How Continuously Do Brazilian Software Startups Practice Requirements Engineering?

Guilherme Pedro Santos Jardim*
Universidade Estadual de Campinas
Campinas, Brasil

Breno Bernard Nicolau de França
Universidade Estadual de Campinas
Campinas, Brasil
bfranca@unicamp.br

Abstract

Software startups operate in highly dynamic, uncertain environments with limited resources, where traditional Requirements Engineering (RE) approaches often fail to provide the necessary flexibility. While agile and user-centered practices have gained attention, there is limited empirical evidence on how Brazilian software startups continuously manage stakeholder involvement and requirement evolution. This study investigates current RE practices in Brazilian software startups, focusing on how requirements are discovered, specified, validated, and kept aligned with frequently evolving business and user needs. We conducted a survey with 34 practitioners in product-related roles within the Brazilian startup ecosystem. The findings reveal that Brazilian startups favor rapid, lightweight methods over traditional approaches. In discovery, document analysis and brainstorming are the most frequently used techniques. User Stories and Prototyping dominate requirement specification, while Prototype Demos and Meetings are the primary validation methods. Startups maintain frequent stakeholder contact, typically daily or weekly, to ensure alignment. Requirement changes are primarily driven by external stakeholders and product owners, with prioritization based on business value and impact. Surprisingly, the study found no evidence of proactive RE that revisits its product concepts without prior change requests. Overall, RE in Brazilian startups is iterative, collaborative, and stakeholder-driven, relying on interconnected lightweight practices to sustain product relevance in fast-paced markets.

Keywords

Continuous Requirements Engineering, Software Startups

1 Introduction

Requirements Engineering (RE) is generally considered a foundational phase of the software development process, responsible for gathering, analyzing, and documenting stakeholder needs [18]. To complete RE activities, organizations adopt practices and techniques to concretely accomplish domain and problem understanding, scope definition, feature identification, documentation, stakeholder validation, change management, and other goals.

From plan-driven to agile development processes, the existing RE literature has extensively investigated several challenges and solutions [15][26]. And while agile RE effectively addresses many shortcomings of traditional methods, several challenges persist. These challenges generally fall into categories involving documentation, stakeholder dynamics, non-functional requirements, and organizational constraints. Here, we are primarily concerned with the organizational contexts of startups with respect to uncertainty and dynamic environments, how they cope with changes as requirements evolve, and the dynamics of stakeholders' involvement.

Gupta et al. [14] argue that traditional RE approaches may be insufficient, as startups operate in a highly competitive, dynamic environment where uncertainty and rapid responses to changing market demands are crucial for success, yet also pose unique challenges. They also found a lack of solutions and experience reports that could directly support startups in their RE efforts.

To address this gap, researchers have explored specific RE practices and challenges software startups face. One of these critical challenges is balancing development speed with the ability to capture and respond to evolving customer needs [11]. Assyne et al. [2] proposed a dynamic competency model for software startups, highlighting the need for specialized competencies in RE, such as the ability to adapt quickly to changing market conditions and to communicate effectively with diverse stakeholders. Continuous Requirements Engineering (CRE) approaches have sought to address this challenge by emphasizing ongoing elicitation, analysis, and management of evolving requirements throughout the software development life cycle [14].

Although the literature covers RE across various organizational contexts, the discussion of how these findings apply to startups is limited. It becomes more evident when narrowing to specific issues related to stakeholders' involvement and requirements management and evolution. In particular, we found little evidence specific to the Brazilian startup ecosystem regarding these issues.

This study aims to characterize the state of practice regarding CRE in Brazilian software startups and their capacity to handle change. Thus, the main contributions of this paper are threefold. First, it provides an empirical characterization of current RE practices in Brazilian software startups. Second, it examines the extent to which these practices support CRE, particularly in accommodating change, maintaining alignment between requirements and evolving business and user needs, proactively managing requirements, and fostering stakeholder involvement. Third, the study reveals that change management in the investigated startups tends to be conservative: once a concept or requirement is established, it is rarely revisited unless prompted by explicit external requests. This finding contributes to a more nuanced understanding of how startups balance agility with requirements stability in practice.

The remaining sections are organized as follows. Section 2 presents the background. Section 3 presents the most closely related work. Section 4 presents our research method for conducting the study. Section 5 details the study results. Section 6 discusses the threats to validity. Section 7 concludes the paper.

*In Memoriam

2 Background

We grounded our study in existing literature on RE practices and techniques, as well as the specific challenges and characteristics of startups. Here, we first clarify how RE is performed across software development contexts, including the transition from traditional to continuous RE through agile, and focusing on the challenges and practices involved in discovery, change management, and evolution.

Batool et al. [4] recall that traditional RE comprises several activities for gathering requirements based on users' needs and demands. They also highlight the following activities: Requirements Elicitation, Analysis, Documentation, Validation, and Management.

Since traditional RE relies on input from a small set of users over a limited, predetermined period, it conflicts with the ever-changing nature of software requirements, making it challenging to maintain continuity in meeting expectations. Agile RE emerges from that conflict, moving away from the stringency of traditional RE toward implementing agile principles, particularly prioritizing the delivery of working software and adopting a shorter time scale. By doing so, it encourages practices that accommodate changes in requirements at any stage of the development process [7]. Furthermore, stakeholders should actively participate in the development process to enable regular feedback and insights.

Agile RE takes an iterative discovery approach, which is more appropriate for the contemporary environment, where developing unambiguous and complete requirements is necessary [5]. Cao and Ramesh conducted a multi-case study on 16 companies that characterized themselves as agile. The study identified seven practices favoring agile RE in the organizations, from which we highlight: i) Requirements prioritization in every development cycle; ii) Constant planning to manage requirements change; iii) Prototyping; iv) Validation through acceptance tests and review meetings.

Pagano [22] argues that novel requirements elicitation involves continuously gathering and communicating user input throughout the software development process, learning to filter, prioritize, and propose requirements aligned with stakeholders' preferences could address the ever-growing volume of available data while meeting customers' needs. In this sense, the idea of Continuous Requirements Engineering (CRE) shares several principles with the Agile Manifesto to support new approaches, methods, and tools for embracing the ever-growing opportunities and challenges in an ever-changing environment [10]. Therefore, Forbrig claims that Human-Centered Design (HCD) must be a prominent component of continuous RE because both aim to continuously deliver value through features. Finally, he states that the principle of continuity requires that project teams adapt their size to the current situation and *repeat RE efforts throughout the project life cycle.*

This way, we understand that although frequent changes requests may occur in many scenarios, working solely in a reactive manner on changes, like in regular change management, does not characterize CRE. For that, the team must proactively revisit the product concept and derived requirements continuously, enforcing requirements discovery and ideation activities being executed in a regular pace. This rationale leads us to the RQ4 in Section 4.1.

Kirikova [16] proposes frameworks for different settings (including startups) to provide scalability, flexibility, and context-adjusted CRE practices. According to them, agile techniques differ slightly, but their principles pervade our understanding of CRE, so the author summarizes them under the CRE lenses, including: i) Continuous communication and collaboration with customers; ii) Iterative requirements engineering; iii) Continuous requirements prioritization; iv) Just-in-time specification; v) Managing requirements change; vi) Review meetings and acceptance tests for validation.

Curcio et al. [6] corroborate this, explaining that although continuous and non-continuous RE share several techniques (e.g., prototyping, inception, interviews, HCD), the processes surrounding these activities should incorporate repetition, following the philosophy of Continuous* proposed by Fitzgerald and Stöl [9]. If non-continuous practices are slower than continuous ones, it becomes more difficult to execute them frequently. Therefore, they require adaptation, such as splitting activities into smaller routines.

Software Startup is an organization with little to no operational history, limited resources, influenced by multiple factors, and handling ever-changing technologies and markets [27]. Moreover, Giardino *et al.* [12] identified two additional characteristics: high uncertainty and rapid evolution, which distinguish them from the more stable environments of well-established companies. Still, Paternoster *et al.* [23] found inconsistencies in the characterization of software startups across related studies and developed an additional set of characteristics: innovation, rapid growth, time pressure, third-party dependence, focus on a single product, and a flat organizational structure. These characteristics relate to CRE and reinforce the assumption that continuous practices are better suited for them.

3 Related Works

In 2017, Fernandez et al. [8] surveyed companies worldwide to examine RE challenges and found that 92 of 228 (40%) adopted agile processes such as Scrum and XP, and 58 (25%) adopted a mixed approach. They also identified traditional processes like RUP, Waterfall, and V-Model XT as plan-driven. This way, approximately 65% of participants reported using agile practices in their software development processes. And practitioners recognized changes in goals, business processes, and/or requirements, as well as timeboxing (insufficient time to conduct RE), as major problems to be addressed. This reveals the characteristics of processes handling requirements more discretely. The study does mention small companies, but there are no clues that they are startups. The focus of their survey is to characterize problems and their causes. For that, they also capture current practices, but no results are reported on changing requirements, frequency, or stakeholder involvement in how companies handle them, although there is a unique question about it in the questionnaire.

Another report of the NaPiRE family describes the status quo of RE [28]. There is an entire research question and corresponding section in the questionnaire about requirements changes, but in terms of the consequences of those changes, rather than their frequency and nature. The authors also discussed the continuous improvement of RE, which is also not the same of CRE.

In a multiple-case study of three software startups, Rafiq *et al.* [25] observed that they gathered requirements informally as their products evolved. Common requirement elicitation practices include benchmarking, team discussions, prototyping, and interviews. Startups often focus on gathering requirements based on

founders' ideas in the early stages, refining them as the product evolves. This leads to assumption-based solutions and minimal formal documentation, with only rough notes, to-do lists, and feedback emails being produced. However, this work did not provide information on practices and techniques for other RE activities, such as specification, validation, and change management.

A Grounded Theory study [13] analyzed the evolution of 16 software startups and the RE process within a three-stage model. It outlined the evolution of practices within six dimensions: requirements artifacts, knowledge management, requirements-related roles, planning, technical debt, and product quality. Their scope also does not provide evidence regarding specific practices and techniques, nor does it address any issues related to changes in requirements.

Startups often have minimal documentation for their product requirements in the early stages, but this becomes more comprehensive as the startup matures. As companies expand, knowledge management goes through three stages of organization. The evolution of requirements-related roles is categorized into three phases, and the planning process also evolves.

Melegati et al. [21] interviewed 17 startup practitioners and used Grounded Theory to propose a model suggesting that software startups should not follow a fixed set of practices but rather tailor their processes to evolve as the company grows. They also showed the susceptibility of startups to various factors, including influences from founders, development managers, developers, market conditions, business models, and the broader ecosystem. This challenges the notion that newer software startups are less successful simply because they have less experience. In summary, software startups employ practices similar to those in agile contexts, particularly for analysis and documentation. However, the lack of accessible customers significantly influences software startups, as it does in general agile contexts. This work conceptually overlaps with ours, but the scope of our findings is quite different, as it focuses on practices and techniques, i.e., more engineering-oriented than business models and the general RE process. Furthermore, their study is limited to 17 practitioners and does not provide demographic data, whereas we targeted the reality of 34 Brazilian software startups.

In [14], the authors conducted a systematic mapping to outline RE research for the software startup context. The results revealed several key research areas, listed in order of frequency: Generic Approaches for RE, Product Validation, Requirement Elicitation, Requirement Validation, and Requirement Prioritization and Documentation, all of which are closely related.

4 Research Method

4.1 Research Questions

We used the Goal Question Metric (GQM) [3] template to establish a measurable goal, which is to **analyze** RE practices and techniques, **with the purpose of** characterizing them, **focused on** the capacity of handling change, **from the point of view of** Brazilian professionals, **in the context of** software startups. This goal shaped the following research (or internal [19]) questions:

RQ1: What practices and techniques do software startups use to support requirements engineering (discovery, specification, and validation)?

RQ2: How do software startups ensure the requirements remain aligned with the evolving needs of the business and the end users over time?

RQ3: How does the involvement of stakeholders, such as customers, end users, and product owners, contribute to requirements engineering in software startups?

RQ4: Do startups proactively manage and integrate requirement changes during the life cycle to ensure the product evolves inherently, or do they primarily react to change requests?

4.2 Study Design

To investigate our research questions, we designed a survey for professionals working in product-related roles within the Brazilian software startup ecosystem. The survey included both open-ended and closed-ended questions.

4.2.1 Sampling Strategy. First, our sampling strategy used LinkedIn as the sampling frame, with a premium subscription. The target audience comprises Brazilian professionals working in software startups and engaged in RE. To ensure a representative sample, we targeted individuals in product-related roles within the Brazilian software startup ecosystem, such as product managers, product owners, and business analysts.

To be eligible, participants had to meet the following criteria:

(1) Work experience in a software startup, either as a founder, product/project manager, product owner, or other relevant product role;
(2) Involvement in discovery activities within the startup;
(3) Work experience within the Brazilian context.

We searched for participants using keywords and recruited them via simple random sampling. However, this strategy yielded a very low response rate because the social network requires prior acceptance to send direct messages, thereby introducing an additional barrier to reaching professionals. Later, we transitioned to snowball sampling through the authors' contacts in software startup companies and direct outreach to software startup communities. We detail these strategies in Section 4.4.

Additionally, we adopted a supervised approach to conduct the survey. The researchers were present and supported the participants as they completed the questionnaire.

4.2.2 Questionnaire. The questionnaire contained four sections: **(1)** Demographic information, **(2)** Requirements engineering practices and techniques, **(3)** Process continuity, and **(4)** Product evolution. These sections correspond to the four dimensions through which we wanted to guide our questions.

In certain instances, the closed-ended questions provided response options based on codes found in the works of Melegati et al. [20] and Zowghi et al. [29]. In other instances, we used closed-ended questions with a Likert scale to measure frequency, ranging from "Never" (0) to "Always" (4), as in [1]. All closed-ended questions were made mandatory. Table 1 details the survey questions.

4.2.3 Analysis Dimensions. To analyze the responses, we adopted five dimensions inspired by the works of Fernandez et al. [8] and an adaptation of Kirikova el al. [16] CRE framework core principles:

Table 1: Questionnaire Structure. oe: open-ended, sr: single response, mr: multiple response.

No.	Question	Type
1	How many years of experience do you have in requirements engineering?	oe
2	Which company do you work for?	oe
3	On average, how many people are on your immediate team?	oe
4	How long has your company been on the market?	oe
5	What is your job title?	oe
6	Who is your typical customer/end-user?	oe
7	To what extent does your team employ the following techniques for requirements discovery?	Likert
8	Which practices or techniques does your startup utilize for requirements specification?	mr
9	Which practices or techniques does your startup utilize for requirements validation?	mr
10	How often do you contact the customer/end-user during the discovery process?	Likert
11	How often do you perform feedback loops?	Likert
12	How often do you do user testing?	Likert
13	How often do you revisit discovered requirements?	Likert
14	Is the discovery a discrete (eventual) activity?	sr
15	How often does your team experience change requests during development?	Likert
16	Who typically triggers changes in your startup?	mr
17	Which project management frameworks help manage requirement changes?	mr
18	What criteria are used to evaluate a change request?	oe
19	What are the consequences of a change request to your process?	mr
20	Feel free to provide any comments regarding change requests.	oe

(1) **Organizational context:** characteristics of software startups, such as team size, development lifecycle, and maturity, could potentially influence RE practices and techniques.
(2) **Software discovery practices and techniques:** includes the practices and techniques employed for requirements discovery, documentation, prioritization, and validation.
(3) **Alignment with evolving business and user needs:** startups' ways of keeping requirements aligned with changing conditions over the development life cycle. This considers practices for identifying and capturing evolving requirements, timeframes, and trigger points for updates.
(4) **Stakeholder involvement:** the level of involvement of various stakeholders (customers, end users, product owners, etc.) in the RE process and the mechanisms for their engagement and feedback incorporation.
(5) **Change management:** the degree of proactive integration of changes during development and mechanisms to ensure the product evolves based on changing needs.

4.3 Pilot

Following the recommendations of Linaker et al. [19], we conducted a pilot study with product-related professionals in the context of software startups in Brazil before deploying the full survey. The goal was to validate the questionnaire's understandability and relevance. Based on the feedback received, we made minor adjustments to the wording and structure of the questions to improve clarity and flow.

Contrary to our initial plan, we conducted the pilot session in real-time rather than asynchronously. Participants completed the survey while we were present and provided immediate feedback, which we noted. This enabled us to promptly identify areas for improvement and implement adjustments before data collection.

After completing the pilot study, we consolidated some questions and shortened the questionnaire. Furthermore, we refined the semantics and formatting to improve the participants' experience, recognizing that academic and industry vocabularies may utilize different taxonomies for the same techniques and processes (e.g., requirement elicitation vs. ideation or software discovery). The questionnaire responses collected in the pilot study were not included in our analysis. The feedback was useful to rephrase some questions considering the target audience.

4.4 Participant Recruitment

For our initial recruitment effort, we directly contacted individuals on LinkedIn whose profiles matched our specified criteria (Section 4.2). These contacts were obtained from startups listed in the Brazilian Startup Association's annual mapping, accessible via a free registration form on the association's website.

Subsequent efforts leveraged personal and professional networks, including outreach to members of the Brazilian Product Managers Community and the Brazilian Agile Community. However, only four participants completed the survey over three months. In this approach, we monitored responses daily.

To address the low response rate, we shifted to convenience sampling and snowball sampling, as described in [17], to identify and recruit potential participants. We asked them to complete the survey during a synchronous interview, in which we guided them through the questionnaire and addressed questions about terminology or context. Accordingly, we collected 34 complete responses, which is a relatively large sample for an exploratory study when compared to most related work. In a 95% confidence interval, considering the proportion of 9.2% technology startups from the annual mapping population (14.000 startups), our sample has 9.7% margin of error, which is considered a safe interval to draw conclusions.

The supervised approach enabled us to collect complete data and mitigate dropout (incomplete responses).

4.5 Analysis Procedure

For quantitative data, i.e., single-response, multiple-response, and Likert-scale questions, we primarily conducted exploratory data

Table 2: Technique Usage at Different Percentages. A: Always, Of: Often, Oc: Occasionally, R: Rarely, N: Never.

Technique	A	Of	Oc	R	N
Document Analysis	45.45	21.21	12.12	18.18	3.03
Brainstorming	36.36	48.48	9.09	6.06	0.00
Interviews	21.21	36.36	27.27	12.12	3.03
Observation	30.30	27.27	21.21	12.12	9.09
Prototyping	27.27	27.27	30.30	12.12	3.03
Surveys	9.09	21.21	24.24	15.15	30.30
Workshops	6.06	0.00	27.27	36.36	30.30

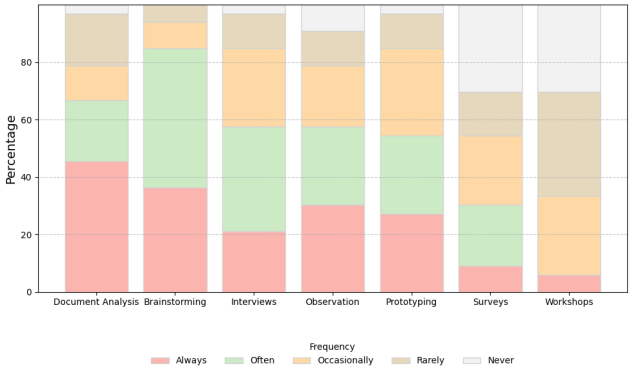

Figure 1: Techniques for Requirements Discovery

analysis using charts and descriptive statistics. To characterize potential relationships, we used heatmaps. For the open-ended questions, we adopted a simple qualitative coding procedure similar to open coding. We decided on that approach as the answers were short and objective, demanding only terminology harmonization.

5 Results

5.1 RQ1: RE Practices and Techniques

5.1.1 Discovery. To explore the practices and techniques startups employ in requirements discovery, we asked participants about the frequency of use for seven commonly adopted approaches reported in related work. The aggregated data, summarized in Table 2, highlights significant variation in the adoption of these techniques. Figure 1 provides a more intuitive view of relative adoption rates by showing the distribution of responses across five frequency categories: "Always," "Often," "Occasionally," "Rarely," and "Never."

Document Analysis has the highest proportion of "Always" responses (45.45%), indicating it is a foundational technique in requirements discovery for these startups. This consistent use suggests that startups rely heavily on existing documentation to establish initial requirements or validate assumptions. Brainstorming shows widespread adoption, with 36.36% of respondents selecting "Always" and 48.48% selecting "Often." This suggests that collaborative ideation is central to the discovery process.

Interviews, Observation, and Prototyping fall in the middle tier of usage. While fewer respondents use these techniques "Always" (21.21%–30.30%), they are significantly represented in the "Often" and "Occasionally" categories. For instance, Interviews are used "Often" by 36.36% and "Occasionally" by 27.27%, indicating their use depends on the situation. Prototyping (27.27% "Always") is likely employed when visual or tangible artifacts are necessary to gather stakeholder feedback or refine requirements iteratively.

Surveys and Workshops are the least adopted techniques. Surveys are "Never" used by 30.30% of respondents, and only 9.09% report using them "Always." Similarly, Workshops have an even lower adoption rate, with 30.30% reporting "Never" use and just 6.06% reporting consistent use. These results indicate a preference for more interactive or immediate feedback methods over formalized or asynchronous approaches.

In summary, techniques such as document analysis and brainstorming exhibit consistent, high levels of adoption, with most respondents reporting frequent use. In contrast, methods such as surveys and workshops exhibit different usage patterns, with a substantial proportion of participants rarely or never using them. This chart highlights the tendency of startups to rely on rapid, lightweight methods (e.g., brainstorming, document analysis) while de-prioritizing techniques that require extensive planning or resources (e.g., surveys, workshops). These findings align with the iterative, resource-constrained nature of startups, which prioritize techniques that provide flexibility and rapid feedback.

5.1.2 Specification. Participants identified several practices and techniques used by software startups for specifying requirements (see the distribution in Figure 2). Figure 3 explores the relationships and overlaps in their usage.

As shown in Figure 2, User Stories dominate as the most widely adopted technique, with 32 occurrences among participants. This result reflects the prominence of Agile methods, where user stories are central to capturing concise, user-centered requirements. The second most frequent practice is the use of Wireframes and Prototypes, which appear 24 times, underscoring the importance of visual artifacts in clarifying and validating requirements during early development stages.

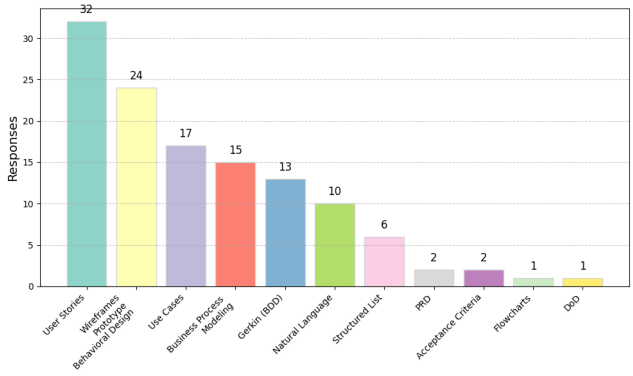

Figure 2: Techniques for Requirements Specification

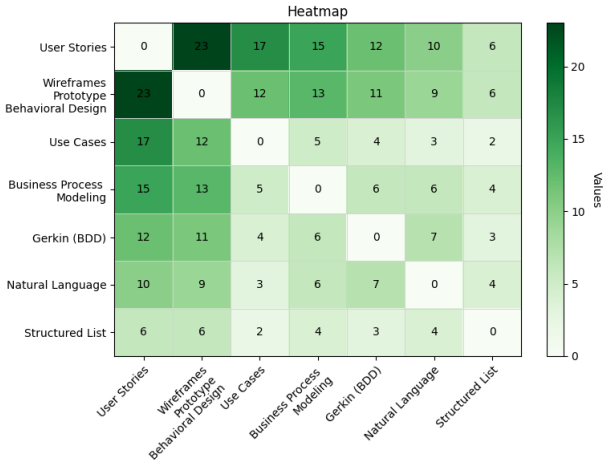

Figure 3: Techniques for Requirements Specification.

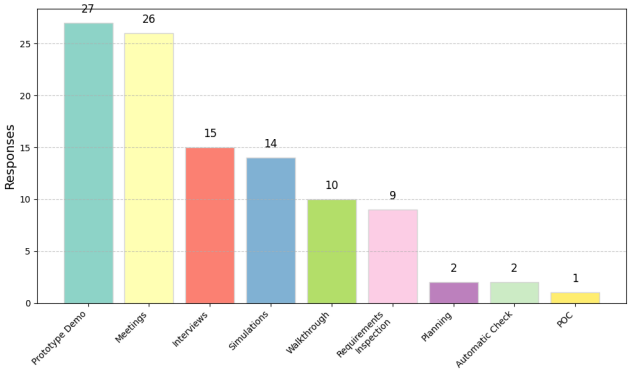

Figure 4: Techniques for Requirements Validation

Following these, techniques like Use Cases (17 occurrences) and Business Process Modeling (15 occurrences) are also notable. These are likely used when a deeper understanding of workflows or interactions is necessary, complementing user stories and prototypes.

Less frequently mentioned techniques include Gherkin (Behavior-Driven Development) (13) and Natural Language Specifications (10). While these are valuable in specific contexts, their adoption appears to be limited, possibly tied to the prioritization of testable requirements or structured communication.

At the lower end of adoption, Structured Lists, Product Requirements Documents (PRD), Acceptance Criteria, Flowcharts, and Definitions of Done (DoD) are rarely reported. These findings may suggest that document-heavy approaches are less favored in startups' fast-paced environments.

The heatmap (Figure 3) reveals substantial overlap between User Stories and Wireframes/Prototypes, emphasizing the frequent co-occurrence of these complementary techniques, with prototypes being the most prevalent means of making requirements more detailed or concrete. Similarly, Use Cases and Business Process Modeling often appear together, indicating their combined utility in capturing detailed process flows. Conversely, techniques such as DoD and Flowcharts show limited overlap with other techniques, further confirming their niche application.

These findings underline a preference for lightweight, iterative, and user-centered approaches to requirements specification, which align with startups' flexible and adaptive nature.

5.1.3 Validation. To identify the techniques startups use to validate requirements, participants reported adopting various practices. Figure 4 presents the frequency distribution of the techniques, while Figure 5 explores the relationships among their usage [1].

Prototype Demo emerges as the most commonly used technique in Figure 4, reported by 27 participants. This finding highlights a preference for visual, interactive validation methods that enable stakeholders to engage directly with a working model of the product. Meetings are frequent, with 26 occurrences, underscoring the importance of direct communication and collaborative discussions in the validation process.

Other frequently used techniques include Interviews (15) and Simulations (14). Interviews enable direct feedback from stakeholders, while simulations help validate requirements in controlled, realistic scenarios. These techniques complement the iterative and collaborative nature of validation in startups.

Less frequently mentioned techniques, such as Walkthroughs (10 occurrences) and Requirements Inspections (9 occurrences), likely play more specialized roles. These methods involve structured requirements reviews but may require more time and effort than lightweight techniques such as prototype demonstrations.

Planning[2], Automatic Checks, and Proof of Concept (POC) are rarely used at the lower end of the spectrum, with 2–1 occurrences. These techniques may be perceived as resource-intensive or as more suitable for specific use cases than for general validation practices.

Figure 5 reveals substantial overlaps between Prototype Demos and Meetings, suggesting these are often used together for interactive validation. Similarly, Interviews and Simulations show moderate co-occurrence, reflecting their complementary roles in gathering feedback and testing requirements under realistic conditions.

Overall, these findings underscore the importance of lightweight, interactive, and collaborative approaches in requirements validation. Techniques such as prototype demonstrations and meetings align well with startups' fast-paced, iterative environments, where flexibility and direct stakeholder engagement are paramount.

5.2 RQ2: Requirements Alignment with the Evolving Needs

Maintaining alignment between requirements, evolving business goals, and user needs is a critical challenge for software startups operating in dynamic environments. To address this, startups employ various strategies to adapt to changing contexts while ensuring their products remain relevant and valuable. These strategies often

[1]Numbers indicate the frequency of co-occurrences among variables observed in the dataset across all heat maps in this paper

[2]Planning means validating requirements in planning meetings. We did not merge with Meetings, as the dynamics of a validation meeting are typically focused on validation. In contrast, these meetings focus on developing a plan for the next iteration.

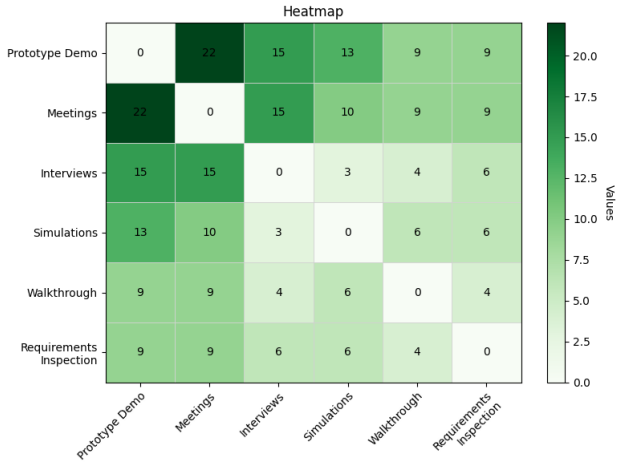

Figure 5: Techniques for Requirements Validation

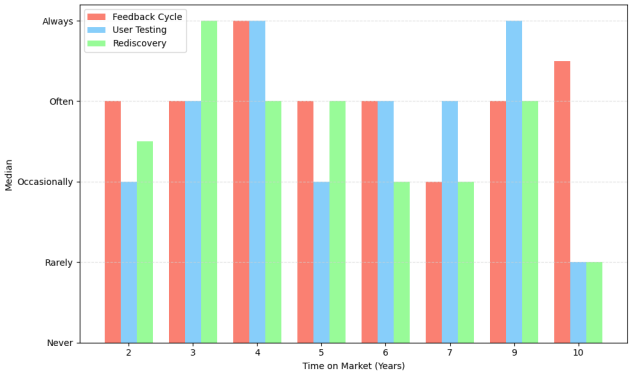

Figure 6: Practices Frequency vs. Time on Market

revolve around iterative processes, continuous stakeholder engagement, and periodic re-evaluation of requirements. The following results (Figure 6) detail the frequency and maturity of practices such as feedback cycles, user testing, and iterative (re)discovery of requirements, providing insights into how startups navigate the complexities of aligning their requirements over time.

Feedback cycles are the most consistently used practice, with a significant proportion of respondents reporting frequent use ("Often" and "Always"). Feedback cycles facilitate continuous stakeholder input, enabling startups to identify changes in user needs or business priorities early. As startups mature, their reliance on feedback cycles remains steady, reflecting their enduring importance.

User tests also play a critical role in maintaining alignment, with most respondents indicating regular usage. These tests allow validating whether requirements and implementations resonate with end users. Figure 6 shows it becomes even more pronounced for startups with longer market presence (more than 3 years), indicating its importance as products evolve and user bases expand.

The practice of rediscovery, where requirements are periodically re-evaluated or refined, is another important strategy. While less

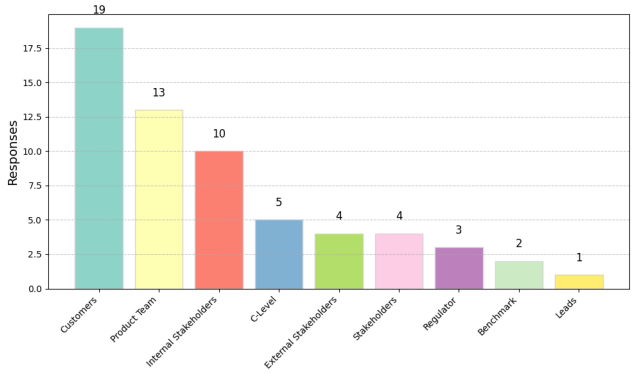

Figure 7: Source of Requirements

frequently adopted than feedback cycles or user testing, rediscovery remains essential for startups adapting to significant market or internal changes. Startups with shorter market presence report more frequent rediscovery efforts, suggesting that this is a strategy leveraged to address uncertainty in the early years and to address eventual needs for pivoting.

In summary, startups prioritize mechanisms that provide frequent, actionable input, such as feedback cycles and user tests, while leveraging rediscovery to achieve maturity and to allow for increased product complexity. These strategies collectively enable them to remain agile and responsive to evolving needs.

5.3 RQ3: Stakeholders' Involvement

The sources of requirements in software startups vary widely, with a clear emphasis on external and internal stakeholders. As shown in Figure 7, customers are the most frequently cited source, appearing in 19 responses. This highlights the central role of customers in shaping requirements, underscoring the customer-centric nature of startups that aim to address specific market needs.

The product team is the second most prominent source, mentioned 13 times, indicating the importance of internal expertise and strategic alignment in identifying requirements. Similarly, internal stakeholders, such as team members and leadership, play a significant role, with 10 occurrences underscoring their contributions to decision-making processes and the product vision. While less frequent, other sources still contribute to the discovery, such as:

- **C-level** executives (5 mentions) influence strategic directions and priorities;
- **External stakeholders** and **regulators** (4 and 3 mentions, respectively) are essential in shaping compliance-related and partnership-driven requirements;
- Less-commonly cited sources include **benchmark** (2 mentions) and **leads** (1 mention), suggesting they are used in specific competitive or sales contexts.

These findings illustrate a diverse and dynamic set of requirement sources in software startups, with a predominant focus on customer needs and internal collaboration.

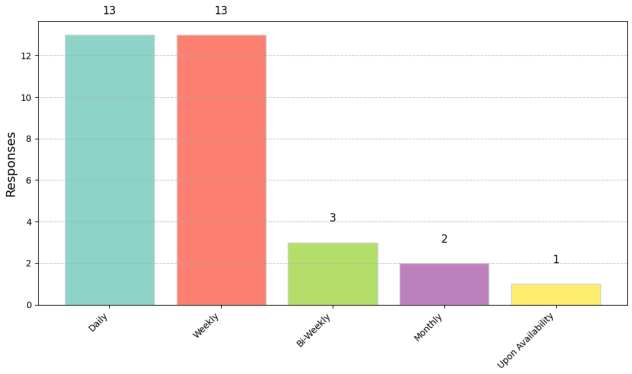

Figure 8: Stakeholder Contact Frequency

Table 3: Top 3 Discovery Techniques per Frequency

Frequency	1st Tech.	2nd Tech.	3rd Tech.
Monthly	Prototyping (2)	Brainstorming (2)	Document Analysis (2)
Bi-Weekly	Brainstorming (3)	Document Analysis (3)	Interviews (3)
Weekly	Brainstorming (12)	Document Analysis (9)	Interviews (8)
Daily	Brainstorming (13)	Document Analysis (13)	Interviews (8)
Upon Availability	Brainstorming (1)	Document Analysis (1)	Interviews (1)

The frequency of stakeholder contact varies significantly among startups (Figure 8). The results indicate that most startups maintain high-frequency interactions with stakeholders, with daily and weekly contact equally prevalent (13 occurrences each). This highlights the importance of regular, consistent communication to align requirements with evolving user needs and business objectives.

Less frequent interactions, such as bi-weekly (3 occurrences) and monthly (2 occurrences), suggest that some startups may rely on periodic check-ins, likely due to resource constraints or the nature of their development cycles. One startup reported contact only "upon availability," indicating a more *ad-hoc* approach.

These findings underscore startups' emphasis on fostering continuous stakeholder engagement to ensure real-time feedback and alignment. Daily and weekly contact, in particular, likely supports rapid iteration and responsiveness to changes, which are critical in fast-moving startup environments.

To further analyze the contribution of stakeholder involvement in RE, the top 3 discovery techniques were ranked by frequency of occurrence across stakeholder contact frequencies. Table 3 shows the most commonly used techniques at each contact interval (daily, weekly, bi-weekly, monthly, and upon availability).

Table 4: Top 3 Validation Techniques per Frequency

Recurrence	1st Tech.	2nd Tech.	3rd Tech.
Monthly	Prototype Demo (2)	Interviews (1)	Meetings (1)
Bi-Weekly	Meetings (3)	Prototype Demo (3)	Interviews (2)
Weekly	Prototype Demo (12)	Meetings (10)	Interviews (7)
Daily	Meetings (11)	Prototype Demo (10)	Interviews (6)
Upon Availability	Meetings (1)	Requirements Inspection (1)	Simulations (1)

Brainstorming, Document Analysis, and Interviews consistently rank as the top 3 techniques for daily and weekly stakeholder interactions. With Brainstorming leading (13 daily and 12 weekly mentions), this finding underscores its adaptability and effectiveness in fast-paced environments where continuous idea generation and collaboration are essential. Similarly, Document Analysis (13 daily and 9 weekly mentions) reflects the reliance on existing materials to contextualize requirements. Interviews rank third in both cases, demonstrating their importance in collecting qualitative information directly from stakeholders.

At the biweekly level, the same three techniques (Brainstorming, Document Analysis, and Interviews) remain dominant, each occurring three times. This suggests that startups prioritize techniques that enable collaborative ideation and direct feedback, even with less frequent interactions.

When stakeholder contact is less frequent, techniques such as prototyping become among the top three (e.g., two mentions per month). This shift likely reflects the use of more comprehensive artifacts, such as prototypes, to maximize the value of limited engagement opportunities. In *ad-hoc* contexts ("upon availability"), Brainstorming, Document Analysis, and Interviews remain prominent, underscoring their versatility and importance regardless of interaction frequency.

In summary, the recurrence of Brainstorming, Document Analysis, and Interviews across all frequencies highlights their central role in discovery and validation processes, with additional techniques like Prototyping becoming more relevant as interaction frequency decreases.

In addition to analyzing discovery techniques, we ranked validation techniques by their frequency across different stakeholder contact frequencies. Table 4 summarizes the most commonly used validation techniques, highlighting variations in their usage across daily, weekly, bi-weekly, and other contact intervals.

Meetings and Prototype Demos dominate the rankings for daily and weekly interactions. Meetings are the most frequently used validation technique for daily contact (11), emphasizing their role in fostering collaboration and real-time alignment among stakeholders. Additionally, we understand that the widespread agile practice of daily meetings contributed to this behavior. Conversely, Prototype Demos lead in weekly interactions (12), reinforcing the

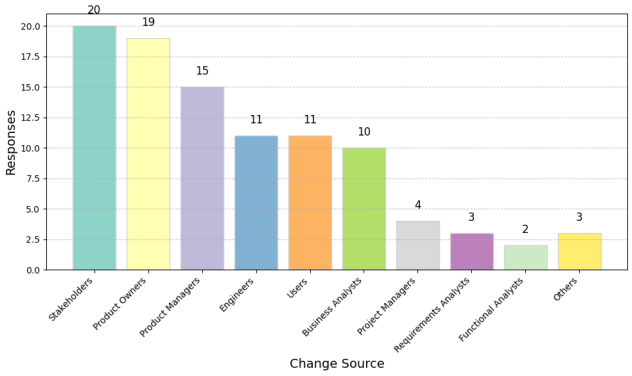

Figure 9: Change Source

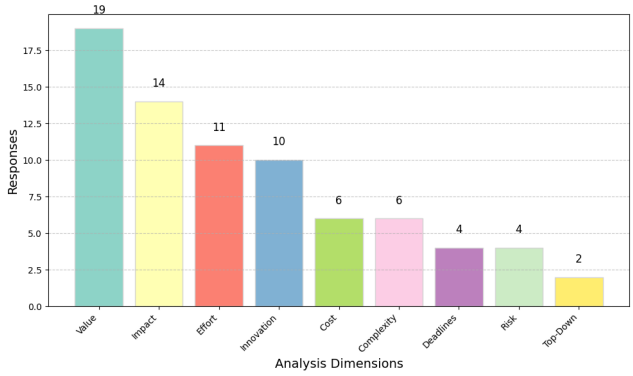

Figure 10: Prioritization Criteria

importance of demonstrating functional prototypes to gather actionable feedback. Interviews are consistently the third technique for daily (6) and weekly (7) interactions, indicating their value in providing direct stakeholder input.

The top 3 techniques remain similar at the bi-weekly level, with Meetings and Prototype Demos sharing the top spot (3 occurrences each) and Interviews ranking third (2 occurrences). This consistency suggests these techniques are versatile and effective regardless of the frequency of engagement. For less frequent contact, such as monthly or *ad-hoc* scenarios, the prominence of Prototype Demos (2 mentions monthly) remains. It likely helps stakeholders validate progress or align on requirements during infrequent interactions. Interviews and Meetings appear less frequently in these contexts, reflecting the time-consuming nature of these methods.

Ad-hoc ("upon availability") scenarios shift, with less conventional techniques like Requirements Inspection and Simulations appearing alongside Meetings. These techniques may be preferred when the focus is on structured evaluation or scenario testing during periods of sporadic stakeholder involvement.

The frequency of Meetings and Prototype Demos across all high-frequency scenarios highlights their importance in facilitating effective validation. These techniques are complemented by Interviews, which provide qualitative insights, and specialized methods such as requirements inspection and Simulations, which are used in lower-frequency or *ad-hoc* interactions.

5.4 RQ4: Proactive Requirements Management vs. React to Change Requests

The sources of requirement changes in software startups encompass a diverse range of internal and external stakeholders. As shown in Figure 9, stakeholders (20 mentions) and product owners (19 mentions) are the most common initiators of requirement changes. This highlights the critical role of external feedback and product leadership in driving adjustments to align with business goals and user needs. Product managers (15 mentions) frequently contribute to changes in requirements, reflecting their central role in balancing business objectives and development constraints.

Other significant sources include engineers and users (11 mentions each), emphasizing the role of technical feasibility and end-user input in identifying necessary adjustments. Business analysts (10 mentions) further underscore the importance of translating business insights into actionable changes.

Less frequently cited sources, such as project managers (4 mentions) and requirements analysts (3), suggest these roles are more specialized in their involvement. Even rarer mentions of functional analysts (2), and others, including software and business architects, and CEOs, indicate their limited but situationally critical input.

The criteria used to evaluate and prioritize requirement changes in software startups are diverse, reflecting the need to balance business value, feasibility, and strategic goals. Figure 10 shows that the most frequently cited criteria are value (19 mentions) and impact (14 mentions). This highlights the emphasis on ensuring that changes align with the delivery of measurable benefits to users and the business, while also assessing their broader implications.

Effort (11 mentions) and innovation (10) are also key criteria, suggesting that startups consider the resources required to implement changes, given their limitations, and their potential to differentiate the product in the market. Secondary considerations include cost and complexity (6 mentions each), which likely influence decisions in resource-constrained environments. Deadlines and risk (4 mentions each) play a lesser role but remain critical in specific contexts where timing or uncertainty are significant factors.

The consequences of changes to requirements are multifaceted and affect multiple aspects of the development process. Figure 11 highlights the most common consequences, with impact analysis and risk assessment tied as the most frequently mentioned (25 each) outcomes. This reflects the need to thoroughly understand the implications of changes to avoid negative downstream effects.

Implementation (18 mentions) and change validation (17 mentions) are also critical, indicating that startups invest effort in ensuring that changes are not only actionable but also align with stakeholder expectations and product goals. Sprint reviews (14 mentions) and backlog reviews (13 mentions) emphasize the iterative and collaborative nature of change management, enabling teams to adapt their priorities and workflows dynamically.

Finally, the change qualification (10 mentions) is less frequently noted but underscores the importance of systematically assessing the relevance and necessity of changes before proceeding. These findings underline that startups adopt structured processes to manage change while maintaining agility and alignment with objectives.

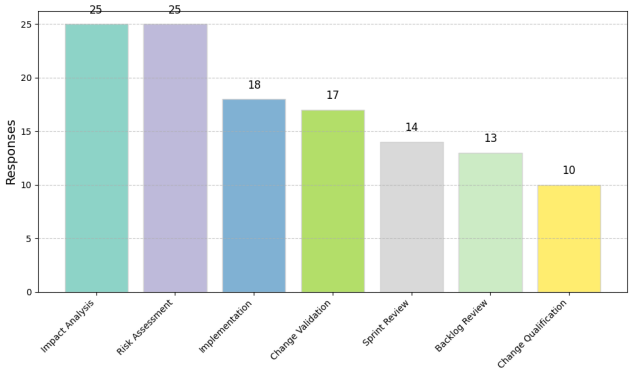

Figure 11: Change Consequences

Software startups employ a variety of management methods to effectively handle and integrate change requests, with the choice of methodology often influenced by the frequency of changes.

Agile-based approaches, such as combinations of Kanban, Scrum, and XP, are the most frequently used management methods for handling change requests, particularly when changes occur "often" or "always." This finding reflects startups' preference for methods that support iterative development and flexibility. The ability to reprioritize tasks dynamically and involve stakeholders in iterative reviews aligns with the high pace of change in startup environments.

The V-Model, which emphasizes validation and verification phases, is primarily associated with "occasionally" or "rarely" occurring changes, suggesting a preference for more structured and predictable development, such as startups operating in regulated environments. This model aligns with scenarios in which changes must be meticulously evaluated and integrated into well-defined stages.

PBB appears less frequently as a primary management method but is associated with change requests that occur "occasionally" or "rarely". This approach likely complements other agile methods by providing a mechanism for prioritizing and tracking backlog items, particularly in teams focused on incremental delivery.

The predominance of agile methods reflects their flexibility and adaptability, making them well-suited to startups operating in dynamic environments. The variation in method usage across different frequencies underscores the importance of tailoring the approach to the pace and nature of change requests. While methodologies such as Kanban, Scrum, and XP thrive in frequent change scenarios, structured approaches such as the V-Model and PBB provide value in more stable, less dynamic settings.

Finally, we highlight that no evidence was found of proactive handling of requirements changes, in the sense that the RE process is not continuously executed, and the product concept is not revisited and evolved without prior change requests.

6 Threats to Validity

The survey design is primarily quantitative, so we adopted a positivist worldview to address threats to validity [24].

In terms of Construct Validity, the design foresees three main strategies for mitigating potential misunderstandings of concepts by participants: i) the use of alternatives in closed-ended questions consolidated from empirical evidence in the literature; ii) the questionnaire was designed, reviewed, and tested in a pilot; iii) the use of a supervised approach for data collecion allowed for participants to make questions when they could not fully understand any concept.

To address internal validity, we mitigated external confounding factors by recruiting participants according to established criteria and using a supervised approach to verify participants' expertise.

In terms of Conclusion Validity, all statistics are consistent across the analysis. However, we did not formulate any prior hypotheses for testing due to the exploratory nature of the study. Additionally, the snowball sampling strategy used for recruitment precludes the calculation of confidence intervals and margins of error.

Regarding External Validity, although generalizable results cannot be claimed, we have a representative sample of Brazilian software startups, given the recruitment sources used.

7 Conclusions

Our study provides an exploratory overview of different practices and techniques employed at Brazilian software startups to support discovery, specification, and validation activities in RE processes.

The study also highlights the critical role of frequent stakeholder interactions in all stages of requirements engineering. Daily and weekly engagements are the norm, fostering agile adaptation and continuous feedback. Stakeholders such as customers and product owners are pivotal sources of requirements and change requests, underscoring the importance of aligning product development with both external and internal priorities.

Changes in requirements are managed proactively and reactively, balancing value, impact, and feasibility. Key criteria for evaluating changes include value, effort, and innovation, while consequences such as impact analysis, risk assessment, and sprint reviews ensure structured integration of changes. Agile methods, particularly Kanban and Scrum, dominate as the preferred management approaches, offering flexibility and adaptability in dynamic environments.

The interconnectedness of techniques and practices across discovery, specification, validation, and change management underscores the iterative nature of requirements engineering in startups. Techniques and management approaches are consistently chosen to optimize stakeholder involvement, rapid feedback, and iterative improvement, aligning with the lean and agile principles common in startup settings.

Future studies could expand on these findings by examining how specific industry domains, team compositions, or maturity levels influence the adoption of these practices. Additionally, investigating the long-term outcomes of these practices on product success and market fit would provide valuable insights into their effectiveness. Finally, we recognize the need for lean RE processes tailored to scenarios of instability and uncertainty in a dynamic, resource-constrained environment such as software startups.

Artifact Availability

The raw dataset cannot be made publicly available because the informed consent agreement specified that the collected data would be reported only in aggregate form. Releasing individual responses would therefore be inconsistent with the commitments made to participants and with the ethical requirements governing the study.

References

[1] Larissa Mangolim Amaral, Fábio Levy Siqueira, and Anarosa Alves Franco Brandão. 2022. A survey on requirements notations in software engineering research. In *Proceedings of the ACM Symposium on Applied Computing*. ACM, New York, NY, USA, 1291–1298. doi:10.1145/3477314.3507253

[2] Nana Assyne and Isaac Wiafe. 2019. A dynamic software startup competency model. In *Lecture Notes in Business Information Processing*. Lecture Notes in Business Information Processing, Vol. 370. Springer, 419–422. doi:10.1007/978-3-030-33742-1_35

[3] Victor R. Basili and Gianluigi Caldiera. 2000. The Goal Question Metric Paradigm. *Encyclopedia of Software Engineering - 2 Volume Set* (2000), 528–532. https://www.cs.umd.edu/$\sim$basili/publications/technical/T89.pdf

[4] Asma Batool, Yasir Hafeez Motla, Bushra Hamid, Sohail Asghar, Muhammad Riaz, Mehwish Mukhtar, and Mehmood Ahmed. 2013. Comparative study of traditional requirement engineering and Agile requirement engineering. *2013 15th International Conference on Advanced Communications Technology (ICACT)* (2013), 1006–1014.

[5] Lan Cao and Balasubramaniam Ramesh. 2008. Agile Requirements Engineering Practices: An Empirical Study. *IEEE Software* 25, 1 (2008), 60–67. doi:10.1109/MS.2008.1

[6] Karina Curcio, Tiago Navarro, Andreia Malucelli, and Sheila Reinehr. 2018. Requirements engineering: A systematic mapping study in agile software development. *Journal of Systems and Software* 139 (9 2018), 32–50. doi:10.1016/j.jss.2018.01.036

[7] Torgeir Dingsøyr, Sridhar Nerur, VenuGopal Balijepally, and Nils Brede Moe. 2012. A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software* 85 (9 2012), 1213–1221. Issue 6. doi:10.1016/j.jss.2012.02.033

[8] D. Méndez Fernández, S. Wagner, M. Kalinowski, M. Felderer, P. Mafra, A. Vetrò, T. Conte, M. T. Christiansson, D. Greer, C. Lassenius, T. Männistö, M. Nayabi, M. Oivo, B. Penzenstadler, D. Pfahl, R. Prikladnicki, G. Ruhe, A. Schekelmann, S. Sen, R. Spinola, A. Tuzcu, J. L. de la Vara, and R. Wieringa. 2017. Naming the pain in requirements engineering: Contemporary problems, causes, and effects in practice. *Empirical Software Engineering* 22, 5 (oct 2017), 2298–2338. arXiv:1611.10288 doi:10.1007/s10664-016-9451-7

[9] Brian Fitzgerald and Klaas-Jan Stol. 2017. Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software* 123 (9 2017), 176–189. doi:10.1016/j.jss.2015.06.063

[10] Peter Forbrig. 2017. Does Continuous Requirements Engineering need Continuous Software Engineering? *REFSQ Workshops* (2017).

[11] Carmine Giardino, Sohaib Shahid Bajwa, Xiaofeng Wang, and Pekka Abrahamsson. 2015. Key challenges in early-stage software startups. *Lecture Notes in Business Information Processing* 212 (2015), 52–63. doi:10.1007/978-3-319-18612-2_5

[12] Carmine Giardino, Michael Unterkalmsteiner, Nicolo Paternoster, Tony Gorschek, and Pekka Abrahamsson. 2014. What Do We Know about Software Development in Startups? *IEEE Software* 31 (9 2014), 28–32. Issue 5. doi:10.1109/MS.2014.129

[13] Catarina Gralha, Daniela Damian, Anthony I.Tony Wasserman, Miguel Goulão, and João Araújo. 2018. The evolution of requirements practices in software startups. *Proceedings - International Conference on Software Engineering* (jan 2018), 823–833. doi:10.1145/3180155.3180158

[14] Varun Gupta, Jose Maria Fernandez-Crehuet, Thomas Hanne, and Rainer Telesko. 2020. Requirements engineering in software startups: A systematic mapping study. *Applied Sciences (Switzerland)* 10, 17 (nov 2020), 6125. doi:10.3390/app10176125

[15] Irum Inayat, Siti Salwah Salim, Sabrina Marczak, Maya Daneva, and Shahaboddin Shamshirband. 2015. A systematic literature review on agile requirements engineering practices and challenges. *Computers in human behavior* 51 (2015), 915–929.

[16] Marite Kirikova. 2017. Continuous Requirements Engineering. *Proceedings of the 18th International Conference on Computer Systems and Technologies* (9 2017), 1–10. doi:10.1145/3134302.3134304

[17] Barbara Kitchenham and Shari Lawrence Pfleeger. 2002. Principles of survey research. *ACM SIGSOFT Software Engineering Notes* 27, 5 (sep 2002), 17–20. doi:10.1145/571681.571686

[18] Phillip A Laplante and Mohamad Kassab. 2022. *Requirements engineering for software and systems*. Auerbach Publications.

[19] Linaker J, Sulaman S, Johan Linåker, Sardar Muhammad Sulaman, Martin Höst, and Rafael Maiani De Mello. 2015. Guidelines for Conducting Surveys in Software Engineering. *Department of Computer Science, Lund University* May (2015), 1–64.

[20] Jorge Melegati and Alfredo Goldman. 2016. Requirements engineering in software startups: A grounded theory approach. *2016 International Conference on Engineering, Technology and Innovation/IEEE International Technology Management Conference, ICE/ITMC 2016 - Proceedings* (jun 2016). doi:10.1109/ICE/ITMC39735.2016.9026036

[21] Jorge Melegati, Alfredo Goldman, Fabio Kon, and Xiaofeng Wang. 2019. A model of requirements engineering in software startups. *Information and Software Technology* 109 (2019), 92–107. doi:10.1016/j.infsof.2019.02.001

[22] Dennis Pagano. 2011. Towards systematic analysis of continuous user input. In *SSE'11 - Proceedings of the 4th International Workshop on Social Software Engineering*. ACM, New York, NY, USA, 6–10. doi:10.1145/2024645.2024648

[23] Nicolò Paternoster, Carmine Giardino, Michael Unterkalmsteiner, Tony Gorschek, and Pekka Abrahamsson. 2014. Software development in startup companies: A systematic mapping study. *Information and Software Technology* 56 (9 2014), 1200–1218. Issue 10. doi:10.1016/j.infsof.2014.04.014

[24] Kai Petersen and Cigdem Gencel. 2013. Worldviews, research methods, and their relationship to validity in empirical software engineering research. In *2013 joint conference of the 23rd international workshop on software measurement and the 8th international conference on software process and product measurement*. IEEE, 81–89.

[25] Usman Rafiq, Sohaib Shahid Bajwa, Xiaofeng Wang, and Ilaria Lunesu. 2017. Requirements elicitation techniques applied in software startups. In *Proceedings - 43rd Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2017*. IEEE, 141–144. doi:10.1109/SEAA.2017.73

[26] Eva-Maria Schön, Jörg Thomaschewski, and María José Escalona. 2017. Agile Requirements Engineering: A systematic literature review. *Computer standards & interfaces* 49 (2017), 79–91.

[27] S M Sutton. 2000. The role of process in software start-up. *IEEE Software* 17 (2000), 33–39. Issue 4. doi:10.1109/52.854066

[28] Stefan Wagner, Daniel Méndez Fernández, Michael Felderer, Antonio Vetrò, Marcos Kalinowski, Roel Wieringa, Dietmar Pfahl, Tayana Conte, Marie-Therese Christiansson, Desmond Greer, Casper Lassenius, Tomi Männistö, Maleknaz Nayebi, Markku Oivo, Birgit Penzenstadler, Rafael Prikladnicki, Guenther Ruhe, André Schekelmann, Sagar Sen, Rodrigo Spínola, Ahmed Tuzcu, Jose Luis De La Vara, and Dietmar Winkler. 2019. Status Quo in Requirements Engineering: A Theory and a Global Family of Surveys. *ACM Trans. Softw. Eng. Methodol.* 28, 2, Article 9 (Feb. 2019), 48 pages. doi:10.1145/3306607

[29] Didar Zowghi and Chad Coulin. 2005. Requirements elicitation: A survey of techniques, approaches, and tools. *Engineering and Managing Software Requirements* (2005), 19–46. doi:10.1007/3-540-28244-0_2