Author(s): Unterkalmsteiner, Michael; Abrahamsson, Pekka; Wang, XiaoFeng; Nguyen-Duc, Anh; Shah, Syed; Bajwa, Sohaib Shahid; Baltes, Guido H.; Conboy, Kieran; Cullina, Eoin; Dennehy, Denis; Edison, Henry; Fernandez-Sanchez, Carlos; Garbajosa, Juan; Gorschek, Tony; Klotins, Eriks; Hokkanen, Laura; Kon, Fabio; Lunesu, Ilaria; Marchesi, Michele; Morgan, Lorraine; Oivo, Markku; Selig, Christoph; Seppänen, Pertti; Sweetman, Roger; Tyrväinen, Pasi; Ungerer, Christina; Yagüe, Agustin

Title: Software Startups : A Research Agenda

Year: 2016

Version:

Please cite the original version:

Unterkalmsteiner, M., Abrahamsson, P., Wang, X., Nguyen-Duc, A., Shah, S., Bajwa, S. S., Baltes, G. H., Conboy, K., Cullina, E., Dennehy, D., Edison, H., Fernandez-Sanchez, C., Garbajosa, J., Gorschek, T., Klotins, E., Hokkanen, L., Kon, F., Lunesu, I., Marchesi, M., . . . Yagüe, A. (2016). Software Startups : A Research Agenda. e-Informatica Software Engineering Journal, 10(1), 89-123. https://doi.org/10.5277/e-Inf160105 e-Informatica Software Engineering Journal, Volume 3, Issue 1, 2009

Software Startups - A Research Agenda

Michael Unterkalmsteiner*, Pekka Abrahamsson**, XiaoFeng Wang***, Anh Nguyen-Duc****, Syed Shah*****, Sohaib Shahid Bajwa******, Guido H. Baltes*******, Kieran Conboy********, Eoin Cullina********, Denis Dennehy********, Henry Edison*********, Carlos Fernandez-Sanchez**********, Juan Garbajosa***********, Tony Gorschek************, Eriks Klotins************, Laura Hokkanen*************, Fabio Kon**************, Ilaria Lunesu***************, Michele Marchesi***************, Lorraine Morgan****************, Markku Oivo*****************, Christoph Selig******************, Pertti Seppänen*******************, Roger Sweetman********************, Pasi Tyrväinen*********************, Christina Ungerer**********************, Agustin Yagüe***********************

*Blekinge Institute of Technology, Sweden
**Norwegian University of Science and Technology, Norway
***Free University of Bolzano-Bozen, Italy
****Norwegian University of Science and Technology, Norway
*****SICS, Sweden
******Free University of Bolzano-Bozen, Italy
*******Lake Constance University, Germany
********National University of Ireland Galway, Ireland
*********Free University of Bolzano-Bozen, Italy
**********Universidad Politécnica de Madrid, Spain
***********Technical University of Madrid, Spain
************Blekinge Institute of Technology, Sweden
*************Tampere University of Technology, Finland
**************University of São Paulo, Brazil
***************University of Cagliari, Italy
****************National University of Ireland Maynooth, Ireland
*****************University of Oulu, Finland
******************Hochschule Konstanz, Germany
*******************University of Oulu, Finland
********************National University of Ireland Galway, Ireland
*********************University of Jyväskylä, Finland
**********************Hochschule Konstanz, Germany
***********************Universidad Politécnica de Madrid, Spain

mun@bth.se, pekkaa@ntnu.no, xiaofeng.wang@unibz.it, anhn@idi.ntnu.no, shah@sics.se, bajwa@inf.unibz.it, guido.baltes@cetim.org, kieran.conboy@nuigalway.ie, eoin.cullina@outlook.com, denis.dennehy@nuigalway.ie, henry.edison@inf.unibz.it, carlos.fernandez@upm.es, jgs@eui.upm.es, tgo@bth.se, ekx@bth.se, laura.hokkanen@tut.fi, fabio.kon@ime.usp.br, ilaria.lunesu@diee.unica.it, michele@diee.unica.it, lorraine.morgan@nuim.ie, markku.oivo@oulu.fi, cselig@htwg-konstanz.de, pertti.seppanen@oulu.fi, roger.sweetman@nuigalway.ie, pasi.tyrvainen@jyu.fi, christina.ungerer@htwg-konstanz.de, ayague@etsisi.upm.es

Abstract

Software startup companies develop innovative, software-intensive products within limited time frames and with few resources, searching for sustainable and scalable business models. Software startups are quite distinct from traditional mature software companies, but also from micro-, small-, and medium-sized enterprises, introducing new challenges relevant for software engineering research. This paper's research agenda focuses on software engineering in startups, identifying, in particular, 70+ research questions in the areas of supporting startup engineering activities, startup evolution models and patterns, ecosystems and innovation hubs, human aspects in software startups, applying startup concepts in non-startup environments, and methodologies and theories for startup research. We connect and motivate this research agenda with past studies in software startup research, while pointing out possible future directions. While all authors of this research agenda have their main background in Software Engineering or Computer Science, their interest in software startups broadens the perspective to the challenges, but also to the opportunities that emerge from multi-disciplinary research. Our audience is therefore primarily software engineering researchers, even though we aim at stimulating collaborations and research that crosses disciplinary boundaries. We believe that with this research agenda we cover a wide spectrum of the software startup industry current needs.

1. Introduction

Researchers are naturally drawn to complex phenomena that challenge their understanding of the world. Software startup companies are an intriguing phenomenon, because they develop innovative software-intensive[1] products under time constraints and with a lack of resources [142], and constantly search for sustainable and scalable business models. Over the past few years, software startups have garnered increased research interest in the Software Engineering (SE) community.

While one could argue that software startups represent an exceptional case of how software products are developed and brought to the market, several factors suggest a broader impact. From an economical perspective, startups contribute considerably to overall wealth and progress by creating jobs and innovation [3]. Digital software startups[2] are responsible for an astonishing variety of services and products [165]. In the farming sector, venture investment in so-called "AgTech" start-ups reached $2.06 billion in just the first half of 2015; this figure neared the $2.36 billion raised during the whole of 2014 [184]. From an innovation perspective, startups often pave the way for the introduction of even more new and disruptive innovations [158]. Kickstarter is changing the retail and finance industries, Spotify is offering a new way to listen to and purchase music, and Airbnb is reinventing the hospitality industry [153]. From an engineering perspective, startups must inventively apply existing knowledge in order to open up unexpected avenues for improvement [68]; e.g., they must provide education for full stack engineers, develop techniques for continuous lightweight requirements engineering, or develop strategies to control technical debt.

Despite these promising conditions, software startups face challenges to survival, even in contexts where they play a key role in developing new technology and markets, such as cloud computing [167]. These challenges may arise because, while developing a product can be easy, selling it can be quite

[1] ISO 42010:2011 [2] defines software-intensive systems as "any system where software contributes essential influences to the design, construction, deployment, and evolution of the system as a whole" to encompass "individual applications, systems in the traditional sense, subsystems, systems of systems, product lines, product families, whole enterprises, and other aggregations of interest".

[2] In our article, digital startups refer specifically to startups in which the business value of the solution is created by means of software [131].

difficult [166]. Software startups face other challenges, such as developing cutting-edge products, acquiring paying customers, and building entrepreneurial teams [67]. Such diverse factors underscore the need to conduct research on software startups, which will benefit both scholarly communities and startup leaders.

This paper's research agenda is driven by past and current work on software startups. We outline the various research tracks to provide a snapshot of ongoing work and to preview future research, creating a platform for identifying collaborations with both research and startup environments and ecosystems. This effort is not a one-way path. We have therefore founded a research network, the Software Startup Research Network (SSRN)[3], which enables interactions and collaborations among researchers and interested startups. SSRN envisions to: (1) spread novel research findings in the context of software startups; and (2) inform entrepreneurs with necessary knowledge, tools and methods that minimize threats and maximize opportunities for success. As part of the network initiatives, an International Workshop of Software Startups was established in 2015. The first edition of the workshop was held in Bolzano[4] (Italy) in 2015, and the second took place in Trondheim[5] (Norway) in 2016. This paper provides a research agenda based on the activities carried out by the researchers in the network.

The rest of the paper is organized as follows. After we clarify the meaning of *software startup* and what we know about software startups from prior research in the Background section, Section 3 introduces the research topics on software startups, organized under six main tracks that we have either investigated or envision investigating in the future. Wherever possible, each topic is illustrated and motivated by previous studies. Section 4 highlights the implications of these main tracks for future research. The paper concludes with Section 5, which points out future actions that can establish and consolidate software startups as a research area.

2. Background

2.1. What is a Software Startup?

To understand software startups, we must first clarify what a startup is. According to Ries [148], a startup is a human institution designed to create a new product/service under conditions of extreme uncertainty. Similarly, Blank [10] describes a startup as a temporary organization that creates high-tech innovative products and has no prior operating history. These definitions distinguish startups from established organizations that have more resources and already command a mature market. In addition, Blank [10, 12] defines a startup as a temporary organization that seeks a scalable, repeatable, and profitable business model, and therefore aims to grow. Blank's definition highlights the difference between a startup and a small business, which does not necessarily intend to grow, and consequently lacks a scalable business model.

Even though sharing common characteristics with other types of startups, such as resource scarcity and a lack of operational history, software startups are often caught up in the wave of technological change frequently happening in software industry, such as new computing and network technologies, and an increasing variety of computing devices. They also need to use cutting-edge tools and techniques to develop innovative software products and services [161]. All these make software startups challenging endeavours and meanwhile fascinating research phenomena for software engineering researchers and those from related disciplines.

[3] https://softwarestartups.org
[4] http://ssu2015.inf.unibz.it/
[5] https://iwssublog.wordpress.com/

In 1994, Carmel first introduced the term *software startup*, or, to be more precise, *software package startup*, in SE literature [22]. Carmel [22] argued that software was increasingly becoming a fully realized product. Since then, other researchers have offered their own definitions of *software startup*. Sutton [161] considers software startups as organizations that are challenged by limited resources, immaturity, multiple influences, vibrant technologies, and turbulent markets. Hilmola et al. [79] claim that most software startups are product-oriented and develop cutting edge software products. Coleman and Connor [31] describe software startups as unique companies that develop software through various processes and without a prescriptive methodology.

Currently, there is no consensus on the definition of *software startup*, even though many share an understanding that software startups deal with uncertain conditions, grow quickly, develop innovative products, and aim for scalability. Different definitions emphasize distinct aspects, and consequently may have varying implications for how studies that adopt them should be designed, e.g., who qualifies as study subjects, or which factor is worth exploring. For this reason, despite the lack of a single agreed-upon definition of *software startup*, it is important and recommended that researchers provide an explicit characterization of the software startups they study in their work. The research track in Section 3.1.1 is dedicated to develop a software startup context model that would allow for such a characterization.

2.2. What are the Major Challenges of Software Startups?

Software startups are challenging endeavours, due to their nature as newly created companies operating in uncertain markets and working with cutting edge technology. Giardino et al. [69] highlight software startups' main challenges as: their lack of resources, that they are highly reactive, that they are by definition a new company, that they are comprised of small teams with little experience, their reliance on a single product and innovation, and their conditions of uncertainty, rapid evolution, time pressure, third-party dependency, high risk, and dependency (they are not self-sustained). Further, Giardino et al. [67] apply the MacMillan et al. [122] framework in the software startup context, categorizing the key challenges faced by early stage software startups into four holistic dimensions: product, finance, market, and team. The findings of Giardino et al. [67] reveal that thriving in technological uncertainty and acquiring the first paying customer are the top key challenges faced by many startups. In another study, Giardino et al. [70] discover that inconsistency between managerial strategies and execution could lead to startup failure.

Although research exists on the challenges software startups face, there is no study dedicated to their success factors. Block and Macmillan's [14] study highlights the success factors for any new business, including generating ideas to complete product testing, completing a prototype, and consistently re-designing or making amendments. Researchers have yet to explore these general factors' applicability to the specific software startup context.

2.3. What do We Know about Software Engineering in Software Startups?

Software development comprises a software startup's core activity. However, some initial research studies report a lack of software engineering activities in software startups. A systematic mapping study conducted by Paternoster et al. [142] allows us to start understanding how software startups perform software development. The study reveals that software requirements are often market driven and are not very well documented. Software development practices are only partially adopted; instead, pair programming and code refactoring sessions supported by ad-hoc code metrics are common practices. Testing is sometimes outsourced or conducted through customer acceptance and focus groups, and team members are empowered and encouraged to adapt to several

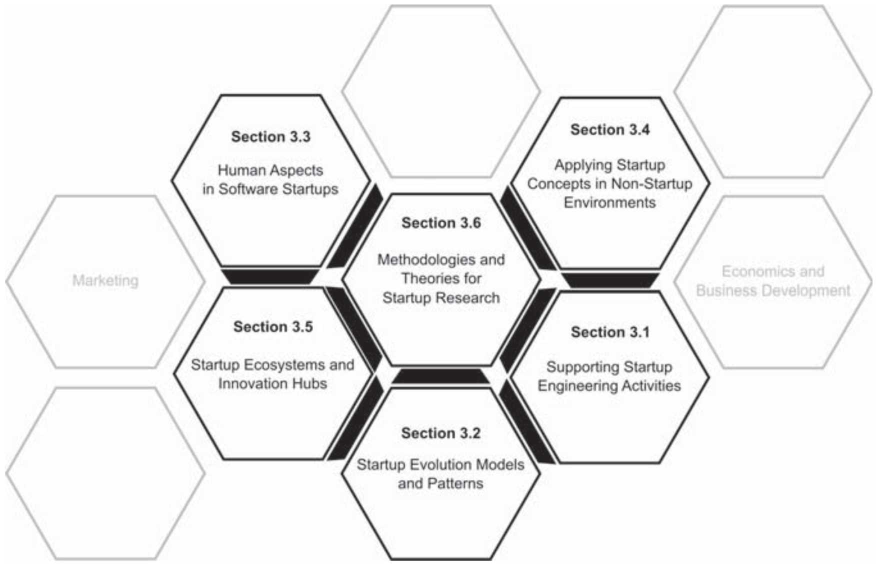

Figure 1. Overview of the Software Startup Research Agenda

roles. Similarly, Giardino et al. [69] highlight the most common development practices that have been used in software startup companies, such as: using well-known frameworks to quickly change the product according to market needs, evolutionary prototyping and experimenting via existing components, ongoing customer acceptance through early adopters' focus groups, continuous value delivery, focusing on core functionalities that engage paying customers, empowerment of teams to influence final outcomes, employing metrics to quickly learn from consumers' feedback and demand, and engaging easy-to-implement tools to facilitate product development.

Although a few studies provide snapshots of software engineering practices in software startups [106, 68], the state of the art presented in literature is not enough to base an understanding of how software engineering practices could help software startups. Researchers must build a more comprehensive, empirical knowledge base in order to support forthcoming software startups. The research agenda presented in this paper intends to inspire and facilitate researchers interested in software startup related topics to start building such knowledge base.

3. Research Agenda

The Software Startup Research Agenda, initialized in June 2015, was developed by a network of researchers interested in studying the startup phenomenon from different angles and perspectives. This variety of research interests not only opens up new avenues for collaboration, but also sheds light on the complexity of the studied phenomenon. Initially, ten researchers created a mind map of different research areas, aiming to provide an overview of software startup research areas and how they connect to each other. Over a period of six months, more researchers joined the network, added their research tracks, and continuously expanded the map. A working session with twenty researchers at the 1st workshop on software startup research in December 2015 was devoted at discussing the identified areas and finding potential interest overlaps among the participants. After this meeting, the authors of this paper prepared eighteen research track descriptions according to the following pattern: background of the area, motivation and relevance for software engineering in startups, research questions, potential impact of answering these research questions on practice and research, potential research methodologies that can be employed to answer the proposed research questions, and related past or ongoing work. Most of the authors interacted in the past or are currently active as advisory board members, mentors, founders or team members of software startups.

The leading authors of this paper grouped the eighteen research tracks into six major clusters, based on the thematic similarities and differences of the tracks. While this grouping is one of the several possible ways to create the clusters, it served the purpose to ease the presentation and discussion of the research agenda, shown in Figure 1. Supporting Startup Engineering Activities (Section 3.1) encompasses research foci that address specific software engineering challenges encountered by startup companies. Startup Evolution Models and Patterns (Section 3.2) focuses on the progression of startups over time, trying to understand the underlying mechanics that drive a company towards success or failure. Human Aspects in Software Startups (Section 3.3) covers research tracks that investigate factors related to the actors involved in startups. The research on Applying Startup Concepts in Non-Startup Environments (Section 3.4) seeks to strengthen innovation by extracting successful software startup practices and integrating them in traditional environments. Startup Ecosystems and Innovation Hubs (Section 3.5), on the other hand, investigates whether and how a thriving environment for software startups can be designed. Finally, all of these areas are connected by research tracks that develop methodologies and theories for software startup research (Section 3.6).

Figure 1's illustration of the research agenda includes reference to research areas outside this paper's current scope. Marketing and Business and Economic Development are directions that are likely relevant for the performance of software startups. These and other areas may be added to the research agenda in later editions when more evidence exists regarding whether and how they interact with software startup engineering, i.e. the "use of scientific, engineering, managerial and systematic approaches with the aim of successfully developing software systems in startup companies" [68].

3.1. Supporting Startup Engineering Activities

The research tracks in this cluster share the theme of studying, identifying, transferring, and evaluating processes, methods, framework, models, and tools aimed at supporting software startup engineering activities.

3.1.1. The context of software intensive product engineering in startups

Rapid development technologies have enabled small companies to quickly build and launch software-intensive products with few resources. Many of these attempts fail due to market conditions, team breakup, depletion of resources, or a bad product idea. However, the role of software engineering practices in startups and their impact on product success has not yet been explored in depth. Inadequacies in applying engineering practices could be a significant contributing factor to startup failure.

Studies show that startups use ad-hoc engineering practices or attempt to adopt practices from agile approaches [187, 105]. However, such practices often focus on issues present in larger companies and neglect startup-specific challenges. For example, Yau and Murphy [187] report that test-driven development and pair programming provide increased software quality at an expense of cost and time. Also keeping to a strict backlog may hinder innovation. Since neglecting engineering challenges can lead to sub-optimal product quality and generate waste, engineering practices specific to the startup context are needed. The overarching questions in this research track are:

- RQ1: To what degree is the actual engineering a critical success factor for startups?
- RQ2: How can the startup context be defined such that informed decisions on engineering choices can be made?
- RQ3: What engineering practices, processes and methods/models are used today, and do they work in a startup context?

An answer to RQ1 could help practitioners to decide on what activities to focus on and prioritize allocation of resources. Several studies, e.g Paternoster et al. [142], Giardino et al. [67] and Sutton [161], emphasize the differences between established companies and startups, noting that startups are defined by limited resources and dynamic technologies. However, these characterizations are not granular enough to support a comparison of engineering contexts in different companies, making the transfer of practices from company to company difficult [144]. Thus, understanding the engineering context of startups (RQ2) is an important milestone in developing startup context specific engineering practices (RQ3). While there exists work that provides systematic context classifications for the field of software engineering in general [144, 27, 47, 101, 100], these models are not validated and adapted for use within startups. The work in this research track aims to develop such a software startup context model by analysing data from startup experience reports [106]. Provided that engineering contexts among startups and established companies can be compared at a fine level of detail, the context model can be used to identify candidate practices. Moreover, researchers can develop decision support by mapping specific challenges with useful practices, thereby validating the model and helping practitioners select a set engineering practices for their specific context and set of challenges.

3.1.2. Technical debt management

The software market changes rapidly. As discussed by Feng et al. [59], in fast changing environments, the product management focus evolves from the more traditional cost or quality orientation to a time orientation. New product development speed is increasingly important for organizations, and a commonly shared belief is that time-to-market of new products can build a competitive advantage [59]. In the software startup context, it may be vital to be the first to market in order to obtain customers. Since software startups also lack resources, quality assurance is often largely absent [142]. However, long-term problems will only be relevant if the product obtains customers in the short term [168]. This short-term vision may produce software code that is low-quality and difficult to change, compelling the company to invest all of its efforts into keeping the system running, rather than increasing its value by adding new capabilities [168]. Scaling-up the system may become an obstacle, which will prevent the company from gaining new customers. Finding a viable trade-off between time-to-market demands and evolution needs is thus vital for software startups.

One promising approach to performing such a trade-off is technical debt management. Technical debt management consists of identifying the sources of extra costs in software maintenance and analysing when it is profitable to invest effort into improving a software system [168]. Hence, technical debt management could assist startups in making decisions on when and what to focus effort on in product development. Technical debt management entails identifying the technical debt sources, the impact estimation of the problems detected, and the decision process on whether it is profitable to invest effort in solving the detected sources of technical debt [60, 116]. Only those sources of technical debt that provide return on investment should be resolved. More importantly, technical debt should be managed during project development [119] in order to control the internal quality of the developed software. Several research questions need to be answered to successfully manage technical debt in this way:

- RQ1: What kind of evolution problems are relevant in the software startup context? How can we identify them?
- RQ2: How can we prioritize the possible improvements/changes in the context of software startups?
- RQ3. What factors beyond time-to-market and resource availability must be considered in trade-offs?
- RQ4: How can we make decisions about when to implement the improvements/changes within the software startup roadmap?
- RQ5: How can we provide agility to technical debt management, necessary in an environment plenty of uncertainty and changes?

Answering these questions will impact on both practitioners and researchers focused on software startups. Practitioners will be able to make better decisions considering the characteristics of the current software product implementation. The current implementation could make it impossible to reach a deadline (time to market), because of the complexity of the changes to perform to implement a new feature, assuming a given amount (and qualifications) of effort to be deployed. Furthermore, it will be also possible to decide between two alternative implementations, with different costs, but also with different potential for the future, assuming that the "future" has been previously outlined. For researchers, answering these questions could help clarify the role of design decisions in software development in the context of a software product roadmap, similarly to what happens in other engineering disciplines.

Technical debt is context dependent since quality tradeoffs are context dependent [154]. While technical debt is as important to software startups as it is to mature companies, the kind of decisions to take and the consequences of making the wrong decisions are not the same, justifying research on technical debt specifically in software startups.

In general, there is a lack of specific studies on technical debt management in software startups, and current literature reviews on technical debt management do not address this topic [60, 116]. Moreover, there are several specific challenges to managing technical debt that are of special relevance for software startups. For one, very few studies address how to prioritize improvements to solve technical debt problems, especially for commercial software development [116]. In addition, technical debt management literature often refers to time-to-market, but very few studies actually address it [60], perhaps because it is a topic that straddles engineering and economics.

3.1.3. Software product innovation assessment

Startup companies strive to create innovative products. For firms in general, and software startups in particular, it is critical to know as soon as possible if a product aligns with the market, or whether they can increase their chances to lead the market and recruit the highest possible number of customers [91].

The need to invest in infrastructures to measure the impact of innovation in software was highlighted by OECD [120], and more recently by Edison et al. [50]. These measures will enable companies to assess the impact of innovation factors and achieve the expected business goals, as well as to improve the understanding of success yield high returns on investments in the innovation process [120]. Product innovation assessment is thus very relevant for product developers, and especially for startups, which are more sensitive to market reactions. Product innovation assessment is complex, particularly for software products [58].

Product innovation assessment is reported in literature as the combination of a number of multi-dimensional factors impacting the success or failure of a software product [38]. Factor's measures intend to engage people in the innovation process to think more deeply about factors affecting product innovation. Factors such as time-to-market, perceived value, technology route, incremental product, product liability, risk distribution, competitive environment, life cycle of product, or strength of market could be grouped into dimensions like market, organization, environment, or any other terms of impact on the market and business drivers [5]. These factors can act as innovation enablers or blockers [35].

Since these factors are not always independent, it is critical to identify the existing dependencies and gain a better understanding of each factor's impact. It would be necessary to relate these factors to characteristics specific to software products, such as, but not limited to, software quality attributes proposed by ISO/IEC [1].

There is a lack of specific literature on *software* product innovation assessment; most of the past research refers to products in general, and not specifically to software products [186, 50], leading to the following research questions:

- RQ1: What should be the components of a software product innovation assessment/estimation model?
- RQ2: What factors can help measure innovation from a software product and a market perspective?
- RQ3: To what extent are factors that can help measure innovation dependent on the software product and the market perspective?
- RQ4: What is the relation between software product innovation factors and quality factors?
- RQ5: What kind of tools for software product innovation estimation could support software startups in decision making?

While innovation has been widely studied from the process perspective, the product perspective, by nature, has been addressed mainly from the viewpoint of specific products and industries. However, software products are different compared to other kinds of products [145] and innovations in the software industry happen fast. Hence, answers to RQ1-RQ4 would provide a fundamental understanding on software product innovation assessment and be beneficial for both researchers and practitioners. Software startups need to be fast and spend resources in an efficient way. Therefore, to be able to estimate existing products or design new products, considering those characteristics that experience shows that are relevant from an innovation point of view, can be essential for software startups to develop successful products (RQ5).

3.1.4. Empirical prototype engineering

Startups often start with a prototype, which serves as a form to validate either a new technology or knowledge about targeted customers [142]. Traditionally, prototyping implies a quick and economic approach to determining final products [118, 7, 156]. Defined as a concrete representation of part or all of an interactive system, prototypes has been intensively researched and used in Software Engineering, with well-developed taxonomies, such as horizontal and vertical, low-fidelity and high-fidelity prototypes [156]. The strategy of developing a prototype can greatly vary due to a great variety of prototype types, their development efforts and value they can produce.

While much about prototyping techniques can be learnt from the SE body of knowledge, the discussion about prototyping in the context of business development process is rare. Recent work on startup methodologies, such as Lean Startup [148] and Design Thinking [19] emphasizes the adoption of prototypes to increase chances of success through validated learning. Alternatively, startup prototypes need to be developed to satisfactorily serve their purposes, i.e. technical feasibility test, demonstration to early customers, and fund raising. We argue that the prevalent

Software Engineering practices used by startups to develop their first product inefficiently integrate into startups' dynamic contexts. Hence we call for research in understanding the development and usage of prototypes in startup contexts:

- RQ1: How can prototyping be used to maximize learning experience?
- RQ2: How can prototyping be used for optimization?
- RQ3: How can prototyping be used to support communication with external stakeholders?
- RQ4: How do prototypes evolve under the multiple influences of startups' stakeholders?

Early stage startups are lacking actionable guidelines for making effective prototypes that can serve multiple purposes. We believe that many startups will economically and strategically benefit by having proper practices in prototyping, such as technology evaluation (RQ1), strategic planning (RQ2) and customer involvement (RQ3).

To understand prototype development and its usage in startups, i.e. answering the first three research questions, exploratory case studies can be conducted. Cases would be selected to cover different types of startup prototypes at different phase of startup progress. A large-scale survey can be used to understand the prototype usage patterns, i.e. answering RQ4.

Despite an increasing body of knowledge on software startups [142], empirical research on prototyping processes and practices are rare. A few studies have investigated the adoption of software prototypes in combination with Design Thinking [53] and proposed prototyping techniques [136, 53, 73]. However, these studies rely on a very limited number of cases. Moreover, different constraints on prototyping decisions are often neglected. Future work can address antecedence factors, i.e. the involvement of lead-users, available human resources, and technological push, and how they impact prototyping strategies and usages in different startup contexts [137].

3.1.5. Risk Management Tools for Software Startups

The management of risk, namely the risk of failing to meet one's goals within given constraints in budget and/or time, is of paramount importance in every human activity. In the context of software startups, risk management looks unconventional, because startups naturally involve a much higher risk than traditional businesses. Yet, perhaps even more so than in traditional contexts, evaluating and managing risk in the software startup context might be a key factor for success.

Risk factors can be identified as a check-list of the incidents or challenges to face. Each of them could be categorized and prioritized according to its probability and the impact level of its consequences. This research track aims to study, model, and quantify various aspects related to risk management in software startups, with the goal of providing tools, based on process simulation, that control risk. Being able to efficiently model and simulate the startup process and its dynamics, would support startups in timely decision making. While numerous other approaches to risk control exist [146], we have found in our previous work [30, 34] that process simulations can be effective in risk management. Therefore, the overarching questions in this research track are:

- RQ1: To what extent do software startups explicitly manage risk?
- RQ2: To what degree is it feasible to model software development processes in startups?
- RQ3: To what extent can these models be used to quantify the risk of exceeding project budget or time?
- RQ4: What systematic ways exist to understand when to pivot or persevere [148], and what might be the cost of a wrong or untimely decision?

Following our previous experiences in software process modelling and simulation, to gain a better understanding is necessary to identify and analyse significant activities, not limited to the software development phase, of a software startup (RQ1). This is necessary to be able to identify the critical aspects of startup development risks that are suitable for simulation. In our previous work we studied the application of Event-Driven models and/or System Dynamics to the software development processes. From this work we know that it is possible analyse project variations in time and budget with a Monte Carlo approach, by performing several simulations of the same project, varying the unknown parameters according to given distributions, and calculating the resulting distributions of cost and time of the simulated projects. Such analysis allows one to compute the Value At Risk (VAR) of these quantities, at given VAR levels. While Cocco et al. [30] and Concas et al. [34] provide exemplar studies of the application of these techniques in mature (agile) software development contexts, the question is whether such an approach is suitable and beneficial for software startups, and under what conditions (RQ2). By simulating the evolution of a startup as a process, we might be able to make predictions on its future development. Such predictions, or a result that can be rapidly be drawn from simulations, might be crucial for startups to understand which decisions are less costly and/or risky (RQ3). This is particularly true for decisions related to fields such as market strategies, team management, financial issues or product development (RQ4).

3.1.6. Startup support tools

Support tools can help software startups get their business off the ground with less pain and more guidance. These tools generally embed crucial knowledge regarding startup processes and activities. A plethora of tools (mostly software tools) exist for meeting the different needs of entrepreneurs and supporting various startup activities. For example, the web-page[6] by Steve Blank, a renowned entrepreneurship educator, author, and researcher from Stanford University, contains a list of more than 1000 tools. Well-designed portals such as Startupstash.com ease access to these supporting tools.

However, due to the lack of time, resources, and/or necessary knowledge, entrepreneurs cannot easily find the tools that best suit their needs, or cannot effectively utilize these tools to their potential. Existing studies provide limited insights on how entrepreneurial teams could find, use and benefit from support tools. Hence, the overarching questions in this research track are:

- RQ1: What are the needs of software startups that can be supported by software tools?
- RQ2: What are the tools that support different startup activities?
- RQ3: How can support tools be evaluated with respect to their efficiency, effectiveness, and return-on-investment?
- RQ4: How can support tools be effectively recommended to entrepreneurs and used by them?

RQ1 and RQ2 are targeted at identifying a match between the needs of software startups and the available tool support. To enable robust recommendations, both the individual startups and the software tools need to be objectively characterized allowing for their evaluation w.r.t. certain quality criteria (RQ3). There are potential synergies with the research track looking at the context characterization of software startups (Section 3.1.1). Answers to these research questions can be also valuable input for software tool vendors to develop the right tools that are needed by startups. In addition, the findings can be useful for future studies that develop proof-of-concept prototypes to support startup activities.

To investigate the proposed questions, various research methods can be applied, including survey of software startups regarding their needs and usage of support tools, in-depth case study of

[6] http://steveblank.com/tools-and-blogs-for-entrepreneurs/

adoption and use of support tools, and design science approach to develop recommender systems of support tools (RQ4).

Research on tooling aspects in the software startup context is scarce. Edison et al. [51] argue that, despite the fact that different startup supporting tools have been developed and published over the Internet, new entrepreneurs might not have sufficient knowledge of what tools they need when compared to experienced entrepreneurs. In addition, not all tools will help entrepreneurs in certain tasks or situations. Entrepreneurs' experiences using the tools can serve as the basis for evaluating and recommending appropriate tools. Besides suggesting a new categorization of existing startup support tools, Edison et al. [51] propose a new design of a tool portal that will incorporate new ways to recommend tools to entrepreneurs, especially to those who engage for the first time in a software startup endeavour.

3.1.7. Supporting software testing

Testing software is costly and often compromised in startups [189], as it is challenging for startups to fulfil customer needs on time, while simultaneously delivering a high quality product. In many software startups there is a common slogan that says "done is better than perfect", which indicates a general tendency toward a lack of testing and quality assurance activities [98]. However, it is sometimes also observed that startups do not know how and what to test; they lack expertise to test requirements as they do not have knowledge about their customers and users [98]. Therefore considering testing in software startups poses the following research questions:

- RQ1: To what extent does software testing in startup companies differ from traditional companies?
- RQ2: To what extent does testing evolve over time in software startup companies?
- RQ3: What is an optimal balance between cost/time spent on testing and development activities?
- RQ4: How can a software startup leverage customers/users for testing?

Answering RQ1 would provide insights on the aspects that differentiate the software testing process in startups from mature companies. For example, integration testing is likely very important for startups due to the fast paced product development. At the same time however, startups tend to work with cutting edge technologies, requiring a robust and flexible test integration platform. Connected to this is the question whether testing needs change over time, while the software startup matures. Answers to RQ2 and RQ3 would be particularly valuable for practitioners who could then better allocate resources. Users of software could be used for different testing purposes. On one hand, users provide valuable feedback in testing assumptions on customers needs. On the other hand, early adopters that are more robust towards deficiencies can help to improve product quality before targeting a larger market. Answers to RQ4 would provide strategies to harvest these resources.

In order to answer these research questions, various empirical research methods could be utilized. The studies would be devised in a way that "contrasting results but for anticipatable reasons" could be expected [188], i.e. different software startup companies would be taken into account to acquire a broad view of testing in software startups.

To the best of our knowledge, software testing in software startups has been scarcely researched. Paternoster et al. [142] highlighted the quality assurance activities in software startups in their mapping study. They found that it is important to provide software startups effective and efficient testing strategies to develop, execute, and maintain tests. In addition, they highlighted the importance of more research to develop practical, commercial testing solutions for startups.

3.1.8. User experience

User experience (UX) is described as "a person's perceptions and responses that result from the use or anticipated use of a product, system or service" [87]. Good UX can be seen as providing value to users, as well as creating a competitive advantage. UX is important for software startups from their earliest stages. Firstly, human-centred design methods such as user research and user testing can help startups better understand how they can provide value to users and customers, as well as what features and qualities need testing for users to be satisfied with their product. Combined with business strategy, this human-centred approach helps startups move towards successful, sustainable business creation. Secondly, providing an initially strong UX in the first product versions can create positive word of mouth [64], as well as keep users interested in the product for a longer time [84]. Genuine interest from users for the product idea while the product is still a prototype helps gain meaningful feedback [84]. Compared to more established businesses, software startups may pivot resulting in new target markets and user groups. This means efforts put into designing UX need to be faster and less resource consuming. Furthermore, failing to deliver satisfying UX can be fatal to small startups that can not cover the costs of redesigning. The overarching questions in this research track are:

- RQ1: What useful methods and practices exist for creating UX in startups?
- RQ2: What is UX's role during different phases of a startup's life-cycle?
- RQ3: To what extent are UX and business models connected in customer value creation?

An answer to RQ1 can provide software startups methods for developing strong UX in the first product versions which can keep users interested in the product for a longer time [84]. Genuine interest from users for the product idea while the product is still a prototype helps to gain meaningful feedback [84]. For business creation, understanding the value of UX for startups (RQ2) helps assigning enough resources for creation of UX while not wasting resources where there is no value to be gained (RQ3).

Research on startups and UX has been very limited. Some case studies report UX's role in building successful startups [126, 162]. Practices and methods for UX work in startups have been reported in [84, 81, 83]. A framework for creating strong early UX was presented by Hokkanen et al. [82]. These provide some results on feasible and beneficial UX development in startups, but more generalizable results are needed.

3.2. Startup Evolution Models and Patterns

The research tracks in this cluster share the theme of studying, identifying, and differentiating the transformation of startups in different stages. This also includes studies about different business and technical decision-making practices.

3.2.1. Pivots in software startups

It is very difficult for software startups to understand from start what are the real problems to solve and what are the right software solutions and suitable business models. This is evidenced by the fact that many successful software startups are different from what they started with. For example, Flickr, a popular online photo sharing web application, originally was a multiplayer online role playing game [133]. Twitter, a famous microblogging application, was born from a failed attempt to offer personal podcast service [133].

Due to their dynamic nature, software startups must constantly make crucial decisions on whether to change directions or stay on the chosen course. These decisions are known as *pivot* or *persevere* in the terms of Lean Startup [148]. A pivot is a strategic decision used to test fundamental hypothesis about a product, market, or the engine of growth [148]. Software startups develop technology intensive products in nature. Due to this, these are more prone to the rapidly changing technology causing pivots. Similarly, certain types of pivots are more relevant to software startups e.g. zoom in pivot: a pivot where one feature of a product become the whole product as in the case of Flickr. Pivot is closely linked to validated learning, another key concept from Lean Startup. The process to test a business hypothesis and measure it to validate its effect is called validated learning [148], whereas pivot is often the outcome of validated learning. A recent study [70] reveals that startups often neglect the validated learning process, and neglect pivoting when they need to, which leads to failure. This shows the importance of pivoting for a startup to survive, grow, and eventually attain a sustainable business model. In order to better understand and explore the pivoting process in the software startup context, the following fundamental research questions can be formed:

- RQ1: To what extent is pivoting crucial for software startups?
- RQ2: How do software startups pivot during the entrepreneurial/startup process?
- RQ3: What are the existing process/strategies/methods to make a pivoting decision in a startup context?
- RQ4: How do pivots occur during different product development and customer development life cycles?

Answering RQ1-RQ2 is necessary to understand pivoting in the context of software startups, building a fundamental framework on reasons for pivoting and their types. RQ3-RQ4, on the other hand, are targeted at understanding pivoting decisions and mechanisms. The overall contribution of answering the stated research questions has implications for both researchers and practitioners. The answers would provide an empirically validated conceptual and theoretical basis for the researchers to conduct further studies regarding the pivot phenomenon. For the practitioners, it would help them to make informed decision regarding when and how to pivot in order to increase the chances of success.

Due to the nascent nature of software startup research area, exploratory cases studies is a suitable approach to answer the research questions. Followed by the case studies, quantitative surveys can also be conducted to further generalize the results regarding pivoting in software startups.

Recently, there were some studies conducted on pivots in software startups. A study by Van der Van and Bosch [175] compares pivoting decisions with software architecture decisions. Another study by Terho et al. [164] describes how different types of pivots may change business hypothesis on lean canvass model. However, these studies lack the sufficient detail to understand different types of pivots and the factors triggering pivots. A study by Bajwa et al. [152], presents an initial understanding of different types of pivots occurred at different software development stages, however it lacks the deeper understanding of the pivoting decision that can only be achieved by a longitudinal study.

3.2.2. Determination of Software Startup Survival Capability through Business Plans

Software startups are highly specialized from a technological point of view. Focusing on the economic exploitation of technological innovations [121], they belong to the group of new technology-based firms. Literature suggests that one of their major challenges is the transformation of technological know-how into marketable products [65, 18]. New technology-based firms often struggle with unlocking the product-market fit [125] and commercializing their technological products [65]. Applying a resource-based view does thus not suffice for explaining survival and growth of software startups [107, 115]: a crucial success factor is the ability of new technology-based firms to understand and interact with the market environment to position their products accordingly [71, 28].

Particularly in early lifecycle stages, new technology-based firms need to build network relations with the market. Network theory literature suggests that with increasing network maturity, the chances for survival and growth increase [135, 151, 183]. The ability to transform resources in response to triggers resulting from market interactions can be described as a dynamic capability [54, 163, 134, 117] which helps software startups commercialize their products. This transformation process captures the evolution of new technology-based firms in their early-stages. Current research is based on the construct of "venture emergence", which provides a perspective on the evolutionary change process of new technology-based firms [71, 21]. Venture emergence reflects the interaction process with agents and their environments [96]. Business plans of new technology-based firms are used as the artefact for measuring the status of venture emergence. They contain descriptions of transaction relations [85, 102, 95] new technology-based firms build in four market dimensions: customer, partner, investor, and human resources [110]. This research track intends to answer a number of research questions:

- RQ1: How reliably can annotated transaction relations from business plan texts determine the venture emergence status of technology-based startups?
- RQ2: To what extent are the number and strength ("level") of identified transaction relationships useful as an indicator of survival capability?
- RQ3: How can patterns of transaction relations be used as an indicator for evaluating strengths and weaknesses of new technology-based firms, and thus be used to more effectively direct support measures?

While it is possible to measure the venture emergence status even in a software startup's very early stages, the predictive strength of transaction relations needs to be evaluated (RQ1-RQ2). This use of network theory to operationalize the venture emergence construct is a new approach, which adds to network theory literature in the context of the survival of new technology-based firms. It further confirms the business plans of new technology-based firms as a valuable source of information on startup potential. Finally, the resource-based approach to explain venture survival is enriched by applying a process-oriented perspective: we analyse resource transformation, rather than only looking at the initial resource configuration (RQ3). Furthermore, the research can contribute to the effectiveness of the innovation system by investigating indicators that reveal strengths and weaknesses of new technology-based firms. These can be used to direct support measures to software startups more effectively.

To answer the stated research questions, one can use content analysis [46, 56], combining human and computer-based coding of business plans, to determine the number and strength of transaction relations [111, 172].

Initial statistical tests that have been performed on a sample of 40 business plans of new technology-based firms confirm the relationship between the status of venture emergence of new technology-based firms and venture survival [172]. Earlier work led to the development of the concept for analysing early-stage startup networks and the relevance for survival [110]. Based on this concept, a coding method for transaction relations in business plans has been developed and validated with 120 business plans [111].

3.3. Cooperative and Human Aspects in Software Startups

The research tracks in this cluster address challenges and practices related to how people cooperate and work is software startups.

3.3.1. Competencies and competency needs in software startups

Software startups set different competency requirements on their personnel than more established companies. The biggest differences occur in two phases of the evolution of startups which have an impact on the nature of software development and competence needs: (1) in the early stages of rapid software development when there is a lack of resources and immature competencies in many key areas, and (2) when the rapid business growth of successful startups requires management of a fast growing personnel and amount of software with limited management resources and competencies. In the early phases strong competition requires the software startup to innovate and react quickly [142], and deployment of systematic software engineering processes is many times replaced by light-weight ad-hoc processes and methods [105, 142]. The nature of software makes it possible for successful startups to scale fast [142]. Rapid software-driven growth requires fast scaling of the software production, distribution, and maintenance. The required competences also quickly evolve when software development moves from rapid greenfield prototyping to professional software development and management. Mastering this demanding situation often requires a broad prior skill basis from the startup team, including an ability to adjust to changes, and learn quickly.

Research on specific skills and competency needs in software startups broadens not only the knowledge on software startups themselves, but also broadens the knowledge on software engineering conducted under the challenging circumstances of startups. Focusing the research on the early stages and on the growth period of the software startups, when the challenges of the software startups are the greatest [67, 70], brings the most valuable knowledge to both academia and practitioners. Competency research also brings human factors into focus [124, 88], and reinforces the results of existing software startup research towards a more comprehensive modelling and understanding. The research questions for studies on competencies and competency needs in software startups include:

- RQ1: Software startup challenges and competency needs — what software development knowledge and skills are needed to overcome the challenges?
- RQ2: What are the competency needs specific for software startups compared to the more established software companies?
- RQ3: How do the competency needs change over the evolution of software startups?
- RQ4: How do the competency needs map onto the roles and responsibilities of the startup teams in software startups?
- RQ5: How can the growth of software startups be managed in terms of competency needs for software development practices, processes and recruitment?

Research on software startups, including research on competency needs, provides the research and development of software engineering with new knowledge and viewpoints on how to direct the work in order to best address the specific challenges of the software startups (RQ1). In particular, differences to mature software companies are interesting to study (RQ2) considering software startups evolve, if they survive, to established companies. Knowing how competency needs change might turn out as one key factor for this transition (RQ3). Theoretical models describing the evolution paths of software startups have been created [148, 16], but competency needs and how they map to roles and responsibilities have been to a large degree ignored (RQ4). Similarly, while software development work [142] and software engineering practices [105] have also been studied, it is unclear how competency needs can be managed in growing software startups (RQ5).

3.3.2. Teamwork in software startups

The importance of human aspects in software development is increasingly recognized by software engineering researchers and practitioners. Teamwork effectiveness is crucial for the successes of any product development project [45]. A common definition of a team is "a small number of people with complementary skills who are committed to a common purpose, set of performance goals, and approach for which they hold themselves mutually accountable" [97]. A startup team is special in the wide range of variety, including both technicians and entrepreneurs.

While an innovative idea is important for the formation of a startup, startup success or failure ultimately rests on the ability of the team to execute. Entrepreneurship research showed that over 80 percent of startups that survive longer than two years were founded by a group of two or more individuals [140]. The dynamic and intertwined startups activities require the close collaboration not only among startup team members, but also with external stakeholders, such as mentors and investors. Given the diversity in mindsets and skill sets among founders, it is essential that they can work well together along with the startup life-cycle. The movement with recent methodology in Lean startup introduces an opportunity to look at startup teams from various angles, i.e. pivoting, startup culture, team formation, and decision-making. The overarching questions in this research track are:

- RQ1: Is there a common cultural / organizational / team characteristic among successful software startups?
- RQ2: How can a software startup team effectively communicate with other stakeholders, i.e. mentors and investors?
- RQ3: How can a software startup manage team internal relationships?
- RQ4: What are the common patterns of competence growth among software startup teams?

Understanding software startup team behaviour to internal and external environments and relating them to startup success measures would help to identify characteristics and teamwork patterns of successful startups. Answering RQ1 would provide practitioners some guidance on how to form startup teams while answers to RQ2-RQ3 would provide an understanding how internal end external team dynamics work and can improved. An answer to RQ4 would also support the work in Section 3.3.1, looking however specifically at competence growth patterns that could be valuable for practitioners when deciding on what to focus on in competence development. Empirical studies, i.e. case studies, surveys and action research are all suitable to investigate the stated research questions. Among them, comparative case studies would be the first option to discover the difference in startup teamwork patterns.

There exists a large body of literature in business management, entrepreneurship, and small ventures about entrepreneurial teams' characteristics and their relationship to startup outcomes [140, 94, 62]. In Software Engineering, few empirical studies identified team factors in the failure of software startups. Giardino et al. found that building entrepreneurial teams is one of the key challenges for early-stage software startups from idea conceptualization to the first launch [67]. Crowne et al. described issues with founder teamwork, team commitment and skill shortages [39]. Ensley et al. investigated the relative influence of vertical versus shared leadership within new venture top management teams on the performance of startups [57]. Other team dimensions are explored in the business and engineering management domain in specific geographies. E.g., Oechslein analysed influencing variables on the relational capital dimension trust within IT startup companies in China [140]. How generalizable these influencing variables to other geographies is yet to be seen.

3.4. Applying Startup Concepts in Non-Startup Contexts

One of the Lean Startup principles claims that entrepreneurs are everywhere, and that entrepreneurial spirits and approaches may be applied in any size company, in any sector or industry [148]. On the other hand, established organizations face the challenge of innovation dilemma and inertia caused by the organization's stability and the maturity of markets [36]. Therefore, applying startup concepts in non-startup contexts seems an promising avenue for established organizations to improve their innovation potential.

3.4.1. Internal software startups in large software companies

The internal software startup concept has been promoted as a way to nurture product innovation in large companies. An internal software startup operates within the corporation and takes responsibility for everything from finding a business idea to developing a new product and introducing it to market [6]. Internal software startups can help established companies master the challenge of improving existing businesses, while simultaneously exploring new future business that sometimes can be very different from existing ones [78]. Usually, this involves a conflict of interest in terms of learning modes [89] or risk propensity [132], which can be prevented by establishing dual structures within the organization for implementing internal software startups [112]. Compared to the traditional R&D activities of larger companies, an internal software startup develops products or services faster [142] and with higher market orientation [114]. This helps established companies maintain their competitiveness in volatile markets [141].

Besides the fact that the successful implementation of internal software startups faces various barriers, such as cultural conflicts [66] or the fear of cannibalization of existing businesses [52], internal software startups can also benefit from being part of established companies. Shared resources, such as capital, human resources [24, 32], and the access to the corporates' internal and external network [44] are just some benefits.

Earlier research on analysing the results of startups' value creation cycle has taken place in the context of the evolution of the enterprise [37]. However, this occurs over too long of a time period to be useful for guiding software development. Measuring the cycle time of the software engineering process to the completion of a software feature is also insufficient. The Lean startup approach [148] has been commonly adopted to new business creation in software intensive ventures. They use the learning loop to discover the customer value and potential of the new product concept, as well as to find new means to produce software. Tyrväinen et al. [171] propose that measuring the cycle time from development to analysis of customer acceptance of the feature enables faster learning of market needs. In addition, receiving fast feedback from users makes changing the software easier for the programmers who have not yet forgotten the code. Relevant research questions regarding internal software startups can be formulated as follows:

- RQ1: How can Lean startup be adopted and adapted for software product innovation in large software companies?
- RQ2: What are the challenges and enablers of Lean startup in large software companies?
- RQ3: How should internal software startups be managed / lead?
- RQ4: What metrics can be used to evaluate software product innovation in internal startups?
- RQ5: To what extent do internal startups have a competitive advantage compared to independent startups (through shared resources, etc.)?

Lean startup approach gains more interest from scholars and academics as a new way to foster innovation since it helps to avoid building products that nobody wants [55]. Some evidence shows that mature software companies and startups differ in applying Lean startup approach [93]; e.g. mature firms start the cycle by collecting data from existing users and then generating a hypothesis based on that data, whereas software startups generate ideas and collect data from new users to validate the ideas. However, it seems that, to a large extent, the approach can be used both in startups and established enterprises. By answering RQ1-RQ3 we aim at defining structured guidelines on how to introduce Lean startup in large software companies, supporting practitioners, while answering RQ4-RQ5 would provide a motivation for this approach, allowing to compare effectiveness on a quantitative level.

Due to the complex nature of the research phenomenon and the intention to achieve an in-depth understanding of it, we consider multiple case studies [188] as a suitable research approach. The case organizations can be selected based on the following criteria: (1) the organization develops software in-house, (2) a dedicated team is responsible from ideation to commercialization of a new software, and (3) the software falls out of the current main product line. The unit of analysis in this study would be a development team.

Very few studies have investigated how the Lean startup [148] can leverage internal startups in large software companies to improve their competency and capabilities of product innovation. Initial steps have been taken and some of the results have been published to fill this observed gap (e.g. [52, 49]). Marijarvi et al. [123] report on Finnish large companies' experience in developing new software through internal startups. They also discuss the lifecycle phases of innovation work in large companies. The authors argue that different types of internal organization may take place in each stage of new product development. For example, problem/solution fit can be done in an internal startup or company subsidiary.

3.4.2. Lean Startup for project portfolio management and open innovation

Building on the challenges proposed in Section 3.4.1, we propose that Lean startup could also be applied within both (i) project portfolio management (PPM), to co-ordinate multiple startup initiatives within an organization, and (ii) open innovation, wherein internal startups involve multiple organizations, individuals, or even unknown participants. Both PPM and open innovation and their main challenges are briefly introduced below, followed by research questions that require investigation before Lean startup principles can be successfully applied in these new contexts.

Software engineering PPM describes the ongoing identification, selection, prioritization, and management of the complete set of an organization's software engineering projects, which share common resources in order to maximize returns to the organization and achieve strategic business objectives [127, 35, 13, 170]. Open innovation is defined as the use of "purposive inflows and outflows of knowledge to accelerate internal innovation and to expand the markets for external use of innovation, respectively" [26]. Popular examples of open innovation include open source software development, crowd-sourcing, and inner source.

Effective PPM is critical to achieving business value [75, 147], improving cost and time savings, and eliminating redundancies [99, 113]. Unfortunately, existing portfolio management practices, which are based on the effective completion of individual projects with only episodic portfolio level reviews [147], fail to manage either the dynamic nature of contemporary projects, or problems associated with portfolios comprising too many projects [109, 147]. Indeed, many portfolios report an unwillingness to cancel projects that no longer contribute to the achievement of strategy [147].

Open innovation (OI) presents numerous advantages for organizations, such as access to a requisite variety of experts, a prospective reduction in overall R&D spending, reduced time-to-market, improved software development processes, and the integration of the firm into new and collaborative value networks [26, 4, 176]. Nonetheless, adopting open innovation processes can be significantly challenging. For example, adopters often lack internal commitment, in addition to challenges associated with aligning innovation strategies to extend beyond the boundaries of the firm. Moreover, there are concerns regarding intellectual property and managing unknown contributors/contributions, as well as managing the higher costs and risks associated with managing both internal and external innovations [174, 180, 25]. The role of Lean startup principles in addressing these challenges in both PPM and OI is worthy of further research:

- RQ1: How can Lean start-up be implemented within a portfolio management or open innovation context?
- RQ2: How can Lean startup initiatives drive or accelerate open innovation?
- RQ3: What Lean startup concepts could be adapted to facilitate open innovation processes in an organization?
- RQ4: How can one ensure Lean startup initiatives conducted across multiple projects or organizations align with strategy?
- RQ5: How do you reconcile potential conflicts between portfolio / open innovation processes and Lean startup processes?
- RQ6: How do you achieve consensus in defining the minimum viable product (MVP) in networks comprised of multiple autonomous (and sometime anonymous) agents?

The successful application of Lean startup principles (RQ1-RQ3) has the potential to reduce the costs arising from the poor implementation of PPM and OI practices and increase the value achieved from these initiatives. However, because such approaches are often practice led, it is necessary for academic research to develop effective theory to underpin practice and provide empirical data to support, or refute claims of effectiveness (RQ4-RQ6). Rich human interactions are at the heart of software engineering PPM and open innovation. Accordingly, phenomena in these domains can be examined using interpretive, qualitative methods such as semi-structured interviews, case studies and ethnography.

While the principles of lean have been applied to PPM (e.g. [86, 42], there is little research looking at the application of Lean startup principles to PPM. Similarly, while there is interest in the application of Lean startup principles in open innovation contexts, to date, such applications have predominantly been driven by practice.

3.5. Software Startup Ecosystems and Innovation Hubs

Successful software startups do not live in isolation. Normally, they are inserted in a rich environment that includes a number of relevant players, such as entrepreneurs, developers, investors, scientists, as well as business and intellectual property consultants. To support these players, a number of support programs from the private and public sectors are required to provide funding, incubation, acceleration, training, networking, and consulting. All these elements combine into what scholars and practitioners have called Startup Ecosystems [108]. In our software startups research agenda, we focus on Software Startup Ecosystems (SSE) and the elements that are relevant for startups that have software as a key part of their products or services.

By studying how SSEs are created, their main characteristics, and how they can evolve, one can better understand the environments that favour, or not, the birth and development of successful software startups. Research in this field can provide, to the relevant stakeholders, the concrete actions (e.g., public policies, private activities) that will establish a fruitful and vibrant environment for the execution of high-growth innovative projects within nascent software companies. The main research questions that need to be answered are the following:

- RQ1: What are the key elements of a fruitful SSE?
- RQ2: Are there different types of SSEs, e.g. differentiated by size, technology sectors, country economy or other factors?
- RQ3: How do SSEs evolve over time?
- RQ4: How can one measure the output and qualities of an SSE?

By answering RQ1, researchers will provide a better understanding of the way how SSEs and innovation hubs work, instrumenting key stakeholders in taking actions to improve their ecosystems. By identifying what factors promote or hinder the development of successful startups within a certain SSE, policy makers will get support in decision making (RQ2). Entrepreneurs will also be able to better understand what are the environmental factors and forces that can help or hinder the success of their enterprises.

Researchers from Brazil, Israel, and the USA have developed a methodology to map a specific software startup ecosystem; this methodology has been applied to Israel [108], São Paulo [61] and New York [41]. Currently, with the help of dozens of experts worldwide, they are developing a maturity model for SSEs [108, 40], addressing RQ3 and RQ4. This maturity model needs further research and validation before it can be applied in real scenarios to help practitioners and policy makers.

The Global Startup Ecosystem Ranking [77] is crafted by a group of experts that have been proposing metrics to evaluate regional ecosystems around the world and compare them according to multiple criteria. Frenkel and Maital [63] have developed a methodology to map national innovation ecosystems and use this map to propose policies to promote improvement. Jayshree has studied the influence of environmental factors on entrepreneurial success [90]. Finally, Sternberg [160] researched the role of regional government support programs and the regional environment as success factors for startups.

3.6. Theory and Methodologies for Software Startup Research

The tracks in this cluster direct their research towards identifying means to better study and understand software startups.

3.6.1. Overview of the possible theoretical lenses for studying software startups

Theories are important in any scientific field, as they form the foundation to understand a contemporary phenomenon better. Theories provide answers to the “why” questions, and are therefore useful for explaining why certain events occur while others do not. Software startup research does not operate in a vacuum, but rather can borrow theories from both the software engineering and information systems fields, business and management literature, as well as from the fields of organizational and social sciences.

We have identified a few potential theories that can be meaningfully applied in the context of software startup companies. The proposed theories are the hunter-gatherer model [159], Cynefin model [155], Effectuation theory [150] and Boundary Spanning theory [182]. These theories are briefly outlined in this section.

Although 90% of human history was occupied by hunters and gatherers, who forged for wild plants and killed wild animal to survive, only recently was the hunter-gatherer model re-discovered by Steinert and Leifer [159] to explain how designers pursue their endeavours in search of the best design outcome. The model shows the changes in the design process, as well as subsequently in the design outcome. The model portrays a distinction between a hunter who aims to find an innovative idea, and a gatherer who aims to implement the idea. Both are needed to achieve concrete results. While hunting the idea through ambiguous spaces has a change-driven, analytical, and qualitative nature; gathering the idea across predetermined paths has a plan-oriented, manageable, and quantitative nature. The model has recently been applied in software startup research to explain startups' evolutionary paths [138].

Complexity theory has been used as a frame of reference, by analysing its implications on software design and development (e.g. Pelrine [143], Rikkilä et al. [149]). Software projects can be characterized as endeavours wherein a dynamic network of customers, software designers, developers, 3rd party partners, and external stakeholders interact and can be seen as a Complex Adaptive System (CAS). To reason about decision-making in different situations, Snowden et al. [155] proposed a sense-making framework for such systems. The model has five sub-domains and divides the world in two parts - ordered and unordered main domains. The ordered domain is the one in which cause-effect (CE) relationships are known (the Known domain), or at least knowable after analysis (the Complicate domain). In contrast, the unordered domain includes a complexity situation, wherein the CE relationship can only be perceived in retrospect, but not in advance (the Complex domain), and a chaotic situation, wherein behaviours are completely random, lacking any expected consequence when acted upon. Depending on the problem domain, suitable approaches include categorizing, analysing, probing or acting [155]. The Cynefin model provides a framework that can be used to analyse the decisions made by software startuppers in developing their products. Often they find themselves in the unordered domain, attempting to make sense out of the current situation and navigate to the ordered domain.

Effectuation theory is a simple model, rooted in entrepreneurship, of decision-making under uncertainty. The effectual thinking is in the opposite of causal reasoning which starts from desired ends to necessary means (top-down). Experienced entrepreneurs reason from means to ends (bottom-up), trying to work out meanings and goals based on the resources they have at hand. The theory is embodied by five principles: the bird-in-hand principle, the affordable loss principle, the crazy quilt principle, the lemonade principle, and the pilot-in-the-plane principle [150]. The effectuation theory can help to make better sense of entrepreneurs' decision-making process in the evolution of software startups, such as problem validation, value proposition definition, design of MVPs, and pivoting processes. Good practices could be discovered using the effectuation theory as a theoretical lens.

Startups operate in a dynamic environment and face expectations and influences from many directions. In order to survive, they need to effectively collaborate within their team, but also outside it. Boundary spanning is a concept that deals with the structures of organizations that are transitioning from a rigid hierarchical structure towards a network-based expert organization, which gives rise to informal boundaries rather than structural ones [182]. Boundary spanners are those people and entities who bridge these boundaries and opportunities. In the software engineering context, boundary spanning has been studied in the context of global software development [92]. Startuppers can be seen as boundary spanners when they need to bridge between various stakeholders. While boundaries are always unavoidable, but also necessary and useful, knowledge is required on how they can be crossed, rearranged, or even dissolved when considered harmful [178]. Startuppers should see boundaries as tools that facilitate and support making sense out of the environment. Boundary spanning helps in discovering how to overcome the challenges of distributed global work, where motivations, work styles, and knowledge domains vary across boundaries. Startuppers can become knowledge brokers, transferring and sharing their knowledge.

There are other theoretical lenses that can be used to study software startups. Startups deal with innovative services and products, often for new or emerging markets. Birkinshaw et al. [8]

analyse the innovation theories presented and propose a framework for management innovation process. This could be applied to the startup innovation process context to explore how product development moves from problem-driven search through trial and error to a finished prototype. The analysis can be complemented with Van de Ven and Poole's [177] four views into organizational changes, in which they present alternate processes for organizations to transform.

Theorizing software startups is important, since there is a current lack of understanding of the dynamics in startups. Theoretical advancements need to be achieved so that researchers can make better sense out the diverse contexts, situations, and places where startuppers strive for success.

3.6.2. Defining the Lean Startup concept and evaluating practice

Many positive drivers underpin the Lean Startup movement. The literature is abound with claims of reduced risk [55, 148], the benefits of evidenced-based trials [11, 148], and shorter time-to-market [148]. We certainly know that these benefits are needed, given the challenges experienced by early stage software startups [70, 67] and the percentage that fail [148]. Indeed, many software start-ups fail [129, 39] because they waste too much time and money building the wrong product before realising too late what the right product should have been [139, 16]. These challenges coupled with high uncertainty make the Lean Startup Methodology attractive to software startups as it supposedly offers an integrated approach to creating products and services that fit the market [74]. This research builds on previous research conducted by Dennehy, Kasraian, O'Raghallaigh, and Conboy [43], which identified a significant absence of frameworks that assisted startups to efficiently and effectively progress their Minimum Viable Products (MVP) to a Product Market Fit (PMV). The theoretical advancement of the lean concept in contemporary software engineering and software development literature has been arrested, mainly because the academic research community has followed "fads and fancies" which characterize academic research. The implications for the arrested theoretical development of lean concept, listed next, are the motivation for this research.

As is often the case with new and emerging phenomena, Lean Startup practice has led research, with the creation, promotion, and dissemination of these methods almost completely due to the efforts of practitioners and consultants. Now, Lean Startup research is beginning to gain momentum, as is evident from the increasing number of dedicated journal special issues, conferences, conference tracks, and workshops. While there are merits to adopting such a practice-oriented focus, little if any research effort has focused on the conceptual development of Lean Startup and its underlying components. As practice has lead research, the definition of Lean Startup has emerged through how it is used in practice. As a result, Lean Startup adoption is often defined by how the practices are adhered to, rather than the value gleaned from their use, adaptation, or, in some cases, abandonment. We see this in many other methods such as in agile, where many define "being agile" as how many Scrum or XP practices are used, rather than the value obtained by their use [33]. As a result, the current body of software startup knowledge suffers from a number of limitations, including:

(1) Lack of clarity: While there is broad agreement in principle regarding what constitutes key concepts such as MVP, assumptions regarding the specific definitions, interpretations, use, and evaluations are often unclear in many existing Lean Startup studies. This makes critical appraisal, evidence-based evaluation, and comparison across studies extremely difficult.

(2) Lack of cohesion and cumulative tradition: A good concept or theory should cumulatively build on existing research. Very little academic research has examined Lean Startup using concepts that have more mature and substantive bodies of research with theories, frameworks and other lenses that have been thoroughly tested over time. The lean concept has been applied in manufacturing since WW1, and yet in Lean Startup research we see very myopic and limited use of the broad

lean frameworks available. Other concepts that influence Lean Startup include agility, flow, and innovation.

(3) Limited applicability: Adherence-based measures of Lean Startup inhibit the ability to apply Lean Startup in domains other than that originally intended. Research now attempts to apply Lean Startup in other environments, such as large organizations and regulated environments, and so this will become a more prevalent issue as this trend continues. Therefore, questions relevant for this research track include:

- RQ1: What are the core concepts that underpin Lean Startup?
- RQ2: What are the components of a higher abstract Lean Startup that allows the concept to be applied and evaluated in a value-based manner?
- RQ3: What theories, frameworks, metrics, and other instruments from these existing related bodies of knowledge can be applied to Lean Startup?
- RQ4: How can these be effectively applied to improve the use of Lean Startup in practice, and the study and improvement of Lean Startup in research?
- RQ5: How can Lean Startup then be tailored to suit environments it was not originally designed to support, e.g. large organizations, regulated environments, or peer production?
- RQ6: Does Lean Startup enable or inhibit fundamental leaps in business and software business ideas? For example, does MVP place an invisible ceiling, wherein once you reach MVP you subconsciously stop looking for the truly significant innovation?

As there is reciprocal relationship between practice and academia, where academic research is informed by practice and practice is informed by academic research, this research would impact on research and on practice. By answering RQ4-RQ6, this research track would provide practice with empirical evidence on the utility of lean practices in diverse environments, while also positioning the lean method at the core of academic research (RQ1-RQ3). As case study research is an empirical inquiry that "investigates a contemporary phenomenon in depth and within its real-life context" [188], it would be highly suited to addressing the theoretical limitations of lean and for answering the questions listed above. Specifically, the use of a multiple-case design would allow a cross-case pattern to develop more sophisticated descriptions and powerful explanations [128] of the lean concept.

The challenges of new product development are not confined to software startups. Therefore, software engineering teams working in distributed or regulated environments such as financial services and within multinational companies would provide rich insights to the advancement of the lean concept.

3.6.3. Research collaboration strategies with software startups

Empirical research in the area of software engineering normally requires access to organizations and artefacts from companies developing software intensive products and services [185]. In the case of startups, such access is very limited, due to several challenges:

1. startups have limited resources both in terms of person hours and calendar time for anything but working on their MVP,
2. startups want all investments to yield almost immediate results, thus investments in long-term potential are not prioritized, and
3. artefacts and actual products are often very sensitive, as the startup is very vulnerable.

These and other reasons limit empirical research, as reflected in both academic knowledge about startups overall, but also in the superficial nature of what is available. For this reason, any initiative to seriously collect empirical data as well as conduct research on core challenges facing startups has to originate with a strategy that overcomes these obstacles. One possible strategy is to pool resources and access to startups, in essence sharing empirical data and coordinating research into startup software engineering. Coordination should be seen as equally central, as it enables researchers to limit the impact and costs as each study and project part can be focused and small, and several larger issues can be tackled through coordination. Concrete examples of joint activities include, but are not limited to:

1. joint surveys at the superficial level (pooling resources to collect many data points),
2. complementary surveys and case studies where each partner does a part only, but the results can be combined in analysis and synthesis,
3. formulating a complementary research agenda with clear interfaces and joint research questions, and
4. pooling resources in relation to testing "solutions" emerging from the collaboration.

While this strategy opens the possibility to share the resource requirements among the studied startups, there are open questions regarding its implementation:

– RQ1: To what extent is data from different startups and startup ecosystems comparable? In other words, which techniques exist to perform meta-analysis of the gathered heterogeneous data?
– RQ2: How can we efficiently transfer technology between researchers and startups, and how can we measure the impact of transferred solutions?

We conjecture that the software startup context model discussed in Section 3.1.1 would be an enabler for answering RQ1. Confounding variables [173] could then be easier identified, allowing for sample stratification and robust statistical analyses [103]. In particular, data collected from different researchers could be aggregated and increase the strength of the conclusions drawn from the analysis, i.e. enabling meta-analysis [76].

Answering RQ2 would allow us to actually support software startups on a broad basis with the knowledge gained from the research proposed in this agenda. While different approaches exist to transfer knowledge from academia to industry [72, 181], they are mostly targeted at mature companies that have the resources to collaborate with researchers over a longer period of time. We think that software startup ecosystems, discussed in Section 3.5, can contribute to technology transfer if researchers are active in these structures and can create a win-win situation where both startups and researchers benefit.

4. Discussion

In this section we give a brief overview of the research tracks in relation to other work in software engineering and their potential impact on the field. We conclude this section with a discussion on the study's limitations.

Software startup engineering research centers around the core knowledge base in Software Engineering [17]. This is illustrated by the research tracks proposed in Section 3.1 that encompass providing support for startup engineering activities. Noticing what is considered "good" software engineering practice [17], and the challenges that software startups encounter [67, 106], we see potential in directing research towards efficient and effective requirements for engineering practices in startups. Klotins et al. [106] studied 88 experience reports from startups and identified lack of requirements validation, classification (to enable prioritization), and identification of requirements sources (to identify a relevant value proposition) as causes for engineering uncertainty, which maps to the early-stage startup challenges of technology uncertainty and delivering customer value, identified by Giardino et al. [67]. Unlike large companies, software startups have unique time and resource constraints and thus cannot afford to develop features and services that will not be used or valued by the customers. We believe that lightweight practices to identify, and, most importantly, analyse requirements for their business value can help software startups in their decision process. Looking at the research tracks in Section 3.1, several of them touch upon requirements engineering aspects. Prototypes can be used to communicate with customers to elicit requirements (Section 3.1.4), while product innovation assessment (Section 3.1.3) is relevant in the context of analysing the customers' perceived value of the offered solutions. Even optimizing the effort spent in requirements engineering and quality assurance, for example by using test cases as requirements [9], involving product users for testing (Section 3.1.7), addresses requirements engineering aspects.

The focus on requirements in software startup engineering research directly relates to the research tracks presented in Section 3.2, startup evolution models and patterns, as the cost of pivoting could be reduced by earlier and less ad-hoc analysis of requirements and value propositions of the envisioned products. The patterns emerging from the research on survival capabilities of software startups, proposed in Section 3.2.2, could provide valuable heuristics leading to a lightweight analysis of product value propositions. The research on pivoting and survival capabilities is likely to affect software startup practitioners on a strategic level by providing them managerial decision support that draws from models rooted in software engineering practice. An example where such a cross-discipline approach has been very successful is value-based software engineering [15].

The research tracks described in Section 3.3 were grouped under the name "cooperative and human aspects in software startups", borrowed from the research area in software engineering that is interested in studying the impact of cognitive abilities, team composition, workload, informal communication, expertise identification and other human aspects on software construction [157]. We conjecture that studying and understanding these aspects better has a large potential as software startups are driven by motivated individuals rather than a corporate agenda. Lessons from this research can both benefit startup practitioners, in particular in conjunction with the work on software startups ecosystems (Section 3.5), and more mature companies, for example by applying models of competency needs that could emerge from the work presented in Section 3.3.1.

The remaining research tracks described in Sections 3.4 - 3.5 take a step back from what happens *inside* a software startup. The research tracks in Section 3.4 propose to apply startup concepts in non-startup contexts. The idea of extracting a concept from one context and applying it in another has proven successful in other areas, such as in systematic literature reviews [130, 104] and open source principles [169, 80, 179]. The premise of internal startups is that the positive traits of "startups in the wild" can be transferred to a corporate environment, fostering innovation and faster product development. The overall aim of the research tracks described in Section 3.4 is to evaluate whether the traits of startups can actually produce thriving environments within mature companies. In comparison, the research on startup ecosystems and innovation hubs (Section 3.5) takes a broader and higher level view of software startup phenomenon. Neither independent startups nor mature companies adopting internal startup initiatives live in isolation. A better understanding of startup ecosystems and innovation hubs might thereby provide key insights into the factors that create a fruitful software startup environment.

Finally, the research tracks in Section 3.6 look at aspects relevant for implementing the research agenda described in this paper. In particular, theories that can be used to better understand the dynamics in and around software startups are of value when attempting to construct a more holistic understanding of software startups in their various contexts. For the research on defining the Lean Startup concept, parallels to and lessons from similar endeavours around research on agile software development [48] should be taken into consideration. In this paper, we followed a recommendation by Dybå and Dingsøyr to develop a research agenda on the phenomenon of interest [48]. However, in order to implement this research agenda, we need to also answer the questions about how to enable efficient and effective research collaborations with software startups (Section 3.6.3).

4.1. Limitations

The research agenda presented in this paper was developed “bottom-up”, i.e. the areas of interest were proposed and described by a sample of software startup researchers without any restriction on covering certain aspects of the software engineering body of knowledge but guided by their past, current and future work in the field. Often, these researchers have both a leg in academia and in the startup community, either as mentors, founders, or simply as part of the development team. This approach to develop a research agenda is not uncommon (see e.g. [20, 23, 29]), but is threatened by a potential bias towards the preferences of individual researchers. This is why we invited a large number of our peers to contribute to the agenda. Even though the research tracks cover many software engineering aspects and beyond, the agenda is only a sample of the potentially relevant future research on software startups. This means that potentially interesting and relevant research topics, such as use of open source software, business model development, legal issues and intellectual property rights, are not discussed in this paper. However, we expect that the agenda will grow together with the research community as soon as the work on the proposed research tracks bears fruits, leading to new research questions.

5. Outlook and Conclusions

Software startups are an interesting and stimulating phenomenon in the modern economy and are of paramount importance for the societies of today. Despite of high failure rates, communities, cities and countries are investing on stimulating the creation of software startups. While these startups may not solve the unemployment problems of many countries they stimulate a new type of positive dynamism in societies encouraging people to collaborate and develop their personal skills in novel ways. The emergence of the software startup research area reflects the fact that we need to better understand this phenomenon to learn valuable lessons and accumulate valid knowledge to benefit future entrepreneurial initiatives. The research agenda described in this paper is one of the first attempts to establish the software startup as a nascent, yet fast growing research area, and to depict its landscape by highlighting the interesting research topics and questions to explore.

It is worth emphasizing again that software engineering is only one of the multiple disciplines that are relevant and can inform software startup practice. Other disciplines include Economics, Entrepreneurship, Design, Finance, Sociology, and Psychology. Therefore, there is a need to collaborate with researchers from these disciplines in order to increase the potential of achieving relevant and useful research results that can benefit practice.

Due to the emerging nature of the field, there is still much to be done to establish software startups as a research area. Relevant concepts need clear definitions, substantive theories need to be developed, and initial research findings need to be validated by future studies. Software startups are very diversified in terms of entrepreneurs’ varying approaches to their startup endeavours. Without the sound foundation mentioned above for this research area, there are risks of asking irrelevant research questions and not being able to attain rigorous results.

Last but not least, this research agenda is not meant to be exhaustive, and we are aware that we may exclude some important Software Engineering topics relevant to software startups. The research agenda is open to additions of new tracks, topics, and research questions by other researchers interested in the research area. With contributions and commitments from researchers from different institutions and backgrounds, collectively we can establish software startup as a promising and significant research area that attracts more exciting discovery and contribution. We welcome those interested in joining the Software Startup Research Network in fostering the collaboration between researchers and taking the research agenda further.

References

[1] Software engineering – software product quality requirements and evaluation (SQuaRE) – guide to SQuaRE - ISO/IEC 25000:2005. Technical report, International Organization for Standardization, 2005.

[2] ISO/IEC/IEEE Systems and software engineering – Architecture description. *ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000)*, pages 1–46, 2011.

[3] Frequently asked questions about small business. Technical report, U.S. Small Business Administration, 2014.

[4] Y. Z. Anbardan and M. Raeyat. Open Innovation: Creating Value Through Co-Creation. In *Proceedings 7th World Conference on Mass Customization, Personalization, and Co-Creation (MCPC 2014)*, pages 437–447, Aalborg, Denmark, 2014. Springer.

[5] R. Balachandra and J. Friar. Factors for success in R&D projects and new product innovation: a contextual framework. *IEEE Transactions on Engineering Management*, 44(3):276 –287, 1997.

[6] C. K. Bart. New venture units: use them wisely to manage innovation. *Sloan Management Review*, 29(4):35–43, 1988.

[7] M. Beaudouin-Lafon and W. E. Mackay. Prototyping development and tools. *Handbook of Human-Computer Interaction*, pages 1006–1031, 2002.

[8] J. Birkinshaw, G. Hamel, and M. J. Mol. Management innovation. *Academy of management Review*, 33(4):825–845, 2008.

[9] E. Bjarnason, M. Unterkalmsteiner, E. Engström, and M. Borg. A Multi-Case Study of Agile Requirements Engineering and Using Test Cases as Requirements. *Information and Software Technology*, 77:61–79, 2016.

[10] S. Blank. *The Four Steps to the Epiphany: Successful Strategies for Products that Win*. Cafepress.com, 2005.

[11] S. Blank. Why the Lean Start-Up Changes Everything. *Harvard Business Review*, 91(5), 2013.

[12] S. Blank and B. Dorf. *The Startup Owner's Manual: The Step-By-Step Guide for Building a Great Company*. K & S Ranch, 2012.

[13] B. S. Blichfeldt and P. Eskerod. Project portfolio management - There's more to it than what management enacts. *International Journal of Project Management*, 26(4):357–365, 2008.

[14] Z. Block and I. C. MacMillan. Milestones for successful venture planning. *Harvard Business Review*, 63(5):184–196, 1985.

[15] B. Boehm. Value-based Software Engineering. *SIGSOFT Softw. Eng. Notes*, 28(2):1–12, 2003.

[16] J. Bosch, H. H. Olsson, J. Björk, and J. Ljungblad. The Early Stage Software Startup Development Model: A Framework for Operationalizing Lean Principles in Software Startups. In *Proceedings 4th International Conference on Lean Enterprise Software and Systems (LESS)*, pages 1–15, Galway, Ireland, 2013. Springer.

[17] P. Bourque and R. E. Fairley, editors. *Guide to the Software Engineering Body of Knowledge*. IEEE, 3rd edition, 2014.

[18] A. Brem and K.-I. Voigt. Integration of market pull and technology push in the corporate front end and innovation management—Insights from the German software industry. *Technovation*, 29(5):351–367, 2009.

[19] T. Brown. *Change by Design: How Design Thinking Transforms Organizations and Inspires Innovation.* HarperBusiness, New York, 2009.

[20] M. Broy. Challenges in Automotive Software Engineering. In *Proceedings 28th International Conference on Software Engineering (ICSE)*, pages 33–42, Shanghai, China, 2006. ACM.

[21] C. G. Brush, T. S. Manolova, and L. F. Edelman. Properties of emerging organizations: An empirical test. *Journal of Business Venturing*, 23(5):547–566, 2008.

[22] E. Carmel. Time-to-completion in software package startups. In *Proceedings 27th Hawaii International Conference on System Sciences (HICSS)*, pages 498–507. IEEE, 1994.
[23] S. Chandra, V. S. Sinha, S. Sinha, and K. Ratakonda. Software Services: A Research Roadmap. In *Future of Software Engineering (FOSE)*, pages 40–54, Hyderabad, India, 2014. ACM.
[24] C.-J. Chen. Technology commercialization, incubator and venture capital, and new venture performance. *Journal of Business Research*, 62(1):93–103, 2009.
[25] H. Chesbrough and A. Crowther. Beyond high tech: early adopters of open innovation in other industries. *R&d Management*, 36(3):229–236, 2006.
[26] H. W. Chesbrough. *Open innovation: The new imperative for creating and profiting from technology*. Harvard Business Press, 2006.
[27] P. Clarke and R. V. O'Connor. The situational factors that affect the software development process: Towards a comprehensive reference framework. *Information and Software Technology*, 54(5):433–447, 2012.
[28] B. Clarysse, J. Bruneel, and M. Wright. Explaining growth paths of young technology-based firms: structuring resource portfolios in different competitive environments. *Strategic Entrepreneurship Journal*, 5(2):137–157, 2011.
[29] J. Cleland-Huang, O. C. Z. Gotel, J. Huffman Hayes, P. Mäder, and A. Zisman. Software Traceability: Trends and Future Directions. In *Proceedings Future of Software Engineering*, pages 55–69, Hyderabad, India, 2014. ACM.
[30] L. Cocco, K. Mannaro, G. Concas, and M. Marchesi. Simulating Kanban and Scrum vs. Waterfall with System Dynamics. In *Proceedings 12th Internation XP Conference (XP)*, pages 117–131, Madrid, Spain, 2011. Springer.
[31] G. Coleman and R. V. O'Connor. An investigation into software development process formation in software start-ups. *Journal of Enterprise Information Management*, 21(6):633–648, 2008.
[32] S. Coleman, C. Cotei, and J. Farhat. A Resource-based view of new firm survival: new perspectives on the role of industry and exit route. *Journal of Developmental Entrepreneurship*, 18(01):1–25, 2013.
[33] K. Conboy. Agility from first principles: Reconstructing the concept of agility in information systems development. *Information Systems Research*, 20(3):329–354, 2009.
[34] G. Concas, M. I. Lunesu, M. Marchesi, and H. Zhang. Simulation of software maintenance process, with and without a work-in-process limit. *Journal of Software: Evolution and Process*, 25(12):1225–1248, 2013.
[35] R. G. Cooper. From experience - the invisible success factors in product innovation. *Journal of Product Innovation Management*, 16(2):115–133, 1999.
[36] R. G. Cooper. Perspective: The innovation dilemma: How to innovate when the market is mature. *Journal of Product Innovation Management*, 28(SUPPL. 1):2–27, 2011.
[37] A. Croll and B. Yoskovitz. *Lean Analytics: Use Data to Build a Better Startup Faster*. O'Reilly Media, Sebastopol, USA, 1st edition, 2013.
[38] M. M. Crossan and M. Apaydin. A multi-dimensional framework of organizational innovation: A systematic review of the literature. *Journal of Management Studies*, 47(6):1154–1191, 2009.
[39] M. Crowne. Why software product startups fail and what to do about it. In *International Engineering Management Conference (IEMC)*, pages 338–343, Cambridge, UK, 2002. IEEE.
[40] D. Cukier, F. Kon, and N. Krueger. Designing a Maturity Model for Software Startup Ecosystems. In *Proceedings 1st International Workshop on Software Startups*, pages 600–606, Bolzano, Italy, 2015. Springer.
[41] D. Cukier, F. Kon, and L. S. Thomas. Software Startup Ecosystems Evolution: The New York City Case Study. In *Proceedings 2nd International Workshop on Software Startups*, Trondheim, Norway, 2016. IEEE.
[42] M. A. Cusumano and K. Nobeoka. *Thinking Beyond Lean: How Multi-project Management is Transforming Product Development at Toyota and Other Companies*. Simon and Schuster, 1998.
[43] D. Dennehy, L. Kasraian, O. O'Raghallaigh, and K. Conboy. Product Market Fit Frameworks for Lean Product Development. In *Proceedings R&D Management Conference 2016 "From Science to Society: Innovation and Value Creation"*, Cambridge, UK, 2016.
[44] G. G. Dess, R. D. Ireland, S. A. Zahra, S. W. Floyd, J. J. Janney, and P. J. Lane. Emerging Issues in Corporate Entrepreneurship. *Journal of Management*, 29(3):351–378, 2003.

[45] T. Dingsøyr and Y. Lindsjørn. Team Performance in Agile Development Teams: Findings from 18 Focus Groups. In *Proceedings 14th International Conference on Agile Software Development*, pages 46–60, Vienna, Austria, 2013.

[46] K. Dovan. Reliability in content analysis: Some common misconceptions and recommendations. *Human Communication Research*, 30(3):411–433, 1998.

[47] T. Dybå, D. I. Sjøberg, and D. S. Cruzes. What Works for Whom, Where, When, and Why?: On the Role of Context in Empirical Software Engineering. In *Proceedings International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 19–28, Lund, Sweden, 2012. ACM.

[48] T. Dybå and T. Dingsøyr. Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50(9–10):833–859, 2008.

[49] H. Edison. A Conceptual Framework of Lean Startup Enabled Internal Corporate Venture. In *Proceedings 1st International Workshop on Software Startups*, pages 607–613, Bolzano-Bozen, Italy, 2015. Springer.

[50] H. Edison, N. bin Ali, and R. Torkar. Towards innovation measurement in the software industry. *Journal of Systems and Software*, 86(5):1390–1407, 2013.

[51] H. Edison, D. Khanna, S. S. Bajwa, V. Brancaleoni, and L. Bellettati. Towards a Software Tool Portal to Support Startup Process. In *Proceedings 1st International Workshop on Software Startups*, pages 577–583, Bolzano, Italy, 2015. Springer.

[52] H. Edison, X. Wang, and P. Abrahamsson. Lean startup. In *Scientific Workshop Proceedings of the XP conference*, pages 1–7, Helsinki, Finland, 2015. ACM.

[53] A. Efeoğlu, C. Møller, and M. Sérié. Solution Prototyping with Design Thinking – Social Media for SAP Store: A Case Study. In *Proceedings European Design Science Symposium (EDSS)*, pages 99–110, Dublin, Ireland, 2013. Springer.

[54] K. M. Eisenhardt and J. A. Martin. Dynamic capabilities: what are they? *Strategic Management Journal*, 21(10-11):1105–1121, 2000.

[55] T. R. Eisenmann, E. Ries, and S. Dillard. Hypothesis-Driven Entrepreneurship: The Lean Startup. *Harvard Business School*, 2012.

[56] S. Elo and H. Kyngäs. The qualitative content analysis process. *Journal of Advanced Nursing*, 62(1):107–115, 2008.

[57] M. D. Ensley, K. M. Hmieleski, and C. L. Pearce. The importance of vertical and shared leadership within new venture top management teams: Implications for the performance of startups. *The Leadership Quarterly*, 17(3):217–231, 2006.

[58] W. Eversheim. *Innovation Management for Technical Products: Systematic and Integrated Product Development and Production Planning*. Springer Science & Business Media, 2008.

[59] T. Feng, L. Sun, C. Zhu, and A. S. Sohal. Customer orientation for decreasing time-to-market of new products IT implementation as a complementary asset. *Industrial Marketing Management*, 41(6):929 – 939, 2012.

[60] C. Fernández-Sánchez, J. Garbajosa, and A. Yagüe. A framework to aid in decision making for technical debt management. In *Proceedings 7th International Workshop on Managing Technical Debt (MTD)*, pages 69–76, Bremen, Germany, 2015. IEEE.

[61] M. C. Fonseca. O ecossistema de startups de software da cidade de São Paulo. Master's thesis, University of São Paulo, 2016.

[62] D. Francis and W. Sandberg. Friendship within entrepreneurial teams and its association with team and venture performance. *Entrepreneurship: Theory and Practice*, 25(2):5–21, 2000.

[63] A. Frenkel and S. Maital. *Mapping National Innovation Ecosystems: Foundations for Policy Consensus*. Edward Elgar Publishing, London, UK, 2014.

[64] J. Füller, R. Schroll, and E. von Hippel. User generated brands and their contribution to the diffusion of user innovations. *Research Policy*, 42(6–7):1197–1209, 2013.

[65] J. S. Gans and S. Stern. The product market and the market for "ideas": commercialization strategies for technology entrepreneurs. *Research Policy*, 32(2):333–350, 2003.

[66] D. A. Garvin and L. C. Levesque. Meeting the challenge of corporate entrepreneurship. *Harvard business review*, 84(10):102–12, 150, 2006.

[67] C. Giardino, S. S. Bajwa, X. Wang, and P. Abrahamsson. Key Challenges in Early-Stage Software Startups. In *Proceedings 16th International XP Conference (XP)*, pages 52–63, Helsinki, Finland, 2015.

Springer.
[68] C. Giardino, N. Paternoster, M. Unterkalmsteiner, T. Gorschek, and P. Abrahamsson. Software Development in Startup Companies: The Greenfield Startup Model. *Transactions on Software Engineering*, 42(6):585–604, 2016.
[69] C. Giardino, M. Unterkalmsteiner, N. Paternoster, T. Gorschek, and P. Abrahamsson. What do we know about software development in startups? *IEEE Software*, 31(5):28–32, 2014.
[70] C. Giardino, X. Wang, and P. Abrahamsson. Why Early-Stage Software Startups Fail: A Behavioral Framework. In *Proceedings 5th International Conference on Software Business (ICSOB)*, pages 27–41, Paphos, Cyprus, 2014. Springer.
[71] F. Giones and F. Miralles. Strategic Signaling in Dynamic Technology Markets: Lessons From Three IT Startups in Spain. *Global Business and Organizational Excellence*, 34(6):42–50, 2015.
[72] T. Gorschek, C. Wohlin, P. Carre, and S. Larsson. A Model for Technology Transfer in Practice. *IEEE Software*, 23(6):88–95, 2006.
[73] C. Grevet and E. Gilbert. Piggyback prototyping: Using existing, large-scale social computing systems to prototype new ones. In *Proceedings 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 4047–4056, Seoul, Korea, 2015. ACM.
[74] Y. Harb, C. Noteboom, and S. Sarnikar. Evaluating Project Characteristics for Selecting the Best-fit Agile Software Development Methodology: A Teaching Case. *Journal of the Midwest Association for Information Systems (JMWAIS)*, 1(1), 2015.
[75] T. Hatzakis, M. Lycett, and A. Serrano. A programme management approach for ensuring curriculum coherence in IS (higher) education. *European Journal of Information Systems*, 16(5):643–657, 2007.
[76] W. Hayes. Research synthesis in software engineering: a case for meta-analysis. In *Proceedings 6th International Software Metrics Symposium*, pages 143–151, Boca Raton, USA, 1999. IEEE.
[77] B. L. Herrmann, J.-F. Gauthier, D. Holtschke, R. Berman, and M. Marmer. The Global Startup Ecosystem Ranking 2015. Technical Report August, 2015.
[78] S. A. Hill and J. Birkinshaw. Ambidexterity and Survival in Corporate Venture Units. *Journal of Management*, 40(7):1899–1931, 2014.
[79] O.-P. Hilmola, P. Helo, and L. Ojala. The value of product development lead time in software startup. *System Dynamics Review*, 19(1):75–82, 2003.
[80] E. v. Hippel. Innovation by User Communities: Learning from Open-Source Software. *MIT Sloan Management Review*, 42(4):82–82, 2001.
[81] L. Hokkanen, K. Kuusinen, and K. Väänänen. Early Product Design in Startups: Towards a UX Strategy. In *Proceedings 16th International Conference on Product-Focused Software Process Improvement (PROFES)*, pages 217–224, Bolzano-Bozen, Italy, 2015. Springer.
[82] L. Hokkanen, K. Kuusinen, and K. Väänänen. Minimum viable user experience: A framework for supporting product design in startups. In *Proceedings 17th International XP Conference (XP)*, Edinburgh, Scotland, 2016. Springer. In press.
[83] L. Hokkanen and M. Leppänen. Three Patterns for User Involvement in Startups. In *Proceedings 20th European Conference on Pattern Languages of Programs (EuroPLoP)*, pages 51:1–51:8, Kloster Irsee, Germany, 2015. ACM.
[84] L. Hokkanen and K. Väänänen-Vainio-Mattila. UX work in startups: current practices and future needs. In *Proceedings 16th International XP Conference (XP)*, pages 81–92, Helsinki, Finland, 2015. Springer.
[85] B. Honig and T. Karlsson. Institutional forces and the written business plan. *Journal of Management*, 30(1):29–48, 2004.
[86] G. Hu, L. Wang, S. Fetch, and B. Bidanda. A multi-objective model for project portfolio selection to implement lean and Six Sigma concepts. *International Journal of Production Research*, 46(23):6611–6625, 2008.
[87] International Organization for Standardization. *Ergonomics of Human-system Interaction: Part 210: Human-centred Design for Interactive Systems*. ISO, 2010.
[88] S. Jack, J. Hyman, and F. Osborne. Small entrepreneurial ventures culture, change and the impact on HRM: A critical review. *Human Resource Management Review*, 16(4):456–466, 2006.
[89] J. J. P. Jansen, M. P. Tempelaar, F. A. J. van den Bosch, and H. W. Volberda. Structural Differentiation and Ambidexterity: The Mediating Role of Integration Mechanisms. *Organization Science*,

20(4):797–811, 2009.

[90] S. Jayshree and R. Ramraj. Entrepreneurial Ecosystem: Case Study on the Influence of Environmental Factors on Entrepreneurial Success. *European Journal of Business and Management*, 4(16):95–102, 2012.

[91] F. Johne and P. A. Snelson. Success factors in product innovation: A selective review of the literature. *Journal of Product Innovation Management*, 5(2):114–128, 1988.

[92] A. Johri. Boundary spanning knowledge broker: An emerging role in global engineering firms. In *Proceedings 38th Annual Frontiers in Education Conference*, pages 7–12. IEEE, 2008.

[93] J. Järvinen, T. Huomo, T. Mikkonen, and P. Tyrväinen. From Agile Software Development to Mercury Business. In *Proceedings 5th International Conference on Software Business (ICSOB)*, pages 58–71, Paphos, Cyprus, 2014. Springer.

[94] J. Kamm, J. Shuman, J. Seeger, and A. Nurick. Entrepreneurial teams in new venture creation: A research agenda. *Entrepreneurship Theory and Practice*, 14(4):7–17, 1990.

[95] T. Karlsson and B. Honig. Judging a business by its cover: An institutional perspective on new ventures and the business plan. *Journal of Business Venturing*, 24(1):27–45, 2009.

[96] J. Katz and W. B. Gartner. Properties of Emerging Organizations. *Academy of Management Review*, 13(3):429–441, 1988.

[97] J. R. Katzenbach and D. K. Smith. The discipline of teams. *Harvard Business Review*, 71(2):111–120, 1993.

[98] M. D. Kelly. Lessons Learned from Software Testing at Startups. In *EuroStar-Software Testing Conference*, Amsterdam, The Netherlands, 2012.

[99] B. Kersten and C. Verhoef. IT portfolio management: A banker's perspective on IT. *Cutter IT Journal*, 2003.

[100] D. Kirk and S. MacDonell. Categorising Software Contexts. In *Proceedings 2014 Americas Conference on Information Systems (AMCIS)*, Savannah, USA, 2014. AIS Electronic Library.

[101] D. Kirk and S. G. MacDonell. Investigating a Conceptual Construct for Software Context. In *Proceedings 18th International Conference on Evaluation and Assessment in Software Engineering (EASE)*, pages 27:1–27:10, London, UK, 2014. ACM.

[102] D. Kirsh, B. Goldfarb, and A. Gera. Firm or substance: the role of business plans in venture capital decision making process. *Strategic Management Journal*, (30):487–515, 2009.

[103] B. Kitchenham, L. Madeyski, D. Budgen, J. Keung, P. Brereton, S. Charters, S. Gibbs, and A. Pohthong. Robust Statistical Methods for Empirical Software Engineering. *Empirical Software Engineering*, pages 1–52, 2016.

[104] B. A. Kitchenham, T. Dybå, and M. Jørgensen. Evidence-Based Software Engineering. In *Proceedings 26th International Conference on Software Engineering (ICSE)*, pages 273–281, Edinburgh, UK, 2004. IEEE.

[105] E. Klotins, M. Unterkalmsteiner, and T. Gorschek. Software engineering practices in start-up companies: A mapping study. In *6th International Conference on Software Business*, pages 245–257. Springer, 2015.

[106] E. Klotins, M. Unterkalmsteiner, and T. Gorschek. Software Engineering in Start-up Companies: an Exploratory Study of 88 Startups. *Empirical Software Engineering*, 2016. In Submission.

[107] K. Klyver and M. T. Schenkel. From Resource Access to Use: Exploring the Impact of Resource Combinations on Nascent Entrepreneurship. *Journal of Small Business Management*, 51(4):539–556, 2013.

[108] F. Kon, D. Cukier, C. Melo, O. Hazzan, and H. Yuklea. A conceptual framework for software startup ecosystems: the case of israel. Technical report, Technical Report RT-MAC-2015-01, Department of Computer Science, University of São Paulo, 2015.

[109] J. Krebs. *Agile portfolio management.* Microsoft Press, 1st edition, 2008.

[110] M. König, G. Baltes, and B. Katzy. On the role of value-network strength as an indicator of technology-based venture's survival and growth: Increasing innovation system efficiency by leveraging transaction relations to prioritize venture support. In *Proceedings International Conference on Engineering, Technology and Innovation/ International Technology Management Conference (ICE/ITMC)*, pages 1–9. IEEE, 2015.

[111] M. König, C. Ungerer, R. Büchele, and G. Baltes. Agreement on the Venture's Reality Presented

in Business Plans. In *Proceedings 22nd International Conference on Engineering, Technology and Innovation (ICE)*, Trondheim, Norway, 2016. IEEE.

[112] D. Lavie, U. Stettner, and M. L. Tushman. Exploration and Exploitation Within and Across Organizations. *The Academy of Management Annals*, 4(1):109–155, 2010.

[113] R. LeFave, B. Branch, and C. Brown. How Sprint Nextel Reconfigured IT Resources for Results. *MIS Quarterly*, 2008.

[114] J. Lerner. Corporate venturing. *Harvard Business Review*, 91(10):86–94, 2013.

[115] J. Levie and B. B. Lichtenstein. A Terminal Assessment of Stages Theory: Introducing a Dynamic States Approach to Entrepreneurship. *Entrepreneurship Theory and Practice*, 34(2):317–350, 2010.

[116] Z. Li, P. Avgeriou, and P. Liang. A systematic mapping study on technical debt and its management. *Journal of Systems and Software*, 101:193–220, 2015.

[117] B. B. Lichtenstein, K. J. Dooley, and G. T. Lumpkin. Measuring emergence in the dynamics of new venture creation. *Journal of Business Venturing*, 21(2):153–175, 2006.

[118] H. Lichter, M. Schneider-Hufschmidt, and H. Züllighoven. Prototyping in Industrial Software Projects—Bridging the Gap Between Theory and Practice. In *Proceedings 15th International Conference on Software Engineering (ICSE)*, pages 221–229, Baltimore, USA, 1993. IEEE.

[119] E. Lim, N. Taksande, and C. Seaman. A balancing act: What software practitioners have to say about technical debt. *IEEE Software*, 29(6):22–27, 2012.

[120] D. Lippoldt and P. Stryszowskim. Innovation in the software sector. Technical report, Organisation for Economic Co-operation and Development, 2009.

[121] H. Löfsten and P. Lindelöf. Science Parks and the growth of new technology-based firms—academic-industry links, innovation and markets. *Research Policy*, 31(6):859–876, 2002.

[122] I. C. Macmillan, L. Zemann, and P. Subbanarasimha. Criteria distinguishing successful from unsuccessful ventures in the venture screening process. *Journal of Business Venturing*, 2(2):123–137, 1987.

[123] J. Märijärvi, L. Hokkanen, M. Komssi, H. Kiljander, Y. Xu, M. Raatikainen, P. Seppänen, J. Heininen, M. Koivulahti-Ojala, M. Helenius, and J. Järvinen. *The Cookbook for Successful Internal Startups*. DIGILE and N4S, 2016.

[124] S. Marlow. Human resource management in smaller firms: A contradiction in terms? *Human Resource Management Review*, 16(4):467–477, 2006.

[125] A. Maurya. *Running Lean: Iterate from Plan A to a Plan That Works*. O'Reilly Media, Inc., 2012.

[126] B. May. Applying Lean Startup: An Experience Report – Lean & Lean UX by a UX Veteran: Lessons Learned in Creating & Launching a Complex Consumer App. In *Proceedings Agile Conference (AGILE)*, pages 141–147, Dallas, USA, 2012. IEEE.

[127] S. Meskendahl. The influence of business strategy on project portfolio management and its success - A conceptual framework. *International Journal of Project Management*, 28(8):807–817, 2010.

[128] M. B. Miles and A. Huberman. *Qualitative data analysis: An expanded sourcebook*. Sage Publications, Inc, Thousand Oaks, US, 1994.

[129] J. W. Mullins and R. Komisar. *Getting to Plan B: Breaking Through to a Better Business Model*. Harvard Business Press, 2009.

[130] C. D. Mulrow. Rationale for systematic reviews. *BMJ : British Medical Journal*, 309(6954):597–599, 1994.

[131] S. Nambisan, K. Lyytinen, A. Majchrzak, and M. Song. Digital Innovation Management: Reinventing Innovation Management Research in a Digital World. *MIS Quarterly*, 2016. In press.

[132] R. Nanda and M. Rhodes-Kropf. Investment cycles and startup innovation. *Journal of Financial Economics*, 110(2):403–418, 2013.

[133] J. Nazar. 14 famous business pivots. *[AVAILABLE ONLINE] http://www.forbes.com/sites/jasonnazar/2013/10/08/14-famous-business-pivots/*, 2013.

[134] S. L. Newbert, S. Gopalakrishnan, and B. A. Kirchhoff. Looking beyond resources: Exploring the importance of entrepreneurship to firm-level competitive advantage in technologically intensive industries. *Technovation*, 28(1–2):6–19, 2008.

[135] S. L. Newbert and E. T. Tornikoski. Supporter networks and network growth: a contingency model of organizational emergence. *Small Business Economics*, 39(1):141–159, 2010.

[136] P. Newman, M. A. Ferrario, W. Simm, S. Forshaw, A. Friday, and J. Whittle. The role of design thinking and physical prototyping in social software engineering. *37th IEEE International Conference*

on *Software Engineering*, 2015.
[137] A. Nguyen Duc and P. Abrahamsson. Minimum viable product or multiple facet product? The Role of MVP in software startups. In *Proceedings 17th International XP Conference*, Edinburgh, UK, 2016. Springer.
[138] A. Nguyen Duc, P. Seppänen, and P. K. Abrahamsson. Hunter-gatherer cycle: a conceptual model of the evolution of software startups. pages 199–203, Tallin, Estonia, 2015. ACM.
[139] C. Nobel. Teaching a 'Lean Startup' Strategy. *HBS Working Knowledge*, 2011.
[140] O. Oechslein and A. Tumasjan. Examining Trust within the Team in IT Startup Companies–An Empirical Study in the People's Republic of China. In *Proceedings 45th Hawaii International Conference on System Science (HICSS)*, pages 5102–5111, Maui, USA, 2012.
[141] C. A. O'Reilly and M. L. Tushman. Organizational Ambidexterity: Past, Present, and Future. *Academy of Management Perspectives*, 27(4):324–338, 2013.
[142] N. Paternoster, C. Giardino, M. Unterkalmsteiner, T. Gorschek, and P. Abrahamsson. Software Development in Startup Companies: A Systematic Mapping Study. *Information and Software Technology*, 56(10):1200–1218, 2014.
[143] J. Pelrine. On Understanding Software Agility: A Social Complexity Point Of View. *Emergence: Complexity & Organization*, 13(1/2):26–37, 2011.
[144] K. Petersen and C. Wohlin. Context in Industrial Software Engineering Research. In *Proceedings 3rd International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 401–404, Orlando, USA, 2009. IEEE.
[145] M. Pikkarainen, W. Codenie, N. Boucart, and J. A. Heredia Alvaro, editors. *The Art of Software Innovation*. Springer, Berlin, Germany, 2011.
[146] T. Raz and E. Michael. Use and benefits of tools for project risk management. *International Journal of Project Management*, 19(1):9–17, 2001.
[147] B. D. Reyck, Y. Grushka-Cockayne, M. Lockett, S. R. Calderini, M. Moura, and A. Sloper. The impact of project portfolio management on information technology projects. *International Journal of Project Management*, 23(7):524–537, 2005.
[148] E. Ries. *The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses*. Crown Books, 2011.
[149] J. Rikkila, P. Abrahamsson, and X. Wang. The Implications of a Complexity Perspective for Software Engineering Practice and Research. *Journal of Computer Engineering & Information Technology*, 2012.
[150] S. D. Sarasvathy. Causation and Effectuation: Toward a Theoretical Shift from Economic Inevitability to Entrepreneurial Contingency. *Academy of Management*, 26(2):243–263, 2001.
[151] T. Semrau and S. Sigmund. Networking Ability and the Financial Performance of New Ventures: A Mediation Analysis among Younger and More Mature Firms. *Strategic Entrepreneurship Journal*, 6(4):335–354, 2012.
[152] S. Shahid Bajwa, X. Wang, A. Nguven Duc, and P. Abrahamsson. How Do Software Startups Pivot? Empirical Results from a Multiple Case Study. In *7th International Conference on Software Business (ICSOB 2016)*, pages 169–176, Ljubljana, Slovenia, 2016.
[153] A. Shontell. The 11 most disruptive startups. *Business Insider*, 07/12 2012.
[154] F. Shull, D. Falessi, C. Seaman, M. Diep, and L. Layman. Technical Debt: Showing the Way for Better Transfer of Empirical Results. In *Perspectives on the Future of Software Engineering*, pages 179–190. Springer, 2013.
[155] D. J. Snowden and M. E. Boone. A leader's framework for decision making. *Harvard Business Review*, 85(11):69–76, 2007.
[156] I. Sommerville. *Software Engineering*. Pearson, Boston, 9th edition, 2010.
[157] C. R. B. d. Souza, H. Sharp, J. Singer, L. T. Cheng, and G. Venolia. Guest Editors' Introduction: Cooperative and Human Aspects of Software Engineering. *IEEE Software*, 26(6):17–19, 2009.
[158] S. Srinivasan, I. Barchas, M. Gorenberg, and E. Simoudis. Venture Capital: Fueling the Innovation Economy. *Computer*, 47(8):40–47, 2014.
[159] M. Steinert and L. J. Leifer. 'Finding One's Way': Re-Discovering a Hunter - Gatherer Model based on Wayfaring. *International Journal of Engineering Education*, 28(2):251–252, 2012.
[160] R. Sternberg. Success factors of university-spin-offs: Regional government support programs versus

regional environment. *Technovation*, pages 1–12, 2013.
[161] S. M. Sutton. The Role of Process in a Software Start-up. *IEEE Softw.*, 17(4):33–39, 2000.
[162] M. Taipale. Huitale–A Story of a Finnish Lean Startup. In *Proceedings 1st International Conference on Lean Enterprise and Software Systems (LESS)*, pages 111–114, Helsinki, Finland, 2010. Springer.
[163] D. J. Teece, G. Pisano, and A. Shuen. Dynamic capabilities and strategic management. *Strategic Management Journal*, 18(7):509–533, 1997.
[164] H. Terho, S. Suonsyrj, A. Karisalo, and T. O. Mikkonen. Ways to cross the rubicon: Pivoting in software startups. In *Proceedings 1st International Workshop on Software Startups*, pages 555–568, Bolzano-Bozen, Italy, 2015. Springer.
[165] The Economist. A cambrian moment cheap and ubiquitous building blocks for digital products and services have caused an explosion in startups. special report: Tech startups. *The Economist*, 01/18 2014.
[166] The Economist. Testing, testing launching a startup has become fairly easy, but what follows is back-breaking work. special report: Tech startups. *The Economist*, 01/18 2014.
[167] The Economist. Progress without profits. a flock of startups is making cloud computing faster and more flexible, but most of them will not survive. *The Economist*, 09/19 2015.
[168] E. Tom, A. Aurum, and R. Vidgen. An exploration of technical debt. *Journal of Systems and Software*, 86(6):1498–1516, 2013.
[169] R. Torkar, P. Minoves, and J. Garrigós. Adopting free/libre/open source software practices, techniques and methods for industrial use. *Journal of the Association for Information Systems*, 12(1):88, 2011.
[170] J. R. Turner. *The handbook of project-based management*. McGraw-hill, 2014.
[171] P. Tyrväinen, M. Saarikallio, T. Aho, T. Lehtonen, and R. Paukeri. Metrics Framework for Cycle-Time Reduction in Software Value Creation. In *Proceedings 10th International Conference on Software Engineering Advances (ICSEA)*, Barcelona, Spain, 2015. IARIA.
[172] C. Ungerer, M. König, F. Giones, and G. Baltes. Measuring Venture Emergence and Survival by Analyzing Transaction Relations in Business Plans. In *Proceedings 22nd International Conference on Engineering, Technology and Innovation (ICE)*, Trondheim, Norway, 2016. IEEE.
[173] M. Unterkalmsteiner, T. Gorschek, A. Islam, C. Cheng, R. Permadi, and R. Feldt. A conceptual framework for SPI evaluation. *Journal of Software: Evolution and Process*, 26(2):251–279, 2014.
[174] V. van de Vrande, J. P. de Jong, W. Vanhaverbeke, and M. de Rochemont. Open innovation in SMEs: Trends, motives and management challenges. *Technovation*, 29(6):423–437, 2009.
[175] J. S. van der Ven and J. Bosch. Pivots and Architectural Decisions: Two Sides of the Same Medal? In *Proceedings 8th International Conference on Software Engineering Advances (ICSEA)*, pages 310–317, Venice, Italy, 2013.
[176] W. Vanhaverbeke and M. Cloodt. Open innovation in value networks. *Open innovation: Researching a New Paradigm*, 2006.
[177] A. H. V. D. Ven and M. S. Poole. Explaining Development and Change in Organizations. *Academy of Management Review*, 20(3):510–540, 1995.
[178] E. Wenger. *Communities of Practice: Learning, Meaning, and Identity*. Cambridge University Press, 1998.
[179] J. West. How open is open enough?: Melding proprietary and open source platform strategies. *Research Policy*, 32(7):1259–1285, 2003.
[180] J. West and S. Gallagher. Challenges of open innovation: the paradox of firm investment in open-source software. *R & D Management*, 36(3):319–331, 2006.
[181] R. Wieringa. Empirical research methods for technology validation: Scaling up to practice. *Journal of Systems and Software*, 95:19–31, 2014.
[182] P. Williams. The Competent Boundary Spanner. *Public Administration*, 80(1):103–124, 2002.
[183] P. Witt. Entrepreneurs' networks and the success of start-ups. *Entrepreneurship & Regional Development*, 16(5):391–412, 2004.
[184] WMF. Intelligent assets unlocking the circular economy potential. Technical report, World Economic Forum, December 2015.
[185] J. Wohlin, Claes; Aurum, Aybuke; Angelis, Lefteris; Phillips, Laura; Dittrich, Yvonne; Gorschek, Tony; Grahn, Hakan; Henningsson, Kennet; Kagstrom, Simon; Low, Graham; Rovegard, Per; Tomaszewski, Piotr; van Toorn, Christine; Winter. The Success Factors Powering Academia Collaboration. *IEEE*

Software, 29(2):67–73, 2012.

[186] A. Yagüe, J. Garbajosa, J. Pérez, and J. Díaz. Analyzing Software Product Innovation Assessment by Using a Systematic Literature Review. In *Proceedings 47th Hawaii International Conference on System Sciences (HICSS)*, pages 5049–5058, Waikoloa, USA, 2014. IEEE.

[187] A. Yau and C. Murphy. Is a Rigorous Agile Methodology the Best Development Strategy for Small Scale Tech Startups? Technical Report MS-CIS-13-01, 2013.

[188] R. K. Yin. *Case Study Research: Design and Methods*. Sage Publications, 3rd edition, 2003.

[189] J. Zettel, F. Maurer, J. Münch, and L. Wong. LIPE: a lightweight process for e-business startup companies based on extreme programming. In *Proceedings 3rd International Conference on Product-Focused Software Process Improvement (PROFES)*, pages 255–270. Springer, Kaiserslautern, Germany, 2001.