XPro: a Model to Explain the Limited Adoption and Implementation of Experimentation in Software Startups

Jorge Melegati, Henry Edison, and Xiaofeng Wang

Abstract—Software startups develop innovative, software-intensive products or services. Such innovativeness translates into uncertainty regarding a matching need for a product from potential customers, representing a possible determinant reason for startup failure. Research has shown that experimentation, an approach based on the use of experiments to guide several aspects of software development, could improve these companies' success rate by fostering the evaluation of assumptions about customers' needs before developing a full-fledged product. Nevertheless, software startups are not using experimentation as expected. In this study, we investigated the reasons behind such a mismatch between theory and practice. To achieve it, we performed a qualitative survey study of 106 failed software startups. We built the eXperimentation Progression model (XPro), demonstrating that the effective adoption and implementation of experimentation is a staged process: first, teams should be aware of experimentation, then they need to develop an intention to experiment, perform the experiments, analyze the results, and finally act based on the obtained learning. Based on the XPro model, we further identified 25 inhibitors that prevent a team from progressing along the stages properly. Our findings inform researchers of how to develop practices and techniques to improve experimentation adoption in software startups. Practitioners could learn various factors that could lead to their startup failure so they could take action to avoid them.

Index Terms—software startups, experimentation, experiment-driven software development, startups

1 Introduction

In the last 20 years, innovative software-intensive products changed many aspects of our lives. For instance, nowadays, people move around in many cities worldwide using ride-sharing apps like Uber or Lyft rather than taxi services. We are getting used to booking accommodations for our vacations or business trips through Airbnb rather than on hotel websites. Our daily communication tools are Whatsapp, Twitter, and Zoom instead of phone calls or even emails. These software products, disrupting long-standing traditional industries, are generally created by new and emerging companies, the so-called software startups. These organizations develop innovative software-intensive products or services and search for repeatable and scalable business models [1]. This quest for a viable business model along with the liability of newness leads to higher uncertainties than in established companies [2]. It is essential to emphasize that the startup is a temporary stage leading to a consolidated company or the activities' end. Despite many successful cases as those mentioned above, more than 90% of software startups fail to become durable and profitable businesses [3]. There are various potential reasons for failure, including demanding market conditions, lack of team commitment, and financial issues [4] but wrong business development is one key determinant [5].

In the entrepreneurship literature, a recognized crucial element for business development is experimentation [6], represented by several "trials and errors" along various dimensions of a business model [7]. In the software startup context, experiments, such as problem or solution interviews and prototype testing, could help startup teams evaluate if software products are worth building, whether they attract and retain customers and users, etc. With evaluation results, teams could make more informed decisions and avoid investing their limited resources on a flawed business idea. In this context, the software development focus shifts from developing features following determined requirements to creating experiments informing requirements definition or implemented features evaluation.

Experimentation resonates among practitioners, as evidenced by the Lean Startup methodology success [8]. The approach claims that, instead of developing a product without contacting customers, a startup should treat its business idea as hypotheses, build minimum viable products that allow the team to test the hypotheses, and, based on the data collected, persevere on the idea or pivot. The company should repeat these so-called Build-Measure-Learn cycles until it reaches a repeatable and scalable business model. The Lean Startup is mainly based on anecdotal evidence, but several researchers argued that experimentation is an essential element of the methodology [9], [10], [11].

Despite the Lean Startup popularity (evident from the number of book copies sold[1]), and the support of cloud services making experiments cheaper [12], adoption and implementation of experimentation in software startups as a way of searching repeatable and scalable business

• *J. Melegati and X. Wang are with the Faculty of Computer Science, Free University of Bozen-Bolzano, Bolzano, Italy.*

• *H. Edison is with Lero at NUI Galway, Galway, Ireland*

Manuscript received ; revised .

1. https://www.forbes.com/sites/danschawbel/2017/10/17/eric-ries-why-companies-need-to-create-an-entrepreneurial-culture/

models are limited. Most of these companies still focus on developing complete products or services based on their original ideas instead of experimenting [13], [14], [15]. The literature offers a scarce explanation of why the adoption of experimentation is still limited in software development teams. However, the aspects described above bring to the attention why this fact also happens in software startups. The goal of this study is to understand this phenomenon better. Viewing experimentation as a process innovation, a large body of research useful to understand such a problem is the diffusion of innovations since it is targeted at the adoption and implementation of a new technology [16]. Using these concepts, we formulated the following research question to guided this study:

RQ: Why do software startups fail to adopt and implement an experimentation approach?

A commonality across different models of diffusion and adoption of innovations is their staged nature. Seminal works such as the Technology Acceptance Model (TAM) [17], or Cooper and Zmud's technology implementation model [18] have theorized the steps of adoption and implementation of an innovation. Such a linear process is a simplification of a complex reality. Nevertheless, it allows researchers and practitioners to better understand and act on the process. Therefore, it is reasonable and valuable that, in our study, we took a staged view on the adoption and implementation of experimentation in software startups. Consequently, the first sub-question for our study is:

RQ1: What are the stages of the adoption and implementation process of experimentation in software startups?

Given the final goal of improving the adoption of experimentation in software startups, it is valuable to understand what prevents these companies from reaching more advanced stages in the adoption process. These inhibitors should be handled to reach the final goal. Therefore, the second sub-questions for this study is:

RQ2: What are the inhibitors that prevent software startups from progressing along the stages of adopting and implementing experimentation?

To answer these research questions, we performed a qualitative survey study based on the postmortems of 106 failed startups collected from CBInsights[2]. We analyzed the data using thematic synthesis and built the eXperimentation Progression (XPro) model to explain experimentation adoption and implementation in software startups. It presents the process consisted of a series of steps that, ideally, a software startup team should follow to implement experimentation properly: after having an idea, a software startup team may be aware of experimentation, develop the intention to experiment, then perform one or more experiments, analyze the results, and finally, act upon the conclusions obtained. However, a team may not progress from one step to the next or may perform wrong activities in a step. Based on the stages, we identified 25 inhibitors acting on the different stages that prevent teams from progressing and implementing experimentation to validate their business models.

2. Available at https://www.cbinsights.com/research/startup-failure-post-mortem/

The remaining of this paper is organized as follows: Section 2 introduces the background literature for our study. Section 3 summarizes the related work. Section 4 presents the research methodology applied in this study and Section 5 the results obtained. In Section 6, we discuss the results, comparing them with the literature. Finally, Section 7 concludes the paper and presents possible future work.

2 Background

In this section, we describe the core concepts of our study. First, Section 2.1 discusses what a software startup is and Section 2.2, what experimentation is in the context of software engineering. Then, Section 2.3 displays the importance of experimentation to software startups given their innovative nature. Since experimentation can be viewed as a process innovation, Section 2.4 presents the concept of innovation diffusion applied to software process innovations.

2.1 Software startups

Despite a growing scientific interest in the topic, there is still no consensus on the definition of "software startup" [19]. In a systematic mapping study (SMS) on the topic, Berg et al. [19] compared the terms used to describe these companies in the scientific papers. They observed that in those studies published during the 2013 to 2017 period, there were no agreed-upon concepts used by the papers to characterize startups, and the themes most used were innovation, uncertainty, and small team. In a 2016 research agenda paper, Unterkalmsteiner et al. [1] stated that software startups "develop innovative software-intensive products under time constraints and with a lack of resources, and constantly search for sustainable and scalable business models."

Based on the reviewed definitions, it is evident that the innovation concept is essential to characterize a software startup. The term innovation also has "different definitions, classifications and/or types of innovations at the firm level" [20]. In a seminal work, Garcia and Calantone [21] reviewed the marketing, engineering, and new product development literature and described the term "product innovativeness" to measure the "newness" degree of an innovation. The authors argued that such an aspect represents a discontinuity in the status quo regarding marketing, technological, or both aspects. Here, the marketing perspective refers to the newness of a product to the market. More recently, Purchase et al. [22] used a similar classification: technical, commercialization, and ambidextrous. Based on Garcia and Calantone's product innovativeness concept, Melegati and Wang [23] analyzed the papers covered by Berg et al.'s SMS. They classified the startups studied according to the nature of their innovation (technological and/or marketing). They concluded that the Software Engineering literature did not differentiate these two types of innovation, but there is a prevalence of the studied software startup companies which innovate from the marketing perspective. That is, these companies use consolidated technologies, like Web development or mobile apps, to create novel products. The authors acknowledged the difficulty in reaching a consensus on the definition, thus urged future studies to clearly state their understanding of *software startup* and clearly describe the studied cases allowing readers to interpret the results better.

Another common aspect of software startups reported in the literature is its transitory aspect; that is, a startup is a temporary stage of a new organization towards creating a new sustainable business. In one of the first scientific papers on the topic, Crowne [24] proposed a model consisted of three phases: startup, stabilization, and growth. Klotins et al. [25] built an improved and more detailed version with the following phases: inception, stabilization, growth, and maturity. In the first phase, a startup's goal is to develop the first version of the product; in the stabilization phase, based on customers' input, the product is developed further and prepared to scale; in the growth stage, the goal is to achieve the desired market share and; finally, in the last phase, the startup transitions into an established organization. This temporal limitation idea is also present in Steve Blank's definition. According to the author, a startup is "a temporary organization designed to search for a repeatable and scalable business model" [26]. This pursuit for a business model, along with the liability of newness, is a fundamental difference to established companies [2].

Along these lines, we use the following definition throughout the paper: software startups are organizations looking for a repeatable and scalable business model for an innovative product or service they develop where software represents a core element. In this regard, repeatability is often associated with desirability (if customers want that product), feasibility (if it is functionally possible), and viability (if it is sustainable concerning the costs of building it and the revenue from offering it to the market) [27]. Meanwhile, a scalable business model is flexible, and the use of more resources leads to increasing returns [28].

It is worth emphasizing that software startups are not stand-alone entities. They operate in highly dynamic business ecosystems composed of other startups, established companies, competitors, incubators/accelerators, governments, etc. [29]. Therefore software startups are frequently under multiple influences, which increases the uncertainty they face. This aspect is one of the defining characteristics of software startups [30].

2.2 Experimentation

Experimentation is an overloaded term with several meanings in different or even in the same research field. In the Innovation and Entrepreneurship disciplines, experimentation is described as "a form of problem-solving" [31] based on trials and errors [31], [32]. In Software Engineering, the term was first used to enforce empirical methods in scientific studies, as seen in books by Wohlin et al. [33] and Juristo et al. [34].

More recently, the term has been used to describe different techniques ranging from prototypes in startups to controlled experimentation [35] as well as problem or solution interviews [36]. In this sense, a commonly used variant is "continuous experimentation" to stress the constant use of experiments (e.g., [37]) related to the concept of continuous software engineering [38]. Other term variants stress the link with software like experiment-driven software development [39] and outcome/data-driven development [40].

In our study, we adopted the definition of experimentation as an approach characterized by continuously identifying critical product assumptions, transforming them into hypotheses, prioritizing, and testing them with experiments following the scientific method in order to support or refute the hypotheses [36]. It is essential to highlight, though, that the concept is diverse from the scientific meaning of experiment referring to a broader sense of data-driven rather than opinion-based decision making [36], [41] with the goal to make well-founded decisions and reduce the risk.

2.3 Experimentation in Software Startups

In the context of software startups, experimentation is linked naturally to the innovative nature of these companies, as its purpose is to validate that the company is able to create a repeatable and sustainable business model based on its innovative product/service. The innovative nature brings higher uncertainty to startup development, and experimentation is a way to tackle the uncertainty. Research shows that experimentation could lead to more efficient use of human and financial resources in uncertain and innovative endeavors. For instance, Thomke [42] argued that "no product can be a product without having first been an idea that was shaped, to one degree or another, through the process of experimentation." According to Andries et al. [7], "simultaneous experimentation implies lower initial growth levels, but facilitates long-term survival by enacting variety in a resource-effective manner." Lynn et al. [43] proposed a model for fast team learning in new product development, including the use of experiments, tested it with 171 teams and concluded that these teams launched products quicker and with an increased probability of success. In an attempt to provide scientific ground to the Lean Startup approach, Frederiksen and Brem [9] argued that the methodology had a "clear and explicit emphasis on experimentation over long-term planning." In summary, such an approach would help founders understand better potential customers and the market they want to operate in without spending unnecessary time and resources to develop a full-fledged product.

Two factors explain why experimentation is particularly suitable in software startups compared with startups creating non-software innovation: cost and compatibility with current practices.

First, software plays a prominent role in these companies, and its development has a lower cost compared to other sectors such as hardware building or biotechnology. Software startups can build products in shorter periods and at lower costs. Therefore, creating software prototypes for experimentation is fast and cheap, so is modifying software products as responses to experimentation results. This advantage is even more substantial with the advent of cloud solutions that made it easier and cheaper to perform experiments [12]. For instance, cloud platforms allow websites to be deployed with few clicks rather than software startups handling their own infrastructures or renting space in physical data centers.

Second, experimentation is closely related to the idea of frequent customer feedback that is strongly advocated in agile methodologies [44]. Since agile practices are well-known among software developer teams, it is reasonable to expect that software startups are keener to adopt experimentation compared to companies in other segments that demand stricter development processes.

Different experimentation techniques are expected to be used in distinct stages of a startup. Fabijan et al. [45] grouped feedback techniques according to product development stages: pre-development, development, and post-development. For the first stage, examples of experimentation are interviews, observations, surveys, or online ads; for the second, prototype testing, operational data, and developers as customers; and finally, in the third stage, A/B testing or social network data. Since it is possible to map the product development stages to the software startup stages mentioned above [46], the experiments suitable for one product development stage can be applied in the corresponding startup stage. For instance, in the inception stage, experiments are those from the pre-development stages, such as interviews or surveys. In this study, we mainly focus on experiments that could help a startup reach its final goal: find a viable business model for an innovative product. These experiments should evaluate key assumptions made on the business model, e.g., customers' willingness to use the product, usefulness of proposed features to targeted customer segments, solution's technical feasibility, team's capability to develop or distribute the solution, or its dependency on external partners from the startup ecosystem.

2.4 Software process innovation diffusion

Since "an innovation is any idea, practice or object that is perceived as new by the adopter" [47], experimentation can be viewed as a novel approach to develop an innovative software-intensive product or service based on experiments [36], [40], [48] rather than building the whole solution based on a pre-defined plan without validating the assumptions. Therefore, experimentation usage in software startups can be regarded as an innovation adoption phenomenon. A seminal model in this regard is Rogers' theory of diffusion of innovations [47]. Within the theory, the author described an innovation-decision process consisted of five stages. The first one, knowledge, occurs when a decision-making unit is exposed to the innovation. Persuasion happens when the decision-making unit develops an opinion about the innovation, either positive or negative; then, a decision is made to adopt it or not. Implementation occurs when the artifact is put into use. In the confirmation stage, an individual "seeks reinforcement of an innovation-decision already made, but he or she may reverse this previous decision."

Fichman [16] differentiated between individual and organizational adoption of an innovation and reviewed adaptations needed to the theories developed for the individual adoption to be extended to the organizational level. Regarding organizational adoption, a seminal model in the context of information systems (IS) is proposed by Cooper and Zmud [18]. According to the authors, the process consisted of six stages: initiation, adoption, adaptation, acceptance, routinization, and infusion. In initiation, the need for a change is identified, and a suitable innovation is found. Then, in the next step, a decision is made to adopt or not the innovation. In adaptation, it is adapted to the organization's context needs; in acceptance, the organization's members start to use the innovation; in routinization, its use becomes a regular activity. Finally, in the infusion stage, the usage increases in a comprehensive manner leading to better results. One way to split these stages is in adoptive behavior (initiation, adoption, and adaptation) and post-adoptive behavior (acceptance, routinization, and infusion) [49].

An aspect that has raised a particular interest in IS research has been technology adoption. In this regard, a seminal work is the Technology Adoption Model (TAM) [17]. According to the model, the intention to use a new technology is influenced by the perceived usefulness and its perceived ease of use. The intention to use then can or cannot lead to usage. In an extension of the model, Venkatesh et al. [50] concluded that this transition from intention to use to final use is influenced by subjective norm, image, job relevance, output quality, and result demonstrability.

3 Related work

In the Software Engineering literature, several studies focused on or touched upon the adoption and implementation of experimentation in the development process. Few studies, though, treated this problem in the context of software startups. In this section, we review these studies showing the gap our study aims to fulfill.

First, several studies focused on how software development teams or companies employ experimentation. Bosch [41] investigated how a large company used experiments in product development. The author observed three main characteristics: a frequent deployment of new software versions, the central role of usage data in the development process, and the focus on innovation and idea testing to improve customer satisfaction and revenue. Several other authors proposed models to describe how this process is or should be performed. Olsson and Bosch introduced both the HYPEX (Hypothesis Experiment Data-Driven Development) [51] and QCD (Qualitative/quantitative Customer-driven Development) [52]. Fagerholm et al. [48] proposed the RIGHT model (Rapid Iterative value creation Gained through High-frequency Testing). A common ground of all these models is a process consisted of identifying hypotheses regarding the product, designing experiments, executing them, analyzing the results obtained, and based on that, derive learning that updates the hypotheses [53].

Second, other authors investigated the current status of experimentation in software development and discussed why these teams do not use it more often. Ros et al. [54] performed a systematic mapping study on continuous experimentation that included 62 scientific papers. The authors identified ten research topics and, according to their analysis, only three of them are supported by some empirical evidence: experiment process, infrastructure, and challenges. The first topic, experiment process, relates to processes used to perform experimentation as described previously. The second topic, infrastructure, partially overlaps with the previous one since it discusses "system and software architecture, roles required, and organizational culture" necessary to conduct experiments. Finally, challenges are the most frequent topic among the surveyed studies. The authors classified them in four types: *technical*, e.g., enabling continuous deployment to do experiments online or offline easily; *statistical*, regarding difficulties while employing statistical methods; *management and organizational*, such as adopting experiments; and *business*, such as prioritization and lack of users. Lindgren and Münch [36] explored the state of the practice of experimentation in the software industry. The authors performed a qualitative survey with ten Finnish companies consisted of thirteen interviews. According to them, the most significant result was that the biggest challenges to its use reside in the categories, including organizational culture, product management, and resourcing, rather than in the supporting technology. In summary, several aspects have been identified so far that prevent the adoption and implementation of experimentation in software development.

Nevertheless, in the context of software startups, the number of studies is more limited. Pantiuchina et al. [15] reported a broad survey of 1526 software startups to understand the use of agile practices in this context, including a comparison between those that applied Lean Startup and those that did not. To distinguish these two types, the authors considered teams that used "the key Lean Startups concepts": hypothesis-driven, minimum viable product (MVP), and pivot. According to these criteria, only 226, 15% of the total, were considered to use Lean Startup. Nevertheless, the authors concluded that these software startups tend to use agile practices more often than the rest of the sample.

Gutbrod et al. [13] performed a multiple-case study in four German software startups that had not reached product-market fit to understand the use of experimentation in this context, the challenges these companies face, and the benefits they observe. The authors concluded that startups spend much time developing their solutions without testing their assumptions, and the main reasons for that are the lack of awareness of the possibility of early testing and the lack of knowledge and support on how to identify, prioritize, and test hypotheses.

Melegati et al. [46] investigated the enablers and inhibitors of experimentation in early-stage software startups. To achieve this, they performed a multiple-case study in four companies (three Italian and one Brazilian). They identified twelve themes grouped in the categories of individual, organizational context, and environmental factors.

On the individual level, the identified enablers were the founders' previous experience with startups and agile methodologies. Generally, if founders have founded another startup before the current one, they were aware that doing everything upfront did not work. Regarding agile methodologies, since these practices foster iterative development, practitioners considered that a similar approach would be valuable to business development. On the other hand, it was common for founders to be in love with their idea and have not considered experimenting as relevant. Besides that, they misunderstood concepts like MVP, viewing it as a prototype instead of an artifact to test a hypothesis. Such confusion led them to prepare an initial version of the final features instead of creating artifacts to test their ideas.

On the organizational level, the authors grouped the factors related to the startups, its internal and surrounding dynamics, such as its business model, practices, and tools used. The flexibility and support for experiments from software platforms were enablers, and the lack of them acted as an inhibitor. Other inhibitors in this category were the lack of resources to run experiments, a small number of users to do controlled experiments, and fear of losing clients, especially regarding B2B products.

On the environmental level, the identified enablers were the presence of accelerators and incubators or courses on the topic taught by local universities. These elements made knowledge about experimentation and practices based on it, like Lean Startup, available to startup founders and early employees. They also observed that some mentors working on accelerators and incubators came from traditional businesses and were not aware of experimentation practices. Besides that, the difficulty of getting capital was an inhibitor.

Although previous studies have reached important results, the research on the limited usage of experimentation in software startups is still in paucity. The studies focused on a deep understanding of a small number of cases restricted to specific geographical locations and business domains. Also, the results focused on listing enablers and inhibitors, not aiming to propose a comprehensive understanding of the adoption and implementation of experimentation.

Regarding diffusion and adoption of software process innovations, there are some studies in the literature. On an individual level, Riemenschneider et al. [55] tested constructs from five different models of technology adoption (Technology Acceptance Model (TAM), TAM2, Perceived Characteristics of Innovating (PCI), Theory of Planned Behavior (TPB), and the Model of Personal Computer Utilization (MPCU)) to explain software developers' adoption of methodologies. The authors concluded that adoption intention is driven by an organizational mandate, the compatibility with previous working practices, and the opinions of coworkers and supervisors towards the methodology. At a team level, Senapathi and colleagues [49], [56] created and improved a sustained usage model for agile practices. In their refined model, the sustained usage of agile is determined by three aspects: agile team factors (experience, mindset, coach, result, and demonstrability), technological support (agile practices and tool support), and organizational factors (top management support methodology champion, and organizational structure). Mangalaraj et al. [57] investigated the acceptance of extreme programming. They identified factors to the organization-wide acceptance of the innovation divided into five categories: individual, team, technology, task, and environment. However, one can argue that experimentation is a different approach for software development compared to requirement-driven [40]. Thus, one study focused on this more complex context would be valuable to understand if these theories would be enough to explain the (lack of) adoption and implementation of experimentation in software startups.

4 Research Methodology

To achieve our goal of a more comprehensive understanding of the experimentation adoption and implementation in software startups, we employed a qualitative survey method. Jansen [58] defined this method in social research to analyze "the diversity of member characteristics within a population." The author contrasted this method with statistical surveys, whose goal is to draw general conclusions about the population based on a sample. This difference is reflected in the sampling strategies. While in a statistical survey, the research should aim at a probability sample, that is, based on the frequencies observed or estimated for the whole population, in a qualitative survey, the sample should be diverse to cover all the varieties of the phenomenon [58].

Specific to our study, the population is software startups. Given that the lack of experimentation increases the probability of software startups' failure, as we argued previously, a reasonable research approach is to survey failed software startups to understand if and why they have not used experimentation.

Since failed startups do not exist anymore, to reach them becomes a challenging endeavor. One way to overcome this challenge is to look for postmortems, texts available online, reporting the failure of a startup. There are several compilations of such descriptions available online.

4.1 Data collection

Although the most common way to collect data in statistical or qualitative surveys is by questioning people, observing artifacts is also a valid approach [58]. As the data source, we used the CBInsights[3] compilation of startup postmortems. The company is well-known among practitioners for collecting and analyzing data about startups. Its reports have been used as a source for major outlets such as *The New York Times*[4] and *The Financial Times*[5]. With such a reach in the industry, it is reasonable to believe that CBInsights data represents a good startups sample. The company put online its first compilation of startup postmortems in 2014 with 50 cases, and since then, it has updated the list regularly. Several studies have employed this dataset. For instance, Klotins et al. [4] used a compilation published in 2015 to investigate how startups used software engineering practices.

The compilation consisted of a list of failed companies or products and links to postmortems. The postmortems did not have a unique structure, ranging from a founder blog post regarding the company's failure or its history to a press article describing the closure repercussions. For the current study, we used the 2019 second update dated 2019-06-19, accessed in August 2019, with 311 postmortems.

Based on the CBInsights list, we built the data corpus to have the startup names, descriptions, and working links to the postmortems. We obtained the startup descriptions mainly through Crunchbase[6]. If a company was not present in this database, we looked at a startup social network called AngelList[7]. The third option to obtain the company description was to read the postmortem and extract it from there. If it was not possible to get a description after all these attempts, we discarded the postmortem and did not include it in the following steps. Regarding the postmortem, we followed the link provided to check if it was still available. In case the link was broken, we used WebArchive[8], an archive of Internet websites, to get a link to the last working version available.

Since the compilation comprehends all types of startups, the first step was to select software startups from the data corpus. Using the software startup definition adopted in this study, two authors separately analyzed all the company descriptions and decided if a company was a software startup or not. The postmortem was also checked for this purpose when deemed necessary. In the case of disagreement, the third author analyzed the case. To operationalize this step, we came up with the following two criteria:

- the startup has to build software to deliver its vision, that is, the software is at the core of the business model; and
- the idea should be innovative, proposing either a technological or marketing disruption or both.

Applying the first criteria, we excluded startups that developed hardware and software since this aspect may impose different types of challenges for experimentation that is not strictly relevant to software startups. For instance, hardware startups may not perform experiments as cheaply as a software startup. Besides that, except for special cases like embedded systems, developers can deliver newer versions of software throughout the solution lifetime, while hardware, as Fredriksen and Brem [9] argued, "at a certain point the specifications of a physical product have to be locked down and be substantial enough that a form of end is seen." Regarding the second criteria, the innovativeness should be evaluated considering the time when the development started since, in some cases, a similar product is available in the market at the time of this study, which may render the idea under evaluation less innovative. Also, the second criterion is satisfied if the software itself is not innovative, but the way the startup employs it is. Finally, an existent product or service proposed in a different or new market was also considered innovative.

In the first analysis, the two authors agreed on 276 out of 311 startups: an agreement rate of 89%. The third author further analyzed the 35 disagreed startups. As a result, we identified 259 companies as software startups. Examples of excluded companies are WOW Air, a low-cost airline, and Faster Faster, a lightweight electric motorcycle manufacturer. Hardware startups such as Seven Dreamers laboratories specialized in healthcare devices using AI and robotics, although having a component of software development, were also excluded in this step.

The postmortems consisted of a heterogeneous set of documents, including founders' blog posts, company statements, and media news about the startup failure. For instance, some statements include instructions on migrating to a different service, a notice about ending operations, or thank-you notes to customers and investors. To get an insider's view of inhibitors to experimentation, we considered only the founders' statements portraying the company dynamics valuable for this study. Therefore, in the last step, we removed the startups, which postmortems consisted only

3. https://www.cbinsights.com/research/startup-failure-post-mortem/
4. https://www.nytimes.com/2019/02/10/technology/new-wave-unicorn-start-ups.html
5. https://www.ft.com/content/65f08660-a762-11e9-984c-fac8325aaa04
6. https://www.crunchbase.com/
7. https://angel.co/
8. http://web.archive.org/

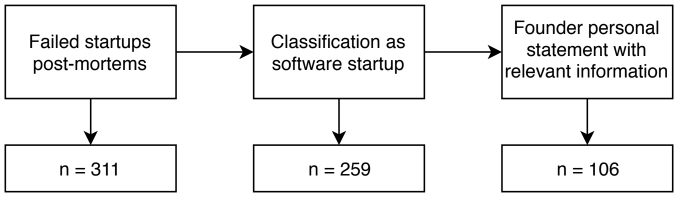
Failed startups post-mortems
Classification as software startup
Founder personal statement with relevant information
n = 311
n = 259
n = 106

Fig. 1. Selection process

of media news articles or statements without the founder's viewpoint on the company's life or failure. The final set consisted of 106 failed companies. Fig. 1 summarizes the selection process.

We stored all the 106 postmortems to be analyzed. In some cases, the link led to a statement that was part of a larger piece of data, e.g., a series of posts, then all pieces were added. The final data-set contained around 215,000 words. To better describe our sample, we extracted some information for each startup, either from the postmortem or other available sources, as mentioned in the previous section. Since all the startups have closed, not all information was available online. We made publicly available [59] all the links considered, including demographic information of each company that was considered a software startup.

4.2 Data analysis

There are various techniques available to synthesize evidence from empirical studies. Even though the postmortems are not scientific studies, some authors (e.g., [60], [61]) used multiple-case synthesis techniques on this type of data. In this study, the data analysis followed the logic of thematic synthesis of multiple cases. Cruzes et al. [62] performed a comparison between thematic, cross-case, and narrative synthesis. The authors recognized that thematic synthesis is a suitable method whenever "one attempts to incorporate a large number of cases into a single synthesis", which is the case in the current study. Thematic synthesis draws on thematic analysis principles that "is an approach that is often used for identifying, analyzing and reporting patterns (themes) within data" [63]. To apply this technique, we followed the steps proposed by Cruzes et al. [63]. Following our study's research objective and research questions, the unit of analysis is software startup.

The initial data analysis step is to read the whole data to get immersed in it. Such a step is essential to start having ideas or identifying possible patterns [64]. We conducted this step in parallel with identifying postmortems that had information useful to our research goal. In the next step, we coded specific segments of text that were related to the research questions. Codes are "labels that assign symbolic meaning to the descriptive or inferential information compiled during a study" [65]. In this step, we followed an inductive approach, the so-called open coding [66], that is, the codes were created based on emerging concepts while the text was read and without a previous set of codes.

In the next steps, the recommended course of action is grouping codes into themes and then in higher-order themes. In our study, though, the large amount of data generated many codes in the first step. Then, we grouped them

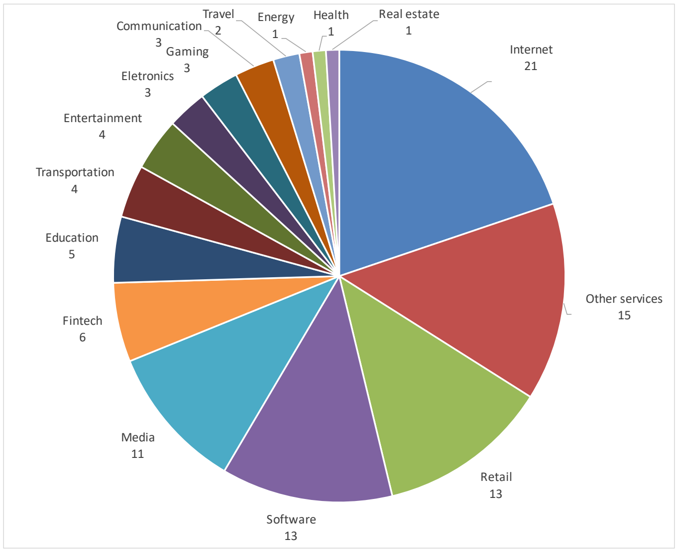

Fig. 2. Number of startups by business domain.

according to which research question they would help to answer. We identified themes that represented stages in the adoption and implementation process and inhibitors that prevented teams from reaching these stages. Then, within each of these top-level categories, we created intermediate themes grouping related codes. Of course, during these steps, we changed, merged, or even added new codes. While creating intermediate themes, some codes were also moved among different high-order themes. This process does not follow a strictly sequential set of steps but was based on an iterative process.

The first author performed the data analysis process primarily. To improve internal validity, the second author reviewed the codes assigned and proposed changes. Then, the authors engaged in a continuous discussion to reach an agreement on the final coding. A similar process was followed regarding the categorization into themes.

During this process, we used NVivo 12[9] to code the text and group codes in high-order themes. As suggested by Cruzes et al. [62], these tools improve the synthesis validity by keeping a clear chain of evidence from data to themes.

5 Results

The analyzed sample is heterogeneous, consisting of software startups that have operated in a wide range of business domains, as depicted in Fig. 2. The life length of the companies was concentrated on less than five years as expected for this type of company, as shown in Fig. 3, which reports the results approximating to the closest number of years. Unfortunately, for five startups, it was not possible to determine the length of the active period. The reported period generally indicates the whole company's existence even it had tried different products before. Therefore, this data should be taken as an approximation. Table 1 shows the geographical distribution of the software startups analyzed. The majority was located in the USA (77 out of 106).

9. https://www.qsrinternational.com/nvivo/home

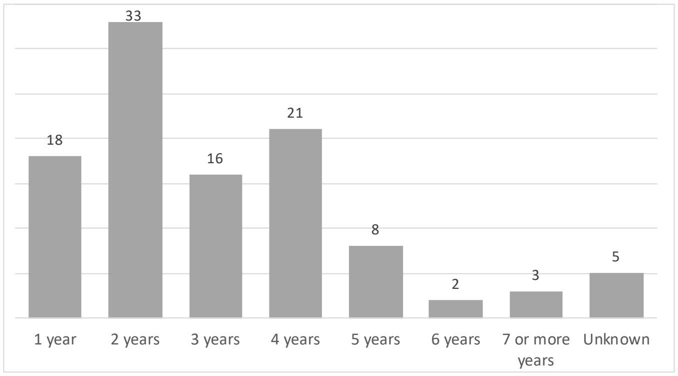

Fig. 3. Number of startups by the period of activity.

TABLE 1
Number of startups by country.

Country	Startups per country
USA	77
UK	7
France, India	3
Canada, Denmark	2
Australia, Belgium, Brazil, Chile, Finland, Germany, Hong Kong, Hungary, Ireland, Israel, Poland, Singapore	1

The identified themes revealed a complex adoption and implementation process consisted of five progression stages. We named the overall model as eXperimentation Progression (XPro) and presented it in Fig. 4.

The first step to adopt and implement experimentation is the founders' or founding team's awareness of it. From now on, we will use only the term *founders* to represent founders or a founding team. In the next step, the founders may decide to perform experiments to validate their idea; that is, they may develop the intention to experiment. Then, they will perform one or more experiments. Finally, they will analyze the results and then act based on the conclusions.

It is essential, though, to highlight that such a model is a simplified representation of reality. Many software startup products are complex and based on several assumptions, and teams may have diverse approaches for each of them. Besides that, the reality is not as linear as the model. For instance, in many startups, although a version of the product was not regarded as an experiment, the usage metrics and founders' observations could be analyzed as experiment results. Therefore, throughout our analysis, we considered this aspect while forging higher-level themes. Having that said, it is not valuable to analyze which startups reached each stage. Such a classification would be imprecise and not representative of reality. For instance, since products are based on several assumptions, startups may behave differently for each one, performing experiments about some but taking others as validated. Another factor that reinforces this argument is the data characteristics. It is composed of a heterogeneous set of claims with different depth levels (quantity and quality of details) and breadth (time interval described), making a strict comparison of startups inadequate. Indeed, the data strength is the number of startups analyzed and the comparisons it allowed among different companies, featuring inhibitors of several frequencies.

For each of the above-described stages, we identified a set of inhibitors that may hinder the team from reaching the next step, and the actions teams ended up performing instead of the expected behavior. Table 2 presents the inhibitors identified. We classified the identified inhibitors according to the three aspects specified in Melegati et al. [46]: individual (I), organizational (O), and environmental (E). At the individual level, we group inhibitors that emerged from tendencies or biases from team members as individuals. Then, at the organizational level, we gather inhibitors related to the internal arrangement and dynamics close to the startup operation. Finally, at the environmental level, we group inhibitors from the external ecosystem in which the company runs, including law, economic, social, and political systems. Like Cruzes et al. [62], we used quasi-statistics to bring forward the most frequently occurred inhibitors for each stage. Fig. 5 shows the distribution of inhibitors from the three aspects for each stage. In the awareness of experimentation, there are few inhibitors, and they regard environmental aspects. Then, in the intention to experiment, there is the biggest number of different factors, besides one regarding environmental factors, inhibitors relate to individual and organizational aspects. To proper experiments, there are still some inhibitors, but all are related to organizational context factors. Finally, in the last two stages, proper analysis and consideration of results, inhibitors are only regarding individual aspects.

In the following sub-sections, we describe each stage in detail, presenting the identified inhibitors and the alternative actions observed in the startups. Additionally, the consequences of not experimenting are shown in the last sub-section.

5.1 Awareness of experimentation

The first step to adopt experimentation is to be aware of such a possibility. The problem of lack of awareness was rarely observed (only three startups).

5.1.1 Inhibitors

Environment factors. The inhibitors identified for this stage were grouped under environment aspects regarding **lack of accessible knowledge**. The coded excerpts showed a lack of guidance and the lack of publicity of failed startups. Such awareness could show newer companies that failure is possible and that they should avoid the same mistakes.

5.2 Intention to experiment

Once founders are aware of experimentation, the next stage is to develop an intention to perform experiments to validate the idea. There were no explicit excerpts in data showing this desire, but inhibitors that prevent teams from reaching this stage and its alternative, the lack of experiments, demonstrated its existence.

5.2.1 Inhibitors

In this theme, the identified factors prevent the founders and the team from thinking about experimenting either because

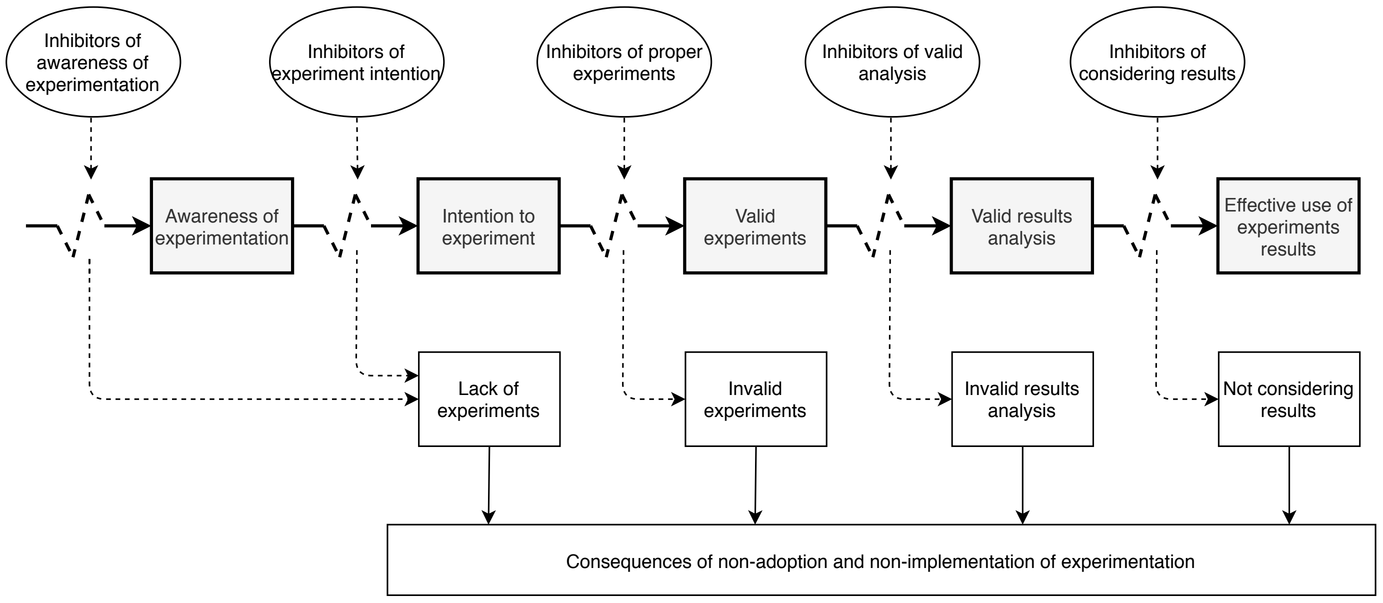
Inhibitors of awareness of experimentation
Inhibitors of experiment intention
Inhibitors of proper experiments
Inhibitors of valid analysis
Inhibitors of considering results
Awareness of experimentation
Intention to experiment
Valid experiments
Valid results analysis
Effective use of experiments results
Lack of experiments
Invalid experiments
Invalid results analysis
Not considering results
Consequences of non-adoption and non-implementation of experimentation

Fig. 4. XPro: a progression model of experimentation adoption and implementation in software startups.

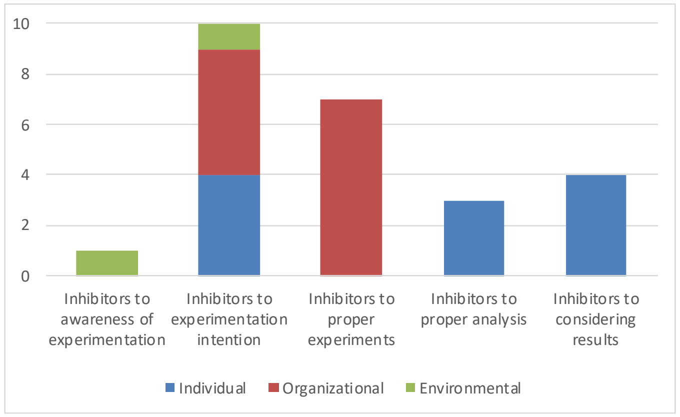

Fig. 5. Number of identified inhibitors by stage divided by individual, organizational, and environmental aspects.

they do not believe they are necessary or it is not possible to perform them.

Individual factors. The most frequent codes are related to this category. First, in 42 startups, there was an **over-focus on presenting a better, or even a perfect product**. The reasons include a fear that an unfinished product will not bring customers or the team had an idea of a solution without a clear problem in mind. An interesting element came from an Austrian founder, Michael Bohanes, from Dinnr, an online ingredient delivery service where the user could select a recipe online, and the ingredients needed would be sent to the user's home. He partly blamed his country's culture for his failure. According to him, it is focused on *"studying"* and *"preparing,"* then the team spent much time planning the product without getting feedback from customers.

Second, in 40 startups, there was an evident **over confidence on own ideas** leading the startup not to doubt the need for the product and focus on its implementation. For instance, Frans Ekman was the founder of Disruptive Media, which idea consisted of an online service to document the story of people's lives "in a collaborative way with your friends." He wrote: *"we thought it was a killer idea to quickly build up a social network that so many companies had tried to but failed."* In this regard, many founders (9 startups) expected user behavior change. They imagined that the product would have made the user change their behavior, and, in most cases, that did not happen.

Third, in 28 startups, the team **considered the idea validated** and took different signs as indicators of the idea value without really checking the average user or customer. For instance, media appearances, business contest wins, and investments received induced the founders to believe they had a good idea. The most common code in this category is the founder as a user, implicitly taking herself as the representation of customers and building the product to meet her own needs. If such code was found, this extrapolation was not correct; there were not many customers interested in the final product, and, consequently, the startup failed.

An interesting element mentioned was the **developers bias towards implementation** and their discomfort with changing directions fast. According to Ben Yoskovitz, founder of Standout Jobs, a company that provided a suite of web-tools for recruiting, *"the fact is that having a specification and building to that specification is a lot easier for a developer; constantly changing requirements [...] makes a developer's job harder."* Developers generally measure progress with the number of lines of code produced. The founders of Devver, a startup that provided cloud-based tools to make Ruby developers more efficient, wrote on the startup's blog: *"Hackers are passionate about, well, hacking. And so we tend to measure progress in terms of features completed or lines of code written. Of course, code needs to be written, but ideally, a startup would have a founder who is working on important non-technical tasks: talking with customers, measuring key metrics, developing distribution channels, etc."*

Organizational context factors. Several elements are grouped here, although with a lower frequency for each.

TABLE 2
Identified Inhibitors for Reaching Different Stages of Experimentation Adoption and Implementation. The letters indicate inhibitor nature: I for individual factors, O for organizational, and E for environmental.

	Theme	Number of startups	Description
Inhibitors to awareness of experimentation	Lack of accessible knowledge (E)	3	Unawareness of experimentation or its importance prevented some founders from testing assumptions. Founders are not aware of failed cases given the lack of publicity to them which leads startups not to consider this possibility.
Inhibitors to experimentation intention	Over-focusing on product and its perfection (I)	42	Founders aim for a perfect product to be offered to customers. In this scenario, they see experiments as not ready versions that could threaten the business.
	Over confidence on own ideas (I)	40	Founders strongly believe that their ideas are good and testing them is not necessary. In some cases, they expect a change on user behavior.
	Supposedly validated idea (I)	28	Several facts may make founders believe that their idea is valuable without further experiments. Examples range from an early high traffic to the product usefulness to the founder.
	Issues regarding large amounts of capital (O)	6	For capital-intensive products, founders are led to think they need many users to keep the service running and focus on that from the beginning. Another problem is actually large amounts of capital that make founders spending to build the product instead of following a more cautious approach of validating users' needs.
	Investors preferring the fast growth of the startup (E)	5	Investors usually have a pool of startups and know that most of them will fail, and they want those that survive to make money. Therefore, they influence startups to grow as soon as possible.
	Scarcity of technical resources (O)	5	Developing and running experiments demand technical resources that are scarce in software startups.
	Over-focusing on customer base growth in early phase (O)	3	Some products are based on the scale so founders are led to grow the number of customers in a early phase rather than testing their ideas.
	Developers bias towards implementation (I)	2	Developers prefer clear guidelines on what to do instead of the changing aspect of experimentation. They measure their progress by lines of code.
	Demand for a reliable product (O)	2	Mission-critical products demand a more ready version to be sold. Experiments are deemed as imperfect products.
	Time pressure (O)	1	Doing experiments takes time that some startups do not have because of constrained resources.
Inhibitors to valid experiments	Demanding B2B customers (O)	12	Selling to large companies is complex and time-consuming. These customers also demand more complete products.
	Complex multiple-sided business models (O)	12	Products that reach different types of customers (such as platforms) present a challenge for experimentation creation since it is hard to test about one customer type without the other types.
	Lack of development resources (O)	10	The lack of developers hinders the use of experiments. For instance, startups that outsourced their development struggled with experimentation.
	Changing business context (O)	6	In a fast changing market, it is hard to create experiments which results will be useful after a period of time or in a different context.
	Dispersed customer base or usage (O)	3	These challenges are related to the quality (how representative) and quantity of experiment subjects. For instance, it is hard to develop comprehensive experiments valid for several customer segments or if the solution is used for a short period in a year.
	Concern on negative user perception of the product (O)	2	Some founders reported difficulties to make experiments without risking on how customers perceived their products through the user interface and experience.
	Highly coupled assumptions (O)	2	Business is complex and many assumptions should be tested. Sometimes it is hard to separate the effect produced by different experiments.
Inhibitors to valid analysis	Confirmation bias (I)	2	Founders often tend to analyze the results in a way to support their claims instead of taking a neutral view.
	Choice-supportive bias (I)	1	Tendency to stick to the choice previously made even if the results are not good.
	Misled by the limited number of users (I)	1	Some product may need a large pool of users to be attractive. Some experiment results are ignored because founders attribute the results to the lack of users.
Inhibitors to considering results	Resistance to change (I)	5	Several factors at individual level make founders ignore experiment results. Examples are too much time spent to quit, desire to look successful, and stubbornness.
	Fear of losing customers (I)	1	An experiment result may indicate that a product should be abandoned. But it may already have active users and losing them is hard.
	Fear of losing revenue (I)	1	Abandoning a product may represent a loss of revenue which is vital for a resource-constrained company.
	Obfuscated by the number of new users (I)	1	Founders may ignore the experiment results because new users continue to be acquired, for instance, as the result of marketing strategies.

First, there are **issues regarding large amounts of capital** like the cost to run some products (e.g., expensive operation). For instance, GameLayers was producing an innovative Passive Massive Multiplayer Game that demanded many people to handle the complex task of building and supporting such a game.

Second, there is a **scarcity of technical resources** that makes running experiments harder or too demanding for an already time and resources constrained team. For instance, keeping up-to-date with third-party libraries, more time needed to change a platform built in-house, or underestimating the coding effort make teams focus on software development instead of considering experiments about the customer. Still related to technical inhibitors, there are mission-critical products that cannot fail and **demand a reliable product** to be sold. Such a constraint prevents the creation of experiments that only partially presents some features.

Some products demand a big scale to be profitable, leading founders to **over-focus on customer base growth** even though the idea is not validated yet. For example, Take Eat Easy was a food delivery service for highly-rated restaurants based on a platform connecting customers, couriers, and restaurants. Such a business model is heavily dependent on the number of individuals for each role; that is, the attractiveness (financially worth) for couriers will depend on the number of customers and restaurants. In these cases, it is hard to do valuable experiments without a minimum number of users.

Finally, in one startup, it was observed a time-related inhibitor where the **time pressure** present in the startup context made them abandon experiments. Frans Ekman from Disruptive Media, an online service to document your whole life, said: *"In the beginning we were kind of on the right track with mockups and wireframes, even doing usability tests, but the time pressure made us cut corners and just get it out."*

Environment factors. The codes related to these factors were the least common. They were grouped in one category represented by **investors preferring fast growth** because having more users makes it easier to attract investors. Since a venture capital bet has a low probability of success, investors look for companies that, if successful, will pay for those that were not and generate a profit on the overall portfolio. Therefore, startups that received investments are compelled to grow their customer base as soon as possible, and an experiment-oriented approach is slow. For instance, UDesign, an online tool to create unique clothing pieces using mathematical simulation, reported pressure from investors to grow the customer base. In summary, startups tend to concentrate on getting customers instead of experimenting on the market.

5.2.2 *Alternative: Lack of experiments*

In this category, we grouped examples of startups that did not do experiments. The most common example was the **focus on getting customers**. As Kevin Gibbon from Shyp, a service to ship packages, mentioned: *"growth at all costs is a dangerous trap that many startups fall into, mine included."*

Because of not doing experiments, teams realize, usually late, that they based their businesses on **wrong assumptions**. Kyle Hill, from HomeHero, a platform to find caregivers to elderly people, mentioned: *"we overestimated the ability of health systems and insurance companies to pay for non-medical home care."*

5.3 Valid experiments

Once the founders develop an intention to experiment, they should create and execute experiments. Nevertheless, the intention is not enough to perform valid experiments, which would lead to reliable results. Another set of inhibitors is responsible for leading founders to do invalid experiments.

In the analyzed startups, we observed several valid experiments performed by startups. An interesting example is Vatler, whose idea was to build a mobile app to make parking easier. As the founder Hamza Chahdi told: *"Getting no answer to our emails and calls, we decided to experiment our model to see if it could work. Within 2 weeks, we had 4 restaurants on-board. We were parking their customers only on weekend nights at first to understand the needs better and iterate on our product. My co-founder and I were parking cars ourselves every weekend night."* Another example was a crowdfunding campaign, that is characterized by the need for a minimum pool of interested customers to build the product. Such an offer can be considered an experiment that probes the customers' desire for a product.

5.3.1 *Inhibitors*

In this theme, the identified inhibitors prevent the creation and execution of reliable experiments, which lead the founders intending to test their hypotheses to perform imperfect or invaluable experiments.

Organizational context factors. All identified inhibitors were grouped under this category. The two most common aspects were **complex multiple-sided businesses** and **demanding B2B customers**, both coded in 12 startups. Regarding multiple-sided businesses that depend on different user classes to work, such diverse types of users hinder the creation of experiments. A common example is a sharing economy platform where some users must be interested in offering a service and others willing to pay for it. In these cases, it is hard to perform proper tests to evaluate the desire of one type of user without having several users on the other side. Dalton Caldwell from App.net, a startup that proposed a network of social apps, called this problem the *"chicken-and-egg issue."* In their case, it was related to developers and users of the applications. An associated problem mentioned by founders was **highly coupled assumptions** or as Andy Young from GroupSpaces, which developed web-based tools for clubs and other real-world groups, mentioned *"too many assumptions to test meaningfully."* This aspect is similar to over-focusing on customer base growth in the early phase, an inhibitor to the intention of experimentation, but regarding a different moment in the process. While the inhibitor to intention prevents founders even to consider experiments, the inhibitor to create valid experiments hinders founder that already want to experiment but find difficulties to perform valid experiments. Regarding B2B products or services, several founders complained about how hard it is to sell to big companies, including time-consuming processes for buying a product or signing a contract. This type of customer is reluctant to prototypes, which are generally used to test ideas, and only pay for ready products.

Then, several inhibitors were related to the **lack of development resources**. Different aspects were mentioned in the ten startups in this category, but, as a commonality, they regard the difficulty of creating experiments with limited development resources or a technically challenging problem. An example is outsourcing software development. As Michal Bohanes from Dinner mentioned: *"As one of my first investors later told me, you must have developers on your core team in the same room. You need to be able to iterate and execute fast. There is no slower way than doing this with developers overseas, even if your company isn't tech-heavy (such as Dinnr)."*

Although rarely mentioned, an interesting inhibitor is a **changing business context**. In this regard, the founders mentioned market changes at high speed and, consequently, a different context between the startup's beginning and later stages. In both cases, the results of experiments may change.

Regarding **dispersed customer base or usage**, a rarely mentioned yet interesting inhibitor was the problem of seasonal use. Zen99 provided tax and insurance tools to support independent workers. An issue for them to experiment was that there was a peak of tax tools use in the weeks before income tax forms' due date, followed by a drop to almost none. Such a tight opportunity window hinders the execution of a large number of experiments. Another challenge in this category is the existence of different customer segments hindering the creation of comprehensive experiments.

Finally, there were also **concern on negative user perception of the product** regarding how to perform experiments without risking the product image and user experience. For instance, Adam Zerner from College Inside View, an online tool to help applications to universities, mentioned: *"I think that users (mostly unconsciously) look for signals of competence. [...] A good design is a signal of competence."*

5.3.2 Alternative: Invalid experiments

In this section, we present the most common examples of experiments that, not being performed systematically, lead to wrong conclusions.

The most common code (12 startups) in this category was that several teams focused on **asking direct questions to users**. There are several issues with this approach. First, users may concentrate on their problems, and the team ends doing a solution for that specific customer or a small set of people. Jonas Bogh, from Hivebeat, an online tool to help student organizations to promote and manage events on campuses, described this problem: *"Listening to your customers is critical, but don't always let them tell you what to build. Your customers know the problem, but in most cases, they don't know the right solution to the problem (remember the story about Henry Ford and the horses?). That's your job to figure out. We didn't realize that and we therefore ended up building too many features with no real vision for where the product was heading."* Second, a simple set of questions may not get the real intention from those potential customers. Michal Bohanes from Dinnr gave an example: *"[...] we committed the big mistake of presenting people with the idea and asking them if they liked it and would buy it. And when people said yes, WE [author's highlight] thought they meant 'launch it and I will buy' [but] in reality, they mean 'I'm not entirely excluding the possibility that one day [in an extreme, absurd scenario] I might be tempted to purchase a trial product from you."* This mismatch is related to the **difference between interest and paying**.

A related issue is **to use close people to experiment**. Some teams used relatives or friends in experiments, but these people might be biased to help founders for other reasons besides the product itself. An example came from EventVue, which built online communities for conferences to improve networking among participants. The founders said: *"[...] because we were basically calling on friends of friends who ran events to be our customers, we didn't learn what event organizers in general wanted or how to acquire them as customers in a scalable way with the 'private social network product."'*

The second most common factor is getting the **wrong idea of an MVP**. Jonas Bogh from Hivebeat discussed that *"however, it seems like lots of startups get the idea of MVP wrong. We did too. Instead of building a great and very simple MVP, we built an unstable product with too many features."* In summary, founders usually describe an MVP as a product with the minimum set of features that allows it to be sold. This idea does not give the notion of hypotheses checking that would be useful to experimentation as in Lean Startup, the methodology that popularized it. Within Lean Startup, the MVP is the minimum artifact to test an assumption that the team has about the customer or market, and it is not necessarily the final product. For instance, it can be a landing page or an explaining video.

Another related issue found is **spending a long time doing the experiment**. For instance, IntroNet, an online service to make introductions between people easier, took eight months to develop their *''alpha"*, which, interestingly, they had called MVE, Minimal Viable Experiment.

5.4 Valid result analysis

Once experiments are planned and executed, the results are available to be analyzed. This step is not straightforward as it may seem. Several factors may lead founders to reach wrong conclusions, e.g., extrapolating results to bigger audiences. Again, this step was not directly observed and coded in data, but the inhibitors and alternatives found demonstrated its existence.

5.4.1 Inhibitors

In this theme, we categorized inhibitors that prevented a valid analysis of experiment results. Here again, some biases were mentioned, like the **choice-supportive bias**. As Martin Erlic from UDesign mentioned *"when you choose something, you tend to feel positive about it, even if the choice has flaws. You think your dog is awesome - even if it bites people every once in a while."* He also mentioned **confirmation bias**: *"we tend to listen only to the information that confirms our preconceptions."* Finally, some products need a large pool of users, and founders are **misled by the limited number of users** and think the results could get better if adoption grows. Don Smithmier of Kinly, a social network tailored for families, called it *the distribution challenge*.

5.4.2 Alternative: Invalid results analysis

In this theme, some startups where the teams misunderstood experiment results are represented. The most common codes in this category were **initial success** and **not general results**. As an example of the first, there is an excerpt from Mark Hendrickson from Plancast, a social network that would allow users to see which events in the area any friend is planning to attend. He mentioned *"while the initial launch and traction proved extremely exciting, it misled us into believing there was a larger market ready to adopt our product."* Regarding the latter, Chris Hoogewerff from HelloParking, a company described as an Airbnb for parking, mentioned *"And though I didn't realize it at the time, doing 'xyz for parking' isn't the way to build product and discover value. The transitive property doesn't apply as well in the startup world ('If x worked for industry y, then x must work for industry z')."*

5.5 Effective use of experiments results

From the experiments' conclusions, founders should continue developing, probably performing other experiments, or reconsider their plans adapting them to the results. In the analyzed data, some teams observed the experiments' conclusions and acted based on that. A typical example was startups that closed, although still having money to spend, and returned the capital to investors. We observed seven startups like that. For instance, Kyle Hill from HomeHero mentioned: *"The good news is we are pulling out early enough, with significant cash remaining, so we can find a new path forward and continue our same mission of promoting health and wellness in the home."* Nevertheless, we also detected a set of inhibitors to achieve this step.

5.5.1 Inhibitors

Several times, founders ignore some experiment results because of different reasons. Since there are several filters before a team succeeds in experimenting, fewer pieces of text are coded in this theme. Nevertheless, they bring interesting insights.

Individual factors. Most of the elements in this theme are in this category, generally related to a **resistance to change**. The codes include: entrepreneurs should be seen as successful, gone too far to quit, not admitting that it is a bad idea. Martin Erlic from UDesign mentioned a cognitive bias called *"ostrich effect, [...] the decision to ignore dangerous or negative information by 'burying' one's head in the sand, like an ostrich."*

Organizational context factors. Here, two different classes of problems happen. First, there is a need to continue even with bad experiment results because startups have **fear of losing revenue** needed to survive or the **fear of losing the current customers**. Second, results are **obfuscated by the number of new users**, generally caused by marketing efforts. These practices lead to a steady income of interested people hiding the lack of a sustainable business.

5.5.2 Alternative: Not considering experiment results

An experiment's results may tell startups that they have to give up on their idea or pivot. Sometimes, startups do not take these attitudes quickly enough; that is, they are **slow to respond to results**. For instance, Thomas Pun from Delight, a tool to allow developers and designers to see how users interact with apps, described a pivot his startup has done inside a section titled *"we didn't pivot hard enough"*. He also said *"Even though we saw promising signs early on, we waited too long to fully commit to it. The team was constantly distracted and didn't know where the company was really heading."* This situation indicates a lack of vision shared by the startup team, leading it not to grasp the experiment purpose, not learn properly from the results, and act upon them promptly. In other cases, startups **continue even with bad results**. Mike Krupit from IntroNet mentioned *"We couldn't validate many of our hypotheses, but launched beta (MIP, Minimal Investable Product) a few months later."*

5.6 Consequences of non-adoption and non-implementation of experiments

In this theme, we gathered all codes related to facts that happened within the startups that could have been avoided if the teams had performed experiments. The most common identified element was **scaling before the right moment**. For instance, David Hyman from Bling.ly, an streaming video app for performing arts, wrote *"make sure you've built an application that is charting with high retention metrics BEFORE [original author's highlight] you give away any equity or pay for what you hope to be a large scale funnel."* Sometimes, the startup is even able to receive investment like Standout Jobs. The founder Ben Yoskovitz wrote *"I raised too much money, too early for Standout Jobs ($1.8M). We didn't have the validation needed to justify raising the money we did."*

A common consequence, though, in the founders' words, was **never found the product-market fit**. We coded the term 15 times. For instance, Chris Hoogewerff from HelloParking wrote *"we never managed to find product-market fit, and we were having trouble scaling."* Hsu Ken Ooi from Decide.com, an e-commerce website for electronics powered with best-time-to-buy prediction features, wrote *"expanding your target market, like we did by supporting more categories and building when to buy features, doesn't help you get to market-fit. In fact, I'd argue it makes it harder because there's more needs to satisfy."*

Sometimes, founders realize the startup was **not solving a real problem**. Michal Bohanes from Dinnr wrote *"this will be the number one lesson I will never forget and the absolute key to understanding Dinnr's failure - we were not solving anyone's problem."*

5.7 Reasons for failure beyond a lack of experimentation

In addition to the previously presented, we observed in data some reasons for failure beyond a lack of experimentation. Such consideration is essential to highlight that a software startup is a risky endeavor. Even if all the recommended practices are followed, the final result may not be a repeatable and scalable business. The most common factor identified was **environment issues**. For instance, in this category, there are problems with regulations like HomeHero, which proposed a platform to find caregivers for older adults. The company was getting traction in the area of San Francisco with 1200 professionals on the platform. However, a change in federal regulations would force the company to hire all the professionals to continue running. Such imposition made the business model not viable, and the startup closed.

Another interesting example is Vatler that performed an interesting experiment as described in Section 5.3, where the founders themselves were parking cars for restaurants to evaluate the need for such a service. Although the results were good, the company failed because of legal reasons. Hamza Chahdi, one of the founders, explained: *"Suddenly, two months ago, we received a phone call from the city explaining that traditional parking companies were not happy with the way we were poaching business from them and that we had to slow down our growth. We ignored this warning. Ten days later, we received a phone call from the police department telling us that our permits had not been granted and they gave us a warning because we were operating illegally in most of our locations."*

6 Discussion

The XPro model describes the adoption and implementation of experimentation in software startups similar to previously described diffusion models but extends the with specific stages. Answering RQ1, we identified the following steps: awareness of experimentation, the intention of experimenting, performing experiments, analyzing the results, and acting based on the conclusions. The first two stages (awareness and intention) are similar to what previous adoption models describe (e.g., the process of innovation-decision in the diffusion of innovations theory [47] or TAM [50]), while the latter regards implementation or post-adoption. Nevertheless, the process of experimentation adoption and implementation is more complex than what is described for agile methodologies [49], for instance. An explanation for this may be that experimentation is more a concept than a method with a defined set of practices. Even Lean Startup, which is strongly based on experimentation as mentioned before, lacks a set of clearly defined practices [67]. Another difference is that experiment results may lead to unpredictable outcomes, e.g., a completely different product or even a complete stop of the development process. Such a consequence is different from a change from waterfall to an agile approach, which is oriented to process change rather than product overhaul.

Regarding RQ2, a various number of inhibitors prevent the progression between different stages. As expected, the popularity of methodologies, such as Lean Startup, makes founders aware of experimentation-inspired techniques such as MVP. Then, the majority of inhibitor occurrences happen in the second stage: the intention to experiment. As shown in Fig. 5, it is possible to observe the predominance of individual-related inhibitors in this category, such as overconfidence on own ideas and over-focusing on the product and its perfection. Then, in the next stage, the larger part of inhibitors is related to the organizational context (such as the business model or development team). Finally, for the smaller number of startups that managed to perform experiments and analyzed them correctly, the most significant obstacle to utilizing the experiment results is human-related inhibitors.

Regarding the inhibitors to the intention of experimenting, the most observed factors were related to individuals, mainly founders' over-confidence in their ideas. In this regard, our model can be better understood drawing upon other adoption models. For instance, in the persuasion stage, Rogers [47] mentioned five aspects regarding how adopters perceive innovation: relative advantage, compatibility, complexity, trialability, observability. In the context of our study, many of the identified individual inhibitors are related to how founders perceive the relative advantage and compatibility of experimentation with the process they want to follow to implement their ideas.

Once the intention barrier is overcome, most of the issues lie in the organizational context, such as business nature. Especially multiple-sided business models present a challenging context for experimentation. In Rysman's words [68]: "a two-sided market is one in which 1) two sets of agents interact through an intermediary or platform, and 2) the decisions of each set of agents affect the outcomes of the other set of agents, typically through an externality." This definition can be extended to multiple-sided businesses. Such dynamics may imply a "chicken-and-egg issue" (borrowing the term used by one of the reviewed founders) for a startup. Besides that, as the total of assumptions grows based on the number of sides, it becomes harder to develop experiments that could evaluate one side's wish without the other, given the difficulty of emulating users on other sides.

Regarding the stage of valid analysis of the results, the inhibitors return to individual aspects. This fact may be a consequence of this experimentation step, interpreting the results obtained, being highly cognitive intense. Because of this fact, cognitive biases appear as major inhibitors at this step of experimentation implementation. We identified two, i.e., confirmation and choice-supportive biases. Mohanani [69] performed a systematic mapping study on the topic in software engineering literature and demonstrated that cognitive biases are prevalent among developers. Our findings extend the range of cognitive biases beyond software development processes and include the broader software startup context.

Finally, inhibitors to properly consider the experiment results are still on individual aspects as cognitive biases' prominence shows. This phenomenon can be a consequence of the nature of this last stage. Here, founders have to accept experiment results and act based on them, which may sometimes be against their inner beliefs.

Compared to our previous exploratory study on enablers and inhibitors of experimentation in early-stage software startups [46], this study focused on inhibitors, identified more of them, and described how they impact each step of the adoption and implementation of experimentation. The study is not restricted to early-stage software startups and is based on many cases. As a result, in addition to the newly identified inhibitors, we extended and refined some inhibitors identified in [46]. Regarding individual inhibitors, e.g., the inhibitor *founder in love with the idea* is renamed as *over confidence on own ideas*. *Understanding an MVP as a prototype* is refined into two inhibitors - *lack of accessible knowledge* and *over-focusing on product*. Two inhibitors in the organizational context category are extended: *lack of support and flexibility of software platforms* becomes *lack of development resources*, as the lack of resources to run experiments is also related to issues with large amounts of capital; *the small number of customers that make hard to run experiments* becomes *dispersed customer base or usage*. Instead, *fear of losing customer in B2B cases* is refined into two inhibitors - *demanding B2B customers* and *fear of losing customers*. Finally, regarding environmental inhibitors, *difficulty to get capital* is extended as *issues with large amounts of capital*.

In summary, our results corroborate the findings from previous research regarding the adoption of experimentation in general and in software startups in specific. In their exploration of the use of experimentation in the software industry, Lindgren and Münch concluded that it is not mature yet and that "[t]he overall impression from this study suggests that technology has a supporting role in an experiment system, and that the more significant issues lie elsewhere" [36], that is, "the technical challenges have been solved." Our results also show the predominance of individual and organizational inhibitors, not necessarily related to technical aspects in adopting and implementing experimentation. Regarding software startups, Gutbrod et al. [13] observed a focus on developing the solution rather than validating the need. The authors suggested that the main reasons were lack of awareness of such possibility and lack of knowledge on how "to identify, prioritize and test" assumptions. Such an element is related to the individual factors we observed in the intention to experimentation stage. However, our results improve the understanding showing that these aspects may happen even after executing an experiment. Still related, the focus on developing the solution was already identified as one of the key challenges for early-stage software startups [70].

6.1 Threats to validity

Runeson and Host [71] described a scheme to assess threats to validity composed of four aspects: construct validity, internal validity, external validity, and reliability. We have followed the authors' advice and addressed the validity threats throughout the whole study. Below, we discuss each aspect and how we mitigated the related threats.

Construct validity reflects "to what extent the operational measures that are studied really represent what the researcher have in mind" [71]. In this regard, a threat is the lack of agreed definitions of software startup and experimentation concepts. To mitigate this threat, we defined a shared view among the co-authors from the beginning, and all were involved in startups' identification.

A threat to internal validity is the use of data without contacting the original authors. This aspect concerns with causal relationships and a possible erroneous inference of a factor determining a consequence when, in reality, another factor not taken into account is the determinant. A key aspect in this regard is the varying amount of data for each startup. Founders presented different levels of details of their startups that hindered us from describing the startups better and evaluating if different inhibitors happen more often in contexts with specific characteristics. For instance, the information about development methods used, such as DevOps, which could improve experimentation use, was never mentioned in the dataset. Another dimension that would be interesting to evaluate was startup size, including the number of employees or customers, which had already been described as turning points for requirements engineering activities in startups [72]. However, the available information regarding this aspect was limited. Additionally, there was no systematic information about the founders' background. Generally, they presented themselves as the startups' founders without properly describing their previous background. What we did know is that, at the start of these companies, teams were small, as mentioned in several statements, and founders performed different activities from strategic decisions to bureaucratic issues, including developing software. Still, the comparison of multiple instances of the phenomenon permits a more reliable indication of a cause-effect relation. An additional threat to internal validity is that, due to the dataset limitations, we could not detect founders' motivations for publishing the collected postmortems. Some of these motivations might compromise the quality of the data because the founders may hide or mispresent their experiences for various reasons. However, our reading of these postmortems did not render the motivation as a concern.

External validity concerns how generalizable the results are. In this regard, a typical threat is a small sample size. The high number of investigated startups in this study contributes positively to the findings' generalizability to other failed startups. Still, the lack of control over a convenient sample might have been a problem. One notable aspect is the percentage of startups located in the USA. There are several reasons for it: first, the original database is built by an American company. Second, the language considered explicitly was English. Even if we gathered other postmortems online, this choice could still favor American companies' inclusion since it is expected that, when describing failure, founders use their mother languages. Besides that, as we observed in one of the selected startups, in some cultures such as Asian or European, failure is a shame or a sign of incompetence, which could hinder founders from these cultures to publish postmortems. Nevertheless, the lack of a clear difference between the startups from the USA to those from other countries leads us to believe that this is a minor threat. Regarding the other aspects of these startups, like business domain or the active period length, the diversity in these aspects diminished the threat to external validity.

We cannot generalize this study's findings to active software startups due to the sampling strategy. Therefore the study is not in the position to answer questions such as "whether active software startups use experimentation" and "what inhibits active software startups from adopting and implementing experimentation." To do so, researchers should collaborate with active startups, which generally is demanding, given the limited resource and time they have.

Besides that, our deliberate choice not to consider products that demanded hardware development did not allow us to identify inhibitors related to software development interfacing with hardware development. As mentioned earlier, we believe that there may be inhibitors related to hardware development or the interface with software development. However, we decided to focus on software development to better understand inhibitors in such a context. The inhibitors related to experimentation in hardware startups could be investigated in future work.

Reliability concerns "to what extent the data and the analysis are dependent on specific researchers." A study is reliable if the researchers make results independent of them in such a way that another group of researchers doing the same study would reach the same results [73]. In this study, we tried to make the study process as transparent as possible by describing all steps in detail. For instance, we made the links to postmortems available online. Additionally, in the coding phase, which may present a significant threat to reliability, two authors discussed the codes, excerpts, and emerged themes to mitigate the results researcher-dependency. Using two authors in the analysis process also mitigated another potential threat to reliability, which is related to the fact that the founders wrote the postmortems as general reflections. We needed to interpret the data and associate it with experimentation, which could reduce the study's reliability as data analysis was more dependent on the specific researchers.

Finally, the founders wrote the postmortems as a general reflection; we needed to interpret the data and associate it with experimentation. This interpretation reduced the study's reliability as the data analysis was more dependent on the specific researchers. To mitigate this potential threat, we have used two researchers in the data analysis process: what one researcher did was checked by the other.

Another aspect to reflect on is that our study could not identify *all* possible inhibitors. Such a goal is almost impossible to reach, given the plethora of possible scenarios a software startup can face. One noticeable limitation regarding this is that, despite the importance of a business context in which a software startup operates, we have identified relatively few inhibitors related to it. This limitation is due to the nature of the data we had and the limited possibility to obtain more information related to the business contexts of the studied software startups. However, the stages and the initial list of inhibitors identified can serve as a start in searching for additional inhibitors.

7 Conclusions and future work

Although experimentation could improve the success rate, software startups still do not use it so often. To analyze this phenomenon, we conducted a qualitative survey study using 106 postmortems from a database of materials on failed startups. Through a thematic synthesis, we developed the eXperimentation Progression (XPro) model to explain the adoption and implementation of experimentation in software startups. It consists of the following steps: awareness of experimentation, intention to experiment, execution of experiments, analysis of the results, and application of the conclusions. For each step, we identified a set of inhibitors that prevent the step from being reached and the alternative situations that may happen.

Our study is one of the first to investigate the adoption and implementation of experimentation in software startups. Our results contribute to the adoption and implementation theories regarding software process innovations such as experimentation. Compared to previous models in the literature, XPro proposes another acceptance step regarding the consequences of experiment results. Besides that, the study improved the literature regarding inhibitors, their relevance, and how they hinder experimentation in software startups. These results could inform the development of new techniques to foster the adoption of experimentation in software startups. For practitioners, the catalog of identified inhibitors acts as a list of possible problems that founders, development teams, and other stakeholders should avoid or, at least, be aware of the existence. Investors, accelerators, and other stakeholders could use the XPro model as a guide to monitor startups with which they are working. Our study has a broader implication to software engineering practice in general since uncertainty in development environments is increasing, calling for experimentation as an essential element of development processes. For instance, developing a feature will go beyond satisfying a requirement and will evaluate if the consequences of that feature match the initial expectations.

Our next step is to validate the progression model and inhibitors using primary data by conducting in-depth case studies of active software startups, ideally in a longitudinal manner. We could also use "member checking" [74], involving entrepreneurs that had their startups analyzed in the study to check our results. A survey study of a larger number of software startups could be conducted to validate this study's findings further and improve generalizability. Such work could also evaluate if startup characteristics such as size and business domain are moderating factors to specific inhibitors' presence and impact. Other studies could focus on a determined stage, probably using more intrusive research methods like case studies, to reveal other inhibitors, like environmental factors, that were limited given the data we analyzed here. Our study focused on inhibitors, but an interesting investigation would be to explore the enablers better. In our previous work, we had already identified some factors, but as our work showed, other enablers could be found by analyzing a larger sample. Future work could also focus on developing new practices to foster experimentation in software startups. In this regard, a natural path is to focus on mitigating or eliminating the identified inhibitors. For instance, given the founders' over-confidence in their ideas, new tactics could incorporate debiasing elements to encourage founders to challenge their ideas.

Future work could evaluate the applicability of the XPro model in other contexts. For instance, to which extent the identified stages happen in consolidated companies? Whether other software process innovations would follow similar stages, either in startups or established software organizations. Another interesting avenue to explore is relationships among different inhibitors, to understand, for instance, if an inhibitor's presence increases the probability of another one. Finally, our study showed that interesting empirical results could be obtained using postmortems collections. Other researchers could employ the same data to investigate other research problems in this context.

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 754489 and with the financial support of the Science Foundation Ireland grant 13/RC/2094.

References

[1] M. Unterkalmsteiner, P. Abrahamsson, A. Nguyen-duc, G. H. Baltes, K. Conboy, D. Dennehy, R. Sweetman, H. Edison, S. Shahid, X. Wang, J. Garbajosa, T. Gorschek, L. Hokkanen, I. Lunesu,

M. Marchesi, L. Morgan, C. Selig, M. Oivo, S. Shah, and F. Kon, "Software Startups - A Research Agenda," *e-Informatica Software Engineering Journal*, vol. 10, no. 1, pp. 1–28, 2016.

[2] D. M. Steininger, "Linking information systems and entrepreneurship: A review and agenda for IT-associated and digital entrepreneurship research," *Information Systems Journal*, vol. 29, no. 2, pp. 363–407, mar 2019.

[3] B. L. Herrmann, M. Marmer, E. Dogrultan, and D. Holtschke, "Startup ecosystem report 2012," *Telefonica Digital and Startup Genome*, 2012.

[4] E. Klotins, M. Unterkalmsteiner, and T. Gorschek, "Software engineering in start-up companies: An analysis of 88 experience reports," *Empirical Software Engineering*, pp. 1–19, may 2018.

[5] M. Cantamessa, V. Gatteschi, G. Perboli, and M. Rosano, "Startups' Roads to Failure," *Sustainability*, vol. 10, no. 7, p. 2346, jul 2018.

[6] W. R. Kerr, R. Nanda, and M. Rhodes-Kropf, "Entrepreneurship as Experimentation," *Journal of Economic Perspectives*, vol. 28, no. 3, pp. 25–48, 2014.

[7] P. Andries, K. Debackere, and B. van Looy, "Simultaneous Experimentation as a Learning Strategy: Business Model Development Under Uncertainty," *Strategic Entrepreneurship Journal*, vol. 7, no. 4, pp. 288–310, dec 2013.

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

[9] D. L. Frederiksen and A. Brem, "How do entrepreneurs think they create value? A scientific reflection of Eric Ries' Lean Startup approach," *International Entrepreneurship and Management Journal*, vol. 13, no. 1, pp. 169–189, 2017.

[10] R. F. Bortolini, M. Nogueira Cortimiglia, A. d. M. F. Danilevicz, and A. Ghezzi, "Lean Startup: a comprehensive historical review," *Management Decision*, no. August, aug 2018.

[11] A. Contigiani and D. A. Levinthal, "Situating the construct of lean start-up: adjacent conversations and possible future directions," *Industrial and Corporate Change*, vol. 28, no. 3, pp. 551–564, jun 2019.

[12] M. Ewens, R. Nanda, and M. Rhodes-Kropf, "Cost of experimentation and the evolution of venture capital," *Journal of Financial Economics*, vol. 128, no. 3, pp. 422–442, 2018.

[13] M. Gutbrod, J. Münch, and M. Tichy, "How Do Software Startups Approach Experimentation? Empirical Results from a Qualitative Interview Study," in *Product-Focused Software Process Improvement*, M. Felderer, D. Méndez Fernández, B. Turhan, M. Kalinowski, F. Sarro, and D. Winkler, Eds. Cham: Springer International Publishing, 2017, pp. 297–304.

[14] C. Giardino, X. Wang, and P. Abrahamsson, "Why early-stage software startups fail: A behavioral framework," *Lecture Notes in Business Information Processing*, vol. 182 LNBIP, pp. 27–41, 2014.

[15] J. Pantiuchina, M. Mondini, D. Khanna, X. Wang, and P. Abrahamsson, "Are software startups applying agile practices? the state of the practice from a large survey," in *Agile Processes in Software Engineering and Extreme Programming*. Cham: Springer International Publishing, 2017, pp. 167–183.

[16] R. G. Fichman, "Information Technology Diffusion: A Review of Empirical Research," in *International Conference on Information Systems*, 1992.

[17] F. D. Davis, "Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology," *MIS Quarterly*, vol. 13, no. 3, p. 319, sep 1989.

[18] R. Cooper and R. Zmud, "Information Technology Implementation Research : A Technological Diffusion Approach," *Management Science*, vol. 36, no. 2, pp. 123–139, 1990.

[19] V. Berg, J. Birkeland, A. Nguyen-Duc, I. O. Pappas, and L. Jaccheri, "Software startup engineering: A systematic mapping study," *Journal of Systems and Software*, vol. 144, no. February, pp. 255–274, oct 2018.

[20] C. Geldes, C. Felzensztein, and J. Palacios-Fenech, "Technological and non-technological innovations, performance and propensity to innovate across industriesThe case of an emerging economy," *Industrial Marketing Management*, vol. 61, pp. 55–66, 2017.

[21] R. Garcia and R. Calantone, "A critical look at technological innovation typology and innovativeness terminology: A literature review," *Journal of Product Innovation Management*, vol. 19, no. 2, pp. 110–132, 2002.

[22] S. Purchase, C. Kum, and D. Olaru, "Paths, events and resource use: New developments in understanding innovation processes," *Industrial Marketing Management*, vol. 58, pp. 123–136, 2016.

[23] J. Melegati and X. Wang, "Do software startups innovate in the same way? A case survey study," in *International Workshop on Software-intensive Business: Start-ups, Ecosystems and Platforms (SiBW 2018)*. CEUR-WS.org, 2018, pp. 193–201.

[24] M. Crowne, "Why software product startups fail and what to do about it. Evolution of software product development in startup companies," *IEEE International Engineering Management Conference*, vol. 1, pp. 338–343, 2002.

[25] E. Klotins, M. Unterkalmsteiner, P. Chatzipetrou, T. Gorschek, R. Prikladniki, N. Tripathi, and L. Pompermaier, "A progression model of software engineering goals, challenges, and practices in start-ups," *IEEE Transactions on Software Engineering*, vol. 13, no. 9, pp. 1–1, 2019.

[26] S. Blank, "Why the Lean Start-Up changes everything," *Harvard Business Review*, vol. 91, no. 5, pp. 63–72, 2013.

[27] D. Dennehy, L. Kasraian, P. O'Raghallaigh, K. Conboy, D. Sammon, and P. Lynch, "A Lean Start-up approach for developing minimum viable products in an established company," *Journal of Decision Systems*, vol. 28, no. 3, pp. 224–232, 2019.

[28] C. Nielsen and M. Lund, "Building Scalable Business Models," *MIT Sloan Management Review*, vol. 59, no. 2, 2018.

[29] D. Cukier and F. Kon, "A maturity model for software startup ecosystems," *Journal of Innovation and Entrepreneurship*, no. 7(1), 2018.

[30] S. M. Sutton Jr., "Role of process in a software start-up," *IEEE Software*, vol. 17, no. 4, pp. 33–39, 2000.

[31] S. H. Thomke, "Managing Experimentation in the Design of New Products," *Management Science*, vol. 44, no. 6, pp. 743–762, 1998.

[32] H. Chesbrough, "Business model innovation: Opportunities and barriers," *Long Range Planning*, vol. 43, no. 2-3, pp. 354–363, 2010.

[33] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wessln, *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated, 2012.

[34] N. Juristo and A. M. Moreno, *Basics of software engineering experimentation*. Springer Science & Business Media, 2013.

[35] A. Fabijan, P. Dmitriev, C. McFarland, L. Vermeer, H. Holmström Olsson, and J. Bosch, "Experimentation growth: Evolving trustworthy A/B testing capabilities in online software companies," *Journal of Software: Evolution and Process*, no. December 2017, p. e2113, 2018.

[36] E. Lindgren and J. Münch, "Raising the odds of success: the current state of experimentation in product development," *Information and Software Technology*, vol. 77, pp. 80–91, 2016.

[37] A. Fabijan, P. Dmitriev, H. H. Olsson, and J. Bosch, "The Evolution of Continuous Experimentation in Software Product Development: From Data to a Data-Driven Organization at Scale," in *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. IEEE, may 2017, pp. 770–780.

[38] B. Fitzgerald and K. J. Stol, "Continuous software engineering: A roadmap and agenda," *Journal of Systems and Software*, vol. 123, pp. 176–189, 2017.

[39] S. G. Yaman, M. Munezero, J. Münch, F. Fagerholm, O. Syd, M. Aaltola, C. Palmu, and T. Männistö, "Introducing continuous experimentation in large software-intensive product and service organisations," *Journal of Systems and Software*, vol. 133, pp. 195–211, 2017.

[40] J. Bosch, H. H. Olsson, and I. Crnkovic, "It Takes Three to Tango : Requirement , Outcome / data , and AI Driven Development," in *Software-intensive Business Workshop on Start-ups, Platforms and Ecosystems (SiBW 2018)*. Espoo: CEUR-WS.org, 2018, pp. 177–192.

[41] J. Bosch, "Building Products as Innovation Experiment Systems," in *Lecture Notes in Business Information Processing*, 2012, vol. 114 LNBIP, pp. 27–39.

[42] S. Thomke, "Enlightened experimentation. The new imperative for innovation." *Harvard Business Review*, vol. 79, no. 2, pp. 66–75, 2001.

[43] G. S. Lynn, A. E. Akgün, and H. Keskin, "Accelerated learning in new product development teams," *European Journal of Innovation Management*, vol. 6, no. 4, pp. 201–212, dec 2003.

[44] L. Williams and A. Cockburn, "Agile software development: it's about feedback and change," *Computer*, vol. 36, no. 6, pp. 39–43, jun 2003.

[45] A. Fabijan, H. H. Olsson, and J. Bosch, "Customer Feedback and Data Collection Techniques in Software R&D: A Literature Review," in *Lecture Notes in Business Information Processing*, 2015, vol. 210, pp. 139–153.

[46] J. Melegati, R. Chanin, X. Wang, A. Sales, and R. Prikladnicki, "Enablers and Inhibitors of Experimentation in Early-Stage Software Startups." Springer International Publishing, 2019, vol. 1, pp. 554–569.
[47] E. M. Rogers, *Diffusion of innovations*. Simon and Schuster, 2010.
[48] F. Fagerholm, A. Sanchez Guinea, H. Mäenpää, and J. Münch, "The RIGHT model for Continuous Experimentation," *Journal of Systems and Software*, vol. 123, pp. 292–305, 2017.
[49] M. Senapathi and A. Srinivasan, "Understanding post-adoptive agile usage: An exploratory cross-case analysis," *Journal of Systems and Software*, vol. 85, no. 6, pp. 1255–1268, 2012.
[50] V. Venkatesh and F. D. Davis, "A Theoretical Extension of the Technology Acceptance Model: Four Longitudinal Field Studies," *Management Science*, vol. 46, no. 2, pp. 186–204, feb 2000.
[51] H. H. Olsson and J. Bosch, "From Opinions to Data-Driven Software R&D: A Multi-case Study on How to Close the 'Open Loop' Problem," in *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*. IEEE, aug 2014, pp. 9–16.
[52] ——, "Towards Continuous Customer Validation: A Conceptual Model for Combining Qualitative Customer Feedback with Quantitative Customer Observation," in *Lecture Notes in Business Information Processing*, 2015, vol. 210, pp. 154–166.
[53] J. Melegati, X. Wang, and P. Abrahamsson, "Hypotheses Engineering: First Essential Steps of Experiment-Driven Software Development," in *2019 IEEE/ACM Joint 4th International Workshop on Rapid Continuous Software Engineering and 1st International Workshop on Data-Driven Decisions, Experimentation and Evolution (RCoSE/DDrEE)*. IEEE, may 2019, pp. 16–19.
[54] R. Ros and P. Runeson, "Continuous experimentation and A/B testing," in *Proceedings of the 4th International Workshop on Rapid Continuous Software Engineering - RCoSE '18*. New York, New York, USA: ACM Press, 2018, pp. 35–41.
[55] C. K. Riemenschneider, B. C. Hardgrave, and F. D. Davis, "Explaining software developer acceptance of methodologies: A comparison of five theoretical models," *IEEE Transactions on Software Engineering*, vol. 28, no. 12, pp. 1135–1145, 2002.
[56] M. Senapathi and M. L. Drury-Grogan, "Refining a model for sustained usage of agile methodologies," *Journal of Systems and Software*, vol. 132, pp. 298–316, 2017.
[57] G. Mangalaraj, R. Mahapatra, and S. Nerur, "Acceptance of software process innovations- The case of extreme programming," *European Journal of Information Systems*, vol. 18, no. 4, pp. 344–354, 2009.
[58] H. Jansen, "The logic of qualitative survey research and its position in the field of social research methods," *Forum Qualitative Sozialforschung*, vol. 11, no. 2, 2010.
[59] J. Melegati, H. Edison, and X. Wang, "Failed software startups postmortem urls," Sep. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.4017712
[60] S. S. Bajwa, X. Wang, A. Nguyen Duc, and P. Abrahamsson, ""Failures" to be celebrated: an analysis of major pivots of software startups," *Empirical Software Engineering*, vol. 22, no. 5, pp. 2373–2408, oct 2017.
[61] N. Tripathi, E. Klotins, R. Prikladnicki, M. Oivo, L. Pompermaier, M. Arun Sojan Kudakacheril, A., Unterkalmsteiner, K. Liukkunen, and T. Gorschek, "An anatomy of requirement engineering in software startups using multi-vocal literature and case survey," *Journal of Systems and Software*, vol. 146, pp. 130–151, 2018.
[62] D. S. Cruzes, T. Dybå, P. Runeson, and M. Höst, "Case studies synthesis: a thematic, cross-case, and narrative synthesis worked example," *Empirical Software Engineering*, vol. 20, no. 6, pp. 1634–1665, 2015.
[63] D. S. Cruzes and T. Dyba, "Recommended Steps for Thematic Synthesis in Software Engineering," *2011 International Symposium on Empirical Software Engineering and Measurement*, no. 7491, pp. 275–284, 2011.
[64] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative Research in Psychology*, vol. 3, no. 2, pp. 77–101, jan 2006.
[65] M. Miles, A. Huberman, and J. Saldana, *Qualitative Data Analysis*. SAGE Publications, 2014.
[66] A. L. Strauss and J. M. Corbin, *Basics of qualitative research: grounded theory procedures and techniques*. Sage Publications, 1990.
[67] 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," *Lean Enterprise Software and Systems*, pp. 1–15, 2013.
[68] M. Rysman, "The Economics of Two-Sided Markets," *Journal of Economic Perspectives*, vol. 23, no. 3, pp. 125–143, aug 2009.
[69] R. Mohanani, I. Salman, B. Turhan, P. Rodriguez, and P. Ralph, "Cognitive Biases in Software Engineering: A Systematic Mapping Study," *IEEE Transactions on Software Engineering*, vol. 5589, no. c, 2018.
[70] C. Giardino, S. S. Bajwa, X. Wang, and P. Abrahamsson, "Key Challenges in Early-Stage Software Startups," in *Lecture Notes in Business Information Processing*, 2015, vol. 212, pp. 52–63.
[71] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, no. 2, pp. 131–164, 2009.
[72] C. Gralha, D. Damian, A. I. T. Wasserman, M. Goulão, and J. Araújo, "The evolution of requirements practices in software startups," in *Proceedings of the 40th International Conference on Software Engineering - ICSE '18*. New York, New York, USA: ACM Press, 2018, pp. 823–833.
[73] R. Yin, *Case Study Research: Design and Methods*, ser. Applied Social Research Methods. SAGE Publications, 2003.
[74] S. Jantunen and D. C. Gause, "Using a grounded theory approach for exploring software product management challenges," *Journal of Systems and Software*, vol. 95, pp. 32–51, 2014.

Jorge Melegati is a PhD student at the Faculty of Computer Science of the Free University of Bozen-Bolzano. He received his master's degree in Computer Science from the University of São Paulo, Brazil and he is a Computer Engineer by the Technological Institute of Aeronautics, Brazil. He has more than seven years experience as a software developer including five years spent in a startup. His research is focused on experimentation, startups, and software engineering education.

Henry Edison is a Marie Curie Fellow with Lero - the Irish Software Research Centre at NUI Galway, Ireland. He received a Ph.D. in Computer Science from Free University of Bozen-Bolzano, Italy. His research interests include empirical software engineering in the area of software product innovation, software startup, Lean startup, inner source, and large-scale agile. He has published in leading journals and conferences in his field.

Xiaofeng Wang is an associate professor at the Computer Science Faculty of Unibz. Her main research areas include software startups, agile and lean software development and innovation, and human factors in software engineering. She is actively publishing in Software Engineering venues, including IEEE Software, Journal of Systems and Software, Empirical Software Engineering, etc.. She is also active in serving various Software Engineering conferences and workshops. She has collaborated with large companies and software startups on national and European projects.