# A Systematic Literature Review on Implementing Non-functional Requirements in Agile Software Development: Issues and Facilitating Practices

Aleksander Jarzębowicz 0000-0003-3181-4210 ✉, and
Paweł Weichbroth 0000-0002-1645-0941

Gdańsk University of Technology,
Faculty of Electronics, Telecommunications and Informatics,
Department of Software Engineering,
11/12 Gabriela Narutowicza Street,
80-233 Gdańsk, Poland
{aleksander.jarzebowicz,pawel.weichbroth}@pg.edu.pl
http://www.pg.edu.pl

**Abstract.** Agile Software Development methods have become a widespread approach used by the software industry. Non-functional requirements (NFRs) are often reported to be a problematic issue for such methods. We aimed to identify (within the context of Agile projects): (1) the issues (challenges and problems) reported as affecting the implementation of NFRs; and (2) practices that facilitate the successful implementation of NFRs. We conducted a systematic literature review and processed its results to obtain a comprehensive summary. We were able to present two lists, dedicated to issues and practices, respectively. Most items from both lists, but not all, are related to the requirements engineering area. We found out that the issues reported are mostly related to the common themes of: NFR documentation techniques, NFR traceability, elicitation and communication activities. The facilitating practices mostly cover similar topics and the recommendation is to start focusing on NFRs early in the project.

**Keywords:** Non-functional Requirements; Quality Requirements; Agile Software Development; Agile Requirements Engineering; Systematic Literature Review

## 1 Introduction

Agile Software Development (ASD) is an iterative approach to delivering software products. The term "agility" implies adaptability [1], flexibility [2], and close collaboration with the customer [3]. An Agile approach assumes sensible values such as trust [4], responsibility [5] and loyalty [6]. Around half of organizations have now been applying Agile practices for over three years to adopt change and transformation management [7]. Moreover, the results from a survey conducted in 2018 among software industry practitioners show that 97 percent of respondents declared using Agile methods [8]. In fact, the benefits of adopting Agile practices have been reported in many studies [9,10,11], indicating an increase of team productivity, motivation and discipline, as well as overall software quality, just to name a few.

Indeed, software quality is an important aspect to be considered during the software lifecycle [12,13], usually defined in terms of high-level attributes [14]. Alternatively, one can impose additional constraints on the behavior of the system. In other words, the required properties (attributes and constraints) are specified as non-functional requirements (hereafter, NFRs), in addition to functional requirements (FRs). Since the beginning of software development as a job role, NFRs have been recognized as critical factors that affect the acceptance and use of the products by the users [15].

In fact, to mitigate the risk of users' dissatisfaction by misunderstanding or disregarding their expectations and needs, active user involvement is imperative in ASD [16,17,18]. However, one question arises naturally: Does this user engagement bring other risks, and does the development team need to find a balance between risk and benefits?

Undeniably, the search for the answer to this question has been the subject of vast research [19,20,21], since the introduction of the Agile Manifesto [22]. Nevertheless, few studies provide an evidence-based review and analysis on the subject of implementing NFRs, in particular regarding the issues that could arise with the advance of ASD, as well as the practices that have been documented as successful facilitators.

The values and principles followed in ASD also result in practices different than those used in more traditional software development methods. It includes requirements engineering practices [23], which e.g. assume continuous close cooperation with the customer [24], put more emphasis on face-to-face communication [25], and use less formal techniques like collaborative games [26].

Both researchers and practitioners have repeatedly noted the challenges in Agile requirements engineering. For example, the results from a Delphi study [27], performed in 2017 in a group of 26 experts, show that one of the recognized challenges is to "establish non-functional requirements", which has been reported by prior other studies [28,29,25]. The comprehensive know-how with regard to the more detailed challenges and relevant counteractions is not available though. The only available secondary study focusing on NFRs in ASD at the time we started our research was the SLR by Alsaquaf et al. [30]. That SLR was considered by its authors as a starting point for further empirical studies and several primary studies were published since then. To systematize the current state of the art, in this paper, we put forward these two following research questions (RQs):

1. What issues affect the identification and implementation of non-functional requirements in ASD?
2. What practices facilitate the successful identification and implementation of non-functional requirements in ASD?

Therefore, the goal of this study is to review and analyze the existing studies and their outcomes and to summarize the documented issues and applied practices, in the extent of NFR identification and implementation, within the ASD context. To provide evidence-based and state-of-the-art answers to the above questions we conducted a systematic literature review (SLR).

By design, the results of this study are complementary to the existing body of knowledge by providing the following contributions to the software engineering discipline: the collections of (*i*) the current issues (challenges and problems), and (*ii*) the explicit practices that, respectively, affect and facilitate the identification and implementation of NFRs within ASD. Moreover, the findings in this paper entail useful implications for researchers and practitioners alike. In this context, while the former group might be interested in investigating the impact of particular issues on the success (failure) of ASD projects, the latter group might be willing to mitigate those issues by adopting the practices in the scope and content due to the current needs and priorities.

The remainder of this paper is laid out as follows. Section 2 describes the rationale behind implementing NFRs. Section 3 provides the description of the research methodology, applied to conduct the systematic literature review. The results are given in Section 4, followed by their discussion in Section 5. Finally, the paper is concluded in Section 6.

## 2 Rationale behind implementing NFRs

Generally speaking, non-functional requirements (NFRs), also known as quality requirements, define the users' expectations and needs regarding a software product, as well as their particular notions of its qualities. According to Svensson *et al.* [31], the most important quality attributes in industrial practice relate to usability, performance, reliability, stability, safety, security/integrity, compliance, maintainability, reusability and interoperability. Unmistakably, NFRs have great importance in software product development [31,32,33].

Besides this, NFRs can also impose global constraints on a software product [34], arising from all of its parts as well as from interdependencies between them [35]. In other words, NFRs put constraints on how the product's functions must work [36]. Overlooking or even neglecting information related to quality facets negatively affects the final product. Ironically, although it might be surprisingly different from common sense, NFR-related errors are still claimed to be the most difficult to correct, and the most expensive [37]. It is a major risk, especially considering that in recent years software defects have become the dominant cause of user outage [38].

Undeniably, both researchers and practitioners from ASD communities have seen the need to capture, document and prioritise NFRs [39]. For instance, Microsoft, the largest software and programming company worldwide [40], recommends capturing functional and non-functional requirements alike, since the former indicate whether the application does the right thing, while the latter determine whether the application does those things well [41]. Oracle, the second largest software corporation, argues that "the key to successful software development is that all stakeholders develop a clear and uniform understanding of application requirements" [42].

Furthermore, we also acknowledge the importance of NFRs as the major external quality facets of the software products from the user's perspective [43]. The questions addressed in this study are narrowed to ASD, which assumes having the user(s) actively involved. If one compares Agile with traditional approaches, this involvement is not limited to the early stages of the development process. On the contrary, Agile development principles encourage active user involvement, being generally considered to contributing to user satisfaction [44,45] and project success [46].

## 3 Methodology

We designed and executed the systematic literature review following the guidelines for SLR studies in software engineering elaborated by Kitchenham and Charters [47]. The definition of the search query and query execution in Scopus (phase 1 of SLR process) are shared with our other study aimed at identification of particular NFR-related requirements engineering techniques [48]. The inclusion/exclusion criteria were however defined with respect to this study's aim and subsequent phases of the SLR process were conducted separately in each of two studies.

We chose to rely on a single publication database (Elsevier Scopus). Scopus was selected because it indexes a large number of journals and conferences [49] and enables a single search query to access items from a broad variety of publishers [50]. It is worth noting here that in several other SLR studies similar to ours (e.g. [51,30]) similar strategies were applied, in particular exclusively relying on the Scopus database.

### 3.1 Inclusion and exclusion criteria

The papers were eligible based on the five following inclusion criteria:

– peer-reviewed papers (I1);
– papers in English (I2);
– papers published since 2008 (I3);
– papers related to the software engineering domain (I4);
– papers covering Agile development and NFRs (I5).

The papers were screened prior to acceptance and were further rejected if they had any of the following exclusion criteria:

– papers not providing any information about NFR issues or practices in ASD (E1);
– papers not available for download, despite extensive search (E2);
– papers reporting the same results covered by another source included in SLR - in such cases the latest paper was included (E3);

– papers dedicated to a very specific subarea of NFRs (e.g. with proposals of advanced methods of establishing security requirements) (E4).

We focused on papers published since 2008 to include all works published in the last 12 years before the conduction of the SLR. We also decided to include papers dedicated to a specific project context (e.g. large-scale distributed development), but to exclude papers with very narrow scope (E4) e.g. with advanced dedicated analysis methods suggested as facilitating practices for security requirements, which are hard to consider as an issue or practice regarding the whole category of NFRs.

## 3.2 Search query definition

As we had performed some initial searches before planning the SLR, we were aware that sources dedicated to this topic of interest are rather scarce. This led us to the decision to cast a wider net and try to identify all sources focusing on NFRs in Agile, thus we used more generic keywords instead of those exactly matching our RQs (e.g. "challenges" or "practices").

The following search string was used:

*TITLE-ABS-KEY ((agile OR scrum OR lean OR xp OR kanban) AND (nfr OR "non-functional requirements" OR "quality requirements")) AND PUBYEAR > 2007 AND (LIMIT-TO(DOCTYPE, "cp") OR LIMIT-TO (DOCTYPE, "ar") OR LIMIT-TO (DOCTYPE, "ch")) AND (LIMIT-TO (SUBJAREA, "COMP") OR LIMIT-TO (SUBJAREA, "ENGI") OR LIMIT-TO (SUBJAREA , "MATH") OR LIMIT-TO (SUBJAREA, "BUSI") OR LIMIT-TO (SUBJAREA, "DECI")) AND (LIMIT-TO (LANGUAGE, "English"))*

The string includes various methods that could possibly be mentioned in the title, keywords etc. instead of the generic "Agile" term. We also provided alternative terms commonly used to denote an NFR. The types of documents mentioned in the search string match peer-reviewed papers. The specification of subject areas resulted from our knowledge, in particular on how some sources (especially the series that include conference proceedings as its volumes) are classified and indexed by Scopus. The search in titles, abstracts and keywords was chosen as the most comprehensive option available (Scopus does not enable searching the contents of full texts).

## 3.3 Search strategy

We defined a process that comprised 3 main phases:

1. Execution of the search query.
2. Manual review of titles, keywords and abstracts of the papers retrieved from the search to exclude those not related to the topic of NFR in an Agile context.
3. Manual review of each remaining paper's full text in order to decide whether to finally include it or not. Identification of information pieces relevant to our RQs and assigning codes to them.

### 3.4 Search execution

The results of the 3 phases defined in the previous section were as follows:

**Phase 1**: The search was executed on November 28th 2019. Despite including several alternative keywords in the search string, the search returned only 159 papers. This confirmed our initial suspicions that the topic of "NFR in Agile" is not widely addressed in the scientific papers, at least those indexed by Scopus.

**Phase 2**: The results retrieved by the Scopus search engine (that include title, keywords and the abstract of each paper found) were manually reviewed. It allowed us to verify the findings against I5 criterion more precisely than in the case of relying on an automated search and to reject papers that reported nothing on NFRs (for example, several papers referring to "quality requirements" turned out to interpret this term as "well-documented/valid requirements" instead of "requirements regarding system quality"). As a result, 71 papers were retained at the end of this phase.

**Phase 3**: In this phase, the papers were reviewed and checked against exclusion criteria E1–E4. Finally, 44 papers were qualified to extract information. Moreover, we evaluated the papers with regard to *(i)* the use of appropriate and rigorous research methods, *(ii)* clarity and coherence of the research findings, and *(iii)* providing a validation of the proposed approach. During the review, apart from just deciding on the paper's final classification, the fragments relevant to the RQs were identified and provided with codes to summarize the findings. Next, the codes were reviewed to identify similarities, and related codes were grouped into the more generic ones presented in the Results section.

## 4 Results

The final results of the SLR are presented in Tables 1 and 2. For each issue reported and facilitating practice suggested, a list of papers mentioning it is provided ("Sources" column). We also explicitly distinguish issues/practices related to requirements engineering activities ("RE" column) from those that should rather be associated with e.g. testing, architectural design or project management. Both tables are sorted starting from with the items quoted by the most sources. The elaboration of results with respect to the answers they provide to RQs is provided in 4.1 and 4.2.

### 4.1 What issues affect the identification and implementation of non-functional requirements in ASD?

The most frequently reported issue concerns neglecting NFRs (I1) i.e. the situation in which developers and/or stakeholders focus on the system's functionality and do not identify NFRs in a sufficient manner, often postponing such task to a later stage of the project. Unfortunately, it often results in significant rework effort, as NFRs are not necessarily simple additions and are likely to substantially affect the system architecture. It should be stated that this issue, while mentioned by many papers, is not always based on experience or empirical findings

**Table 1.** Problems and challenges affecting development of NFRs in ASD

| ID | Problem/challenge | RE | Sources |
|---|---|---|---|
| I1 | Neglecting NFRs (usually while focusing on FR) | + | [52], [53], [54], [55], [56], [57], [58], [59], [60], [61], [62], [63], [64], [65], [66], [67], [68] |
| I2 | Misunderstandings regarding NFRs specified as User Stories (or similar simplified representation) | + | [62], [69], [70], [71], [72], [73], [74] |
| I3 | Lack of recognition of NFRs by stakeholders | + | [52], [55], [60], [67], [75], [76] |
| I4 | Difficulties with documenting the NFRs in a way that exposes their dependencies | + | [52], [55], [63], [65], [68] |
| I5 | Lack of traceability mechanisms of NFRs | + | [58], [68], [77], [78] |
| I6 | Inadequate NFR test specification to verify their implementation | + | [52], [57], [75], [79] |
| I7 | Insufficient knowledge/competencies (advanced NFR concepts) in the project team | + | [52], [55], [76] |
| I8 | Overlooking sources of NFRs (stakeholders) | + | [52], [55], [75] |
| I9 | Unclear conceptual definition of NFRs (how to document them) | + | [52], [67], [80] |
| I10 | NFRs are affected by changes in FRs | + | [68], [81], [82] |
| I11 | Sporadic adherence to quality guidelines by Agile teams | | [52], [55], [75], |
| I12 | Suboptimal inter-team organization (around components, scenarios or functional teams e.g. testers) leading to poor implementation of NFRs | | [52], [55], [75] |
| I13 | Late detection of NFRs' infeasibility | + | [52], [75] |
| I14 | Ambiguous NFRs communication process | + | [52], [68] |
| I15 | NFRs stored outside of backlog, in an external document and thus not always addressed | + | [39], [72] |
| I16 | Hidden assumptions regarding NFRs implementation in inter-team collaboration (in a large scale project) | | [52], [75] |
| I17 | Misunderstanding the architecture drivers (priorities of NFRs) between teams | | [52], [75] |
| I18 | Lengthy NFR acceptance checklist (e.g. DoD) | + | [52] |
| I19 | Agile process does not include a feedback loop regarding NFRs | + | [55] |
| I20 | Unmanaged architecture changes | | [52] |
| I21 | Lack of cost-effective real integration test | | [52] |
| I22 | Adopting legacy architectural decisions complicate the implementation of NFRs of the new system | | [75] |
| I23 | Moving to Agile with a waterfall mind-set | | [75] |
| I24 | Difficult testing to verify NFRs as it requires associated FR to be already implemented | | [83] |

**Table 2.** Practices facilitating implementing NFRs in ASD

| ID | Practice | RE | Sources |
|---|---|---|---|
| P1 | Use modified or additional specification techniques for NFRs (including those adopted from plan-driven approaches) | + | [60], [61], [62], [63], [65], [66], [68], [71], [72], [76] |
| P2 | Maintain traceability between FRs and NFRs | + | [58], [63], [65], [77], [78], [81], [82], [84] |
| P3 | Start focusing on NFRs early in the project | + | [61], [64], [66], [80], [85], [86], [87] |
| P4 | Document NFRs using standard ARE specification techniques (e.g. US, DoD, AC) | + | [55], [56], [72], [74], [85], [88], [89] |
| P5 | Use automated monitoring tools, e.g. SONAR, to monitor quality of software under development | | [52], [53], [54], [55], [75], [88], [90] |
| P6 | Involve NFR specialists (e.g. a team of specialists that ensures proper implementation of NFRs or an NFR stakeholder) | + | [52], [55], [57], [68], [75] |
| P7 | Involve multiple roles and viewpoints to elicit and/or review NFRs | + | [62], [66], [68], [79], [91] |
| P8 | Educate and raise awareness about the importance of (particular) NFRs | + | [55], [80], [91] |
| P9 | Use patterns/templates catalogue to specify NFRs | + | [53], [62], [92] |
| P10 | Establish preparation team (responsible for NFRs, architecture and distribution of backlog items to development teams) | | [52], [75], [88] |
| P11 | Use abstract but easy to grasp terms by user and/or alternatives to elicit NFRs from stakeholders | + | [79], [85] |
| P12 | Use multiple product backlogs to include requirements of different viewpoints | + | [52], [75] |
| P13 | Use supporting systems providing NFR recommendations | + | [54], [93] |
| P14 | Instead of specifying NFRs as epics, user stories etc., use a similar but distinct structure dedicated to NFRs | + | [70], [94] |
| P15 | Reserve part of the sprint for important NFRs | | [52], [75] |
| P16 | Introduce Sprint allocation based on multiple Product Backlogs (e.g. 1 – FRs, 2 – NFRs, 3 – CI/CD requirements) | | [52], [75] |
| P17 | Establish components teams (each team solely responsible for a given component and its quality) | | [52], [75] |
| P18 | Introduce innovation and planning iteration (IP, term from SAFe) to resolve technical debts related to NFRs | | [52], [75] |
| P19 | Conduct NFR-oriented code reviews | | [55], [88] |
| P20 | Explain to the stakeholders the consequences of overspecified NFRs | + | [85] |
| P21 | Maintain an assumption wiki-page | + | [52] |
| P22 | Use CI environment to utilize automated NFR testing | | [83] |
| P23 | Establish an independent team to test NFRs' implementation | | [80] |

but sometimes treated as "common knowledge" or quoted from other referenced papers. On the other hand, the frequent occurrence of such issue is confirmed by more general studies not dedicated to NFRs but listing the general problems and challenges related to ASD and/or requirements engineering (examples are given in Section 5.1).

A number of issues can be attributed to limitations of the simplified requirements documentation techniques (e.g. user stories, story cards) commonly used in Agile methods. In application to NFRs, such techniques can turn out to be insufficient to express NFRs in an unambiguous way (I2). Another reported shortcoming of such techniques is the difficulty in representing the dependencies between a given NFR and other related requirements (I4). An open issue of how to represent NFRs is also reported as a doubt explicitly expressed by Agile teams (I9). While in some projects, a workaround in the form of a separate document dedicated to NFRs is used, it can also cause difficulties as the project team can focus on the FRs typically stored in the product backlog and do not sufficiently rely on external documents including that for NFRs (I15).

NFRs are more difficult to capture and cause problems both for stakeholders and for the project team. The stakeholders may even not recognize their needs that have to be captured as NFRs (I3). The project team may in turn lack the knowledge and competencies necessary to identify and implement some NFRs, especially when advanced concepts related to e.g. security or performance need to be used (I7).

The elicitation and communication of NFRs is another category of issues. Requirements elicitation can fail to involve all of the relevant stakeholders (I8) and result in NFRs that do not reflect all viewpoints or even omit some important requirements. NFRs are also quite hard to express, thus their communication (both from the stakeholder to the project team, and between team members) can be prone to errors (I14). Moreover, in large scale development projects, involving multiple teams, additional communication problems are likely to arise (I12, I16, I17). Several drawbacks in handling NFRs can result in a situation of late detection of NFR infeasibility (I13), especially considering the lack of a feedback loop regarding NFRs (I19).

Several issues related to NFR traceability and verifiability are reported as well. A lack of NFR traceability mechanisms is claimed in general (I5), but also several more specific issues are described. Traceability of NFRs is even more important as NFRs are frequently affected by changes in FRs (I10). It is difficult to develop test specifications associated with NFRs, which are intended to verify their implementation (I6). Moreover the execution of such tests requires the associated FRs to be already implemented (I24). The cost-effectiveness of some tests is also disputed (I21). The manual verification of DoD can be cumbersome as well, especially in case of a lengthy checklist (I18).

The remaining issues are either related to project team members' attitudes (I11, I23) or architectural design activities (I20, I22).

## 4.2 What practices facilitate the successful identification and implementation of non-functional requirements in ASD?

A number of practices dedicated to the documentation of NFRs can be found in the literature, even though some of them seem to be mutually contradictory. The issue of the insufficiency of the popular Agile requirements documentation techniques can be addressed by utilizing modified or additional specification techniques (P1). Such techniques are to be applied to NFRs only (while FRs are still recorded as e.g. user stories). Some proposals include techniques adopted from plan-driven approaches.

Alternatively, other sources recommend making sure that NFRs are documented together with FRs, using the same, typical representations, e.g. user stories, Definition of Done, Acceptance Criteria (P4). There is also a kind of intermediate solution suggested – instead of specifying NFRs as epics, user stories etc. and mixing them with FRs, a similar but distinct structure dedicated to NFRs can be used (P14). Also, assumptions related to the implementation of NFRs are worth documenting using, e.g. a wiki-page (P21).

Focusing on NFRs in an early phase of the project (P3) is a suggestion that can possibly minimize the rework caused by omitted NFRs. Multiple roles and viewpoints should be involved to elicit and/or review NFRs (P7). A number of more detailed practices facilitating requirements elicitation from stakeholders can also be found, e.g. using proper terms (P11) or explaining the consequences of NFRs expressed by stakeholders (P20), especially in the case of "over-specification". Another good idea is to educate and raise awareness about importance of NFRs (P8) - in general or with respect to some categories of NFRs which are not sufficiently recognized. Both stakeholders and project team members can be educated in such a manner.

As NFRs can be difficult to identify and even harder to implement, it is possible to strengthen the competencies of the project team by involving NFR specialists (P6). For example, a security expert can enable the elicitation of relevant security requirements and later verify their implementation. As the stakeholders may not be able to identify the NFRs themselves, external resources such as catalogues of NFR patterns/templates (P9) or dedicated supporting systems providing recommendations (P13) can be used as additional sources of NFRs.

Maintenance of traceability between FRs and NFRs is a frequently recommended practice (P2) that enables proper requirements management activities, including configuration and change management. Various solutions, including tool support, are proposed to ensure traceability maintenance. There are also other practices including the use of automated tools, in particular: tools to monitor the quality of the software under development, including the aspects expressed in NFRs (P5) and CI environments facilitating automated NFR testing (P22). Apart from tool-based testing, NFR-oriented code reviews (P19) and external tests conducted by an independent team (P23) can be practiced.

To minimize the risk of neglecting NFRs, several actions in the software project organization can be undertaken. They can concern the organization of the project team(s) (P10, P17); the development process - dedicating an iteration

(P18) or a part of each iteration (P15) to the implementation of NFRs; or using multiple requirements registers, which make NFRs (or specific NFR categories) more visible (P12, P16).

## 5 Discussion

Non-functional requirements (NFRs) have become an important research area, mainly due to the abundance of project failures caused by neglecting quality attributes related to user values. While there is no consensus on the reasons for this, Maxim and Kessentini point out that NFRs "are not easy for stakeholders to articulate, but they know that the software will not be usable without some of these non-functional characteristics" [95]. Similarly, the four most frequent issues identified in our study concern neglecting (I1), a lack of (I3), or misunderstanding (I2) NFRs. Further to this, even when one manages to write them down, difficulties are encountered along the way while attempting to document particular qualities and their dependencies (I4).

Notwithstanding these observations, one question arises: Why have been NFRs disregarded? The first reason is the insufficient knowledge and low competence of the employees (I7), particularly in terms of their analytical skills and professional experience, reflected by their inability to perform a required task at a targeted level of proficiency. Moreover, their incompetence is also demonstrated by overlooking sources of NFRs (I8), vague definitions and obscure descriptions (I9). These burdens might be considered to be a result of ambiguous communication (I14).

The remaining aspects of the discussion, namely: comparison to works by other authors (5.1), study limitations (5.2) and implications (5.3) are respectively given below.

### 5.1 Comparison with Related Works

There are only a few sources dedicated to identification of NFR-related issues and/or practices in ASD. Alsaquaf *et al.* consider NFRs in a more specific context of Agile Large-Scale Distributed projects. These authors conducted an SLR study to summarize the challenges and practices mentioned in the literature [30] and further investigated additional challenges through a series of interviews with industry practitioners [52]. Behutiye *et al.* [86] consider NFRs in the generic context of ASD. They used situational method engineering to analyze NFR management practices and interviews to identify challenges and practices of NFR documentation [72].

The SLR by Alsaquaf et al. [30] uncovered 12 NFR-related issues and 13 practices used as solutions. We were able to identify more items of both categories. The results of both above-mentioned primary studies ([52] and [86]) were retrieved in our SLR study and included in its results, together with a wider set of issues/practices from other sources. It is worth to notice that 16 out of 24 issues found in our SLR were identified in a series of interviews dedicated to NFRs challenges in Agile Large-Scale Distributed (ALSD) development [52], which indicates that such challenges are not limited to ALSD, but apply to ASD projects in general. Such findings are also corroborated by reports in other sources we were able to retrieve in our SLR study.

A larger number of research studies on issues and/or practices is available, however their scope is wider and concerns e.g. the challenges of Agile requirements engineering in general. An SLR study by Inayat *et al.* summarizes existing requirements engineering challenges, Agile practices that address such challenges as well as additional challenges related to Agile practices [28]. Heikilla *et al.* identified, through a mapping study, Agile requirements engineering benefits as well as its problematic areas [96]. Medeiros *et al.* conducted a mapping study focused on Agile requirements engineering practices and techniques dedicated to requirements elicitation and documentation [51]. Schon *et al.* focuses on introducing the user's perspective to ASD and joint application of ASD and User-Centered Design (UCD). They conducted an SLR study to summarize the related practices and identify essential aspects of Agile requirements engineering in the UCD context [27].

Apart from [30], [72] and [86], none these studies considers NFRs specifically and as such, they do not address detailed practices nor issues. Some of them enumerate single NFR-related issues, mostly mentioning the risk of neglecting NFRs in ASD projects [28,27] and to the typical Agile documentation insufficient to successfully capture NFRs [28,96]. It is worth noting here that usually, the issues/practices mentioned in such papers are not explicitly assigned to NFRs but to requirements in general, also including FRs, constraints, business goals etc. With respect to this, our study has a much more narrow scope, but within this scope provides much more detailed findings.

Finally, there are studies that focus on a single issue and/or practice, as they propose a new method, technique or practice to address some known issue, e.g. a lack of NFR traceability or difficulty in documenting NFRs of some kind. In our study, we made an attempt to include such contributions into the summary lists that provide the reader with an overview of the state of the art on NFRs in the ASD topic.

## 5.2 Limitations

While we try to minimize risks by following the guidelines by Kitchenham and Charters [47], we are aware that our study has several possible limitations remaining. A limitation of any SLR study is the potential bias in selecting the sources. This starts with the decisions regarding the publication databases to be searched and the search string to be used for this purpose.

Our study conducted the search in Elsevier Scopus only. Scopus is known to enable a single search query to access items from a large number of journals and conferences [49,50]. However, it is possible that it misses some sources that could in turn be found in other databases. Moreover, scientific databases do not include so called "grey literature" which can potentially include industry experiences in non-scientific publications, such as reports and web articles.

We paid attention to the construction of the search string and included several synonyms and alternative terms to increase our chances of finding all of the relevant sources. The SLR is, however, strongly dependent on vocabulary and we cannot rule out that some authors used less common expressions which would lead to failure to find their papers. Moreover, as our search string implied that NFRs must be explicitly mentioned in the paper's title, abstract or keywords, more generic papers (e.g. dedicated to ASD challenges or practices), which just mention an issue or practice related to NFRs among many others, could be missed.

The extraction of the data from the sources is also a task prone to bias, as it is done by humans, who interpret the contents of the sources. Following SLR guidelines minimizes such a threat, but cannot entirely eliminate it.

### 5.3 Implications for Research and Practice

This study has several implications for both researchers and practitioners. For researchers, the relatively small number of sources retrieved for the SLR indicates that there is a need for more studies regarding NFRs in ASD projects. Moreover, the issues and practices listed in the findings of our study can be considered by researchers as potential subjects of dedicated empirical studies, further exploring, e.g. the root causes of reported issues or the effectiveness of the facilitating practices.

Industrial practitioners can use our findings to anticipate issues in projects they participate in, and to select facilitating practices to be applied in their projects. Our study can also be used to raise awareness on NFRs in ASD, the related issues and practices as well as the overall importance of such a topic – which still tends to be neglected or lack sufficient attention.

## 6 Conclusions

In this paper, we explored the topic of the implementation of non-functional requirements (NFRs) in Agile Software Development (ASD), focusing on the issues and facilitating practices, gathered from the existing body of literature. The main motivation underpinning this study was to investigate the state-of-the-art in implementing NFRs within ASD projects, through a systematic literature review (SLR), in order to identify what issues have been documented, as well as by what means one can facilitate the implementation of NFRs. This research was driven by the guidelines elaborated by Kitchenham and Charters, a well-known and widely-applied research framework in the field of software engineering.

The number of industrial projects that deliver different and specific lessons regarding NFRs, makes comparison of the studies a time-consuming and intricate task, since they do not frequently deal with the same focus or goals. Nevertheless, we were able to collect and unambiguously classify a bulk of issues and practices, extracted from peer-reviewed scientific sources. Obviously, our report neither exhausts the topic nor provides external validity. However, we believe that the obtained results may serve as a useful reference repository to be used by both experienced and novice researchers, as well as senior and junior practitioners.

Moreover, a number of open issues and related research directions were identified through this study, which can be considered as an input for future work. A clear lack of consensus on which requirements documentation techniques should be used in order to specify NFRs. Some sources suggest using the techniques available in Agile methods e.g. User Stories (possibly with some adjustments), while others recommend introducing additional techniques (including those used in more traditional, plan-driven approaches). Such selection of the suitable techniques may be a context-dependent issue and requires further investigation. Other areas we identify as potential future work directions are: the relationship between requirements engineering and testing areas (test specifications to verify the implementation of NFRs, which can also be considered as part of NFRs documentation); and the facilitation of an unambiguous NFRs communication process.

We plan to address the latter topic by designing and testing an ontology-based approach, using Controlled Natural Language (CNL) [97] and the Fluent Editor [98], simulating and modelling requirements specification. Our observations gathered during professional work indicate that there is still a need to implement suitable methods and tools to support communication among different groups of stakeholders and development teams.

## References

1. S. Amjad, N. Ahmad, T. Saba, A. Anjum, U. Manzoor, M. A. Balubaid, and S. U. R. Malik, "Calculating completeness of agile scope in scaled agile development," *IEEE Access*, vol. 6, pp. 5822–5847, 2017.
2. M. Adnan and M. Afzal, "Ontology based multiagent effort estimation system for Scrum agile method," *IEEE Access*, vol. 5, pp. 25993–26005, 2017.
3. P. E. Strandberg, E. P. Enoiu, W. Afzal, D. Sundmark, and R. Feldt, "Information flow in software testing–an interview study with embedded software engineering practitioners," *IEEE Access*, vol. 7, pp. 46434–46453, 2019.
4. G. Tjørnehøj, M. Fransgård, and S. Skalkam, "Trust in agile teams in distributed software development," in *Information System Research Seminar in Scandinavia 2012 Information Systems Research Seminar in Scandinavia*, pp. 1–15, 2012.
5. R. C. Martin, *Agile software development: principles, patterns, and practices*. Prentice Hall, 2002.
6. S. Roy, A. Raju, and S. Mandal, "An empirical investigation on e-retailer agility, customer satisfaction, commitment and loyalty," *Business: Theory and Practice*, vol. 18, pp. 97–108, 2017.
7. Consultancy.eu, "Half of companies applying agile methodologies & practices," 2020 (accessed: 10.11.2020). `https://www.consultancy.eu/news/4153/half-of-companies-applying-agile-methodologies-practices`.
8. Version One, "13th annual state of agile report," 2019 (accessed: 10.11.2020). `https://stateofagile.com/`.
9. E. Bjarnason, K. Wnuk, and B. Regnell, "A case study on benefits and side-effects of agile practices in large-scale requirements engineering," in *1st Workshop on Agile Requirements Engineering*, pp. 1–5, 2011.

10. K. Kaur, A. Jajoo, *et al.*, "Applying agile methodologies in industry projects: Benefits and challenges," in *2015 International Conference on Computing Communication Control and Automation*, pp. 832–836, IEEE, 2015.
11. P. Diebold and U. Mayer, "On the usage and benefits of agile methods & practices," in *International Conference on Agile Software Development*, pp. 243–250, Springer, Cham, 2017.
12. L. Guzmán, M. Oriol, P. Rodríguez, X. Franch, A. Jedlitschka, and M. Oivo, "How can quality awareness support rapid software development? – a research preview," in *International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ)*, pp. 167–173, Springer, 2017.
13. D. S. Guamán, J. M. Del Alamo, and J. C. Caiza, "A systematic mapping study on software quality control techniques for assessing privacy in information systems," *IEEE Access*, vol. 8, pp. 74808–74833, 2020.
14. A. Jarzębowicz and K. Połocka, "Selecting requirements documentation techniques for software projects: a survey study," in *2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pp. 1189–1198, IEEE, 2017.
15. A. J. Ryan, "An approach to quantitative non-functional requirements in software development," in *34th Annual Government Electronics and Information Association Conference*, pp. 13–20, 2000.
16. K. Kautz, "Customer and user involvement in agile software development," in *International Conference on Agile Processes and Extreme Programming in Software Engineering*, pp. 168–173, Springer, 2009.
17. A. Jarzębowicz and P. Marciniak, "A survey on identifying and addressing business analysis problems," *Foundations of Computing and Decision Sciences*, vol. 42, no. 4, pp. 315–337, 2017.
18. A. Jarzębowicz and W. Ślesiński, "What is troubling IT analysts? A survey report from Poland on requirements-related problems," in *Engineering Software Systems: Research and Praxis*, pp. 3–19, Springer International Publishing, 2019.
19. S. Mohammadi, B. Nikkhahan, and S. Sohrabi, "Challenges of user involvement in Extreme Programming projects," *International Journal of Software Engineering and Its Applications*, vol. 3, no. 1, pp. 19–32, 2009.
20. M. Bano and D. Zowghi, "A systematic review on the relationship between user involvement and system success," *Information and Software Technology*, vol. 58, pp. 148–169, 2015.
21. K. Schmitz, R. Mahapatra, and S. Nerur, "User engagement in the era of hybrid agile methodology," *IEEE Software*, vol. 36, no. 4, pp. 32–40, 2018.
22. K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, *et al.*, "The agile manifesto," 2001 (accessed: 10.11.2020). `https://agilemanifesto.org/`.
23. D. Leffingwell, *Agile software requirements: lean requirements practices for teams, programs, and the enterprise.* Addison-Wesley Professional, 2010.
24. J. Miler and P. Gaida, "On the agile mindset of an effective team–an industrial opinion survey," in *2019 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pp. 841–849, IEEE, 2019.
25. B. Ramesh, L. Cao, and R. Baskerville, "Agile requirements engineering practices and challenges: an empirical study," *Information Systems Journal*, vol. 20, no. 5, pp. 449–480, 2010.
26. M. Zakrzewski, D. Kotecka, Y. Y. Ng, and A. Przybyłek, "Adopting collaborative games into agile software development," in *International Conference on Evaluation of Novel Approaches to Software Engineering*, pp. 119–136, Springer, 2018.

27. E.-M. Schön, D. Winter, M. J. Escalona, and J. Thomaschewski, "Key challenges in agile requirements engineering," in *International Conference on Agile Software Development*, pp. 37–51, Springer, Cham, 2017.
28. I. Inayat, S. S. Salim, S. Marczak, M. Daneva, and S. Shamshirband, "A systematic literature review on agile requirements engineering practices and challenges," *Computers in human behavior*, vol. 51, pp. 915–929, 2015.
29. H. F. Soares, N. S. Alves, T. S. Mendes, M. Mendonça, and R. O. Spínola, "Investigating the link between user stories and documentation debt on software projects," in *2015 12th International Conference on Information Technology-New Generations*, pp. 385–390, IEEE, 2015.
30. W. Alsaqaf, M. Daneva, and R. Wieringa, "Quality requirements in large-scale distributed agile projects–a systematic literature review," in *International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ)*, pp. 219–234, Springer, 2017.
31. R. B. Svensson, T. Gorschek, B. Regnell, R. Torkar, A. Shahrokni, and R. Feldt, "Quality requirements in industrial practice—an extended interview study at eleven companies," *IEEE Transactions on Software Engineering*, vol. 38, no. 4, pp. 923–935, 2011.
32. M. Umar and N. A. Khan, "Analyzing non-functional requirements (NFRs) for software development," in *2011 IEEE 2nd International Conference on Software Engineering and Service Science*, pp. 675–678, IEEE, 2011.
33. X. Zhang and X. Wang, "Tradeoff analysis for conflicting software non-functional requirements," *IEEE Access*, vol. 7, pp. 156463–156475, 2019.
34. P. Weichbroth, "Delivering usability in IT products: empirical lessons from the field," *International Journal of Software Engineering and Knowledge Engineering*, vol. 28, no. 07, pp. 1027–1045, 2018.
35. T. Suryawanshi and G. Rao, "A survey to support NFRs in agile software development process," *International Journal of Computer Science and Information Technologies*, vol. 6, no. 6, pp. 5487–5489, 2015.
36. N. S. Rosa, G. R. Justo, and P. R. Cunha, "A framework for building non-functional software architectures," in *2001 ACM Symposium on Applied Computing*, pp. 141–147, 2001.
37. R. Mizouni and A. Salah, "Towards a framework for estimating system NFRs on behavioral models," *Knowledge-Based Systems*, vol. 23, no. 7, pp. 721–731, 2010.
38. R. N. Charette, "The biggest IT failures of 2018," 2018 (accessed: 18.09.2020). `https://spectrum.ieee.org/riskfactor/computing/it/it-failures-2018-all-the-old-familiar-faces`.
39. R. R. Maiti and F. J. Mitropoulos, "Capturing, eliciting, predicting and prioritizing (CEPP) non-functional requirements metadata during the early stages of agile software development," in *SoutheastCon 2015*, pp. 1–8, IEEE, 2015.
40. Statista, "Largest software and programming companies worldwide by sales revenue from 2017 to 2020," 2020 (accessed: 10.11.2020). `https://www.statista.com/statistics/790179/worldwide-largest-software-programming-companies-by-sales/`.
41. Microsoft, "Build for the needs of the business," 2020 (accessed: 19.09.2020). `https://docs.microsoft.com/en-us/azure/architecture/guide/design-principles/build-for-business`.
42. Oracle, "Best practices for WLI application life cycle," 2020 (accessed: 10.11.2020). `https://docs.oracle.com/cd/E13214_01/wli/docs102/bestpract/requirements.html`.

43. K. Ossowska, L. Szewc, P. Weichbroth, I. Garnik, and M. Sikorski, "Exploring an ontological approach for user requirements elicitation in the design of online virtual agents," in *9th EuroSymposium on Systems Analysis and Design*, pp. 40–55, Springer, 2016.
44. K. Redlarski, "The impact of end-user participation in IT projects on product usability," in *2013 International Conference on Multimedia, Interaction, Design and Innovation (MIDI)*, pp. 1–8, 2013.
45. K. Redlarski and P. Weichbroth, "Hard lessons learned: delivering usability in IT projects," in *2016 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pp. 1379–1382, IEEE, 2016.
46. J. Buchan, M. Bano, D. Zowghi, S. MacDonell, and A. Shinde, "Alignment of stakeholder expectations about user involvement in agile software development," in *21st International Conference on Evaluation and Assessment in Software Engineering*, pp. 334–343, 2017.
47. B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," 2007. Technical Report EBSE-2007-01.
48. A. Jarzębowicz and P. Weichbroth, "A qualitative study on non-functional requirements in agile software development," *submitted, under review*, 2020.
49. Scopus, "Scopus content coverage guide," 2020 (accessed: 10.11.2020). `https://www.elsevier.com/__data/assets/pdf_file/0007/69451/Scopus_ContentCoverage_Guide_WEB.pdf`.
50. M. Daneva, D. Damian, A. Marchetto, and O. Pastor, "Empirical research methodologies and studies in requirements engineering: How far did we come?," *Journal of systems and software*, vol. 95, pp. 1–9, 2014.
51. J. Medeiros, D. C. Alves, A. Vasconcelos, C. Silva, and E. Wanderley, "Requirements engineering in agile projects: A systematic mapping based in evidences of industry," in *XVIII Ibero-American Conference on Software Engineering (CIBSE)*, pp. 460–476, 2015.
52. W. Alsaqaf, M. Daneva, and R. Wieringa, "Quality requirements challenges in the context of large-scale distributed agile: An empirical study," *Information and Software Technology*, vol. 110, pp. 39–55, 2019.
53. M. Oriol, P. Seppänen, W. Behutiye, C. Farré, R. Kozik, S. Martínez-Fernández, P. Rodríguez, X. Franch, S. Aaramaa, A. Abhervé, *et al.*, "Data-driven elicitation of quality requirements in agile companies," in *International Conference on the Quality of Information and Communications Technology*, pp. 49–63, Springer, 2019.
54. F. B. A. Ramos, A. A. M. Costa, M. Perkusich, H. O. Almeida, and A. Perkusich, "A non-functional requirements recommendation system for Scrum-based projects," in *30th International Conference on Software Engineering & Knowledge Engineering (SEKE)*, pp. 149–148, 2018.
55. E. Terpstra, M. Daneva, and C. Wang, "Agile practitioners' understanding of security requirements: insights from a grounded theory analysis," in *25th International Requirements Engineering Conference Workshops (REW)*, pp. 439–442, IEEE, 2017.
56. V. Sachdeva and L. Chung, "Handling non-functional requirements for big data and IOT projects in Scrum," in *7th International Conference on Cloud Computing, Data Science & Engineering-Confluence*, pp. 216–221, IEEE, 2017.
57. E.-M. Schön, J. Thomaschewski, and M. J. Escalona, "Agile requirements engineering: A systematic literature review," *Computer Standards & Interfaces*, vol. 49, pp. 79–91, 2017.

58. B. M. Aljallabi and A. Mansour, "Enhancement approach for non-functional requirements analysis in agile environment," in *2015 International Conference on Computing, Control, Networking, Electronics and Embedded Systems Engineering (ICCNEEE)*, pp. 428–433, IEEE, 2015.
59. M. Käpyaho and M. Kauppinen, "Agile requirements engineering with prototyping: A case study," in *23rd International Requirements Engineering Conference (RE)*, pp. 334–343, IEEE, 2015.
60. D. Domah and F. J. Mitropoulos, "The NERV methodology: A lightweight process for addressing non-functional requirements in agile software development," in *SoutheastCon 2015*, pp. 1–7, IEEE, 2015.
61. S. Dragicevic, S. Celar, and L. Novak, "Use of method for elicitation, documentation, and validation of software user requirements (MEDoV) in agile software development projects," in *6th International Conference on Computational Intelligence, Communication Systems and Networks*, pp. 65–70, IEEE, 2014.
62. J. Nawrocki, M. Ochodek, J. Jurkiewicz, S. Kopczyńska, and B. Alchimowicz, "Agile requirements engineering: A research perspective," in *40 th International Conference on Current Trends in Theory and Practice of Informatics (SOFSEM)*, pp. 40–51, Springer, 2014.
63. W. M. Farid and F. J. Mitropoulos, "Visualization and scheduling of non-functional requirements for agile processes," in *SoutheastCon 2013*, pp. 1–8, IEEE, 2013.
64. M. Bourimi and D. Kesdogan, "Experiences by using AFFINE for building collaborative applications for online communities," in *International Conference on Online Communities and Social Computing*, pp. 345–354, Springer, 2013.
65. W. Farid and F. Mitropoulos, "Novel lightweight engineering artifacts for modeling non-functional requirements in agile processes," in *SoutheastCon 2012*, pp. 1–7, IEEE, 03 2012.
66. T. Um, N. Kim, D. Lee, and H. P. In, "A quality attributes evaluation method for an agile approach," in *1st ACIS/JNU International Conference on Computers, Networks, Systems and Industrial Engineering*, pp. 460–461, IEEE, 2011.
67. B. Ramesh, L. Cao, and R. Baskerville, "Agile requirements engineering practices and challenges: an empirical study," *Information Systems Journal*, vol. 20, no. 5, pp. 449–480, 2010.
68. M. Bourimi, T. Barth, J. M. Haake, B. Ueberschär, and D. Kesdogan, "AFFINE for enforcing earlier consideration of NFRs and human factors when building sociotechnical systems following agile methodologies," in *International Conference on Human-Centred Software Engineering*, pp. 182–189, Springer, 2010.
69. B. Boehm, D. Rosenberg, and N. Siegel, "Critical quality factors for rapid, scalable, agile development," in *19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pp. 514–515, IEEE, 2019.
70. D. Ionita, C. van der Velden, H.-J. K. Ikkink, E. Neven, M. Daneva, and M. Kuipers, "Towards risk-driven security requirements management in agile software development," in *International Conference on Advanced Information Systems Engineering*, pp. 133–144, Springer, 2019.
71. J. Medeiros, A. Vasconcelos, M. Goulão, C. Silva, and J. Araújo, "An approach based on design practices to specify requirements in agile projects," in *ACM Symposium on Applied Computing*, pp. 1114–1121, 2017.
72. W. Behutiye, P. Karhapää, D. Costal, M. Oivo, and X. Franch, "Non-functional requirements documentation in agile software development: challenges and solution proposal," in *International Conference on Product-focused Software Process Improvement (PROFES)*, pp. 515–522, Springer, 2017.

73. C. Patel and M. Ramachandran, “Story Card Maturity Model (SMM): A process improvement framework for agile requirements engineering practices,” *Journal of Software (JSW)*, vol. 4, no. 5, pp. 422–435, 2009.
74. C. Patel and M. Ramachandran, “Bridging best traditional SWD practices with XP to improve the quality of XP projects,” in *International Symposium on Computer Science and its Applications*, pp. 357–360, IEEE, 2008.
75. W. Alsaqaf, M. Daneva, and R. Wieringa, “Understanding challenging situations in agile quality requirements engineering and their solution strategies: insights from a case study,” in *26th International Requirements Engineering Conference (RE)*, pp. 274–285, IEEE, 2018.
76. M. Younas, D. Jawawi, I. Ghani, and R. Kazmi, “Non-functional requirements elicitation guideline for agile methods,” *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, vol. 9, no. 3-4, pp. 137–142, 2017.
77. D. Jawawi, A. Arbain, W. Kadir, and I. Ghani, “Requirement traceability model for agile development: Results from empirical studies,” *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 8S, pp. 402–405, 2019.
78. A. F. Arbain, D. N. A. Jawawi, I. Ghani, and W. M. W. Kadir, “Non-functional requirement traceability process model for agile software development,” *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, vol. 9, no. 3-5, pp. 203–211, 2017.
79. R. J. Macasaet, L. Chung, J. L. Garrido, M. Noguera, and M. L. Rodríguez, “An agile requirements elicitation approach based on NFRs and business process models for micro-businesses,” in *12th International Conference on Product-Focused Software Development and Process Improvement (PROFES)*, pp. 50–56, 2011.
80. S. W. Ambler, “Beyond functional requirements on agile projects-strategies for addressing nonfunctional requirements,” 2008. Dr. Dobb’s Journal.
81. A. Firdaus, I. Ghani, D. N. A. Jawawi, and W. M. N. W. Kadir, “Non functional requirements (NFRs) traceability metamodel for agile development,” *Jurnal Teknologi*, vol. 77, no. 9, 2015.
82. A. F. B. Arbain, I. Ghani, and W. M. N. W. Kadir, “Agile non functional requirements (NFR) traceability metamodel,” in *8th Malaysian Software Engineering Conference (MySEC)*, pp. 228–233, IEEE, 2014.
83. L. Yu, E. Alégroth, P. Chatzipetrou, and T. Gorschek, “Utilising ci environment for efficient and effective testing of NFRs,” *Information and Software Technology*, vol. 117, p. 106199, 2020.
84. A. D. Sinnhofer, F. J. Oppermann, K. Potzmader, C. Orthacker, C. Steger, and C. Kreiner, “Increasing the visibility of requirements based on combined variability management,” in *International Symposium on Business Modeling and Software Design*, pp. 203–220, Springer, 2018.
85. S. Kopczyńska, M. Ochodek, and J. Nawrocki, “On importance of non-functional requirements in agile software projects—a survey,” in *Integrating Research and Practice in Software Engineering*, pp. 145–158, Springer, 2020.
86. L. López, W. Behutiye, P. Karhapää, J. Ralyté, X. Franch, and M. Oivo, “Agile quality requirements management best practices portfolio: A situational method engineering approach,” in *18 th International Conference on Product-Focused Software Process Improvement (PROFES)*, pp. 548–555, Springer, 2017.
87. W. Alsaqaf, “Engineering quality requirements in large scale distributed agile environment,” in *International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ) Workshops*, 2016.

88. P. Mohagheghi and M. E. Aparicio, “An industry experience report on managing product quality requirements in a large organization,” *Information and Software Technology*, vol. 88, pp. 96–109, 2017.
89. A. Silva, T. Araújo, J. Nunes, M. Perkusich, E. Dilorenzo, H. Almeida, and A. Perkusich, “A systematic review on the use of definition of done on agile software development projects,” in *21st International Conference on Evaluation and Assessment in Software Engineering (EASE)*, pp. 364–373, 2017.
90. L. López, S. Martínez-Fernández, C. Gómez, M. Choraś, R. Kozik, L. Guzmán, A. M. Vollmer, X. Franch, and A. Jedlitschka, “Q-rapids tool prototype: Supporting decision-makers in managing quality in rapid software development,” in *International Conference on Advanced Information Systems Engineering*, pp. 200–208, Springer, 2018.
91. C. R. Camacho, S. Marczak, and D. S. Cruzes, “Agile team members perceptions on non-functional testing: influencing factors from an empirical study,” in *11th International Conference on Availability, Reliability and Security (ARES)*, pp. 582–589, IEEE, 2016.
92. X. Franch, C. Gómez, A. Jedlitschka, L. López, S. Martínez-Fernández, M. Oriol, and J. Partanen, “Data-driven elicitation, assessment and documentation of quality requirements in agile software development,” in *International Conference on Advanced Information Systems Engineering*, pp. 587–602, Springer, 2018.
93. F. B. A. Ramos, A. Pedro, M. Cesar, A. A. M. Costa, M. Perkusich, H. O. de Almeida, and A. Perkusich, “Evaluating software developers’ acceptance of a tool for supporting agile non-functional requirement elicitation,” in *31st International Conference on Software Engineering & Knowledge Engineering (SEKE)*, pp. 26–42, 2019.
94. C. Pecchia, M. Trincardi, and P. Di Bello, “Expressing, managing, and validating user stories: Experiences from the market,” in *4th International Conference in Software Engineering for Defence Applications*, pp. 103–111, Springer, 2016.
95. B. R. Maxim and M. Kessentini, “An introduction to modern software quality assurance,” in *Software Quality Assurance*, pp. 19–46, Elsevier, 2016.
96. V. T. Heikkilä, D. Damian, C. Lassenius, and M. Paasivaara, “A mapping study on requirements engineering in agile software development,” in *41st Euromicro SEAA Conference*, pp. 199–207, IEEE, 2015.
97. P. Kapłański, “Controlled English interface for knowledge bases,” *Studia Informatica*, vol. 32, no. 2A, pp. 485–494, 2011.
98. P. Weichbroth, “Fluent editor and controlled natural language in ontology development,” *International Journal on Artificial Intelligence Tools*, vol. 28, no. 04, p. 1940007, 2019.