Tampereen teknillinen yliopisto. Tietotekniikan laitos. Raportti 4
Tampere University of Technology. Department of Pervasive Computing. Report 4

Anuradha Dande, Veli-Pekka Eloranta, Hadaytullah, Antti-Jussi Kovalainen, Timo Lehtonen, Marko Leppänen, Taru Salmimaa, Mahbubul Sayeed, Matti Vuori, Claude Rubattel, Wolfgang Weck & Kai Koskimies

Software Startup Patterns – An Empirical Study

TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

Tampereen teknillinen yliopisto. Tietotekniikan laitos. Raportti 4
Tampere University of Technology. Department of Pervasive Computing. Report 4

Anuradha Dande, Veli-Pekka Eloranta, Hadaytullah, Antti-Jussi Kovalainen, Timo Lehtonen, Marko Leppänen, Taru Salmimaa, Mahbubul Sayeed, Matti Vuori, Claude Rubattel, Wolfgang Weck & Kai Koskimies

Software Startup Patterns – An Empirical Study

Tampere University of Technology. Department of Pervasive Computing
Tampere 2014

ISBN 978-952-15-3251-1 (printed)
ISBN 978-952-15-3252-8 (PDF)
ISSN 2323-9174

Abstract

This report gives an account of the results of a joint research effort between Tampere University of Technology, Department of Pervasive Computing and University of Applied Sciences and Arts Northwestern Switzerland, School of Engineering. The research was integrated with a seminar participated by PhD students at TUT. The aim was to identify working practices in software startups through interviews and formulate them as patterns. The results consist of the set of 63 found practices, their relationships, and a set of 14 patterns formulated for a selected set of the practices.

Acknowledgements

We wish to thank the participating companies for their support on our study: CAScination AG, Celkee, Design Thinking Startup AG, Doodle AG, E-Day, easypsim, edorasware, Flockler, Frontify, Intopalo, Leonidas, naneos particle solutions gmbh, Piceasoft, Sanovation AG, ServiceHunter AG, TeamUp, Vincit, and Wirepas. The Finnish side of this research has been funded by Tekes in the context of the N4S Digile program. The Swiss work part has been funded by Hasler Foundation.

Contents

1 Introduction

A growing trend in industrial software engineering is that new software products and information services are developed under conditions of notable uncertainty. This is especially visible in startup enterprises which aim at new kinds of products and services in rapidly changing social web, where potential customers can quickly adopt new behavior. However, also established, large companies are more and more seeking proactively new business opportunities with their customers, without having clear understanding of the requirements of the products and services that are actually desired by the customers. Another trend in information industry is the emphasis of fast development: in many areas, and especially in startups, the development speed is much more important than the cost level, as far as the economic value of a product is concerned. These trends have resulted in a software engineering paradigm (called here Startup Software Engineering, SSE), where the development and the usage of a product or service are not separated, but merged into a fast build-use-measure-learn cycle. This is in sharp contrast with existing software engineering paradigms which assume that requirements have been elicited before the actual development starts, and which emphasize controlled process rather than fast delivery and learning.

This report gives an account of the outcome of the Research Seminar on Startup Software Engineering at Tampere University of Technology, held during September 2013 – January 2014 (TIE-12206). The purpose of the seminar was to carry out a small-scale empirical study on the software engineering practices in software startups, as a joint research between Tampere University of Technology and University of Applied Sciences and Arts Northwestern Switzerland, School of Engineering. As part of the study, 18 companies, 9 both in Tampere region and in Switzerland were interviewed. The companies were CAScination AG, Celkee, Design Thinking Startup AG, Doodle AG, E-Day, easypsim, edorasware, Flockler, Frontify, Intopalo, Leonidas, naneos particle solutions gmbh, Piceasoft, Sanovation AG, ServiceHunter AG, TeamUp, Vincit, and Wirepas.

Special benefits of the Finnish-Swiss research cooperation were not just the duplication of the number of companies surveyed, but also a chance to see a similarity of many of the general patterns found in both regions. The companies selected in Switzerland ranged from internet-based services for a consumer market to web-based development tools for professional software developers to software providing critical business support to other companies to software being marked solely as part of a physical product. In Finland, the companies were mostly developing web-based or mobile software products. Most of the companies were classified as typical startups, but we wanted to include also some companies that had passed the startup phase, and ask about their post-startup experiences. Due to the diversity of the companies and software ecosystems, we expected to cover a wide range of practices.

The found practices were presented as so-called patterns. Patterns have been used in many areas of software engineering to describe solutions to problems in a particular context. Pattern mining is an activity in which instances of new or known patterns are identified in actual usage contexts. In this work we have carried out pattern mining in software startups using interview method. Thus, there is evidence that the found patterns are useful in an actual software startup context. It is expected that the resulting collection of patterns can be used as a source of solutions for various problems faced in software startups.

The study resulted in 63 pattern candidates. After a screening process, 14 pattern candidates were further elaborated using the pattern workshop method into a more refined form. The main outcome of the research is the list of pattern candidates given in Section 5, and the list of the elaborated patterns presented in Section 4. A preliminary analysis of the relationships of the patterns is presented in Section 6. Before going into the found patterns, we briefly describe the pattern concept used here in Section 2, and the research process in Section 3. The form used in the company interviews is given as Appendix.

2 Patterns

The pattern concept stems from building architects. Originally, a pattern was presented as a context-dependent construct which includes a problem and a reproducible solution to such a problem in designing buildings [Ale77]. Since mid 1990's, patterns have been one of the most successful concepts in software engineering. Collections of patterns have been presented in literature and in web for various subareas of software engineering, like analysis, design, user interfaces, web development, software architecture, etc. Besides technical areas, patterns have been applied also to software development organization and process [CH05]. Our patterns resemble these so-called *organizational patterns* in that they concern typically humans rather than software components. However, we have not explicitly excluded pure software patterns in this study, either.

Name
Descriptive name of the pattern, expressing the core idea
Category
Category of the pattern, that is, major concern in software startups mostly related to the pattern
Context
The context in a startup company where this problem occurs and the solution can be applied (e.g. the phase of start-up)
Problem
The problem solved by the solution
Forces
The possibly conflicting facts or interests causing the problem
Solution
Description of the solution
Consequences
Benefits and liabilities of the solution
Related patterns
Patterns related to this one, possibly used in concert with this pattern or as alternative solutions
Example story
Most interesting war story heard in the interviews that demonstrates the usage of the pattern. Company details are abstracted away.
References
Possible references to literature

Figure 1. Pattern format used for elaborating found practices

Many different formats have been proposed for presenting patterns. We have adopted here a slightly modified version of the so-called Coplien format [Cop96]. A notable addition is the "Example story" item, where a real but abstracted usage story from some startup company is presented. We have also added an item giving the concern the pattern addresses in a

startup company (Category). Our format, used for the elaborated patterns, is described in Figure 1. The last three parts of the template are in principle optional, although present in most patterns.

While collecting pattern ideas based on the interviews in startup companies, we used another, much shorter format, so called Portland Form introduced in the first pattern repository, Portland Pattern Repository [Por14]. In this format, the pattern idea is presented roughly as a problem-solution pair (Problem-Therefore).

3 Research process

The research process is depicted in Figure 2. The research can be viewed as a pattern mining activity based on interviews of practitioners. The main outcome is the list of potential pattern candidates, expressed in a short problem-solution form. A small subset of these pattern candidates were also selected for more detailed description.

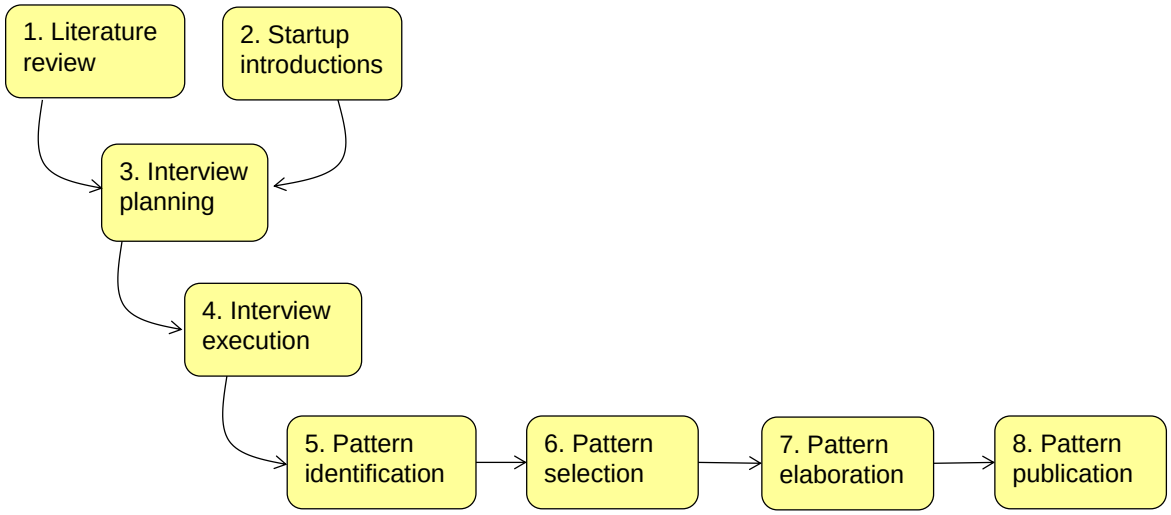


Figure 2. Research process

In the first stage (steps 1 and 2 in Figure 2), the participants first mapped out the existing literature on software startups (in particular, [COC08], [Cro02], [GP12], [Hei06], [Hei07], [May12], [Rie11], [Str06], [Tai10], [Tho03], [Zet01]). Each participant gave a presentation of one or more articles. In addition, invited speakers from startup companies presented their views on the real work in the startup companies.

In the second stage (step 3 and 4), an interview form was designed (see Appendix), and the participants interviewed the developers and architects in startup companies using that form. The companies were selected from the Tampere region and from Switzerland. Altogether 18 companies were interviewed, 9 from Tampere and 9 from Switzerland.

In the third stage (step 5), patterns were identified from the interview transcripts. 69 potential pattern candidates were identified and presented using the Portland Form (see Section 2). At this stage, we accepted patterns that had been identified in a real context in at least one company. However, most of the patterns were identified in 2-3 companies, some in more than 3 companies. Although the focus was in software startups, many of the found patterns could also have been used in other than software companies, and similarly many of the patterns could also have been used in other than startup companies. The fact that a pattern

might be useful outside of software startups was not considered to be a reason to reject a pattern candidate at this point.

After identifying the patterns, they were given unique identification numbers and grouped according to the main concern of the pattern, to facilitate the processing of the patterns and to establish preliminary structure for the pattern set. The categories (that is, the main concerns of software startups) were recognized from the patterns themselves, rather than deciding them independently of the patterns. Thus, it is possible that the found categories do not necessarily cover all aspects of software startup working practices. Altogether 11 categories were identified: Goals, Culture, Funding, Customer, Organization, Competence, Team, Process, Design, Testing, and Technology. During this process, some patterns were also merged, and some patterns were removed, resulting in a set of 63 pattern candidates.

In the fourth stage (step 6), a small set of 14 patterns was selected to be elaborated into a more refined representation. The basic idea of the selection was to concentrate on patterns that are particularly relevant for startups and/or that deal particularly with the challenges of software engineering. This is depicted in Figure 3. The selection was performed by voting among the participants: each participant could give ranking both for startup relevance and for overall importance in software development. Based on these rankings, 14 patterns were selected for further elaboration. It should be noted, however, that the ranking process was fairly superficial and not intended to be in any way scientifically sound, but rather the aim was to select a representable sample pattern set. A more thought-out selection process might lead to a different pattern set.

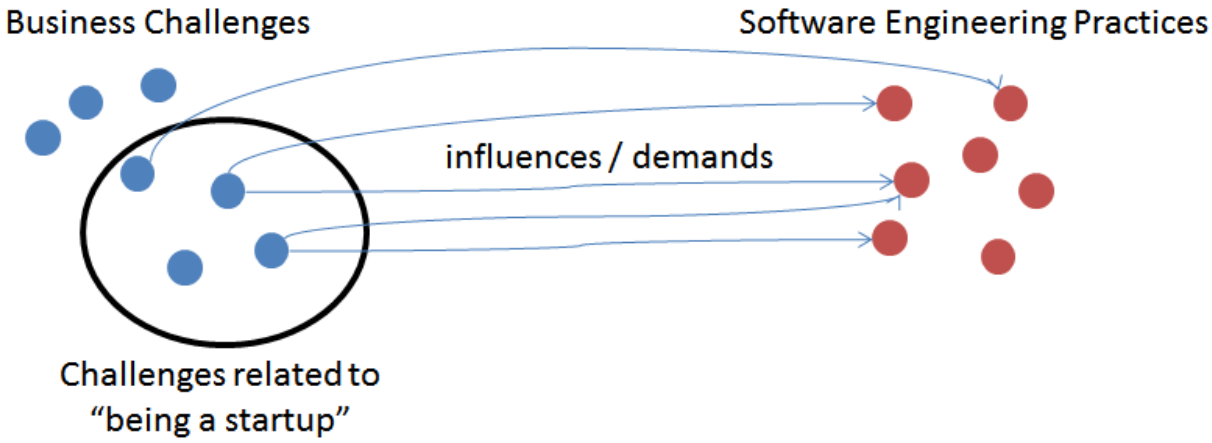


Figure 3. Pattern selection

In the fifth stage (step 7), the chosen patterns were workshopped and elaborated into more complete representation using a modified version of the Coplien format [Cop96]. In a pattern workshop, each pattern has a dedicated author, and the patterns are subjected to discussion among the participants. The procedure follows the one used in PLoP workshops (see e.g. [EuroPLoP]). The outcome of these workshops is presented in the patterns of Section 4. This report is the first publication of the patterns (step 8).

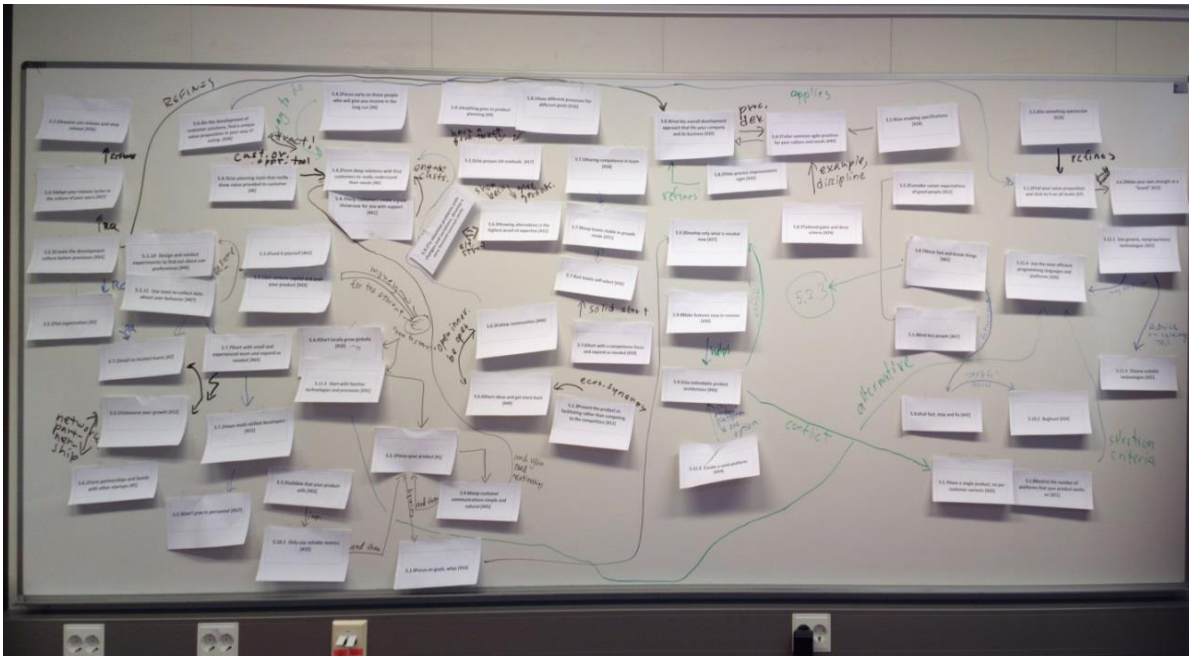


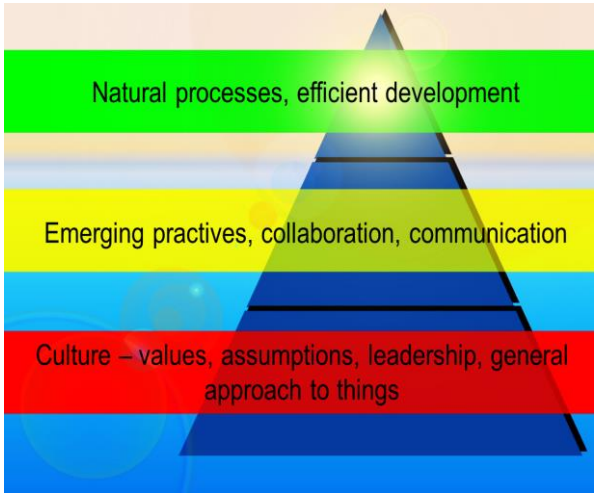
Figure 4. Building relationships between the patterns

In a final stage a holistic view of the whole pattern set was taken by considering the relationships between all the patterns. This was done by placing the patterns on a whiteboard, and drawing arrows between the patterns to indicate specific relationships (Figure 4). The relationships are assumed to serve as a first step to build a complete pattern language for software startups. Typically, certain patterns assumed the usage of some other patterns or refined other patterns, but some solutions provided by the patterns were also found conflicting. This is not surprising as such: in different kinds of contexts, even opposite kinds of solutions may be required. The main outcome of the relationship session is presented in Section 6.

4 Software startup patterns: elaborated patterns

This section introduces the 14 software startup patterns that have been elaborated into full-fledged patterns using the format given in Section 2, using pattern worshops. The title contains in brackets the identification number of the pattern. The order of the patterns here follows the order of the categories in Section 5.

4.1 Create the development culture before processes [#54]



Category

Culture

Context

Often, when a new company is formed, all the elements of organizational activity are missing – there is only a core team of people and its competences, a goal and some vague visions of how to proceed. Everything else must be built. But what is the right order of building the more mature company?

Problem

At the beginning of the company's lifecycle conscious efforts must be made to improve internal things that most influence success. Quite soon the team must be capable of producing software systems that provide good value to the first customers. That ability requires many things and decisions must be made upon what to develop first? Should processes be the first priority or something else?

Forces

At the very start, things are simple. There is typically only one product, which is not yet complex and one carefully selected pilot / key customer whose needs the developers can concentrate on. But things will change later and everything that aids

the company's competences should be developed for the future. Processes are something that are needed when things get more complicated, the product grows and there are new customers. But what the process should be like is not known yet, as they are just tools that support the future needs of development and business. There is a need to build those cornerstones of success that will carry the company through all the changes and aid in being flexible and in working in the team and the customers. Startups are usually based on people, shared goals and identity and ways of releasing their motivation to correct activity. That is the organizational and leadership view to success and that sometimes contradicts with the views of traditional software engineering. Culture is also hard to change, so it needs to start on a right foot.

Solution

Consciously develop from the beginning a company culture that supports what you want to be. (Note: Culture here means the values of the company, company's identity, assumptions, general ways of activity: what we are, what is special about us, how we approach things, are we for example more creative than systematic, or do we put preference to human issues over technology etc.)

For example, if your key success factor is flexibility, develop competence, general professional skill, collaboration, quality culture and not rigid processes (on the approaches, see Find the overall development approach that fits your company and its business [#39]). Develop processes only when you reach the stabilization phase and know what business and process development goals the processes should support (see Time process improvements [#32]). Let organization evolve organically (see Let teams self-select [#56], Keep teams stable in growth mode [#55]) and minimize distractions and organizational violence.

Also, select the first recruits based on cultural fit, not only on technical competence ([#59]).

Consequences

- Good culture provides capabilities that are essential for the first steps of a company and allow building of practices on a solid basis.
- Too much focus on culture will leave technical debt, so all elements of the activity need to be monitored for whether they really meet the needs of business and software development. The key persons must be aware of the changing situations and of when the company is moving into next phases of their lifecycle. Sometimes that can be hard to see, if all energy goes into everyday tasks.
- But when processes are weak, team and personal competences are critical. Because of that, the teams should be kept stable so people understand how others do things and how the collaboration works, and developers must be very competent.

All this is essential for startups as they have the most critical demands for successful building of overall capability. However, for established companies, the same focus to culture may improve flexibility, when businesses, customers or technologies change.

Related patterns

- Time process improvements right [#32] – about the timing of improvements, assuming that the culture is in place
- Find the overall development approach that fits your company and its business [#39] – this is the operational style that mirrors the culture
- Let teams self-select [#56] – teams grown organically are prone to finding good practices themselves
- Keep teams stable in growth mode [#55] – stable teams can find their own style of working
- Start with a competence focus and expand as needed [#59] – with low process orientation, everything depends on people

Example story

A company that does development projects for customers on many domains sees this as the very key to their success. For them human-centric culture is everything and a key driver for working with the customers, developing systems and for the internal organizational activity and development. Thus, culture is a key differentiator that supports all of their business. A practical example: Often, when a person enters a traditional company, she will be given a quality manual or a technical book to read. This company gives the new recruit a book about communicating with people to read. But that does not mean that the key practices and workflows are not developed. They are, but in a flexible way, relying on the teams to choose how they work.

Another company with a similar business model realized these issues when their business faces problems at the very start, after first customer projects. The leaders become aware that something was clearly missing – not competence, not process capability, but the basis and glue that holds things together -- the overall culture. They immediately started to build that and got the company soon on the right track towards success.

4.2 Self-funding [#62]

Category

Funding

Context

You are starting up a company or entering a new phase and need funding. The idea/product/domain is such that self-funding is possible. This pattern works at least 1) for early stage startups, 2) when the idea is difficult to fund, 3) when founders want freedom in making decision, or 4) when you want to encourage creation of something that is financially viable.

Problem

The primary problem is “How to create a product when there is a lack of funding?” Secondary problems related to this pattern are “How to keep freedom

in decision making/control over the company?" and "How to encourage yourself to build something that can be successful?"

Forces

Need money: Companies generally need funding to start-up, to build, to live and to grow. There are exceptions to this rule: some ideas/companies/products might not need any funding for the company to function. But even then the people working at the company need money for living.

Getting funding is difficult: Sometimes you need to self-fund to get the ball rolling. You can then aim to get funding from elsewhere later, or aim to make the company financially viable during the self-funding period.

Preserving outside influences: Retaining the freedom in decision making/control is important to some founders. Self-funding retains these freedoms.

Product validation: With self-funding, the need to create a financially viable company/product is greater than normally. This need creates drive, which encourages creating something that sells and is thus financially viable.

Solution

Do not seek external funding, but fund it yourself and do the work yourself or (at a later stage) get funds from acquired customers (= customer funding). When bootstrapping, it's quite popular to 1) work only part-time on the product, 2) get money from consulting, 3) use savings, or 4) use other means of self-funding.

Consequences

- Freedom: By bootstrapping, you do not rely on external funding and there is no-one else controlling anything, such as steering decision towards their interests
- Lack of funding forces you to build something that works, something that generates revenue and funding. It forces you to find ways to earn money.
- Bootstrapping for too long: you need to set a deadline for bootstrapping, because it might hinder your growth and let the competition run you down.
- Worst-case scenario: Wasted all your savings, wasted time and money trying to get the company off the ground.
- Competitors might get ahead, because they might have the funding

Related patterns

- Validate idea [#63]: One possible motivation for self-funding is product validation.

Example story

The son of the founder of an interviewed company was playing in a football team and they were having difficulty finding a sponsor. This personal interest and motivation has lead him to the idea of a platform where talents can find and connect with sponsors and vice versa. The founders have funded the company

from start from their own personal funds as they believe the idea will thrive soon or later.

4.3 Validate idea [#63]

Category

Funding

Context

Historically, engineers tend to get an idea, acquire funding and start developing the product right away. They never stop to think and validate the idea. When a company realizes this, they often follow this pattern. This pattern solves the problem of idea validation that is inherent in the old idea-oriented process. Many companies create products that simply won't sell, especially in the B2C sector. If your goal is to sell something and get money from that, you should first validate your product and not waste time and money. Be lean and validate your idea in a lean way. You seldom have a need for a truly comprehensive validation, but instead just a view from "the outside". This is essential for a startup company in any phase.

Problem

How to validate your product in the sense that it will sell? How to ensure you are building something that is financially viable? How to make sure that your product is desirable?

Forces

It is naturally impossible to know beforehand, and without research, if something is financially viable. You might also have doubts or uncertainty: you can't be sure if you are building "the right thing". Investors often ask "How does your product sell?" "What is your plan for selling it?". Validation will give you better answers for these questions.

Solution

Be lean. Customer discovery => Validation of selected criteria (e.g. will people pay for it) => Creation. Ask around and validate your idea. Seek validation most from possible customers, but also from investors mentors and experts. Use a slide deck, paper prototype, or some other quick and dirty method to get the basic idea across. Try to get a few paying customers before building anything. Use zero-budget approaches to see if people need the product/service. E.g. if you are building a nanny hiring service, post an ad on craigslist to see how much demand there is for finding a nanny. Use a newsletter and marketing approach [1].

Consequences

- Lowers the risk of ending up with a product that is not financially viable

- Danger that your validate using a small niche batch of people, that don't represent the actual users
- Customers might get a feeling that the product is of low quality if you use shoddy prototypes
- Might not sell after all, even if validation seems good

Related patterns

- Time process improvements right [#32]: it's hard to forecast future. Yet, you should validate at the correct moment.
- Minimal Viable Product pattern: in both patterns the point is to be lean and move forward with the simplest version.

Example story

The founder attended networking events to find possible customers and opinions of people. Armed with nothing but a slide deck, went to pitch for customers. Customers liked it and wanted to help build it.

References

[1] Casual Corp. 2013. How to go from zero to revenue in under five weeks without building a product. http://notes.casualcorp.com/post/55805301226/how-to-go-from-zero-to-revenue-in-under-five-weeks.

4.4 Keep customer communication simple and natural [#45]

Category

Customer

Context

When a customer can trust in a product/service a startup is providing to her/him, it is easier to establish a natural and simple customer communication between

the startup and her/his customer. When the communication is natural it is easier to learn about the customer's way of thinking and culture to which the startup is developing its product. Added to that when the communication is also simple enough, the setting it up doesn't take too much effort from the busy startup. This pattern is an essential issue throughout the lifecycle of startups.

Problem

Startups are mostly getting feedback to development. They have limited resources to put effort on a customer relationship management although this is important for establishing long term customer relationships. How to keep the customer communication simple and natural within these constraints?

Forces

Sometimes the startup can be forced to communication manners coming from the customer's culture which lead to formal or distant communication. Filling forms is not as natural as sending emails or talking face-to-face.

Very informal communication can lead to ambiguous requirements or decisions when it is difficult to have a consensus at the critical moment (e.g. changes needed in the product development).

Using a social media can work in some context but is not a solution to all.

In some cases a customer might want to talk about her/his business ideas or organizational issues which are not really a focus of customer product development. Being a good listener and a guide for relevant activities within the product development both are included to successful customer communication.

Solution

Build a trust relationship with the customer. Give customers natural ways to communicate continuously. As a main point of customer contact, be present at the customer's office often and communicate face-to-face. Informal communication should be unambiguous. Allow direct contacts by email or some other communication channels with which customers feel comfortable. When the startup has a larger customer base, a web communication technology probably is needed. Focus on the relevant and efficient communication forms.

Consequences

When behaving naturally and openly in communication with a customer, it is easier to establish a trust relationship and have a fully understanding of the requests of the customer.

Related patterns

- Flat organization [#3] - as people are committed to a common good, trust that their decisions are sound and they can make them autonomously (keep a flat organization)

- Focus on goals, whys [#14] - Find the real motivation behind wishes. What is the reason for that wish? Why is it important? What are the goals? Find out these things when discussing with customers.

Example story

When working together with a customer as a business partner or a friend instead of a software provider it is more natural to share thoughts and ideas, and also be honest if some wish is not realistic or feasible. Instead of seeing a customer formally once a month, a startup goes into the customer's office often. Startup wants to share a product version as soon as it is ready for a validation. Design alternatives and solutions are discussed openly throughout the design process.

Customer communication is also socializing with the customer. But a time for socializing is spent cleverly that it doesn't delay the product development.

4.5 Don't grow in personnel [#57]

Category

Organization

Context

The start-up company is in stabilization phase and the team has fused together. The company has utilized Make your idea into a product pattern [#68] and they have a solid working culture.

Problem

Startups are prone to follow traditions and domain customs. One of them is growing the personnel count and becoming a "proper" organization. But there is no law that states so.

Forces

Usually the pressure to hire more people stems from having too much to do with a few individuals. A team is more efficient than single individuals. However, there is a limit how much a team can handle.

Usually a successful business acquires more customers, causing more work. Even if the product is fully productized, it may require more support for the growing customer base.

However, adding people during a project rarely helps to achieve time goals, as absorption rate is so slow that effective work time is lost while instructing the newcomers [1]. Any person, no matter how competent, will be a new-comer for a certain time as working cultures differ between organizations.

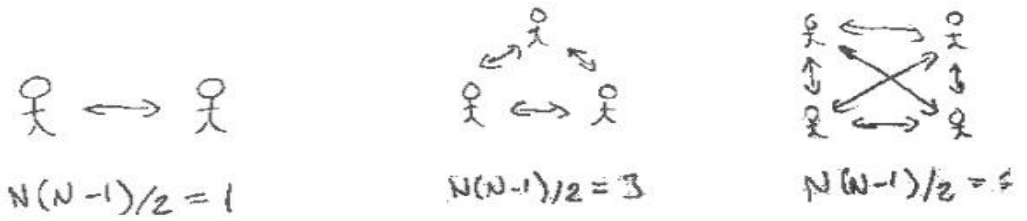
Companies' success is often measured in volume of business and its growth rate. This creates a mindset to grow. However, this kind of growth may be driven by ego and greed, not solid business strategies [2]. Investors may aim at short term profits and even customers may want only to deal with a company that grows.

However, a business that is sustainable and does not burn fast, is more profitable in the long run.

Solution

If a small size is good for you and there is no need to get more resources or competences, don't grow. Period. If your team size resides in the sweet spot, you are off good. Experts claim that 10 or 6-7 person teams are the best [3]. Christopher Alexander's studies suggest two to eight people in a group; see Small Work Groups [4]. In same pattern, Takano is referenced to have results for five people being most efficient configuration.

If your team is still too small to handle the workload, you should Phase It In [3] and grow gradually. It should be kept in mind that adding a person to the team increases the amount of communication channels exponentially; by the formula n(n-1)/2, where n is the amount of people involved, see the figure below. Additionally, people occupy space, so the facilities may not be enough. For example, there should be a big enough meeting room for accommodating everyone.



The amount of communication grows exponentially

People usually want a better company, not a bigger one [2].Thus, improving the work environment pays off in more committed people. However, you can't trust that your personnel will always be with you, even if you Bind key people [#67]. Even if they stay, it would be nice to have some kind of a career. So, you have to integrate newcomers to your team. If new people come in, the old employees should still have their roles (Keep Teams Stable In Growth Mode [#55] and Stable Roles [3]). If possible, Let teams self-select the newcomer [#56] (also Self-selecting Teams [3]). Give the newcomer an Apprenticeship [3] and have her to do specific tasks (Generics and Specifics [3]).

Consequences

- Your team is stable and the Community of Trust [3] is working well. No time is wasted in absorbing newcomers into team. Your business focuses into more product-oriented way. You can scale in product, not manpower.
- The company wage costs remain at low level.
- It is easier to do pivots as the size of personnel is small and people can be assigned quickly new tasks.
- There is no need for financial squeezes as the company consists only of the bare minimum amount of people.

- However, if people leave the team, there will be stress and tension as turnover culture may be underdeveloped.
- In the strictest sense, a startup is a temporary organization aiming to become a real company. Thus, keeping your options open for too long will prevent your transition into a proper company. The startup will be in a constant state of change and never entering the growth phase.

Related patterns

- Outsource your growth [#12] – another way for growing
- Small co-located teams [#2] – the optimal team structure
- Form partnership and bonds with other startups [#5] – one way to more work done with aid of other start-ups
- Have a single product, no per customer variants [#20] – one way to save in work effort
- Start with a competence focus and expand as needed [#59] – who to recruit if growing
- Keep teams stable in growth mode [#55] – how to preserve existing team dynamics
- Let teams self-select [#56] – a way to hire new people to team if needed
- Make your idea into a product [#68] – a way to scale business without adding personnel

Example story

A small company has been selling their blog service product for a while. They have enough talent to market and develop their product, so no additional team members are needed.

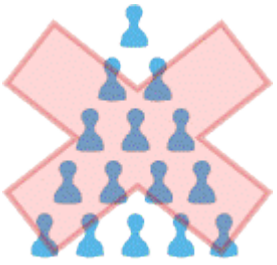
References

[1] Frederick P. Brooks, Jr. "The Mythical Man-Month". 1995 [1975]. Addison-Wesley

[2] James O. Coplien, Neil B. Harrison “Organizational Patterns of Agile Software Development” 2004 Prentice Hall

[3] Semler, R. (1993). Maverick: The success story behind the world's most unusual workplace. New York: Warner Books.

4.6 Flat Organization [#3]



Category

Organization

Context

Your start-up is growing and has passed the initial steps towards success. Organization grows and has already around 20 persons. You have established organization culture and Develop the development culture before process [#54] has been applied. You need to decide what will be the structure of the organization as the old ways of communicating

become too slow as more and more people are hired. In addition, someone needs to make business decisions in the organization. One person or a few persons, i.e. the founders may not have time anymore to be involved in all decisions.

Problem

As the organization grows, one or a few persons might not be able to make all decisions as they cannot master all the things that go on in the organization anymore.

Forces

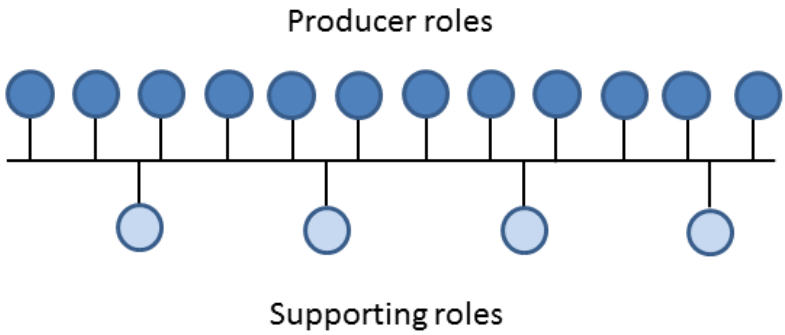
Communication via an intermediator takes time. If all decisions need to be addressed to a certain person, the decision making in the organization becomes slow. As the organization is a startup and thus interacting in highly turbulent environment, the decision making should be fast. In addition, if all or most of the decisions are made by a few persons they might become a bottleneck as the organization grows.

The best decision maker usually is the person the decision to be made is concerning. Thus, the developers should be given the authority to make decisions that concern their way of working or the part of the product they are working on. In addition, to be able to follow lean principle decide as late as possible [1], the person who knows the last responsible moment, i.e. the developer should be allowed to make the decisions. If someone else is making the decisions, the response times increase as handover may take some time and have wait states. Thus, the last responsible moment may have already passed when the decision is made. Drucker also argues in [4] that the decision making should be left to the knowledge worker the decision concerns.

Autonomy is a good motivator for knowledge workers (see e.g. [2]). As it is in the high interest of the startup to keep the people motivated, enough autonomy should be given to the developers.

Solution

In order to keep the communication fast and allow people to have autonomy in their work, keep the organization structure flat (see figure below). As people are committed to a common good, trust that their decisions are sound and they can make them autonomously. So, the corner stone of Flat Organization is casted on Community of Trust [3] and 54 Develop the Development Culture Before Process. The organization culture guides the individuals in the organization to make correct decisions. People self-organize themselves into the teams or with slight guidance from the supporting personnel (see figure below. If the team needs a leader, it self-selects it from the team.



The figure above describes the structure of Flat organization. There are only a few supporting roles in the company and developers who are all peers.

As the company grows there might be need for some supporting roles that take care of everyday company routines, e.g. training of the staff, arranging summer parties and offering the teams new projects. For example, sales people are supporting roles, but also the producer roles may include sales. The supporting roles should act more like coaches of sports teams and they should try to enable people to perform at their best. They should not try to lead the teams or make decisions for them. Team members can ask the supporting personnel advice on the decisions, etc.

The person who is facing an important decision usually has most of the contextual information. She only may need help and consulting on the big picture, but she doesn't need the decisions made for her.

Flat Organization gives the teams opportunity to decide on important matters for themselves. This also makes it easier to maintain direct customer-supplier connection which is one of the important lean principles [5].

Consequences

- Decisions can be made quickly by the people having the most contextual information. This enables the organization to quickly react to the changes in business environment.
- No extraneous bureaucracy involved in the decision making. If a person needs a new laptop, she has right to buy that. Of course, such approach requires trust and the employees need to be trustworthy. This also enables people to work autonomously which might be motivating and lead to better well-being in work.
- New employees may feel little disoriented when they are hired to the organization. There are no managers to guide the new developer to her work. She needs to find out her own position in the organization. This problem can be alleviated by writing a handbook for new employee. See e.g. Valve's handbook for new employee.
- Flat organization requires that all hired developers are outgoing and can communicate with other workers to find suitable place in the organization. This might require more selective approach to recruitment of new employees.
- If there are members of multiple organizations in the start-up effort, e.g. project, it might be hard to achieve flat organization structure in the start-up as

there are different factors in play. For example, amount of funding, previous experience, etc. might lead to situation where one party might want more power to make decisions and thus preventing the use of flat organization model.

- Autonomy of people also brings responsibility for individuals. One should give credit for good decisions but not to punish persons because of bad decisions. Each individual must take responsibility of bad decisions and take care of the consequences.
- As people can decide for themselves, the decision making process may lead to local optimization as the persons does not have the big picture. Thus, one should make sure that the company vision and culture is well communicated to each employee so that they can make decisions optimizing the global maximum.

Related patterns

- Small co-located teams [#2] – Flat organization relies heavily on the teams that are co-located and are formed on demand basis.
- Have different processes for different goals [#16] – Flat organization does not force any process; teams can select freely which process model to use.
- Start with a competence focus and expand as needed [#59] – Flat organization requires good personnel to function properly. Thus, start-up needs to hire only the best people available. One might also consider applying Have multi-skilled developers [#53].
- Create the development culture before processes [#54] – Development culture is important in Flat organization as teams self-select the process they use. Thus, the culture is of high importance to guide the teams to select the right processes.
- Let them self-select [#56] – describes how managers appointed outside can disrupt the team and thus the teams should self-organize and select their own leaders.

Example story

A small software startup had about 8 employees. Once the company started to grow and had around 20 employees, they realized that they need to decide on the organization structure. Would the founders of the company quit programming and become project managers, product owners and newly hired people would then do the programming. They decided to use Flat organization pattern to form the organization structure. Company did not force any roles to any of the employees, but gave power to anyone to decide on different matters. If a project manager was needed, someone would then become project manager for the project. However, it wasn't forced. Developers could decide on their own how to organize the work, when to buy new laptops, etc. Each decision had to be rationalized though and the explanation needed to be ready if somebody asked.

The startup also wanted to have a few supporting roles. They hired HR manager who was responsible for hiring new talented developers. As legislation ordered, the company also had a CEO. In addition, the start-up had one dedicated sales

manager, whose job was to sell new projects. Lastly, they had a “janitor” role whose job was to take care of all other matters that nobody would take care of.

References

[1] Mary Poppendieck and Tom Poppendieck. 2003. *Lean Software Development: An Agile Toolkit*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[2] Daniel H. Pink. Drive: the surprising truth about what motivates us. 978-1594484803, 2011, Riverhead Books

[3] James O. Coplien and Neil B. Harrison. 2004, *Organizational Patterns of Agile Software Development*. . ISBN 0131467409, Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[4] Drucker P.F. (1999) Management Challenges for the 21st Century, Butterworth-Heinemann, Oxford.

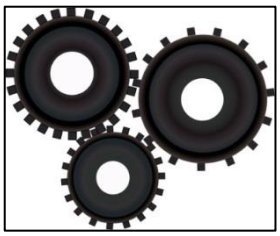
[5] Ohno, Taiichi (2007), Workplace Management. Translated by Jon Miller, Gemba Press, ISBN 978-0-9786387-5-7, ISBN 0-9786387-5-1.

4.7 Unique value proposition [#38]

Category

Competence

Context



Being startups, companies often need to make their ground on a competitive market to take off at the first place, and then to sustain. In this setup, a unique value proposition combining both technical and social values is needed to be adopted and practiced for long term sustainability.

Problem

Startups must have to stand out in the competition with established market players. Thus it is a concern to adopt a viable approach embedding a unique value proposition to support human centric services along with technical competences.

Forces

Distinctive competence is required to cope with dynamically changing market demand. Adoption to changing market needs is essential to contend with competitors.

Solution

Find a unique approach that differentiates the company in the market and is good for business. Make the approach visible as a part of the brand, and cultivate it within everyday activity. For instance, adopt a human centered development approach that incorporates values, e.g., economic, functional, emotional, and symbolic. Then find a conscious way of action/competence to practice the adopted approach which would effectively separate you from the competitors. Emphasizing on communication skills in order to better understand the customers and their needs would be a bonus. Market the competence as a service to the customer. Gradually develop competences, practices and tools within everyday practices.

Consequences

- Company stands out in the market
- Easier to find customers
- Unique and natural way of doing things would ensure increased motivation and satisfaction to the involved personnel, e.g., employees and customers
- It might be an extravagance for startups to focus on issues other than the product
- Building such unique value proposition may require additional resources

Related patterns

- Form deep relations with first customers to really understand their needs [#6] - Deep understanding of the customer is needed to identify all relevant business issues rapidly.
- Use planning tools that show value provided to customer [#8] - Some planning tools are always needed to demonstrate things in a quantitative way to customers. This in a way built customer value.

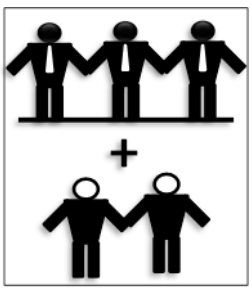
Example story

A company that develops mostly customer solution has chosen human centeredness as their key value. It is part of their brand and shows clearly in their project practices and also internal activities, where they wish to develop their personnel as human beings that are in the path of fulfilling their natural role in ICT. This profiling compasses everything they do and clearly separates them from larger competitors.

4.8 Start with small and experienced team and expand as needed [#64]

Category

Team



Context

Startups usually try to enter a market full with competitors. The timing is essential for entering the market and no time can be wasted in anything but on product development related activities. The communication among the team members and

their experience matters when avoiding operational overheads.

Problem

How to enable smooth product development with fewer overheads?

Forces

The amount communication increases with the size of team. An increased communication complexity can slow down the development. An unexperienced team member will require time to learn about the focused area and has to acquire skills.

Solution

The team size matters when it comes to the communication overhead. A team of 4 to 5 people can be considered a small team; however, the definition of a small team may vary from person to person and from project to project. The co-location of team members can also bring added value. Many tacit communications and understandings can be preserved in a co-located small team. The experience of team members always matter, however, in a small startup context it can make a huge difference. The team lead should hire people qualified enough for the job as well as multi skilled. The experience smoothens the work flow while the small size of the team reduces the communication overhead.

The experienced team members are asset to the organization. In growth phase, they should be provided with opportunities to grow in skills. For example, the company could support their participation in seminars and courses at the academic institutes. Furthermore, their work should be appreciated at every level. In the growth stage, risks associated with hiring un-experienced team members reduces. Thus, then un-experienced team members can be hired and trained for the job by the experienced members.

Consequences

- Comparatively less operational overheads in startup phase.
- A good quality product can be achieved by a starting company within reasonable time.
- The experienced team members may have arrived from different organizations with different cultures. A conflict of cultures may arise.

Related patterns

- Small co-located team [#2] - Small co-located teams involved less communication overheads.
- Flat organization [#3] - Less hierarchy leads to less communication overhead.
- Do something spectacular [#18] - To standout a startup must produce a product better than the competitors. The experience in the focused area is one of the precursors to reaching this goal.
- Consider career expectation of good people [#52] - Career oriented people like to work in startups where they can excel. In an environment where everyone is well experienced there is great opportunity to learn new things.

Example story

A startup that develops content marketing solutions understood that a small team of highly competent multi-skilled people is the best way to start a company. It was seen essential that all developers have years of experience and true competence for developing the critical first products. Small team would be efficient, well communicating, could adapt new ideas and technologies fast and could thus really push the development. Less experienced developers could be hired later as needed should there be a need for expansion.

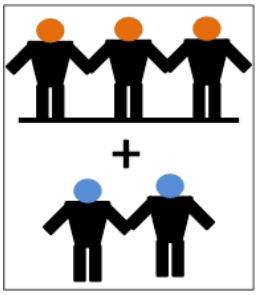
4.9 Start with a competence focus and expand as needed [#59]

Category

Team

Context

A startup usually has limited human resources and skills portfolio. Some may have competence to build web systems while others may be good at embedded system. Sticking to one specific competence area could be useful in the long run.



Problem

How startups can build their image as being pioneer in one competence area and capitalize on it?

Forces

To be pioneer in many competences is challenging and resource consuming. On the other hand, to be best in one competence area is better than to be average in many.

Solution

A small team with focused competence can be built by hiring people with skills and knowledge in the targeted competence. The experienced people can speed up the development and research needed to reach the goal. Growth opportunities can also be provided in the form of courses and seminar to the team members to further excel in their skills. The experienced people are assets to a company and therefore they should be entertained with rewards to appreciate their hard work. This also motivates the people to stick to the company.

The focused competence could also lead to generation of new knowledge during the product development. The company can also capitalize on the knowledge, for example, patenting the knowledge is one of the options. Furthermore, the generation of new knowledge gives the company recognition as a leading company in the focused competence area. The company can market their image as the best in the focused area. Once the image is recognized, many established competitors may also wish to do partnership with you. On the extreme, the competitors may want to buy your company in ordered to own the knowledge base you have created.

If needed, the company can also expand to new competences by hiring people expert in the new competence area. Timing of the expansion is crucial as it takes time to find good people in the new competence area.

Consequences

- Company's technical knowledge base will expand rapidly in a focused direction
- Company can grow comparatively faster in the markets requiring the focused competence.
- Competence is a unique value proposition.
- Focused competence could lead to focused products.
- Customers tend to trust products from companies well known for a specific competence.
- Startups with narrowed down competences have little flexibility to quickly venture into new markets requiring a different set of skills.
- A startup focused on a single competence may develop an inertia that will make it difficult to build portfolio for new competences when needed.

Related patterns

- Consider career expectations of good people [#52] - People are interested in long term career wise advantages of the skills they learn at a company or startup. The career oriented people will be more interested in joining an organization with focus on the competence they want to build career in.

4.10 Time process improvements right [#32]

Category

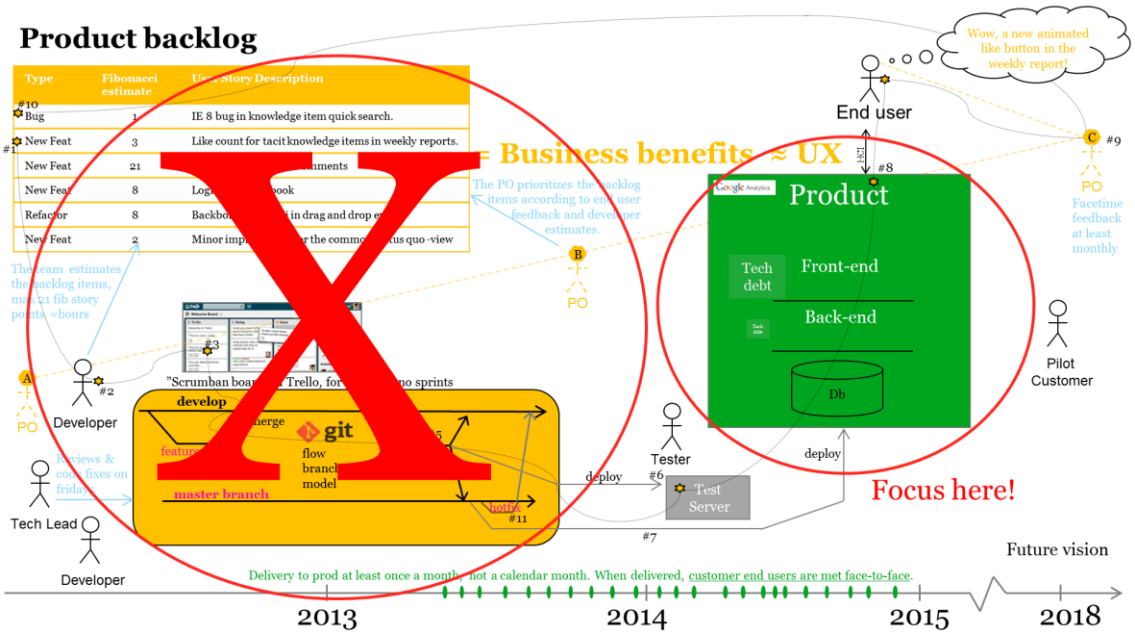
Process

Context

Startup is in growing phase and both the product and the software evolve. The development team has some process it uses to develop the software. The process may consist of scrum like agile practices.

Problem

It is important to pay attention to process improvement, but the timing for improvements should be right. At some point, in preparation for growth, the startup needs to plan for better processes and for example workflow tools. But doing this may require the focus and effort of the best people in the small company. That is why the improvements must be planned and time carefully.



The software development process may be complicated.

The process is never perfect and the team can always improve the process. In general, the team should pay attention to the product, not the process. If the product is good, the software process is mostly only matter of programming and releasing the software often to the end users.

Forces

The team may face difficulties with the process. The product may not evolve quickly enough. The quality of the software may not be on a high level. The team is forced to improve the process at some point.

Solution

Plan and time process improvements right. Have a retrospective scheduled and use the outcome of the session to improve the process. Do not improve all the time – focus on the product instead. There is a right time for process improvements.

Consequences

- The process is improved, but the improvement itself does not consume all the resources of the team all the time.
- The team has the focus in the product.
- The software development process is good enough.

Related patterns

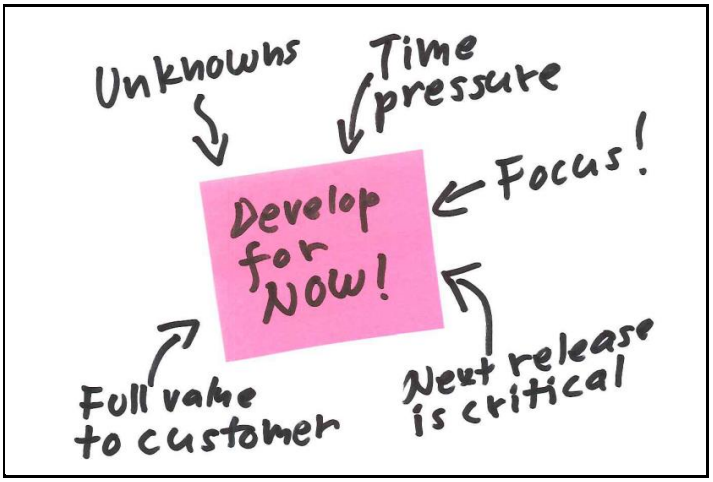
- Develop the development culture before processes [#54]

Example story

A company that develops content management system is currently in the stabilized phase, in which they can manage their product development and quality assurance with rudimentary manual workflows. They clearly understand that when they move to

a high volume product business, they will need more defined processes and better automation for example for deployment, virtualization and testing. But while the vision for future needs is clear, they have yet not done the developments, as there is still uncertainty over the technologies the new tools should support and the process technologies and tools change every year. So they tolerate the "process debt", but gather ideas and readiness to design and implement the new tooled workflow when exactly the right time comes.

4.11 Develop only what is needed now [#27]



Category

Design

Context

Startups often think of the extensibility of their products. It is clear from the beginning that the fist versions will be extended later for certain customers or to the general market. This is an essential issue in all startup lifecycle phases.

Problem

How to tackle the extensibility issue? How to find the best basic approach to development that makes the extension and alterations as easy as possible?

Forces

This is a hard question, because so much of the future is unknown and it is difficult to plan for what is not known. Also, the future directions of development are very unclear, so planning ahead might cause working for something that is not needed. Traditional advice is to generalize any designs, thinking that new implementations, extensions and alternatives could be easily derived from the generalization than by refactoring a current implementation. Yet, all the available resources and time are barely sufficient for creating the implementation that is needed now, for the first customers or for testing.

There may also be time pressures: customer deadlines, time to market – there is never a second to waste.

Solution

For efficient extensibility, develop only for what you need soon: what this customer requires, what the next release should include, what the current user stories require. Don't generalize designs. Don't implement for the next project. Plan only for what is known.

This is very important for startups due to the lack of resources. Once the directions for products stabilize, other strategies may gain value and that applies to later stages of companies.

Consequences

- Due to a focus of direct activity, this pattern helps maximize the speed of feedback, and improves the efficiency of development and increases developers' motivation. Development is fast, as less time is spent on architecture and design – which could prove later to have been spent on wrong things and to be waste.
- When carried too far, in too many situations, this pattern will produce technical debt, especially if the focused designs are not done professionally. Emphasis should be placed on architecture and refactorability of implementations (see Make features easy to remove [#30]) and the focus on the competence of developers (see Start with a competence focus and expand as needed [#59]). Good scalability of the selected technologies helps here (Choose scalable technologies [#25]).

Related patterns

- Choose scalable technologies [#25] – another view to preparing for future
- Make features easy to remove [#30] – architectural and design choices
- Start with a competence focus and expand as needed [#59] – about developer skills

Example story

A company that develops new innovative information systems for industrial businesses customers faces in all its projects the question of how the systems can evolve later. Should they abstract the business processes the system supports and base the first implementations on those abstractions, or just design for the direct, concrete needs of the current situation? The company has found the latter tactic to suit them well. The choice is aided by the good experience of the developers, which helps minimize technical debt.

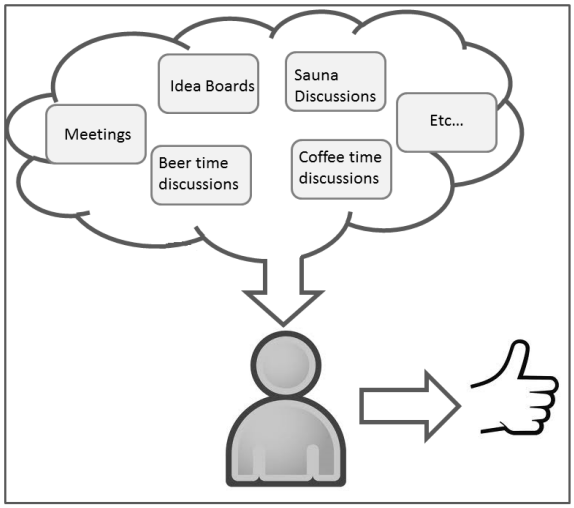
4.12 Anything goes in product planning [#9]

Category

Design

Context

In any company, preparing an effective product plan is necessary. Gathering new product, new system concepts and new features is not easy. Startups follow all the possible ways to get new ideas during product planning.



Problem

How to make a product plan? What are the right ways to do? Is there any best way? How to find them?

Forces

Product plan determines how work is carried out at various stages of building the product and the required aspects while building the product. Startups have very limited resources and very less number of team members. Maximum use of personnel and resources is necessary. Thus the team needs sharing and approving new ideas. According to their convenience, team preferences, convenience, culture and visionary, startups follow their own procedures during product planning.

Solution

It is very important to prepare a good product plan. But, the procedure - how it is done and where it is done is not really important. Procedure to prepare a product plan is based on working style that suites the situation in the company.

General steps in preparing a product plan includes - understanding the problem, thinking about alternatives, based on the working style of the members in the team gathering ideas by all possible means - meetings, workshops, coffee time, idea boards, sauna time etc…, validating gathered ideas by person or team in-charge and sending approved ideas as input to the development of the product. Use planning tools that really show value provided to customer [#8] – tools to plan various design alternatives that lead to success.

Consequences

- Generates better ideas. As ideas are gathered in all possible ways from every Individual in the team. This also results in maximizing the use of resources.
- Supports good collaboration in team. Since team members has lot of scope to discuss about the product whenever and where ever possible.
- Sometimes there can be too many ideas and to determine whether to consider all or not may take lot of time. Considering wrong ideas may lead to wastage of time and resources.

Example story

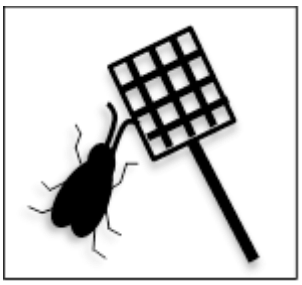
In a company ideas are gathered during meetings and from idea board, where new ideas are written by the team members. All these ideas are analyzed by the CEO and the approved ideas are sent to the development team.

4.13 Bughunt [#34]

Category

Testing

Context



New startups suffer from the urge to deliver functioning product in shorter possible time. This leads to investment of all the resources to development of features while software quality is ignored. For stable startups with good resources, the situation could not be any better. The focus is on development while other aspects suffer due to the lack of interest. Establishing a separate full blown software testing team is not feasibly for freshly starting startup. The limited testing setup at the startups could not uncover all the issues in a product.

Problem

How to improve the product quality within the available resources in a startup?

Forces

Startups usually have limited resources for example staff, money and time to setup complete testing department. The general quality of the product declines as everyone is keen on developing new features and honing the quality seems a dull chore. Fresh startups usually lack experience in software quality and its effect on product/company's image.

Startups are keen on reaching the market as soon as possible for many reasons. For example, to get the return on the investment to sustain the operation, to get the feedback on the product and/or to get there before anybody else who is working on a similar kind of product etc.

Solution

It is essential that members of a startup understand the importance of product quality. Perhaps, the team lead could present the team members with case studies where software companies were successful due to their better quality compared to the competitors. Software issues or bugs lead to reduced quality. It is always wise to resolve the bugs as they appear. However, special bug hunting sessions could be arranged by the team lead. Bug hunting in essence related to the concept of exploratory testing and modern hackathon methods.

In a bug hunting session, the aim is to find new bugs and fix them. The sessions do not require special experts in bug hunting or lot of resources. Everyone on the team can participate in a bug hunting session. The sessions can also serve as a socializing event (e.g., having sauna during the session) with informal atmosphere. So, that the participants look forward to future sessions. The frequency of the sessions depends on the commitment of the team to quality and of course flexibility available in their schedule. It can be one session every week or two or perhaps one every month. In a startup context, the sessions should not last more than a day.

In a bug hunting session, already known bugs can also be presented to the participants. The participant can select bugs (new/old) to fix them. Moreover, the team lead can encourage participants to take on the responsibility for the bugs. The code conflicts can be avoided through open communication. The individuals actively contributing to bug discovery and fixing should be rewarded (for example, a drink or name on the wall of fame in the office etc.). The team lead should update the bug tracking system as bugs are fixed or new are discovered. The team lead could end the session by reviewing what has been done and plan for the next session.

Consequences

- Product quality improves due to fewer bugs.
- People will see quality maintenance as an interesting and import activity.
- Positive code refactoring or design changes may also happen that could provide long term quality related benefits.
- A bug costs less when fixed early.
- Time will be taken off the development.
- Fixing of many bugs at the same time could result in many more possibly complicated bugs.

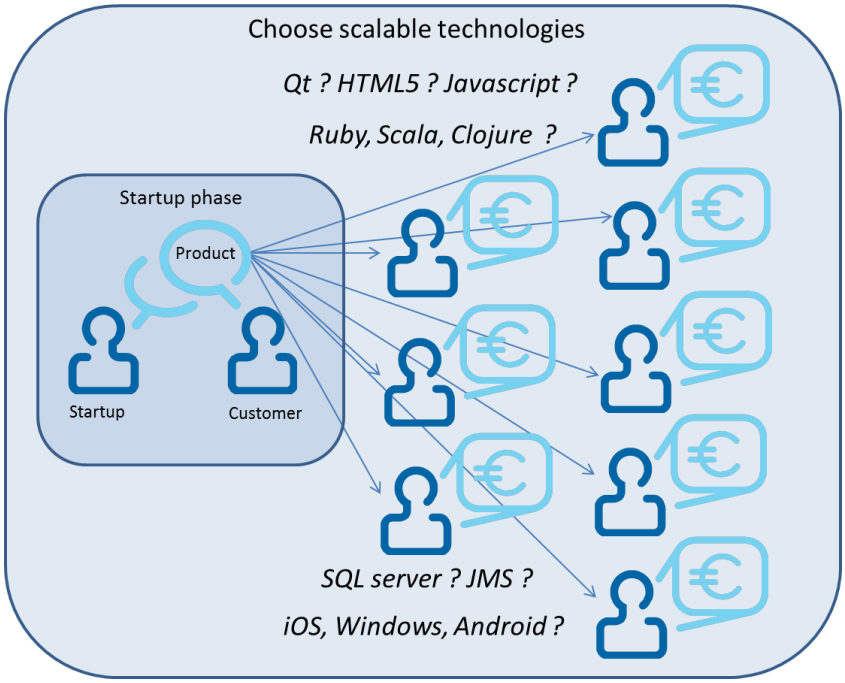
Related patterns

- Fail fast, stop and fix [#42] - Develop fast and freely with high risk of breaking things. Stop and fix it. This pattern also suggests stopping and fixing the faults before moving onward. One way to fix the faults is to arrange bug hunting sessions.

Example story

A startup creating their own products noticed that product quality is important for their customers. They decided to organize monthly bug hunt days. At the start of the day, they would choose a certain release to test. During the day, everyone would use the product and write down bugs. These bugs would then be fixed later on.

4.14 Choose Scalable Technologies [#25]



Category

Technology

Context

When a technology is scalable, it is flexible, efficient and handy to program. It also adapts well to a scaled use of the product. Whenever there were chances to sell the product in a mass market, a technology should not be hindrance for scaling up with the product. Thus choosing a scalable technology should be a part of design decisions. This pattern concerns a product development startup who wants to grow by scaling the use of its product in the market. In that case a startup has to be prepared for a variation in customer needs and a higher performance required. This pattern is an essential issue at the startup and stabilizing phase.

Problem

Good design is an essence for building a quality software product. How well do technologies chosen support the design goals when the product is used in the big scale (e.g. in a consumer market)?

It would be a mistake to need to change product technologies and tools at this point when people in the product development have spent months for learning and stabilizing those. There are many other more urgent things to focus on when the startup is a mode of growing and scaling up.

Forces

It is sometimes difficult to plan for a long term scaled product while focusing on the most important things now (see also Develop what is needed now [#27]). Also giving

up some not scaling up technology can be difficult at the early stage if there are still some paying customers for that. Finding the right timing to make decisions is not easy.

Startup also has very limited time and resources to learn new technologies from the product scaling's point of view (see also Time process improvements right [#32]).

Solution

Choosing a right technology is a part of architectural choices (see also Make features easy to remove [#30]). Use non-proprietary technologies which do not constraint a fast development and scaling (see also Use generic, nonproprietary technologies [#23]). Use scalable technologies from the beginning (see also Create a solid platform [#24]). Assess the alternative product and development technologies and choose the one that most likely takes you through to growth phase without switching.

Also worth noting is that startups usually make technology choices in a non-formal manner, based on experience or general feeling. Yet, if there is time for that, proper analysis of alternatives, with some fast trials can be recommended. At the very least, it should be checked what experiences others have with the possible technologies. This is something to ask in web-based forums and other social media channels.

Consequences

- Technology analysis and proper selection process takes time from productive work. Due to that, the choices must be made with a lean approach by the developers who know the alternatives and who can make justified choices.

Related patterns

- Use generic, nonproprietary technologies [#23] – use platform or software supplier independent technologies
- Create a solid platform [#24] – create a platform that supports scaling quickly
- Develop what is needed now [#27] – develop only for what this customer requires, what the next release should include and what the current user stories require
- Make features easy to remove [#30] – architectural and design choices
- Time process improvements right [#32] – about the timing of improvements, assuming that the culture is in place

Example story

For a company that develops advanced content marketing solutions to media corporations, other businesses and even individuals, there is a great need for technology that allows for a first-rate user experience than exceeds the state of the art at any time, for any specific use. As the needs of the customers are diverse and will diversify at an accelerated rate, the product platform and technologies need to be ready for that. One part of the problem is that this is not a case of just product or programming technologies, but applies to deployment and hosting too. The company bases it choices on freeform technology monitoring in developer societies and other personal "knowledge mining".

For this company, Ruby on Rails has proved to be the current choice, and a good one, but still it is understood that the technologies will evolve. But this choice is a solid one that will take a company trough to a managed growth.

5 Observed practices

The companies where the practice has been identified are mentioned at the end of the description, using a letter code. It should be emphasized that this does not mean that the practice would not be used in any other of the interviewed companies, but rather that the interviewers have observed the practice in the company.

5.1 Goals

5.1.1 Focus your product [#1]

Problem:
You have a small team and resources are limited. The focus of the product is unclear as there are multiple market segments and customers that the product could be targeted for.

Therefore:
Find the most potential customer segment through feedback and focus on that. Keep the other segments in mind for future, but for now focus on a single customer segment. Be prepared to change the focus rapidly by using agile methods if new information emerges. Retain focus by judging change requests by contribution to main vision. (Witnessed in companies J, F, H, G, B)

5.1.2 Find your value proposition and stick to it on all levels [#7]

Problem:
Ideally startups are involved in an investment decision making process at the customer. This should be aligned with the customer's business development. Also this kind of managerial consultancy can be a source of income for the startup because an own product development is not established for ringing up a profit.
Working on many levels of the customer organization enables discussions about the real problems with a diverse set of experts and decision makers. Also when the value proposition is introduced to the "right" people in the customer organization, it is more likely to be accepted.

Therefore:
Make a valid value proposition to the customer. Aim to get in the discussions with the experts from both strategic and operational levels. Introduce the value proposition to the people who really can make decisions about the investment. (Witnessed in company L)

5.1.3 Present the product as facilitating rather than competing to the competitors [#11]

Problem:
The competitor will resist your product from gaining ground in the market.

Therefore:
Develop a product that has facilitating aspect which may be attractive to the competitors. The competitors will see the product as savior than enemy. (Witnessed in company F)

5.1.4 Focus on goals, whys [#14]

Problem:
To find a proper concept / solution, one should not plan or design based on the customers' stated wants, because that level is prone to change rapidly. One should find out a deeper level that is more stable and based on which new designs can be created that just implement / express the real issue at a type and style which is currently the right choice.

Therefore:
Find the real motivation behind wishes. What is the reason for that wish? Why is it important? What are the goals. Find out this things when discussing with customers. Old anecdote from japanese auto manufacturing. The customer says that the car must be blue. Should we paint them blue? No. We need to find out why blu is a good colour for the customers. It turns out that blue expresses wealth and status in that culture. Now we can find more, better ways to express those things and not just literally paint on the desirability! (Witnessed in companies M, L)

5.1.5 Use proven UX methods [#17]

Problem:
UX is often the main success factor for startups. That's why it must be developed properly - better than the competition. Besides relying skilled developers, pracices should be used that help share the understanding inside company and help move into next lifecycle phases of the company.

Therefore:
Use proper UX development methods from the start, such as Goal Directed Design. Consider the speed of getting ideas validated (including prototyping). (Witnessed in company E)

5.1.6 Do something spectacular [#18]

Problem:
There is a need to stand out in the competition and in relation to bigger, established companies. Bigger companies are accustomed to their old solutions and bränds and are unable to really surprise the customers and the market. That leaves room for startups to show their special approach and competence.

Therefore:
Do something spectacular that really stands out! Create WOW effects that lead to WOW feelings. E.g. concentrate on excellent UX or an unique brand. That is the shortest road to income. (Witnessed in companies E, M, G)

5.1.7 Have a single product, no per customer variants [#20]

Problem:
Having per customer variants of the product results in poor maintainability and poor flexibility. Creates extra much work for development and maintenance.

Therefore:
Have a single product that is modular and highly configurable/flexible. A product that in a sense allows for different configurable package per customer, yet is a single product. (Witnessed in companies M, G)

5.1.8 Restrict the number of platforms that your product works on [#21]

Problem:
Startups have limited resources for implementing products for many platforms or platform variations. Handling of those and re-implementing features would consume all of the resources.

Therefore:
Make strict business decisions about what platforms you support. For example, support only the most important browsers and operating system versions (no old Windows versions for example). Negotiate with the customer what they are willing to pay for you to support. (Witnessed in companies E, G)

5.1.9 Use enabling specifications [#28]

Problem:
There is often too much work for every employee. It is good when developers and teams can work independently without constant, unrealistic, help from product owner or customer.

Therefore:
Enabling specification can be used to guide work efficiently without restricting design space. (Witnessed in company E)

5.1.10 Design and conduct experiments to find out about user preferences [#46]

Problem:
You need to know relatively quickly and with relatively low effort in which directions your product should develop to attract more users.

Therefore:
With a user base sufficiently large you can provide different variations simultaneously (e.g. in different areas, or randomly) and record the user behavior to get evidence which versions work better or create more signups etc. (A/B Tests). There are other ways to design such experiments too. What is common is an attitude somewhat matching that of a scientist who has a theory and designs experiments with the goal to validate or falsify this theory. (Witnessed in companies M, L, C, K)

5.1.11 Use tools to collect data about user behavior [#47]

Problem:
Startups serving to a larger community / consumer market need to gain visibility quickly but have only limited budget for marketing. This needs to be spend very focused and effective.

Therefore:
You need to control permanently the effect of your marketing activities and test individual channels before you can justify larger investments. To collect the data necessary for this, instrument your product and your web pages with tools such as Google analytics. (Witnessed in companies C, N, A)

5.1.12 Make your idea into a product [#68]

Problem:
Projects don't scale easily.

Therefore:
Make your ideas into products, instead of focusing on projects. (Witnessed in company R)

5.2 Culture

5.2.1 Outsource your growth [#12]

Problem:
Searching for skilled personnel and managing the growing team is difficult for new startups (with little HR experience)

Therefore:
Outsource the work to other companies while you focus on the product. (Witnessed in companies F, J)

5.2.2 Anyone can release and stop release [#36]

Problem:
Rigid release procedure postpones feedback from users.

Therefore:
Allow anyone to make a release any time they want to. Also, allow anyone to stop the release at any point if they want to. (Witnessed in companies G, J)

5.2.3 Create the development culture before processes [#54]

Problem:
At the beginning of the company's lifecycle conscious efforts must be made to improve internal things that most influence success. At the very start, processes are not that and they are likely to change radically as the company evolves. So the other things must be developed first and those things should be concentrated on that support for example flexibility.

Therefore:
Consciously develop from the beginning a company culture that supports what you want to be. For example, if your key success factor is flexibility, develop competence, general professional skill, collaboration, quality culture and not rigid processes. (Witnessed in companies E, P)

5.3 Funding

5.3.1 Get venture capital and push your product [#43]

Problem:
It may be tempting to develop your product in silence with low resources (as a hobby) and with a small user base, but with low or insufficient revenue. Despite the never dying hope that “one day” the product will “really take off”, you have no evidence that the venture may get profitable at all.

Therefore:
Push your product and try to get profitable within a reasonable time frame. If you succeed, it is a win. If you fail, you know and can allocate your resource to other endeavors. Don’t be shy to use venture capital if needed. However, contain the negative effects of a failure (e.g. by making sure that your loss will be clearly limited, e.g. by founding a stock company). (Witnessed in companies K, C)

5.3.2 Fund it yourself [#62]

Problem:
Funding agencies are reluctant to invest in a proof of concept product

Therefore:
If you believe in the product then fund it yourself first with your investment and time. (Witnessed in companies H, F, J)

5.3.3 Validate that your product sells [#63]

Problem:
Everyone needs money to keep a roof above their head and to buy food to eat.

Therefore:
Only start building something that is sure to sell. Use some of your “free time” to ask around and validate your idea. Try to get a few customers before starting the build. Even better, get them to fund your initial product. Create it together and get valuable feedback along the way. (Witnessed in company M)

5.4 Customer

5.4.1 Focus early on those people who will give you income in the long run [#4]

Problem:
Technical people tend to focus on “nice to have” things and want to grow a user community rapidly. In networked platforms, a simple approach is by offering something for free. This gives you no insight, however, that you can create income in the long run, as many users of free tools rather stop using the “open” product than starting to cover its costs.

Therefore:
Start quickly to run your business model at least in small scale. If you plan to go for a freemium license, find paying clients immediately (the fee may be reduced though, but not to 0). If you plan for advertisements as your source of income, get professional in the boat who understand the advertising market and can get in campaigns. (Witnessed in companies B, C, K, N)

5.4.2 Form deep relations with first customers to really understand their needs [#6]

Problem:
Deep understanding of the customer segment is crucial and must be developed fast. A startup must find all relevant business issues rapidly and have sparring partners that represent the latest trends in the industry.

Therefore:
Develop as deep relations with the first customers as possible. That way you can really understand the business. Understand the customers and know the culture of the people you serve. For example, analyze systematically what kinds of customers you have. (Witnessed in companies E, M, L)

5.4.3 Use planning tools that really show value provided to customer [#8]

Problem:
Some planning tools are always needed. Sometimes they give visibility to things that indirectly produce customer value, but hide the value. For example, user stories are just stories, but they do not express the customer's business goals.

Therefore:
Find tools that map the high level customer value to everything that is done. For example, how to various design alternatives aid in the success of the customer (that is, make money). (Witnessed in companies E, L)

5.4.4 Start locally grow globally [#10]

Problem:
The resources on hand are not enough to satisfy global audience

Therefore:
Target local customer segment with global growth plans in mind. The selection of processes, methods and technologies will be done with some consideration for the global growth. This will ease the product transition towards global audience when needed. (Witnessed in company F)

5.4.5 Adapt your release cycles to the culture of your users [#37]

Problem:
You need to know when and how often you should release new versions of your product and how much change people will expect / accept.

Therefore:
Know the culture of the community you work for. People in a (maybe even open source) developer community expect permanent improvement and fast reaction to new ideas. Even minor failures may be accepted if being quickly repaired. Normal consumers rather expect moderate improvements over time but more or less constant availability and (total) absence of defects. Professionals (such as companies paying relatively high fees) expect reliable services, announcement of changes in time and smooth transitions. On the other hand, they understand that improvements take some time. (Witnessed in companies C, A, K)

5.4.6 Keep customer communications simple and natural [#45]

Problem:
A startup needs good, rapid information from the customer that reacher product development efficiently. Information systems are good for that for large companies, but other types suit smaller companies better.

Therefore:
Give customers natural ways to give feedback. Encourage direct contacts by email. Be present at customers' offices often.
Further: Integrate feedback mechanisms into your product, such that users can provide immediate feedback in the context of the user interface (press a feedback button “within” the application). (Witnessed in companies M, F, L, G)

5.4.7 Help customers create a great showcase for you with support [#61]

Problem:
Your first customers can be an invaluable provider for a great, visible showcase that attracts other customer. For that, those showcases must be top-notch.

Therefore:
Give the key customers every type of support that helps them in using your product. But remember to keep focus and do not turn into a service company if that is not your focus. (Witnessed in company M)

5.5 Organization

5.5.1 Flat organization [#3]

Problem:
Communication via an intermediator takes time. Speed is lost if decisions are made only by certain person.

Therefore:
Keep the organization flat. As people are committed to a common good, trust that their decisions are sound and they can make them autonomously. The person who is facing an important decision usually has most of the contextual information. She only may need help and consulting on the bigger picture, not the decisions made for her. (Witnessed in companies G, P)

5.5.2 Consider career expectations of good people [#52]

Problem:
Specifically startups need to rely on the skills of their team members and to keep them happy at least until after some stabilization has occurred. Good people focus not only on today’s job but also on future opportunities.

Therefore:
Provide your team members with opportunities to build up skills that raise their market value. This keeps people happy and also serves as kind of insurance for them: In case of a failure of the startup they can expect better chances on the job market. This again raises the acceptance of a risky work situation. Do not hire people who have expectations that cannot be met in a startup (such as “want to lead team of 20 people”). (Witnessed in companies J, K)

5.5.3 Don't grow in personnel [#57]

Problem:
Startups are prone to follow traditions and domain customs. One of them is growing the personnel count and becoming a "proper" organisation. But there is no law that states so.

Therefore:
If a small size is good for you and there is no need to get more resources or competences, don't grow. Period. (Witnessed in M, L, P)

5.5.4 Bind key people [#67]

Problem:
Critical information is easily lost.

Therefore:
People holding key information should be shareholders/partners/founders. If they leave, you can harass them for information, because they should then be interested in the wellbeing of the firm. (Witnessed in company R)

5.6 Competence

5.6.1 Form partnerships and bonds with other startups [#5]

Problem:
You require certain type of services or expertise. E.g. localization services, a simple to use content marketing platform or integration experts.

Therefore:
Utilize partners and other startups. No point deriving from your core business–creating the product–if a partner can give you what you need easy and fast. Other startups are usually especially eager to cooperate. (Witnessed in company G)

5.6.2 Make your own strength as a "brand" [#13]

Problem:
All the startups should have some exceptional skills and knowledge (or a good product) with which it is easy to stand out in the competition. Identification of the own strength (in relation to the others) is easy. Still some effort is needed for making the own strength as a brand in the market.

Therefore:
Whatever is your strength (e.g. a competence or a product) make sure that it is experienced as a high value brand in the market. (Witnessed in company L)

5.6.3 Showing alternatives is the highest proof of expertise [#15]

Problem:
When finding a good solution for a customer, it is a result from a walk-through of different alternative paths. If an expert is not able to figure out these alternatives paths, she/he might not be an expert for driving a good solution.

Therefore:
When being able to show different alternatives for a solution, a company is showing its expertise. Identify and explore alternatives for a good solution. (Witnessed in company L)

5.6.4 In the development of customer solutions, find a unique value proposition in your way of acting [#38]

Problem:
There is a need to stand out in the competition and in relation to bigger, established companies. Besides technological solutions, the ways of working with customers is important and startups have possibilities that bigger companies may lack in finding new innovative people-centric ways of action.

Therefore:
Find a conscious way of action that separates you from competitors (for example: people-centric, flexible, simple, super fast). Develop all your competences, practices and tools for that. (Witnessed in companies E, L)

5.6.5 Follow communities [#44]

Problem:
There is an urgent need to know everything that is happening in the critical technology areas. In larger companies there may be technology monitoring tasks assigned or knowledge about new things just flows in due to volume of activity without any special actions, but small companies need to do more about it.

Therefore:
Everyone should at least follow some communities to continuously know what is happening and where the world is going to. Everyone doesn't need to be active or spend time in

community collaboration of any sort, but just follow discussions and announcements -- anything new that would help the startup to find new value. (Witnessed in company M)

5.6.6 Share ideas and get more back [#49]

Problem:
Good ideas are generated by many teams in the world. Seldom one company is breaking through in isolation without sharing its thoughts with the others.

Therefore:
Share your thoughts/ideas and you get more in return. Do not be paranoid. It is more likely that you get valuable feedback after talking to others than that they steal your idea (and are more successful than yourself). (Witnessed in companies L, N)

5.7 Team

5.7.1 Small co-located teams [#2]

Problem:
Good communication is a key requirement for surviving in an uncertain situation with limited resources.

Therefore:
Keep your team small, perhaps under 10 people. All members of the team should be within an earshot from each other, so that information will naturally flow between individuals. Avoid separate rooms, as a confined space creates information breaks. However, note that sometimes skills and stability of the team are more important than location. (Witnessed in companies J, F, H, G, P)

5.7.2 Have multi-skilled developers [#53]

Problem:
In a startup the personnel count is low, yet there are many different things to to. It is not possible to have a specialist for any task. Also, people working in a startup company value agility in software development. Additionally they have a desire to work hands-on in software development activities and get good results out from processes they are involved in. One reason for leaving behind a large company is to be more uninhibited of the bureaucracy.

Therefore:
Have multi-skilled and self-motivated developers and other personnel, with passionate attitude towards software development. For example, a UX expert should be able to program user interfaces, programmers should be able to do test automation, CEO should be able to do exploratory testing. But also be prepared to train them as needed. The willingness is often more important than existing skills. (Witnessed in companies E, M, H, L, G)

5.7.3 Keep teams stable in growth mode [#55]

Problem:
A startup is in continuous change and at the same time the changes and their effects must be minimized especially when going into growth mode. Especially team dynamics need to be handled to provide a safe environment for learning and developing processes.

Therefore:
In growth mode, keep the teams stable. Let people work with the same colleagues in the same role (mostly). (Witnessed in companies E, L)

5.7.4 Let teams self-select [#56]

Problem:
Startups need trust, real deep collaboration. Outside appointed managers can distrup personnel dynamics and reduce capability.

Therefore:
Let teams select their leads organically and self-organise team composition. (Witnessed in companies E, P)

5.7.5 Sharing competence in team [#58]

Problem:
Startups usually work with new technologies that have good usage patterns that not many people know (unlike for example how Java could be used). Only one developer is the most proficient with any usage patterns, but others need to be raised on the same level. Also, find various kinds of external experts to give sparring.

Therefore:
Share your skills and knowledge about how things should be done. Look how others do things and give advice. Use reviews for learning. (Witnessed in companies M, G, L)

5.7.6 Start with a competence focus and expand as needed [#59]

Problem:
A startup relies on the competences of its personnel. Typically a startup company is founded on specific competences (e.g. very experienced people in embedded systems). How to make sure that the competences are fit to the market needs?

Therefore:
Focus on using (and also improving) on a specific competence with a small group of people. When a competence is valued and asked by the customer, it can be expanded in the company by recruiting new people. Make sure that you can trust these people and their skills by either knowing them directly, by a recommendation of a trusted party or by letting them work for you as a trainee. Investment in a specific competence (hiring specific knowledge) can be a signal to focus on the specific market. (Witnessed in companies L, E, P, G)

5.7.7 Start with small and experienced team and expand as needed [#64]

Problem:
Startups often face a diversity of problems of different technical and business nature, in an unanticipated pace and manner. How to enable smooth product development with fewer overheads in such a dynamically changing product development environment?

Therefore:
Rather than trying to anticipate all the necessary skills, start with a small but experienced team that has efficient communication and wide scale of knowledge and expertise. (Variant of #59)

5.8 Process

5.8.1 Have different processes for different goals [#16]

Problem:
When startups develop their first processes, there is a tendency to simplicity and to having just one process. But different tasks require different practices. Requirements-based development is very much different than creative innovation. This will become essential at least in the growth mode.

Therefore:
Have different practices for different tasks, especially for requirements-based development and innovation. (Witnessed in companies E, L)

5.8.2 Tailored gates and done criteria [#29]

Problem:
All process phases that lead to something being done or something getting assessed or accepted must be selected so that they reflect the overall process, customer collaboration and business.

Therefore:
Use tailored gates and done criteria according your needs, instead of using standard steps. (Witnessed in company E)

5.8.3 Time process improvements right [#32]

Problem:
At some point, in preparation for growth, the startup needs to plan for better processes and for example workflow tools. But doing this may require the focus and effort of the best people in the small company. That is why the improvements must be planned and time carefully.

Therefore:
Do any process improvements only once you understand the needs well and you will be needing the improved systems shortly. That is how the return of investment is best possible, "process debt" is balanced with business, you can scale the solutions correctly and utilize the tools that are at that very time the best ones possible (for you!). This requires the courage of being honest of the deficiencies but not yet doing anything about them. (Witnessed in companies M, L)

5.8.4 Find the overall development approach that fits your company and its business [#39]

Problem:
There is a tendency for small companies to follow the latest hype. That may lead to approaches that are not the best for the startup's business. The result may be too reflective and light. This is critical because the first projects really must succeed.

Therefore:
Find the approach to development that is best for your business, projects and most likely gives success in next projects. Don't be afraid to be methodological. Especially focus on how to work on critical issues, such as UX. (Witnessed in companies E, M, L)

5.8.5 Tailor common agile practices for your culture and needs [#40]

Problem:
For efficiency, any practices need to match the company's natural way of working. Any textbook practices may not be best for the company as they are in generic form. A unique startup is not generic, but must find the unique practices that reflect that uniqueness.

Therefore:
Tailor common agile practices for your culture and needs. Those include any meetings and their agendas, pre-game activities, communication practices, customer collaboration, monitoring progress, experimentation, testing. (Witnessed in companies E, M, H, F, L, P, J, F, B, A)

5.8.6 Fail fast, stop and fix [#42]

Problem:
Doing the wrong things or the right things wrong can lead to big problems, especially for startups, where you don’t have the time or resources to do the wrong things.

Therefore:
Every developer and everyone else must have the freedom to do things quickly, stop immediately when things go wrong, and fix the approach or start the fixing process in the team. This is a culture thing, not a process feature. (Witnessed in companies E, P)

5.8.7 Move fast and break things [#65]

Problem:
In a startup, the resources are often limited and time is of the essence. Yet, you can't create garbage or useless things.

Therefore:
Create a culture of agile, fast development, where things are made fast and failing is completely acceptable, even preferred. (Witnessed in company M)

5.8.8 Forget Software Engineering [#69]

Problem:
Some companies do software as a part of a physical product. The software controls devices, collects, stores, and analyses data, and the like. Such software may be quite small and relatively simple and the cost to recreate it from scratch may be marginal in relation to the production cost of the physical device actually being developed and marketed.

Therefore:
Unless one of the founders happens to have a specific software engineering background, software development may be quite ad hoc and unorganized as long as it is good enough in the context of the physical device actually being marked. (Witnessed in companies I, D)

5.9 Design

5.9.1 Anything goes in product planning [#9]

Problem:
Figuring out new projects, new system concepts and new main features is critical. Every company must find a way to get the product pipeline started and flowing. But what is the best way for it?

Therefore:
There is no general best way. It all depends on the company, its culture and how people want to do things. Sometimes a visionary CEO with a backlog in her pocket is the right way, sometimes planning in the kitchen, sometimes having someone in change of new product development. (Witnessed in companies M, P)

5.9.2 To minimize problems with changes and variations, develop a very focused concept [#19]

Problem:
When startups start generalizing their product, they run into problems with variations and changes. The hassles with those must be minimized at their origin.

Therefore:
Develop a very focused and validated concept. Deep customer collaboration, customer development and design methods such as Goal Directed Design are essential tools for that. But don't let that mean anti-change -- a good concept is pro change! (Witnessed in companies E, F, H)

5.9.3 Develop only what is needed now [#27]

Problem:
Startups often think of the extensibility of their products. There are various strategies for that and one is to generalize any designs, thinking that new implementations could be easily derived from the generalisation than by refactoring a current implementation. Sometimes the latter is the best strategy.

Therefore:
For efficient extensibility, develop only what is needed now. Don't generalize designs. Don't implement for the next project. (Witnessed in companies M, E, F, L, G, J)

5.9.4 Make features easy to remove [#30]

Problem:
When the products evolve, features need to be removed, turned into modules or separate products. And this must be done fast as every developer needs to have time to turn out new designs. That requires attention to the architecture and implementation.

Therefore:
Use architecture, design and implementation techniques that make features modular and independent so that they are easy to remove, turn into modules or separate products. Keep architecture always in mind. (Witnessed in company M)

5.9.5 Use extendable product architecture [#66]

Problem:
Software architectural updates in response to adding/removing features may lead to system wide issues.

Therefore:
Employ architectural practices that enables easy extension of the design e.g. pluggable architecture where features can be added and removed as plugins. (Witnessed in companies H, L, G, Q, B)

5.10 Testing

5.10.1 Only use reliable metrics [#33]

Problem:
Every company wants to validate things with metrics, but wrong metrics can do more harm than help.

Therefore:
Only use metrics if they give reliably the information you need. If they don't use other means to get the information you need to develop the product. (Witnessed in companies M, G)

5.10.2 Bughunt [#34]

Problem:
The general quality of the product declines as everyone is keen on developing new features and honing the quality seems a dull chore. “Good enough” quality is the worst enemy of good quality.

Therefore:
Arrange bug hunt days where everyone can attend to. Make bug hunting a fun occasion. For example, collect all open bugs on a black board visible for everyone and let the team start fixing them. Anyone who loses an issue will get a small recognition of this (a tick on the table or similar). Keep the communication flowing in order to avoid code conflicts and redundant bug fixing. (Witnessed in company G)

5.10.3 Test APIs automatically, UIs manually [#50]

Problem:
Tools for unit and integration testing of programming interfaces are abundant, established, and relatively easy and cheap to use and operate. Thus such tools are used by most companies from early phases onward. Tools to test UIs are less simple and (most important) sensitive to changes in the UI. Since such changes are typical in the context of startups, they do not pay of.

Therefore:
Use the established technical tools (JUnit, Jenkins etc.) to test APIs. For UIs and system tests use manual testing. The later can be formalized in a way similar to API-testing. For instance use test scripts to be executed manually or checklists of things that should be accomplished (e.g. register new user, pay using credit card). To create these, use similar strategies as with automatic testing (create a test from each fixed bug and so on…). To time such tests, set aside a day before deployment and focus the entire team on these tests. (Witnessed in companies A, K, C)

5.11 Technology

5.11.1 Use generic, nonproprietary technologies [#23]

Problem:
Startups have limited resources for implementing products with proprietary platform technologies. Handling of those and re-implementing features would consume all of the resources.

Therefore:
Use only platform-independent technologies. For example, base user interfaces on HTML 5 instead of mobile platform's UI library and implementation language. Or use Qt to develop the app for multiple platforms. (Witnessed in companies E, H, F, L, G)

5.11.2 Create a solid platform [#24]

Problem:
When operating in a field that is continuously creating new technological opportunities, it might be hard to move fast and reap the benefits of the new opportunities before others.

Therefore:
Create a platform that supports scaling quickly. For example, if you are creating a consumer app that is meant to work with ever growing amount of consumer mobile devices, you should create a platform that is easy to extend/scale to support new devices. This might create business advantages, if you are always the quickest. (Witnessed in company G)

5.11.3 Choose scalable technologies [#25]

Problem:
While a startup must concentrate on today's problems, at some point comes the time to start scaling. It would be a mistake to need to change product technologies and tools at that point.

Therefore:
When choosing product, deployment and development techniques, favour those that scale well (but not at the expense of how good they are for _now_). (Witnessed in companies M, F)

5.11.4 Use the most efficient programming languages and platforms [#26]

Problem:
Development must be fast and efficient as there are a small number of developers. Therefore, the most efficient tools must be found that as directly as possible support the task at hand.

Therefore:
Use the most efficient programming languages and development platforms no matter what the tradition of the domain is. Be the leader, not the follower! For example, consider Clojure

instead of Java or use Ruby on Rails for its special characteristics. (Witnessed in companies E, M, L)

5.11.5 Start with familiar technologies and processes [#31]

Problem:
Learning new technologies/processes consumes time and can cause delays

Therefore:
Use technologies/processes the team members are experienced with. It will save time otherwise invested in learning the new processes or technologies. (Witnessed in company G)

6 Relationships between patterns

This chapter presents some of the most essential relationships between the patterns in visual form. The relationships were identified in a short, one hour workshop (see Figure 4) and somewhat refactored during editing. Note that some patterns are included in more than one graph and that there are more relationships than shown, but in order for the graphs to be communicative, we do not aim to show everything.

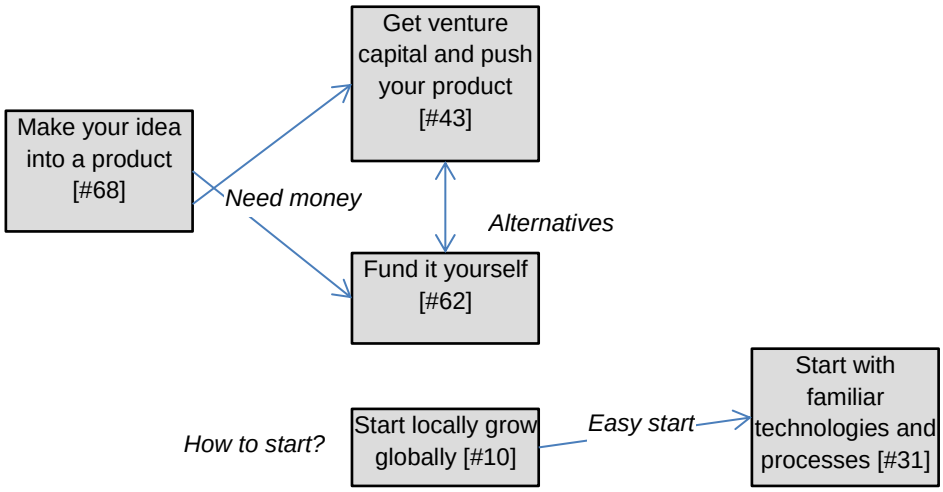


Figure 5. Starting the company

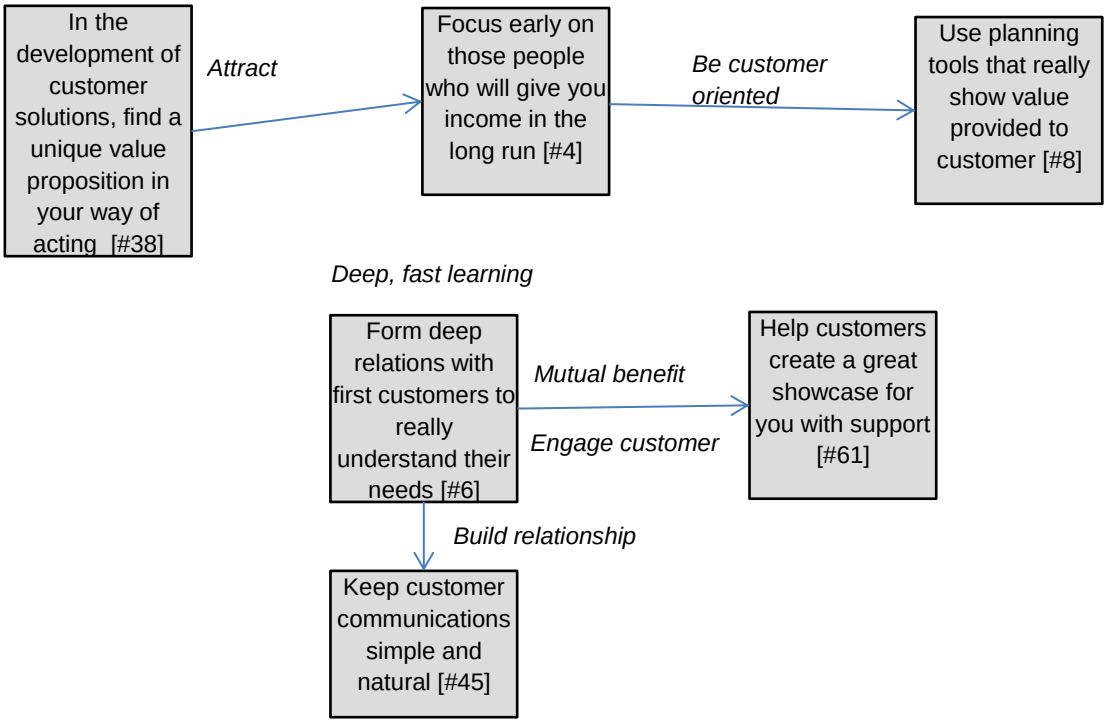


Figure 6. Customer relations

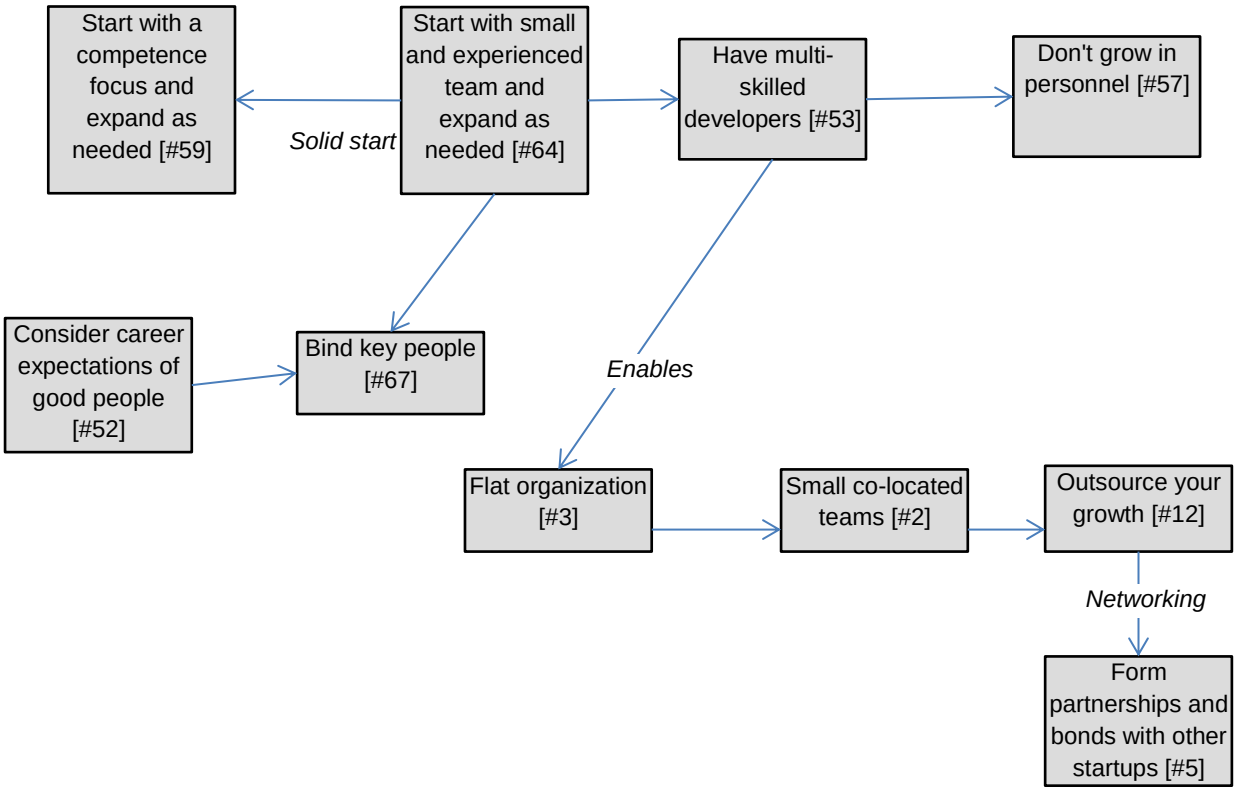


Figure 6. Customer relations

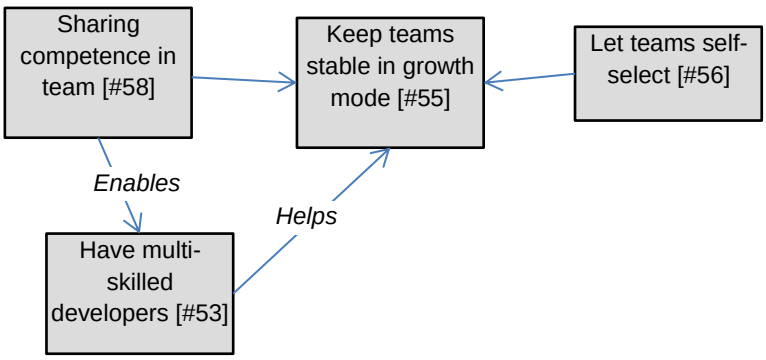


Figure 8. Stability in organization

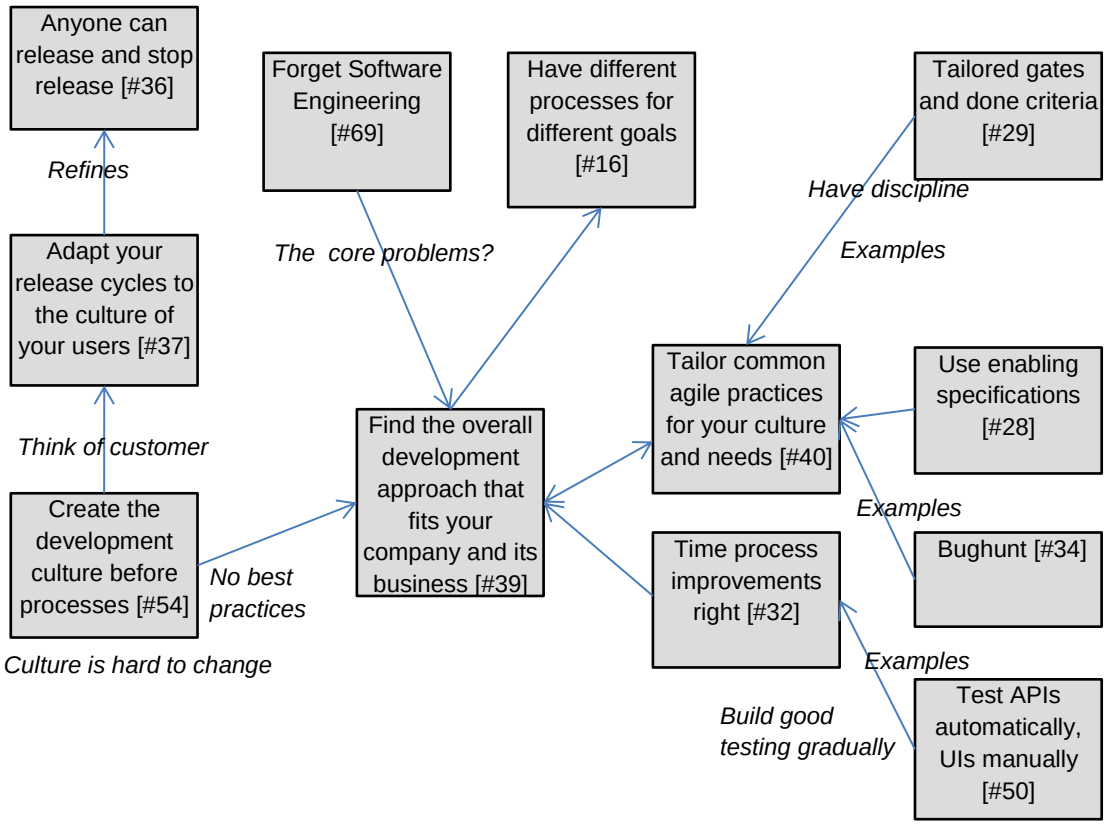


Figure 9. Finding the development style

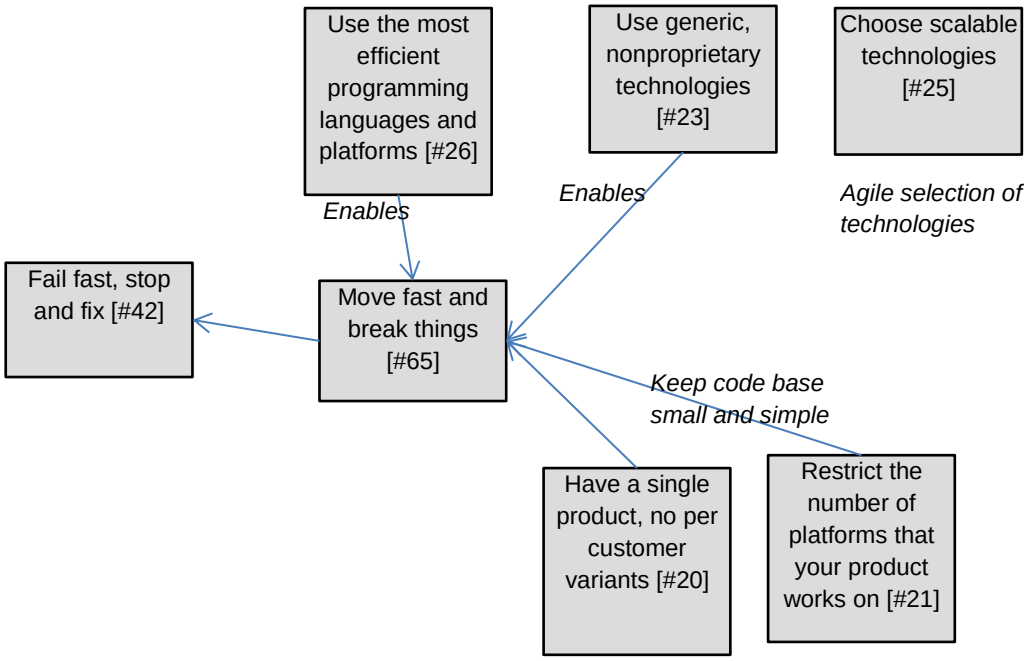


Figure 10. Being fast and flexible

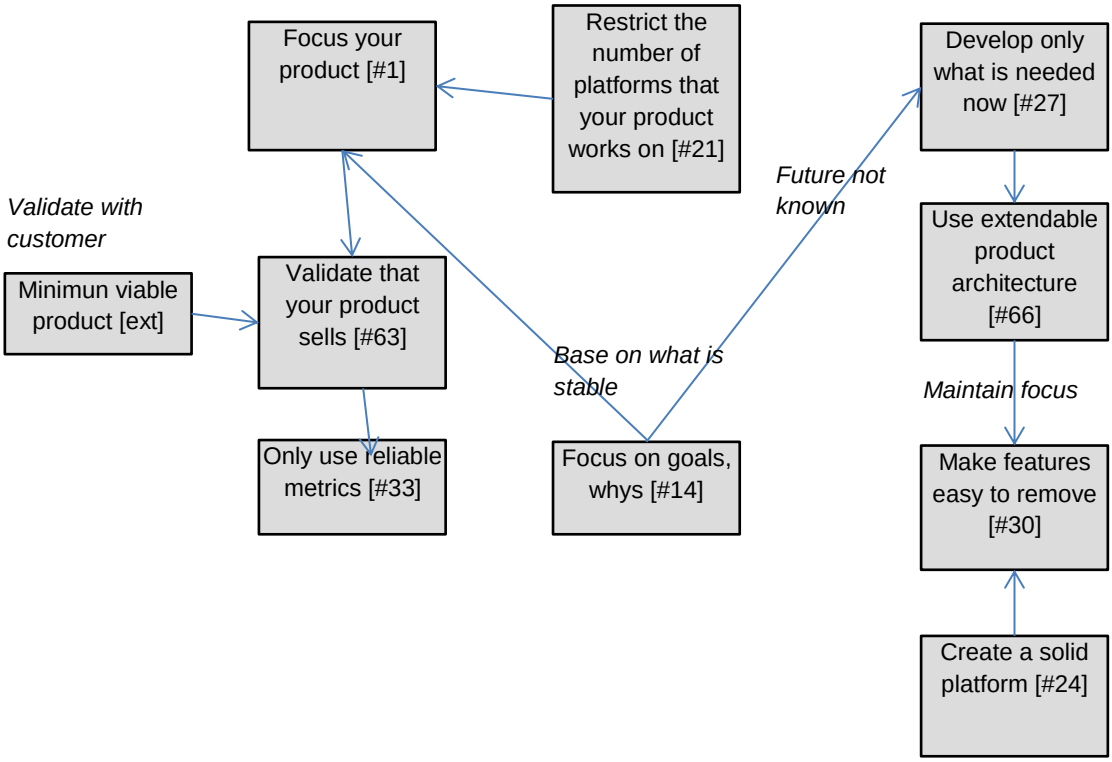


Figure 11. Focusing the product

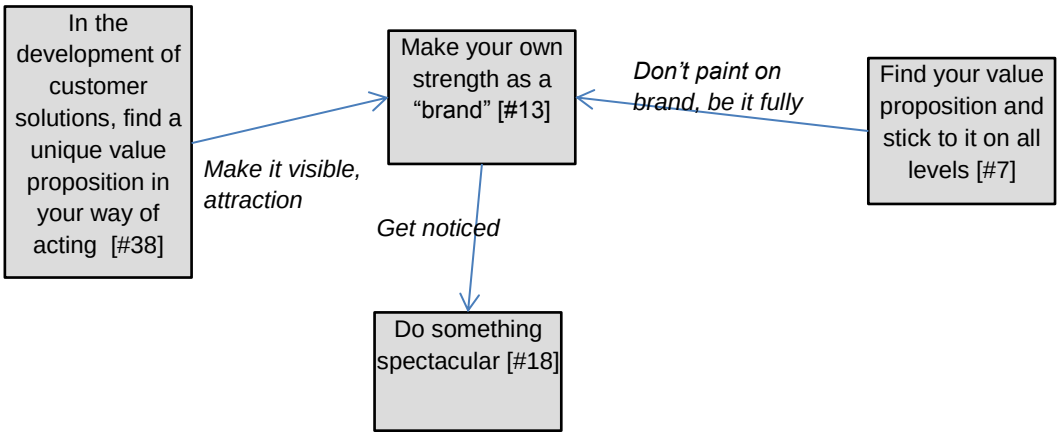


Figure 12. Standing out in competition

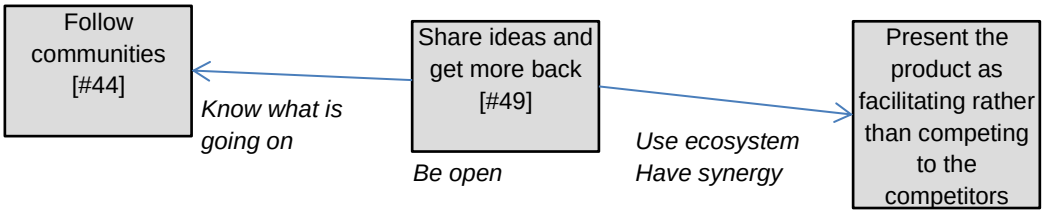


Figure 13. Being part of innovation community

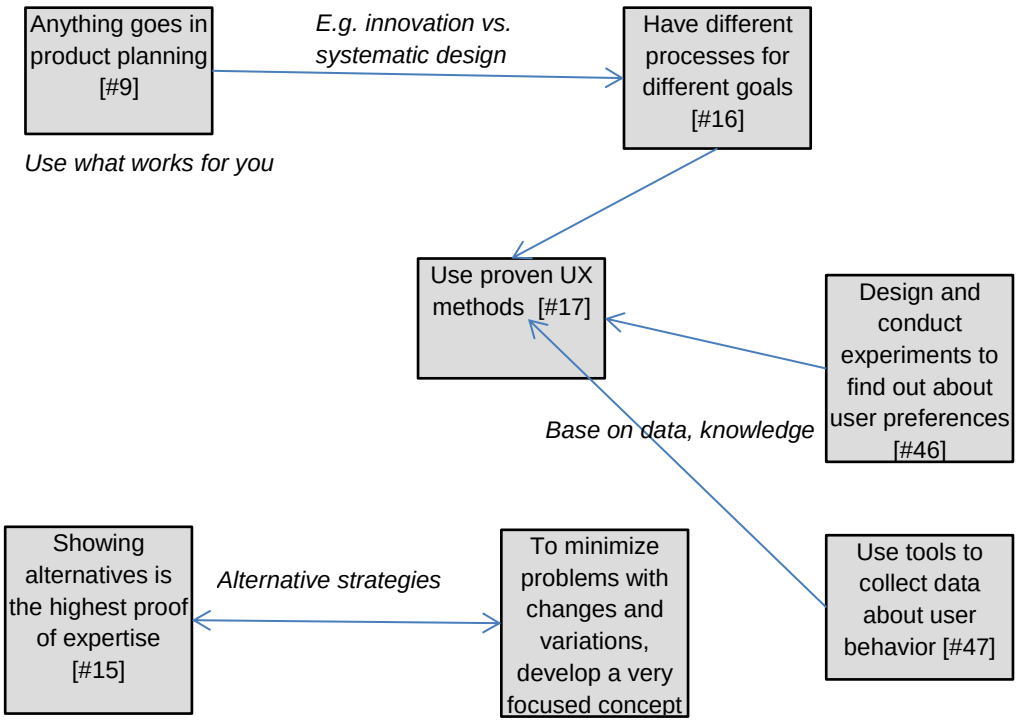


Figure 14. Product design

7 Conclusions

Software startup working practices have been empirically explored and represented as patterns in this work. Clearly, this must be seen only as a first step in creating a pattern language to be used by practitioners in software startups. Even the patterns chosen for more refined representation still need further consideration, and many of the pattern candidates in Section 5 are just rough ideas. Similarly, the relationships found between the patterns in Section 6 are only a first step in developing a proper pattern language. We see the material in this report more as a starting point for investigating organizational patterns in startup software engineering. Taking into account the growing economic significance of software startups, we see the future work based on this material very relevant.

Literature

[Ale77] Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, M. and Angel, S.: A Pattern Language: Towns, Buildings, Construction. Oxford University Press, New York, 1977.

[Cop96] Coplien, J.O.: Software Patterns. SIGS Books, New York, 1996.

[CH05] Coplien J.O., Harrison, N.B.: Organizational Patterns of Agile Software Development. Lucent Technologies, Pearson Prentice Hall 2005.

[COC08] Coleman G. and O'Connor R.V.: An investigation into software development process formation in software start-ups. Journal of Enterprise Information Management, vol. 21, no. 6, pp. 633–648, 2008.

[Cro02] Crowne M.: Why software product startups fail and what to do about it - Evolution of software product development in startup companies. In: Proc. Int. Engineering Management Conference (IEMC '02), IEEE CS 2002, 338 – 343.

[EuroPLoP] EuroPLoP – About Writers' Workshops. http://www.europlop.net/content/writers-workshops. Retrieved February 2014.

[GP12] Giardino C., Paternoster N.: Software development in startup companies. MSc thesis MSE-2012-99, Blekinge Institute of Technology, Sweden, September 2012.

[Hei06] Heitlager I., Jansen S., Helms R., Brinkkemper S.: Understanding the dynamics of product software development using the concept of co-evolution. In: Proc. Software Evolvability (SE '06), IEEE CS 2006, 16 – 22.

[Hei07] Heitlager I., Helms R., Brinkkemper S.: A Tentative Technique for the Study and Planning of Co-Evolution in Product Software Startups. In: Software Evolvability (SE '07), IEEE CS 2007, 42 – 47.

[Hig07] Highsmith J.: Agile Project Management – Creating Innovative Products. Wiley 2004.

[JBH08] Jansen S., Brinkkemper S., Hunink I.: Pragmatic and Opportunistic Reuse in Innovative Start-up Companies. Software, IEEE, pp. 42–49, 2008.

[May12] May B.: Applying Lean Startup: An Experience Report. In: Proc. Agile Conference '12, IEEE CS 2012, 141 – 147.

[Por14] Portland Repository. http://c2.com/ppr/about/portland.html.

[Rie11] Ries E.: Lean Startup - How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. Crown Business 2011.

[RW07] Richardson I., Wangenheim C.: Why Are Small Software Organizations Different? IEEE Software, pp. 18–22, 2007.

[Str06] Striebeck, M.: Ssh! We are adding a process... [agile practices]. In: Proc. Agile Conference 2006, IEEE CS, 9 pp.

[Tai10] Taipale M.: Huitale – A Story of a Finnish Lean Startup. In: P. Abrahamsson and N. Oza (Eds.): LESS 2010, LNBIP 65, 111–114.

[Tho03] Thompsen J.A.: Achieving Return on Critical Talent - A Case Study of a Software Development Organization. In: Proc. Engineering Management Conference (IEMC '03), IEEE CS 2003, 67 – 71.

[Wal09] Walsh B.: The Web Startup Success Guide. Springer 2009.

[Zet01] Zettel J., Maurer F., Münch J., Wong L.: LIPE - A Lightweight Process for E-business Startup Companies Based on Extreme Programming. F. Bomarius and S. Komi-Sirviö (Eds.): PROFES 2001, LNCS 2188, 255-270.

Appendix: Interview form

STARTUP SOFTWARE ENGINEERING INTERVIEW

Introduction

The purpose of this study is to identify and analyze the software development practices that startup companies have found useful. The hypothesis is that these practices differ at least to some extent from the practices in more established software industry. To analyze the rationale of the practices, the interview will also cover questions related to the character of the company. The summarizing results of this study will be distributed to the participating companies, and possibly later published in scientific forums, but company-specific information will remain in the research team. This study is carried out in cooperation with University of Applied Sciences and Arts Northwestern Switzerland (http://www.fhnw.ch/imvs).

1 Basic interview information

Company	
Date and time	
Interviewees & their roles in the company	
TUT interviewers & primary roles	

2 Company background

2.1 Company details

- *Give a brief history of the company (how did you start) covering*
  - *Company age*
  - *Vision*
  - *Business area*
  - *Business model (current and desired)*
  - *If current and desired differ, ask later about software engineering practices that help the company reach the desired goal*
  - *What is your path to reach your desired future goal?*
  - *Are you planning to reach new markets, new kind of customers? What and how?*
  - *Current phase (startup, stabilization, growth) The real startup phase means that you are just forming the company around the founder team, in stabilization phase you create infrastructure, processes and have a couple of customers, in growth phase you are growing.*

2.2 Staff

- *Number of employees and their roles (possibly primary roles and secondary roles)*
- *How experienced are your employees (based on employees' professional ages?)*
- *Do employees have background in software engineering?*

2.3 Customers

- *Number of customers*
- *What is the customer segment? Who do you create products for?*
- *How are you unique & stand out in the competition? What is your value proposition?*
- *Do you have business partners?*

2.4 Product

- *What is the type of your software product (web, some, mobile, traditional)*
- *If you can draw a picture of you product ecosystem and it clarifies something, do so.*
- *What is the domain of your product (enterprise, entertainment,...)*

2.5 Business challenges

- *What challenges does your business present to software development and how are you approaching them?*
  - *Short term (what is essential to survive the coming month(s)) and long term (what is essential for long term strategy, growth, etc.).*
  - *Possible challenge areas: Financial challenges (funding, budget), personnel challenges, user-related challenges, competition challenges, technical challenges*
- *What means, especially s/w development are you using to address these challenges?*

3 Software development

- *What aspects of software engineering you find the most important for your company:*

A Goal setting	B Design	C Development	D Quality	E Deployment	F Version and code management

- *We will go through the question set in the order implied by the markings of the company, first questions related to the topic areas that the company considers important. And the questions in the General topic we'll discuss last with all companies.*

3.1 Goal setting

- *How did you come up with your product concept and scope? (by whom, where, dynamics of creation, key persons involved...)*
- *How did you test / assess the preliminary ideas? (proto, demo, MVP, focus group, key customers…)?*
- *How do you respond to new feature requests?*
- *How do you keep up with the changing needs of the customers?*
- *What do you do if you see that some feature is not used or really popular?*
- *How do you prioritize customer's wants and needs?*
- *How do you collect feedback from the customers? What kind of feedback (are people using feature, how do you track this).*
- *What are you doing in software engineering terms to reach your desired business model?*

3.2 Design

- *What is the platform for the product? Why did you choose it?*
- *(for example, Supercell focused on iOS as their primary platform because it has the biggest market)*
- *Do you have multiple platforms for your product? How do you handle that?*
- *What and why did you choose the key technologies for the product?*
- *How do you keep the product simple enough to ensure short time to market? What is in the product that enables the shorter time to market.*
- *How do you keep the product flexible enough to allow new features?*
- *Describe your practices for designing desirable products. How do you know the users' satisfaction and experiences in relation to your product?*
- *What kinds of solution concepts (algorithms, architectures, acquired libraries, services etc.) you use in the product to satisfy the goals?*

3.3 Development

- *Can you describe your development process? Why did you choose that process? What are the characteristics of the process that especially serve the company goals?*
- *Are you following some standard process (e.g. Scrum), and if so, how have you adapted that and why?*
- *How adaptable is your way of developing products to changing situations (e.g. rapid growth of market)? What makes it adaptable?*
- *How do you manage to give your developers a possibility to focus to product development and not to please individual customers?*
- *How do you maintain key knowledge and competence about the product and processes in the company? (e.g. if a key person leaves)*

3.4 Quality

- *What methods for measuring product quality are you using?*
- *How do you test your product? Do you use automatic testing?*
- *How do you prevent technical debt? How do you handle any accumulated technical debt?*
- *How do you maintain the product quality in the long run?*

3.5 Deployment

- *What triggers the release of your product? Do you have continuous delivery?*
- *How do you deploy new versions to customers?*
- *How do you minimize downtime when deploying?*

3.6 Version & code management

- *How do you manage different versions of the product?*
- *Do you remove code of features that are not any more available for users?*
- *Do you have practices to organize code so that it supports continuous improvement, agility and rapid growth?*
- *How do you minimize "legacy code infestation"? (= keep the codebase fresh and free of code that no-one wants to touch)*

3.7 General

- *Did you try some process practices that did not work?*
- *Are you using some other practices characteristic to your company, not so far discussed?*
- *How do you manage growth of the team? (Roles, tasks, competence transfer, team dynamic…)*

3.8 Other issues

- *Anything else that you think is essential?*

Tampereen teknillinen yliopisto
PL 527
33101 Tampere

Tampere University of Technology
P.O.B. 527
FIN-33101 Tampere, Finland

ISBN 978-952-15-3251-1 (printed)
ISBN 978-952-15-3252-8 (PDF)
ISSN 2323-9174