Software Startup Practices – Software Development in Startups through the Lens of the Essence Theory of Software Engineering

Kai-Kristian Kemell[1][0000-0002-0225-4560] Ville Ravaska[1],
Anh Nguyen-Duc[2][0000-0002-7063-9200] and Pekka Abrahamsson[1][0000-0002-4360-2226]

[1] University of Jyväskylä, Jyväskylä 40014, Finland
[2] University of Southeast Norway, Norway
kai-kristian.o.kemell@jyu.fi

Abstract. Software startups continue to be important drivers of economy globally. As the initial investment required to found a new software company becomes smaller and smaller resulting from technological advances such as cloud technology, increasing numbers of new software startups are born. Startups are considered to differ from other types of software organizations in various ways, including software development. In this paper, we study software development in startups from the point of view of practices to better understand how startups develop software. Using extant literature and case study data, we devise a list of practices which we categorize using the Essence Theory of Software Engineering (Essence). Based on the data, we propose a list of common practices utilized by software startups. Additionally, we propose potential changes to Essence to make it better suited for the software startup context.

Keywords: Software Startup, Essence Theory of Software Engineering, Software Development, Software Development Practice, Case Study.

1 Introduction

Software startups continue to be important drivers of economy globally. As the initial investment required to found a new software company becomes smaller and smaller as a result of technological progress, more and more startups are founded. While most startups fail [4], just like most new companies [13], some go on to become mature, established software organizations, or even multinational technology giants.

Typically, the main argument for studying software startups is that they differ from mature software organizations in various ways, thus making the findings of many existing studies not directly applicable to them. This is a result of there still being no accurate definition for what a startup is [21][23]. Various characteristics such as time pressure or resource scarcity are attributed to startups to differentiate them from mature companies [21], but academically drawing an exact line has been a challenge in the area [13]. The way software startups develop software has been one area of study.

For example, Paternoster et al. [21] conducted a more general, large-scale study aiming to understand how software startups develop software. They noted that software startups operate mostly using various Agile practices or ad hoc methods. Specific facets of software development (SWD) in software startups, such as prototyping [19] have also been studied. However, studies focusing on Software Engineering (SE) practices in software startups are still scarce, and studies into SWD in software startups in general are still needed [23]. Some high-profile startup practices such as the Five Whys are commonly discussed in e.g. startup education, but systematic studies are lacking.

Thus, to better understand how software startups develop software, we study practices in this paper. Specifically, we seek to understand what practices are commonly used by software startups. In addition, we approach this topic through the lens of the Essence Theory of Software Engineering and seek to understand how this theory fits into the context of software startups. To this end, we study how the seven alphas of the theory (section 2.3) fit the context of software startups, and whether other alphas would be needed to make the theory better suited for this context.

2 Background – Software Startups, Software Development Practices, and the Essence Theory of Software Engineering

This section is split into three subsections. First, we discuss SWD in software startups. Then, we define SWD practices in this context. Finally, we discuss Essence.

2.1 Software Development in Software Startups

Typically, software startups do not strictly follow any formal software development method [21]. Instead, they combine practices from different methods that suit their needs at the moment or simply use ad hoc practices [18].

As the aim of this study is to uncover software development practices universal to (most) software startups, a notable paper is that of Dande et al. [6]. Dande et al. [6] studied software startups in Finland and Switzerland and devised a list of 63 practices commonly utilized by software startups. However, these practices are not solely software development ones but also include practices related to customers and business. Kamulegeya et al. [11] studied these practices and reported that they seemed to apply in the Ugandan startup context as well, further validating this list of practices. They do add, however, that culture and location might influence commonly used practices.

Other studies focusing on practices have not aimed to create such extensive lists of practices but have nonetheless studied software startup practices in different contexts. Klotins, Unterkalmsteiner, and Gorschek [15], for example, created a framework for categorizing software startup practices that differs from the one proposed by Dande et al. [6]. Giardino et al. [9] propose the Greenfield Startup Model to explain software development in early-stage software startups. In the process, they uncovered various practices that supplement and confirm the findings of Dande et al. [6]. Paternoster et al. [21] in their study on how software startups develop software discuss having found 213

practices, which, however, were not listed in their paper. Nonetheless, their findings to lend support to those of Dande et al. [6].

2.2 Software Development Practice as a Construct

Jacobson et al. [10] suggest that a set of practices is what forms a method in the context of SE. Methods, according them, describe ways-of-working, i.e. how work should be carried out. A way-of-working exists in an organization even if a formal SE method is not utilized [10]. A practice, then, describes a more atomic unit of work.

Historically in academic literature, and particularly in Information Systems, the construct *technique* has been used for the same purpose in the context of method engineering [22]. Tolvanen [22] defines a technique to be a set of steps and rules that define how a representation of information system is derived and handled using conceptual structure and related notation. A tool, in this context, refers to a computer-based application supporting the use of a technique.

2.3 The Essence Theory of Software Engineering

The Essence Theory of Software Engineering [10] provides a way of describing methods and practices. It consists of a notational language and a so-called kernel, which includes building blocks that can be used as a basis for constructing methods. The kernel, its authors argue [10], contains basic elements that are universal in any SE project.

The Essence kernel contains three types of objects: alphas (i.e. things to work with), activities (i.e. things to do), and competencies (skills required to carry out the work). In this study, we focus on the alphas in the context of software startups. The seven Essence alphas are as follows: (1) Stakeholders, (2) Opportunity, (3) Requirements, (4) Software System, (5) Team, (6) Way of Working, and (7) Work. These alphas are split into three areas of concern. The first two belong in the customer area of concern, numbers three and four in the solution area of concern, and the last three in the endeavor area of concern. Furthermore, each alpha has alpha states used to track progress on the alpha. [10]

The authors of Essence posit [10] that these are the essential elements that are present in every SE project. Every project, then, has its own unique context, which most likely contains more things to work with, but those are not universal to every project. In order to reap the most benefits out of Essence, its users would then extend this basic kernel with the Essence language to include these unique features of their particular project or company to describe their method(s) with it.

In this paper, the role of Essence is two-fold. First, it serves as a framework for analyzing our data. We utilize the alphas to sort the software startup practices we discover into categories. Secondly, in the process of doing so, we study whether all the uncovered practices fit into these seven alphas. I.e., do the alphas also present all the essential elements of software development in software startups?

We chose to utilize Essence as the framework for this study for two reasons. First, Essence is an OMG standard. Standards can shape the industry and should be studied. In this case, we are particularly interested in seeing whether Essence suits startups as well. Secondly, Essence provides one framework for categorizing work in SE projects

through its kernel and alphas. In studying practices, we considered it important that we have a framework for categorizing them in some fashion.

3 Study Design

The goal of this study is outlined at the end of the introduction. We approached this topic using a qualitative multiple case study approach. Aside from this empirical data, we utilized the list of 63 startup practices presented by Dande et al. [6].

3.1 Data Collection

The empirical data for this study was collected by means of a multiple case study (n=13) (Table 1). The interviews were conducted F2F. The audio was recorded, and the recordings were transcribed for analysis. All the respondents were CEOs or founders, as we wished to interview respondents with extensive knowledge of the case startups.

Table 1. Cases.

Case	Employees	Company Domain	Respondents	Age (in years, at the time of interview)
1	6	Software/ Hardware	1	<1
2	5	Software	3	1-3
3	3	Software / Hardware	2	<1
4	5	Software	1	1-3
5	7	Software / Consulting	1	<1
6	3	Software / Hardware	1	1-3
7	8	Software	1	>3
8	12	Software	1	>3
9	6	Software	1	1-3
10	5	Software	1	>3
11	85	Software / Hardware	1	1-3
12	5	Software / Hardware	1	>3
13	6	Software	1	>3

We utilized a qualitative, thematic interview approach. We chose a thematic approach because most software startups develop software ad hoc [18][21]. Data were then collected with one of two interview instruments depending on how technical the respondent(s) were. With technical respondents, we utilized an interview instrument (found on Figshare[1]) more focused on the technical aspects of software development (interviews 6 to 13 in Table 1). With less technical respondents and in group interviews, we utilized an interview instrument built around the Essence alphas (same Figshare link below).

In utilizing two interview instruments, we wanted to gain a deeper understanding of the practices used through triangulation in terms of data collection methods, as suggested by Langley [16] in the context of process data. Using different types of data can

[1] https://doi.org/10.6084/m9.figshare.13017227.v1

provide a more comprehensive understanding of the phenomenon. In this case, we felt that focusing solely on the technical aspects might omit some less technical practices.

3.2 Data Analysis

The analysis of the empirical material in this paper was conducted following the thematic synthesis guidelines of Cruzes and Dyba [5]. The material was first transcribed for analysis. The material was then read thoroughly for an initial overview of the data. After this, the coding process was started, and each interview was coded. These codes were then arranged into themes. The coding process was done inductively, with codes and themes arising from the data (as opposed to e.g. using Essence as the framework at this stage). E.g., codes included such codes 'team', 'funding', and 'prototype'. Using this approach, we analyzed the data to find practices, either ones already discussed by Dande et al. [6] or novel ones, with the novel ones made into a list.

Practices that were discussed by two or more of the case startups were considered prevalent enough to be included into the list of practices. Once the empirical data had been analyzed and new practices had been formulated, we took the list of 63 software startup practices of Dande et al. [6] and these new practices and inserted them into the framework of the Essence Theory of Software Engineering [10] and its alphas. I.e., we categorized each practice, if possible, under one of these alphas (see section 5.2 for critical discussion about this approach). The categorized practices were then reviewed by three other authors to form a consensus.

4 Results

This section is divided into 9 subsections. In the first one, we present the new practices we uncovered through the case study. In the next seven, we go over the results in relation to each Essence alpha, discussing the practices found in each category. In the ninth and final one, we discuss practices that did not fit under any of these alphas.

Given the space limitations of this paper, the clarifying descriptions for the 63 practices of Dande et al. [6] have not been included in the tables in this section. Such descriptions have, on the other hand, been added for any novel practices proposed by us. Each practice has an identified (Pn), where practices P64 and up are practices based on the empirical data and practices P63 and below are from Dande et al. [6].

4.1 New Practices

Based on the data, we propose 13 new practices (Table 2) that were not present in the list of Dande et al. [6]. These practices were mentioned by at least two case startups. Other new practices were also uncovered but discussed by only one case startup. These practices were not considered common based on this set of data.

Table 2. New practices based on our data.

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

4.2 Opportunity

The opportunity alpha is related to understanding the needs the system is to fulfill and is within the customer area of concern. Practices for this alpha are presented in Table 3 below. No new practices for this category were found in the data.

Table 3. Practices for the Opportunity alpha.

ID	Practice	Cases Supporting
P1	Focus your product	1,2,6,7,8,9,11,12,13
P2	Find your value proposition and stick to it on all levels	9,13
P4	Focus on goals, whys	9
P18	Validate that your product sells	1,2,4,5,7,8,11
P20	Form deep relations with the first customers to really understand their needs	1,6,9,11,13
P33	In the development of customer solutions, find a unique value proposition in your way of acting	1,2,3,5,6,8,9
P34	Follow communities	1,2

The case startups were highly focused on understanding their customers and fulfilling the needs of the customer (segments). This is in line with the idea of software startups

being product-oriented and customer-focused. On the other hand, the lack of support for P4 makes it seem that these startups were more focused on fulfilling the needs they had uncovered rather than understanding why these needs were important.

Focusing on the system and the needs it was intended to fulfill was considered important from the point of view of competition as well. Focusing on one's unique value proposition is conventionally considered an important strategy for differentiating from one's competitors.

4.3 Stakeholders

Four practices were categorized under the stakeholder alpha (Table 4), which is another alpha in the customer area of concern in Essence. For startups, most notable stakeholders are typically investors and customers or users. In addition, nearly half of the case startups discussed the importance of their advisors as stakeholders (P68).

Table 4. Practices for the Stakeholders alpha.

ID	Practice	Cases Supporting
P24	Keep customer communications simple and natural	6
P32	Showing alternatives is the highest proof of expertise	-
P35	Share ideas and get more back	1,2
P68	Get advisors	1,4,5,6,8,9

Especially early-stage startups tend to rely on advisors. For example, startup ecosystems tend to foster advisor relationships in various ways. Startups working in incubators are likely to receive guidance from various experts. Advisors can provide startups with capabilities they are lacking and help them expand their contact networks.

The practice of sharing ideas to hone them and to get feedback was also discussed by some case startups. While in some cases companies may be reluctant to share their ideas in fears of having them stolen, none of the case startups indicated this type of concerns. To this end, advisors can also provide feedback if a startup is afraid of revealing their ideas to potential investors due to such concerns.

4.4 Requirements

Requirements help provide scope for the work being done on the system. Four new practices were uncovered in this category and most existing practices in this category were well-supported by the cases (Table 5).

However, P3 was in conflict of what some of the case startups stated. P3 posits that a startup should present its product as facilitating rather than competing. While this is one valid approach, startups do also seek to compete in some cases.

The requirements alpha, in the data, was closely related to the stakeholders alpha: uncovering customer needs was the main focus in requirements (P10). In the case startups, prototypes were typically used to do carry out validation (P67, P74). While a startup should be open to new features and needs (P51), they should be prioritized (P76) to create a clear core product (P52, P53).

Table 5. Practices for the Requirements alpha.

ID	Practice	Cases supporting	Cases conflicting
P3	Present the product as facilitating rather than competing to the competitors	-	1,2,6
P5	Use proven UX methods	12	-
P10	Design and conduct experiments to find out about user preferences	1,2,4,6,9,12,13	-
P21	Use planning tools that really show value provided to customers	2	-
P51	Anything goes in product planning	1,2,11	-
P52	To minimize problems with changes and variations develop a very focused concept	1,2,3,4,5,6,7,12,13	-
P53	Develop only what is needed now	1,2,3,12	-
P66	Create an MVP in the beginning	1,2,4,13	-
P67	Test features with customers	1,3,4,5,6,7,8,9,11	-
P74	Create prototype	1,2,3,4,5,6,9,12	-
P76	Prioritize features	1,2,3,9,11	-

4.5 Software System

The software system alpha is focused on the product itself, i.e. the system; software or hardware. The software system alpha is in the solution area of concern of the Essence kernel. Some of the previously proposed practices were largely prevalent in the cases while some received little support from our data. More technical practices (P23, P54, P57) would have required a more technical focus from the interviews. No new practices were proposed for this category. The practices for this category are in Table 6.

Table 6. Practices for the Software System alpha.

ID	Practice	Cases supporting	Cases conflicting
P7	Have a single product, no per customer variants	1,2,3,5,7,8,11,12	6,13
P8	Restrict the number of platforms that your product works on	1,2,3,4,7,12	-
P14	Anyone can release and stop release	2	-
P23	Adapt your release cycles to the culture of your users	-	-
P54	Make features easy to remove	-	-
P55	Use extendable product architecture	1,2,3,9,11	-
P57	Bughunt	-	-
P58	Test APIs automatically, UIs manually	2,13	-
P59	Use generic, non-proprietary technologies	2,7	-
P60	Create a solid platform	3,8,9,11	-

Out of the practices of this category, only P7 had some conflicts in the data. This practice is largely B2C focused, whereas a B2B startup might understandably focus on tailoring its system especially for larger customers. However, it is perhaps worth aiming for a modular product where such manual tailoring is not needed.

Overall, these practices further underlined that startups should have a clear focus in their development. For example, they should focus on a limited number of platforms, possibly only one initially (P8). Additionally, startups are conventionally seen as agile and their systems as prone to changes based on feedback. Indeed, these practices support the idea that the system should be developed with modifications in mind (P60). Features should be easily added (P55) or removed (P54) when necessary.

4.6 Work

Work in the context of Essence refers to the work tasks required to produce the system. It is under the endeavor area of concern in the Essence kernel. For software startups, this also involves business model development. How the work is carried out from the point of view of e.g. methods, belongs into the way of working category, on the other hand. Few existing practices were considered to belong into this category and no new practices for this category were found (Table 7).

Table 7. Practices for the Work alpha.

ID	Practice	Cases supporting
P44	Tailored gates and done criteria	8
P48	Fail fast, stop and fix	1
P62	Use the most efficient programming languages and platforms	2,3,7

While P48 is arguably closely related to prototyping and validation activities which were extensively discussed by the respondents, it was seldom discussed directly. On the other hand, P62 was discussed in relation to system architecture. Efficiency in this case was considered subjectively: the developers focused on languages and platforms they had prior experience with and could thus start working the fastest with.

4.7 Team

The team comprises the individuals working on the startup, the founders or owners and the employees or unpaid ones. It is under the endeavor area of concern in the Essence kernel. The team sizes for the case startups are in Table 1 in Section 3. One new practice (P64) was added into this category based on the data (Table 8).

The most mentioned practices were P41 and P42. The initial team is important as it needs to have the required competencies (P41). To this end, an experienced team may be required (P42). Some of the cases conflicted with P42, although not because the teams did not want an experienced team but simply because they could not find one. However, this did not mean that the startups did not want and experienced team. Rather, they simply did not have one due to being founded by a group of students with little prior experience.

If the team is lacking competencies and expanding the team is not possible or feasible in a given situation, the existing team members may be have to learn new skills instead (P64). This also ties to P37, as the small team sizes often result in a single

employee having to take on various different tasks. A developer is often involved in business decisions as well, especially in early-stage startups.

Flat organization structures (P26) are associated with startups and this was also the case in our data. Involving employees in decision-making may also serve to better bind them (P29). With a small, focused team, staff turnover can be damaging (P38).

Table 8. Practices for the Team alpha.

ID	Practice	Cases supporting	Cases conflicting
P26	Flat organization	1,2,3,5,9	-
P27	Consider career expectations of good people	4,9	-
P28	Don't grow in personnel	1,2,3,12	-
P29	Bind key people	2,3,6,7	-
P36	Small co-located teams	1,2,3,4,5,6	12
P37	Have multi-skilled developers	1,2,3,12	-
P38	Keep teams stable in growth mode	1,2,3,4,6,7,13	9
P40	Sharing competence in team	4,5	-
P41	Start with competence focus and expand as needed	1,2,3,4,6,8,9,13	-
P42	Start with small experienced team and expand as needed	1,2,3,4,7,8,12,13	1,2,3
P64	Study skills and topics that support your startup	1,2,3,4,8,9	-

4.8 Way of Working

Way of Working refers to how the work is carried out, including practices, tools, processes, and methods [10]. It is under the endeavor area of concern in the Essence kernel. Most previously proposed practices were supported by our data in this category. Four new practices were proposed for this category (Table 9).

Most case startups discussed having taken some existing agile practices and tailoring them rather than using them by the book (P47). While this ties to P72 in that frequent team meetings are common in agile development, it gained enough emphasis to be its own separate practice. On the other hand, the use of by-the-book methods (P46) was not discussed by any of the startups, with the startups using various mixed practices.
Communication in general is an important part of agile development, and arguably development in general. The case startups frequently discussed the importance of tools in facilitating communication (P75). While shared physical workspaces can reduce the need for tools, their importance is highlighted when working remotely. An early-stage startup may not have a physical workspace at all, or its members may have erratic work hours due to having a day job, resulting in communication tools becoming important.

Self-organizing teams are recommended in agile development and this is also arguably common for startups (P39, P73).

Table 9. Practices for the Way of Working alpha.

ID	Practice	Cases supporting	Cases conflicting
P9	Use enabling specifications	1,2,3	-
P15	Create the development culture before processes	1,8,11	-
P39	Let teams self-select	1,2,3,5,8	-
P43	Have different processes for different goals	-	-
P45	Time process improvements right	3	-
P46	Find the overall development approach that fits your company and its business	-	-
P47	Tailor common agile practices for your culture and needs	1,2,3,4,6,7,8,13	-
P49	Move fast and break things	4,7	-
P50	Forget Software Engineering	1	-
P61	Choose scalable technologies	2,3,9,11	-
P63	Start with familiar technologies and processes	1,2,3,7	-
P70	Test different tools	1,3	-
P72	Have frequent meetings with the whole team	1,2,3,4,5,8,12	-
P73	Don't have strict roles	1,2,3	9
P75	Use efficient communication tools	2,3,5	-

4.9 Other Practices Unsuited for Existing Essence Alphas

Not all of the practices we propose, or the ones proposed by Dande et al. [6], fit under any of the existing Essence alphas. These were practices related to the business aspect of software startups, such as marketing, business model development, or funding. Whereas Essence focuses on SE in mature software organizations, the business aspect in software startups is closely intertwined with software development. For example, the needs of the customers or the customers in general, may not be clear to a software startup, which results in the requirements evolving over time.

Practices P6, P11, P25, P31, and P71 concern marketing activities. For example, P25 is about getting a few initial customers who are particularly interested in the system and who can then be used as reference customers in marketing, or who themselves can market the product. P6 and P31 are more general marketing practices. These types of activities are difficult to incorporate into any existing Essence alpha. While marketing is a customer related activity and thus could be linked to stakeholders, the existing stakeholder alpha focuses on clearly identified and involved stakeholders such as the organization commissioning a project, as opposed to obtaining new customers (stakeholders).

P16 and P17 are related to funding. Funding or simply available cash to burn is something that is constantly tracked in a startup, much like the alphas are tracked in Essence. No existing alpha supports funding with clear emphasis. Some of the alpha states of the Work alpha include mentions of securing sufficient funding, but this process is seldom so straightforward in a startup.

The remaining practices in this category are related to overall business model development and business planning. For example, P13 suggests that outsourcing some part of the business can help the startup focus on the core product, and P22 suggests a strat-

egy for rapid and high growth. P30, on the other hand, could be filed under the Stakeholders alpha, but doing so might not place sufficient emphasis on the strategic importance of such decisions from a business point of view.

As we do not formally develop new alphas in this paper, we leave the proposals related to these observations for the following discussion section.

Table 10. Practices not applicable to any existing Essence alpha.

ID	Practice	Case supporting	Case conflicting
P6	Do something spectacular	-	-
P11	Use tools to collect data about user behavior	1,2,7	-
P12	Make your idea into a product	1,2,3,4,5,6,7,8,12,13	11
P13	Outsource your growth	5,9,11,12,13	3
P16	Get venture capital and push your product	1,2,4,5,8,9	3
P17	Fund it yourself	1,2,3,7,9	-
P19	Focus early on those people who will give you income in the long run	5,6,7,8,11,13	-
P22	Start locally grow globally	1,2,3,6,7,8,9,13	-
P25	Help customers create a great showcase for you with support	1,6,8,9	-
P30	Form partnerships and bonds with other startups	1,3,4,5,13	-
P31	Make your own strength as a “brand”	8	-
P56	Only use reliable metrics	5,6,7	-
P65	Attend startup events	1,2,3,4,8	-
P69	Use efficient tools to plan your business model	1,2,3	-
P71	Conduct market research	1,2,6,12	-

5 Discussion

The primary contributions of this study are (1) this list of practices 76 and its implications we discuss here, and (2) the implications these practices have for utilizing Essence in the startup context. First, In terms of the practices and the data overall, our findings seem to support existing literature. Paternoster et al. [21] argued that startups develop software using various agile practices or ad hoc. The case startups of this study did discuss the utilization of methods either, only occasionally mentioning singular practices that could be seen as Agile. Many of the practices, such as focusing on a set of functionalities or utilizing MVPs, are also discussed in the Greenfield Startup Model of Giardino et al. [9].

It is common for larger software organizations, too, to take a method such as SCRUM and then omit some practices to create yet another "scrumbut,” with quality practices often the first ones to go [8]. Startups, on the other hand, seem to seldom even use tailored methods, pointing to an even higher degree of unsystematic approaches to SE – based on both our data and existing studies (e.g. [18][21]).

In terms of how startups differ from mature organizations, aside from the aforementioned use of ad hoc methods and singular agile practices, technical debt is one element

typically associated with startups [1][9]. Some of the practices were ones that would arguably generate technical debt (e.g.” move fast, break things”), but the case startups did not explicitly discuss technical debt as an issue.

The list of practices in this paper presents a closer look at the way software startups develop software. These existing studies have focused on method use and specific issues faced by startups such as technical debt accumulation, or MVPs. By better understanding what practices startups use we can further our understanding of how they differ from larger software organizations. This is arguably important as it possible that one factor contributing to the lack of method use in startups may be that they feel that existing methods are not well-suited for the startup context. The practices listed in this paper support existing literature. For example, P66 posits that an MVP should built early on, which is in line with Klotins et al. [14] who argue that one common issue for software startups is taking too long with an initial version of the product.

The other contribution of this paper is related to Essence, which we have used as a theoretical framework for categorizing the practices in this paper. Essence is intended to be used in any SE endeavor. Its so-called kernel, its authors argue [10], contains the elements present in every SE endeavor. This kernel acts as a set of building blocks that can then be extended using the Essence language to describe methods.

In this paper, we looked at Essence from the point of view of software startups. Based on our data and extant literature (e.g. [14, 15]), the business aspect is deeply intertwined with software development in the startup context. In fact, Klotins et al. [14] argue that software startups largely fail due to business issues that originate from SE processes. This supports the idea that SE and business aspects are difficult to separate in software startups. If the goal of Essence is to contain the elements present in every SE endeavor, for the startup context this would thus seem to include business elements.

For example, a conventional software project that is commissioned has clear requirements which have been agreed upon with the customer(s). On the other hand, software startups spend significant effort trying to ascertain whether their idea addresses a real need of a real customer (segment) at all. These idea or business validation activities to hand-in-hand with development activities. Moreover, whereas a developer in a large organization simply develops, in startups roles are seldom so clear-cut, especially early on. In an early-stage startup, a developer may be involved in business activities as well.

Some of the practices in this paper, namely the business-related ones, were not well-suited for any existing Essence alpha. To better incorporate the business aspect into Essence in order to make it more suitable for the startup context, we propose the following: (1) a fourth area of concern for business aspects should be added, and (2) new alphas for this business area of concern should be added. We suggest that funding, business model, and marketing could be new alphas for this area of concern.

Alphas are things to work with and while using Essence one tracks progress on the alphas, each of which is split into alpha states to aid in this process. Therefore, each of these three new alphas should be in some way measurable. First, funding pivotal for any startup [3], and can be quantitatively measured with various metrics, making it a straightforward alpha. Progress on this alpha is likely to fluctuate as cash is burned and new funding is obtained. Secondly, business model development is at the core of a startup [17]. Indeed, one widely used definition for what is a startup posits that a startup

is a "temporary organization designed to look for a business model that is repeatable and scalable" [2]. Startups constantly invest resources into validating that they are trying to address a real need. Progress on business model development could be tracked by evaluating how well the current business model is functioning and to what extent it is already operational. Thirdly and finally, marketing may warrant its own alpha. Marketing is as important to startups as it is to any other company [4]. Startups generally have less capital to spend on marketing, forcing them to get creative with it.

Alternatively, one other option would be to look at other theories and frameworks commonly utilized by startups for business model development. Potential business-related alphas could be derived e.g. from the Business Model Canvas [20].

5.1 Practical Implications

The primary practical contribution of this study are the practices listed in the tables in the results section. These practices can help guide work in software startups. Moreover, they can be used to construct methods in conjunction with other practices. Additionally, based on these practices and the data, we suggest the following implications:

- Flat organization and self-organizing teams seem to be an effective way for constructing the initial team. Self-organizing teams have been noted to be beneficial in Agile [12]. It may also be beneficial to avoid strict roles.
- You should have a clear idea of what is the core product and what features are the key features at any given moment. Having a scope too large for the product or an MVP is a frequent reason for failure in software startups [14].
- Forming close relationships with initial customers and users is beneficial. They can help you develop your product and participate in development. They can also aid in marketing. For example, user communities on social media platforms built around your (future) product can be beneficial in various ways.

5.2 Limitations of the Study

There are several limitations in this study. First, defining practices is a challenge in various ways. The level of abstraction in defining a practice can be subjective, and a single practice, when trying to describe how work should be carried out, can be described with varying levels of detail. Thus, some practices could be combined under a single practice of a higher level of abstraction rather than being split into multiple, more detailed practices. This is something that should be taken into account when looking at the practices discussed in this paper.

Secondly, practices in Essence can belong under multiple alphas. For the clarity of presentation, we chose to separate them into categories by alpha. However, some practices under one alpha could also justifiably be assigned under another alpha. Thus, the categorization in this paper is not conclusive and was used to 1) structure the analysis section, and 2) to evaluate whether each practice would fit under *any* existing alpha. Some of the business-focused practices could not clearly fit under any existing alpha, which was one of the main contributions of this study.

Thirdly, eliciting practices is also a challenge. Aside from practices explicitly considered practices by the respondents (e.g. pair programming), practices need to be defined based on what the respondents tell about their startup and its team and their work. This, too, is not a fully process if the practices are defined by an external party (researchers). We present but one way of categorizing work in startups into practices. Indeed, though they never listed them, Paternoster et al. [21] report to have found 213 practices, indicating that many more practices could be outlined based on different data.

Finally, qualitative studies can suffer from generalizability issues due to the nature of the approach. We, however, argue that 13 cases is a large enough number for some generalizability. E.g., Eisenhardt [7] suggests five cases to be sufficient for novel areas.

6 Conclusions

In this paper, we have studied Software Engineering (SE) in software startups from the point of view of practices, by means of a case study of 13 startups. Data were collected through semi-structured interviews. This set of data was used to complement and expand upon the results of an existing study that produced a list of 63 practices [6]. Based on our empirical data and this list, we propose 76 software startup practices that can be used in method engineering in the startup context.

We then took these practices and inserted them into the framework of the Essence Theory of Software Engineering to understand whether Essence also covers the aspects of SE in software startups and not just conventional SE projects. Our results suggest that the business aspect of startups is so intertwined with SE that the more business-oriented practices could not fit into the framework of Essene. We propose that Essence either be extended to include these business aspects for the startup context, or that other theories and tools are used in conjunction with it to cover the business aspect. We propose potential new alphas that could be used to extend Essence that future studies.

References

1. Besker, T., Martini, A., Lokuge, R. E., Blincoe, K., Bosch, J.: Embracing Technical Debt, from a Startup Company Perspective. In Proceedings of the 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME), IEEE (2018).
2. Blank, S.: The four steps to the epiphany: successful strategies for products that win. BookBaby (2007).
3. Chang, S. J.: Venture capital financing, strategic alliances, and the initial public offerings of Internet startups. Journal of Business Venturing, 19(5), 721-741, (2004).
4. Crowne, M.: Why software product startups fail and what to do about it. Evolution of software product development in startup companies. In Proceedings of the 2002 Engineering Management Conference IEMC'02, pp. 338-343, IEEE (2002).
5. Cruzes, D. S., Dyba, T.: Recommended steps for thematic synthesis in software engineering. In Proceedings of the 2011 Symposium on Empirical Software Engineering and Measurement (ESEM), pp. 275-284, IEEE (2011).

6. Dande, A., Eloranta, V. P., Kovalainen, A. J., Lehtonen, T., Leppänen, M., Salmimaa, T., ... Koskimies, K.: Software startup patterns - an empirical study. Tampereen teknillinen yliopisto. Tietotekniikan laitos. Raportti-Tampere University of Technology. Department of Pervasive Computing. Report; 4 (2014).
7. Eisenhardt, K. M.: Building theories from case study research. Academy of Management Review, 14(4), 532-550 (1989).
8. Ghanbari, H., Vartiainen, T., Siponen, M.: Omission of Quality Software Development Practices: A Systematic Literature Review. ACM Computing Surveys, 51(2), (2018).
9. Giardino, C., Paternoster, N., Unterkalmsteiner, M., Gorschek, T., Abrahamsson, P.: Software Development in Startup Companies: The Greenfield Startup Model. IEEE Transactions on Software Engineering, 42(6), 585-604 (2016).
10. Jacobson, I., Ng, P. W., McMahon, P., Spence, I., Lidman, S.: The essence of software engineering: the SEMAT kernel. ACM Queue, 10(10), 40 (2012).
11. Kamulegeya, G., Hebig, R., Hammouda, I., Chaudron, M., Mugwanya, R.: Exploring the Applicability of Software Startup Patterns in the Ugandan Context. In Proceedings of the 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp. 116-124, IEEE, (2017).
12. Karhatsu, H., Ikonen, M., Kettunen, P., Fagerholm, F., Abrahamsson, P.: Building Blocks for Self-Organizing Software Development Teams a Framework Model and Empirical Pilot Study. In Proceedings of the 2nd International Conference on Software Technology and Engineering (ICSTE) (2010).
13. Klotins, E.: Software start-ups through an empirical lens: are start-ups snowflakes? In Proceedings of the International Workshop on Software-intensive Business: Start-ups, Ecosystems and Platforms (SiBW) (2018).
14. Klotins, E., Unterkalmsteiner, M., Gorschek, T.: Software Engineering Antipatterns in start-ups. IEEE Software, 36(2), 118-126, (2018).
15. Klotins, E., Unterkalmsteiner, M., Gorschek, T.: Software engineering in start-up companies: An analysis of 88 experience reports. Empirical Software Engineering, 24(1), 68-102. (2019)
16. Langley, A.: Strategies for Theorizing from Process Data. Academy of Management Review, 24(4), (1999).
17. Lueg, R., Malinauskaite, L., Marinova, I.: The vital role of business processes for a business model: the case of a startup company. Problems and Perspectives in Management, (12, Iss. 4 (contin.)), 213-220, (2014).
18. Melegati, J., Goldman, A., Paulo, S.: Requirements Engineering in Software Startups: a Grounded Theory approach. 2nd Int. Work. Softw. Startups, Trondheim, Norw. (2016).
19. Nguyen-Duc, A., Wang, X., Abrahamsson, P.: What Influences the Speed of Prototyping? An Empirical Investigation of Twenty Software Startups. In Proceedings of the 2017 International Conference on Agile Software Development (XP2017), pp. 20-36. (2017).
20. Osterwalder, A., Pigneur, Y., Clark, T.: Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers. Hoboken, NJ: Wiley (2010).
21. Paternoster, N., Giardino, C., Unterkalmsteiner, M., Gorschek, T., Abrahamsson, P.: Software development in startup companies: A systematic mapping study. Information and Software Technology, 56(10), 1200-1218 (2014).
22. Tolvanen, J. P.: Incremental method engineering with modeling tools: theoretical principles and empirical evidence. Ph. D. Thesis, University of Jyvaskyla (1998).
23. Unterkalmsteiner et al.: Software Startups - A Research Agenda. E-Informatica Software Engineering Journal, 1, 89-124 (2016).