On the Challenges to Documenting Requirements in Agile Software Development: A Practitioners' Perspective

Edna Dias Canedo[1], **Angelica Toffano S. Calazans**[2], **Geovana Ramos Sousa Silva**[1], **Eloisa Toffano Seidel Masson**[2], **Isabel Sofia Brito**[3]

[1]University of Brasília (UnB) –Brasília – DF –Brazil

[2]University center – UniCEUB – Brasília – DF –Brazil

[3]Polytechnic Institute of Beja – Beja – Portugal
ednacanedo@unb.br, angelica.toffano@gmail.com, geovannna.1998@gmail.com, eloisa.masson@ceub.edu.br, isabel.sofia@ipbeja.pt

Abstract. *Agile Software Development (ASD) is an iterative and incremental methodology designed to accelerate project deliveries. In this dynamic environment characterized by constant changes, the task of documenting requirements becomes increasingly challenging, leading to the emergence of the technical debt issue. This research involved a survey of 84 practitioners to identify the techniques and practices employed in documenting software requirements within ASD teams, as well as their perceptions of the documentation process and the challenges, regarding what factors influence it and its consequences. Our key findings indicate that user stories are the most commonly utilized technique by practitioners for documenting requirements. Furthermore, a deficient documentation process results in two primary consequences: rework and a knowledge deficit. To address these challenges, various techniques are implemented across different development phases, including requirement refactoring, documentation refinement meetings, and template reviews. Participants also emphasized the significance of having a requirements expert to enhance the documentation process and expressed uncertainty regarding the adequacy of their existing requirements documentation.*

1. Introduction

Over the past two decades, the adoption of agile methodologies has increased. However, numerous studies have highlighted deficiencies in both functional and non-functional requirements (NFR) documentation within Agile Software Development (ASD) context [Theunissen et al. 2022, Behutiye et al. 2020, Behutiye et al. 2022]. ASD promotes continuous software delivery with a focus on value addition and minimal documentation while accommodating late changes to meet dynamic business demands. However, the rapid pace of change in this methodology creates challenges in requirements documentation, leading to the emergence of technical debt within ASD contexts[Behutiye et al. 2017, Mendes et al. 2019]. Technical debt serves as a metaphor to convey the repercussions of poor software development practices to non-technical stakeholders [Behutiye et al. 2017]. According to Lenarduzzi and Fucci [Lenarduzzi and Fucci 2019], there are 10 types of Technical Debt (TD): Requirements, Architectural, Design, Code, Test, Build, Documentation, Infrastructure, and Versioning.

Technical debt associated with requirements, often referred to as requirements debt, manifests itself when there is underspecification of requirements, meaning that they are outdated, incomplete, or inadequately documented [Behutiye et al. 2017, Mendes et al. 2019]. More broadly, when we refer to requirements debt, it means "the distance between the optimal requirements specification and the actual system implementation, under domain assumptions and constraints" [Ernst 2012]. There are three types of requirements debt [Lenarduzzi and Fucci 2019]: 1) incomplete user needs, when user needs are neglected; 2) requirement smells, when the user needs expressed through feedback channels (i.e., app stores, social media, interviews) are neglected; and 3) mismatch implementation. These categories directly relate to software requirements.

Documentation technical debt encompasses issues related to nonexistent, inadequate, or incomplete project documentation [Mendes et al. 2019]. Agile teams commonly employ documentation techniques like user stories and story cards, which can introduce limitations in software requirements and their documentation, often oversimplifying them [Marek et al. 2021, Jarzebowicz and Weichbroth 2021]. The lack of documentation is considered one of the most significant weaknesses of Requirements Engineering. The inadequacy of user story formats and the challenge of validating their consistency and verifiability reinforce the deficiency in documentation in ASD contexts [Curcio et al. 2018].

The consequences of technical debt in the context of ASD are far-reaching, including [Mendes et al. 2019]: inefficient testing, user dissatisfaction due to inconsistent documentation, difficulties in knowledge transfer to new team members, software maintenance and evolution problems, rework during implementation and testing, limited understanding of the system among team members, ineffective artifact utilization due to disorganized information, duplicated efforts to compensate for documentation gaps, communication problems due to poor-quality or missing documentation, schedule delays, and increased project costs due to documentation issues. Additionally, difficulties in understanding documentation and a lack of synchronization between software and its documentation compound these challenges within ASD teams [Theunissen et al. 2022].

Several authors have investigated the importance of increasing the quantity and quality of documentation in ASD teams, identifying it as a persistent challenge [Theunissen et al. 2022, Curcio et al. 2018, Behutiye et al. 2022]. According to Curcio et al. [Curcio et al. 2018], requirements engineering in the context of ASD is still immature and needs more research, particularly studies that present empirical evaluation. In this context, this work aims to identify practices and techniques utilized in literature and industry for software requirements documentation within ASD teams and assess how the artifacts support this documentation. The study also investigates ASD teams' perceptions, in terms of what factors influence it and its consequences, regarding their current documentation practices and the proposed solutions for addressing identified challenges, especially concerning the impact of technical debt on requirements and documentation.

Our main findings highlight that user stories are the predominant technique for documenting software requirements in ASD teams, and the development team's knowledge of the software solution plays a significant role in requirements documentation. Moreover, the primary consequence of absent or outdated requirements documentation is the need for rework in delivered software and a resultant decline in quality. Our findings further underscore that: i) requirements documentation is typically not carried out by practitioners with specific requirements engineering expertise; ii) maintaining updated and high-quality requirements documentation poses a challenge; iii) there is a lack of a cultural emphasis on continuous requirements documentation within the agile process.

2. Research Design

This study investigates how software requirements are documented by Agile Software Development (ASD) teams, what factors influence it, as well as the consequences of missing or outdated requirements documentation. To reach this goal, we defined the following research questions (RQs): RQ.1. How do ASD teams document software requirements? RQ.2. What factors influence requirements documentation and lack of documentation in ASD? RQ.3. What techniques do practitioners use in requirements documentation to make it easier to understand and communicate with stakeholders? RQ.4. What issues or challenges do ASD teams face in requirements documentation?

We opted for an online survey to elicit data and apply qualitative and qualitative analysis. We followed the guideline of Linaker et al. [Linaker et al. 2015]. The survey was chosen as the primary research method to capture practitioners' perceptions of how ASD teams document software requirements, what factors influence documentation, what techniques teams use, and the challenges they face in their daily activities. Therefore, the survey was performed based on two criteria: (1) obtain practitioners' profiles; (2) collect practitioners' perceptions. The survey presented open-ended and close-ended questions for both criteria.

The survey population consisted of practitioners working with agile software development from several organizations in Brazil and Portugal. A control question was included at the beginning of the survey to filter the participants and ensure that responses were only from the target audience. The participants were recruited through our personal contacts, and we also advertised the questionnaire on our social networks for self-recruitment of participants. All of this paper's authors were involved in designing and validating the survey questions. Two authors, with more than 23 years of experience in software development and requirements elicitation, described the survey questions, and the other authors (two) validated them. The survey consists of 21 questions, nine closed-ended and twelve open-ended questions. All survey questions and answer options are available in Zenodo at https://zenodo.org/records/10157806.

We used the Google Forms platform [1] to create the survey. At the beginning of the research, we presented the consent form, including the conditions and stipulations. We also presented our contact information (email). The survey was anonymous and respondents were not asked to provide any contact information. However, we asked them to provide their e-mail address if they wished. The survey was available from October 16th to November 15th, 2023 (30 days). The average response time of each participant to answer the survey was 33 minutes and 2 seconds. In total, 84 practitioners responded to the survey. After preparing the survey questions, we conducted a pilot. We validated the survey pilot with three researchers from the study and 5 representatives from agile software development teams. Grounded Theory (GT) was used for qualitative analysis to perform open and axial coding [Corbin and Strauss 2014] on the open questions.

[1]https://www.google.com/forms

3. Results and Discussions

We collected the responses of 84 practitioners, which are available in Zenodo at https://zenodo.org/records/10157806. Most participants have considerable experience in software development and agile software development and work in the requirements elicitation, development/programming, or maintenance/evolution phases of the agile software development project lifecycle in which they participate.

3.1. RQ.1. How do ASD teams document software requirements?

To answer RQ.1, we asked practitioners how requirements are documented in their ASD projects. The answers were analyzed with Grounded Theory [Corbin and Strauss 2014]. To enable traceability, we include a direct quote from one respondent along with the answer identified in open-ended questions and we present the discovered codes slanted: "The documentation of requirements by the ASD team is performed using user stories, and they are sent to the development team to implement sprints based on the importance and priority of the requirement." **Main code:** *Requirements documentation is performed using user stories.* – **Category:** *Technique* – **Subcategory:** *User Stories.* The most used technique by ASD teams to document software requirements are `User Stories and Business Rules`.

One of the participants mentioned that "Stakeholder involvement is imperative and the documentation must evolve iteratively, and all documentation must be created responsibly and reflect all the system's needs. In addition, the analysis of what needs to be included in the requirements documentation must reflect the needs of the organization, the project's specific needs, the team's, and the business rules". Another participant mentioned that "the software requirements documentation must be elaborated by the requirements engineer with the participation of the stakeholders. In addition, it is important to use prototypes and user stories to facilitate communication between team members and stakeholders, to resolve all doubts and/or inconsistencies".

It is interesting to point out that Marek et al. [Marek et al. 2021] and Jarzebowicz et al. [Jarzebowicz and Weichbroth 2021] also identify user stories as the techniques commonly used by ASD teams, although the authors also identified a common problem with the use of this technique: limitation and simplification of requirements. According to Curcio et al. [Curcio et al. 2018], the lack of data in the user stories templates and the difficulty in validating their consistency and verifiability are factors that contribute to the fragility of Requirements Engineering in the context of ASD. Heikkila et al. [Heikkilä et al. 2015] mentioned the insufficiency of the user story format, stating that properties such as consistency and verifiability of user stories are challenging to validate and user stories do not convey enough information for software design, and separate system and subsystem requirements are required, among other issues. In the answers to this question in the survey, some terms were mentioned by practitioners, such as: "insufficient", "negligent", "without due rigor", and "quickly". Thus, our findings allow us to infer that despite being the technique most used by ASD teams, this technique has some challenges that have not yet been resolved.

According to Perera et al. [Perera et al. 2023], requirements documentation debt is the insufficient or incomplete requirements specifications found in use cases and user stories. Furthermore, they are low-quality specifications or do not describe the software under development. The authors also mention outdated requirements, which are cases where specifications were developed to an appropriate level of quality in early versions of the software but are subsequently not updated with new requirements or changes to existing requirements. In our findings, user stories are the most used techniques in the literature for requirements documentation in ASD. This allows us to conclude that even though it is a technique that presents several problems, it is the one most used by software development teams.

RQ.1 Summary: Most practitioners are experienced in software development (more than ten years – 58.2%) and ASD (more than four years) and work in the requirements elicitation phase and development/programming. The ASD teams consider the requirements documentation important and should be performed by a RE practitioner, and the technique they use most is the user stories.

3.2. RQ.2. What factors influence requirements documentation and lack of documentation in ASD?

To answer RQ.2, we asked practitioners what factors might influence requirements documentation in ASD. Most participants (84%) stated the team's knowledge of the software solution; for 62%, the team members turnover; for 62%, having requirements specialists on the team; for 57%, the involvement of business representatives during software development; for 55%, the organization's requirements, such as processes and auditing; for 44%, the deadline for each iteration; for 41%, the contractual or other agency requirements; and for 35%, the criticality/risk of the solution.

Our results partially confirm the findings of Mendes et al. [Mendes et al. 2019], who identified as possible causes of documentation technical debt the lack of technical capacity of the practitioner responsible for documenting the requirements, high staff turnover, poor time management, political influences and hierarchies of the organization. Our findings are also similar to those of Behutiye et al.[Behutiye et al. 2017], who identified a lack of understanding of the system being built. In another work, Behutiye et al. [Behutiye et al. 2020] identified that the experience of developers might affect the documentation of quality requirements. Our findings ratify the findings in the literature and indicate the possibility of adding two more causes of documentation technical debt: 1) contractual or other agency requirements and 2) the criticality/risk of the solution. We also ask about the consequences of missing documentation or outdated documentation in ASD. Below is one of the statements made by a participant:"The main consequence of the absence of requirements documentation is communication problems, both between the requirements teams and the development team and between the contracting organization and the stakeholders. In addition, there may be requirement prioritization and rework issues."

Table 1 presents the coding performed to identify in survey responses what are the consequences of missing or outdated documentation. Among the categories of consequences, although rework has only one subcategory, it was mentioned by 47 participants, whereas *Communication* has the most subcategories but totaled 32 mentions. Among the subcategories, the most significant consequence was `Rework over requirements implementation`, followed by `Lack of quality in deliveries` with 28 mentions and `Project planning issues` with 17 mentions.

Table 1. Consequences of the lack or non-update documentation

Category	Subcategory	#Cited
Rework	Rework over requirements implementation	32
Communication	Communication difficulties among team members and stakeholders	11
	Difficulties to execute the products' maintenance or evolution	8
	Difficulties in requirements implementation	6
	Difficulties in understanding the software	3
	Lack of reference for team members	2
Prioritization	Difficulty in prioritizing requirements	5
Planning	Project planning issues	15
	Project schedule issues	11
	Project cost issues	9
Software Quality	Lack of quality in deliveries	17
	Dissatisfaction of end users and stakeholders	5
Risks	Non-compliance risks	7
	Financial risks	5
	Information security risks	3

These results partially ratify the findings by Mendes et al. [Mendes et al. 2019], who identified as consequences of faulty documentation: the rework over implementation and tests; communication problems due to the poor quality or absence of documentation; problems in the maintenance of the software; the difficulty of the team members having complete knowledge of the system that was being developed; lack of perception of the existing risks due to the lack of documentation. Behutiye et al. [Behutiye et al. 2017] also identified system quality degradation, increased cost of maintenance, and complete redesign or rework of the system as a consequence. Our findings also confirm these results found by the authors.

We asked which documentation issues are most relevant in ASD, and practitioners mentioned the lack of documentation for functional and non-functional requirements and system architecture, as seen in other works [Behutiye et al. 2020, Behutiye et al. 2022, Robiolo et al. 2019]. In addition, we also identified that the documentation needs to be updated at each change of the software, also reported by Theunissen et al. [Theunissen et al. 2022] for the smooth running of projects. One of the participants mentioned that "the lack of specification of business requirements that impact legal requirements, such as the Brazilian General Data Protection Law (LGPD)[Canedo et al. 2022, Canedo et al. 2021] or General Data Protection Regulation (GDPR) [Aberkane et al. 2021] may cause legal sanctions, such as changing legislation changing tax calculations or changing input/output layouts". Theunissen et al. [Theunissen et al. 2022] classified the information necessary for documenting the requirements in context and environment, which encompasses anything that affects the system but is not included in the primary goals, such as legal and environmental issues. The authors stated that the absence of this information could lead to documentation that is often lacking, incomplete, out-of-date, or of low quality.

Another participant mentioned that "The most relevant documentation problems in ASD are the difficulty developers find in implementing the functional and non-functional requirements, the dissatisfaction of the stakeholders and the team's lack of responsibility in the project's initial phases in carrying out a good requirements documentation". This statement confirms the previous findings of Behutiye et al. [Behutiye et al. 2017, Behutiye et al. 2020] and Mendes et al. [Mendes et al. 2019]. Perera et al. [Perera et al. 2023] also stated that although requirements technical debt is similar in many ways to technical debt in software code, it also has its own components, such as requirements artifacts, which have a cycle feedback that involves the user to accurately collect their needs. This involves additional costs due to sub-optimal decisions concerning requirements, which may occur during Requirements Engineering and Software Implementation activities. Our results presented in Table 1 corroborate the authors, where the rework category is the most mentioned by survey participants, followed by software quality and planning.

We also investigated what solutions are applied by the ASD teams when these documentation issues happen. The participants mentioned the following measures: requirements refactoring; recurrent meetings with the team to refine the documentation (layout, business rules, technical validation, and business validation) since the beginning of the process; review of the documentation template, workflow, or specification; correction of the requirements documentation; and improvement of the requirements detailing to depict precisely the functionalities and their respective restrictions. Regarding who has the responsibility of documenting the software requirements in the organization in which the ASD teams work, they informed that they are carried out by 1) senior software engineers together with the lead UX designer; 2) product owner; 3) scrum master; 4) all team members; 5) requirements analyst; 6) business analyst; 7) developers; 8) project managers; and 9) architects. According to the perception of most survey participants, this attribution by the ASD team of the practitioner responsible for documenting the software requirements in the organizations they work with is incorrect. Some participants mentioned: "This assignment is not correct. The organization should hire a requirements specialist to carry out the entire Requirements Engineering phase of the software to be developed by the ASD team."; and "No. The elicitation, documentation, and validation of software requirements should be carried out by a requirements practitioner who knows the creative techniques for eliciting requirements."

RQ.2 Summary 2: The main consequence of the lack of requirements documentation is rework and lack of quality in product deliveries. The factors that influence requirements documentation are the team's lack of knowledge of the software solution, staff turnover, and lack of requirements experts on the team.

3.3. RQ.3. What techniques do practitioners use in requirements documentation to make it easier to understand and communicate with stakeholders?

It is important to highlight that to answer this RQ, it was decided to research the techniques used to document requirements in three contexts: 1) the techniques used in organizations by participants; 2) the techniques used, in the participants' perception, to facilitate understanding and communication with customers; and 3) the techniques used, in the participants' perception, to facilitate understanding and cooperation between analysts and developers. According to Bomström et al. [Bomström et al. 2023], practitioners with different perspectives require information in different ways to complete their daily tasks and it is essential to identify the techniques used to understand this information.

Regarding the techniques used by requirements analysts/engineers to document software requirements in order to understand them or facilitate understanding and communication with interested parties, the most used techniques used by ASD teams are user stories (69%), business rules (60%), meetings (60%), prototyping (51%), functional description (43%), epics (33%), requirements list (27%), organization's own templates and scenarios (21%). Regarding which techniques analysts use to document the requirements for cooperation purposes between analysts and development teams to facilitate the maintenance/evolution of the product, the most used techniques are user stories (70%), meetings (49%), prototyping (46%), rule of business (46%), functional description (41%), list of requirements (31%), use cases (31%), epics (26%), organization-specific templates (20%), UML models (14%), and change decisions recording made by stakeholders (13%).

Concerning requirements specification techniques practitioners use to document requirements in ASD teams, user stories (69%) are the most commonly used technique, followed by meetings, business rules, prototyping, functional descriptions, use cases, epics, requirements lists, and the organization's own templates. These results support the findings by [Jarzebowicz and Sitko 2019], who identified user stories as the most commonly used technique by practitioners for requirements elicitation with stakeholders and for carrying out software maintenance or evolution. The authors also identified the use of epics, use cases, and prototyping techniques. As well as Jarzebowicz et al. [Jarzebowicz and Sitko 2019], we have identified a small variation in the use of these techniques by ASD teams.

We identified little variation in the use of requirements specification techniques by ASD teams, considering the two contexts analyzed in this research, customers and development teams. However, we identified a greater use of user stories by software development teams and a lesser use of this technique by customers. These findings allow us to infer that the technique most used by ASD teams is user stories. This finding corroborates the findings of Mendes et al. [Mendes et al. 2019]. However, Heikkila et al. [Heikkilä et al. 2015] stated that user stories might not be sufficient when used as the sole source of requirements documentation, leading to difficult prioritization of requirements. This also allows us to infer that part of the problems related to requirements documentation can be derived from problems already identified in previous studies due to the use of user stories. Considering that the agile process follows the software development life cycle that includes requirements specification, analysis, design, coding, testing, and delivery of partially implemented software [d Darothi Sarkar and Gupta 2012], [Sajjade 2020], we chose to rename these phases to facilitate understanding, and inserted the Modeling, Maintenance/Evolution and Management phases, as per the Swebok version 3 [Bourque 2023] and considering that other authors also identify these other phases in the software development cycle [Saravanan et al. 2020], [Gurung et al. 2020].

A significant number of practitioners (71%) reported that the techniques are used to document software requirements in the Requirements Specification phase of the software development cycle; 57% of them reported that the documentation is carried out in the Analysis and Design phase; 54% respondents said they are in the Development/Programming phase; 35% practitioners stated that it is in the Modeling phase; 31% that documentation is carried out in the Maintenance/Evolution phase; 20% practitioners claimed to be in the Testing phase; and 20% of them are in the Project Management phase. This result is not a surprise to us, since some of the techniques mentioned by the participants (meetings, scenarios, and personas) were also mentioned by Chang et al. and Dev et al. [Chang et al. 2008, Dev et al. 2023] as techniques that are used in other phases of agile software development process. This finding allows us to infer that part of the techniques used by ADS teams can and are used, even in smaller quantities, at different stages of the development process.

Most practitioners (86%) stated that the requirements artifacts derived from using the techniques to document software requirements are used later in the Development/Programming phase of the software development lifecycle. Some participants (62%) stated that artifacts are used in the Testing phase, 58% stated that they are used in the Maintenance/Evolution phase, 35% stated that they are used in the Management phase, and 35% stated that it is in the modeling, respectively. 29% of the practitioners stated that they are used in the Analysis and design phase. Finally, 23% of them stated they are used in the Elicitation phase. Our findings differ from those of Alsaqqa et al. [Alsaqqa et al. 2020]. According to the authors, documents are rarely used after the requirements identification and documentation phases, and project progress is measured through software testing rather than documentation. Our findings allow us to infer that, in practice, the use of documentation by ASD teams is more constant and frequent, especially in the requirements elicitation and development/programming phase. Heimicke et al. [Heimicke et al. 2020] presented several proposals for hybrid agile models, recognizing the need to adapt the agile manifesto [Fowler et al. 2001] in the software industry. Thus, this recognition of the use of documentation in all stages of the software development process confirms the need to make some adaptations to the principles of the manifest.

Some practitioners mentioned that the artifacts generated to document the requirements are sufficient to ensure an adequate software evolution. Below are some statements: "The generated artifacts clearly describe what should be done by the development team and help the testing team on how they should proceed."; and "By registering the business needs and the needs identified in the execution of the sprints, applying the proper prioritization, the need for two teams (one for development and one for support) is avoided, the team specializes in the product, avoiding rework and promoting a constant delivery of value to the business." Other practitioners stated that the artifacts generated in the documentation of software requirements are not sufficient to ensure proper software maintenance and evolution: "It is not enough, other documents are necessary to guarantee the maintenance of the product, for example, diagrams of behavior and structure of the system, modeled from the requirements."; and "It is not enough, the evolution of the product also depends on the management of the knowledge base of the artifacts produced by the requirements documentation. Thus, the access, organization, and understanding of the artifacts produced need to be simple and of good quality."

Regarding practices and techniques used to document requirements being consistent with Agile guidelines, practitioners mentioned that: "Are perfectly adherent to Agile. The model is for identifying and building the product backlog that will be maintained and evolved throughout the project, both by the needs identified by the business and by the needs identified by the development team. The requirements prioritization determines what goes into each sprint according to the business need and the team's ability to deliver, constantly adding value. The other characteristics of agile are preserved and can be adjusted according to the maturity and capacity of each organization."; and "It is adherent because it favors face-to-face communication and reduces waste on volatile or non-reusable artifacts. "

In the perception of a few practitioners, the practices and techniques currently adopted by ASD teams are not adherent to Agile: " I understand not. Agile privileges software over comprehensive documentation. Furthermore, it is not clear what an ideal documentation should be. In financial systems with risk of damage to the organization, internal audits and controlling bodies that require documentation, and high turnover of teams, documentation is essential and necessary, contradicting the Agile principles."; " The practices and techniques we adopt are not Agile compliant. We use cases to document the requirements because we believe that for the proper maintenance of the systems, it is necessary to have good documentation." Summarizing some answers about the adherence of documentation practices to agile, practitioners mentioned that "when complete"; "if updated"; "if they follow the evolution of the product"; "yes, but the implicit knowledge of the participants is essential"; "lack of sufficient artifacts". This allows us to infer that several factors may interfere with whether or not the artifacts generated to document the software requirements are sufficient to perform maintenance. It is noteworthy that this finding is different from other studies [Behutiye et al. 2017, Mendes et al. 2019] which identified that the documentation performed by ASD teams could be considered a technical debt, as they have several problems that result in problems such as rework, low productivity, and communication failures.

RQ.3 Summary: The techniques most used by ASD teams to document software requirements are user stories, business rules, and meetings. Furthermore, practitioners stated that these techniques are used mainly in the requirements elicitation and development phases. Small differences in usage were identified between the techniques used with customers and the techniques used by the development team.

3.4. RQ.4: What issues or challenges do ASD teams face in requirements documentation?

To answer RQ.4, we asked practitioners what challenges related to specific ASD documentation and practices they face daily in their projects. In total, 71 participants reported the challenges their teams faced in documenting software requirements, which were coded into categories. Table 2 presents all categories and subcategories identified in the process of coding the challenges faced by ASD teams in requirements documentation. We have identified 4 categories: 1) Training; 2) Knowledge; 3) ASD team culture; and 4) Documentation. The category most mentioned by participants was Training – participants mentioned that the lack of trained professionals makes it difficult to correctly understand systems requirements, leading to a very high occurrence of turnover in ASD teams. They believe that this phenomenon occurs because professionals working in Agile Software Development (ASD) need to stay consistently updated, undergoing improvement courses to enhance their job performance.

Some participants mentioned that the lack of training means that others do not know the techniques and tools available in the literature and on the market to carry out requirements documentation. The second most mentioned category was Knowledge –

participants believe that the lack of requirements specialists in ASD teams generates several problems, such as difficulty in keeping functional requirements documentation updated and accessible to all team members and in selecting existing techniques and tools in the literature to document requirements. Some of these challenges presented in Table 2 were also identified by Theunissen et al. [Theunissen et al. 2022], Behutiye et al. [Behutiye et al. 2017], and Mendes et al. [Mendes et al. 2019]. The practitioners' perceptions presented in Table 2 allow us to infer that the requirements documentation in the ASD may suffer from a problem inherent to the agile process itself due to the culture/idea that the product coding stage should start as early as possible fast as possible. Theunissen et al. [Theunissen et al. 2022] identified that documentation is hard to understand, considered a waste when it does not appear to contribute directly to the end product, and is easily out-of-sync with implementation. These concepts inherent to agile models can hinder or compromise the identification, specification, documentation, validation, updating, and tracking of software requirements.

Table 2. Challenges faced by ASD teams in requirements documentation

Category	Subcategory	#Cited
Training	Lack of trained professionals	56
Knowledge	Lack of a qualified professional	54
	Lack of requirements specialist to carry out the specification activity	44
	Lack of requirements professionals on teams	38
	Lack of a practitioner on the team with knowledge of existing techniques and practices for documenting requirements	37
	Difficulty in registering the relevant facts of the sprint and the lessons learned	33
ASD team culture	Requirements documentation is not necessary	51
	Turnover on ASD teams	47
	Lack of commitment from team members to document requirements decisions	11
Documentation	Lack of documentation	39
	Keeping the documentation up to date	38
	Lack of documentation of business rules	33
	Misconception that the agile methodology does not require documentation	32
	Lack of understanding of the importance of documentation	27
	Poorly specified and incomplete requirements	19

Our findings collaborate with the findings of Perera et al. [Perera et al. 2023] who stated that the causes of requirements technical debt are: a) neglecting user needs; b) missing to capture user feedback from one or more user feedback channels and ambiguities introduced during formalization of requirements smells; c) implementing a suboptimal solution to a requirements problem and insufficient and incomplete requirements missed due to the lack of requirements traceability or documentation inefficiencies occurred intentionally or unintentionally; d) inconsistent management of requirements from different stakeholders; e) requirements that are only partly implemented; and e) inadequate or poorly conducted requirements elicitation and analysis.

RQ.4 Summary: In the practitioners' perception, the challenges related to documentation and practices in ASD are the lack of trained practitioners in the requirements area to carry out the RE phase, lack of knowledge, and the culture of the ASD teams and organizations that documentation is not an important artifact.

4. Threats to Validity

External validity: first, given the delicacy and complexity of the subject, we cannot guarantee that participants expressed their thoughts without considering them professionally or socially desirable. To mitigate this threat, we informed participants that the information would not be made available to the organizations they work for. They were only used within the scope of this research. Secondly, the sample of participants can be considered small and only represents the perception of Brazilian and Portugal agile practitioners. To mitigate this threat, we collected the perceptions of different practitioners from different organizations and states in Brazil. Capturing the perceptions of practitioners working on ASD teams in other countries may require more effort and the use of multiple techniques. Thus, we cannot generalize the results obtained. Furthermore, participants' answers are not necessarily coherent and consistent and may contradict their words and activities in the labor market. Despite this, practitioners' perception demonstrates the complexity of Requirements Engineering activities, which emphasizes the importance of research concerning the lack of recognition of the importance of requirements documentation in ASD.

Construct validity: despite our confidence in the scientific rigor employed in conducting the study, we acknowledge the inevitability of participants' subjective interpretation of the questions. To mitigate this threat, we validated the survey with three researchers from the study and five representatives from agile software development teams. Conclusion validity: although we are confident in the scientific rigor of conducting the study, it is inevitable the subjectivity in the researchers' interpretation of the data [Seaman 1999]. To mitigate this threat the analysis of the survey's open questions was first carried out by two researchers, and a third researcher carried out the validation. In addition, any inconsistencies were discussed with the fourth researcher. These measures also allowed to avoid bias and validate the closed questions, ensuring that the list of options is sufficiently complete. Despite this, open-ended questions allow respondents to report factors beyond the list of options in closed-ended questions. Internal validity: collecting survey data allowed us to minimize threats to internal validity because the participants were selected randomly through email contacts, and the researcher did not interact with the participants. We also advertised the questionnaire on our social networks for self-recruitment, so there was no influence from the researcher on the results. Moreover, the survey has a description and questions designed to identify practitioners who meet the desired criteria, i.e. practitioners working with agile software development.

5. Conclusions and Future Work

The techniques most used by ASD teams are user stories and business rules. User stories have several problems when documenting requirements, although they are the most used by ASD teams. In the perception of the ASD teams, the main factors that influence the documentation of the requirements in the ASD are the knowledge of the software solution by the team members, team member turnover, having a requirements specialist in the team, the involvement of the business representatives during the development of the software, and the business and audit processes of their organizations.

Most practitioners mentioned that the lack or non-update of requirements documentation in ASD generates rework in the implementation of software requirements and various difficulties related to communication, mainly between agile teams and stakeholders. Furthermore, the lack of documentation significantly impacts project planning and software quality. We identified some actions that ASD teams use to solve problems related to failures in requirements documentation, such as refactoring the requirement, holding meetings with the ASD team to refine documentation, reviewing the template/flowchart used to carry out requirement documentation, and updating/correcting requirements documentation. As future work, we intend to replicate this research with practitioners from other countries to understand the challenges of the lack of documentation and what are the agile principles that led to the incorporation of the culture that agility values only the quick delivery of the product without taking into account the quality of the artifacts generated during the software development lifecycle, mainly from the software requirements documentation.

References

Aberkane, A., Poels, G., and vanden Broucke, S. (2021). Exploring automated gdpr-compliance in requirements engineering: A systematic mapping study. *IEEE Access*, 9:66542–66559.

Alsaqqa, S., Sawalha, S., and Abdel-Nabi, H. (2020). Agile software development: Methodologies and trends. *Int. J. Interact. Mob. Technol.*, 14(11):246–270.

Behutiye, W., Rodríguez, P., and Oivo, M. (2022). Quality requirement documentation guidelines for agile software development. *IEEE Access*, 10:70154–70173.

Behutiye, W., Seppänen, P., Rodríguez, P., and Oivo, M. (2020). Documentation of quality requirements in agile software development. In Li, J., Jaccheri, L., Dingsøyr, T., and Chitchyan, R., editors, *EASE '20: Evaluation and Assessment in Software Engineering, Trondheim, Norway, April 15-17, 2020*, pages 250–259. ACM.

Behutiye, W. N., Rodríguez, P., Oivo, M., and Tosun, A. (2017). Analyzing the concept of technical debt in the context of agile software development: A systematic literature review. *Inf. Softw. Technol.*, 82:139–158.

Bomström, H., Kelanti, M., Annanperä, E., Liukkunen, K., Kilamo, T., Sievi-Korte, O., and Systä, K. (2023). Information needs and presentation in agile software development. *Inf. Softw. Technol.*, 162:107265.

Bourque, P. (2023). Software Engineering Body of Knowledge (SWEBOK). [Online; accessed 18. Nov. 2023].

Canedo, E. D., Calazans, A. T. S., Bandeira, I. N., Costa, P. H. T., and Masson, E. T. S. (2022). Guidelines adopted by agile teams in privacy requirements elicitation after the brazilian general data protection law (LGPD) implementation. *Requir. Eng.*, 27(4):545–567.

Canedo, E. D., Calazans, A. T. S., Cerqueira, A. J., Costa, P. H. T., and Masson, E. T. S. (2021). Agile teams' perception in privacy requirements elicitation: Lgpd's compliance in brazil. In *29th IEEE International Requirements Engineering Conference, RE 2021, Notre Dame, IN, USA, September 20-24, 2021*, pages 58–69. IEEE.

Chang, Y., Lim, Y., and Stolterman, E. (2008). Personas: from theory to practices. In Gulz, A., Magnusson, C., Malmborg, L., Eftring, H., Jönsson, B., and Tollmar, K., editors, *Proceedings of the 5th Nordic Conference on Human-Computer Interaction 2008, Lund, Sweden, October 20-22, 2008*, volume 358 of *ACM International Conference Proceeding Series*, pages 439–442. ACM.

Corbin, J. and Strauss, A. (2014). *Basics of qualitative research: Techniques and procedures for developing grounded theory*. Sage publications.

Curcio, K., Navarro, T., Malucelli, A., and Reinehr, S. S. (2018). Requirements engineering: A systematic mapping study in agile software development. *J. Syst. Softw.*, 139:32–50.

d Darothi Sarkar, S. S. and Gupta, D. (2012). Agile processes and methodologies: A conceptual study. *International Journal on Computer Science and Engineering (IJCSE)*, 4 (05).

Dev, J., Rashidi, B., and Garg, V. (2023). Models of applied privacy (MAP): A persona based approach to threat modeling. In Schmidt, A., Väänänen, K., Goyal, T., Kristensson, P. O., Peters, A., Mueller, S., Williamson, J. R., and Wilson, M. L., editors, *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems, CHI 2023, Hamburg, Germany, April 23-28, 2023*, pages 189:1–189:15. ACM.

Ernst, N. A. (2012). On the role of requirements in understanding and managing technical debt. In Kruchten, P., Nord, R. L., Ozkaya, I., and Visser, J., editors, *Proceedings of the Third International Workshop on Managing Technical Debt, MTD 2012, Zurich, Switzerland, June 5, 2012*, pages 61–64. IEEE/ACM.

Fowler, M., Highsmith, J., et al. (2001). The agile manifesto. *Software development*, 9(8):28–35.

Gurung, G., Shah, R., and Jaiswal, D. P. (2020). Software development life cycle models-a comparative study. In *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*.

Heikkilä, V. T., Damian, D. E., Lassenius, C., and Paasivaara, M. (2015). A mapping study on requirements engineering in agile software development. In *41st Euromicro Conference on Software Engineering and Advanced Applications, EUROMICRO-SEAA 2015, Madeira, Portugal, August 26-28, 2015*, pages 199–207. IEEE Computer Society.

Heimicke, J., Chen, R., and Albers, A. (2020). Agile meets plan-driven – hybrid approaches in product development: A systematic literature review. *Proceedings of the Design Society: DESIGN Conference*, pages 577–586.

Jarzebowicz, A. and Sitko, N. (2019). Communication and documentation practices in agile requirements engineering: A survey in polish software industry. In Wrycza, S. and Maslankowski, J., editors, *Information Systems: Research, Development, Applications, Education - 12th SIGSAND/PLAIS EuroSymposium 2019, Gdansk, Poland, September 19, 2019, Proceedings*, volume 359 of *Lecture Notes in Business Information Processing*, pages 147–158. Springer.

Jarzebowicz, A. and Weichbroth, P. (2021). A systematic literature review on implementing non-functional requirements in agile software development: Issues and facilitating practices. In Przybylek, A., Miler, J., Poth, A., and Riel, A., editors, *Lean and Agile Software Development - 5th International Conference, LASD 2021, Virtual Event, January 23, 2021, Proceedings*, volume 408 of *Lecture Notes in Business Information Processing*, pages 91–110. Springer.

Lenarduzzi, V. and Fucci, D. (2019). Towards a holistic definition of requirements debt. In *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2019, Porto de Galinhas, Recife, Brazil, September 19-20, 2019*, pages 1–5. IEEE.

Linaker, J., Sulaman, S. M., Höst, M., and de Mello, R. M. (2015). Guidelines for conducting surveys in software engineering v. 1.1. *Lund University*, 50.

Marek, K., Winska, E., and Dabrowski, W. (2021). The state of agile software development teams during the covid-19 pandemic. In *Lean and Agile Software Development - 5th International Conference, LASD 2021, Virtual Event, January 23, 2021, Proceedings*, volume 408 of *Lecture Notes in Business Information Processing*, pages 24–39, https://doi.org/10.1007/978-3-030-67084-9_2. Springer.

Mendes, L., Cerdeiral, C., and Santos, G. (2019). Documentation technical debt: A qualitative study in a software development organization. In do Carmo Machado, I., Souza, R., Maciel, R. S. P., and Sant'Anna, C., editors, *Proceedings of the XXXIII Brazilian Symposium on Software Engineering, SBES 2019, Salvador, Brazil, September 23-27, 2019*, pages 447–451. ACM.

Perera, J., Tempero, E. D., Tu, Y., and Blincoe, K. (2023). Quantifying requirements technical debt: A systematic mapping study and a conceptual model. In Schneider, K., Dalpiaz, F., and Horkoff, J., editors, *31st IEEE International Requirements Engineering Conference, RE 2023, Hannover, Germany, September 4-8, 2023*, pages 123–133. IEEE.

Robiolo, G., Scott, E., Matalonga, S., and Felderer, M. (2019). Technical debt and waste in non-functional requirements documentation: An exploratory study. In Franch, X., Männistö, T., and Martínez-Fernández, S., editors, *Product-Focused Software Process Improvement - 20th International Conference, PROFES 2019, Barcelona, Spain, November 27-29, 2019, Proceedings*, volume 11915 of *Lecture Notes in Computer Science*, pages 220–235. Springer.

Sajjade, Z.-S. Z. (2020). Agile process model for software development. *Journal of Advancement in Software Engineering and Testing*, 3 (03).

Saravanan, T., Jha, S., Sabharwal, G., and Narayan, S. (2020). Comparative analysis of software life cycle models. In *2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, volume 2, pages 906–909.

Seaman, C. B. (1999). Qualitative methods in empirical studies of software engineering. *IEEE Trans. Software Eng.*, 25(4):557–572.

Theunissen, T., van Heesch, U., and Avgeriou, P. (2022). A mapping study on documentation in continuous software development. *Inf. Softw. Technol.*, 142:106733.