trondheim, Norw.

Myers, M. D., & Newman, M. (2007). The qualitative interview in IS research: Examining the craft. Information and organization, 17(1), 2-26.

Ng, P. W., & Huang, S. (2013). Essence: A framework to help bridge the gap between software engineering education and industry needs. In Software Engineering Education and Training (CSEE&T), 2013 IEEE 26th Conference on (pp. 304-308). IEEE.

Object Management Group (OMG). 2008. Software Process Engineering Metamodel (SPEM). Available: http://www.omg.org/spec/SPEM/2.0/

Park, J. S., Jacobson, I., Myburgh, B., Johnson, P., & McMahon, P. E. (n.d.) SEMAT Yesterday, Today and Tomorrow An Industry Perspective. Available: https://pdfs.semanticscholar.org/100e/dda4f5306cfb9cb8383110b23fe6175de41f.pdf

Paternoster, N., Giardino, C., Unterkalmsteiner, M., Gorschek, T., & Abrahamsson, P. (2014). Software development in startup companies: A systematic mapping study. Information and Software Technology, 56(10), 1200-1218.

Patton, M. Q. (1990). Qualitative evaluation and research methods. SAGE Publications, inc.

Rafiq, U., Bajwa, S. S., Wang, X., & Lunesu, I. (2017, August). Requirements Elicitation Techniques Applied in Software Startups. In Software Engineering and Advanced Applications (SEAA), 2017 43rd Euromicro Conference on (pp. 141-144). IEEE.

Ries, E. (2011). The Lean Start-up. How Constant Innovation Creates Radically Successful Business. Lloc de publicació: Londres. Portfolio Penguin.

Ries, E. (2011). The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses. Crown Books.

Robehmed, N. (2013) What is A Startup?. Forbes.

Schön, D. A. (2017). The reflective practitioner: How professionals think in action. Routledge.

Seppänen, P., Liukkunen, K., & Oivo, M. (2015, December). On the feasibility of startup models as a framework for research on competence needs in software startups. In International Conference on Product-Focused Software Process Improvement (pp. 569-576). Springer, Cham.

Seppänen, P., Liukkunen, K., & Oivo, M. (2017, November). Little Big Team: Acquiring Human Capital in Software Startups. In International Conference on Product-Focused Software Process Improvement (pp. 280-296). Springer, Cham.

Smolander, K., & Päivärinta, T. (2013, May). Forming theories of practices for software engineering. In Software Engineering (GTSE), 2013 2nd SEMAT Workshop on a General Theory of (pp. 27-34). IEEE.

Striewe, M., Mcneile, A. T., & Berre, A. J. (2012). Towards an Agile Foundation for the Creation and Enactment of Software Engineering Methods: The SEMAT Approach.

Striewe, M., Mcneile, A. T., & Berre, A. J. (2012). Towards an Agile Foundation for the Creation and Enactment of Software Engineering Methods: The SEMAT Approach.

Submitters, O. (2012). Essence–Kernel and Language for Software Engineering Methods.

Sutton, S. M. (2000). The role of process in software start-up. IEEE software, 17(4), 33-39.

Swenson, M. J., Rhoads, G. K., & Whitlark, D. B. (2014). Startup marketing: Leveraging leverage. The Journal of Applied Business and Economics, 16(6), 56.

Tolvanen, J. P. (1998). Incremental method engineering with modeling tools: theoretical principles and empirical evidence. Ph. D. Thesis, University of Jyvaskyla.

Tripathi, N., Seppänen, P., Oivo, M., Similä, J., & Liukkunen, K. (2017, August). The Effect of Competitor Interaction on Startup's Product Development. In Software Engineering and Advanced Applications (SEAA), 2017 43rd Euromicro Conference on (pp. 125-132). IEEE.

Unterkalmsteiner, M., Abrahamsson, P., Wang, X., Nguyen-Duc, A., Shah, S., Bajwa, S. S., ... & Edison, H. (2016). Software startups–a research agenda. e-Informatica Software Engineering Journal, 10(1).

Varantola, K., Launis, V., Helin, M., Spoof, S. K., & Jäppinen, S. (2013). Hyvä tieteellinen käytäntö ja sen loukkausepäilyjen käsitteleminen Suomessa. Helsinki: Tutkimuseettinen neuvottelukunta.

Wall, D. A. (2001). Using open source for a profitable startup. Computer, 34(12), 158-160.

Wang, X., Edison, H., Bajwa, S. S., Giardino, C., & Abrahamsson, P. (2016, May). Key challenges in software startups across life cycle stages. In International Conference on Agile Software Development (pp. 169-182). Springer, Cham.

Weerawardena, J. (2003). The role of marketing capability in innovation-based competitive strategy. Journal of strategic marketing, 11(1), 15-35.

Wennberg, K., Wiklund, J., DeTienne, D. R., & Cardon, M. S. (2010). Reconceptualizing entrepreneurial exit: Divergent exit routes and their drivers. Journal of Business Venturing, 25(4), 361-375.

Yin, R. K. (2013). Case study research and applications: Design and methods. Sage publications.

Yu, Y. W., Chang, Y. S., Chen, Y. F., & Chu, L. S. (2012, July). Entrepreneurial Success for High-Tech Start-Ups--Case Study of Taiwan High-Tech Companies. In Innovative Mobile and Internet Services in Ubiquitous

Computing (IMIS), 2012 Sixth International Conference on (pp. 933-937). IEEE.

Zalewski, J. (2013). Review, The essence of software engineering : applying the SEMAT kernel. Computing Reviews. Available: http://www.computingreviews.com/review/review_review.cfm?review_id=141474

APPENDIX 1 LIST OF THE STARTUP PRACTICES

Practice	Description
Focus your product	Focus on the most potential customer segment. Be prepared to change the focus
Find your value proposition and stick to it on all levels	Create a valid value proposition. Discuss with experts from strategic and operational levels at customer's organization.
Present the product as facilitating rather than competing to the competitors	Develop a product that can co-operate rather than compete with competitors
Focus on goals, whys	Find real motivations behind customers' wishes by asking why they want something rather than just what they want. This way you understand customers' needs deeper and can address them in other situations as well.
Use proven UX methods	Use proven UX development methods from the beginning. Validate ideas quickly by using prototypes.
Do something spectacular	Create WOW effects and feelings to the customer to stand out in the competition.
Have a single product, no per customer variants	Have a modular and flexible single product rather than multiple per customer variants.
Restrict the number of platforms that your product works on	Make business decisions on what platforms you want to support. Focus on the most important ones. For example the most used browsers and operating systems.
Use enabling specifications	Enable specification to guide work efficiently. Let team work independently without constant intervene from the owner or customer.
Design and conduct experiments to find out about user preferences	Use experiments and communication with user to determinate in which directions product should be developed.
Use tools to collect data about user behaviour	Use data to acknowledge user behaviour and choose best marketing channels.
Make your idea into a product	Turn your ideas into products rather than projects. Projects are not easily scaled.
Outsource your growth	Use outsourcing to keep your focus on the product.
Anyone can release and stop release	Allow anyone to make a release or stop it. Fast releases allow quick feedback from users.
Create the development culture before processes	In the beginning develop a culture that supports what you want to be. Processes are likely to change as company evolves so focus first on building the culture that fits your goals and future processes.
Get venture capital and push your product	Try to get your product profitable fast with venture capital rather than develop it slowly in silence with low resources.
Fund it yourself	Getting funding with proof of concept is not easy. Fund first yourself and get investment later.
Validate that your product sells	Validate your idea before starting development or try to get a few customers before you start developing.
Focus early on those people who will give you income in the long run	Try to get your business model running from the start, even in small scale. Focus on paying customers to ensure that the company is profitable.

Form deep relations with first customers to really understand their needs	To understand the customers and the business develop as deep relations as you can with the first customers.
Use planning tools that really show value provided to customer	Choose tools that allow mapping the value customer gets from what is done and planned.
Start locally grow globally	Target local customers in the beginning but make all decisions considering the global growth.
Adapt your release cycles to the culture of your users	Depending on your customers choose how fast releases are and how much can be changed at once.
Keep customer communications simple and natural	A startup needs quick and good feedback from customers for development decisions. Try to encourage direct contacts by email or through integrated feedback mechanisms.
Help customers create a great showcase for you with support	The first customers can provide a visible showcase to attract other customers.
Flat organization	In flat organization people are committed to a common good and communications are easy as they don't require intermediates.
Consider career expectations of good people	Keep team happy by offering opportunities to build up their skills. They can raise their market value as an insurance for the case that startup fails.
Don't grow in personnel	If you don't need more resources or competence don't grow in personnel.
Bind key people	Most important people should be shareholders, partners or founders because critical information is easily lost.
Form partnerships and bonds with other startups	Focus on developing your product and on your core business. For other issues find partnering startup. Startups are usually keen to co-operate.
Make your own strength as a “brand”	All startups should have exceptional skills or product. Turn this strength into a brand in the market.
Showing alternatives is the highest proof of expertise	Finding different alternatives for a solution is expertise. Explore alternatives to find a good solution.
In the development of customer solutions, find a unique value proposition in your way of acting	Find the way of acting that differs from your competitors. For example super-fast or people centric.
Follow communities	Everyone should follow communities to know what is happening and to find new values for customer.
Share ideas and get more back	Sharing ideas will help you get valuable feedback.
Small co-located teams	Small teams with scarce resources need good communication to survive. Speaking in the same room is the most effective way to communicate.
Have multi-skilled developers	Startups have usually small teams, yet there are lots of different things to do. Multi-skilled developers are needed to address all the issues in startup without growing in personnel.
Keep teams stable in growth mode	While growing as a company try to keep teams and individual roles stable.
Let teams self-select	Teams should be allowed to self-organize.
Sharing competence in team	In team everyone has slightly different expertise. Since startups need skilled developers sharing competence inside the team is necessary.
Start with a competence focus and expand as needed	In the beginning focus on specific competence with a small group of people. Expand team and competences later.
Start with small and experienced team and expand as needed	Start with small and experienced team that has efficient ways to communicate. Anticipating all needed skills beforehand is hard.
Have different processes for different goals	Choose different practices for different tasks if needed.

Tailored gates and done criteria	Process phases leading to something being done or assessed or accepted should reflect the overall process and business.
Time process improvements right	Improve and change processes only when it is absolutely needed. At some point of the growth startup might need to change its preliminary processes.
Find the overall development approach that fits your company and its business	Find the best approach for your business. Don't follow latest trends if it's not best fit for you.
Tailor common agile practices for your culture and needs	Most textbook practices are highly general. Tailor them to fit your needs and culture.
Fail fast, stop and fix	Allow developers to do things quickly and freely and stop if something goes wrong. They will then fix the problem and process in the team.
Move fast and break things	Prefer culture with fast development and where failing is acceptable.
Forget Software Engineering	Software development may be ad hoc and unorganized if it us good enough with the physical product.
Anything goes in product planning	Startup needs to figure out new features, system concepts and new projects.
To minimize problems with changes and variations, develop a very focused concept	Develop a validated and focused concept to minimize risks with changes. Be still ready to do changes if needed.
Develop only what is needed now	Be efficient by developing only what is needed now.
Make features easy to remove	Use techniques and architecture that make features easy to remove if needed.
Use extendable product architecture	Use architecture and techniques that allow to extent design easily.
Only use reliable metrics	Use reliable metrics to validate things. Wrong metrics might do harm for validation.
Bughunt	During fast development of new features arrange days for bughunt. Make bughunt fun occasion when everyone is searching for bugs.
Test APIs automatically, UIs manually	APIs can be tested by tools that are easy to find and cheap. Test UI manually in the beginning.
Use generic, non-proprietary technologies	Use platform independent technologies to avoid re-implementing features.
Create a solid platform	Keep scaling in mind while developing a platform.
Choose scalable technologies	Favour development techniques that scale easily.
Use the most efficient programming languages and platforms	With a small team choose the most efficient programming languages and development platforms.
Start with familiar technologies and processes	Save the time of learning new technologies and processes by using those that team is familiar with.
Study subjects that support startup	Studying while working on a startup gains competence in the team without growing in personnel.
Attend startup events	Startup events provide opportunity for feedback from experts and allows you to meet potential investors
Create an MVP in the beginning	MVP helps you to focus on the most important features in the beginning
Test features with customers	Testing features with real customers gets you the best feedback
Get advisors	Experienced professionals or investors can help startup to grow in advisor or mentor role
Use efficient tools to plan your business model	Business model canvas, pitch deck etc. help you to focus your business idea and are easy to change if needed
Test different tools	Start with tools team is familiar with and test different ones to find those that work the best for you

Conduct market research	Research the markets and competitors to focus your idea and to find your unique value proposition.
Have frequent meetings with whole team	Use meetings to organize and plan your work at least once a week
Don't have strict roles	Let the team co-operate in all of the tasks
Create prototype	Create prototype to validate your product or features
Use efficient communication tools	Use tools that allow natural communication inside the team when not working in the same space.
Prioritize features	Choose which features are needed now and plan others for future releases.

APPENDIX 2 LIST OF ABBREVIATIONS

EC: Empirical conclusion
MVP: Minimum viable product
OMG: Object Management Group
PEC: Primary empirical conclusion
SEMAT: Software Engineering Methods and Theory
SPEM: Software and Systems Process Engineering Metamodel
SWEBOK: Software Engineering Body of Knowledge
UX: User Experience

APPENDIX 3 THE FRAME OF THE INTERVIEWS

Basic information about the interview

- The purpose of the study (I'm studying software startups. Goal is to find out more information how startups work and provide help for future startups), the interview process (takes around an hour), and how results are dealt in the thesis.
- Confidentiality, anonymity.
- Permission to record the interview.

Interviewee's background

- Job title and responsibilities in the startup.
- How long has been involved in the startup.
- Startup's domain.
- When the startup was found.
- Team size.
- Other background information.

Startup's story

- How the idea was born?
- What is the current situation?
- How have they got to this point?
- What are the most important milestones?
    - What led to these? What was done to achieve a milestone?
- What have been the challenges?
- What has been important for the success?

Themes

- Opportunity: Customer's needs? Revenue streams? Funding?
- Stakeholder: What kind of stakeholders? How have they taken customers in to consideration while planning the business/product? Have they had a prototype?
- Requirements: How the work has been guided and planned? Have they validated the idea?
- Software System: What is the product? Have they used MVP? How is the architecture?
- Work

- Team: How big is the team? What kind of know-how do they have? Have they expanded the team? When and why?
- Way-of-working: How they work? Have they certain methods or practices they follow? Have these changed?
- Metrics: What they measure on their work?
- Business model: How they composed their business model? Do they have a model? Has it changed?

Other possible questions

- Can you tell how you have organized your work?
- How have you managed the communications?
- What have you found to be important for your product? What about your working?
- How have the working practices changed?