The Evolution of Requirements Practices in Software Startups

Catarina Gralha
NOVA LINCS, DI, FCT
Universidade NOVA de Lisboa
acg.almeida@campus.fct.unl.pt

Daniela Damian
Department of Computer Science
University of Victoria
danielad@uvic.ca

Anthony I. (Tony) Wasserman
Carnegie Mellon University
Silicon Valley
tonyw@sv.cmu.edu

Miguel Goulão
NOVA LINCS, DI, FCT
Universidade NOVA de Lisboa
mgoul@fct.unl.pt

João Araújo
NOVA LINCS, DI, FCT
Universidade NOVA de Lisboa
joao.araujo@fct.unl.pt

ABSTRACT

We use Grounded Theory to study the evolution of requirements practices of 16 software startups as they grow and introduce new products and services. These startups operate in a dynamic environment, with significant time and market pressure, and rarely have time for systematic requirements analysis. Our theory describes the evolution of practice along six dimensions that emerged as relevant to their requirements activities: requirements artefacts, knowledge management, requirements-related roles, planning, technical debt and product quality. Beyond the relationships among the dimensions, our theory also explains the turning points that drove the evolution along these dimensions. These changes are reactive, rather than planned, suggesting an overall *pragmatic lightness*, i.e., flexibility, in the startups' evolution towards engineering practices for requirements. Our theory organises knowledge about evolving requirements practice in maturing startups, and provides practical insights for startups' assessing their own evolution as they face challenges to their growth. Our research also suggests that a startup's evolution along the six dimensions is not fundamental to its success, but has significant effects on their product, their employees and the company.

CCS CONCEPTS

•Software and its engineering →Software creation and management;

KEYWORDS

requirements engineering, startups, evolution, grounded theory

ACM Reference format:
Catarina Gralha, Daniela Damian, Anthony I. (Tony) Wasserman, Miguel Goulão, and João Araújo. 2018. The Evolution of Requirements Practices in Software Startups. In *Proceedings of ICSE '18: 40th International Conference on Software Engineering, Gothenburg, Sweden, May 27-June 3, 2018 (ICSE'18),* 11 pages.
DOI: 10.1145/3180155.3180158

1 INTRODUCTION

Entrepreneurship and innovation is a hot topic, reflected in the huge number of software startups around the world. Many successful products and services developed by startups, such as Uber or AirBnB, are used by millions. Startups take advantage of modern software technologies that allow them to quickly build and release software. They typically follow agile practices [18, 29], lean development [9, 44], continuous integration [17] and DevOps [16], as part of a largely unstructured approach to software engineering [20]. These startup companies are attempting to successfully identify a set of user requirements and develop effective business models around innovative products and services. They are under significant time pressure to bring their products to market with limited resources and to find a business model that allows them to survive and grow. Many startups fail to achieve these goals and shut down within in their first two years [42]. We are interested in seeing how requirements practices affect their longer term success.

Companies that make it through the early phases are interesting from a software engineering perspective, and are the focus of this work. Very little is known about how these "emerging companies" discover, prioritise and manage information about requirements over time. While the initial requirements may have been created informally – from a largely unknown market or a niche market of users willing to take risks on a new product [37] – the informal approach often does not scale well as the company grows [20]. A startup's ecosystem is highly dynamic, leading to challenges in adapting to the pressures given by the clients' requests, market pressure, developer insights or changes in platform requirements.

While a previous study documented requirements practices in small companies [1], we are not aware of empirical evidence about how requirements practices evolve as these companies mature, leading to several open questions: *How do requirements practices evolve over time in startups? What are the frictions and drivers for changes in how they manage their product and user requirements?*

We present a theory of requirements practice evolution in such emerging companies. Using a Straussian Grounded Theory (GT) approach [48], we studied the journey of 16 startups with a particular focus on how they managed their product requirements from company inception. Our theory explains the *turning points* that caused changes in their practice along 6 dimensions of requirements practice, such as the *requirements artefacts* produced and maintained, the *knowledge management* practices in documenting and communicating product requirements, the *requirements-related roles* involved in the discovery and prioritisation of requirements, the *planning* of requirements implementation and prioritisation, the *technical debt* as related to handling new requirements and the *product quality* as driven by implementation of user-driven requirements. Our theory also identifies propositions as relationships among dimensions. Our empirical data suggests clear trends towards more structured, plan-based and documentation-oriented practices. These changes are *reactive*, rather than planned, suggesting an ongoing *pragmatic lightness*, or flexibility, in the startups' evolution towards an "engineering" of requirements. Our research extends the empirical evidence about software engineering practices in early startups [20], with an emphasis on the details of evolving requirements practices. Our theory provides both a conceptual framework to organise the knowledge pertinent to the evolution of requirements in maturing startups and a practical path for such startups to improve their practices in face of specific challenges in their own journeys.

2 RELATED WORK

There has been relatively little exploration of the ways that startups develop and manage requirements [42]. Startup software requirements are usually market-driven and tend to be volatile, particularly in innovative areas. Up-front approaches are not well suited to cope with rapid changes of requirements and technology, as requirements collected at the early stages often become obsolete, before project delivery [10]. Melegati et al. [35] proposed a RE process model for startups. Based on an interview study of the state-of-the-practice in 9 Brazilian companies, the model highlights contextual elements relevant to the requirements practice: the founders, the software development manager, the market, the business model, and the entrepreneurship ecosystem. However, this model provides no information on how these practices evolved in these companies.

Agile practices, particularly flexible and reactive methods designed to stimulate customer feedback, such as lean agile approaches [43], are perceived as adequate for fast changing requirements [19]. However, they typically suffer from inadequate treatment of *domain knowledge* and differences in opinions in *customer input* [15], as well as of *non-functional requirements* (NFRs) [27]. In the absence of actual customers, alternative strategies for requirements elicitation include mining requirements from online sources (e.g., social networks, discussion forums) [24], or use of *personas* [12]. User stories [5] are often adopted in startups [42]. However the information they provide to developers has been found to be insufficient [34].

In a study of 13 startups with less than 3 years of operation, Giardino et al. [20] developed the Greenfield Startup Model describing the activities that characterise *early phase* startups. They relate to finding the product/market fit within uncertain conditions and with severe lack of resources. The model, however, does not apply to companies that have matured beyond the early phase nor discusses the evolution of their requirements practices. Similar to Hoda and Noble's study of agile approaches adoption in 31 organisations [26], Giardino et al.'s study used a GT approach [22] showing its value for studying software development processes.

With respect to startups' stages of development, previous work documents a progression from: (i) product conception and initial requirements, and first sale, to (ii) a stabilisation stage where new requirements may become unmanageable, to (iii) a growth stage where skills shortage and platform creep delay development [14], suggesting the startups' need to adapt their requirements practices as they mature. For instance, knowledge management becomes increasingly important as a risk prevention and mitigation strategy [7, 36, 45]. Similarly, Nguyen-Duc et al.'s two-cycle model [37] indicates early stage startups emphasis on a *hunting* cycle when generating ideas, eliciting requirements and customer development predominate, followed by a *gathering* cycle when the product-market fit becomes better understood and activities of requirements description, prototype implementation, automated testing and system integration dominate while the *hunting* continues, bringing innovation to the product. This model does not, however, specifically indicate the changes startups go through, why they occur, and their impact on RE practices and artefacts.

3 RESEARCH METHODOLOGY

Two research questions guided our investigation into the evolution of practices of managing requirements in "emergent companies":

RQ1. How do requirements practices change over time in these companies?

RQ2. What factors and turning points drive those changes?

We used a Straussian GT [48] approach to investigate the journey of 16 software startups that had already released software products. Acknowledging definitions of a startup company such as an *organisation in search of a scalable, repeatable, profitable business model* [8] or *a human institution designed to create a new product or service under conditions of extreme uncertainty* [44], we sampled startups based on the definition similar to that of Nguyen-Duc et al. [38]: a) a company that has recently spin-off from a large company; or b) a 4-5 year-old company that is still at a stage without a solid revenue stream; or c) an older company that has not yet gone public (i.e., has not started the initial public offering (IPO) process).

Given our focus on the startup's evolving practices, however, our sample focused on what we called "emergent companies" – those that survived the initial stages. In Table 1, we provide companies information: their age, founding date, and employee count. All still consider themselves as startups, though that term is not strictly accurate for the "older" ones. For simplicity, we will refer to these companies as *startups* instead of *emergent companies* henceforth, noting that the practices we analysed pertain to the changes that startups experience as they grow their business and operations. The companies are based in Canada, USA and Argentina, with the majority having remote workers in US, Brazil, India and Europe. In contacting the companies, 4 were included by leveraging personal contacts, while 12 were recommended by these companies as being relevant to our focus of investigation. Although two of our companies are less than 2 years old, we included them to understand where such companies start in their requirements practices.

We **collected data** by a mixed-method approach that combined interviews, all-day observations and attending project meetings. Two of the co-authors conducted 1-hour long semi-structured interviews (in person or via Skype) with at least one founding member in each company, in a total of 27 participants from 16 companies. Two of these participants have started several other companies. Following Straussian GT [48], the interview questions evolved over time: originally developed during our initial interactions with the first 4 companies (observations and attended meetings), they were later refined through the subsequent interviews, to fit the specific operation context in startups. The questions cover 5 main areas: **a)** Company growth: e.g., *How has the number of employees changed over time? (the rate of hiring and reason for hiring)*; **b)** Requirements gathering: e.g., *What/who are the sources of your requirements/features? How have the sources changed over time?* **c)** Requirements prioritisation: e.g., *How do you prioritise features? What influences the decision? How has that changed?* **d)** Features and knowledge management: e.g., *Have you changed your documentation process? How do you manage the information that is in the backlog?* **e)** Tools: e.g., *Which tools do you use (e.g., for requirements documentation, project management, communication with the team, communication with clients)? When have you started using them and why?* In addition to the interviews, we conducted full-day observations over 6 different days (1 company), attended project meetings such as stand-ups, Sprint planning, feature specification and grooming (5 companies), and conducted focus groups (3-4 people involved) (3 companies).

Table 1: Companies and participants demographics

C#	Age/founded/ #employees	P#	Role	Data collection method (frequency)
C01	4 / 2013 / 21-30	P01 P02 P03	CEO Product Manager CTO	all-day observations (6) meetings attendance (4) focus groups (4) interviews (3)
C02	6 / 2011 / 41-50	P04 P05	Director of Products CTO	meetings attendance (3) focus groups (2) interviews (1)
C03	4 / 2013 / 11-20	P06 P07 P08 P09	Developer People Operations Customer Support Developer	meetings attendance (2) focus groupn (1) interviews (1)
C04	3 / 2014 / 11-20	P10 P11	Business Analyst COO	interviews (2)
C05	4 / 2013 / 11-20	P12	CTO	interviews (1)
C06	7 / 2010 / 51-60	P13 P14 P15	Director of Products Product Manager Product Designer	meetings attendance (2) interviews (1)
C07	6 / 2011 / 21-30	P16 P17	CTO COO	interviews (1)
C08	1 / 2016 / 1-10	P18	CTO	interviews (1)
C09	4 / 2013 / 21-30	P19	CTO	interviews (1)
C10	6 / 2011 / 21-30	P20	CTO	interviews (1)
C11	9 / 2008 / 11-20	P21	CEO	meetings attendance (3) interviews (1)
C12	10 / 2007 / 51-60	P22 P23	Developer CTO	focus group (1) interviews (2)
C13	10 / 2007 / 51-60	P24	CEO	interviews (1)
C14	5 / 2012 / 51-60	P25	CEO	interviews (1)
C15	2 / 2015 / 1-10	P26	CEO	interviews (1)
C16	4 / 2013 / 21-30	P27	CEO	interviews (1)

Two of the co-authors led the **data analysis** process. We transcribed all interviews and used open coding [48] to identify patterns in the aspects of evolving requirements practice of these companies. As we iterated through the interviews, we conducted constant comparison between interviews and observation data by updating dimensions when necessary, and adjusted the elements in our emerging theory. To explain these procedures, we show an example of the raw data and the key points, dimensions, phases, and turning points that emerged. Key points are a summarised version of the sections of the interview, and one key point can lead to several dimensions, phases and turning points.

Raw data: *"Product quality was not that important, but now it is (...) It is our number one concern. When your clients are unsatisfied, it's a game breaker and it is almost in some cases a contract breaker"*; **Key points:** when the company is losing clients, product quality gains even more importance; **Dimensions:** product quality; **Phases:** from somewhat important to top priority; **Turning point:** decreased clients retention rate.

The elements in our theory represent a trend observed in the data from the majority of respondents, with exceptions noted when necessary. When new participants gave examples but no new concepts, it indicated theoretical saturation [23]. The final step in our data analysis was axial coding [48], which involves the identification of relationships between the elements, described as propositions.

Our **Theory of Requirements Practice Evolution**, according to guidelines on generating theories in software engineering [46], includes a number of *elements*: the 6 dimensions relevant to the requirements practice in our studied companies, with different phases and corresponding turning points (Fig. 1), as well as a number of *propositions* in the form of relationships describing how the evolution along one dimension brings advances in other dimensions (Fig. 2). For improving the validity of this study, we performed member checking [33], i.e., we asked all interviewees to review a draft of this paper and offer comments on our interpretations. We discussed the placing of their particular company in the phases of each dimension, and we adjusted our interpretation of their practice based on their feedback. The received comments were addressed in this version of the paper. Additional research design details such as the semi-structured interview questions and coding in our analysis, as well as company information and their placement in each dimension, can be found in the paper's supplemental material [40].

GT is classified as part of the constructivist paradigm, and its theoretical perspective is interpretivism [13]. Therefore, we structure the discussion on **threats to validity** according to interpretivists' criteria [33]: Our theory and its underlying elements and relationships should be **transferable** and relate to startups in general, not only the ones studied in this work. This threat is mitigated as a result of the theoretical saturation concept, and by selecting companies from different countries and at different phases on their evolution journey. Nonetheless, we could have potentially obtained different results had we conducted this study in even more diverse geographical and cultural settings. By using semi-structured interviews we centred on participants' opinions, which are subject to **respondent bias**. This opinion is the participant's view or perception of what is taking place, which may be at odds with reality and it is dependent on memory. In addition, it is possible that the participants may report what they believe the researchers wish to hear. This may be particularly true for companies which are reluctant to admit that they are not following perceived best practices. We attempted to mitigate this threat by performing all-day observations, attending different meetings and carrying out focus groups with some of the companies. Furthermore, we interviewed, whenever possible, people having different roles, and tried to cover the positions of CEO and CTO, since they have a broader perspective on the company evolution. On the other hand, our assumptions and personal beliefs might have introduced a **researcher bias**. We might have misinterpreted the answers from participants and therefore misidentified dimensions, phases and turning points. We attempted to mitigate this threat by performing member checking, as previously explained. The member checking process was also used to mitigate the **confirmability** threat, making sure that the findings were shaped by the participants and not by the researcher.

4 THEORY OF REQUIREMENTS PRACTICE EVOLUTION IN SOFTWARE STARTUPS

Fundamental to understanding startups' evolving requirements practice is understanding how they decide and prioritise the features to be included in successive versions of their product(s). Largely unique to startups is the speed at which they make decisions on product features. While the initial product vision is driven by the founders, the discovery and prioritisation of features are ongoing, adaptive processes that take place in a complex and rapidly changing environment. Once the product is in the market, future release planning must consider clients' requests, market and competitive analysis, bug fixing, and unplanned events, such as new platform releases, e.g., a new version of iOS [49]. These forces will drive the prioritisation of product requirements and sometimes result in a pivot [4] towards a complete new product or a redefinition of the user base and its associated requirements. Startups make these changes at a much faster pace than established firms.

But how do companies that survive longer in the market evolve to successfully respond to such pressures in their environment and how does their requirements practice evolve? Our study led us to define 6 *dimensions* relevant to the evolution of requirements practice: ① *requirements artefacts*, ② *knowledge management*, ③ *requirements-related roles*, ④ *planning*, ⑤ *technical debt* and ⑥ *product quality.* Each dimension is characterised by *3 phases of evolution*, each with corresponding *turning points.* A turning point can be either a discrete event or the "tipping point" for an ongoing trend. In either case, it precipitates changes in one or more dimensions of a company's requirements practice. We have identified 8 *turning points*, which reflect changes in: ① *number of clients*, ② *input from clients*, ③ *negative feedback*, ④ *clients retention rate*, ⑤ *revenue*, ⑥ *number of employees*, ⑦ *number of remote workers* or *flexible work hours*, and ⑧ *number of features or products.* Each turning point directly affects at least one dimension. Since dimensions have inter-relationships (discussed in Section 4.7), a turning point may indirectly affect several other dimensions. Each dimension and phases of evolution, together with its relevant turning points (TPs) are described in the following subsections. The application of our theory to the studied startups is described in Section 5.

While we limit the discussion of external funding, we note that funding plays a central role in the life of a startup and the speed at which it evolves. Raising money from friends, family, crowdfunding, angel investors, and/or venture capital firms can significantly affect the long-term success of a startup, giving it the necessary capital to hire people, and to make prospective customers aware of the company's products and services. In short, sufficient funding makes it possible to evolve requirements practices through the various dimensions and turning points described below, while inadequate funding slows a company's growth and often leads to its demise.

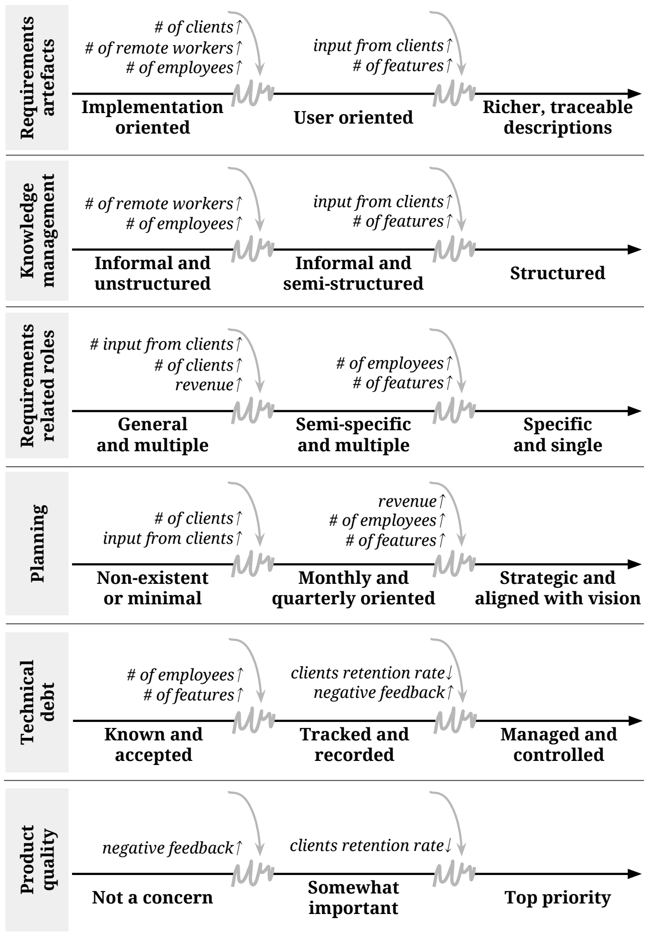

Fig. 1: Dimensions, Phases and Turning Points in the Theory of Requirements Practice Evolution

4.1 Requirements artefacts

In the early stage of a startup, there is little documentation. Product requirements may be written down on *post-it notes* visible to everyone physically present, as reported by P12, *"The requirements? It was tracked using sticky notes on a whiteboard (...) When we had an idea, when we had a thing that we needed to do, we got that on sticky note".* While startups often use a *modern collaboration platform*, such as GitHub [28] or Bitbucket [6] from the start, along with a *project management tool*, such as Asana [2], JIRA [3] or Kanbanize [30], these tools are rarely used to manage product requirements. From an understanding of requirements, **implementation-oriented** *high-level tasks* artefacts are often recorded without much rationale. Quoting P03, *"It pretty much just came to very high level details and very high progress (...) no [time or effort] estimates, no logic behind, nothing, just a simple task".* NFRs are not explicitly addressed [20].

In this early phase, the *co-founders' background* and the initial company *culture* strongly affect the types of artefacts created. P03 stated, *"When we started it was probably more implementation focused and all this other stuff, if it was getting in the way of me writing* *more code it was probably a problem".* If a team member is knowledgeable on User eXperience (UX), user needs are more likely to arise early. Lack of documentation becomes an issue when *new employees* join the company (TP), and struggle to learn the existing tacit knowledge. As P03 noted, *"I started realising onboarding problems (...) I thought well, that's probably because of the lack of documentation. And our conversation always had this, 'it was nice [if] this was documented'. (...) I see the value of that for onboarding now".* The problem is magnified if people work *remotely* (TP). When the product is first released to users, the company gains its *first clients* (TP), making it essential to view the product from their perspective.

The company then enters the second phase, and the requirements artefacts become **user oriented**. *Personas* emerge and *user stories* are used to describe existing and desired features [21, 36, 51]. Quoting P02, *"Now the users matter, they matter more, hence the user stories".* The post-it notes and whiteboards can still be used, but the information that is available in the *project management tools* starts to become more complete in order to help new hires and remote workers, as reported by P12, *"In our perspective, the best way to support remote work is that everything must be written down, no exceptions. So we have Confluence and JIRA and everything, every task, is well detailed.".* At this point, and if it was not the case before, companies may begin creating *UI designs and UX models* and testing them with users. In fact, the overall UX is important and drives some of the decisions concerning the product (see Section 4.4).

The growing number of clients increases *input from clients* (TP), such as feature requests and bug reports. That, in turn, increases the need for product support, leading to both internal documentation for support staff and to external documentation, such as FAQs, to help users with common issues. As the number of employees grows, the company has the resources to move beyond the initial product to introduce *new features* (TP) or additional product(s), both of which drive the need for greater use of *project management tools.* Developers may use burndown charts to identify the features that are being added in the current iteration, while others may use more general task-tracking and project management tools. Tracking feature requests becomes important. As P03 noted, *"We had things that weren't actually getting recorded, that information is now lost".*

When a company enters the third phase, requirements artefacts evolve into **richer, traceable descriptions**. To tackle the growing *number of features or products*, the use of the *project management tool* is often improved and systematised: each high-level task is decomposed into small actionable tasks, associated with effort and time estimates, as confirmed by P08, *"We started assigning an effort level to it, based on how much engineering may be involved in building this feature, what clients it might land us, priority levels like, do we need to do [it] right now or can this wait like 9 months".* Tasks are organised into projects or releases, prioritised and then assigned. In the most advanced startups, a feature request may include its rationale, the source of the request, an evaluation of its importance relative to the company goals, its acceptance criteria, task dependencies and other associated information. As companies refine their product management process, some requests are discarded or given low priority as they do not fit company or product goals. Some companies create a new *feature or product proposal* documenting all the needed information. Quoting P07, *"We'll take an idea and build a product proposal. Requirements gathering and research is done in order to build that proposal, which states that this new feature or this existing feature will take care of this customer, sales will be affected in this way, customer success will be affected in this way, here is what we expect the outcome of the product, [it] explains what we are doing and why". Traceability links* support managing artefacts evolution. P06 said, *"(...) we have traceability for issues and bugs that track back to Asana that map to a Google Doc for the proposal. So I've actually been trying to implement things that help the team, using the technology that we are familiar with".*

4.2 Knowledge management

Companies accumulate knowledge as they grow. At first, the knowledge is **informal and unstructured**, with the team using tacit knowledge and informal anytime, anywhere discussions. Founders of the startup team, typically 2-4 people, normally know and rely upon one another, as confirmed by P16, *"I can rely on him to remember certain things and do things in a certain way and be very precise and specific and so forth. And, I assume, he can rely on me in a similar way".* Moreover, the initial product only has a small set of features, which are fairly easy to manage, as stated by P18, *"Our product was so simple (...) it was really easy to keep track of everything without having a process or anything else but code".*

The accumulated tacit knowledge about features, products and clients becomes harder to manage as the *number of employees* (TP) and *remote workers* (TP) increases and as early hires depart [31]. This difficulty drives a company to enter the second phase, where knowledge management becomes **informal and semi-structured**. Some knowledge is valuable to everyone and can be handled through *regular team or company meetings.* Other knowledge is only relevant to a subset of employees, e.g., developers, and can be handled in meetings connected to their agile software development activity. The informal discussions and *ad hoc* verbal communication still exist, but the meetings facilitate shared knowledge about the product and the features, and are a more structured way to pass that information, as declared by P08, *"(...) regular meetings, to formalise a little bit more our communication process and have just the right amount of structure. It helped us a ton, keeping everyone in the loop".*

In this phase, *online communication tools* replace, to some extent, the *ad hoc* verbal communication. Tools like Slack [47] help in keeping the remote workers up to date and are also seen as a way to document decisions. Quoting P10, *"Slack is nice because it has a history. One thing that we are learning is that if I have a conversation with someone and I don't write everything down, it's really hard to remember every single piece of information. Slack helps with that. (...) We do daily scrums but we do them over Slack (...) because lots of the people are out of the office".* Although all content inside these tools is searchable, including files and people, it is still not a structured way to document knowledge and sometimes it is hard to separate useful information from off-topic conversations and noise.

As companies grow, they not only add staff, but divide into separate departments, such as sales, marketing, finance and development, each with its own need to manage and preserve information. Department and company meetings are inadequate, especially as the company adds *new products* and *new features* (TP) to existing products. As confirmed by P03, *"You have too much information about your product in your mind. And changes can happen everyday, this is a crazy environment. Remembering every little change is not* *possible, even if you have a great mind"*. At the same time, the product is in the market and the company receives *input from clients* (TP), which can be conveyed through sales people, bug reports or even requests for new features. That information needs to be filtered and transmitted to the relevant employees in an efficient and timely manner. Quoting P24, *"There was no information about how to document information, how to inform the rest of the company, which is not minor, because every time you release a new feature, the whole company needs to know. The customer success team needs to inform the customers, marketing needs to promote that new feature"*.

With further expansion, the company enters the third phase and the knowledge management becomes more **structured**. There is a change in the use of project management tools, which are integrated with the collaboration platform and the online communication tools. The knowledge is then documented in a more structured way, as discussed in Section 4.1. This improvement in the documentation helps in the knowledge distribution, as pointed out by P06, *"I think right now we are pretty good at documentation for knowledge. So there are parts of our product that I don't understand because I don't have the technical understanding. But the team that is responsible for that part, they build documentation that allows me to understand how it works and the value it provides"*. Of course, using such tools effectively from day one can help prevent some of the future knowledge management problems, while allowing the company to still maintain the agility and the fast-paced development needed, as reported by P25, *"It's possible to have good practices and improve the knowledge dissemination earlier because the tools are there, early on. Because with effort you could always do that, but there were more important things to do. Now, it's like, 'why not do it? It's already here'. The no-brainers are the ones like Slack, GitHub, everything that Atlassian has to offer. Anything that is between code and kind of some formalised structure, project management related"*.

4.3 Requirements-related roles

In a company's first phase, team roles may be **general and multiple**, with "everyone doing everything" as needed. As such, everyone is responsible for understanding the market, talking to potential customers, identifying key features, and more. As P07 observed, *"Two co-founders, doing everything. Customer support, development, sales, marketing, the whole shebang"*. They are constantly switching contexts, shifting from one role to the next several times during the day. As summarised by P24, *"At the very beginning, when we are 4 or 5, we are seated all together, we are talking 24/7, we are fully committed to the company and to what we are doing, we don't have many resources, we are like an octopus, we do everything"*. In most cases, one person becomes the CEO, with overall responsibility for the business, while another takes the lead in building the product.

A first significant trigger for change is when the first product has been released and has *paying customers* (TP). These customers may request new features, find bugs, and/or ask for support and training, all of which place growing demands on the existing team (TP). These demands may affect the quality and predictability of the overall work. P02 reported, *"I was doing so many things at the same time. (...) What I was doing wasn't scalable and the quality of the product was starting to go down"*. Additionally, the development is slower than it could be, because developers are performing other activities not related with code, as declared by P08, *"We had everyone on the development team doing customer support and that consumed a lot of engineering time, because engineers were on chat all day, live chat"*.

The company enters the second phase when the roles become **semi-specific and multiple** but also with a clear emphasis on requirements-related activities. New hires afforded by the higher *revenue* or external funding have more specific roles, often with a customer-facing role in *marketing*, *sales*, or *support*. P01 observed that, *"Development and product together should be roughly around 40% of the company. It's very tempting, especially in technically led companies, to make it 60, 70, 80% of the company, but the truth is if we are not marketing, [then] we are not selling, we are not supporting, then what do we do?"*. In the majority of the studied companies, when the *number of clients* increased, they found it necessary to hire or to move an employee to a *success/support* position, as confirmed by P07, *"We hired success and support [staff]. We were having more customers, and they were e-mailing us more often"*.

People in these roles not only help clients with the initial installation or use of the product, but also identify necessary new features based on their interactions with these customers. Sales people often bring new feature requests, especially when the absence of a particular feature in the product might be blocking a sale. The marketing team (perhaps a single person at first) usually assumes responsibility for maintaining the product roadmap (product management), as well as identifying the target market, crafting the marketing message, and training the sales team (product marketing). Taken together, the increase in customers and the growth of customer-facing staff brings added focus to the user experience.

While each employee wears fewer different hats than before, they still have more than one responsibility. Quoting P10, *"We started hiring more people for specific roles. We had developers, we needed to have more of those (...) we hired a client success manager to stay on track of all of our clients. But we still need to be more specialised, no one is responsible for only one thing"*.

If everything goes well and the company is successful, the *number of features or products* will increase (TP), adding to the importance and complexity of product management. Also, the *number of employees* will continue to grow (TP), leading to the third phase where the requirement-related roles are **specific and single**. The company may split out quality assurance activities from development, and hire Quality Assurance (QA) specialists who ensure that the product meets the required quality standards. Similarly, a *product manager* or a *product owner* is responsible for creating and prioritising the product backlog. Companies with multiple products will eventually have a *director of products* to oversee the various product managers and determine product priorities. In addition to the previously noted product marketing tasks, the company may add a *community manager*, who will have a key role in gathering product "wishes" from the user community, often through a company forum and social media [25]. As this specialisation proceeds, many day-to-day tasks are well covered, giving the CEO and other senior executives more time to focus on strategic issues, including the long-term vision for the company, as discussed in Section 4.4.

4.4 Planning

In the earliest stage of a startup, there is very little room for planning, which is ***non-existent or minimal***. Startups are still trying to assess market needs and to find users for their initial product.

Product features may change "on the fly", based on the wishes of a potential customer as much as on the founders' concept. In that situation, the company is highly reactive, leaving little room for long-term planning. The team's energy is primarily directed at building and demonstrating the initial version of the product in the hope of attracting customers (and possible investors).

Positive reactions to the emerging product will often be accompanied by feature requests from early adopters, and drive the next steps of product development. Negative reactions may cause the company to revise its thinking and *pivot* to a different idea. As a consequence, the feature set in the product is very fluid, often responding to requests from potential early customers (TP) rather than focusing on the founding team's initial vision. Rather than creating a single product, they would create custom developments or add features as requested by specific customers to make a sale. As P08 observed, *"We were much more hungry for clients before, so we bent over backwards a lot more, doing basically what they wanted us to do. So we would have a bunch of one-off things"*. That approach is incompatible with creating a product-focused company, especially as the *number of clients* grows (TP). At that point, the team shifts to a "standard" product, often with numerous features for customisation. Early customers then have the choice between moving to the product or staying with their "one-off" version.

With this experience of customer wishes, the company then enters the second phase, where planning becomes **monthly and quarterly-oriented**. The growing backlog of customer requests and other needed product changes [51] makes better planning necessary. Schedules, though, may still be loose, unless there are specific events that create deadlines. P02 said, *"We've always been kind of averse to deadlines (...) Deadlines don't make results better, and it's not an enjoyable place to work (...) you feel like you were working at a factory"*. In these cases, the releases are not planned: when a feature is completed, it can be released immediately. In a similar manner, a release can include only a small bug fix, and the company is able to release multiple times a day. Nonetheless, in the studied companies the norm was to plan a month ahead and, in some cases, an entire quarter. Although a fair amount of uncertainty still exists, having a core set of clients gives the startup more stability and a better sense of feature prioritisation. The development team may expand testing the user experience with its customers. P24 noted, *"User experience is really important. We work really closely with our customers, we try to interview them, and have case studies and UI tests (...) They influence our feature prioritisation in a way"*.

Competition is also an important factor in feature prioritisation. In some companies, *mining competitors' forums and social media* is an essential activity, giving the company the opportunity to gain an advantage by implementing key features first. Taken together, these various sources lead to an increase in the *number of features* (TP) or a pivot away from the product towards a different idea. Overall, planning become easier as the *number of employees* grows (TP), *revenue* increases and growth stabilises (TP). People have well-defined roles and responsibilities, and the company gains a better sense of the most promising product direction and target market(s).

The third stage is characterized by the introduction of long-term planning, encompassing *multiple releases*. Planning becomes **strategic and aligned with vision**. Feature requests, whether internally or externally generated, are systematically evaluated; those perceived as valuable additions to the product are assigned to a specific future release. However, high-priority needs, such as security holes, may alter the plans and lead to a previously unplanned release. Planning is also influenced by the company's feelings about addressing a broader or narrower market. If there is a "sweet spot" for the product that meets the needs of a large share of its customers, some existing product features may be deprecated over time. This trades off the cost of losing some customers against the savings from reduced development and support costs that arises from the narrower focus. One common example is to cease support for obsolete platforms or product integrations, particularly if they are no longer aligned with the product vision. At this stage of planning, the company has a good sense of their customers and their needs, as well as the marketplace, making it easier for them to decide on addition and removal of product features. P08 commented, *"Now there's a lot more thought into bringing new things on and what we will do for a customer. (...) We have to decide what actually we think is going to be better for our clients. If they want something that is very particular to them, for their business, we can't do it"*.

4.5 Technical debt

"Technical debt" refers to the accumulated backlog of software development needed because developers favour a "quick and dirty" over a more complete solution, usually to reduce the overall implementation time. Such *shortcuts and workarounds* are often made by startups [50] in their effort to bring their product to market as quickly as possible, and to validate their assumptions about their target market and the problem they are trying to solve [20]. Quickly finding the right product features to fit a particular market is the highest priority, as reported by P06, *"Basically everything was built with popsicle sticks and duct tape, because it was fast, as fast as you can get something out"*. Even though the company **knows and accepts** the existence of technical debt, it is generally not tracked. Albeit it would not be hard to include such information in their product management tool, the developers rarely document which features are affected by the shortcut or workaround, nor the way it might affect future features. Speed is paramount.

As the *number of features* increases (TP) and the product becomes more complex, problems may appear. Earlier shortcuts and workarounds may affect long-term development, performance and maintainability of the product. Implementing a workaround in one feature might break another one. On the other hand, fixing a workaround on one feature might break another feature, dependent on the previous implementation of the first one. With more developers (TP), it becomes possible (and often necessary) to address these problems. Generally, the person who created the debt is in the best position to reduce it, since a new hire will not be as knowledgeable.

Becoming aware of technical debt [32] and then **tracking and recording** it, marks advancement to the second stage for the company. At this point, startups not only know that technical debt exists, but also which features are affected by it and how. Nonetheless, this does not mean that the company will start paying this debt immediately, since first the startup needs to make a conscious decision to tackle it. However, with more people the company has the resources to do so, as confirmed by P02, *"It probably was when we hired more developers. I could start focusing on some of the debt. Not completely pay for it (...) but at least knowing where it is"*. The debt will continue to exist and even grow if it is not causing the product to fail and does not substantially affect the clients. Quoting P18, *"It's a hard decision, but if the debt is not causing the product to crash, we'll not prioritise the payment [of the technical debt]"*.

When the technical debt affects the product quality, perhaps causing it to malfunction or perform poorly, companies begin to have *negative feedback* from clients (TP). With modern public product review mechanisms, this negative feedback may be widely seen.

In most of the studied companies, the business model followed a *subscription plan*, which means that the startup would have a paying client for the entire period of the subscription, regardless of their satisfaction with the product. This provides some financial stability during that period of time, as declared by P19, *"(...) it gives us some time and financial stability to continue working and to tackle some of the problems that might exist"*. On the other hand, if the client is not content with the product, they will not renew their subscription, and the *client retention rate* will decrease (TP), causing a loss in future revenue. That risk usually causes the company to prioritise the most serious technical debt-related problems in their product development plans to improve client satisfaction.

Addressing technical debt systematically is central to the third phase, where it is **managed and controlled**, with the development team addressing accumulated technical debt. This involves rewriting and refactoring portions of the code. Most importantly, this engineering work is integrated into the overall product release planning, making it visible to the product manager, and customers. At this stage, requirements practice is no longer focused exclusively on product features, but also encompasses NFRs that must be addressed by the development team to assure high product quality.

Retirement of technical debt is just one piece of a maturing development organisation, where code reviews, automated tests, and coding standards become standard practice. These changes do not occur at the same time, and normally are motivated by a new feature being dependent on some part of the code that was already written. A logic of "two birds one stone" is followed, as commented by P03, *"it's usually only strategic, the need arises because of a new feature (...) if there is a big feature that is coming along, then I can look at it, fix our technical debt there"*. In spite of this fact, some technical debt will always persist, as reported by P05, *"If you get to zero, you are probably doing something wrong. You are never done"*.

4.6 Product quality

In the early stages of a startup, speed of release takes precedence over quality, the latter being **not a concern**. NFRs have low priority compared to validating the product idea and the market. Testing is minimal, since small problems can be quickly corrected in the next release (which can be deployed in the same day, since they follow practices of continous deployment [41]), diminishing the need to be concerned with every aspect of product quality. Quoting P12, "*Our development speed is something that we really don't want to give up, so just get it out and test it, because we can always go back, we can always change it or fix it quickly"*.

The initial clients are normally aware of this reality, and have a constructive attitude towards such bugs. However, this generous view starts to change over time, since the clients expect an increase in the product quality level, especially if they are depending on it for a critical business function. If the product continues to perform below expectations, this will result in an increased *negative feedback* (TP). As seen in Section 4.5, in the long-term this will cause the company to lose clients and, therefore, revenue. Quoting P13, "*(...) we started caring about quality because we were losing clients. [The product] was great for marketing, but we had problems delivering"*.

Product quality becomes **somewhat important** in the second phase. The company wants to avoid negative feedback, and they now have employees who can address issues of quality, including the NFRs of security and usability, factors that mainly affect the clients. P24 commented, *"We have a small QA team and they are not coming from a tech side, they are worried about the user experience and try to test the product the way a user would do it and try to find flaws. User experience is one of the most important quality aspects"*. Scalability also gains importance, because the company is hoping for rapid growth in the number of users, and must be prepared.

As the company grows, additional NFRs, such as availability and efficiency, gain importance as a way to maximise the *client retention rate* (TP). P06 said, *"Product quality was not that important, but now it is (...) It is our number one concern. When your clients are unsatisfied, it's a game breaker and it is almost in some cases a contract breaker. Even to bring new clients in, they expect a certain number of features and a certain amount of uptime"*.

In the third phase the product quality is a **top priority** for the company. The set of NFRs taken into consideration expands from usability and scabability, to cover efficiency, fault-tolerance, maintainability, and availability, among others. While some startups are slow to address overall issues of product quality, others attach great importance to product quality from the outset, as observed by P10, "*[Compromising quality:] that's something that we really really don't like to do (...) quality is one of the most important things, the clients would not tolerate any other way"*. In these cases, poor quality in the first release might undermine the startup's chances of survival. Additionally, quality (or the lack of it) can sometimes be perceived as a way to gain (or lose) reputation, as commented by P11, *"Quality is something that is really important, because it reflects on us"*. Startups at this third phase devote significant resources to the various aspects of product quality, knowing that product deficiencies will cause them to lose market share to competitors through defecting clients and inability to gain new customers.

4.7 Relationships among dimensions

Following the principles of generating SE theories [46], this study does not just present a set of descriptive elements but also defines the key relationships among them. Our empirical data suggest that an action of the company to advance along one of the dimensions brings along advances in other dimensions. *Culture* has emerged as another strong determining factor affecting the evolution along all the dimensions in our theory, so it deserves to be highlighted here. In Fig. 2 we include the 6 previously identified dimensions alongside *Culture* and the 9 propositions about the relationships among them. We summarise them briefly next.

P1: As requirements artefacts become more complete and structured, the communication of changes becomes more effective and vice versa.

P2: Specialisation in requirements-related roles enables the use of more sophisticated knowledge management practices and tools.

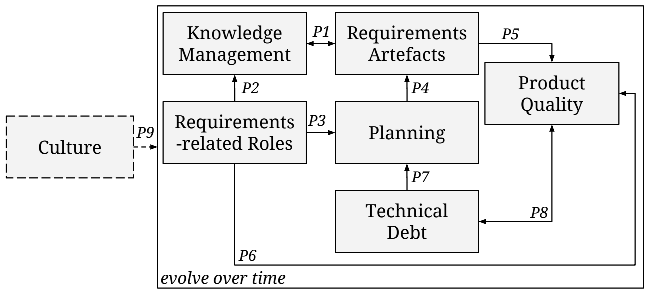

Fig. 2: Dimensions, Culture and Propositions in the Theory of Requirements Practice Evolution

P3: Hiring more people in requirements-related roles allows the company to devote more resources to product planning.

P4: Improved planning allows the creation of more complete and structured requirements artefacts.

P5: Richer requirements artefacts allow for enhanced understanding and treatment of product quality (NFR).

P6: Creating and hiring people into more specific requirements-related roles allows the company to better handle product features and to address NFRs.

P7: Improved ability to handle technical debt results in a higher ability to prioritise requirements with greatest customer impact.

P8: Handling technical debt is enhanced by and increases the company's ability to prioritise product quality.

P9: All changes are influenced by a combination of company culture and the co-founders' backgrounds.

While details of these relationships might be apparent from the richer descriptions of the theory dimensions in Section 4, we illustrate some of them here. Consider propositions P2, P3, and P6. The evolution of requirements related roles, i.e., adding more people and giving them specialised roles (P2), means that the company is better able to plan its product development and release schedule (P3), as well as to assure the quality of the evolving product (P6). A product manager is given responsibility for creating and prioritising the product backlog, planning the releases and deciding what features the startup needs to build next. At the same time, the increase in company headcount forces the company to improve its knowledge management practices. Everyone, from the CEO on down, has the responsibility to collect and manage the knowledge about the company's vision, products, and markets. That makes it easier for the company to make both strategic and tactical decisions, as well as to quickly bring new hires up to speed.

Finally, in all the studied companies, the *culture* and the experience of the co-founders determined how the startup operated during its early days, as also found in [11]. Experienced leaders may know which practices are most effective, and have a better sense of when to bring in people with specific skills, making adjustments based on the skills and the location of employees. Hiring the "right people" is critical at this early stage, where a poor hire can create major problems. Good leaders will encourage collaboration and communication, leading to better product quality and retention of key artefacts (knowledge management). When the co-founders have experience working in a larger software company, there is clear tendency to adapt (and simplify) some of the more heavy-weight processes to the fast-paced world of a startup. In short, the founders set the tone for the company, and have a huge influence on its technical and business practices.

5 DISCUSSION

5.1 Applying the theory

To illustrate the applicability of our theory, we mapped the companies in our study along the 6 dimensions of evolution (available as a web resource [39]). Here we include a radar diagram showing the two startups that changed the least and the most in evolving their requirements practices, C06 and C03 respectively, as well as the mode for each dimension (the value that appears most often in the data set). As the mode shows, the second phase is the most common one in the studied companies, for all the dimensions. We discuss the case of C03 and C06 further in Section 5.3.

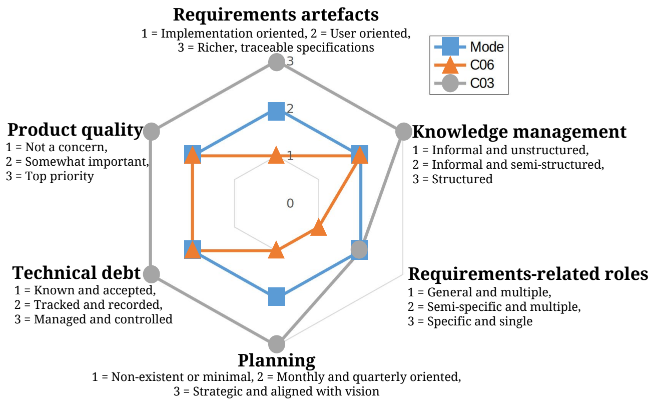

Fig. 3: Positioning the companies along the 6 dimensions of the Theory of Requirements Practice Evolution

As an implication for practice, a startup can use our theory to place itself along each dimension and track their evolution. Seeing the evolution for other companies, the startup can plan ahead by assessing likely transitions in their practice, thus gaining some insight about how to address an identified turning point. However, the influence of people and company culture is an important factor to consider when analysing startups' potential for change. Some companies may resist change based on the beliefs of the co-founders or CEO. Others might evolve earlier and offer less resistance to change due to the previous experience of the team or the impact of a new hire. The way a founder or a CEO runs a startup is related to his or her preconceived notions about how a software system should work and what degree of processes and requirements practices are needed. In the end, the decision to change or not is always related to people and their beliefs, which might also change over time.

5.2 Towards Requirements Engineering for startups

We concluded that startups do not practice an "engineering" approach to requirements development. Our evidence suggests that they generally perceive up-front requirements practices as being a waste of time and a hindrance to releasing their product as soon as possible. Exhaustive documentation or long-term planning are discarded in order to let the company move faster, achieve rapid time-to-market and validate previous assumptions regarding the clients and the product-market fit. Innovation in their products and their positioning in the market – key to the companies' survival or success – takes place in a fast-paced, volatile context, and is only possible through processes that are reactive and evolutionary. Our search for a "final product" that is carefully designed and planned did not take us far. Instead, a startup's product is continuously being iterated and updated based on customer feedback, employer insights, competitive issues, market demands and ongoing changes in the frameworks and platforms used within the product's market ecosystem. Fast adaptation in how product features are generated and prioritised is crucial to the overall success of the company. If the startup is not able to react in a timely manner, it will perish.

However, a deeper look at our findings reveals an evolution towards more structured, planned and documentation-oriented requirements practices. Over time, the companies become more customer-oriented (in the information elicited as well as documentation used), to better understand their clients and their needs. These changes are *reactive*, rather than planned, and only if they bring benefits without compromising the agility and speed of the company. They suggest an overall *pragmatic lightness*, i.e., flexibility, in the startups' evolution towards an "engineering" of requirements.

5.3 Is evolution along the six dimensions fundamental to the success of a startup?

While different interviewees had varying ideas about what it means to be a *successful* startup, two key aspects are *survival* and *longevity*. Long-term success is not easily predictable, and is affected by both internal and external factors, including the type of product and its market, human resources, company culture, available funding, processes and practices, and even luck or serendipity. While some of these factors cannot be controlled, one that can be managed is the choice of processes and practices used to operate and steer the startup on a daily basis, and the conscious decision to make changes and evolve along the dimensions identified in this paper.

One question that arises from our work is whether "maturity" in terms of the evolving requirements practice is necessary to the success of a startup. Although we would have liked to answer positively, the short answer is "no". The long answer is, however, far more complex and interesting, as shown by two companies from our study – shown as C06 and C03 on our map in Fig. 3.

Company C06 (seen as the orange line with triangles in Fig. 3) is almost 10-years-old, has a solid number of clients and revenue, and a growing number of employees. Its products are continually enhanced, with clients from all around the world. This startup is considered successful despite their longstanding unstructured software development processes. The existing requirements artefacts are fairly basic and maintained in tool such as JIRA, used here for managing high-level tasks and bug tickets. The knowledge is managed informally, through weekly meetings but mainly relying on *ad hoc* undocumented verbal communication. Employees are still playing more than one role, with no long-term planning. Technical debt is not well addressed and product quality is not a top priority. Nonetheless, C06 shows that a company can be successful without having a well-defined requirements process. However, useful information is frequently lost because it is not recorded, the work hours are long, and the environment is stressful, which should not be the case for a 10-year-old company.

Company C03 (represented by a line with squares in Fig. 3), being 4-years-old and having a smaller number of employees, presents a contrasting profile. While there is still room for evolution, C03 is in the most advanced phase along most dimensions. They have a well-defined process for defining tasks and product features, with traceability between every requirements artefact, making full use of their project management tools. Everyone can see the answer to the Five Ws (what, who, where, when and why) for new features. The planning is aligned with the company vision, and there is a deep understanding about the product, the clients and the market. In short, company C03 has evolved from an informal approach to requirements to more of an engineering process. C03 can be more confident about their product development and schedules, as well as about growing their company product roadmap to meet their customers and market needs. This approach allows them to build a higher quality product and avoid technical debt.

6 CONCLUSIONS

This study has characterised the evolution of requirements practices in software startups along six interrelated dimensions. Progress along these dimensions reflects improved ability by a company to reliably deliver high quality products to customers, to manage requirements more effectively, and to add staff to do so. Because these dimensions are tied to one another (as shown in Section 4.7), an advance in one dimension often facilitates advances in others. Evolution along the six dimensions is not fundamental to the success of a startup, but it has significant benefits and positively impacts the product, the employees and the company as a whole, since there is a sense of purpose: everyone knows what they are doing and why, and employees are more motivated and less stressed.

According to guidelines on evaluating theories in SE [46], we already discussed the utility of our theory in providing support to startups' assessing their own evolution as they grow. Further, there are many opportunities for refining and evaluating our theory in future work. The companies in our sample set are still operating and all have made progress (at varying speeds) along the six dimensions. We intend to refine our theory to investigate additional factors that might affect the evolution of their requirements practice such as loss of key people, abandoned initial product ideas in favour of a new direction (pivot), and/or regressions in their requirements practices. It would also be interesting to interview founders of failed startups to see the impact (if any) of their requirements practices. Interviews with serial entrepreneurs would be particularly valuable, since they can compare and contrast their experiences across multiple startups. Further studies should also evaluate our theory by testing our propositions in future empirical studies targeting more and diverse companies.

ACKNOWLEDGMENTS

We thank all study participants and financial granting agencies: NOVA LINCS UID/CEC/04516/2013; FCT-MCTES research grant SFRH/BD/108492/2015; NSERC.

REFERENCES

[1] Jorge Aranda, Steve M. Easterbrook, and Greg Wilson. 2007. Requirements in the wild: How small companies do it. In *15th International Requirements Engineering Conference*. 39–48. DOI:https://doi.org/10.1109/RE.2007.54

[2] Asana. 2018. Use Asana to track your team's work & manage projects. (2018). https://asana.com/ (access: February, 2018).

[3] Atlassian. 2018. JIRA — Issue & Project Tracking Software. (2018). https://atlassian.com/software/jira (access: February, 2018).

[4] Sohaib Shahid Bajwa, Xiaofeng Wang, Anh Nguyen-Duc, Rafael Matone Chanin, Rafael Prikladnicki, Leandro Bento Pompermaier, and Pekka Abrahamsson. 2017. Start-Ups Must Be Ready to Pivot. *IEEE Software* 34, 3 (2017), 18–22. DOI:https://doi.org/10.1109/MS.2017.84

[5] Kent Beck. 1990. *Extreme programming explained: embrace change.* Addison-Wesley. http://www.worldcat.org/oclc/41834882

[6] Bitbucket. 2018. The Git solution for professional teams. (2018). https://bitbucket.org/ (access: February, 2018).

[7] Finn Olav Bjørnson and Torgeir Dingsøyr. 2008. Knowledge management in software engineering: A systematic review of studied concepts, findings and research methods used. *Information & Software Technology* 50, 11 (2008), 1055–1068. DOI:https://doi.org/10.1016/j.infsof.2008.03.006

[8] Steve Blank and Bob Dorf. 2012. *The startup owner's manual: The step-by-step guide for building a great company.* K&S Ranch Press.

[9] Jan Bosch, Helena Holmström Olsson, Jens Björk, and Jens Ljungblad. 2013. The Early Stage Software Startup Development Model: A Framework for Operationalizing Lean Principles in Software Startups. In *4th International Conference on Lean Enterprise Software and Systems*. 1–15. DOI:https://doi.org/10.1007/978-3-642-44930-7_1

[10] Lan Cao and Balasubramaniam Ramesh. 2008. Agile requirements engineering practices: An empirical study. *IEEE Software* 25, 1 (2008). DOI:https://doi.org/10.1109/MS.2008.1

[11] Gerry Coleman and Rory V O'Connor. 2008. An investigation into software development process formation in software start-ups. *Journal of Enterprise Information Management* 21, 6 (2008), 633–648. DOI:https://doi.org/10.1108/17410390810911221

[12] Alan Cooper. 1999. The Inmates are Running the Asylum–Why High-Tech Products Drive Us Crazy and How 2 to Restore the Sanity. *SAMS. ISBN: 0-67231-649-8* (1999).

[13] Michael Crotty. 1998. *The foundations of social research: Meaning and perspective in the research process.* Sage.

[14] Mark Crowne. 2002. Why software product startups fail and what to do about it. Evolution of software product development in startup companies. In *IEEE International Engineering Management Conference*, Vol. 1. IEEE, 338–343. DOI:https://doi.org/10.1109/IEMC.2002.1038454

[15] Maya Daneva, Egbert Van Der Veen, Chintan Amrit, Smita Ghaisas, Klaas Sikkel, Ramesh Kumar, Nirav Ajmeri, Uday Ramteerthkar, and Roel Wieringa. 2013. Agile requirements prioritization in large-scale outsourced system projects: An empirical study. *Journal of Systems and Software* 86, 5 (2013), 1333–1353. DOI:https://doi.org/10.1016/j.jss.2012.12.046

[16] Patrick Debois. 2011. Devops: A software revolution in the making. *Journal of Information Technology Management* 24, 8 (2011), 3–39.

[17] Brian Fitzgerald and Klaas-Jan Stol. 2017. Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software* 123 (2017), 176–189. DOI:https://doi.org/10.1016/j.jss.2015.06.063

[18] Martin Fowler and Jim Highsmith. 2001. The Agile Manifesto. *Software Development* 9, 8 (2001), 28–35.

[19] Carmine Giardino, Sohaib Shahid Bajwa, Xiaofeng Wang, and Pekka Abrahamsson. 2015. Key challenges in early-stage software startups. In *International Conference on Agile Software Development, Agile Processes in Software Engineering and Extreme Programming. XP 2015 (LNBIP)*, Vol. 212. Springer, 52–63. DOI:https://doi.org/10.1007/978-3-319-18612-2_5

[20] Carmine Giardino, Nicolo Paternoster, Michael Unterkalmsteiner, Tony Gorschek, and Pekka Abrahamsson. 2016. Software development in startup companies: the greenfield startup model. *IEEE Transactions on Software Engineering* 42, 6 (2016), 585–604. DOI:https://doi.org/10.1109/TSE.2015.2509970

[21] Carmine Giardino, Michael Unterkalmsteiner, Nicolo Paternoster, Tony Gorschek, and Pekka Abrahamsson. 2014. What do we know about software development in startups? *IEEE Software* 31, 5 (2014), 28–32. DOI:https://doi.org/10.1109/MS.2014.129

[22] Barney G. Glaser. 1978. *Theoretical Sensitivity: Advances in the Methodology of Grounded Theory.* Sociology Press, Mill Valley, CA, USA.

[23] Barney G. Glaser and Anselm L. Strauss. 1967. *The Discovery of Grounded Theory: Strategies for Qualitative Research.* Aldine de Gruyter, New York.

[24] Emitza Guzman, Rana Alkadhi, and Norbert Seyff. 2016. A Needle in a Haystack: What Do Twitter Users Say about Software?. In *24th International Requirements Engineering Conference*. IEEE, 96–105. DOI:https://doi.org/10.1109/RE.2016.67

[25] Wu He and Gongjun Yan. 2014. Mining blogs and forums to understand the use of social media in customer co-creation. *The Computer Journal* 58, 9 (2014), 1909–1920. DOI:https://doi.org/10.1093/comjnl/bxu038

[26] Rashina Hoda and James Noble. 2017. Becoming Agile: A Grounded Theory of Agile Transitions in Practice. In *39th International Conference on Software Engineering*. IEEE Press, 141–151. DOI:https://doi.org/10.1109/ICSE.2017.21

[27] Irum Inayat, Siti Salwah Salim, Sabrina Marczak, Maya Daneva, and Shahaboddin Shamshirband. 2015. A systematic literature review on agile requirements engineering practices and challenges. *Computers in Human Behavior* 51 (2015), 915–929. DOI:https://doi.org/10.1016/j.chb.2014.10.046

[28] GitHub Inc. 2018. The worl's leading software development platform. (2018). https://github.com/ (access: February, 2018).

[29] PwC / CB Insights. 2017. *MoneyTree Report, Q1, 2017.* Technical Report. PwC / CB Insights. https://www.pwc.com/us/en/moneytree-report/assets/MoneyTree_Report_Q1_2017_FINAL_F.pdf.

[30] Kanbanize. 2018. Kanban Software — Online Kanban for Lean Project Management. (2018). https://kanbanize.com/ (access: February, 2018).

[31] Andrew J Ko. 2017. A three-year participant observation of software startup software evolution. In *39th International Conference on Software Engineering: Software Engineering in Practice Track.* IEEE Press, 3–12. DOI:https://doi.org/10.1109/ICSE-SEIP.2017.29

[32] Philippe Kruchten, Robert L Nord, and Ipek Ozkaya. 2012. Technical debt: From metaphor to theory and practice. *IEEE Software* 29, 6 (2012), 18–21. DOI:https://doi.org/10.1109/MS.2012.167

[33] Yvonna S Lincoln and Egon G Guba. 1985. *Naturalistic inquiry.* Vol. 75. Sage.

[34] Juliana Medeiros, Alexandre Vasconcelos, Miguel Goulão, Carla Silva, and João Araújo. 2017. An Approach Based on Design Practices to Specify Requirements in Agile Projects. In *Symposium on Applied Computing*. ACM, New York, NY, USA, 1114–1121. DOI:https://doi.org/10.1145/3019612.3019753

[35] Jorge Melegati and Alfredo Goldman. 2016. Requirements Engineering in Software Startups: a Grounded Theory approach. In *22nd ICE/IEEE International Technology Management Conference.* IEEE, 57–65.

[36] Luciana Maria Azevedo Nascimento and Guilherme Horta Travassos. 2017. Software Knowledge Registration Practices at Software Innovation Startups: Results of an Exploratory Study. In *31st Brazilian Symposium on Software Engineering.* 234–243. DOI:https://doi.org/10.1145/3131151.3131172

[37] Anh Nguyen-Duc, Pertti Seppänen, and Pekka Abrahamsson. 2015. Hunter-gatherer cycle: a conceptual model of the evolution of software startups. In *International Conference on Software and System Process.* ACM, 199–203. DOI:https://doi.org/10.1145/2785592.2795368

[38] Anh Nguyen-Duc, Syed Muhammad Ali Shah, and Pekka Ambrahamsson. 2016. Towards an early stage software startups evolution model. In *42th Euromicro Conference on Software Engineering and Advanced Applications.* IEEE, 120–127. DOI:https://doi.org/10.1109/SEAA.2016.21

[39] Theory of Requirements Practice Evolution. 2018. Companies and dimensions. (2018). https://goo.gl/KY1qRP (access: February, 2018).

[40] Theory of Requirements Practice Evolution. 2018. Supplemental Material. (2018). https://goo.gl/VYXAj7 (access: February, 2018).

[41] Chris Parnin, Eric Helms, Chris Atlee, Harley Boughton, Mark Ghattas, Andy Glover, James Holman, John Micco, Brendan Murphy, Tony Savor, and others. 2017. The top 10 adages in continuous deployment. *IEEE Software* 34, 3 (2017), 86–95. DOI:https://doi.org/10.1109/MS.2017.86

[42] Nicolò Paternoster, Carmine Giardino, Michael Unterkalmsteiner, Tony Gorschek, and Pekka Abrahamsson. 2014. Software development in startup companies: A systematic mapping study. *Information and Software Technology* 56, 10 (2014), 1200–1218. DOI:https://doi.org/10.1016/j.infsof.2014.04.014

[43] Mary Poppendieck and Tom Poppendieck. 2003. *Lean Software Development: An Agile Toolkit: An Agile Toolkit.* Addison-Wesley.

[44] Eric Ries. 2011. *The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses.* Crown Business.

[45] Ioana Rus and Mikael Lindvall. 2002. Knowledge management in software engineering. *IEEE Software* 19, 3 (2002), 26. DOI:https://doi.org/10.1109/MS.2002.1003450

[46] Dag I. K. Sjøberg, Tore Dybå, Bente C.D. Anda, and Jo E. Hannay. 2008. Building theories in software engineering. *Guide to advanced empirical software engineering* (2008), 312–336.

[47] Slack. 2018. Slack: Where work happens. (2018). https://slack.com/ (access: February, 2018).

[48] Anselm L. Strauss and Juliet M. Corbin. 1998. *Basic of qualitative research: Techniques and procedures for developing Grounded Theory.* Sage Publications.

[49] Tech.Co. 2017. Apple iOS 11 Update Could End the Life of 200,000 Old Apps. (2017). https://tech.co/apple-ios-11-dead-apps-2017-06 (access: February, 2018).

[50] Edith Tom, AybüKe Aurum, and Richard Vidgen. 2013. An exploration of technical debt. *Journal of Systems and Software* 86, 6 (2013), 1498–1516. DOI:https://doi.org/10.1016/j.jss.2012.12.052

[51] Shi Zhong, Chen Liping, and Chen Tian-en. 2011. Agile planning and development methods. In *3rd International Conference on Computer Research and Development*, Vol. 1. IEEE, 488–491. DOI:https://doi.org/10.1109/ICCRD.2011.5764064