Requirements Engineering in Software Startups: a Qualitative Investigation

Jorge Augusto Melegati Gonçalves

Dissertation submitted
to the
Institute of Mathematics and Statistics
of the
University of São Paulo
for
obtainment of the title
of
Master of Science

Program: Ciência da Computação
Supervisor: Prof. Dr. Alfredo Goldman vel Lejbman

São Paulo, January 2017

Requirements Engineering in Software Startups: a Qualitative Investigation

Acknowledgments

I would like to thank Professor Alfredo for the several lessons taught, for being patient with me and for allowing me to take my time to develop this research.

I am very grateful to all interviewees that spent some precious time talking about their startups and experience. Without their help this work would not be possible. I expect that the results justified your offering. You should be certain that everything was done to make the best out of it. I also feel very thankful to everyone in the Institute that made this possible including professors, secretaries and colleagues.

Lívia, thank you for your support and patience. I am also very grateful to all my family, specially my brother Bruno who really helped me when needed. Finally, I would like to dedicate this work to my grandfather Natal who passed away during this journey. He is one of the most responsible people for my achievements.

Resumo

GONÇALVES, J. A. M. **Engenharia de Requisitos em Startups de Software: uma investigação qualitativa**. 2017. 90 f. Dissertação (Mestrado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2017.

Startups de software enfrentam um mercado muito exigente: elas devem entregar soluções altamente inovativas no menor período de tempo possível. Recursos são limitados e tempo para alcançar o mercado é pequeno. Então, é extremamente importante coletar os requisitos certos e que eles sejam precisos. Entretanto, os requisitos de software geralmente não são claros e as startups fazem um grande esforço para identificar quais serão implementados. Esse contexto afeta como as atividades de engenharia de requisitos são executadas nessas organizações. Este trabalho procura compreender o estado-da-prática da engenharia de requisitos em startups de software. Usando uma abordagem iterativa, dezessete entrevistas foram realizados em três diferentes estágios com fundadores e/ou gestores de diferentes startups de software brasileiras operando em diferentes setores e com diferentes estágios de maturidade. Os dados foram analisados usando técnicas de teoria fundamentada como codificação aberta e axial através da comparação contínua. Como resultado, um modelo conceitual do estado-da-prática da engenharia de requisitos em startups de software foi desenvolvido consistindo da suas influências do contexto (fundadores, gerente de desenvolvimento de software, desenvolvedores, modelo de negócio, mercado e ecossistema) e descrição das atividades (time de produto; levantamento; análise, validação e priorização; e documentação). Técnicas oriundas de metodologias de desenvolvimento de software e desenvolvimento de startups também são apresentadas e seu uso em no contexto de startups é analisado. Finalmente, a partir de uma analogia de maus cheiros presente na literatura de desenvolvimento de software, algumas más práticas e maus comportamentos identificados em startups de software são apresentados e algumas sugestões de solução são propostas.

Palavras-chave: startups de software, engenharia de requisitos, engenharia de software experimental.

Abstract

Gonçalves, J. A. M. **Requirements Engineering in Software Startups: a Qualitative Investigation**. 2017. 90 f. Dissertação (Mestrado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2017.

Software startups face a very demanding market: they must deliver high innovative solutions in the shortest possible period of time. Resources are limited and time to reach market is short. Then, it is extremely important to gather the right requirements and that they are precise. Nevertheless, software requirements are usually not clear and startups struggle to identify what they should build. This context affects how requirements engineering activities are performed in these organizations. This work seeks to characterize the state-of-practice of requirements engineering in software startups. Using an iterative approach, seventeen interviews were conducted during three stages with founders and/or managers of different Brazilian software startups operating in different market sectors and with different maturity levels. Data was analyzed using grounded theory techniques such open and axial coding through continuous comparison. As a result, a conceptual model of requirements engineering state-of-practice in software startups was developed consisting of its context influences (founders, software development manager, developers, business model, market and ecosystem) and activities description (product team; elicitation; analysis, validation and prioritization; product validation and documentation). Software development and startup development techniques are also presented and their use in the startup context is analyzed. Finally, using a bad smell analogy borrowed from software development literature, some bad practices and behaviors identified in software startups are presented and solutions to avoid them proposed.

Keywords: software startups, requirements engineering, empirical software engineering.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Software startups are at the heart of the technological revolution that has taken place in the world for the last fifty years. They are, at same time, an effect of scientific and technological developments and a cause of several changes in the way mankind consume, communicate and live. Facebook, the actual worldwide biggest social network, has approximately 1 billion users[1] and Amazon had total sales of 89 billions dollars in 2014[2]. Facebook and Amazon are only two examples of how deep changes induced by successful startups are.

The current historical period is characterized by an unprecedented speed of technological development, an increasing globalization and several changes in society. Startups and other organizations face this context in their day-to-day life. In this highly changing environment, software requirements are difficult to extract and they change frequently [PGU$^+$14]. Nevertheless, the capacity of innovative solutions delivery is very important for startups success [DD05]. After the "dot-com" bubble burst [TM14] in the beginning of the century, the importance of how startups work has grown. Since then, from the criticism of used practices, some new startups development methodologies have emerged in the industry like Customer Developer [Bla07] and Lean Startup [Rie11]. These new techniques are presented through the description of practical experiences and examples of their use. Even having a gap between theory and practice [Pat13], they are getting increasingly attention in specialized media and industry events. Although a huge publishing and media sucess, there are few evidences in literature of these techniques real use as pointed out by Paternoster et al. [PGU$^+$14].

An important concept to requirements engineering is the *stakeholder*. The term was coined by Freeman and Reed [FR83] in relation to stockholders of a company and refers to "those groups without whose support the organization would cease to exist". The authors also list the common list of stakeholders: "shareowners, employees, customers, suppliers, lenders and society". In software development, the term is commonly used to refer to software users and other people related to the task like other teams in the company, owners, etc. Traditionally, they are the main source of requirements and identifying them is one of the tasks in requirements engineering as will be discussed in section 1.2.

Nevertheless, startups must handle several stakeholders. Not only users but also investors and founders that have committed time and/or money and have their expectations and beliefs on what should be accomplished. The developers themselves have their own expectations and want the best

[1]Available in http://newsroom.fb.com/company-info/. Accessed 2017-01-16.
[2]Available in http://www.bbc.com/news/business-31051044. Accessed 2017-01-16.

for the organization. Still, startups have limited human and financial resources and the time to reach the market is short. Hence, more specific software requirements engineering techniques would be very welcome for software startups.

As a first step on this direction, a detailed description of software requirements engineering state-of-practice would make possible the evolution of the startups development methodologies or even the creation of new practices that aim a better efficiency on startup creation and management.

In a software startups literature systematic mapping, Paternoster et al. [PGU+14] devote a specific section for requirements engineering. The authors have not found any study focused on software requirements engineering practices but acknowledge that requirements engineering practices:

- are limited to some key practices;
- present an increasing difficulty because of a growing number of users;
- demand, from several authors perspectives, more user involvement.

Given such context, the research question that will guide this study is:

How requirements engineering practices are executed in software startups?

In the following sections, some term definitions will be made to determine this study scope. First, a discussion on what a software startup *is*. Then, a description of requirements engineering and some methodologies used in industry on startup development. Finally, this study objectives are presented and the text organization is provided.

1.1 Startup definition

Paternoster et al. recognized in their systematic mapping [PGU+14] that different authors adopted different definitions to what a startup is. In this sense, it is important to make clear which definition is used by this research. Sutton [Sut00] proposed a definition based on which characteristics a startup presents: little or no operation history; limited resources; multiples influences; and dynamic technologies and markets. After analyzing several other studies, Paternoster et al. have expanded this definition adding other aspects: innovation, fast growth, time pressure, third party dependency, focus on one product and flat organizational structures.

Although several studies use definitions similar to this one, it can leave out some representative cases. For instance, Clinkle[3] had received US$30 million to build a brand new mobile payment technology through sound but was still considered by media as a startup. This definition also excludes groups formed inside large companies to develop new products and technologies ([ABG+10], [VMM16], [RS16]). These groups face similar challenges and are in similar situations that the so-called startups. So, we preferred to stay with Ries' definition [Rie11]: "a startup is a human institution designed to deliver a new product or service under conditions of extreme uncertainty". This definition covers innovation, time pressure and focus on only one product that has a direct effect on requirements engineering, this research focus. At the same time, it sets aside characteristics like little or no operation history and limited resources. As said before, these characteristics would exclude organizations that face similar contexts to conventional startups.

[3]Available in http://www.businessinsider.com/inside-story-of-clinkle-2014-4. Accessed 2017-01-16.

Given Ries' definition, it can be tempting to only call startup a company that is building something brand-new. Nevertheless there are companies that are making some kind of technological improvement or bringing a technology to a new market then they are also facing extreme uncertainties. As stated by Amason et al. [AST06]: "Every new venture represents some type of innovation and these various types of innovation can be placed along a continuum. At one end are the revolutionary innovations that spark dramatic and radical change for whole segments of an industry. At the other end are that evolutionary innovations that modify and refine existing practices". In the case of software startups another point should also be made, as Häsel et al. [HKB10] states: "companies can develop a completely new software technology to realize their products or rely on existing standards, frameworks, and components". That is, a software startup can disrupt or innovate in the software itself or software can be a mean to do it.

1.2 Requirements engineering

According to Nuseibeh and Easterbrook [NE00], "the primary measure of success of a software system is the degree to which it meets the purpose for which it was intended". Thereafter the authors define Requirements Engineering (RE) as the process of discovering that purpose by identifying the stakeholders and their needs, documenting the discoveries for future analysis, communication and implementation. In their Requirements Engineering textbook, Kotonya and Sommerville [SK98] enumerate the activities performed under RE: elicitation; analysis and negotiation; documentation and validation.

Elicitation is requirements engineering first stage and its main objective is to find out the problem to be solved and, thereby, define system boundaries. The first task is to identify the system stakeholders, that is as said before, people that have some interest in it. The most common stakeholders are clients, users and developers themselves. From interacting with these stakeholders, it is possible to determine which requirements should be implemented. This task can be extremely hard for startups that focus a large group of people, for instance, one that is building a website or an app.

During elicitation stage, several stakeholders provide different requirements. Inevitably, several inconsistencies appear: conflicts on different sources, lack of necessary details to allow implementation or inadequacy to project scope. Hence, it is necessary a stage to analyze and detail requirements and negotiate with stakeholders which requirements will be implemented. An Extreme Programming principle proposed by Beck and Andres [BA05], which is present in several agile methodologies, is the Customer Always Available. According to this principle, at least one client is always colocated with the development team: working together with them, helping with requirements analysis and negotiation and, even, answering questions that could show up. Although most of startups use agile methodologies [PGU+14], there is no descriptions in literature of how this is done in software startups.

Once the requirements are selected, they are documented to be observed during development and further validation on software deliveries. Under agile methodologies, working software is more important than a comprehensive documentation as pointed out in Agile Manifesto [BBvB+01]. Nevertheless, Selic [Sel09] warns that it is possible to misunderstand this phrase. According to the author, the principle refers to documents describing implementation plans created before any knowledge about the problem or the solution has been absorbed. Instead, he advocates that we need documentation at a higher level closely related to application concepts and requirements over technological details.

Finally, requirements should also be validated before their implementation to check if they are complete and consistent. That is, it should be checked if requirements certainly and completely meet users expectations. It is also important to define tests to verify if a requirement has accomplished its objectives after implementation.

1.3 Startup development methodologies

Requirements engineering is closely related to software startups business activities as shown in figure 1.1. As Cheng and Atlee [CAJ07] state: "requirements reside primarily in the problem space whereas other software artifacts reside primarily in the solution space". This is also important for agile methodologies where there are roles directly related to business like Product Owner in Scrum [Sch04] or the Customer Always Available in Extreme Programming [BA05]. In this sense, it is important to discuss the software product management concept. According to Ebert [Ebe07], Software Product Management is the discipline and business process that guide a product from inception to delivery. Then, it is important to analyze which product management techniques are being used by startups since they will tell how part of requirements engineering is done.

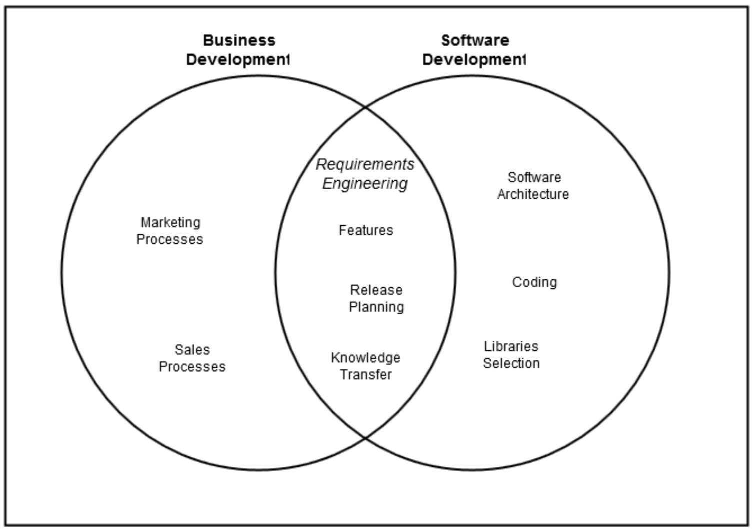


Figure 1.1: *Startups practices: business and software development.*

According to Blank [Bla13], the way a new product was launched by startups followed a traditional formula that could only end in a complete success or a huge failure. This formula consisted of: writing a business plan, present it to possible investors, assemble a team, create and build the product and sell it. The dot-com bubble burst led entrepreneurs to question if this traditional way was the most suitable for startups. In this context, new methodologies focused on startups appearead like Customer Development [Bla07] and Lean Startup [Rie11]. Besides that, techniques brought from product design began to be used like Design Thinking [Bro09].

1.4 Objectives

In summary, we discussed what we want to study (requirements engineering activities) and in which context (software startups). Software development in startups are different from software development in general because they face an unique context in which they should produce a lot with few. Some methodologies and techniques were developed to be used in this context and they might influence how requirements engineering are executed in software startups. In which extent this is true is still not well explored in literature.

Our intent is to reduce the gap between theory and practice providing a detailed study on what actually happens concerning software requirements in software startups. Then, this research main goal is to develop a state-of-practice model of requirements engineering in software startups. Besides that, since startup methodologies are in evidence, a secondary objective is derived: verify if these methodologies are used and to what degree. And once the requirements engineering processes are mapped for startups, another possible secondary objective is to provide some practical advice to these companies.

1.5 Organization

This text is organized as follow. In chapter 2, related work is presented and discussed. In chapter 3, the research design is presented and in chapter 4 the research execution is detailed. In chapter 5, the requirements engineering conceptual framework developed during the research is presented, discussed and related to others works in literature. In chapter 6, some bad practices perceived are presented and recommendations are made to software startups. Finally, in chapter 7, a conclusion is presented including this work major contributions, its major limitations, how these limitations were minimized and possibilities for future studies.

Chapter 2

Related work

As presented previously, there is little discussion on the literature concerning specifically requirements engineering for startups. Some studies focused on software development as a whole in software startups. These studies are presented and discussed in section 2.1. Given this research secondary objective - to verify to which extent startup development methodologies are used in software startups - the most common methodologies are presented in section 2.2. Finally, in section 2.3 an overview of literature in requirements engineering is presented. And since startups tend to use agile methodologies as they are product-driven, focus on small teams and embrace changes instead of avoiding them ([Tai10] and [CO08]), this last section also focus on agile requirements engineering.

2.1 Software development in startups

Paternoster et al. conducted a systematic mapping study on software development in startups [PGU+14]. The authors concluded that software development in startups is "not supported by a scientific body of knowledge". Although the authors examined 43 primary studies, only 4 were focused on software development in startups, used a evidence-based research methodology and made a strong contribution to the field. Besides that, 3 of these 4 studies were based on the same data. The systematic mapping study extracted work practices and divided them in categories: process management practices; software development practices; managerial and organization practices; and tools and technologies. Software development practices were divided in requirements engineering practices; design and architecture practices; implementation, maintenance and deployment practices; and quality assurance practices.

The subsection dedicated to requirement engineering practices focused on requirement elicitation. They state that practices "are often reduced to some key basic activities". They grounded this argument in the works of Crowne [Cro02] and Zettel et al. [ZMMW01]. The first listed reasons to startups failures and included requirements related reasons. For instance, "product isn't a product" or "requirements become unmanageable" that occurs when there are more requirements than the team can deliver and there is no way to decide among them. The latter proposed a lightweight development methodology for startups. This methodology has only few activities related to requirements engineering like "collect scenarios" and lacks discussion on, for instance, how decide between requirements. They also discussed how difficult is to elicit and detail requirements in an innovative market and when final users are unknown - requirements are market-driven rather specific to a customer. On their conclusion, authors acknowledge that demonstrating problem/solution fit is required to discover what the user really needs.

Coleman and O'Connor [CO08] employed Grounded Theory to study software development process formation in Irish software startups. The authors developed a theoretical framework with categories and their relations to explain software development process in startups. The categories found are described in table 2.1.

The software development manager will determine which process model will be used as a guide to process activities and this model will be subject to process tailoring. The software development manager is a founder or someone hired in the beginning of company activities. In both cases, the manager will bring a development culture including a software development methodology like RUP [Kru04] or XP [BA05] that will be the used process foundation. Besides techniques, the manager also brings a management style. The management style could be: "command and control" or "embrace and empower". The first occurs when there is no trust on developers and they must be constantly observed and analyzed. The latter, when developers perform their tasks with more independence because of a greater trust level. The management style is also influenced by the founder(s) and they can have a technological background or not. Therefore, the software development process is formed by the management style within a tailored process based on a well-known methodology.

Software development manager experience	The software development manager (a founder or someone hired in the beginning of the startup life) will mold the process used by the current firm.
Founder experience	S/he can be from IT area or not but s/he will determine, within software development manager, the management style.
Management style	The authors detected two different types: "command and control" when there is no trust on developers and they are kept under strict control and "embrace and empower" when there is big trust on development team and they are allowed to do tasks with less or even without supervision.
Process tailoring	The chosen process are not, generally, strictly followed. They are adapted to the organization needs and market requirements.
Market requirements	The market where the firm operates can demand specific requirements like high availability, extensive documentation or high delivery speed.

Table 2.1: *Software development formation process categories*

Giardino et al. developed a software development model for startups also through grounded theory [GPU+15]. The authors called it the Greenfield Startup Model and it is composed of seven main categories that are:

- Speed-up development;
- Evolutionary approach;
- Product quality has low priority;
- Team is the catalyst of development;

- Accumulated technical debt;
- Initial growth hinders performance;
- Several lack of resources.

As implications of their model, authors recognized that startups use a light-weight methodology, empower team members and focus on a minimal set of functionalities. Their work also discussed technical debt ([Cun93]) in software startups. They concluded that the speed-up in the beginning of the startup creates a lot of technical debt that may hinder the development performance in the future when the startup grows.

2.2 Startup development methodologies

In this section, it will be described some startup development methodologies that are being used. These methodologies have arisen after the dot-com bubble burst. According to Thiel [TM14], during the bubble period "the most 'successful' companies seemed to embrace a sort of anti-business model where they lost money as they grew". Then, a disbelief in the traditional way products have been launched emerged. Following the traditional path, a company should develop a business plan, present it to possible investors, build a team, create the product and then sell it as described by Blank [Bla07].

2.2.1 Customer development

According to Blank, the traditional product development method does not work for startups. Actually, these methods work well for products where there can be problems in development and/or distribution but they do not work well when the product may not be accepted by customers. Nevertheless, in software and web markets, most of problems are acceptance or adoption related. According to the author, most of startups fail because of lack of customers and not due to product developments mistakes. Then, he proposes the Customer Development process that focus in the client and in the market since the beginning. The process comprehends four steps: customer discovery, customer validation, customer creation and company building.

In customer discovery stage, Blank suggested that the possible customers are more listened and hypothesis about the market and the product should be tested. The team should look for customer problems, check if the product could solve them and try to estimate how much customers are willing to pay for it. Ideas should be evaluated through three lens:

- **technological:** if the product can be developed now, how much would be spent in research and development and if there would be intellectual property issues;
- **customer related:** besides evaluating if the product solves a customer problem, the market size should also be checked, if there are other competitors and if your product will have competitive advantages over these possible competitors;
- **oportunity:** if it is a unique idea, nonexistent in the current market and if the idea is possible and not something unsustainable.

In customer validation stage, a repeatable sales process must be built and, mostly, a check done if the business model makes sense. Sales metrics are used in this moment like customer long time value (LVM) and return on investment (ROI). It is possible that the product proves to be unsustainable. In this case, the process should come back to the previous stage, starting over, but more knowledge on market is available.

Once the product is validated, that is, it has been proven that is desirable and viable, the next stage takes place: customer creation. In this stage, the goal is to grow the customer base. There are several marketing techniques to keep up with that ([CSS12]). Finally, an organization must be created (or adapted in case it already exists) and verify if its mission agrees with the developed product. In figure 2.1, the Customer Development process is summarized.

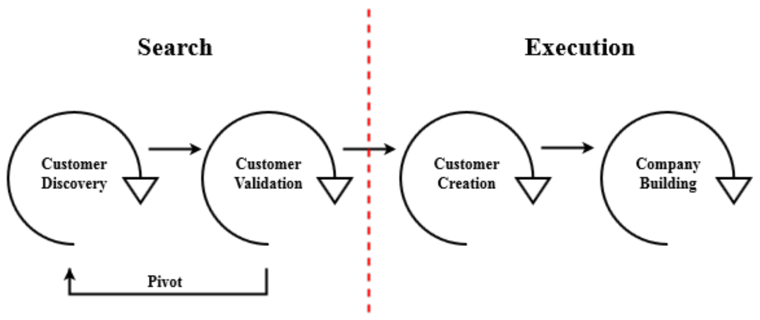


Figure 2.1: *Customer development process.*

2.2.2 Lean startup

Eric Ries was a student in a Blank's course where the latter presented his Customer Development methodology. This was a condition imposed by Blank in his investment on Ries' startup [Bla13]. Ries realized that there was a relation between this new methodology and the Japanese lean manufacturing techniques that were already been used in software development [PP03]. Then, he applied lean principles, mainly elimination of waste, in the process of creating a startup - the Lean Startup methodology arose.

According to the author, startups should avoid a long phase on business plan creation and planning where no revenue is created, the so-called stealth mode. On the contrary, they should use small *Build-Measure-Learn* cycles represented in figure 2.2. During these cycles the team should not take their ideas on the market for granted but instead take them as hypothesis that should be tested as soon as possible. In this sense, he proposes the Minimum Viable Product (MVP) concept that is to build the least minimum set of features (*Build*) that allows the hypothesis evaluation through metrics (*Measure*), validating the initial hypothesis or not, that is, learning something about the market (*Learn*).

A startup should replace a complete business plan for a market vision with an objective to build a business. From this vision, through several Build-Measure-Learn cycles, it would be possible to develop a product that meets the market needs even when it takes to changes on the initial idea. This course change is called pivot.

Lean Startup got a lot of attention. The book that introduced the methodology was still between the most sold even after four years from its publication[1]. College courses about it have been offered

[1]According to http://www.chicagotribune.com/bluesky/originals/chi-sxsw-eric-ries-kickstarter-bsi-20150316-

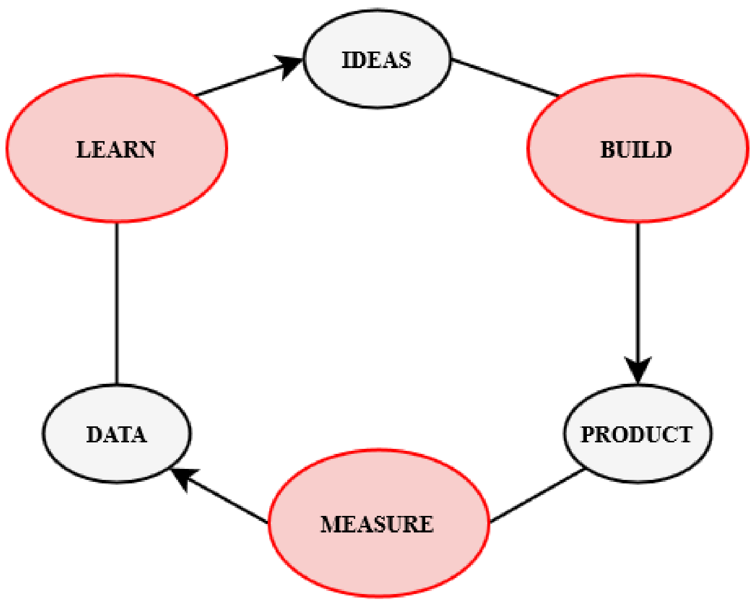


Figure 2.2: *Build Measure Learn loop.*

([Nob11], [Pau15], [Har15]) and studies about it have been performed ([Moo12], [Hui13]) including investigations about its effectiveness like [DS16].

2.2.3 Design thinking

Design Thinking is a set of techniques proposed to systematize activities performed by designers during product development. The name oppose the scientific method in which all parameters of a problem are defined before searching for a solution. Design Thinking is based on solutions, allowing several possibilities to be explored at the same time. Design thinking tackles "wicked problems" [BPI92]. The "wicked" term was first used by Rittel and Webber [RW73] to describe planning problems. The authors call the problems scientists and engineers have usually focused upon as "tame" problems. These problems have clear missions and it can be verified if a solution has been reached or not. Nevertheless, wicked problems can have "an exhaustive inventory of [..] conceivable solutions[...]". There are several versions of the method in which stages have different names but without changing the process core. Simon [Sim96] enumerates the following stages:

- **Determine:** find out what is the problem whose solution is looked for and which is the target audience;
- **Research:** review the problem and try to remember all obstacles, revisit prior proposed solutions;
- **Creation:** identify possible clients needs, create as many as possible ideas to solve the problem (brainstorm);
- **Prototype:** create product drafts and present them to the target audience, obtaining as many as possible comments and critiques (feedback);
- **Choose:** review objects, analyze the results and identify the most promising ideas;

story.html. Accessed 2017-01-16.

- **Implement:** develop the chosen solution and deliver it to the client;
- **Learn:** obtain clients' impressions, discuss what can be improved, measure project success.

Mueller and Thoring [MT12] compare Design Thinking and Lean Startup. The authors describe their similarities like innovation focus; user-centered approach; test prototypes; rapid iteration and their differences like scope; project initiation; user research; synthesis; customer, users and stakeholders; ideation; iteration/pivoting; adaption of deployments; quantitative evaluation; business model and qualitative evaluation.

Johansson-Skoldberg et al. [JSWÇ13] discuss the design thinking discourse. According to the authors, there are actually two discourses: designerly thinking and design thinking. The first one is the academic construction of the professional designer's practices, trying to understand the "non-verbal competence of the designers". While the latter is when the design practice is used beyond the design context including their methods into practical management for instance. The second discourse is more important to our work since as pointed out by the authors, this concept "became a portal for the whole design area to contribute to innovation" and "a way to deal with a complex reality". The authors mention that design thinking in management area could be linked to three different origins and the first one is related to innovation. This first origin is a book written by IDEO's CEO (IDEO is the world's largest design company [JSWÇ13]) Tim Brown [Bro09]. Still according to the authors, the stories in this book are compelling but "there is no published theoretical framework".

IDEO's design thinking became very popular in startups common literature and a brief description will follow. Brown [Bro08] described the design process as a "system of spaces rather than a predefined series of orderly steps". Each space comprises different activities. The first is inspiration focused on the circumstances that lead to the innovation ("a problem, an opportunity, or both"). Example questions in this space are "what's the business problem?" or "where's the opportunity". The second is ideation that is the "process of generating, developing, and testing ideas that may lead to solutions". This space is more related to the ideas of brainstorming and prototyping. Generally, the process will loop through these first two space until the idea is refined. Finally, the last space is implementation when the solution is presented and the team can move on to the next project. In software startups, these steps can be used to foster innovation, new ideas that are very important for the organization.

2.2.4 Desing sprint

Design Sprint [BLW15] is a process developed inside Google Ventures, a Google subsidiary that invest in startups besides providing an environment so they can grow including tools, networking and even human resources[2]. Design Sprint is a five-days process to answer critical questions about business through design, prototyping and customer-based ideas tests. The process promises to any company the possibility to create and test virtually any idea in 40 hours.

It is described as stages performed in a five-day working week and what should be done in each day.

- **Monday:** unpack - during this day the knowledge from different areas should be shared.

[2]Available at http://www.gv.com/sprint/. Accessed 2017-01-16.

- **Tuesday:** sketch - each person should work by herself, developing in details an idea to solve the problem. After the sketches, the best ideas will be chosen through a weighted voting.
- **Wednesday:** decide - in this day, which solutions will have a prototype, how the prototype will be built and last day interview participants start to be chosen.
- **Thursday:** prototype - this day should be the highest working day when the team works to implement the prototypes decided the day before.
- **Friday:** test - the prototypes should be presented to the customers in interviews made one by one and learning should be done at most.

As a summary, Design Sprint brings some ideas from Design Thinking creating a step-by-step to be used in innovative teams. Since several times software startups are very innovate, Design Sprint could become a very important tool. Nevertheless, it is still very new and no other uses besides those described in the book were found.

2.2.5 Discussion

In summary, the methodologies presented although ranging from startups focused (like Lean Startup) to more general relating to innovation (Design Thinking) have some points in common. All of them advocate some kind of prototyping and listening the users more often. These elements attack the main reason given to the dot-com bubble: a long period investing in a product without any feedback from user. These points also are strictly related to requirements engineering.

Although several handbooks have been published and some methodologies are really discussed in the media, software startups seem to not been following them as found out on interviews performed by Bosch et al. [BOBL13]. The authors explain: "our interviewees confirms that it is very difficult to know how to work in a straightforward manner in early stage startups, and that operational process support, i.e. decision-making support, is limited". As a response to this problem, the authors propose the Early Stage Software Startup Development Model (ESSSDM) extending Lean Startup principles.

2.3 Requirements engineering

Requirements Engineering research is very well established based on a strong community. Cheng and Atlee [CAJ07] perform a description of the field state-of-the-art and research future directions. The authors divide the subject in the following fields: elicitation, modeling, requirements analysis, validation and verification, requirements management and evaluation-based research. In elicitation, works had focused on techniques to improve this activity including metaphors and personas, brainstorming and creativity workshops, feedback techniques (models, model animations, simulation and storyboards). In Requirements Analysis, works focused on well-formedness errors like ambiguity, inconsistency or incompleteness. And also techniques to help prioritize, visualize and analyze helping "a manager to select an optimal combination of requirements to be implemented". Research in Validation and Verification has focused on improving stakeholder feedback like animations, simulations and derived invariants. Evaluation-based research is described as an orthogonal to the previous works and "whose mission is to assess the state of the practice and evaluate proposed advances to the state of the art". As recommendations made by the authors we can mention: researchers should work with practitioners so they can have an understanding of the real problems that the latter face and RE researches should not neglect evaluation and empirical research. Another classification on requirements engineering research can be found in [Zav95].

Neill and Laplante [NL03] perform a survey to understand the state of the practice on Requirements Engineering. The participants were obtained from a database of then prospective, current and former graduate students of the authors' university. The authors concluded that their major finds were that formal methods are rarely used, ad-hoc practices do not impact quality, the waterfall model was still popular and object-oriented techniques were not dominant. The last results can be explained since the study is quite old and object oriented was not yet wide-spread. And maybe today the waterfall model is not so popular. But the two first conclusions may still be valid.

An even older work from Siddiqi and Chandra [SC96] draws attention to some important facts like requirements incompleteness, requirements engineering reconciliation between technical and social factors, tacit information elicitation, difficult to handle market-driven innovations through classical requirements engineering techniques, impossibility to make requirements correct at the first time and RE becoming more of a design and integration exercise. They conclude stating their belief that "the key mission for requirements engineering community is to continually narrow the ever-growing gap between research and practice".

Requirements elicitation is the focus of a work of Zowghi and Coulin [ZC05]. The authors present a list of techniques, approaches and tools to support this activity. The authors conclude that requirements elicitation depends on a large number of factors like "the system being developed, the stage of the project, and the application domain" and that "almost all projects a combination of several different techniques will be necessary to a successful outcome".

Milne and Maine [MM12] discuss the influence of power and politics in requirements engineering. They find that power operates through formal and informal channels and seniority does not necessarily translate into effective power. Other characteristics are important as sources of power like expertise and personal characteristics. They conclude that, in many circumstances, to come up with the best requirements "requires political as well as technical skills".

The agile methodologies use has been a concern on requirements engineering point of view since the beginning as pointed out on a position paper by Eberlein and Leite [EL02]. After years, a systematic literature review performed by Inayat et al. [ISM+15] identified 21 works on practices and challenges of agile requirements engineering. Although, according to the authors, "the uneven distribution of authors across geographic regions means that the empirical evidence reported by the 21 studies could not be considered generalizable." The practices are:

- face-to-face communication;
- customer involvement and interaction;
- user stories;
- iterative requirements;
- requirements prioritization;
- change management;

- cross-functional teams;
- prototyping;
- testing before coding;
- requirements modeling;
- requirements management;
- review meetings and acceptance tests;
- code refactoring;
- shared conceptualizations;
- pairing for requirements analysis;
- retrospectives;
- continuous planning.

As challenges of agile requirements engineering, the authors mention:

- minimal documentation;
- customer availability;
- budget and time estimation;
- neglecting non-functional requirements;
- customer inability and agreement;
- contractual limitations;
- requirements change and its evaluation.

An important study was conducted by Cao and Ramesh [CR08] when they extracted data from 16 organizations that employ agile practices to understand how requirements engineering practices are performed in this context. The authors identified the following practices:

- Face-to-face communication over written specifications: instead of formal requirements documentation, agile team rather use stories to define high level requirements that will be discussed in details with customers during development;
- Iterative requirements engineering: requirements are not pre-defined but emerge during development;
- Requirements prioritization goes extreme: the highest priority requirements are first implemented while in traditional methods besides business value also consider risks, costs and implementation dependency;

- Managing requirements change through constant planning: in each cycle finish, changes can be made to requirements, others can be added or removed to better fit customer needs;
- Prototyping: instead of creating formal requirements, agile teams create prototypes to validate and refine requirements although there is a risk of creating unreal expectations to customers;
- Test-driven development: since tests specify code behaviors, they can be seen as a requirements engineering activity as they become an explicit specification and not just a verification code;
- Use review meetings and acceptance tests: at the end of development cycles, meetings are performed to review implemented features and obtain feedback but also to verify if the project is running as expected - this practice leverages customer thrust in the team and allows problem identification as early as possible.

Fogelström et al. investigate the applicability of agile methodologies in market-driven software development [FSO10]. The authors compare properties of the two subjects and present findings from a case study. They warn about some misalignment between these two areas described below.

- **Development context:** agile methods are oriented towards a client (bespoke development) since from beginning and focused in a project instead of product management usual in market-driven products.
- **Business context:** agile methods assume that there is a client responsible to tell what brings her value in addition to finance development as opposed to market-driven products that the own company will finance their development.
- **Understanding of value:** since agile methods focus on a client, understanding of value is simpler considering the client will tell it meanwhile, in market-driven products, this understanding is more difficult demanding more complex tools like customer value analysis.
- **Assumptions about requirements:** agile methods suppose that requirements are vague or unknown in the beginning of the project but, for market-driven products, huge efforts could have been made previously to gather requirements.

The first three items can be observed in startups. Most of them develop a product to an open market (sometimes it does not even exist), the company itself must finance the development and understanding of value is really hard by the same reasons discussed.

Meanwhile, Sillitti et al. [SCRS05] investigate the differences and similarities between agile and document-driven approaches in managing uncertainty in requirements. The authors conclude that agile methodologies are more customer centric and flexible and they seems to provide better results at least in the relationship with the customer.

Another interesting work is a systematic literature review performed by Silva et al. [SMMS11] on user-centered design and agile methods. The authors identified "recurring themes and patterns of the most common activities and artifacts used by teams integrating agile methods and UCD [User-Centric Design]".

Discussion on requirements engineering can also be found in information systems literature like in [LME+12]. In this paper, Lockerbie et al. explore the impact of software requirements on system-wide goals.

2.4 Conclusion

As mentioned earlier, there is no specific work on requirements engineering in software startups. Then the degree to what the related work results can be seen in software startups is unknown. For instance, given agile requirements engineering practices found, will they be present in software startups? And how software startups handle market-driven requirements that most of them face in daily basis? This work will also add to the requirements engineering research: now applied to a new context and could be an inspiration to other different contexts not yet studied. This study secondary objective - to understand to what degree startups development methodologies are used in software startups - is very important to give a feedback to the authors and the community. Depending on how these methodologies are used, a better guide to the improvement of them can be figured out.

Chapter 3

Research design

This research aim is, essentially, to find out how a process or processes set (Requirements Engineering) is performed in a determined context (software startups). That is, it tries to understand a human behavior. According to Seaman [Sea99], human behavior is one of few phenomena demanding qualitative methods to be studied. Hence, a qualitative research is a natural choice.

The selection of the research method to be used is very important. Several elements should be taken in account like research question nature, researcher experience, study scope, etc. Myers [Mye97] mention four research methods for qualitative studies: action research, case study, ethnography and grounded theory. In the following paragraphs each of these methods will be presented and a discussion on their applicability to this study.

Davison et al. [DMK04] say: "the application focus of [Action Research] involves solving organizational problems through intervention while at the same time contributing to knowledge". That is, an action research design consists of a real-life problem solution development combined with a deep study of the related phenomenons. Then, this research method requires a deep involvement between the research and, in this case, the target startup. Although one of the secondary objectives of this research is to give advice to software startups, this research main goal is to understand how a startups works, that is, it is exploratory. In this sense, it would be more valuable to contact several different startups touching different contexts. Then action research would not be the best option to our research.

According to Easterbrook et al. [ESSD08], in software engineering research, "ethnography can help to understand how technical communities build a culture of practices and communication strategies that enables them to perform technical work collaboratively". This method is based on field observation and also demands a huge involvement with the target startup. As for action research, since there are several different startup contexts (different maturity stages, markets on which companies operate, startup staff experience level, etc.), it is important to explore several startups what would be very costly using ethnography without getting much better results.

The same issue would happen with a case study. Even in a multiple-case case study, it is expected to use a variety of different data sources. Besides that, as mentioned by Easterbrook et al., "the major weakness of case studies is that the data collection and analysis is more open to interpretation and researcher bias". This could be very problematic given the researcher inexperience. Better results could be obtained using a more defined set of tools that will guide the researcher getting results.

In this sense, our final option, grounded theory is well suited. Several data collection and analysis techniques are discussed by one of the authors of the method [SC90]. Besides that, grounded theory is "extremely useful in developing context-based, process-oriented descriptions and explanations of the phenomenon" [Mye97]. The objective is to build a theory grounding it in empirical data through continuous data collection and analysis. Then grounded theory seems like a good option to be used in this research: it gets good results for research questions similar to ours through a large set of data and give specific tools that would be a valuable guide to an inexperienced researcher. Also, grounded theory has been used in an important study in this subject: Coleman and O'Connor [CO08] studied how the software development process is formed in software startups through interviews with Irish software startups and used grounded theory to develop a theoretical framework for process formation.

Although grounded theory is a very suitable option, following it strictly is very hard for inexperienced researches. Then we chose to take it as a guide and basically use some of their analysis techniques instead of doing everything it presents. In the following section, data collection and analysis techniques chosen are described and why these choices were made is explained. Then, grounded theory analysis techniques are discussed and some choices made are explained.

3.1 Data collection

Given the research exploratory bias, it is important that the data collection techniques used in this study foster the arising of new facts that were not expected by the researcher. Myers [Mye97] mentions some techniques for collecting empirical data like "interviews, observational techniques such as participant observation and fieldwork, through to archival research". In this study, interviews are one of the best suited options since it makes possible to get in touch to several different contexts. A in-depth participant observation or fieldwork would be time-consuming without creating novel facts. Nevertheless, interviews in the startup workplace would be valuable to the researcher get some information to improve the research results. Archival research also would be time-consuming and could limit companies sample since some of them would not provide data due information security concerns.

Still according to Seaman [Sea99], interviews can be structured when the interview already has a closed set of questions that should be answered by the interviewee or they can be unstructured when the interviewer only have one or more topics about which the interviewee will talk. Between them, there are also semi-structured interviews when the interview has a prior question set but he is allowed to do other questions depending on how the interview goes. In this case, it is common to use a interview guide.

This last approach was chosen since it is possible to let the interviewee tell more details and novel facts but, using an interview guide, restrict unnecessary digressions that could make interviews too long and lacking important elements. Besides interviews, field notes could also be made by the researcher to complement data.

3.1.1 Interview guide

The interview guide was developed consisting of three parts. First, it was important to measure the interviewee background, knowledge level on software development and startup development methodologies, her role in the startup and about the startup itself: a little about its history and product. After that, the main part consisted of four open questions, each one on a requirements engineering stage. And finally, a feedback from the interviewee on the interview to gather any other information that s/he would find interesting for the research. The interview guide is presented in appendix A.

3.2 Data analysis

Once interviews and field notes represent data, it is important to select proper data analysis tools. Grounded Theory was developed by Glaser and Strauss and, according to them, "is the discovery of a theory based on data systematically obtained and analyzed in social research" [GS67]. As pointed out by Urquhart et al. [ULM10], there are several other definitions found in literature, nevertheless according to them, four main characteristics are common:

- the main goal is to build a theory;
- the researcher must avoid that her prior knowledge takes her to create hypothesis before the study - in this case, the research would be confirmatory;
- analysis and conceptualization are made throughout the process of data fusion and comparison where all data chunks are compared with existent categories to see if they create a new category, supplement a pre-existent or create a relation;
- "data chunks" are obtained through theoretical sampling when the researcher decides, from analysis, when he is going to do the next sampling.

After a book [SC90] published by Strauss, now in cooperation with Corbin, there was a clear divergence between both original authors that created two main streams on grounded theory. Heath and Cowley [HC04] perform a comparison between these two streams. The authors discuss the difference between induction (Glaser) and deduction (Strauss) emphasis, the different stages: substantive and theoretical (Glaser) vs open, axial and selective (Strauss) and finally theory discovery (Glaser) vs theory construction (Strauss). According to Kendall [Ken99], the axial coding was the core of the differences and according to him, Glaser considered the other stream more conceptual description than emergent theory. Nevertheless, according to Strauss and Corbin, one of the key factors to write a new book was the difficulty that new researchers found while using grounded theory [SC90]. Given the researcher little experience on the topic, this research is based on techniques described by Strauss and Corbin [SC90]. An interesting discussion on using grounded theory on requirements engineering research is presented in [JG14].

The authors propose three stages for grounded theory: open coding, axial coding and selective coding. This research used only open and axial coding. The final stage, selective coding, is when a theory is built. Since creating a theory is beyond our objective, selective coding will not be done. We expect that at the axial coding end we will have a conceptual framework that will meet our objectives. These two stages will be described in the following subsections and tools used during this research to perform this stage will be described.

3.2.1 Open coding

Open coding is the first stage in the process proposed by Strauss and Corbin [SC90]. According to the authors, "it is the analytical process in which concepts are identified and its properties and dimensions are discovered in data". In this context, it is important to define the concepts used by the authors:

- **Concept:** "is the abstract representation of a fact, an object or an action/interaction", that is, a "labeled phenomenon";
- **Category:** meanwhile concepts are accumulating during conceptualization, they should be grouped together behind more explanatory terms, the so-called categories.

Once the categories are defined, there should be specific characteristics to describe them. These characteristics are properties and dimensions.

- **Property:** "is a characteristic or attribute, general or specific, of a category";
- **Dimension:** "represent the location of a property along a line or a band". For instance, let luminosity be a property of a category, the dimension would vary from light to dark.

Still according to the authors, there are many ways to perform open coding depending on analysis granularity: from line by line, phrase by phrase, paragraph by paragraph or even to read the whole document. After reading first interviews, a phrase by phrase analysis was chosen but, sometimes, phrases fragments could also represent a concept.

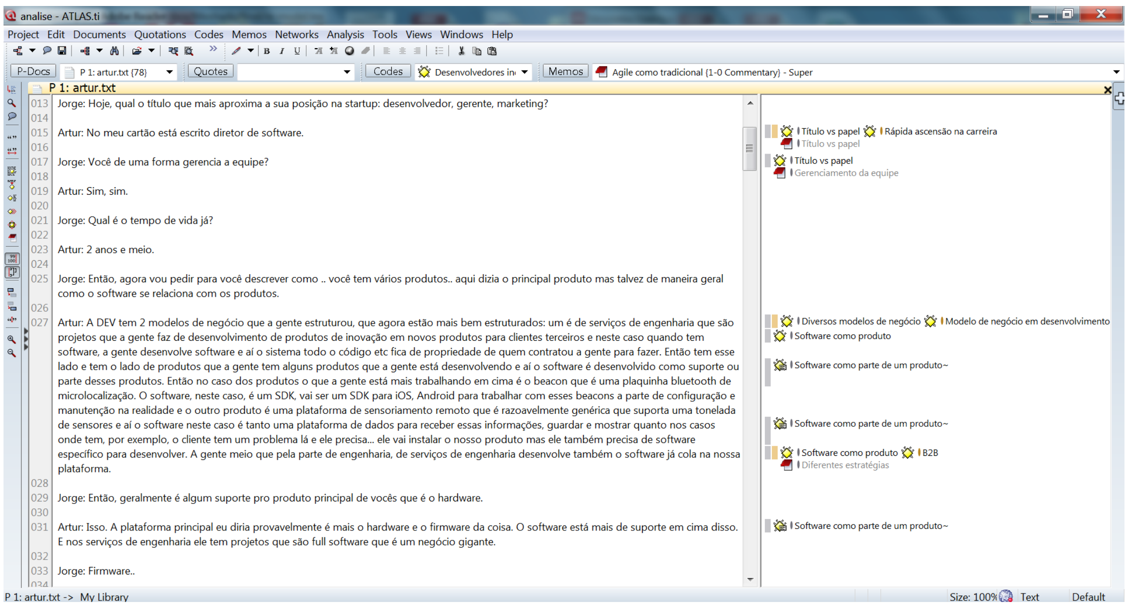
Open coding was made with AtlasTI tool[1] support. AtlasTI is a commercial tool developed to help qualitative research handling data and performing coding. The tool was used to add labels to interviews transcriptions and also group labels into categories. This was very useful because, for instance, after labeling the first nine interviews there were more than two hundred labels. In figures 3.1 and 3.2, the labeling and categories processes in AtlasTI are presented.

3.2.2 Axial coding

The process second stage is axial coding that, according the authors, "is the process of connecting categories to its subcategories, and is called axial because it occurs on a category axis, associating categories on properties and dimensions level". That is, during this stage, relations between categories including subcategories and cause-consequence are made and properties and dimensions are organized (this job already started in open coding).

During this stage, other features from the analysis tool were used to group labels together. These groups represented categories, subcategories, properties and dimensions. To picture categories and identify their relationships, a whiteboard was used to draw categories, relationships and their properties and dimensions. A whiteboard was chosen because drawing and erasing was easier and quicker than in software. Besides that, the whole model could be seen at once. A picture of the whiteboard is shown in figure 3.3.

[1] Avaible online on http://atlasti.com. Accessed 2017-01-16.


analise - ATLAS.ti
Project Edit Documents Quotations Codes Memos Networks Analysis Tools Views Windows Help
P-Docs
P 1: artur.txt {78}
Quotes
Codes
Desenvolvedores in
Memos
Agile como tradicional {1-0 Commentary} - Super
P 1: artur.txt
P 1: artur.txt -> My Library
Size: 100%
Text
Default

Jorge: Hoje, qual o título que mais aproxima a sua posição na startup: desenvolvedor, gerente, marketing?

Artur: No meu cartão está escrito diretor de software.

Jorge: Você de uma forma gerencia a equipe?

Artur: Sim, sim.

Jorge: Qual é o tempo de vida já?

Artur: 2 anos e meio.

Jorge: Então, agora vou pedir para você descrever como .. você tem vários produtos.. aqui dizia o principal produto mas talvez de maneira geral como o software se relaciona com os produtos.

Artur: A DEV tem 2 modelos de negócio que a gente estruturou, que agora estão mais bem estruturados: um é de serviços de engenharia que são projetos que a gente faz de desenvolvimento de produtos de inovação em novos produtos para clientes terceiros e neste caso quando tem software, a gente desenvolve software e aí o sistema todo o código etc fica de propriedade de quem contratou a gente para fazer. Então tem esse lado e tem o lado de produtos que a gente tem alguns produtos que a gente está desenvolvendo e aí o software é desenvolvido como suporte ou parte desses produtos. Então no caso dos produtos o que a gente está mais trabalhando em cima é o beacon que é uma plaquinha bluetooth de microlocalização. O software, neste caso, é um SDK, vai ser um SDK para iOS, Android para trabalhar com esses beacons a parte de configuração e manutenção na realidade e o outro produto é uma plataforma de sensoriamento remoto que é razoavelmente genérica que suporta uma tonelada de sensores e aí o software neste caso é tanto uma plataforma de dados para receber essas informações, guardar e mostrar quanto nos casos onde tem, por exemplo, o cliente tem um problema lá e ele precisa... ele vai instalar o nosso produto mas ele também precisa de software específico para desenvolver. A gente meio que pela parte de engenharia, de serviços de engenharia desenvolve também o software já cola na nossa plataforma.

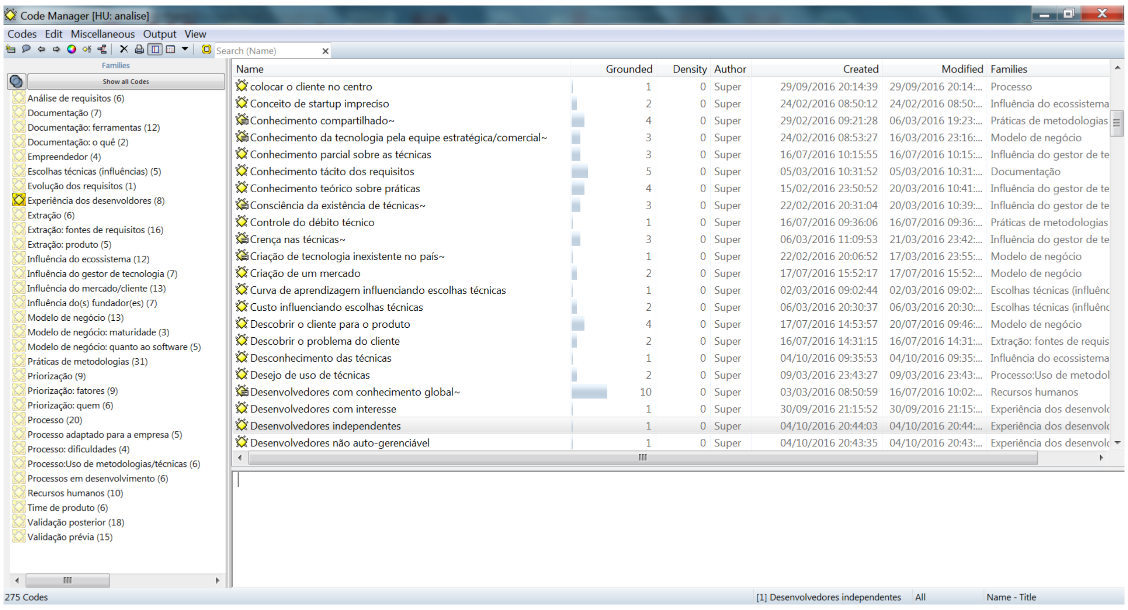
Jorge: Então, geralmente é algum suporte pro produto principal de vocês que é o hardware.

Artur: Isso. A plataforma principal eu diria provavelmente é mais o hardware e o firmware da coisa. O software está mais de suporte em cima disso. E nos serviços de engenharia ele tem projetos que são full software que é um negócio gigante.

Jorge: Firmware..

Título vs papel Rápida ascensão na carreira
Título vs papel
Título vs papel
Gerenciamento da equipe
Diversos modelos de negócio Modelo de negócio em desenvolvimento
Software como produto
Software como parte de um produto~
Software como parte de um produto~
Software como produto B2B
Diferentes estratégias
Software como parte de um produto~

Figure 3.1: *An interview transcription after labeling in AtlasTI tool.*


Code Manager [HU: analise]
Codes Edit Miscellaneous Output View
Search (Name)
Families
Show all Codes
275 Codes
[1] Desenvolvedores independentes
All
Name - Title

Análise de requisitos (6)
Documentação (7)
Documentação: ferramentas (12)
Documentação: o quê (2)
Empreendedor (4)
Escolhas técnicas (influências) (5)
Evolução dos requisitos (1)
Experiência dos desenvoldores (8)
Extração (6)
Extração: fontes de requisitos (16)
Extração: produto (5)
Influência do ecossistema (12)
Influência do gestor de tecnologia (7)
Influência do mercado/cliente (13)
Influência do(s) fundador(es) (7)
Modelo de negócio (13)
Modelo de negócio: maturidade (3)
Modelo de negócio: quanto ao software (5)
Práticas de metodologias (31)
Priorização (9)
Priorização: fatores (9)
Priorização: quem (6)
Processo (20)
Processo adaptado para a empresa (5)
Processo: dificuldades (4)
Processo:Uso de metodologias/técnicas (6)
Processos em desenvolvimento (6)
Recursos humanos (10)
Time de produto (6)
Validação posterior (18)
Validação prévia (15)

Name	Grounded	Density	Author	Created	Modified	Families
colocar o cliente no centro	1	0	Super	29/09/2016 20:14:39	29/09/2016 20:14:...	Processo
Conceito de startup impreciso	2	0	Super	24/02/2016 08:50:12	24/02/2016 08:50:...	Influência do ecossistema
Conhecimento compartilhado~	4	0	Super	29/02/2016 09:21:28	06/03/2016 19:23:...	Práticas de metodologias
Conhecimento da tecnologia pela equipe estratégica/comercial~	3	0	Super	24/02/2016 08:53:27	16/03/2016 23:16:...	Modelo de negócio
Conhecimento parcial sobre as técnicas	3	0	Super	16/07/2016 10:15:55	16/07/2016 10:15:...	Influência do gestor de te
Conhecimento tácito dos requisitos	5	0	Super	05/03/2016 10:31:52	05/03/2016 10:31:...	Documentação
Conhecimento teórico sobre práticas	4	0	Super	15/02/2016 23:50:52	20/03/2016 10:41:...	Influência do gestor de te
Consciência da existência de técnicas~	3	0	Super	22/02/2016 20:31:04	20/03/2016 10:39:...	Influência do gestor de te
Controle do débito técnico	1	0	Super	16/07/2016 09:36:06	16/07/2016 09:36:...	Práticas de metodologias
Crença nas técnicas~	3	0	Super	06/03/2016 11:09:53	21/03/2016 23:42:...	Influência do gestor de te
Criação de tecnologia inexistente no país~	1	0	Super	22/02/2016 20:06:52	17/03/2016 23:55:...	Modelo de negócio
Criação de um mercado	2	0	Super	17/07/2016 15:52:17	17/07/2016 15:52:...	Modelo de negócio
Curva de aprendizagem influenciando escolhas técnicas	1	0	Super	02/03/2016 09:02:44	02/03/2016 09:02:...	Escolhas técnicas (influênc
Custo influenciando escolhas técnicas	2	0	Super	06/03/2016 20:30:37	06/03/2016 20:30:...	Escolhas técnicas (influênc
Descobrir o cliente para o produto	4	0	Super	17/07/2016 14:53:57	20/07/2016 09:46:...	Modelo de negócio
Descobrir o problema do cliente	2	0	Super	16/07/2016 14:31:15	16/07/2016 14:31:...	Extração: fontes de requis
Desconhecimento das técnicas	1	0	Super	04/10/2016 09:35:53	04/10/2016 09:35:...	Influência do ecossistema
Desejo de uso de técnicas	2	0	Super	09/03/2016 23:43:27	09/03/2016 23:43:...	Processo:Uso de metodol
Desenvolvedores com conhecimento global~	10	0	Super	03/03/2016 08:50:59	16/07/2016 10:02:...	Recursos humanos
Desenvolvedores com interesse	1	0	Super	30/09/2016 21:15:52	30/09/2016 21:15:...	Experiência dos desenvolc
Desenvolvedores independentes	1	0	Super	04/10/2016 20:44:03	04/10/2016 20:44:...	Experiência dos desenvolc
Desenvolvedores não auto-gerenciável	1	0	Super	04/10/2016 20:43:35	04/10/2016 20:43:...	Experiência dos desenvolc

Figure 3.2: *Labels and categories in AtlasTI tool.*

3.2.3 Iterative process

At the end of axial coding, it is expected that a set of categories and subcategories within their properties and dimensions have emerged and help to answer how requirements engineering process is made in software startups. Nevertheless, it is possible that, after this process is done once, there are missings points where phenomenons or practices are not well explained and other rounds of data collection and analysis are needed. That is, this research process will be iterative. Actually, this is advocated by Strauss and Corbin [SC90]. This process will stop when new data does not bring any more novelty. This idea is represented in figure 3.4.

Actually, the iterative idea is present in the whole process even during stages themselves. That is, axial coding did not start after open coding ended. Actually it started after labeling few interviews

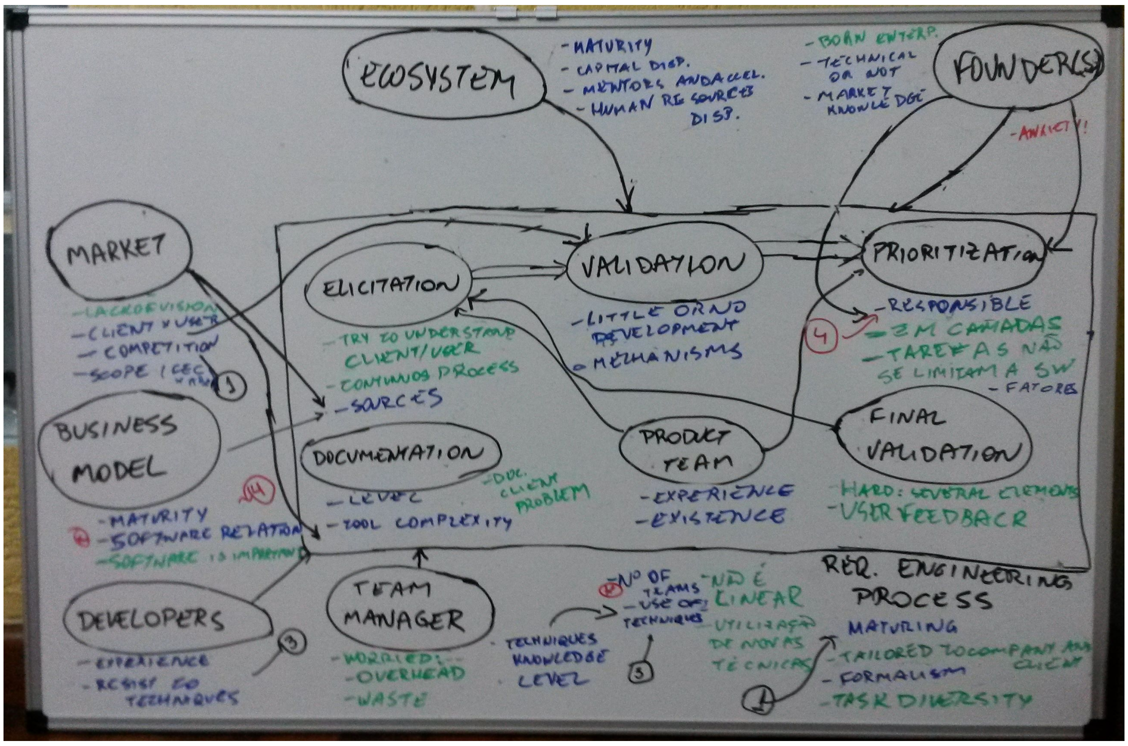


Figure 3.3: *Whiteboard used during data analysis.*

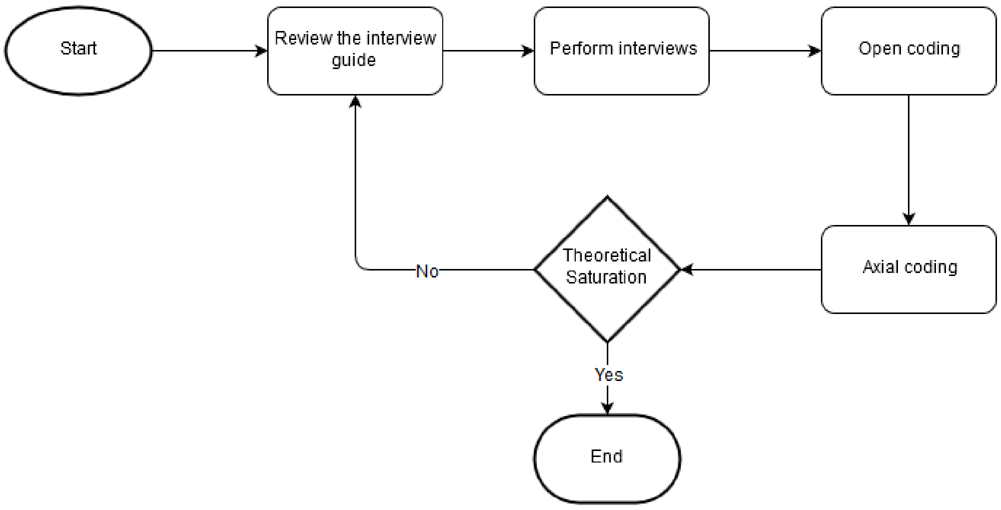


Figure 3.4: *Research process.*

and continued until the analysis end. Labels were also merged, split, grouped, had their group changed and all of these happened several times throughout several days. For instance, after the labeling of the first round of interviews, there were more than two hundreds labels. Then, an iterative process took place to refine labels and categories: some were removed, new were added and others were merged. This process happened due to a growing understanding about the data by the analyst.

In this chapter, the research design was presented. First, qualitative research methods were presented and their applicability to this research was discussed. Grounded theory was chosen as a guide. Data collection consisted of semi-structured interviews performed using an interview guide and field notes regarding observations were also taken when possible. Data analysis consisted of open and axial coding. Each stage consisted of data collection and data analysis and improvements could also be made to the interview guide. In the following chapter, the research execution is described including how many interviews were made in each stage and how the interview guide evolved.

Chapter 4

Data collection and analysis

In this chapter, the research data collection and analysis are presented. This element is important while reporting a qualitative study as described by Miles and Huberman [MHS13]. According to the authors, the reporter of a qualitative study should provide the "natural history of inquiry". In this way, the study can be repeated elsewhere and its credibility can be assessed. This point in also valued in empirical studies on requirements engineering as pointed by Daneva et al. [DDMP14]: "the more explicitly a research design is described, the easier it is for readers of empirical RE papers to evaluate the generalizability of the research being published". In section 4.1, a description of how the research evolved through time is presented and in section 4.2 a summary of the interviews performed is presented.

4.1 Chronology

The research reached theoretical saturation after three stages. Each stage was performed as described in the previous chapter: interview guide creation or improvement, interviews and data analysis. Besides that, another stage was performed before interviews used for data analysis. During this stage, the researcher tried to improve his skills and learn from other researchers that faced similar challenges. This stage is called as Stage 0 and is described in subsection 4.1.1. The following subsections describes details about each other cycle.

4.1.1 Stage 0

As the researcher had not prior experience with interviews and qualitative data analysis, this first stage consisted of three interviews with other researchers from the Department who had already made interviews and/or grounded theory analysis. This period took place between August and September 2015. During interviews, the researcher took notes about tips on each subject. The first two interviews were about interviewing process and themes discussed included: how to control the interview pace; creating a relationship during an interview to increase trust and foster details gathering and also some practical ones like to avoid noisy places. One interviewee said she has lost an interview due to a recorder problem and this induced us to use two recorders to have a backup. For the interview about grounded theory, it was important to get practical advices since books about the subject only give simple examples. One suggestion was to use pieces of paper instead of software to perform coding. Although this suggestion was not followed because it looked slower and would demand more space, it inspired the white board solution to picture the relationship between categories. Besides advice, interviews made during this stage were also important to give practical experience to the researcher in performing future interviews.

4.1.2 Stage 1

Once the research preparation was ready, it was time to reach possible interviewees. The interviewees list was created based on first author's contacts and other university students who own startups. We also used a snowball approach to get other possible interviewees. Some people were also contacted to name people that could be interviewed. Possible interviewees were contacted through emails. Arrangements were made to perform meetings. A meeting were preferred since this way the researcher could observe the startup working environment. But, after some contacts, this requirement became more flexible because of difficulties with interviewees' schedules. Then meetings and calls were performed.

A spreadsheet was created to handle emails with possible interviewees. This spreadsheet had rows grouped by a contact. Each group header contained the interview name, last sent email date, last answer received date and interview date. The line was also colored according to the contact status: green for interview done, yellow to waiting for an answer and red for failed contact. A contact was marked as failed after two emails sent without an answer or when the participation in the research had been denied. After the header, there rows representing emails sent, containing its date, its answer date and a brief summary.

Emails started to be sent in December, 2015 and two interviews were made then. But, most of interviews in this stage were made in January, 2016. In this stage, nine interviews were made and recorded using two devices to prevent data loss. In research plan, it was expected to do few interviews before start analyzing. This number of interviews should not be too small because data from different contexts is needed. But not too big because analyzing interviews could bring insights to the next interviews. Nevertheless, the contact process was very productive and several interviews were arranged in the beginning. Since people could get frustrated and not participate in the future if their interviews were postponed, it was preferred to make the arranged interviews and analyze a bigger amount of data.

Following an advice received during the stage 0, the researcher started transcribing himself the interviews. Although giving several insights and being a start to analysis, transcribing was too time-consuming. After four interviews, recorded interviews were sent to a third party to be transcribed. Nevertheless, the researcher still reviewed the transcription, listening again to the record and fixing transcription errors. This review process is the analysis process start. Both the spreadsheet and transcription processes were made throughout the whole research. In this stage, transcribed interviews summed up 72 pages.

In the beginning, some analysis tools used for grounded theory studies were evaluated. There were few open source solutions and they were still very simple. Some commercial solutions were really expensive like NVivo. AtlasTI was chosen because it has been used in other related studies ([JG14], [GPU$^+$15] and [KDR$^+$07]), it was considered simple to use and it was affordable through a student license.

The analysis then took place during February and March. The coding process consisted of labeling and categorizing labels using tools available at AtlasTI. This process consisted of doing these two activities throughout several days and sometimes more than once a day. This process favored a more in-depth contact with data by the researcher. Another activity was added using the whiteboard to picture relationships. Finally, these three activities: labeling, categorizing and connecting categories were made at the same time, forging a model. In figure 4.1 it is presented how the number of labels and categories evolved through ten days of analysis. This period starts after all files have been labeled once.

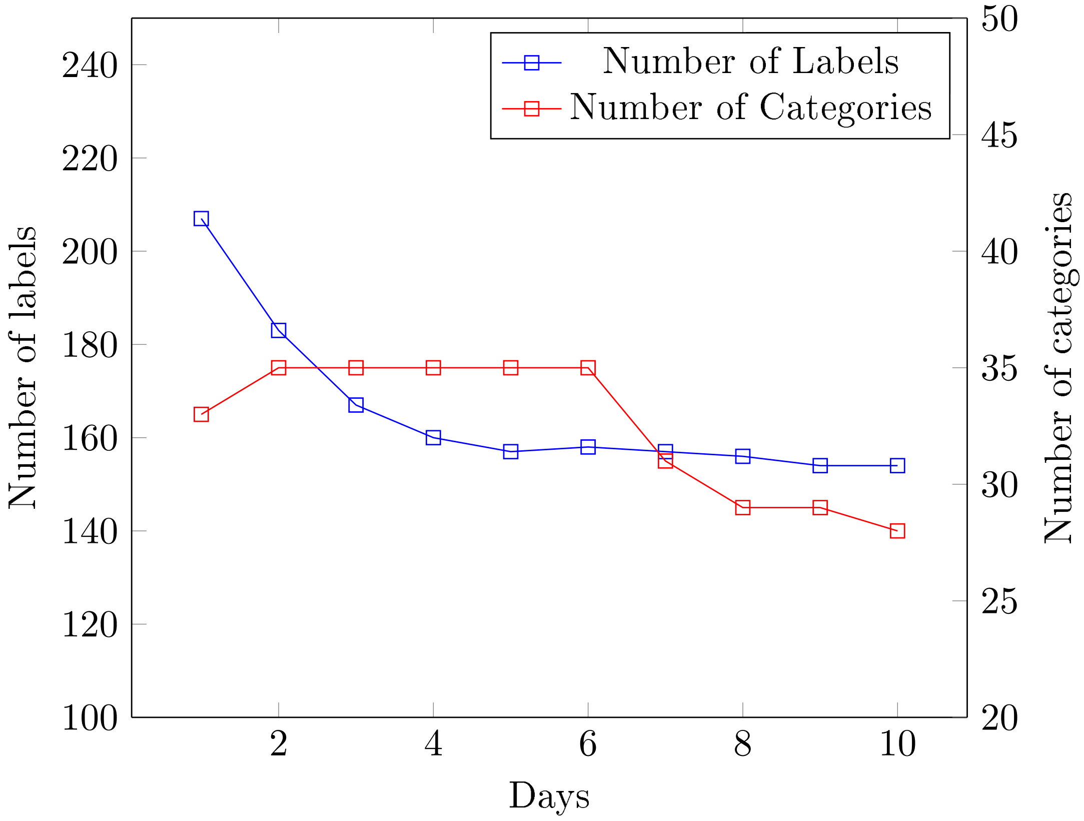


Figure 4.1: *Number of labels and categories through first stage analysis.*

By the beginning of April, a paper [MG16] was submitted to a workshop to present the results of this first stage. In figure 4.2, the model after this first stage is presented. In the figure, we can see that the influences on requirements engineering processes (founders, software development manager, market, business model, ecosystem and founders) were already identified. Though, some details (categories) were still to be found.

In this moment, the product team role was already identified and elicitation techniques were already described. Between elicitation and implementation phases, only prioritization has been identified. Validation and final validation stages were identified but these concepts were not so well detailed. Validation comprehended the stage when a requirement is prototyped or other kind of experiment was made to check if it is worth implementing. Final validation was the stage when the implemented requirements was checked if it was done correctly. The documentation stage was identified as how the team would write requirements details. In the following stages, this stage will comprehend also communication between team members.

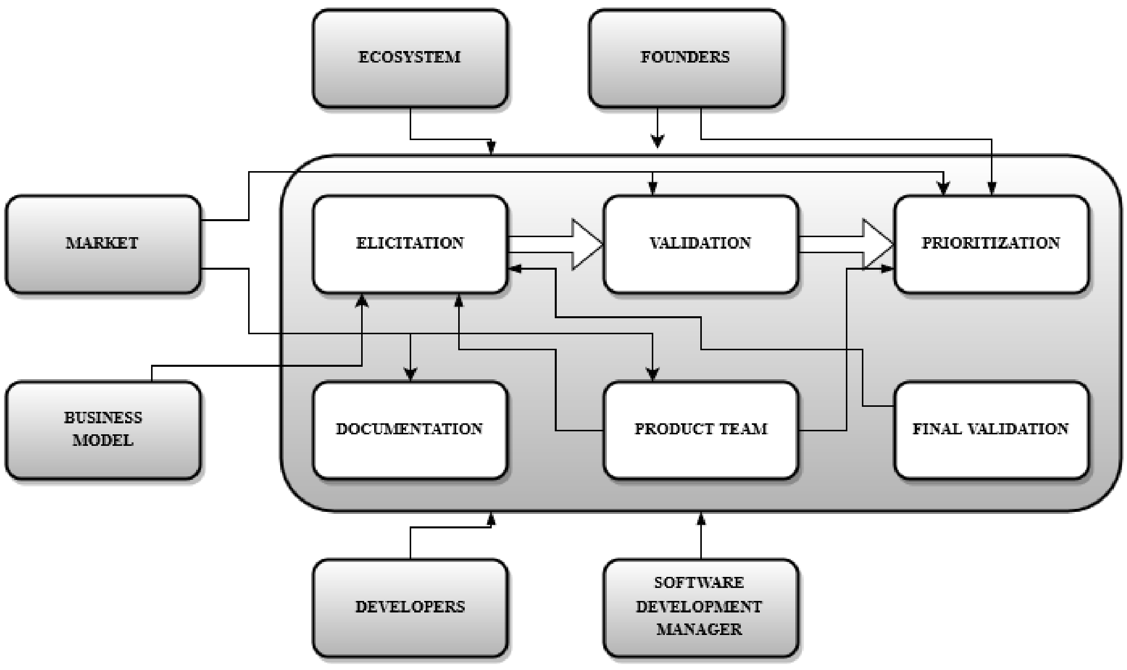


Figure 4.2: *Model after stage 1.*

4.1.3 Stage 2

The interview guide had a few changes after the first stage results. Mainly we stopped asking about Customer Development and Design Sprint since they revealed to be unknown in the context. To get more information about a possible product team or who performs this role, a question was added asking who is responsible for product guidance. We did not use the term product team to avoid an influence on the interviewee. Then a new round of interviews were performed in May, 2016. This round consisted of five interviews. Transcriptions of interviews made during this stage summed 47 pages. The data analysis was made during July and August. Few new phenomenon appeared. That indicates that theoretical saturation was close. First, more focus on requirements analysis was needed. In the first stage, it was resumed to a prioritization stage but it was clear that requirements are treated and evolved before implementation. Then, it was clear that analysis, validation and prioritization worked together to prepare a requirement to be implemented and represented a stage consisting of these three activities. Besides that, one of the interviewees when asked for suggestions for the interview mentioned requirements evolution. Although this concept is very important, developing it further would be beyond the scope of this work and will be left as a suggestion for a future work. Finally, other dimensions of some categories were also grounded in this stage.

4.1.4 Stage 3

Again the interview guide was modified but this time only a question regarding requirements analysis was added. This new round of interviews took place in September, 2016 and consisted of three interviews. As a result, more details about requirements analysis were gathered and for other aspects no new phenomenon appeared. It was concluded that theoretical saturation had been reached.

In figure 4.3, a summary of chronology is presented.

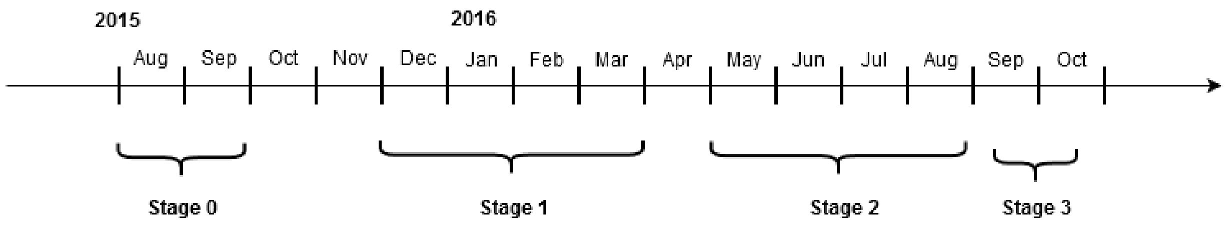


Figure 4.3: *Data collection and analysis chronology.*

4.2 Interviews

All interviews were in Portuguese and took between 21 and 48 minutes and, most of the time, followed the interview guide. While answering questions, the interviewee was set free to talk and tell his/her experience. Sometimes, the interviewee started to digress but the interview guide helped the interviewer to stick to the plan. The interviewees list is described in table 4.1 including interviews details like the research stage during the interview took place, the interviewee role in the startup and the market sector where the startup operates. The interview time summed 9 hours and 37 minutes and the transcriptions using Times New Roman font size 12 have 181 pages.

The startups that participated in this study are located in the cities of São Paulo, São José dos Campos and Campinas. São Paulo is Brazil's main economical city and its metropolitan area has more than 20 million inhabitants. The latter two are medium sized cities, located in São Paulo state and in a radius of less than 100 km from the city of São Paulo. All cities contain great universities and a flourishing innovation culture. The interviewees distribution across the cities were: São Paulo, 14 startups, São José dos Campos, 2 startups, and Campinas, 1 startup.

In most cases, they were companies running for a long time considering they are startups: 14 (82%) of them were running for 2 years or more and 3 (18%) were running for at least 1 year but less than 2. This long-running characteristic was important to the research since most of interviewees were founders or working for a long time in the startup, so they were able to give details about how processes evolved throughout startup history. Nevertheless, in the last stage, new startups were looked for to participate in the research. It was noted that even startups in an early-stage accelerator that we visited were running for some time. Some reasons could explain such effect: founders think they can do it by themselves or a so self-confident about their product in the beginning to search for help; or maybe they are not aware of the existence of such environments; or accelerators demand in their selection process a more well-defined team or idea. Anyway this discussion is beyond this study scope. Another valuable interviewee would be with failed startups. Although this was not made, by the time of writing, at least, two startups interviewed do not exist anymore. In both cases, there were not markets for the products being developed.

Interviewees' roles in startups varied from commercial/strategical roles such as CEOs, technical roles (technical leaders or CTOs) or product-related roles (products directors or managers). Before scheduling the interview, all were asked if they would be able to answer questions related to requirements engineering and software development. It is interesting to say that even CEOs said that they were able to do that. This indicates how important software development is important in a software startup. Of course, when a technical person was interviewed, more details about software development were available. In some interviews, more than one person attended. This was useful to gather more data and perform some kind of triangulation of views inside the company. In total,

Interview	**Research stage**	**Interviewee Position**	**Market sector**
I1	1	Founder and software director	Internet of Things
I2	1	Founder and software director	Automation
I3	1	Founder and CEO	Health
I4	1	Technical Leader	Real state
I5	1	CEO	Finance and Defense
I6	1	Founder and technical leader	E-commerce
I7	1	Mobile leader	Sharing economy
I8	1	Product Director	e-Learning
I9	1	Product Manager	e-Learning
I10	2	CEO, Product Manager and 2 Software Development managers	Advertising
I11	2	CEO	Virtual reality
I12	2	CTO	Social network and Data Mining (2 startups)
I13	2	CTO	E-commerce
I14	2	CTO	Specific office software
I15	3	CTO	Advertising
I16	3	CTO and CEO	Web
I17	3	Product director	E-commerce

Table 4.1: *Interviews*

23 people participated in interviews: in one interview four people participated.

In general, interviewees had a great experience on startups: 2 (12%) of them had worked from 2 to 3 years in startups and 14 (82%) had worked for more than 3 years not necessarily in the same startups. Only one interviewee (6%) had worked for less than one year in startups. At the same time, they had worked in few startups: for 5 interviewees (29,4%) it was their first experience in startups, other 5 it was their second startup. Finally, 7 (41.2%) have worked in 3 or more startups. In interviewees when there were more than one interviewee, it was considered the answer of the interviewee who talked more during the interview.

In this chapter, the research chronology was presented and why some decisions were taken was detailed. The model evolution was described and the interviewees list was presented including some demographics about startups and interviewees that participated in this study. In the next chapter, the research results will be presented including the model final version and details about the methodologies practices use and problems mentioned in the interviews.

Chapter 5

Requirements engineering in software startups model

In this chapter we present the conceptual model developed as a result of the research. In section 5.1, the context elements that influence the RE process in software startups are presented. In section 5.3 the process is described, in section 5.2, the product team role is presented and finally in section 5.4 requirements engineering practices are presented and how influences are related to them. In section 5.5, a summary of the model is presented. Finally, in section 5.6, practices from software and startup development methodologies that were mentioned during interviews and problems faced by these startups are listed.

5.1 Influences

Our main research result is a conceptual model that describes requirements engineering process in software startups. This model is composed by two components: context and activities. The context comprises the environment where the development team is inserted including elements that influence the whole startup, they are: *Founders*, *Software Developer Manager*, *Market*, *Business Model*, *Developers* and *Entrepreneurship Ecosystem*. The first three elements are closely related to categories identified by Coleman and O'Connor [CO08] as presented in table 5.1. In this section, the context influences are presented. The process is the requirements engineering process itself within its properties and practices and it is presented on section 5.3.

This model	**Software development process formation model**
Founders	Background of founder
Software development manager	Background of software development manager
Market	Market requirements and Market sector

Table 5.1: *Comparison between models.*

5.1.1 Founders

Founders are the startup creators and so they are very influent in how the organization works. They can have a technology background or not. In the first case, it is natural that they have an influence on how requirements engineering practices are done but in the latter case this also happens. According to table 4.1 on page 32, I4 told that his team used to have a Scrum based process. But it was abandoned since startup founders viewed it as a "waste of time". After almost an year, the team started again using a Scrum based process because the result without any methodology was worse.

Generally, founders are "born entrepreneurs" having founded several companies (I3, I6, I8 and I12), sometimes still running more than one (I3 and I12) and many times they have never worked for a consolidated firm.

Discussion. An influence of founders on software development has also been found by Seppänen et al. [SOL16] in interviews performed with European software startups. Even when the founder did not do any software development, "s/he participated at higher level: target setting, management and evaluation". Some engineering practices like architecture design may seem as a unnecessary for an outsider because they do not create code. Then even extremely necessary, they can be neglected to alocate resources in "more important" tasks. The founder background is also one of the categories in the process formation in startups by Coleman and O'Connor's model [CO08]. The authors also mention that the founders will influence the management style used in the organization. How much they trust their team will make them follow any of the two management styles: "command and control" or "embrace and empower".

5.1.2 Software development manager

As several interviewees told (I1, I5, I12, I13 and I16), the software development manager is responsible for process and architecture decisions. I13 who have already worked as a manager for several startups and owned others told: "In each context I made a composite of practices that will better fit there and the process will be based on that" And it is very common that s/he also performs programming tasks (I1, I4, I10 and I16). In this sense, s/he is responsible to model the requirements engineering process and choose practices to be used. Then, his or her knowledge and experience on requirements engineering techniques will be determinant to their use. For instance, s/he can:

- prevent a practice use: although not directly requirements related, for instance, I1 said that his team was not using pair programming because he had a bad experience with it and "lost his faith" on it;
- promote a practice use: I14 said: "[...] I've always intended to try simple and lean solutions to check if that feature would really add value";
- postpone practices use: I7 said: "I don't know if it makes sense for a startup at the beginning to invest more time in [learning] tools or more time in [developing] features".

Discussion. Again in Coleman and O'Connor's model, the background of the software development manager is very important in the process formation [CO08]. The manager can be a founder when he has a technological background and is selected to do so or someone hired in the startup history beginning. The authors states that "it was clear that where the software development manager had worked before, what their responsibilities were, what process and process improvement model was used[...] shaped the process that the software development manager used in their current company."

5.1.3 Market

The market where the startups operates is a very powerful determinant for the process followed. According to table 4.1 on page 32, I5 was a founder of a startup operating in the defense and financial markets. To be able to sell its products in these critical markets, the organization had to use very strict process since it was required and checked by clients.

A common classification to startups is dependent on their market - the B2B vs B2C. B2B stands to business-to-business and refers to product or services marketed to other organizations instead of consumers - the case of business-to-consumers (B2C). Although these classes work well for business and strategy concerns, they do not work well for the purpose of this work. For instance, I9 worked for a startup which product was a platform targeted to elementary and high schools, the users are the students, parents and teachers but the schools are going to pay for it. This example would be classified as a B2B startup: marketing, negotiation and operations are related to another business - the school - but the requirements will be related to the final users: students, parents and teachers. This situation is very similar to a classic B2C situation: there are several end users that have different expectations on the software. And it is also very different from B2B when the number of end users is much inferior like a startup that was developing machines to agricultural automation.

Hence, it is necessary another classification in the context of requirements engineering comprising the difference between businesses that have a large number of customers and the others that have a few clients. In this classification, we distinguish between user-targeted startups and client-targeted startups.

- **User-targeted startup:** when the target market is a large number of users even when a company or organization is responsible to pay for the product or service for large groups of users. B2C startups are the common examples of this situation: e-commerces, other websites, mobile apps, etc. But it can also happen to a B2B startup when there is a large group of users. For instance, the school who pays for students, teachers and students' parents to use a software.

- **Client-targeted startup:** when a client or a few group of them are clearly identified and they are easily reachable. This is the case for B2B companies who develop a specific solution and generally have few clients. For instance, the agricultural automation startup who adapts its solution depending on the client.

The market types influence how requirements elicitation and analysis are performed. These difference are detailed in subsections 5.4.1 and 5.4.2.

Discussion. Coleman and O'Connor [CO08] also mention demands specific from a market. In their model, there is a category called *market requirements* and an example used is very similar to our interviewee who operates in defense market. On type of customers, a similar discussion is made by Eberlein and Leite [EL02]. They call users and clients two types of customers. In their view, users are individuals who interact with the system and clients are those that will have their needs fulfilled by the software system. [Bla07] also states the difference between user, people who will actually use the system, and the economic buyer, the one who has budget allocated to spend on the product. In the high school example, students, parents and teachers are users and the school principal is the economic buyer.

A user-targeted startup handles market-driven requirements while a client-targeted startup handles customer-specific development. There are several works in literature about market-driven requirements engineering and its challenges. According to Regnell and Brinkkemper, in the market-driven situation "a software producer develops a product that is offered to an open market with many customers" [RB05]. The authors also recognize differences between market-driven for consumer and for enterprise concerning, for instance, usability issues and product image. Besides that, market-driven elicitation is focused on new requirements combined with market analysis while customer-specific, also called bespoke development, is focused on negotiation and conflict resolution. Requirements analysis is also different as mentioned by Potts [Pot95], "one party does not always 'elicit' requirements from another, nor does it 'play back' requirements so that the other may accept. reject or refine them".

Karlsson et al. searched for which challenges market-driven requirements engineering imply for companies dealing with it [KDR+07]. They conducted 14 interviews and a focus group with practitioners. The challenges identified were:

- marketing and development team communication;
- easy comprehension requirements documentation;
- managing the new requirements constant flow;
- requirements volatility;
- requirements interdependence and traceability;
- requirements are invented rather than discovered;
- implementing and improving requirements engineering within the organization;
- resource allocation to requirements engineering;
- selecting the right process;
- organization stability;
- release planning based on uncertain estimates.

Since challenges faced by market-driven software development and startups we called user-targeted are very similar, we can expect that results in literature could be applied for this type of software startups.

5.1.4 Business model

The startup business model will also influence requirements engineering process and how strong other context elements will be through two different dimensions: **maturity** and **reason for software**.

Maturity. The business model can still be under development, that is, the organization has not understood its market yet and how to make profit. Or the business model can be more mature and the organization has already a business model already complete. In the first scenario, founders participate more in requirements engineering because it can change if the startup will success or fail. I8 said that founders take part in prioritization meetings because what the development team does can affect the company revenue on that month. A similar behaviour has been identified in I1, I3, I4, I6, I9, I12 and I13 (see table 4.1 on page 32). On other hand, if the business model is more mature, it is possible that breakthrough requirements become less common and most of development time is spent on fixing bugs and small improvements. This scenario was observed in I5, I7 and I10.

Reason for software. This influence is related to the reason why the startup develops software. Software can be a product itself like in the example of agricultural automation - a pattern recognition software was responsible to detect when the vegetables are good or not or when the startup develops a mobile game. On the opposite direction, software can only be a way to achieve a business goal. For instance, an online real state agency. Generally, when software is not core, it is very important to create some kind of competitive advantage. For instance, the online real state agency pretended to make the rent process simpler than a traditional real state agency using technology. These differences will reflect on who will tell what the requirements are. If the software is the product itself, the requirements are clearer, but in the other case, the business rules will dictate software requirements.

Discussion. The difference on software startups maturity is discussed by different authors. Wang et al. [WEB+16] try to understand which challenges are faced by software startups in different development stages. They use two development series of stages: problem/solution fit and product/market fit. First, the problem/solution fit is obtained through a process of learning and follows the steps: defining or observing a problem, evaluating the problem, defining a solution, and evaluating the solution. Second, the product/market fit is obtained through a product development process that follows the steps: concept, in development, working prototype, functional product with limited users, functional product with high growth, and mature product. One of identified concerns is problem solution fit and according to the authors this is considered "the biggest challenge faced by software startups at all stages". But they also recognize: "market related challenges such as *customer acquisition* and *scaling* become increasingly perceivable". These findings are closely related to the business model influence in our model since while the concern is building the product, most of requirements are related to this task and while the product development evolves other tasks become more common.

Another work towards an evolution model for software startups was made by Nguyen-Duc et al.[NDSA15]. The authors propose a model based on the Cynefin framework and a design thinking model to achieve this goal. The model consists of representing two different types of activities: hunting (activities in the chaotic domain) that are related to "generating ideas, elicitation requirements and market and customer development" and gathering that are related to "requirements description, prototype implementation, automated testing, system integration and deployment". The authors then describe startups development through picturing the volume of these two type of activities in a plane which axes are early-later phases and unknown-known. Generally, in early phases, hunting activities are greater than gathering activities. As the company evolves, hunter activities decrease and gather activities grow. Figure 5.1 is an example given by the authors.

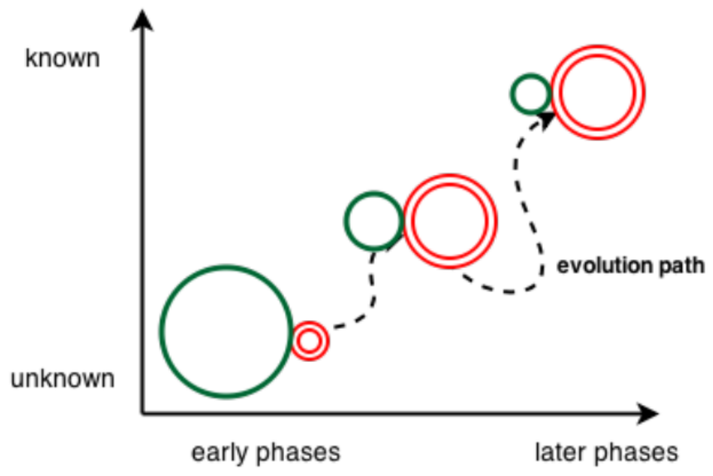


Figure 5.1: *Hunter-Gatherer evolution model for startups. Extracted from [NDSA15].*

In non-scientific literature, there are also frameworks to understand a startup evolution. Croll and Yoskovitz in their *Lean Analytics* book [CY13] propose five stages for a startup: Empathy, Stickiness, Virality, Revenue and Scale.

As mentioned in section 1.1, an innovation can be revolutionary or evolutionary or somewhere between [AST06]. This also plays a role for a startup. Häsel et al. [HKB10] identify four different competences profiles (IT manager, e-entrepreneur, web developer and e-business expert) and concludes that the founders preference between them while hiring people depends on how innovative are the startups' products.

5.1.5 Developers

In software startups, developers play an active role in software development practices selection. Different from consolidated firms, they are listened more often because they have an equity share. Or just because it is hard to startups attract talents and they are treated very well to not leave. We must remember that talents are rare (as mentioned in several interviews like I8 and I14). In this sense, they can:

- refuse to use some techniques: they do not believe the techniques will work or just do not want to perform them. While explaining why he does not use the process he wants, I1 said "to implement concepts I just can't say: guys here it is. I have to go slowly in such manner".

- bring their experience and suggest the use of some techniques: I7 mentioned this phenomenon. I2 said that a former team member used design thinking while in the company because this employee had a previous experience with it.

Another aspect mentioned by interviewees was that they prefer developers that have interest in the business rather than those that just do what is asked for them.

Discussion. Kajko-Mattsson and Niktina [KMN08] while trying to improve the process of startups recognize that developers bring their experience and suggest techniques. Chow and Cao [CC08] verified agile software development project key success factors. In their study, hypothesis concerning team enviroment and team capability were supported. Agile-friendly project team environment was one of elements inside these categories. Nerur et al. [NMM05] also discuss people related problems while migrating to agile development methodologies. Vijayasarathy and Turk [VT08] say that "it is intriguing that personal interest is, by far, the most important factor influencing the agile adoption decision". Given the close relation between software development and startup development methodologies and since requirements engineering is also part of software development, it is expected that this influence exists.

In information systems literature, individual innovation orientation and its influence on the organization innovation competence is also explored. This point is discussed by Nambisan [Nam02]. He focus even more the individual manager as a key component of entrepreneurial success and give examples where successful innovative software products were created by individual developers like Lotus Notes and Java. The understanding of business by software developers and manager is also described as key factor for innovation in [GT07].

5.1.6 Ecosystem

Besides the market that a startup operates, there are other external forces that influence how requirements engineering process are made. They are related to the entrepreneurship ecosystem where the startup is. The ecosystem is formed by startups, entrepreneurs, angels and venture capital firms and universities [MW14]. In this work, we identified three elements that influence requirements engineering process concerning the ecosystem: knowledge spread, human resources availability and capital availability. Each element plays an influence as detailed below:

- **knowledge spread:** related to how much contact startups had with methodologies or stories of other successful startups and entrepreneurs. For instance, acceleration programs and mentors provide knowledge to founders and teams favoring techniques use. According to table 4.1 on page 32, I3 said: "we started at [an acceleration program] and they taught us startup things, MVP and since then our target is to build a MVP".

- **human sources availability:** relates to how difficult is to find people able to work for the startups. The lack of human resources was mentioned several times in interviews. For instance, I8 said that his company had money to hire more people but it was not possible because they were not finding people with the required knowledge. Without proper team size, developers will have to work more, deadlines will longer or harder to achieve. Founders and/or investors will pressure to remove "unnecessary" practices.

- **capital availability:** relates to how much money the startup has and how difficult is to get more capital to grow. In Brazil, it is very hard to get capital to create a startup. A symptom identified in interviews was that several startups run more than one product or run a main product and offer consulting services on the same field. This is necessary because they lack capital and perform these activities to survive. The author had an experience in the Silicon Valley and this approach is discouraged there: you must focus on only one product.

Unfortunately, interviews at Silicon Valley would be necessary to give more details and it would beyond the scope of this study. Working in different threads makes the process more complex and will increase team workload leading to similar effects as described in the previous item.

Discussion. In literature, we can find works comparing ecosystems like Suzuki et al. [SKB02]. In this work, the authors conducted a survey comparing entrepreneurship in Japan and Silicon Valley in four dimensions: *entrepreneurial motivation*, *risks and obstacles*, *perceived growth factors* and *supporting infrastructure*. They concluded that Japanese entrepreneurs were more society-oriented, more concerned about personal and globalization risks, focused on strategic and R&D and have better access to bank loans and government financing. Meanwhile, Silicon Valley entrepreneurs were more motivated by individualistic reasons, more concerned on market and financial risks, focused on growth-orientation, customers and capital inflow and have better access to professional services and venture capital.

Cukier et al. [CKK15] propose a scale to assess a ecosystem maturity. In this sense, it is important to compare São Paulo ecosystem, where all startups interviewed are, to other ecosystems. The scale consisted of the following marks: Nascent, Evolving, Mature and Self-Sustainable. According to the authors, São Paulo is still in an Evolving stage that is there are "a few successful companies, some regional impact, job generation and small local economic impact". Meanwhile Silicon Valley is a Self-Sustainable ecosystem that have "thousand of startups and financial deals, at least a 2nd generation of entrepreneurs mentors, specially angel investors, a strong network of successful entrepreneurs[...] and presence of high quality technical talent". This difference can explain some results and also indicates future works replicating this study on different ecosystem maturity levels. Another problem mentioned was lack of capital. Wonglimpiyarat [Won06] concludes that venture capital financing is one of catalysts for the region economic development.

The previous elements exercise influence on requirements engineering process in software startups from in specific activities or the whole process. On figure 5.2, they are shown and will be discussed in more detail in the following sections.

5.2 Product team

As an actor in requirements engineering process, it is common to see a product team figure. This team is not a software development team neither its members are developers. Nevertheless, they work very closely to the development team and its role is to understand the product goals and to conduct its development. The product team is generally present when the product is user-targeted and it acts like a proxy to real users. They can use different tools that were already mentioned like interviews, surveys and focus groups. The team execute most of requirements engineering stages, specially, elicitation and prioritization phases. This function can also be performed by the founder since s/he generally knows a lot about the market.

In interview I10, the product manager talked about his role: "Part of my job is to be a filter... Even inside the company, I see my role as a filter because we cannot do everything. [...] I have to understand what the real problem is and tackle from only one side and from all". The interviewee

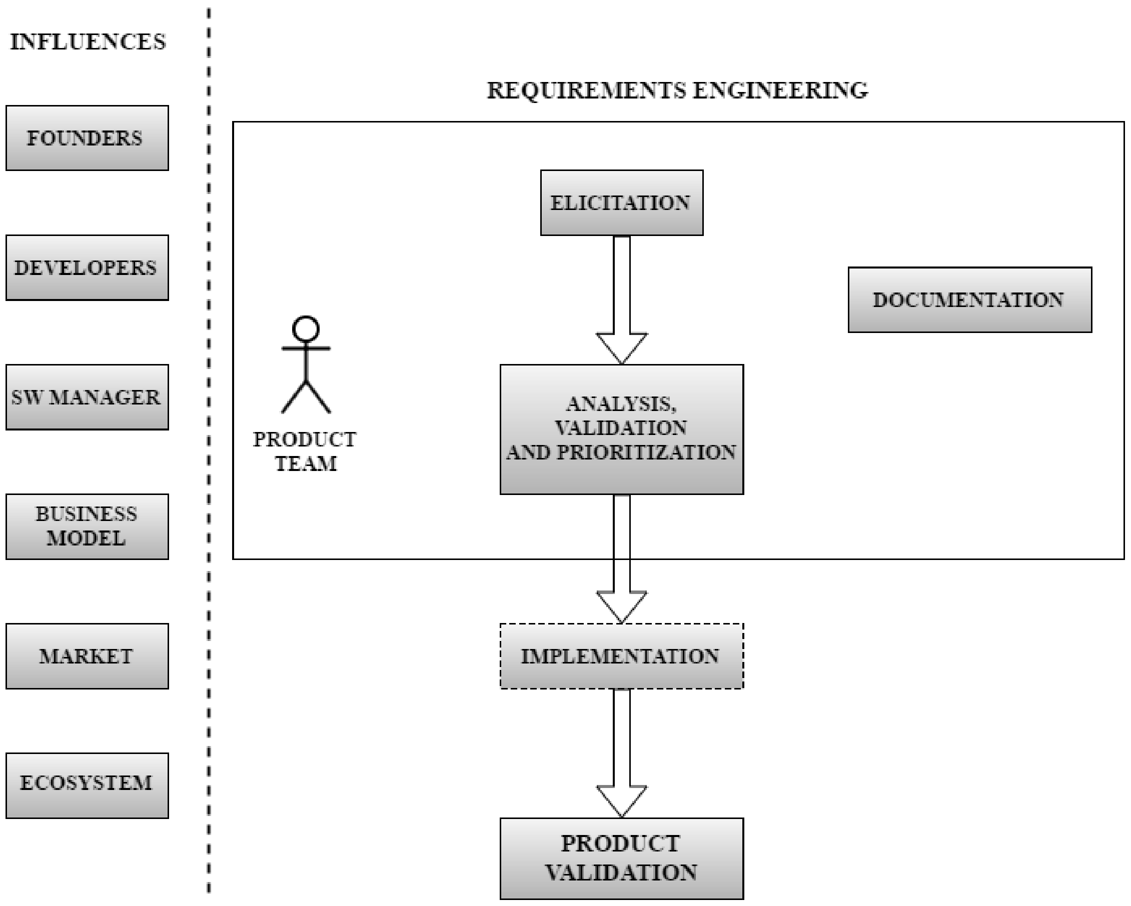


Figure 5.2: *Requirements engineering process model.*

I7 said: "the product team makes some brainstorm sessions". A CTO (I12) said: "we have two guys concerned... trying to develop the product. [...] there must be someone responsible because it is very important. In the beginning, there were no one and everyone was lost. Someone should be responsible for the product to have a vision".

Product team experience is very important for methodologies and good practices use. I7 told differences she has found while working with two different product managers. When a more experienced person took care of the product, "he used AB tests and was very good with it" but when not so experienced people perform this role, she complained that they should gather more data but he was not doing it.

Discussion. Similar roles are present in agile methodologies like Scrum and Extreme Programming. In Scrum [Sch04], the role of Product Owner is to write customer-centric stories, prioritize them inside the Product Backlog. Although originally described as a only person job, the figure of Product Owner team was described in [DSCE08] and [Bas13]. De-Ste-Croix and Easton [DSCE08] says: "this team provided the direction for the product development" and it is exactly the same for our interviewees. For Extreme Programming [BA05], "product managers write stories, pick themes and stories in the quarterly cycle, pick stories in the weekly cycle, and answer questions as implemention uncovers under-specified areas of stories". And "a product manager doesn't just pick a bunch of stories at the beginning of the project and then sit back". It is interesting, though, that none of interviewees that mentioned product teams as something from a agile methodology. They see product team as a natural structure, something that is common sense. Blank [Bla07] distinguishes the product development team and customer development team. The first will implement, develop, assemble the product itself and it is related to a traditional product development process. The latter is the one that will try to understand what should be built. According to the author, it is important that this team has "the capacity to put themselves in their customers' shoes". This customer development team is very similar to what is called product team in the startups interviewed. Nevertheless, Blank highlights that the product development team should not cease to exist, instead both teams should work in parallel. This role can be performed by the founders since they already have knowledge on the market and that is why they created the company in the first place. Seppänen et al. [SOL16] mapped software startups initial team competencies and their results call attention that the "founder tends to be the sole owner of the innovation and its related competency domains."

5.3 Process

The process as a whole had some important characteristics that have been grounded. First of all, software startups do not follow a strict methodology. Instead, they build a process through gluing techniques from different methodologies. As I13 (see table 4.1 on page 32) who have already worked or owned several startups told that he had made a composite of practices that will better fit that startup. A common behavior is to follow the example of an successful startup. The most mentioned example was Spotify[1].

Furthermore, the process is not set at advance and followed indefinitely. Instead, it is changing and improving as the team learns more about the product and the team themselves. Generally, this improvement is achieved incorporating new practices as the team grows in number (I2) or when someone proposes a practice (I7). But it is also possible to regress and stop using any practice, for instance, when a startup stopped using Scrum as reported before. An interesting example is given by I3 that feels that they are not following the best possible process and they can make it better but they prefer to focus on creating the product first.

It was observed a great knowledge spectrum on software and startup methodologies. Some interviewees only heard of Lean Startup that is already a well-known subject. Nevertheless, there are others that know several methodologies and try to apply concepts learned on their startups like I14. More details about mentioned practices are presented in section 5.6.

Discussion. The practice of adapting a methodology to the startup and adding or removing practices was already observed by Coleman and O'Connor [CO08] - they called it *process tailoring*. Actually, this is already discussed in literature before like when Zettel et al. [ZMMW01] proposes a lightweight process for "e-business startups companies" since the processes used then were "immature and ad-hoc".

5.4 Activities

In the following sections, the different activities taken place during requirements engineering process will be detailed. It is important to highlight that the process is not as linear as it looks like

[1]The Spotify development process is presented in https://labs.spotify.com/2014/03/27/spotify-engineering-culture-part-1/ and https://labs.spotify.com/2014/09/20/spotify-engineering-culture-part-2/. Accessed in 2017-01-16.

after these descriptions. Actually, a requirement has different paths throughout the process. New requirements are not created only in the beginning but also during analysis, validation, implementation and a final product validation. A requirement also can return to analysis phase during any other following stage. And of course, many requirements may not be implemented after all because during validation it is detected that it is not worth implementing. The first subsection discusses the product team that was several times mentioned in interviews and have a big role in the process.

5.4.1 Elicitation

During elicitation stage, requirements that software can implement are obtained. But this is extremely hard for startups since they are building something innovative and users may not know what the software will do. One approach used by some startups is to map problems and not solutions proposed by clients or users. As I9, a product manager, said: "(...) several times, people come with the solution but the solution always change, the idea is always map the problem to never lose the point". And I9, a CEO, mentioned: "The client has a problem. Instead of saying that he has to solve that problem he comes with a solution. (...) But this is wrong. How can he imagine a solution better that someone who makes a living of it?" So, the most important objective during this stage is to understand the user or client, that is, their problem.

Nevertheless, understanding your audience is very different if the market is client or user targeted. In the former, the team only have to ask to the clients what is their problem and talk to them until reaching a good understand of the situation. But, in the latter, this cannot be done. There are several users, the team will not be able to reach them all and each user will have a different view of what is most important. Then the product team appeared to act as a proxy to the real users. In elicitation, the product team is responsible to understand users and market and to guide product development, that is, which features will be developed and how the product will be positioned in the market.

During interviews, several requirements sources were mentioned. They are:

- business objectives analysis (I4, I8 and I10);
- use of competitors products (I9);
- ideas from developers or product team (I10);
- must-have requirements, innate to the product ordering a product when developing an e-commerce platform (I13);
- from sales team, specially on client-targeted startups (I3 and I5);
- development team, specially when represent features not visible to the client or user (I4 and I11).

Some techniques used to elicit requirements were also mentioned like:

- user interviews (I4, I7, I8, I9, I11, I12 and I13);
- brainstorm sessions (I7);

- ideation process (I8).

One behavior that was observed through all the companies is that the elicitation process occurs continually. That is, ideas come from everywhere and are continuously created. Besides that, since time-to-market is important and vital to organization financial health, bugs or other problems always appear. This phenomenon creates a continuous flow throughout the whole process. In other words, the next stages will also occur several times because of these always fresh requirements.

Finally, another factor that will influence on how requirements are created is why software is made in first place. The business model software reason discussed in subsection 5.1.4. If software is the product itself, the pre-requirements (ideas, bugs, etc) are more close to the final requirements. But when software is just a mean to reach a business goal, software requirements demand more time and thinking to get from product requirements.

5.4.2 Analysis, Validation and Prioritization

In this subsection, we comprehend several activities that are performed to requirements since their elicitation until they are implemented. These activities comprehend:

- requirements analysis and detailing, including its viability and alignment to startup view;
- validation, that is, to test if that feature will be useful to the user, and;
- prioritization, that is, determine in which order requirements will be developed.

Although textbooks, like [SK98], usually separate these activities we chose to discuss them together because they represent adjustments from elicited requirements before implementation.

Requirements analysis comprises different tasks performed over the requirement preparing it to be implemented and it is very different in each startup. The most common activity mentioned is the discussion on the idea (I3, I15 and I16, see table 4.1 on page 32). During this moment, developers and product managers try to understand which is the best option to implement that idea into a feature or product and how validation could be done. During this stage, an alignment between strategy and technology is discussed (I13, I14, I15 and I16). The feature scope could be reduced to fit a faster release (I17). Consistency check was only mentioned twice (I5 and I15) despite being mentioned many times in requirements engineering textbooks.

Validation is a key activity for software startups. During this moment, the team will check if a requirement is something desirable to the user. Common methodologies discussed in this context like Lean Startup [Rie11] and Design Thinking [Bro09] advocate that the minimum possible should be developed to test the user desire towards the product or service. Although knowledge about these methodologies is limited, as discussed earlier, most of interviewed startups perform some kind of validation of their requirements using little or no development. In a client-targeted startups I1 said "generally we perform a validation before implementation through user interface. We develop a wireframe using what we understood from the requirement and check with the clients". In a user targeted startup I4 said: "we tried to make a test before implementation with a solution wireframe". I8 told an interesting history about a feature that before being added to their website was made by humans to check if it was worth implementing.

Some techniques to validate a requirement were mentioned like:

- MVP (I3, I4, I10, I11 and I14);
- mock-ups (I4) and prototypes (I1, I2, I3, I5 and I11);
- surveys (I16 and I17) and focus groups (I17)

Other related technique also mentioned was smoke tests (I5). The term "smoke test" comes from software development and is "a relatively simple check to see whether the product "smokes" when it runs" [McC96]. The interviewee used the term a marketing technique to check if there is an interested market for the feature as mentioned in non-scientific literature[2]. Another interviewee (I6) said that they check if someone has already made something similar before implementing.

Interviewee I9 said that validation before implementation is risk dependent. That is, if it is something simple, it is not worth testing, but if that feature development will take many resources, then a validation stage takes place.

A mentioned potential problem (I3 and I4) while using validation techniques like MVP is related to the company image to customers. Developing a mock could be seen as a low quality or less commitment to the product and this feeling can make users or clients go away.

Prioritization. To get a requirement implemented or even to analyze or validate it an order has to be defined. Hence, prioritization is being performed always. This was detected both in the case of sprints use (I1, I4, I10 and I14, Scrum influence) or when a prioritized list is maintained (I9 and I12, Kanban influence). A task list containing other tasks not related to software development was also mentioned by I1 and I6. In I2, this was justified because of the small team size.

In startups interviewed (I1, I4, I13), prioritization takes places in two layers: in a upper layer, founders and/or higher managers discusses which milestones will be more important to the organization. This happens because of the importance that software development represents as already mentioned in section 5.1.1. Once milestones are selected and sent to development team, the team or the product team when present are generally responsible to determine the task order to reach those milestones. In client-targeted startups, prioritization is simpler and comes down just to ask to the client as mentioned by I2, I11, I12 and I14.

It was also important to understand which aspects were taken into account while prioritizing. The following elements affected prioritization:

- firm strategy was mentioned by I7, I8, I9, I12 and I13. In the same sense, I2, I3 and I4 said that to develop features to demonstrate the product was the most important;
- value to the user mentioned by (I12 and I15);
- prevent blocking other teams (I1);
- essential features (I16) or high priority situations like critical bugs (I14).

A cost-effective analysis was also mentioned by I14.

[2]One example is https://www.startinno.com/blog/2015/4/8/smoke-testing-what-is-it-and-why-should-you-care. Accessed in 2017-01-16.

5.4.3 Product Validation

After implementing a requirement, a final validation stage takes place. We called it product validation. For startups, it is more important to understand if the requirement correctly satisfies an user desire. This moment is very important for startups: after the software is ready, the learning cycle about the market is closed and market knowledge is derived. Requirements correctness can be effectively tested. That is if what has been implemented is what the user has desired or the client was expecting. For instance, features implemented may not be used (I14) and even a long time has been committed to developing it (I14). Through user feedback and technical analysis, new requirements can emerge like new features, bug fixes or code refactoring, re-feeding the process. Mechanisms to evaluate user interaction with the product through metrics using tools like Google Analytics are very common (I1, I3, I4, I6, I9, I11, I12, I13 and I14). I3 recognized: "[a final product validation and metrics analysis] are very important for us because change how we develop our final product. Our current goal is precisely to iterate over the product with user feedback". In a client-targeted startup, the client is asked if the implemented feature is coherent to what s/he asked (I1, I2 and I12).

Generally, several changes to the product can reach production stage at the same moment and this is described as a difficulty in the whole process. It makes hard to isolate effects caused by each feature and the impact of them is hard to measure as mentioned by I9.

It is interesting to tell that failure is seen as natural. I6 told that several times had thrown away programming hours even performing validation before implementation. When asked if it can be avoided, he said that "I do not think so, it is going to happen things like that". This attitude agrees with Ries [Rie11] that sees failure as part of the learning process.

Discussion. Actually, verification and validation are discussed in requirements engineering literature like in [CAJ07] but they are more concerned about the requirement itself. That is, if the requirement description is valid and verified.

It is interesting to note that startups very often only realize that something built is not what the users wanted in the end of the task. Even though validating assumptions as soon as possible is present in all well-known startup development methodologies and listening to the client is also mentioned in agile methodologies, founders and teams still continue to fail in this task. It is beyond this study's scope to understand why this happen. Nevertheless, knowledge does not seem to be the only reason. Even people aware of these methodologies still made these mistakes. Maybe it is lack of discipline or just an effect of time pressure to achieve goals. Anyway, it is something that compromises startup success rate.

5.4.4 Documentation and Communication

The documentation stage is a set of practices that occurs during all requirements engineering process. But more than *document*, the biggest concern here was to *communicate*. The main reason to create different artifacts was to spread knowledge about users problems and solution proposed throughout the organization.

The documentation level varied substantially through interviewed startups. There are startups that do not have clear practices to keep track of requirements and just count on emails and contracts (I2, see table 4.1 on page 32). On the other hand, the startup that operates on defense market (I5) has a strictly documenting process as a result of market requirements. However, most of startups actually do some kind of requirements documentation generally using simple tools like physical boards or electronic tools simulating them. A tool mentioned several times was Trello[3] that is a simple web-based SaaS emulating a Kanban board. Sometimes, bigger teams also use medium complexity tools like issue trackers or agile project management tools. One startup had tried a traditional project management tool but it did not worked out. This might have happened because of the agile mindset in the team and this was not targeted by the tool.

One important thing that was observed in some startups was that the knowledge is tacit. As I5 said "we did not spend much time documenting because information is in people". This can be very problematic at least in two situations: when the company grows and newcomers must come to others to get information, delaying development process and when a person leaves the company and takes with her information without spreading it in the team. Nevertheless, simple tools help communication between different teams (including non-technical) since they do not demand learning a new tool. This fact was mentioned by I16 and I17.

Discussion. The tacit knowledge problem was also found by Valtanen and Ahonen [VA08] while improving processes for small software companies. The authors say: "in a small organization the information is often transferred in discussions and the documentation is neglected".

5.5 Summary

In previous sections, requirements engineering process was presented including a set of influences that determine why the practices are done this way. Throughout the text, several pieces of interviews were added to illustrate and to ground our conclusions. Although not the same, startups generally do the activities described in traditional requirements engineering: elicitation, analysis, validation and documentation. These activities are not homogeneous across startups. The process is generally not linear but, instead, it is common to come back and forth through different stages. Nevertheless, a linear process flow described in the previous section is represented in figure 5.3 that summarizes the activities.

The influences on requirements engineering process in software startups can be grouped in three groups:

- **Human:** consisting of people related influences that dictates which practices will be used, they are: *Founders*, *Software Development Manager* and *Developers*.
- **Business:** consisting of business related influences that dictates how the practices will be used, they are: *Business model* and *Market*.
- **Ecosystem:** consisting of the entrepreneurship context where the startup exists including accelerators, venture capital funds, universities, government, entrepreneurs, etc.

Finally, table 5.2 summarizes the influences and describe how they affect the processes and activities in software startups. Market influence spawns different set of practices therefore it is more detailed in table 5.3.

[3]Avaible at https://www.trello.com. Accessed 2017-01-16.

Influence	**How?**
Founders	Most of the time, they know the market the startup is operation in and may be responsible to guide the product development. Since development is important to the startup survival, they care a lot about the process. Sometimes they prevent the use of some methodologies practices because they see it as a "waste of time".
Software Development Manager	S/he can be a founder or someone hired in the beginning of the company. S/he is responsible for architectural and process decisions. The manager is the person that will determine which practices will be used.
Developers	Since talent is very important to startups but they are scarce and hard to attract, the developers have great influence on which practices the startup will follow. They can bring their expertise from elsewhere, slow down the adoption of practices or deny their use.
Business Model	There are three ways the business model impact the process. First, how innovative is the business model, if it is disruptive, there will be a need for more attempts to understand what the user will want to pay for. If the innovation is related to an existent market, the problems will be easier since what the product should do is more clear. Second, the reason why software is developed also matters. If software is just a mean to achieve a business goal, requirements are really closer to the business and, for instance, their validation will demand more effort. Meanwhile, if the software is the product itself, requirements are closer to the implementation and validation can be made easier. Finally, how developed the software is determine the presence of founders or other high managers. That is, when the product is in an early-stage, they will be more present than when the product is more mature.
Market	The difference between user and client targeted startup determine different set of practices. This fact is explored in details in table 5.3.
Ecosystem	The ecosystem influence through three aspects: knowledge, capital and human resources. Through acceleration programs and other knowledge spreading initiatives (university courses, entrepreneurs groups), the awareness of methodologies like Lean Startup grows and founders know the challenges they will face and can be prepared to them. Capital availability dictates if the startup will be able to focus on its main product or will have to fight to survive through creating side products or services. It will also impact the time pressure the team will face. Finally, the availability and quality (knowledge on practices) of human resources will dictate which practices will be followed.

Table 5.2: *Summary of influences on requirements engineering process.*

5.6 Used practices and problems

A secondary objective of this research was to map at what extent startup development methodologies are used in software startups. To achieve this goal, as showed in the interview guide A, one of the parts of the interview was dedicated to picture the interviewee background including

Activity	Client targeted	User targeted
Product team	Most tasks can be accomplished asking to the client	There are several users and a product team acts as a proxy
Elicitation	Easier: just ask to the client(s)	Harder: ideas can come from anyone and a deep user investigation may take place
Analysis and Prioritization	Made within the client	Internal discussions
Validation	Simpler: the client is already asking the features	Harder: use of metrics, prototypes, MVPs

Table 5.3: *Summary of differences between user and client targeted startup.*

his knowledge on these methodologies and her use of them. Since agile methodologies for software development are also considered a good option for startups [Tai10], the interview guide also goes into these questions.

In a general way, no startup followed strictly a methodology except the one that operates in critical markets. This result is similar to others in literature [CO08]. Instead of that, techniques from different methodologies are used for product or software development according to the startup needs. The practices mentioned are presented in table 5.4. Some practices are more common and they are discussed below.

On product development subject, although not fully implemented, it is common that interviewees had some level of knowledge of Lean Startup. Almost all of startups run some kind of Build-Measure-Learn cycle. Some really implement the MVP (Minimum Viable Product) concept. They try to develop an experiment to test if an assumption about what the user want is really true: define a metric that will be used throughout the process, develop the minimum necessary to test that hypothesis, measure the results and learn from the results. One interviewee told a story about a new process that was made manually first to check if it would take more users to sign-up to the product and after the result was positive, the automatic process that would demand software development effort was built. Other startups just mentioned less complex tests like prototypes or mock-ups. Generally, these experiments are made by client-targeted startups or as a intermediate step for internal verification if something being built is the expected. In this latter case, some tools to create wireframes are commonly used.

Still on product development, from Design Thinking [Bro09], ideation process was mentioned by one startup. In the same sense, brainstorm sessions were also mentioned by another startup. In both cases, the intent was to foster new ideas to develop the product and were performed by the product team.

Another question during the interview was related to problems faced by startups mostly concerning requirements. The most common problem is how to follow a requirement performing all activities at them and understanding how long they are taking to be implemented. Another re-

Practice	**Original methodology**	**Mentioned by**
Automated tests	XP	I5, I8 and I9
Backlog	Scrum	I4
Brainstorm	Design thinking	I7
Code review	XP	I10
Continuos integration	XP	I4, I7, I8, I9 and I10
Ideation process	Desing thinking	I8
Informative workspace	XP	I7, I8 and I9
Limit work-in-progress	Kanban	I6
Metaphor	XP	I1
MVP	Lean Startup	I10, I11, I14 and I16
Pair programming	XP	I10, I12 and I15 said it was used eventually.
Planning poker	XP	I4 and I10
Prioritized list	Kanban	I9 and I12
Sit together	XP	I1, I2, I3, I9, I10 and I16. It was observed on these interviews. Others could also be like this but it was not observed because they were interviewed through a call.
Small tasks		I6 and I7
Sprints	Scrum	I1, I4, I10, I13 and I17
Technical debt control		I10
Validated learning	Lean startup	I7
Visualizing locks	Kanban	I7

Table 5.4: *Practices mentioned in the interviews.*

lated problem was how to estimate the time to implement a requirement. Time pressure was also mentioned as something that make more difficult to follow a process.

In this chapter, the data analysis results were presented detailing a model on the state-of-practice of requirements engineering in software startups. In the first section, context influences were discussed and in the second section the activities. A discussion is made in third section and finally, in the fourth section, practices used in the interviewed startups are presented and problems faced by startups are mentioned. This chapter aim was to describe the situations startups face. After this research where seventeen interviews were performed and several pages of data were analyzed, some behaviors that led to non-optimal processes were clear. These behaviors are presented in the next chapter using the bad smells metaphor.

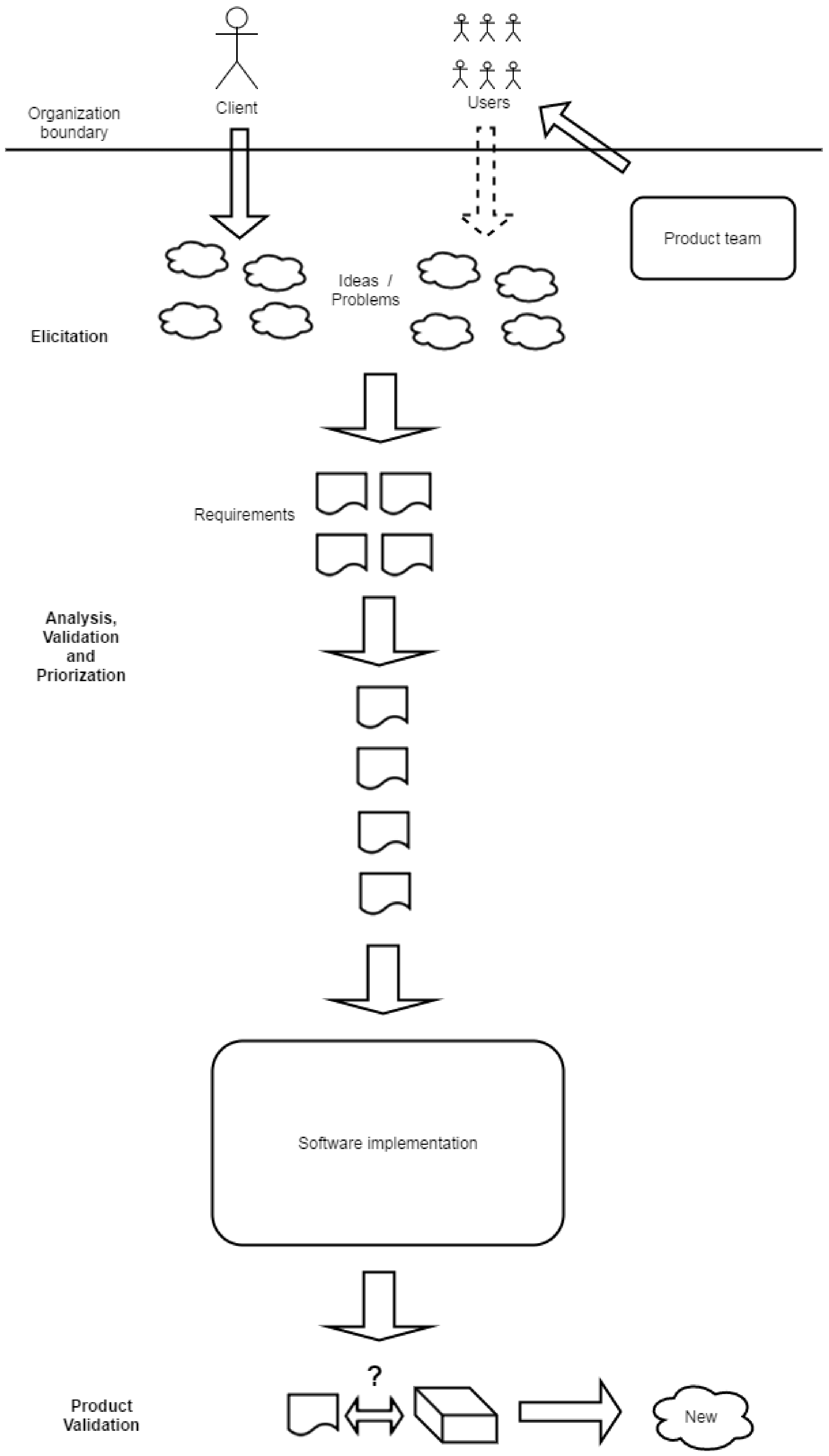


Figure 5.3: *Requirements engineering process.*

Chapter 6

Bad smells and recommendations

A well-known problem for startups is failure. Although several startups have good proposals and ideas, they fail before reaching their commercial potential [Cro02]. And the failure rate is high: Wang et al. mention that 75% of funded startups fail. As a consequence several studies in literature try to tackle this problem by understanding why startups fail or providing better practices to avoid failure.

Crowne [Cro02] lists reason to startup failure according to its stage: startup, stabilization, growth and maturity. In startup stage, reasons are *Developers are inexperienced*, *Product isn't really a product*, *Product has no owner*, *No strategic plan for product development*, *Product platform is unrecognized*. In stabilization phase, they are *Founders won't let go*, *Development team fails to gel*, *Product is unreliable*, *Requirements become unmanageable*, *Product expectations are too high*, *Service provision delays development*. In growth stage: *Skill shortage delays development*, *Platform creep delays development*, *Product pipeline is empty* and *No process for product introduction*. Finally, in maturity stage, "product development is robust and predictable with proven processes".

Giardino et al. [GWA14] develop a behavioral framework to understand why early-stage software startups fail. They performed a literature review and a multiple-case study. The startups researched by them focused on development of the product and its marketing instead of understanding, in the first moment, the user problem and the solution fit to it.

On the other hand, several works try to instruct startups to use better practices. An interesting option to format these suggestions is a pattern structure. A good definition of what a pattern is was given by Gordijn and Akkermans [GA03]: "a pattern describes a problem which occurs over and over again in an environment, and describes one or more solutions for the identified problem as well as consequences (e.g. trade-offs) as a result of applying the pattern". Patterns were initially proposed to architecture by Alexander et al. [AI77] and first used on software development by Gamma et al. [GHJV94]. Some works on startups patterns are presented now. Hokkanen and Leppanen [HL15] propose patterns that validate the startup hypothesis through user involvement. They are Validate product with the right people, Knowing what to ask about the product and Validate usability before measuring. In another paper [LH15], the authors proposed four more patterns: 20% Rule, Incubator, Internal startup and Exit. Leppanen [Lep14] proposed three patterns to start a startup: Self-funding, Venture capital and Validated product. In a related work, Ruseva [Rus15] proposed the patterns: The core of every business and Options for software. Cukier and Kon [CKK15] also proposed patterns to build a startup from scratch.

Eloranta [Elo14] worked towards a pattern language for startups and described three patterns

(Unique value proposition, Serve single customer segment first and Develop only what is need now) and proposed seven others that were not detailed in the paper. In the same sense, we have worked in the same direction [MG15] describing the patterns: Practice agility, On giants' shoulders, Simple tools, Early validations, Empower, In the coulds and Failure can be a good thing.

Although several patterns works were found, code smells, another metaphor used in software development literature ([MVL03], [FBZ12]) could also be used. This metaphor was also used for software architecture ([GPEM09]). Luo et al. [LHC10] discuss the relation between anti-patterns and bad smells in code. Fowler and Beck, in their book about code refactoring [FB99], created the concept of bad smells of code. Bad smells are "certain structures in the code that suggest (sometimes they scream for) the possibility of refactoring. According the authors, this is possible because they had seen lots of code.

After the interviews performed during this research and the contact with several other startups during ecosystem events, the author was able to do a parallel: identify some behaviors and practices that suggest that a startup is not in the right path. In this section, bad smells that were identified are described including elements from interviews performed in this study or literature elements that grounded these smells. Some recommendations are made to prevent or fix these bad behaviors. In table 6.1, bad smells discussed are presented and to which requirements engineering model concepts they are related to.

As concluded by Paternoster et al., the transference of results to industry is weak in software startups [PGU+14]. The bad smell metaphor seemed like a good option to help transferring this research results to the industry. It is not a closed methodology that contains a comprehensive number of practices. They are only pieces of information to help founders, managers and developers on fixing bad behaviors that could lead to bad results or even failure. Since it was observed that startups use only some practices of methodologies and according to their needs, the direct applicability of these pieces of advice should be very useful.

Bad smell	**Model concept(s)**
Know-it-all founder	Founders
Stubborn developer	Developers
Closed-minded team	Market, Business model, Elicitation
Blocked Ideas	Elicitation
Validate and do not waste	Validation, Ecosystem
Not spread knowledge	Documentation and Communication

Table 6.1: *Bad smells and related concepts from model.*

6.1 Know-it-all founder

The startups founders are the most concerned about the startup future. They spend time and financial resources and sometimes they do not even have a fixed income until the startup starts to make revenue. Generally, they create the startup to operate in a market they are already familiar with (I3, I4 and I6, see table 4.1 on page 32). From it all, it is common that founders interfere in the software development process even when they do not have a technological background. And if the development does not follow the expected speed they will try to change the used process although practices are used successfully in several other teams. Then, it is common that development teams in startups follow a "just-do-it" approach that is doomed to failure.

This problem happened in I4. The technical leader interviewed told that the startup used to follow a Scrum process. One year before the interview took place, the founders found that it was too cumbersome and called it "a waste of time". But, since then, the product quality fell and the team morale was also low. So by the time of the interview, they decided to go back and to follow a process.

Founders must control their anxiety and let their development team produce software according to the process they feel appropriated to their case. There is a big knowledge body about process and practices on software development and how to achieve better processes. Even when these practices seem slow, they have a purpose. Founders should trust their software development manager and team and allow them to be more independent. And this is only possible if they hire the right people: committed to the project and with good skills. Of course, they should not leave the team without any supervision. A good advice could be to entrepreneurs willing to create a software startup to read a little about software development. There are lots of good introduction books on the subject out there.

6.2 Stubborn developer

Startups lack financial resources and struggle to reach market as soon as possible to start creating an income and also to avoid any possible competitors. In this context, processes and practices used by a startup team and specially by a development team must take this into account. The software development team cannot do whatever they want and forget that the whole organization depends on them. Maybe they will spend much time trying to do the best solution. As already mentioned, the development team is one of the key factors to agile success ([CC08]).

The opposition to techniques or practices was mentioned several times. I1 has said that he could not tell team members to use a practice at once. He has to be patient and make it happen slowly. I4 told that "several team members do not like meetings so they decided to avoid them". I9 said it was really hard to designers to iterate in small cycles.

Sometimes, the code will not be the best possible nor the architecture will be perfect nor the best tools will be available. Developers and the software developer manager should be aware of best practices, be open-minded and think globally. They should know that, generally, agile is a better approach for startups and there are product development techniques developed specially to startups. Not only developers but also everyone involved in startups creation (managers, founders, others teams' employees) should be aware of these methodologies and take actions to increase familiarity with best practices: participate in meetings, read books or other materials, etc. From these attitudes, not only the startup but the ecosystem will also benefit and then society as a whole with greater companies.

Unfortunately, a good environment not necessary leads to the company success. I6 told that his development had "a very good energy and it was very good". Nevertheless, the company struggled to get customers and the team that was made of six people turned to only one. But, as mentioned by I4, the software quality and team morale went down without good practices and a well-known process. Several discussions on this subject can be found on literature like [Rie11].

6.3 Closed-minded team

It is common that at the startup launch, founders have an idea and pass it to developers to implement. Just after a long time spent on development, they realize that the solution was not being used. Although the methodologies mentioned advocate against this behavior: Blank [Bla07] mentions several examples in this book, startups still follow this path.

The interviewees I1, I10 recognized this error. I1 said that they had an idea and thought that it would be the right thing to do. When they did it, they realized it was wrong. Then they started to work on side projects to survive until they found a sustainable product. The founder in I10 told that they learned Lean Startup "in practice before the book was published". They had worked for two years in a project, developing and selling, and realized that nobody wanted to pay for it. Then, they had an idea, developed marketing artifacts for a week, sell it and then they started to develop the new product. Bosch et al. [BOBL13] also found this problem in their study.

Startup founders, managers and employees must listen to their users. This can be achieved directly through user interviews, feedback forms and other contact ways or indirectly through experiments and metrics. A closed startup that do not examine its environment including its market is doomed to failure. They should be open and try to learn about their users through feedback. Sometimes lessons could be controversial to what insiders are thinking [Rie11] then the latter must rethink about the subject. A pivot can also occur and should be taken naturally.

6.4 Blocked ideas

Usually, there is a responsible for the product guidance or even a product team that is responsible for that. Then they can think that only them can know what should be done next. Nevertheless, innovation is very important and leaving to only one person or small group to think can bias product development. In this sense, startups should foster intrapreneurial behaviors among their employees. According to Antoncic and Hisrich [AH01], intrapreneurial behaviors are attitudes taken by employees innovating and seeking new business opportunities for the whole without being asked to do so. And one way to increase these behaviors is empowerment. This correlation has been supported by studies like [VMM16] and [RW13]. Also, according to [ABG+10], managerial support and tolerance for risk taking have a positive and significant impact on innovation performance. Gordon and Tarafdar [GT07] highlights the fact that the creativity of a team surpass the sum of the individuals capacity and also that business involvement are essential to innovations since they are the incorporation of knowledge into a product, service or process.

Since innovation is so vital, elicitation techniques and policies should allow and foster anyone inside, or even outside the company, to give ideas and suggestions. This is specially true when the product or service is used by all.

6.5 Validate and do not waste

It is common that ideas are taken for granted in startups. A great idea looks like the solution that will disrupt the market. Even though it demands a lot of resources, it is implemented and put in the market. Then, most of the time it does not work as expected. This story is the main motivation to the creation of startup development methodologies like Customer Development and Lean Startup. Nevertheless, there are still startups that commit the same mistake: develop a solution that nobody uses. This phenomenon was also observed by Bosch et al. [BOBL13].

Startup team can decide to not follow any methodology but they must be understand what problems these methodologies are trying to solve. And the most important problem is validation. Validation is very important to reduce development waste even more when resources are scarce. Development should be made in small stages as big as it makes possible to test if the idea is correct.

Most of startups interviewed in this research performed some kind of validation and had good results. I15, from a successful startup, said that "you should put your pride aside" and "look at data to check if clients are engaging with the product". According to table 4.1 on page 32, I17 told that they collect the maximum data available before investing time and money.

6.6 Not spread knowledge

Software startups start their operations with the fewest possible people. In this stage, developers build the solution from the beginning and they know everything about it. As the team grows, the newcomers do not know how the solution was made and the former do not concern on spreading the word. Since "software development is a human-based knowledge-intensive activity" [LH09], newcomers are not able to perform their best, that is, giving new ideas.

Since innovation is very important to startup survive, it is important that the knowledge is shared throughout the organization. Then, documentation can be used to foster knowledge sharing besides helping process organization. Santos et al. [SGSF11] verified in a case study that individual's learning increased and that improved Organizational Learning. This learning increasing can be something towards Scrum implementation. Nevertheless, the authors recognize that the results could not be generalized because it was based on only one organization.

Evidences on knowledge sharing importance for innovation can also be found on Information Systems (IS) literature. Gordon and Tarafdar [GT07] investigated how information technology competences influence its ability to innovate. The authors say: "the idea of 'IT competence' derives from the resource-based view of the firm, which has gained considerable support and acceptance in the strategic management and organizational design literature". They have found that "an IT-based competence in collaboration and communication accelerated the sharing of information and knowledge across departments." And although most of the time this was used to the delivery of goods and services, occasionally this competence allowed innovative services to be delivered. Another competence mentioned was Business Involvement and the knowledge of the business could be related to innovation in the studied organizations.

In the interviews performed in this study, I9 mentioned documentation importance for when a developer leaves the organization. I9 was a startup formed by only three founders and they were the only members of the company. One of them said that documentation was important even for their own communication. It made easier to select technical solutions.

6.7 Discussion

In this chapter, six bad smells or behaviors seen in software startups were presented. Also attitudes to prevent or mitigate them were also discussed. In summary, not understanding the errors already done in the past by other startups perpetuates mistakes in these organizations. Strictly following a methodology is not necessary but, at least, founders and other team members should be aware of the best practices. Be open to other ideas, do not take anything for granted and validate everything and share knowledge between team members. Of course, this is not the success recipe. Startups may fail by several other reasons: conflicts between partners, small market for a product, not economically viable product, etc. But following these hints, they will mitigate other failure reasons.

Chapter 7

Conclusions

This work used techniques from grounded theory to study requirements engineering practices in software startups. Taking our initial research question: *"How requirements engineering practices are executed in software startups?"*, one can take our conceptual framework to answer that. The *Requiremens Engineering Process* is influenced by *Founders*, *Software Development Manager*, *Developers*, *Business Model*, *Market* and *Ecosystem* and it does not follow strictly a methodology but instead it is built according to what is needed and it is also continuously evolving as the team learns about its market, product, user or client and the team itself. The process can be managed by a *Product Team* that resembles the Product Team concept from Scrum [Sch97]. Their duties include elicit, analyze, validate and prioritize requirements giving a direction to the product development. The proposed model organization separated some stages to a requirement: it is elicited (creation), it is analyzed, validated and prioritized among others (preparation), it is built (implementation) and finally its implementation is checked (product validation). Each of these stages receive different influences from the context elements described before.

Once primary objective is discussed, a discussion about secondary objectives takes place. The first secondary objective was to assess startups and software development methodologies level of knowledge and use by interviewees. The level of knowledge turned out to be very heterogeneous ranging from little to full mastery at least at conceptual level. The use as said before is partial since the techniques are done partially and glued together to create a custom process. It is interesting to tell though that some main fundamentals are almost ubiquitous like learning from the user feedback - sometimes this is just an utopia but there is a will to do so.

Finally, to make recommendations to software startups we proposed another metaphor already present in software development literature: smells. The same path has been made to patterns and smells could also be applied here. We proposed six bad smells, that is behaviors and practices that indicate bad solutions. The bad smells were *Know-it-all founder*, *Stubborn developer*, *Closed-minded team*, *Blocked ideas*, *Validate and do not waste* and *Not spread knowledge.* Each description is followed by recommendations to avoid and/or fix that smell.

7.1 Main contributions

This study on Requirements Engineering in the context of Software Startups reached the main contributions as follow.

- Detected the influence that the Ecosystem exercise in software startups practices. In the study discussion, elements like capital availability, human resources capacity and foster institutions like acceleration programs had influences clearly grounded.

- Discussed the importance of developers feeling towards practices and how they help forge practices inside software startups - that is a clear difference to other companies.

- Transferred another metaphor (smells) from software development literature to software startups. Since, as discussed earlier, startups do not follow strictly methodologies, small pieces of information like patterns and bad smells could be a way to reach better results.

- The study also give more evidence to support studies already present in literature related to software development in startups like process formation.

Finally, the discussion of processes in software startups is important. One may think that handling the well-known startups problems like lack of capital and/or time-to-market is just doing everything as fast as possible but process is important. As pointed out by Valtanen and Ahonen [VA08], when employees have a process to follow, they plan their actions better and this is even more important in a startup context.

7.2 Threats to validity

Since the research beginning, threats to validity were a concern. This must be true for all scientific work but in our case it demanded special attention. Since it was difficult to get interviewees, each of them needed to be transcribed and a lot of time was consumed to analyze it, the overall research design was very laborious. Then, redoing everything would be very hard.

Besides that the author is partner of a software startup then a clear threat was the research bias. His prior knowledge and conceptions about the subject could have an effect on data collection and analysis. This threat was mitigated by the use of well-known techniques (grounded theory). Furthermore, the interview guide developed was another way to keep the interview focused and prevent a possible influence on the interviewee. The research inexperience could also be a problem. Then, before even designing the research, a discipline on Experimental Software Engineering has been taken. Unfortunately, a specific discipline on Grounded Theory was not available during the time what would have been very good.

Another threat could be the opportunistic approach used to reach interviewees. But, as described earlier, it was possible to get startups and people who operate in different markets, have different sizes and maturity stages.

An important technique to increase validity was to compare results to other studies in literature. In case of process formation, there are clear similarities to Coleman and O'Connor [CO08] study since four influences identified in the present study are also present in their work. In the case of Developers influence, there were references from agile software development literature ([CC08], [NMM05], [VT08]) that mention this factor and, since the practices are closer, a similar influence is expected. And Ecosystem is a theme already very discussed in software startups literature ([CKK15], [MW14]).

An interesting discussion on assessing grounded theory studies is made by Jantunen and Gause [JG14]. Some techniques they used are also present in this work. They are:

- **Prolonged engagement:** this study took a year, consisting of seventeen interviews in three rounds. Analysis was made through several cycles when data was read multiple times and always compared to new facts.

- **Triangulation:** although this study was conducted by a sole researcher, the preferred place to make interviews were where the startup run their operation that be their own office or a co-working space. Although many of them were made through a call, the in-place interviews made possible to the researcher make some notes about working space.

- **Peer debriefing:** again nevertheless this study was conducted by a sole researcher, it was consistently discussed with his supervisor. Besides that, a paper containing the first research stage results was published and presented to the software startup research community.

- **Referential adequacy:** like the referenced study, this work had all interviews recorded and transcribed and labels and memos are stored in AtlasTI tool.

- **Member checking:** the resulting model was never directly presented to the interviewees but questions added in following versions of the interview guide were based in the results obtained in the previous research stages.

7.3 Future work

This work spawn different streams for future works. Actually each detected influence can be better scrutinized including the ecosystem. This study could be replicated in different ecosystems with different maturity levels and a comparison between them could give more diverse insights. Coming out of processes description, experiments or study cases can be made to check when methodologies are more well followed the startups' success rates will be better. For instance, a deeper investigation on founders influence on software development in software startups even when they are not technical could be performed and if this has a correlation to startup success. Another interesting investigation would be detailing differences or evolution of requirements engineering practices in software startups with different maturity stages. A more detailed study on product team and its role in business and software development would also be important.

A discussion on the difference between requirements engineering in software consolidated firms and startups could be made in depth. In our study, the product validation can already be seen as one aspect of this difference. Interviewing consolidated firms or a literature review focused could give more aspects.

Other points not well discussed by this work could be more explored like non-functional requirements and requirements management. Non-functional requirements have been neglected initially by agile practitioners but some practices have been proposed to tackle this problem [ISM+15]. How startups can tackle this problem and if they are already concerned about this would be a valuable contribution. Requirements management, mainly how they evolve through time, was mentioned during interviews but more investigation on it was left to be done in future works as well. Some question rise from this theme: how startups handle requirements changes during development? Are they aware of it? These theme is very important given requirements volatility present in startups context.

Besides that, given the smell metaphor, several other smells could be identified, explained and more advice can be given to founders and developers in software startups. The smells described here were concerned on requirements engineering. From software startups scientific literature and other books could be rephrased using the smell metaphor. It could also be possible to extrapolate some bad smells from software startups to software development in general or even organizations in general.

7.4 Production during this research

During this course, some contributions have been made not necessarily related to the theme of this work. They are described now.

- The author helped the development of a scientific tool for metabolic calculation using the Extreme Programming methodology. The tool can be used by researchers and practitioners on physical exercise to understand how a person obtains energy during exercise. A paper entitled *"GEDAE-LaB: A Free Software to Calculate the Energy System Contributions during Exercise"* was published in the Public Library of Science One journal [BMB+16].

- A work-in-progress paper entitled *"Software Development Patterns for Startups"* was accepted to the Writing Group of the *10th Latin American Conference on Pattern Languages of Programs (SugarLoafPLoP 2014)* when was presented and discussed.

- The previous paper was improved and became "Seven patterns for Software Startups" that was accepted to the *22nd Conference on Pattern Languages of Programs (PLoP 2015)* [MG15].

- A colaboration started during the conference that led to another paper accepted to the 11th Latin American Conference on Pattern Language of Programs (SugarLoafPLoP 2016) entitled *"Early-Stage Software Startup Patterns - Towards a Pattern Language"*.

- As already mentioned, the stage 1 of this research was presented as the paper entitled *"Requirements Engineering in Software Startups"* during the *2nd International Workshop on Software Startups* [MG16].

Appendix A

Interview guide

In this chapter, the interview guide used during the interviews are presented translated in English from Portuguese that was the language used in the interviews. The final version is presented within comments about changes made during different research stages.

1. In how many startups have you worked for and/or founded?
   a) 1
   b) 2
   c) 3 or more
2. How long at total have you worked for startups?
   a) Less than 1 year
   b) From 1 to 2 years
   c) From 2 to 3 years
   d) More than 3 years
3. Which title best describes the position you had in your last startup?
   a) Software developer
   b) Tech manager or director
   c) Marketing
   d) Operations
   e) Strategy
   f) Other, what?
4. For how long does your current startup exist?
   a) Less than 1 year
   b) From 1 to less than 2 years
   c) 2 years or more
5. Briefly describe your last startup product. How software development is related to it?
6. Which of the following techniques do you know? Do you try to use them?
   a) Lean Startup
   b) Design Thinking

c) Scrum
d) Extreme Programming
e) Kanban
f) AB tests
g) MVP

Elicitation

7. Which of the following phrases best describes how the features that your company develop will have:

   a) Developers' ideas
   b) Ideas from other company teams (marketing, operations, etc.)
   c) Surveys or user interviews
   d) Suggestions form in an website/app?
   e) Other?

   Negotiation

8. Once a idea is created, is it discussed somehow before getting implemented? Who takes into this discussion?

9. Given the possible features list, how, in your startup, is defined which will be implemented in the next release?

   a) Development team meeting (with or without other company members) b) Managers' decision (manager, director, etc)
   c) Survey with users
   d) Developer will (what s/he is most interested in)
   e) Other?

   Validation

10. Given the features that will be implemented in your startup, is there any kind of verification if it will attend an user expectation before getting implemented? If yes, how?

11. In your startup, is it verified if a implemented features is being used by users? If yes, how?

    Documentation

12. How does your startup document the features that will be implemented?

    a) E-mails sent between team members
    b) Paper writing
    c) Stick notes in a board
    d) Wiki
    e) Issue tracker (e.g. Bugzilla)
    f) Project tracking tools (e. g. Jira)
    g) Other?

13. In your opinion, what determines how the requirements engineering process will be made in your startup?

14. Is anyone responsible to guide the product development?
15. Any other details about features to be implementation choice that was not covered in this interview?
16. Any comments?

Bibliography

[ABG+10] Lutfihak Alpkan, Cagri Bulut, Gurhan Gunday, Gunduz Ulusoy, and Kemal Kilic. Organizational support for intrapreneurship and its interaction with human capital to enhance innovative performance. *Management Decision*, 48(5):732–755, 2010. 2, 58

[AH01] Bostjan Antoncic and Robert D Hisrich. Intrapreneurship: Construct refinement and cross-cultural validation. *Journal of business venturing*, 16(5):495–527, 2001. 58

[AI77] Christopher Alexander and Murray Ishikawa, Sara anfd Silverstein. Pattern languages. *Center for Environmental Structure*, 2:1977, 1977. 55

[AST06] Allen C. Amason, Rodney C. Shrader, and George H. Tompson. Newness and novelty: Relating top management team composition to new venture performance. *Journal of Business Venturing*, 21(1):125–148, 2006. 3, 40

[BA05] Kent Beck and Cynthia Andres. *Extreme Programming Explained: Embrace Change*. The XP series. Addison-Wesley, 2005. 3, 4, 8, 43

[Bas13] Julian M. Bass. Agile method tailoring in distributed enterprises: Product owner teams. *Proceedings - IEEE 8th International Conference on Global Software Engineering, ICGSE 2013*, pages 154–163, 2013. 43

[BBvB+01] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. Manifesto for agile software development, 2001. 3

[Bla07] Steve Blank. *The Four Steps to the Epiphany: Successful Strategies for Products that Win*. Cafepress.com, 2007. 1, 4, 9, 38, 43, 58

[Bla13] Steve Blank. Why the Lean Start Up Changes Everything. *Harvard Business Review*, 91(May):64, 2013. 4, 10

[BLW15] Richard Banfield, C. Todd Lombardo, and Trace Wax. *Design Sprint: A Practical Guidebook for Building Great Digital Products*. O'Reilly Media, 2015. 12

[BMB+16] Rômulo Bertuzzi, Jorge Melegati, Salomão Bueno, Thaysa Ghiarone, Leonardo A Pasqua, Arthur Fernandes Gáspari, Adriano E Lima-Silva, and Alfredo Goldman. Gedae-lab: A free software to calculate the energy system contributions during exercise. *PloS one*, 11(1):e0145733, 2016. 64

[BOBL13] Jan Bosch, Helena Holmström Olsson, Jens Björk, and Jens Ljungblad. The Early Stage Software Startup Development Model: A Framework for Operationalizing Lean Principles in Software Startups. *Lean Enterprise Software and Systems*, pages 1–15, 2013. 13, 58, 59

[BPI92] Richard Buchanan, The M I T Press, and Design Issues. Wicked problems in design thinking. *Design Issues*, 8(2):5–21, 1992. 11

[Bro08] Tim Brown. Design thinking. *Harvard business review*, 86(6):84, 2008. 12

[Bro09] Tim Brown. *Change by Design: How Design Thinking Transforms Organizations and Inspires Innovation*. HarperCollins, 2009. 4, 12, 46, 51

[CAJ07] Betty H C Cheng, Joanne M Atlee, and M Joanne. Research Directions in Requirements Engineering. *Proceeding FOSE '07 2007 Future of Software Engineering*, pages 285–303, 2007. 4, 13, 48

[CC08] Tsun Chow and Dac-Buu Cao. A survey study of critical success factors in agile software projects. *Journal of Systems and Software*, 81(6):961–971, 2008. 41, 57, 62

[CKK15] Daniel Cukier, Fabio Kon, and Norris Krueger. Designing a maturity model for software startup ecosystems. In *International Conference on Product-Focused Software Process Improvement*, pages 600–606. Springer, 2015. 42, 55, 62

[CO08] Gerry Coleman and Rory V. O'Connor. An investigation into software development process formation in software start-ups. *Journal of Enterprise Information Management*, 21(6):633–648, 2008. 7, 8, 20, 35, 36, 37, 44, 51, 62

[CR08] Lan Cao and Balasubramaniam Ramesh. Agile requirements engineering practices: An empirical study. *IEEE Software*, 25:60–67, 2008. 15

[Cro02] Mark Crowne. Why software product startups fail and what to do about it. Evolution of software product development in startup companies. *IEEE International Engineering Management Conference*, 1:338–343, 2002. 7, 55

[CSS12] Dave Chaffey, Paul Russell Smith, and Paul Russell Smith. *eMarketing eXcellence: Planning and optimizing your digital marketing*. Routledge, 2012. 10

[Cun93] Ward Cunningham. The wycash portfolio management system. *ACM SIGPLAN OOPS Messenger*, 4(2):29–30, 1993. 9

[CY13] Alistair Croll and Benjamin Yoskovitz. *Lean Analytics: Use Data to Build a Better Startup Faster*. Lean (O'Reilly). O'Reilly Media, Incorporated, 2013. 40

[DD05] Eric Deakins and Stuart Dillon. A helical model for managing innovative product and service initiatives in volatile commercial environments. *International Journal of Project Management*, 23:65–74, 2005. 1

[DDMP14] Maya Daneva, Daniela Damian, Alessandro Marchetto, and Oscar Pastor. Empirical research methodologies and studies in Requirements Engineering: How far did we come? *Journal of Systems and Software*, 95:1–9, 2014. 27

[DMK04] Robert M. Davison, Maris G. Martinsons, and Ned Kock. Information Systems Journal : Principles of Canonical Action Research. 14:65–86, 2004. 19

[DS16] Yngve Dahle and Martin Steinert. Does Lean Startup really work? In *Engineering, Technology and Innovation (ICE) \& IEEE International Technology Management Conference, 2016 International Conference on*, pages 166–170, 2016. 11

[DSCE08] Alan De-Ste-Croix and Alan Easton. The product owner team. *Proceedings - Agile 2008 Conference*, pages 274–279, 2008. 43

[Ebe07] Christof Ebert. The impacts of software product management. *Journal of Systems and Software*, 80:850–861, 2007. 4

[EL02] Armin Eberlein and Julio Leite. Agile Requirements Definition: A View from Requirements Engineering. *International Workshop on Time-Constrained Requirements Engineering, TCRE 2002*, pages 1–5, 2002. 14, 38

[Elo14] Veli-Pekka Eloranta. Towards a pattern language for software start-ups. In *Proceedings of the 19th European Conference on Pattern Languages of Programs*, page 24. ACM, 2014. 55

[ESSD08] Steve Easterbrook, Janice Singer, Margaret-anne Storey, and Daniela Damian. Selecting Empirical Methods for Software Engineering Research. *Guide to Advanced Empirical Software Engineering*, pages 285–311, 2008. 19

[FB99] Martin Fowler and Kent Beck. *Refactoring: Improving the Design of Existing Code.* Component software series. Addison-Wesley, 1999. 56

[FBZ12] Francesca Arcelli Fontana, Pietro Braione, and Marco Zanoni. Automatic detection of bad smells in code: An experimental assessment. *Journal of Object Technology*, 11(2):1–38, 2012. 56

[FR83] R. Edward Freeman and David L. Reed. Stockholders and shareholders: a new perspective on corporate governance. *California Management Review*, 25(3):88–106, 1983. 1

[FSO10] Nina Dzamashvili Fogelström, Tony Gorschek Mikael Svahnberg, and Peo Olsson. The impact of agile principles on market-driven software product development. *Journal of Software Maintenance and Evolution*, 22(May 2009):53–80, 2010. 16

[GA03] Jaap Gordijn and JM Akkermans. Value-based requirements engineering: exploring innovative e-commerce ideas. *Requirements engineering*, 8(2):114–134, 2003. 55

[GHJV94] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns: elements of reusable object-oriented languages and systems.* Addison-Wesley Reading, 1994. 55

[GPEM09] Joshua Garcia, Daniel Popescu, George Edwards, and Nenad Medvidovic. Toward a catalogue of architectural bad smells. In *International Conference on the Quality of Software Architectures*, pages 146–162. Springer, 2009. 56

[GPU$^+$15] Carmine Giardino, Nicolo Paternoster, Michael Unterkalmsteiner, Tony Gorschek, and Pekka Abrahamsson. Software Development in Startup Companies: The Greenfield Startup Model. *IEEE Transactions on Software Engineering*, 42(September):233, 2015. 8, 28

[GS67] Barney Glaser and Anselm Strauss. *The discovery of grounded theory : strategies for qualitative research.* Aldine Pub. Co, Chicago, 1967. 21

[GT07] Steven R. Gordon and Monideepa Tarafdar. How do a company's information technology competences influence its ability to innovate? *Journal of Enterprise Information Management*, 20(3):271–290, 2007. 41, 58, 59

[GWA14] Carmine Giardino, Xiaofeng Wang, and Pekka Abrahamsson. Why early-stage software startups fail: A behavioral framework. *Lecture Notes in Business Information Processing*, 182 LNBIP:27–41, 2014. 55

[Har15] Rainer Harms. Self-regulated learning, team learning and project performance in entrepreneurship education: Learning in a lean startup environment. *Technological Forecasting and Social Change*, 100:21–28, 2015. 11

[HC04] Helen Heath and Sarah Cowley. Developing a grounded theory approach: A comparison of Glaser and Strauss. *International Journal of Nursing Studies*, 41:141–150, 2004. 21

[HKB10] Matthias Häsel, Tobias Kollmann, and Nicola Breugst. IT Competence in Internet Founder Teams. *Bise*, 2(4):209–217, 2010. 3, 40

[HL15] Laura Hokkanen and Marko Leppanen. Three patterns for user involvement in startups. In *Accepted to 20th European Conference on Pattern Languages of Programs*, 2015. 55

[Hui13] Alexis Hui. Lean change: Enabling agile transformation through lean startup, kotter and kanban: An experience report. In *Agile Conference (AGILE), 2013*, pages 169–174. IEEE, 2013. 11

[ISM$^+$15] Irum Inayat, Siti Salwah Salim, Sabrina Marczak, Maya Daneva, and Shahaboddin Shamshirband. A systematic literature review on agile requirements engineering practices and challenges. *Computers in Human Behavior*, 51(0):–, 2015. 14, 63

[JG14] Sami Jantunen and Donald C. Gause. Using a grounded theory approach for exploring software product management challenges. *Journal of Systems and Software*, 95:32–51, 2014. 21, 28, 62

[JSWÇ13] Ulla Johansson-Sköldberg, Jill Woodilla, and Mehves Çetinkaya. Design Thinking: Past, Present and Possible Futures. *Creativity and Innovation Management*, 22(2):121–146, jun 2013. 12

[KDR$^+$07] Lena Karlsson, Asa G Dahlstedt, Björn Regnell, Johan Natt och Dag, and Anne Persson. Requirements engineering challenges in market-driven software development - An interview study with practitioners. *Information and Software Technology*, 49:588–604, 2007. 28, 38

[Ken99] Judy Kendall. Axial coding and the grounded theory controversy. *Western journal of nursing research*, 21(6):743–757, 1999. 21

[KMN08] Mira Kajko-Mattsson and Natalja Nikitina. From Knowing Nothing to Knowing a Little: Experiences Gained from Process Improvement in a Start-Up Company. In *Computer Science and Software Engineering, 2008 International Conference on*, volume 2, pages 617–621. Ieee, 2008. 41

[Kru04] Philippe Kruchten. *The rational unified process: an introduction*. Addison-Wesley Professional, 2004. 8

[Lep14] Marko Leppänen. Patterns for starting up a software startup company. *Proceedings of the 19th European Conference on Pattern Languages of Programs - EuroPLoP '14*, pages 1–7, 2014. 55

[LH09] Meira Levy and Orit Hazzan. Knowledge Management in Practice: The Case of Agile Software Development. *2009 Icse Workshop on Cooperative and Human Aspects of Software Engineering*, pages 60–65, 2009. 59

[LH15] Marko Leppänen and Laura Hokkanen. Four patterns for internal startups. *Proceedings of the 20th European Conference on Pattern Languages of Programs - EuroPLoP '15*, pages 1–10, 2015. 55

[LHC10] Yixin Luo, Allyson Hoss, and Doris L. Carver. An ontological identification of relationships between anti-patterns and code smells. *IEEE Aerospace Conference Proceedings*, 2010. 56

[LME+12] James Lockerbie, Neil Arthur McDougall Maiden, Jorgen Engmann, Debbie Randall, Sean Jones, and David Bush. Exploring the impact of software requirements on system-wide goals: A method using satisfaction arguments and i* goal modelling. *Requirements Engineering*, 17:227–254, 2012. 16

[McC96] Steve McConnell. Daily build and smoke test. *IEEE software*, 13(4):144, 1996. 47

[MG15] Jorge Melegati and Alfredo Goldman. Seven patterns for software startups. In *Proceedings of 22nd Conference on Pattern Languages of Programs*, 2015. 56, 64

[MG16] Jorge Melegati and Alfredo Goldman. Requirements Engineering in software startups : a grounded theory approach. In *2nd Internation Workshop on Software Startups*, 2016. 29, 64

[MHS13] Matthwe B. Miles, A. Michael Huberman, and Johnny Saldaña. *Qualitative Data Analysis: A Methods Sourcebook*. SAGE Publications, 2013. 27

[MM12] Alastair Milne and Neil Maiden. Power and politics in requirements engineering: Embracing the dark side? *Requirements Engineering*, 17:83–98, 2012. 14

[Moo12] Dobrila Rancic Moogk. Minimum viable product and the importance of experimentation in technology startups. *Technology Innovation Management Review*, 2(3):23, 2012. 11

[MT12] Roland M. Mueller and Katja Thoring. Design Thinking Vs Lean Startup: A Comparison of Two Userdriven Innovation Strategies. In *Proceedings of 2012 International Design Management Research Conference*, pages 151–161, 2012. 12

[MVL03] Mika Mantyla, Jari Vanhanen, and Casper Lassenius. A taxonomy and an initial empirical study of bad smells in code. In *Software Maintenance, 2003. ICSM 2003. Proceedings. International Conference on*, pages 381–384, 2003. 56

[MW14] Yasuyuki Motoyama and Karren K Watkins. Examining the connections within the startup ecosystem: A case study of st. louis. *Louis (September 1, 2014). Kauffman Foundation Research Series on City, Metro, and Regional Entrepreneurship*, 2014. 41, 62

[Mye97] Michael D. Myers. Qualitative research in information systems. *Management Information Systems Quarterly*, 21(June):1–18, 1997. 19, 20

[Nam02] Satish Nambisan. Software firm evolution and innovation-orientation. *Journal of Engineering and Technology Management - JET-M*, 19(2):141–165, 2002. 41

[NDSA15] Anh Nguyen-Duc, Pertti Seppänen, and Pekka Abrahamsson. Hunter-gatherer cycle: a conceptual model of the evolution of software startups. In *Proceedings of the 2015 International Conference on Software and System Process - ICSSP 2015*, pages 199–203. ACM, 2015. ix, 39, 40

[NE00] Bashar Nuseibeh and Steve Easterbrook. Requirements engineering: a roadmap. *ICSE 2000 Proceedings of the Conference on The Future of Software Engineering*, 1:35–46, 2000. 3

[NL03] Colin. J. Neill and Phillip A. Laplante. Requirements engineering: the state of the practice. *IEEE Software*, 20(6), 2003. 14

[NMM05] Sridhar Nerur, Radhakanta Mahapatra, and George Mangalaraj. Challenges of Migrating to Agile Methodologies. *Communications of the ACM*, 48(2):72–78, 2005. 41, 62

[Nob11] Carmen Nobel. Teaching a ' Lean Startup ' Strategy. *Harvard Business School*, pages 1–2, 2011. 11

[Pat13] Matthias Patz. Lean startup : adding an experimental learning perspective to the entrepreneural process. Master's thesis, University of Twente, Janeiro 2013. 1

[Pau15] Danielly Ferreira Oliveira de Paula. Model for the innovation teaching (moit): um modelo baseado em design thinking, lean startup e ágil para estudantes de graduação em computação. Master's thesis, Universidade Federal de Pernambuco, 2015. 11

[PGU$^+$14] Nicolò Paternoster, Carmine Giardino, Michael Unterkalmsteiner, Tony Gorschek, and Pekka Abrahamsson. Software development in startup companies: A systematic mapping study. *Information and Software Technology*, April 2014. 1, 2, 3, 7, 56

[Pot95] Colin Potts. Invented requirements and imagined customers: requirements engineering for off-the-shelf software. *Proceedings of 1995 IEEE International Symposium on Requirements Engineering (RE'95)*, pages 128–130, 1995. 38

[PP03] Mary Poppendieck and Tom Poppendieck. *Lean Software Development: An Agile Toolkit: An Agile Toolkit*. Agile Software Development Series. Pearson Education, 2003. 10

[RB05] Björn Regnell and Sjaak Brinkkemper. Market-driven requirements engineering for software products. *Engineering and Managing Software Requirements*, pages 287–308, 2005. 38

[Rie11] Eric Ries. *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. Crown Business, 2011. 1, 2, 4, 46, 48, 58

[RS16] Kevin Reuther and Christian-Andreas Schumman. Intrapreneurship : Increasing Employees' Responsibility for an Enhancement of Innovation Performance. In *Engineering, Technology and Innovation (ICE) \& IEEE International Technology Management Conference, 2016 International Conference on*, pages 147–149, 2016. 2

[Rus15] Radostina Ruseva. Patterns for startup business models. In *Proceedings of the 20th European Conference on Pattern Languages of Programs*, pages 1–11. ACM, 2015. 55

[RW73] Horst W J Rittel and Melvin M. Webber. Dilemmas in a general theory of planning. *Policy Sciences*, 4(2):155–169, 1973. 11

[RW13] JPC Rigtering and Utz Weitzel. Work context and employee behaviour as antecedents for intrapreneurship. *International Entrepreneurship and Management Journal*, 9(3):337–360, 2013. 58

[SC90] Anselm L. Strauss and Juliet M. Corbin. *Basics of qualitative research: grounded theory procedures and techniques*. Sage Publications, 1990. 20, 21, 22, 23

[SC96] Jawed Siddiqi and M Chandra. Requirements engineering: the emerging wisdom. *Ieee Software*, 1996. 14

[Sch97] Ken Schwaber. Scrum development process. In *Business Object Design and Implementation*, pages 117–134. Springer, 1997. 61

[Sch04] Ken Schwaber. *Agile Project Management with Scrum*. Best practices. Microsoft Press, 2004. 4, 43

[SCRS05] Alberto Sillitti, Martina Ceschi, Barbara Russo, and Giancarlo Succi. Managing uncertainty in requirements: A survey in documentation-driven and Agile companies. *Proceedings - International Software Metrics Symposium*, 2005(Metrics):145–154, 2005. 16

[Sea99] Carolyn B. Seaman. Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering*, 25(4):557–572, 1999. 19, 20

[Sel09] Bran Selic. Agile documentation, anyone? *IEEE Software*, 26(6):11–12, 2009. 3

[SGSF11] Viviane Santos, Alfredo Goldman, Ana Carolina M Shinoda, and Andre L. Fischer. A view towards Organizational Learning: An empirical study on Scrum implementation. *SEKE 2011 - Proceedings of the 23rd International Conference on Software Engineering and Knowledge Engineering*, pages 583–589, 2011. 59

[Sim96] Herbert A. Simon. *The Sciences of the Artificial (3rd Ed.)*. MIT Press, Cambridge, MA, USA, 1996. 11

[SK98] Ian Sommerville and Gerald Kotonya. *Requirements Engineering: Processes and Techniques*. John Wiley & Sons, Inc., New York, NY, USA, 1998. 3, 46

[SKB02] Kan Ichiro Suzuki, Sang Hoon Kim, and Zong Tae Bae. Entrepreneurship in Japan and Silicon Valley: A comparative study. *Technovation*, 22(10):595–606, 2002. 42

[SMMS11] Tiago Silva, Angela Martin, Frank Maurer, and Milene Silveira. User-Centered Design and Agile Methods: A Systematic Review. *Agile Conference (AGILE), 2011*, pages 77–86, 2011. 16

[SOL16] Pertti Seppänen, Markku Oivo, and Kari Liukkunen. The initial team of a software startup. In *Engineering, Technology and Innovation (ICE) \& IEEE International Technology Management Conference, 2016 International Conference on*, pages 57–65, 2016. 36, 44

[Sut00] Stanley M Sutton. The Role of Process in a Software Start-up. *IEEE Software*, pages 33–39, 2000. 2

[Tai10] Marko Taipale. Huitale–a story of a finnish lean startup. In *Lean Enterprise Software and Systems*, pages 111–114. Springer, 2010. 7, 51

[TM14] Peter A. Thiel and Blake Masters. *Zero to One: Notes on Startups, Or how to Build the Future*. Crown Business, 2014. 1, 9

[ULM10] Cathy Urquhart, Hans Lehmann, and Michael D. Myers. Putting the 'theory' back into grounded theory: Guidelines for grounded theory studies in information systems. *Information Systems Journal*, 20:357–381, 2010. 21

[VA08] Anu Valtanen and Jarmo J. Ahonen. Big improvements with small changes: Improving the processes of a small software company. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5089 LNCS(2006):258–272, 2008. 49, 62

[VMM16] Sergio Edú Valsania, Juan A. Moriano, and Fernando Molero. Authentic leadership and intrapreneurial behavior: cross-level analysis of the mediator effect of organizational identification and empowerment. *International Entrepreneurship and Management Journal*, 12(1):131–152, 2016. 2, 58

[VT08] Leo R. Vijayasarathy and Dan Turk. Agile software development: A survey of early adopters. *Journal of Information Technology Management*, XIX(2):1–8, 2008. 41, 62

[WEB+16] Xiaofeng Wang, Henry Edison, Sohaib Shahid Bajwa, Carmine Giardino, and Pekka Abrahamsson. Key Challenges in Software Startups Across Life Cycle Stages. In *Lecture Notes in Business Information Processing*, pages 169–182. Springer, 2016. 39

[Won06] Jarunee Wonglimpiyarat. The dynamic economic engine at Silicon Valley and US Government programmes in financing innovations. *Technovation*, 26(9):1081–1089, 2006. 42

[Zav95] Pamela Zave. Classification of research efforts in requirements engineering. *Proceedings of 1995 IEEE International Symposium on Requirements Engineering (RE'95)*, 29(4), 1995. 14

[ZC05] Didar Zowghi and Chad Coulin. Requirements elicitation: A survey of techniques, approaches, and tools. *Engineering and Managing Software Requirements*, pages 19–46, 2005. 14

[ZMMW01] Jörg Zettel, Frank Maurer, Jürgen Münch, and Les Wong. LIPE: a lightweight process for e-business startup companies based on extreme programming. *Product Focused Software Process Improvement*, pages 255–270, 2001. 7, 44