Software Quality Journal (2025) 33:28
https://doi.org/10.1007/s11219-025-09725-4

RESEARCH

A study on reported under-documented non-functional requirements as an indicator of technical debt

Ezequiel Scott[1] · Gabriela Robiolo[2,3] · Santiago Matalonga[4] · Michael Felderer[5,6] · Dietmar Pfahl[1]

Accepted: 10 June 2025 / Published online: 30 June 2025

Abstract
Fulfilling non-functional requirements (NFRs) is essential for the success of software systems. Nevertheless, NFRs are often treated second class when compared to functional requirements in development projects. To better understand the nature and effects of this imbalance, we explore the under-documentation of NFRs as an Indicator of Technical Debt. To achieve our aim, we exploit responses from an independently sourced global survey on requirements engineering: NaPiRE 2018. We analyze the responses under the assumption that NFRs related to quality attributes considered important must be documented. First, we retrieve data about the degree of documentation of NFRs and the perceived importance of quality attributes as defined by the standard ISO25010. Then, we check whether NFRs related to important quality attributes are reported as documented. If they are not, we consider this to be an indication of Technical Debt. Results from the statistical analysis are compared to findings from the literature. Our first finding is that there is no uniform pattern of what respondents consider to be important or unimportant quality attributes. However, the majority of respondents indicated that NFRs related to Maintainability, Reliability, Usability, and Performance were considered important. While the majority of responses confirm our expectation that NFRs related to quality attributes considered important are documented and NFRs related to quality attributes considered unimportant are not, there are responses indicating that NFRs related to important quality attributes are actually not documented. According to our assumption, these responses point to the existence of Technical Debt. As a side-product of our analysis, we also noted the existence of NFRs considered unimportant and documented. The presence of Technical Debt in NFR documentation may create different types of problems with several unwanted consequences, such as dissatisfied customers and inefficient development. Assuming that Technical Debt correlates with insufficiently documented NFRs of important quality attributes, we conclude that more effort should be spent by development organisations on adequate documentation of NFRs. It will pay off.

Keywords Quality characteristics · Quality attribute · Non-functional requirement · Technical debt

1 Introduction

It is hard to argue against the importance of Non-Functional Requirements (NFRs), as they are drivers for defining (Ameller et al., 2013) and evaluating (Kazman et al., 2000) software

Extended author information available on the last page of the article

architectures. However, NFRs often come second class to functional requirements, which represent a problem in software development since a system's utility is given by its functional and non-functional characteristics (Eckhardt et al., 2016; Chung and do Prado Leite, 2009). Documenting requirements is a prescribed practice in quality management models (Chrissis et al., 2007) and standards (ISO, 2015; ISO/IEC/IEEE, 2018), a practice that makes no distinction between functional requirements and NFRs (i.e., both types of requirements must be documented).

We argue that the practice of documenting requirements is of utmost importance in agile settings, too. "Continuous attention to technical excellence" is one of the twelve principles of the agile manifesto (Beck et al., 2001). Moreover, in contrast to those who have naïvely (Hoda and Noble, 2017) adopted agile values and principles, the need of having a shared understanding of requirements is often solved through some form of documentation, particularly in distributed agile settings (Matalonga et al., 2013).

This paper presents an extension of our previous research work (Robiolo et al., 2019), in which we explore whether NFRs that are related to important quality attributes are documented or not. In this paper, we include two additional research questions and we extend the discussion section substantially. Following (Robiolo et al., 2019), we assume that NFRs that are related to quality attributes perceived as important must be adequately documented. Therefore, when NFRs that are related to important quality attributes and not documented, there is an indication of Technical Debt. Technical Debt provides a framework for expressing the notion of trade-off between the short-term benefits and the long-term costs of software development decisions (Shull et al., 2013). Over-documenting involves effort and slows down the development process, while under-documenting costs less effort but exposes risks that may hinder development (Glazer, 2012). Enacting this understanding in the documentation of NFRs, we claim that to manage NFRs adequately, they must be documented because, otherwise, developers would not know about their precise nature or even their existence.

To explore how practitioners report on the documentation of NFRs, we analyzed a subset of data acquired from the most recent NaPiRE (Naming the Pain in Requirements Engineering) survey conducted in 2018 (Méndez, 2018). We analyzed practitioners' responses to understand how often they document NFRs associated with quality attributes that they consider important. As a result, we calculate the occurrence of potential Technical Debt (understood as NFRs not documented although considered important).

Our results show that, for the majority of responses, when a non-functional requirement is considered important, it is usually documented, and when it is not considered important, it is not documented. However, the results also show a clear indication of the existence of Technical Debt, i.e. future cost imposed by problems due to under-documented and thus under-specified NFRs. We also notice the existence of unimportant and documented NFRs. In particular, Technical Debt seems to be a problem with NFRs related to the quality attributes of Maintainability, Reliability, Usability, and Performance.

Furthermore, we exploited a subset of NaPiRE demographic variables to drill down the analysis. We selected four blocking factors, i.e., system class, process type, project size, and project type (distributed or not), to understand if any differences in the indications of Technical Debt could be assigned to these factors. We found no evidence about the influence of these factors on the indication of Technical Debt. Finally, we explored causes, problems, and effects for those respondents who indicated the existence of Technical Debt in their NFR documentation.

Although Technical Debt has been explored from different angles (Behutiye et al., 2017, 2020; Martini et al., 2018; dos Santos, 2013), Technical Debt of requirements is still one of the less-studied types of Technical Debt (Li et al., 2015). With a particular focus on non-functional

requirements, our study contributes to the understanding of this type of Technical Debt by identifying potential sources of Technical Debt in NFRs, the potential problems, and their causes and implications. Moreover, our study is original because our findings are based on the analysis of data taken from a large sample of responses collected from practitioners distributed worldwide. Therefore, increasing existing evidence of previous findings in the field. Applying this viewpoint, we show that most respondents follow the perception that if it is important, it must be documented. However, the indication of Technical Debt with regard to the documentation of NFRs is not uniform across different quality attributes, and they are big enough to merit attention. The potential presence of interactions and trade-offs deserve further investigation.

The remainder of this article is structured as follows. Section 2 provides the relevant background. Section 3 presents the applied research method. Section 4 presents the results, and Section 5 discusses the results. Finally, Section 6 concludes the paper and presents directions of future work.

2 Background and related work

In this section, we provide background information about the **Na**ming the **P**ain **i**n **R**equirements **E**ngineering (NaPiRE) initiative[1] and present an overview on research about NFRs.

2.1 The NaPiRE project

The NaPiRE project is an initiative that aims to establish a comprehensive theory of requirements engineering (RE) practice and to provide empirical evidence to practitioners that helps them address the challenges of requirements engineering in their projects. The main instrument of this project has been a global periodical survey. Since 2012, three installments of this survey have been carried out. The first round of the survey was conducted in Germany and the Netherlands in 2012 (Mendéz, 2013). The second round was conducted between 2014 and 2015 and included respondents from ten countries (Méndez, 2017). Finally, the third round of the survey was conducted in 2018 and collected responses from 42 countries. The research presented in this article contains an analysis based on the responses from the 2018 installment. The previous installments of the NaPiRe survey have been successful in sparking complementing research in several areas of requirements engineering. For instance, by comparing requirements engineering practices across geographical regions (Méndez, 2015; Kalinowski et al., 2016)), analyzing the role of requirements engineering practices in agile projects (Wagner et al., 2017, 2018), or developing a theory of requirements engineering (Wagner et al., 2019).

2.2 Non-functional requirements documentation

Research on NFRs has a long-standing tradition within software engineering in general and requirements engineering in particular. In this section, we focus our review of the literature on NFR documentation in the industry from practitioners' point of view.

Borg et al. (2003) surveyed two software development organizations. The authors interviewed 14 developers. Their results show that, in both organizations, functional requirements

[1] NaPiRE official website – http://napire.org

take precedence over non-functional requirements. The authors observed that, in both organizations, many NFRs remain undiscovered at the early stages of the software development lifecycle and –when discovered– are documented with a relatively low degree of formalism.

Berntsson Svensson et al. (2009) surveyed ten practitioners from five different organizations. They aimed to understand the challenges faced by practitioners when managing NFRs for embedded systems. Their results show that even within this context, the strategies for managing NFRs vary significantly from one setting to another. Their results indicate a general lack of documentation about NFRs, and a gradual dismissal of the importance of NFRs throughout the software development life cycle.

Behutiye et al. (2017) surveyed practitioners in four different organizations. Again, each organization followed different practices when dealing with NFRs. Of relevance to our research is that these authors also found evidence that NFRs are not documented.

Ameller et al. (2012) interviewed 13 software architects to investigate who decides which NFRs are important and who decides to document them. In line with this article, Ameller et al. (2012) intends to identify which types of NFRs are considered important to architects and if these are being documented. Their results show that performance and usability were the most important types of NFRs, and that documentation was usually imprecise and rarely maintained.

Eckhardt et al. (2016) were interested in identifying the characteristics of NFR documentation in different classes of systems. The work involved 11 industrial specifications from 5 different companies for different classes of systems and project sizes with 346 NFRs in total. The authors found a clear difference in the distribution of quality characteristics among the class of systems. For example, for Business information systems (BIS), they classified most NFRs in the categories security and functionality, while for Software-Intensive Embedded Systems (SIES), most NFRs were classified to be in the reliability category. In Hybrid Systems (HYB), i.e., systems combining characteristics of both BIS and SIES, the distribution among the quality characteristics was more balanced. Thus, the results indicate that the type of system class influences the relevancy of quality characteristics. The authors conclude that the specification and analysis procedures should be customized for different system classes.

2.3 Technical debt in NFRs and requirements documentation

The interest in Technical Debt introduced by NFRs has been emerging since 2013, particularly in the context of agile software development (dos Santos, 2013; Soares et al., 2015; Mendes et al., 2016; Behutiye et al., 2017; Behroozi and Kamandi, 2016). Some authors (Martini et al., 2014; Ampatzoglou et al., 2016; Holvitie, 2018; Soares et al., 2015) studied the causes and indicators of Technical Debt and one was associated with SIES.

dos Santos (2013) reported the experience of an architecture team in a software development department with 25 agile teams. The task of the architecture team was to inform decisions regarding technical practices in the context of a big oil company located in Brazil. The team proposed the use of a "technical debt board" with main Technical Debt categories to manage and visualize the high-level debt, combined with automated tools to measure it at low-level (using software metrics and other kinds of static analyses). Also, they used an automated tool that estimates how much effort would be required to fix each project debt. The NFR documentation was not mentioned as the authors focus on the description of the Technical Debt board. However, the documentation is implicit when visualizing and measuring Technical Debt.

Martini et al. (2014) conducted a multiple case study at seven sites of five large Scandinavian companies to shed light on the causes for the accumulation of architectural Technical Debt. They proposed a taxonomy of eight factors and their influence on the accumulation of Technical Debt. As one of these factors, they identified "Design and Architecture documentation: lack of specification/emphasis on critical architectural requirements." They highlight that some of the architectural requirements are not explicitly mentioned in the documentation. This causes misinterpretation by the developers implementing the code that is implicitly supposed to match such requirements. According to the informants, this is also threatening the refactoring activity and its estimation: refactoring a portion of code for which requirements were not written (but the code was "just working", implicitly satisfying them) might decrease satisfaction with such requirements. Although this study does not explicitly mention NFR documentation, critical architectural requirements tend to be associated with NFRs.

Soares et al. (2015) investigated difficulties in identifying and managing requirements with agile methodologies based on a literature review including 19 papers as well as an exploratory study where ten practitioners participated. The authors identified a lack of NFRs in identified requirements. They highlight that the lack of specification of NFRs can trigger future problems. Also, they identified documentation debt of agile requirements indicators and their causes.

Mendes et al. (2016) investigate the impact that Technical Debt documentation brings to projects developed by using agile methods. Particularly, a retrospective analysis was conducted in an industrial software project, i.e., the development of a web application. The application was developed in the context of a small company with about 50 employees. Ten professionals worked in the project selected for the study. The company used agile methods in its software development process and user stories for requirements definition. The analysis focused on data from 132 maintenance and evolution tasks, which had a significant impact on the project effort. Results of the study suggest that Technical Debt due to lack of documentation can significantly increase maintenance effort.

Behutiye et al. (2017) performed a study to analyze and synthesize the state of the art of Technical Debt, and its causes, consequences, and management strategies in the context of agile software development. Using a systematic literature review, 38 primary studies were identified and analyzed out of 346 studies. The authors report that the lack of understanding of the system being built (requirements) and inadequate test coverage are the second most reported causes of incurring Technical Debt. Moreover, "Missing knowledge or inadequate (not up-to-date) documentation" is identified as one of five aspects emphasized in Technical Debt definitions in the context of agile software development; however, it is not the most often mentioned aspect.

Holvitie (2018) focus on how Technical Debt occurs in and affects agile software processes, and how the software development techniques employed accommodate or mitigate the presence of that debt. Based on practitioner insights and experiences, the authors conducted a multi-national survey questionnaire, receiving 184 responses from practitioners in Brazil, Finland, and New Zealand. They reported that: (a) as a cause of Technical Debt "Inadecuate documentation" represents 55% of responses, ranked in the sixth place, (b) from the queried software development phases, in almost 90% of the cases a Technical Debt instance is perceived to affect the implementation, and in 40% of the cases the requirements/analysis, (c) the product backlog process artifact seems to increase the size of technical debt perceived in approx. 45% of the responses. The authors did not explicitly study NFRs.

Ampatzoglou et al. (2016) were interested in how practitioners developing SIES perceive Technical Debt. They conducted a multiple case study in the embedded systems industry to investigate: (a) the expected lifetime of components that have Technical Debt, (b) the most

frequently occurring types of Technical Debt in them, and (c) the significance of Technical Debt against run-time quality attributes. The case study was performed in seven embedded systems domains (telecommunications, printing, smart manufacturing, sensors, etc.) in five countries (Greece, Netherlands, Sweden, Austria, and Finland). The results of the case study suggest that: (a) Maintainability is more seriously considered when the expected lifetime of components is larger than ten years, (b) the most frequent types of Technical Debt are related to test, code, and architecture, and (c) in SIES, run-time quality attributes related to Technical Debt, such as Reliability, Functionality, and Performance, are prioritized over design-time quality attribute Maintainability. The authors also reported the frequency with which different types of Technical Debt occur. Documentation is ranked fourth with an occurrence of 60% percent, and requirements are ranked eighth with an occurrence of 25%.

Lenarduzzi and Fucci (2019) argue that the Technical Debt metaphor applies to requirements engineering in general and requirements engineering activities in particular, but that it is not well understood. Grounded in the existing literature, the authors present a holistic definition of requirements debt for functional requirements and NFRs comprising Technical Debt incurred during the identification, formalization, and implementation of requirements.

In summary, researchers have already started to investigate the relationship between requirements and the concepts of Technical Debt. Investigating these concepts focusing on NFRs is still an open area of research of high practical relevance.

3 Study design

In this section, we first present our understanding of the relevant concepts and the underlying assumptions of our research. Then, we introduce the terminology used in this paper. Afterward, we present our research questions. Finally, we describe the data extraction and analysis procedures that we followed in order to answer the stated research questions.

3.1 Concepts and assumptions

This section discusses the concepts that are used in the research design of our study. First, we describe the difference between quality attributes and NFRs; then, we present our interpretations of Technical Debt.

We draw our understanding of terms relating to requirements and quality attributes from the latest version of the ISO 25010:2011 standard (ISO/IEC Standard, 2011) on systems and software quality requirements and evaluation. According to this standard, an NFR is a "requirement that specifies criteria that can be used to judge the operation of a software system" (ISO/IEC Standard, 2011). The same standard defines quality attribute –or quality characteristic– as a specification of the stakeholders' needs along the following dimensions: Functional suitability, Performance efficiency, Compatibility, Usability, Reliability, Security, Maintainability, and Portability. The concepts *quality attribute* and *NFR* are related, and we argue that they are often (incorrectly) used interchangeably in the industry. With the exception of Functional suitability, all requirements related to *quality attributes* can be classified as *NFRs*.

In this article, we often use the terms *quality attribute* and *NFR* interchangeably. When we present our research method and the results of our analyses we prefer the more precise term *quality attribute*. The research team had an in-depth discussion about this issue. We settled on the term *quality attribute* for the following three reasons. First, the NaPiRE questionnaire

has not made this distinction evident, and the responses of the NaPiRE questionnaire, from which we draw in our study, use the term *quality attribute*, therefore maintaining the term guarantees alignment with the questionnaire construct. Second, our experience bias lets us think that the interchangeable use of terms is pervasive among practitioners and researchers (as observed by Behutiye et al. 2017). Finally, we claim that most practitioners do not care about the subtleties of this differentiation and are more concerned with the effects of Technical Debt in their requirements documentation.

Our research takes a process improvement point of view by assuming that both functional and non-functional requirements have to be documented, at least to a certain degree. To the best of our knowledge, all process improvement frameworks support this claim, in particular model-based frameworks such as CMMI (Chrissis et al., 2007) and standard-based frameworks such as ISO/IEC/IEEE 29148 (ISO/IEC/IEEE, 2018) and 15505 (ISO/IEC, 2004), and IEEE Std 610.12-1990 (IEEE, 1990). We further argue that this claim is valid regardless of whether an agile or plan-driven development approach is taken. In an agile setting, some form of documentation of NFRs must still be present to enable development (for a discussion, see Boehm and Turner 2003).

According to Seaman and Guo (2011), Technical Debt is "a metaphor for immature, incomplete, or inadequate artifacts in the software development life-cycle that cause higher costs and lower quality in the long run. These artifacts remaining in a system affect subsequent development and maintenance activities, and so can be seen as a type of debt that the system developers owe the system." Furthermore, Li et al. (2015) highlight that outdated documentation of requirements is a manifestation of Technical Debt. Given the valid assumption that important NFRs should also have up-to-date documentation, we can consider missing documentation for important NFRs as a manifestation of Technical Debt. Therefore, in our study, when a respondent of the NaPiRE survey indicates that NFRs associated with a quality attribute that is perceived important (see Table 1, Q1) but not documented (see Table 1, Q2), then we interpret this situation as an indication of Technical Debt. Figure 1 illustrates this conceptualization regarding the questionnaire items of NaPiRE.

It is worth noting that this conceptualization also includes a group of responses indicating the existence of unimportant and documented NFRs. In a preliminary study, Robiolo et al. (2019) interpreted this situation as an indication of waste, since waste is defined as any activity or work product that does not add value (Ikonen et al., 2010; Womack et al., 1990). However, the documentation of NFRs that are considered unimportant could also indicate adherence to good practices that practitioners do not perceive as important. These two possible interpretations and the lack of information in the questionnaire to disambiguate the meaning led us to reconsider our previous interpretation and adopt a more conservative approach by using the term "unimportant and documented" when referring to this group of responses.

Finally, the resulting (expected) gold standard scenario would be a development process where all quality attributes related to NFRs that are considered important for a development project are documented, and NFRs for quality attributes that are not considered important are not documented. We consider these two cases as expected situations.

3.2 Research questions

We formulate four research questions to address our research goal. The first question aims to identify an indication of Technical Debt in requirements documentation from a practitioners' perspective. The second question aims to understand whether the practitioners' contexts influence the responses related to the indication of Technical Debt in non-functional

Table 1 NaPiRE questionnaire items used in this study

ID	Questionnaire item	Possible responses	Variables
Q1	Are there quality attributes which are of particularly high importance for your development project? If yes, which one(s)?	Compatibility, Maintainability, Performance, Portability, Reliability, Safety, Security, Usability	v_6-v_13
Q2	Which classes of non-functional requirements do you explicitly consider in your requirements documentation?	Compatibility, Maintainability, Performance, Portability, Reliability, Safety, Security, Usability	v_97-v_102, v_303, v_103
Q3	How many people are involved in your project?	Free text	v_3
Q4	Please select the class of systems or services you work on in the context of your project.	Software-intensive embedded systems (SIES), Business information systems (BIS), Hybrid of both software-intensive embedded systems and business information systems (HYB)	v_4
Q5	How would you personally characterize your way of working in your project?	Agile, Rather agile, Hybrid, Rather plan-driven, Plan-driven	v_24
Q6	Is your project distributed?	Yes, No	v_16
Q7	Considering your personal experiences, how do the following problems in requirements engineering apply to your project?	Predefined list of problems[1]	v_174-v_193
Q8	Considering your personally experienced most critical problems selected in the previous question, which causes do they have?	Free text	v_277-v_281
Q9	Considering your personally experienced most critical problems selected in the previous question, which implications do they have?	Free text	v_282-v_286

[1](a) Communication flaws within the project team; (b) Communication flaws between the project and the customer; (c) Terminological problems; (d) Incomplete or hidden requirements; (e) Insufficient support by project lead; (f) Insufficient support by customer; (g) Stakeholders with difficulties in separating requirements from previously known solution designs; (h) Inconsistent requirements; (i) Missing traceability; (j) Moving targets (changing goals, business processes and/or requirements); (k) Gold plating (implementation of features without corresponding requirements); (l) Weak access to customer needs and/or (internal) business information; (m) Weak knowledge about customer's application domain; (n) Weak relationship between customer and project lead; (o) Time boxing/Not enough time in general; (p) Discrepancy between high degree of innovation and need for formal acceptance of (potentially wrong/incomplete/unknown) requirements; (q) Technically unfeasible requirements; (r) Underspecified requirements that are too abstract and allow for various interpretations; (s) Unclear/unmeasurable non-functional requirements; (t) Volatile customer's business domain regarding, e.g., changing points of contact, business processes or requirements

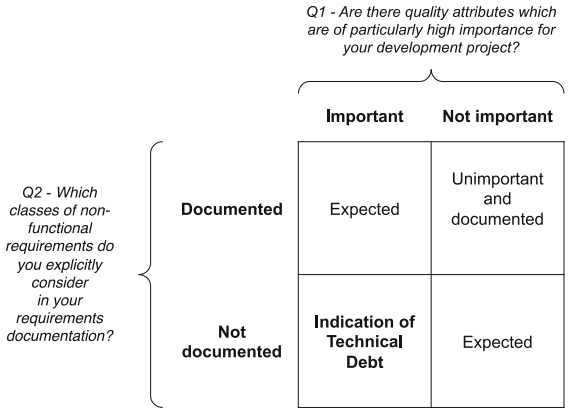

Fig. 1 Graphical representation of the conceptualization made from NaPiRE. Q1 and Q2 refer to the questionnaire items detailed in Table 1

requirements documentation. The third question aims to describe the most frequent problems of respondents indicating Technical Debt in their non-functional requirements. Finally, the fourth research question aims to describe the perceived causes and effects of those problems.

RQ1 For which quality attributes do practitioner responses indicate Technical Debt in their non-functional requirements documentation?

RQ2 How does the practitioners' context influence the indication of Technical Debt in non-functional requirements documentation?
Note: The context is described using the following factors: system class, project size, process type, and whether the project is geographically distributed or not.

RQ3 What are the differences in the problems reported by practitioners when they indicate Technical Debt in their non-functional requirements documentation?

RQ4 What are the differences in the reported causes and implications of the problems when practitioners indicate Technical Debt in their non-functional requirements documentation?

3.3 Data collection

We took the data from the NaPiRE 2018 data set and used the survey responses provided by the NaPiRE core team. This means that, to answer our research questions, we start from a curated data set that has been pre-processed. This pre-processing includes a standardized set of coding for all questionnaire questions and answers.

The NaPiRE 2018 global survey received 488 complete responses from practitioners located in 43 different countries. The survey originally includes 34 questions about requirement engineering in general; out of these, we selected a set of 9 questions that we consider relevant to answer the research questions. Table 1 shows the list of selected questions and their possible responses.

3.4 Data analysis

Figure 2 presents an overview of the analysis process. The path in bold depicts the main analysis path as conclusions were drawn from the analysis. To answer *RQ1 (For which quality attributes do practitioner responses indicate Technical Debt in their non-functional*

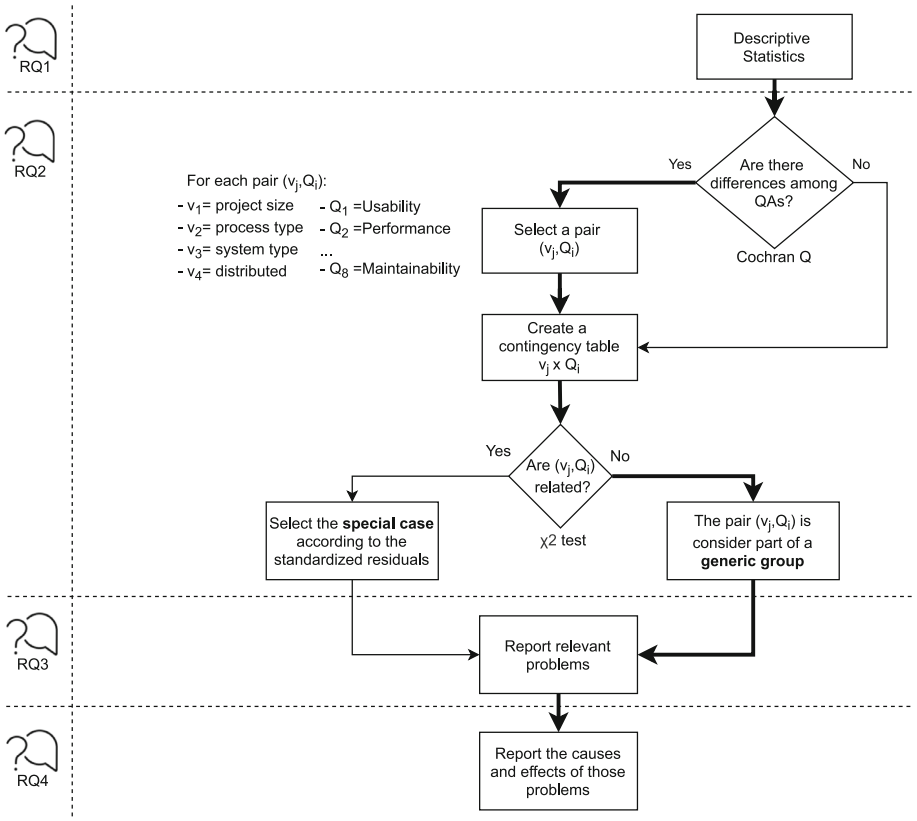

Fig. 2 Data Analysis steps applied to answer the research questions

requirements documentation?) we first cross-reference the responses related to the perceived importance of quality attributes (v_6 to v_13, Table 1) with the availability of documentation (v_97 to v_102, v_303, and v_103, Table 1). If for a quality attribute the result of the cross reference yields a combination of *important* and *not documented*, we interpret these responses as an indication of Technical Debt.

Once the number of responses indicating Technical Debt in non-functional requirements are identified, we rely on descriptive statistics to analyze them with regard to the different types of quality attributes: Usability, Security, Safety, Reliability, Portability, Performance, Maintainability, and Compatibility.

To answer *RQ2 (How does the practitioners' context influence the indication of Technical Debt in non-functional requirements documentation?)*, we first determine if the data show differences in the number of responses indicating Technical Debt in non-functional requirements regarding the types of quality attributes. We apply Chochran Q test (Cochran, 1950) to determine this. Chochran Q is a non-parametric test to verify if different treatments (i.e., the quality attributes) have identical effects on the variable under study (i.e., indication of Technical Debt). In other words, the test assesses whether the proportion of responses indicating Technical Debt is the same between groups of quality attributes. As shown in Section 4.3, the proportion of responses showing Technical Debt varies for the different types of quality attributes, which supports the analysis of each quality attribute separately.

Therefore, we analyze the number of responses related to Technical Debt by type of quality attribute and contextual factor. To do this, we first create contingency tables for all combinations of quality attributes (i.e., Compatibility, Maintainability, Performance, Portability, Reliability, Safety, Security, and Usability) and contextual factors (i.e., project size, process type, system class, and whether project is geographically distributed or not). The possible answer values of the contextual factors are nominal, except for project size, which is a numeric value. We categorize the projects as small-(S), medium- (M), and large-sized (L) projects by using two strategies: (1) we distribute the original responses into equal-sized categories to mitigate the bias introduced by having imbalanced classes, and (2) we follow the same criteria as in Méndez (2017): projects with up to 50 employees are considered small (S), with 51 to 250 are considered medium (M), and with more than 250 are considered large (L). As a result, the pre-processing steps lead to the analysis of $8 \times 5 = 40$ tables. We conduct Chi-square tests ($\chi 2$) to determine whether it is worth interpreting these contingency tables, and apply the Bonferroni correction method to counteract the problem of multiple comparisons ($\alpha = 0.05/40 = 0.00125$).

As a result, we identify the combinations of contextual factors and quality attributes that need further explanations. If no significant combinations are identified, then the responses are analyzed as whole, regardless of any contextual factor.

To answer RQ3, *(What are the differences in the problems reported by practitioners when they indicate Technical Debt in their non-functional requirements documentation?)*, we first divide responses into two groups: responses of participants who indicate Technical Debt in the documentation of at least one quality attribute and responses of participants who do not indicate Technical Debt in their non-functional documentation. Then, we analyze the top-5 problems gathered from question Q7 in both groups. Since respondents could rank the top-5 problems, we calculate the average ranking for each answer choice to determine which answer choice (i.e., problem) was the most selected overall. Thus, the problem with the largest average ranking is the most relevant. The average ranking is calculated as shown in (1), where each w is the weight of the ranked position, and x is the response count. Weights are applied in reverse: the respondent's most preferred choice (which they rank as #1) has the largest weight, and their least preferred choice (which they rank in the last position) has a weight of 1.

$$\text{Average ranking} = \frac{x_1 w_1 + x_2 w_2 + x_3 w_3 + \cdots + x_n w_n}{\text{Total response count}} \tag{1}$$

Using this calculation, we report on the most relevant problems of each group and their differences in the average ranking.

Finally, to answer RQ4 *(What are the differences in the reported causes and implications of the problems when practitioners indicate Technical Debt in their non-functional requirements documentation?)*, we analyze the reported causes and implications, gathered from question Q8 and Q9 respectively, for each of the identified problems in RQ3. Q8 and Q9 are free-text questions and their responses were translated, coded, and reviewed by the NAPiRE research team. We base our analysis on counting how often causes, problems, and both groups of respondents cited implications. We use Sankey diagrams (Riehmann et al., 2005) to illustrate these relationships.

4 Study results

In the following subsections, we present descriptive statistics characterizing the study population and, then, we answer the research questions.

4.1 Study population

As mentioned, the third round of NaPiRE (2018) contains responses from 488 practitioners from 43 different countries. The respondents were asked to answer 34 questionnaire items in total; out of these, we use a subset of 9 questionnaire items in our study. Table 1 shows the questionnaire items used. Q1 and Q2 are used to measure the indication of Technical Debt. Q3 to Q6 are used to characterize the context of the respondents in terms of system class, process type, project size, and whether the development is distributed or not. Table 2 presents the distribution of responses concerning these factors.

In the NaPiRE 2018 dataset, all the recorded responses are complete for Q1 (perceived importance of quality attributes) whereas only 455 responses are complete for Q2 (documentation of requirements for quality attributes). We also removed 57 responses due to incompleteness in the other variables of interest (Q4-Q6). As a result, we analyze a total of 390 responses for answering RQ1 and RQ2.

The responses to items Q7-Q9 were analyzed to answer RQ3 and RQ4. Out of the 390 responses that were analyzed for RQ1 and RQ2, 266 were complete regarding Q7-Q9. Within these responses, we finally use 188 responses that correspond to those participants who indicated Technical Debt in their MFRs documentation related to at least one quality attribute, and the remaining 78 responses that do not indicate the existence of Technical Debt in any quality attribute. Figure 3 illustrates the number of responses analyzed under each research question.

About half of the 390 practitioners who responded to the survey are actively involved in developing software-intensive embedded systems (SIES –described by the respondents as "Automotive, Embedded Software" or "Software for medical devices"), one quarter is involved in developing business intensive systems (BIS – "business intelligence for data centres" or "Software ERP"), and another quarter is active in developing a hybrid of both SIES and BIS (HYB).

Table 2 Distribution of responses with regard to the variables analyzed ($n = 390$)

Factor	Value	Count
System class	Business intensive systems (BIS)	199
	Hybrid systems (HYB)	98
	Software-intensive embedded systems (SIES)	93
Process type	Agile (A)	62
	Hybrid (H)	131
	Plan-driven (P)	35
	Rather agile (RA)	94
	Rather plan-driven (RP)	68
Project size (1)	Large (L)	119
	Medium (M)	133
	Small (S)	138
Project size (2)	Medium (M)	17
	Small (S)	373
Distributed	No	184
	Yes	206

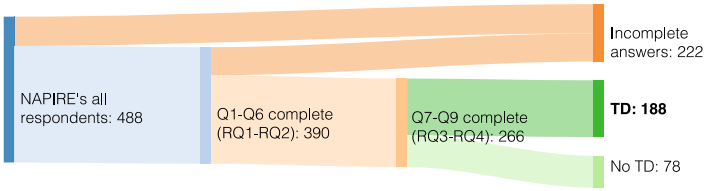

Fig. 3 Number of analyzed responses

The respondents also characterized their projects according to the number of people involved. Following the criteria used in Méndez (2017) to categorize the project size, the responses are mostly related to small-sized projects (i.e., up to 50 members) and a few are related to medium-sized projects (i.e., between 50 and 250 members).

Regarding process types, respondents could choose among five options, i.e., Agile, Rather agile, Hybrid, Rather plan-driven, and Plan-driven. The responses contain relatively large samples of all types, with hybrid being the most frequently mentioned process type.

4.2 Identification of technical debt in NFR documentation (RQ1)

Figure 4 shows a bar chart where the occurrence of Technical Debt in NFR documentation for each quality attribute is highlighted in orange. The percentage of responses indicating Technical Debt ranges from 11% to 31%. The dashed orange line marks the mean percentage, calculated across all quality attributes (22%). Observing the left-hand side of Fig. 4, four quality attributes stand out by exceeding the mean value: Usability (24%), Reliability (31%), Performance (23%), and Maintainability (31%). On the other hand, the quality attribute with the lowest number of responses indicating Technical Debt is Portability (11%).

Figure 4 also shows the number of responses in the category Expected in green. Most of the responses fall into this category, with percentages ranging from 62% to 79%. In addition, the red bars represent the percentages of responses indicating unimportant and documented quality attributes, which range from 7% to 14%. The dashed red line marks the mean percentage, calculated across all quality attributes (10%). Three quality attributes surpass the mean percentage, i.e., Compatibility (12%), Performance (12%), and Security (14%). Portability (10%) can be considered a borderline case, and Maintainability (7%) is the quality attribute with the lowest percentage.

Observation 1.1 The top-3 quality attributes with the highest indication of Technical Debt are Maintainability, Reliability, and Usability. In contrast, Portability shows the lowest value.
Observation 1.2 Overall, 22% of the participants indicate Technical Debt associated with NFR documentation.

4.3 Practitioners' context influence (RQ2)

To answer *RQ2 (How does the practitioners' context influence the perception of Technical Debt in NFR documentation?)*, we first analyzed if the responses indicating Technical Debt vary across the different quality attributes. Although the differences can be seen from the plot

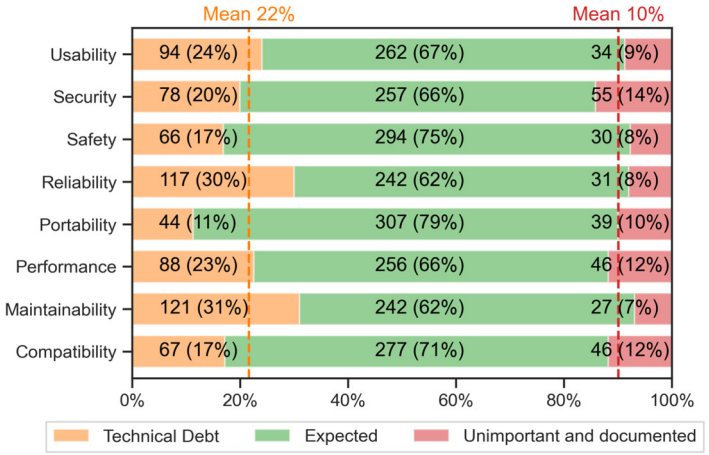

Fig. 4 Bar plot showing the number of responses indicating Technical Debt by Quality Attribute ($n = 390$)

in Fig. 4, we conducted a statistical test (Cochran's Q test) to assess whether the proportion of positive responses (i.e., responses indicating Technical Debt) is the same between groups (i.e., quality attributes). The test result supports the analysis of indication of Technical Debt for each quality attribute separately ($Q = 86.242$; $p < 0.001$; $df = 7$).

Then, we characterize the practitioners' context using the following factors: system class, project size, process type, and whether the project was distributed or not. Table 3 shows the full distribution of responses by factor and Quality Attribute.

On the responses indicating Technical Debt, we were not able to determine any influencing context factor since all the conducted tests indicate non-statistically significant relationships ($p \geq \alpha$). This result suggests that, in the analyzed sample, the occurrence of Technical Debt is independent of contextual factors such as system class, project size, process type, and whether the project was distributed or not.

Observation 2.1 The percentages of responses indicating Technical Debt in NFR documentation varies with regard to the type of quality attribute.
Observation 2.2 The percentages of responses indicating Technical Debt in NFR documentation is independent of the context.

4.4 Observable problems (RQ3)

To answer *RQ3 (What are the differences in the problems reported by practitioners when they indicate Technical Debt in their non-functional requirements documentation?*, we identify the most frequent problems of two groups: the respondents who indicate Technical Debt in at least one type of quality attribute (*TD Group*, $n = 188$) and the respondents who do not indicate Technical Debt in any quality attribute (*Non-TD Group*, $n = 78$). The problems are based on the set of 21 predefined general RE problems listed in the NaPiRE questionnaire.

Table 3 Distribution of responses by blocking factor and quality attribute ($n = 390$)

Factor	Answer[1]	Category[2]	Compat.	Maint.	Performance	Portability	Reliability	Safety	Security	Usability
Q4	BIS	E	144 (72%)	109 (55%)	132 (66%)	162 (81%)	124 (62%)	150 (75%)	128 (64%)	132 (66%)
		TD	35 (18%)	75 (38%)	42 (21%)	18 (9%)	63 (32%)	39 (20%)	37 (19%)	52 (26%)
		UD	20 (10%)	15 (8%)	25 (13%)	19 (10%)	12 (6%)	10 (5%)	34 (17%)	15 (8%)
	HYB	E	72 (73%)	67 (68%)	63 (64%)	76 (78%)	62 (63%)	72 (73%)	64 (65%)	68 (69%)
		TD	15 (15%)	27 (28%)	25 (26%)	15 (15%)	28 (29%)	15 (15%)	21 (21%)	22 (22%)
		UD	11 (11%)	4 (4%)	10 (10%)	7 (7%)	8 (8%)	11 (11%)	13 (13%)	8 (8%)
	SIES	E	61 (66%)	66 (71%)	61 (66%)	69 (74%)	56 (60%)	72 (77%)	65 (70%)	62 (67%)
		TD	17 (18%)	19 (20%)	21 (23%)	11 (12%)	26 (28%)	12 (13%)	20 (22%)	20 (22%)
		UD	15 (16%)	8 (9%)	11 (12%)	13 (14%)	11 (12%)	9 (10%)	8 (9%)	11 (12%)
Q5	A	E	40 (65%)	33 (53%)	37 (60%)	44 (71%)	38 (61%)	45 (73%)	46 (74%)	42 (68%)
		TD	16 (26%)	25 (40%)	20 (32%)	10 (16%)	21 (34%)	14 (23%)	10 (16%)	17 (27%)
		UD	6 (10%)	4 (6%)	5 (8%)	8 (13%)	3 (5%)	3 (5%)	6 (10%)	3 (5%)
	H	E	92 (70%)	83 (63%)	83 (63%)	104 (79%)	92 (70%)	101 (77%)	92 (70%)	93 (71%)
		TD	21 (16%)	40 (31%)	33 (25%)	16 (12%)	30 (23%)	18 (14%)	23 (18%)	26 (20%)
		UD	18 (14%)	8 (6%)	15 (11%)	11 (8%)	9 (7%)	12 (9%)	16 (12%)	12 (9%)
	P	E	27 (77%)	19 (54%)	26 (74%)	29 (83%)	25 (71%)	29 (83%)	21 (60%)	21 (60%)
		TD	4 (11%)	14 (40%)	4 (11%)	4 (11%)	8 (23%)	5 (14%)	10 (29%)	9 (26%)
		UD	4 (11%)	2 (6%)	5 (14%)	2 (6%)	2 (6%)	1 (3%)	4 (11%)	5 (14%)
	RA	E	66 (70%)	65 (69%)	61 (65%)	75 (80%)	48 (51%)	68 (72%)	55 (59%)	61 (65%)
		TD	17 (18%)	26 (28%)	19 (20%)	5 (5%)	38 (40%)	21 (22%)	24 (26%)	26 (28%)
		UD	11 (12%)	3 (3%)	14 (15%)	14 (15%)	8 (9%)	5 (5%)	15 (16%)	7 (7%)
	RP	E	52 (76%)	42 (62%)	49 (72%)	55 (81%)	39 (57%)	51 (75%)	43 (63%)	45 (66%)
		TD	9 (13%)	16 (24%)	12 (18%)	9 (13%)	20 (29%)	8 (12%)	11 (16%)	16 (24%)
		UD	7 (10%)	10 (15%)	7 (10%)	4 (6%)	9 (13%)	9 (13%)	14 (21%)	7 (10%)

Table 3 continued

Factor	Answer[1]	Category[2]	Compat.	Maint.	Performance	Portability	Reliability	Safety	Security	Usability
Q3[3]	L	E	85 (71%)	75 (63%)	82 (69%)	92 (77%)	80 (67%)	87 (73%)	79 (66%)	77 (65%)
		TD	21 (18%)	35 (29%)	27 (23%)	15 (13%)	31 (26%)	19 (16%)	23 (19%)	31 (26%)
		UD	13 (11%)	9 (8%)	10 (8%)	12 (10%)	8 (7%)	13 (11%)	17 (14%)	11 (9%)
	M	E	94 (71%)	85 (64%)	82 (62%)	108 (81%)	80 (60%)	98 (74%)	88 (66%)	99 (74%)
		TD	21 (16%)	37 (28%)	28 (21%)	11 (8%)	39 (29%)	25 (19%)	23 (17%)	23 (17%)
		UD	18 (14%)	11 (8%)	23 (17%)	14 (11%)	14 (11%)	10 (8%)	22 (17%)	11 (8%)
	S	E	98 (71%)	82 (59%)	92 (67%)	107 (78%)	82 (59%)	109 (79%)	90 (65%)	86 (62%)
		TD	25 (18%)	49 (36%)	33 (24%)	18 (13%)	47 (34%)	22 (16%)	32 (23%)	40 (29%)
		UD	15 (11%)	7 (5%)	13 (9%)	13 (9%)	9 (7%)	7 (5%)	16 (12%)	12 (9%)
Q3[4]	M	E	13 (76%)	10 (59%)	8 (47%)	16 (94%)	15 (88%)	16 (94%)	10 (59%)	13 (76%)
		TD	4 (24%)	5 (29%)	6 (35%)	0 (0%)	2 (12%)	1 (6%)	3 (18%)	4 (24%)
		UD	0 (0%)	2 (12%)	3 (18%)	1 (6%)	0 (0%)	0 (0%)	4 (24%)	0 (0%)
	S	E	264 (71%)	232 (62%)	248 (66%)	291 (78%)	227 (61%)	278 (75%)	247 (66%)	249 (67%)
		TD	63 (17%)	116 (31%)	82 (22%)	44 (12%)	115 (31%)	65 (17%)	75 (20%)	90 (24%)
		UD	46 (12%)	25 (7%)	43 (12%)	38 (10%)	31 (8%)	30 (8%)	51 (14%)	34 (9%)
Q6	No	E	133 (72%)	111 (60%)	128 (70%)	151 (82%)	112 (61%)	139 (76%)	125 (68%)	117 (64%)
		TD	31 (17%)	64 (35%)	38 (21%)	17 (9%)	60 (33%)	37 (20%)	31 (17%)	52 (28%)
		UD	20 (11%)	9 (5%)	18 (10%)	16 (9%)	12 (7%)	8 (4%)	28 (15%)	15 (8%)
	Yes	E	144 (70%)	131 (64%)	128 (62%)	156 (76%)	130 (63%)	155 (75%)	132 (64%)	145 (70%)
		TD	36 (17%)	57 (28%)	50 (24%)	27 (13%)	57 (28%)	29 (14%)	47 (23%)	42 (20%)
		UD	26 (13%)	18 (9%)	28 (14%)	23 (11%)	19 (9%)	22 (11%)	27 (13%)	19 (9%)

[1]BIS: Business intensive systems; HYB: Hybrid systems; SIES: Software-intensive embedded systems; A: Agile; H: Hybrid; P: Plan-driven; RA: Rather agile; RP: Rather plan-driven; L: Large; M: Medium; S: Small

[2]E: Expected; TD: Technical Debt; UD: Unimportant and documented

[3]Categories S, M, L were created by distributing the responses into equal-sized buckets: small-sized (< 7), medium-sized (between 7 and 15) and large-sized projects (> 15)

[4]Categories S, M, L follow the criteria used in Méndez (2017): small-sized (< 50), medium-sized (between 50 and 250), and large-sized projects (> 250)

Table 4 shows the top-5 most relevant problems for both groups, as determined by the weighted average of the original rankings. The most relevant problems for respondents in the TD Group are *"Unclear/unmeasurable non-functional requirements"*, followed by *"Technically unfeasible requirements"*, *"Weak knowledge about customer's application domain"*, *"Discrepancy between high degree of innovation and need for formal acceptance of potentially wrong, incomplete, or unknown requirements"*, and *"Weak relationship between customer and project lead"*.

In contrast, respondents in the Non-TD Group ranked problems in a different order than the TD Group. For instance, the problem *"Discrepancy between high degree of innovation and need for formal acceptance of potentially wrong, incomplete, or unknown requirements"* was ranked higher. In addition, some problems are in the top-5 of the TD Group (e.g., *"Inconsistent requirements"*), but they are not listed in the top-5 of the Non-TD Group. Figure 5 shows a barplot with the differences in the average ranking of problems for both groups. The bars are ordered by the difference in the ranking values, that is, the problems with larger differences between groups are shown at the top, whereas the problems with smaller differences are shown at the bottom.

Observation 3.1 The five most relevant problems for respondents indicating Technical Debt in non-functional requirements documentation are:

1) *"Unclear/unmeasurable non-functional requirements"*
2) *"Technically unfeasible requirements"*
3) *"Weak knowledge about customer's application domain"*
4) *"Discrepancy between high degree of innovation and need for formal acceptance of potentially wrong, incomplete, or unknown requirements"*
5) *"Weak relationship between customer and project lead"*

Table 4 Ranking of relevant problems described by respondents who indicate Technical Debt in their non-functional requirements (TD Group) and respondents who do not (Non-TD Group)

Ranking	TD Group	Non-TD Group
1	Unclear/unmeasurable non-functional requirements	Discrepancy between high degree of innovation and need for formal acceptance of (potentially wrong, incomplete, or unknown) requirements
2	Technically unfeasible requirements	Inconsistent requirements
3	Weak knowledge about customer's application domain	Insufficient support by customer
4	Discrepancy between high degree of innovation and need for formal acceptance of (potentially wrong, incomplete, or unknown) requirements	Missing traceability
5	Weak relationship between customer and project lead	Weak relationship between customer and project lead

Fig. 5 Average ranking of problems reported by respondents who indicate Technical Debt in their non-functional requirements (TD Group) and respondents who do not (Non-TD Group)

4.5 Perceived causes and implications of the problems (RQ4)

To answer *RQ4 (What are the differences in the reported causes and implications of the problems when practitioners indicate Technical Debt in their non-functional requirements documentation?*, we described the reported causes and implications of the main problems that we found in RQ3 (see Section 4.4). The results show that there are many causes related to problems and that these problems have multiple implications. We also observe little agreement in the responses, probably due to the wide variety of responses. Nevertheless, we depict the relationships between causes, problems, and effects by using Sankey diagrams. The numbers next to each cause, problem, and implication denote the number of responses that report the same pairs $< cause, problem >$ or $< problem, implication >$.

Figure 6 shows the results for the TD Group. The most frequently mentioned causes are *Lack of time*, *Missing domain knowledge*, and *Missing customer involvement*. In particular, *Missing domain knowledge* is highly cited as a cause of *Weak knowledge about customer's application domain*.

From the figure, it can be seen that the most relevant problems have multiple implications, and several implications have been cited frequently, such as *Customer dissatisfaction*, *Time overrun*, *Poor requirements quality (general)*, and *Inefficient development*.

On the other hand, the Non-TD Group reports on different causes, problems, and implications. Figure 7 shows the results for the Non-TD Group, where the most frequently mentioned causes are *Communications flaws between team and customer*, *Missing customer involvement*, and *Lack of requirements management*. In particular, *Communications flaws between team and customer* is highly cited as a cause of *Weak relationship between customer and project lead*.

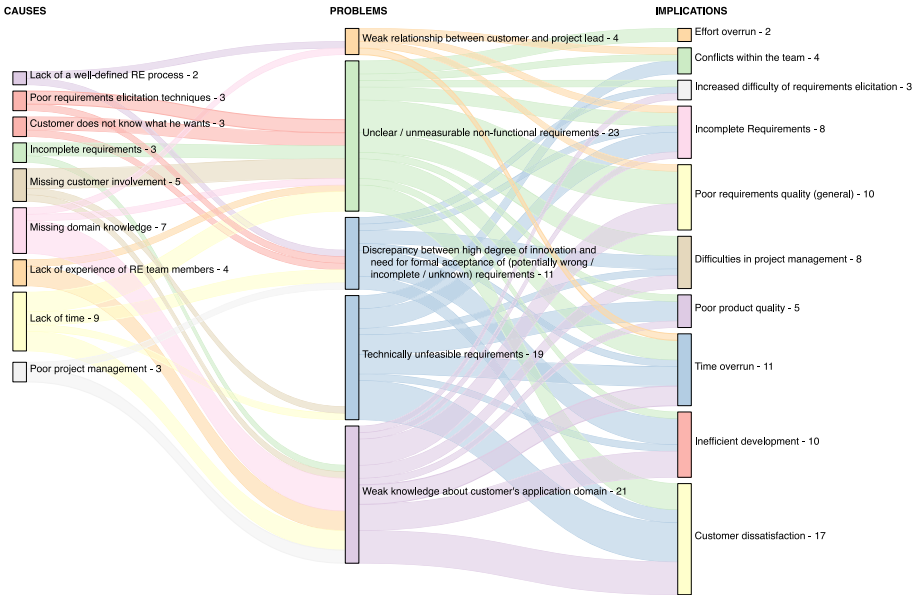

Fig. 6 Causes and implications of the top-5 most relevant problems for respondents indicating Technical Debt in their non-functional requirements documentation. The numbers denote the responses that report the pairs $< cause, problem >$ or $< problem, implication >$

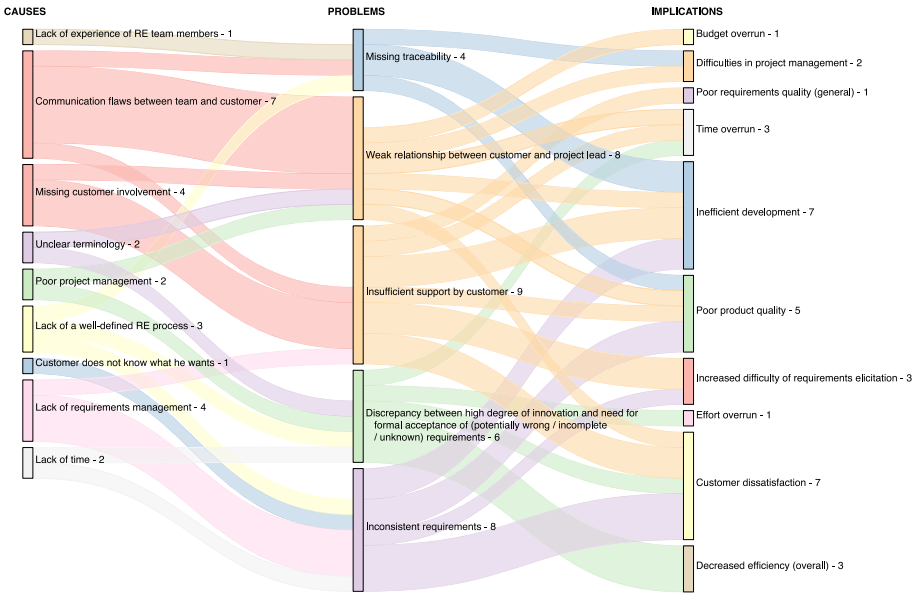

Fig. 7 Causes and implications of the top-5 most relevant problems for respondents who do NOT indicate Technical Debt in their non-functional requirements documentation. The numbers denote the responses that report the pairs $< cause, problem >$ or $< problem, implication >$

As in the TD Group, the reported problems have multiple implications, including *Inefficient development*, *Customer dissatisfaction*, and *Effort overrun*.

Observation 4.1 Respondents indicating Technical-Debt in NFR documentation indicate several causes, such as *Lack of time*, *Missing domain knowledge*, and *Missing customer involvement*, that lead to *Unclear/unmeasurable non-functional requirements*, which is the most relevant problem. This problem has multiple implications such as *Poor requirements quality (general)*, and *Customer dissatisfaction*.

In comparison, both groups show a large variety of different causes that lead to problems, and the problems have multiple implications. This variety can be seen in the spread of the flows in the Sankey diagrams. The groups also show different causes and implications of their problems. For example, the TD Group indicates that several causes lead to *Unclear/unmeasurable non-functional requirements*, which is the most relevant problem, and that problem has multiple implications such as *Poor requirements quality (general)*, and *Customer dissatisfaction*.

5 Discussion

Our research is rooted in the hypothesis that a positive correlation exists between the importance of a quality attribute and the level of documentation of related NFRs. This assumption was put to test with data independently sourced from the NaPiRE global survey. The results presented in the previous section suggest the plausibility of this hypothesis.

In the following, we first describe the relation of our observations to existing evidence published in the scientific literature; then, we discuss the results regarding the relation between NFRs with Technical Debt and associated problems, causes, and effects. Finally, we discuss possible threats to the validity of our study.

5.1 Relation to existing evidence

Observation 1.1 points out that some quality attributes are more frequently related to an indication of Technical Debt in NFR documentation. These quality attributes are Usability, Reliability, and Maintainability. The literature supports the association between Maintainability and Technical Debt, yet we identified few studies connecting Usability and Reliability. For example, Kruchten et al. (2012) claim that Maintainability is strongly connected to the concept of Technical Debt. Arvanitou et al. (2019) consider Maintainability as a proxy of Technical Debt. Furthermore, they state that when a practitioner is not aware of the amount of Technical Debt, any decision that aims at improving Maintainability will lower the amount of Technical Debt. In the context of service- and microservice-based systems, Bogner et al. (2018) state that to avoid Technical Debt in a long-living software system, industry needs to improve their quality control: they need to go beyond source code level metrics and also include an architecture-centric view on software evolution, potentially with scenario-based methods.

In addition, Li et al. (2015) performed a survey to validate an architectural Technical Debt identification approach based on architecture decisions and change scenarios. Their method only considered the quality attribute Maintainability. Sas and Avgeriou (2019) found that

practitioners rarely adopt tools for monitoring Maintainability. This behavior causes them to overlook important trade-offs between quality attributes that lead to incurring Technical Debt.

Regarding Reliability, Ampatzoglou et al. (2016) state that the embedded systems industry prioritizes Reliability, Functionality, and Performance against Maintainability. This low prioritization of Maintainability contributes to the accumulation of Technical Debt. This conclusion is in line with our results since Reliability and Performance are the quality attributes with the highest Technical Debt for the SIES group (see Table 3). The lower percentage of Technical Debt associated with Maintainability –still considerably high– supports the lower priority given to this quality attribute. Moreover, Arvanitou et al. (2020) investigated real-world projects that apply Performance and Portability optimizations (as they are performed in practice) without controlling their consequences on Maintainability. They performed a case study based on six applications in Exascale software development, monitoring the impact of the modifications introduced using SkePU (Performance) and StarPU (Portability) tools. The results reveal that in most cases, SkePU is not hurting the system's maintainability, whereas StarPU seems to affect the maintainability of refactored code negatively. Observation 1.1 is partially supported by the literature. To the best of our knowledge, current studies support the existence of Requirements' Technical Debt related to Reliability and Maintainability. However, we could not find studies related to Usability and Requirements' Technical Debt.

Observation 1.2 shows that, overall, 22% of the participants indicate under-documentation of NFRs related to important quality attributes, which we consider to be an indication of Technical Debt. To the best of our knowledge, this is the first study that explores this phenomenon using a large sample of respondents. On a smaller scale, Behutiye et al. (2020) found that 4 out of 12 interviewees (33%) reported accumulation of technical debt, with increased development and maintenance time and system quality degradation, due to missing and outdated QR documentation (NFR documentation).

Observation 2.1 states that the occurrence of Technical Debt varies across quality attributes as perceived by practitioners. We could not find any literature that compares measurement of Technical Debt in terms of documentation and importance of quality attributes. Slightly related to our study, but interesting as it is an example of automated measurement of Technical Debt based on SonarQube, Letouzey (2012) present a method for estimating the quality and the incurred Technical Debt of source code.

Furthermore, our results show that the indication of Technical Debt in NFR documentation is not dependent on the system class, the process type, the project size, or the distributed characteristics of the project. Although there are surveys, such as Ramač et al. (2021), that show that Respondents of different demographics have reported technical Debt, we did not find a survey study that focuses on Technical Debt and NFR documentation. Our observation seems to support the assumption that the Technical Debt metaphor is applicable in a wide variety of software development contexts, and it encourages researchers to investigate other contextual factors such as the application domain, or the used technologies (Ciolkowski et al., 2021).

Observation 3.1 shows that the most relevant problem for respondents indicating Technical Debt in NFR documentation are *"Unclear/unmeasurable non-functional requirements"*, followed by *"Technically unfeasible requirements"*, and *"Weak knowledge about customer's application domain"*. These problems have multiple implications, as shown in **Observations 4.1**, which are related to the documentation of the requirements. In this line,

Alves et al. (2014) identified five indicators of documentation debt that are related to incomplete requirements, i.e., the documentation does not exist, incomplete design specification, incomplete documentation, insufficient comments in code, and outdated documentation.

Observation 4.1 shows that there is a large variety of causes and implications reported by practitioners indicating Technical Debt in their NFR documentation. These results are in line with findings from several authors. For example, four studies (Behutiye et al., 2017; Martini et al., 2014; Rios et al., 2018; Yli-Huumo et al., 2014) agree that lack of time is a cause of Technical Debt. Two of them (Rios et al., 2018; Yli-Huumo et al., 2014) also mention poor project management as a cause of Technical Debt. Particularly, Rios et al. (2018) point out different aspects of project management within the industrial context. Three of them (Behutiye et al., 2017; Rios et al., 2018; Yli-Huumo et al., 2014), who are interested in the implications of Technical Debt, agree that low product quality and time overrun are important implications. In particular, Martini et al. (2014) point out that developers can implement solutions that mismatch the intended requirements when architectural requirements, which are closely related to NFRs, are not explicitly mentioned in the documentation, threatening the intended product quality.

Table 5 summarizes the observations and the supporting references.

5.1.1 Technical debt induced by NFR documentation (RQ1)

Generally, our results show that most participants in the most recent NaPiRE survey state that they document NFRs (i.e., requirements related to quality attributes) when important, and they do not document them when unimportant. This is what we expected.

However, we observed that a substantial subset of respondents state that they do not document important NFRs. This is what we interpret as an indication of Technical Debt. Certain types of quality attributes related to NFRs were particularly prone to this phenomenon: Maintainability, Reliability, and Usability.

One can only speculate about the reasons why respondents say they do not document this specific subset of NFRs. For example, it might be difficult to document these types of NFRs, or those who should specify the requirements do not know how to document them properly, or appropriate tool support is missing. There can be many more reasons. Also, the standard reason for omitting important work, time pressure (or other pressures), might play a role.

There can be specific reasons for each type of NFR. For example, when looking at the NFRs related to Maintainability, the problem might be that it is difficult to tell how to document such requirements. One could argue that making code maintainable means that it should not contain code smells and, therefore, be regularly refactored. Avoiding code smells and refactoring is often a general development practice that does not have to be specified repeatedly in each project. This phenomenon might also be true for other NFR types, i.e., there exist standard procedures or standard requirements that always hold and do not have to be explicitly stated in each project. In other words, when respondents answered the NaPiRE questionnaire, they might only have thought about project-specific documentation of NFRs.

5.1.2 Influence of context on the occurrence of technical debt in NFR documentation (RQ2)

Our results show differences in the whole set of responses regarding the indication of Technical Debt and the different types of NFRs (e.g., Maintainability, Usability, Performance);

Table 5 Summary of the existing evidence that supports our observations

Obs#	Description	Related Work
1.1	Top-3 quality attributes showing the highest number of responses related to Technical Debt are Maintainability, Reliability, and Usability. In contrast, Portability shows the lowest value.	Kruchten et al. (2012); Arvanitou et al. (2019); Bogner et al. (2018); Sas and Avgeriou (2019); Ampatzoglou et al. (2016); Li et al. (2015); Arvanitou et al. (2020)
1.2	Overall, 22% of the participants indicate they have Technical Debt in their NFRs documentation.	–
2.1	The percentages of responses indicating Technical Debt in NFR documentation varies with regard to the type of quality attribute.	Letouzey (2012)
2.2	The occurrence of Technical Debt is independent of the context.	–
3.1	The top-5 problems for respondents indicating Technical Debt in NFRs documentation are: 1) *"Unclear/unmeasurable non-functional requirements"*, 2) *"Technically unfeasible requirements"*, 3) *"Weak knowledge about customer's application domain"*, 4) *"Discrepancy between high degree of innovation and need for formal acceptance of potentially wrong, incomplete, or unknown requirements"*, and 5) *"Weak relationship between customer and project lead"*	Alves et al. (2014)
4.1	Respondents indicating Technical-Debt in NFRs documentation indicate several causes, such as *Lack of time*, *Missing domain knowledge*, and *Missing customer involvement*, that lead to *Unclear/unmeasurable non-functional requirements*, which is the most relevant problem. This problem has multiple implications such as *Poor requirements quality (general)*, and *Customer dissatisfaction*.	Behutiye et al. (2017); Martini et al. (2014); Rios et al. (2018); Yli-Huumo et al. (2014)

therefore, we analyzed each type separately. The observed variance is expected since the NaPiRE participants constitute a heterogeneous group with different contexts in which certain types of NFRs (grouped by quality attribute) are more relevant than others.

Since the different types of NFRs are not enough to explain the variation in the data regarding the indication of Technical Debt, we analyzed each type of NFRs with regard to the practitioners' context based on four blocking factors: system class, process type, project size, and distributed project.

Our results show that the types of NFRs and the selected blocking factors cannot explain differences in the number of responses related to the indication of Technical Debt. Thus, the indication of Technical Debt in NFR documentation seems to be independent of the context of the project, as defined in this study. The studied factors are mainly related to the software development process, and there are many other factors that we cannot study since they are not part of the NaPiRE questionnaire. In this regard, a broader number of factors need to be further considered when studying Technical Debt, such as the domain of application, the technology used, and the underlying architectural design.

5.1.3 Observable patterns of problems related to technical debt (RQ3)

Problems may be thought of as circumstances present in the requirement engineering process that must be dealt with. While *Time boxing* is the most frequently cited problem, *Communication flaws between the project and the customer* and *Incomplete or hidden requirements* are the second and third most frequently mentioned problems identified. This result seems plausible since time boxing and communication flaws might push developers to produce incomplete or low-quality software that makes difficult maintenance at a later point. At the same time, unclear and technically infeasible requirements might induce extra documentation that later becomes useless.

5.1.4 Perceived causes and implications (RQ4)

Having a poor project management and *Lack of time* are the two most frequent problems related to NFR documentation stated by respondents indicating Technical Debt. Moreover, *Low product quality* and *Time overrun* are the main implications of having Technical Debt related to NFR documentation. These implications are significant for software engineering practitioners since they can lead to unsuccessful projects. Therefore, the measurement and management of Technical Debt in software projects are crucial to avoid low-quality products and time overrun.

Methods and automated tools to facilitate the management of Technical Debt is a topic worth to be studied. In this line, there are some initiatives that manage Technical Debt, such as Software Quality Assessment Based on Lifecycle Expectations (SQALE) Method, which also has an automated tool that supports it (SonarQube) (Letouzey, 2012). However, these kinds of tools can control specific quality attributes from source code (e.g., security), and there is still pending a method and tool oriented to measure Technical Debt in all types of NFR documentation.

5.2 Implications for researchers and practitioners

The results of our study have several implications for researchers and practitioners:

- Based on the responses from NaPiRE, we show that evidence of Technical Debt is present in the documentation of NFRs. Therefore, the documentation of NFRs is critical for software development, and it should be considered a potential source of Technical Debt.
- Although Technical Debt may give short-term benefits, incurring in NFRs Technical Debt is often related to specific problems, which may threaten the project success; thus, practitioners should consider these problems when making informed decisions related to the documentation of NFRs.
- When documenting NFRs, Technical Debt is more frequently associated with certain quality attributes such as Maintainability, Reliability, and Usability. Software solutions that have any of these quality attributes as drivers should pay more attention to the management of Technical Debt, particularly in their documentation of NFRs.
- When studying Technical Debt in NFR documentation, researchers should spend less time investigating certain context factors (i.e., system class, process type, project size, and distributed characteristics) than others, such as application domain, the technology used, and the underlying architectural design.
- Lack of time and Missing domain knowledge are the main causes of problems when NFR documentation is subject to Technical Debt. Consequently, software development teams

should consider the time needed to understand the application domain and document important NFRs.

- Our findings motivate the development and application of tools and techniques for Technical Debt Management of NFRs. Although building useful tools (e.g., accurate, complete, and easy to use) is a challenging task (Ciolkowski et al., 2021), such tools could support developers in keeping track of important NFRs and their documentation. Possibly, there could be semi-automatic support for creating document templates that need to be completed/adjusted by developers. If tools establish traceability between NFRs and their documentation, developers could be alerted if documentation might need to be updated (or has become obsolete) in case the NFRs change. Our study motivates the application of these tools regardless of the system class, process type, project size, and distributed characteristics.

5.3 Limitations

This section presents a discussion of the possible threats to construct, internal, external, and conclusion validity (according to the classification presented by Wohlin et al. (2012)) as well as measures taken to mitigate them. As mentioned by Linaker et al. (2015), when conducting survey-based research, good practice mandates that threats can be identified early in the process so that they can be mitigated. As this research is grounded on the data acquired through the 2018 instance of the NaPiRE survey, it is also subjected to the threats from that survey (Wagner et al., 2019). This section presents the threats and biases that this research process may have injected into our results. When the threat is discussed derived from NaPiRE, the discussion from the research team is conveyed, and so is the rationale for the mitigation strategy (most often to accept the risk).

Regarding *construct validity*, this research may be subject to differences in the interpretation of the concepts used in the NaPiRE survey. Of particular importance for this research, the NaPiRE survey does not appear to differentiate between quality attributes and NFRs. Both concepts are used interchangeably in the questionnaire. As a research team, we decided to accept this threat as: 1) We cannot act on the construct of the questionnaire. 2) We claim there is value in looking at the NaPiRE 2018 data set from a different point of view than the one it was designed for. 3) We have made the case that, following the results from Eckhardt et al. (2016), practitioners are also unlikely to differentiate these terms (see Section 3.1).

Another possible threat to *construct validity* stems from our interpretation of Technical Debt. First, the concept of Technical Debt was not part of the main research goal of the NaPiRE questionnaire. Therefore we are adding a layer of abstraction to the participants' answers. Those answers and our interpretation of Technical Debt might not yield the same output had the participants been questioned directly about the metaphor. Similarly, the respondents were asked about the perceived causes and implications of the problems in requirements engineering rather than the perceived causes and implications of Technical Debt in NFRs. However, we argue that this threat is mitigated by the fact that our definition and interpretation of Technical Debt is broad in the sense that we make few assumptions about the concept. We claim that, at this level of abstraction, the concept is sound to convey the findings of this research. Moreover, we also discussed our results with respect to other researchers' findings as a strategy to mitigate this threat and to enhance the validity and reliability of our observations. Although this strategy gives more confidence in our research findings, we still encourage further studies to test our observations. For instance, a study based on a

specifically designed questionnaire to investigate the perceived causes and implications of Technical Debt in NFRs could strengthen our findings.

The categorization of the variable "project size" is another threat to construct validity. We used two strategies to categorize this variable. The first strategy is based on the criteria used by a previous NAPIRE study Méndez (2017), whereas the second strategy distributes the original responses into equal-sized categories. The first strategy introduces the bias of having imbalanced classes, and the second one introduces boundaries that may be considered counter-intuitive (i.e., too small or too large) for the label assigned (e.g., projects with 7-15 people are considered "medium-sized" projects, which can be debatable). Since there is no standardized and optimal way to identify suitable boundaries of the categories, we applied both strategies and studied the results. In any case, our results show that "project size" is not a significant context factor when practitioners indicate Technical Debt in NFRs.

Regarding *internal validity*, respondents bias is a long-established source of bias in survey-based research (Summers and Hammonds, 1969). The NaPiRE 2018 survey is no exception to this threat. However, it is also a strength that the 2018 instance of the survey is the result of two cycles of improvement (see Section 2.1). In terms of the research process of this work, care has been taken to differentiate the conclusions about the data (see Section 4) from our attempts of generalizing the findings through an unstructured literature review (see Section 5.1).

Regarding *external validity*, we have mitigated this type of threat by being explicit in our decisions about our data cleaning criteria (see Section 3.3) to be able to perform a thorough analysis. For instance, in Section 4, it can be seen how the number of data points is lower as the analysis drills down into RQ3 and RQ4. This inherently limits our capacity to generalize the results. We discussed and understood this risk during the design phase of this research. Had the number of complete answers been different, we argue that increasing data points per affirmation would have strengthened the results from RQ3 and RQ4. Finally, as shown in Section 5.1, we performed an unstructured literature review to identify supporting evidence for all the claims from our findings.

Finally, regarding *conclusion validity*, the analysis process is made available, and we evaluated and aboded by the statistical preconditions of each test.

6 Conclusions

This work has explored the relationship between the reported levels of importance of NFRs, i.e., requirements related to quality attributes, and the degree to which they are documented. We have introduced the concepts of Technical Debt to analyze this relationship. To explore this relationship, we exploit the data from the 2018 execution of the NaPiRE survey.

Our analysis starts from the assumption that requirements related to quality attributes considered important must be documented. And likewise, requirements related to quality attributes **not** considered important should not be documented. To test this hypothesis, we have described how the responses to the NaPiRE questionnaire have been interpreted. The first observation resulting from our analysis is that NFRs related to important quality attributes are documented by the majority of the responses. However, our results show that some quality attributes considered important are not being documented –a phenomenon that we describe as an indication of *Technical Debt*. In addition, but to a lesser extent, requirements related to quality attributes considered unimportant are documented. Maintainability, Reliability, Usability, and Performance are the quality attributes frequently considered important but not documented.

Throughout our analysis process, we have extracted a series of *observations* which we intended to cross-reference to the literature (see Table 5). We envisioned these observations would serve two purposes: to confirm the literature results and guide future research lines.

To conclude, NFR documentation is relevant to improve product quality, prevent effort overrun, avoid requirements issues and customer dissatisfaction, which are all important aspects of software engineering. The management of Technical Debt in NFR documentation becomes critical for the success of software projects, and the use of adequate tools with some degree of amortization may help with this task. Noways, this is a challenge for the research community that needs new proposals.

Managing the presence of TD will minimise the problems and the consequences indentified in this work. We conclude that, that the effort spent in documenting important NFR will be payed off in a reduction of problems experienced by development projects

Acknowledgements The authors would like to thank all practitioners who responded to NaPiRE surveys as well as all colleagues who have supported the NaPiRE initiative.

Author Contributions All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by Ezequiel Scott and Gabriela Robiolo. The literature review was prepared by Gabriela Robiolo. The first draft of the manuscript was written by Santiago Matalonga, and all authors commented on previous versions of the manuscript. Michael Felderer and Dietmar Pfahl contributed substantially to the discussion, limitation, and conclusion sections. All authors read and approved the final manuscript.

Funding Ezequiel Scott and Dietmar Pfahl were supported by the Estonian Center of Excellence in ICT research (EXCITE). In addition, Dietmar Pfahl was supported by the group grant PRG1226 funded by the Estonian Research Council.
Gabriela Robiolo was supported by Universidad Austral.

Availability of Data and Materials The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

Declarations

Competing Interests The authors declare no competing interests.

References

Alves, N.S.R., Ribeiro, L.F., Caires, V., Mendes, T.S., Spínola, R.O. (2014). Towards an Ontology of Terms on Technical Debt. In: 2014 Sixth International Workshop on Managing Technical Debt, pp. 1–7. https://doi.org/10.1109/MTD.2014.9. ISSN: null

Ameller, D., Ayala, C., Cabot, J., Franch, X. (2012) How do software architects consider non-functional requirements: An exploratory study. In: 2012 20th IEEE International Requirements Engineering Conference, RE 2012 - Proceedings, Chicago, USA, pp. 41–50

Ameller, D., Ayala, C., Cabot, J., & Franch, X. (2013). Non-functional requirements in architectural decision making. *IEEE Software, 30*(2), 61–67. https://doi.org/10.1109/MS.2012.176

Ampatzoglou, A., Ampatzoglou, A., Chatzigeorgiou, A., Avgeriou, P., Abrahamsson, P., Martini, A., Zdun, U., Systa, K. (2016). The Perception of Technical Debt in the Embedded Systems Domain: An Industrial Case Study. In: 2016 IEEE 8th International Workshop on Managing Technical Debt (MTD), pp. 9–16. https://doi.org/10.1109/MTD.2016.8

Arvanitou, E.-M., Ampatzoglou, A., Bibi, S., Chatzigeorgiou, A., Stamelos, I. (2019). Monitoring Technical Debt in an Industrial Setting. In: Proceedings of the Evaluation and Assessment on Software Engineering. EASE '19, pp. 123–132. Association for Computing Machinery, Copenhagen, Denmark. https://doi.org/10.1145/3319008.3319019. Accessed 2020-01-21

Arvanitou, E.-M., Ampatzoglou, A., Nikolaidis, N., Tzintzira, A.-A., Ampatzoglou, A., Chatzigeorgiou, A. (2020). Investigating trade-offs between portability, performance and maintainability in exascale systems. In: 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp. 59–63. https://doi.org/10.1109/SEAA51224.2020.00020

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., et al. (2001). Manifesto for Agile Software Development

Behroozi, N., Kamandi, A. (2016). Waste elimination of agile methodologies in web engineering. In: 2016 Second International Conference on Web Research (ICWR), pp. 102–107. https://doi.org/10.1109/ICWR.2016.7498453

Behutiye, W., Pertti, K., Costal, D., Oivo, M., Franch, X. (2017). Non-functional requirements documentation in agile software development: Challenges and solution proposal. In: Product-Focused Software Process Improvement. PROFES 2017. Lecture Notes in Computer Science, Innsbruck, pp. 515–522

Behutiye, W., Rodríguez, P., Oivo, M., Aaramaa, S., Partanen, J., Abhervé, A. (2020). How agile software development practitioners perceive the need for documenting quality requirements: a multiple case study. In: 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp. 93–100. IEEE

Behutiye, W. N., Rodríguez, P., Oivo, M., & Tosun, A. (2017). Analyzing the concept of technical debt in the context of agile software development: A systematic literature review. *Information and Software Technology, 82*, 139–158. https://doi.org/10.1016/j.infsof.2016.10.004. Accessed 2019-11-07

Berntsson Svensson, R., Gorschek, T., Regnell, B. (2009). Quality requirements in practice: An interview study in requirements engineering for embedded systems. In: REFSQ 2009: Requirements Engineering: Foundation for Software Quality, pp. 218–232

Boehm, B., Turner, R. (2003). Balancing Agility and Discipline: A Guide for the Perplexed, 1st edn., p. 304. Addison-Wesley/Pearson Education,, ???

Bogner, J., Fritzsch, J., Wagner, S., Zimmermann, A. (2018). Limiting technical debt with maintainability assurance: an industry survey on used techniques and differences with service- and microservice-based systems. In: Proceedings of the 2018 International Conference on Technical Debt. TechDebt '18, pp. 125–133. Association for Computing Machinery, Gothenburg, Sweden. https://doi.org/10.1145/3194164.3194166. Accessed 2020-01-21

Borg, A., Yong, A., Carlshamre, P., Sandahl, K. (2003). The Bad Conscience of Requirements Engineering : An Investigation in Real-World Treatment of Non-Functional Requirements. *Third Conference on Software Engineering Research and Practice in Sweden* (SERPS'03), Lund

Chrissis, M.B., Konrad, M., Shrum, S. (2007). CMMI: Guidelines for Process Integration and Product Improvement, 2nd edn., p. 676. Addison-Wesley, Upper Saddle River, NJ. http://www.loc.gov/catdir/toc/ecip0617/2006023484.html

Chung, L., do Prado Leite, J.C.S. (2009). On non-functional requirements in software engineering. In: Borgida, A.T., Chaudhri, V.K., Giorgini, P., Yu, E.S. (eds.) Conceptual Modeling: Foundations and Applications: Essays in Honor of John Mylopoulos, pp. 363–379. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-02463-4_19

Ciolkowski, M., Lenarduzzi, V., & Martini, A. (2021). 10 years of technical debt research and practice: Past, present, and future. *IEEE Software, 38*(6), 24–29.

Cochran, W.G. (1950). The comparison of percentages in matched samples. *Biometrika37*(3-4), 256–266. http://oup.prod.sis.lan/biomet/article-pdf/37/3-4/256/421327/37-3-4-256.pdf. https://doi.org/10.1093/biomet/37.3-4.256

dos Santos, P.S.M., Varella, A., Dantas, C.R., Borges, D.B. (2013). Visualizing and Managing Technical Debt in Agile Development: An Experience Report. In: Baumeister, H., Weber, B. (eds.) Agile Processes in Software Engineering and Extreme Programming. Lecture Notes in Business Information Processing, pp. 121–134. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-38314-4_9

Eckhardt, J., Vogelsang, A., Fernández, D.M. (2016). Are "non-functional" requirements really non-functional? In: Proceedings of the 38th International Conference on Software Engineering - ICSE '16, pp. 832–842. ACM Press, New York, USA

Glazer, H. (2012). High Performance Operations: Leverage Compliance to Lower Costs, Increase Profits, and Gain Competitive Advantage. Upper Saddle River, N.J: FT Press.

Hoda, R., Noble, J. (2017). Becoming Agile: A Grounded Theory of Agile Transitions in Practice. In: Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering, ICSE 2017, pp. 141–151

Holvitie, J., Licorish, S. A., Spínola, R. O., Hyrynsalmi, S., MacDonell, S. G., Mendes, T. S., Buchan, J., & Leppänen, V. (2018). Technical debt and agile software development practices and processes: An industry practitioner survey. *Information and Software Technology, 96*, 141–160. https://doi.org/10.1016/j.infsof.2017.11.015. Accessed 2019-11-07

IEEE (1990). IEEE standard glossary of software engineering terminology. *IEEE Std 610.12-1990,* 1–84
Ikonen, M., Kettunen, P., Oza, N., Abrahamsson, P. (2010). Exploring the sources of waste in kanban software development projects. In: 2010 36th EUROMICRO Conference on Software Engineering and Advanced Applications, pp. 376–381
ISO (2015). ISO 9001:2015. Quality Management Systems - Requirements
ISO/IEC (2004). ISO/IEC 15504-1:2004 Information technology Process assessment Part 1: Concepts and vocabulary
ISO/IEC Standard (2011). ISO/IEC 25010:2011 Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models
ISO/IEC/IEEE. (2018). ISO/IEC/IEEE 29148:2018 Systems and software engineering – Life cycle processes – Requirements engineering. Technical report, *International Standards Organization*
Kalinowski, M., Felderer, M., Conte, T., Spínola, R., Prikladnicki, R., Winkler, D., Fernández, D.M., Wagner, S. (2016) Preventing incomplete/hidden requirements: reflections on survey data from Austria and Brazil. In: International Conference on Software Quality, pp. 63–78. Springer
Kazman, R., Klein, M., Clements, P. (2000). Atam: Method for architecture evaluation. Technical Report CMU/SEI-2000-TR-004, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=5177
Kruchten, P., Nord, R. L., & Ozkaya, I. (2012). Technical Debt: From Metaphor to Theory and Practice. *IEEE Software, 29*(6), 18–21. https://doi.org/10.1109/MS.2012.167
Lenarduzzi, V., Fucci, D. (2019). Towards a holistic definition of requirements debt. In: 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), pp. 1–5. IEEE
Letouzey, J.-L. (2012). The SQALE method for evaluating Technical Debt. In: 2012 Third International Workshop on Managing Technical Debt (MTD), pp. 31–36. https://doi.org/10.1109/MTD.2012.6225997. ISSN: null
Li, Z., Avgeriou, P., & Liang, P. (2015). A systematic mapping study on technical debt and its management. *Journal of Systems and Software, 101*, 193–220.
Linaker, J., Sulaman, S.M., Höst, M., de Melo, R.M. (2015). Guidelines for Conducting Surveys in Software Engineering. Technical report, Lund University. https://lup.lub.lu.se/search/publication/5366801
Martini, A., Bosch, J., Chaudron, M. (2014). Architecture Technical Debt: Understanding Causes and a Qualitative Model. In: 2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications, pp. 85–92. https://doi.org/10.1109/SEAA.2014.65. ISSN: 1089-6503, 2376-9505
Martini, A., Besker, T., & Bosch, J. (2018). Technical Debt tracking: Current state of practice: A survey and multiple case study in 15 large organizations. *Science of Computer Programming, 163*, 42–61. https://doi.org/10.1016/j.scico.2018.03.007. Accessed 2020-01-21
Matalonga, S., Solari, M., & Matturro, G. (2013). Factors affecting distributed agile projects: A systematic review. *International Journal of Software Engineering and Knowledge Engineering, 23*(09), 1289–1301. https://doi.org/10.1142/S021819401350040X
Mendes, T.S., de F. Farias, M.A., Mendonça, M., Soares, H.F., Kalinowski, M., Spínola, R.O. (2016). Impacts of Agile Requirements Documentation Debt on Software Projects: A Retrospective Study. In: Proceedings of the 31st Annual ACM Symposium on Applied Computing. SAC '16, pp. 1290–1295. ACM, New York, NY, USA. https://doi.org/10.1145/2851613.2851761. event-place: Pisa, Italy. http://doi.acm.org/10.1145/2851613.2851761 Accessed 2019-11-07
Mendéz Fernández, D., Wagner, S. (2013). Naming the pain in requirements engineering: design of a global family of surveys and first results from Germany. In: EASE - International Conference on Evaluation and Assessment in Software Engineering (17th, 2013, Porto de Galinhas), pp. 2298–2338
Méndez Fernández, D. (2018). Supporting Requirements-Engineering Research That Industry Needs: The NaPiRE Initiative. *IEEE Software, 35*(1), 112–116.
Méndez Fernández, D., Wagner, S., Kalinowski, M., Felderer, M., Mafra, P., Vetrò, A., Conte, T., Christiansson, M.-T., Greer, D., Lassenius, C., Männistö, T., Nayabi, M., Oivo, M., Penzenstadler, B., Pfahl, D., Prikladnicki, R., Ruhe, G., Schekelmann, A., Sen, S., … Wieringa, R. (2017). Naming the pain in requirements engineering. *Empirical Software Engineering, 22*(5), 2298–2338.
Méndez Fernández, D., Wagner, S., Kalinowski, M., Schekelmann, A., Tuzcu, A., Conte, T., Spinola, R., & Prikladnicki, R. (2015). Naming the Pain in Requirements Engineering: Comparing Practices in Brazil and Germany. *IEEE Software, 32*(5), 16–23.
Ramač, R., Mandić, V., Taušan, N., Rios, N., Freire, S., Pérez, B., Castellanos, C., Correal, D., Pacheco, A., Lopez, G., et al. (2021). Prevalence, common causes and effects of technical debt: Results from a family of surveys with the it industry. *Journal of Systems and Software,* 111114
Riehmann, P., Hanfler, M., Froehlich, B. (2005). Interactive sankey diagrams. In: IEEE Symposium on Information Visualization, 2005. INFOVIS 2005., pp. 233–240. IEEE

Rios, N., Spínola, R.O., Mendonça, M., Seaman, C. (2018). The most common causes and effects of technical debt: first results from a global family of industrial surveys. In: Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering And Measurement. ESEM '18, pp. 1–10. Association for Computing Machinery, Oulu, Finland. https://doi.org/10.1145/3239235.3268917. https://doi.org/10.1145/3239235.3268917 Accessed 2020-01-29

Robiolo, G., Scott, E., Matalonga, S., Felderer, M. (2019). Technical Debt and Waste in Non-functional Requirements Documentation: An Exploratory Study. In: 20th International Conference on Product-Focused Software Process Improvement, pp. 220–235. Springer, ???. https://doi.org/10.1007/978-3-030-35333-9_16. http://link.springer.com/10.1007/978-3-030-35333-9_16

Sas, D., & Avgeriou, P. (2019). Quality attribute trade-offs in the embedded systems industry: an exploratory case study. *Software Quality Journal*. https://doi.org/10.1007/s11219-019-09478-x. Accessed 2020-01-21

Seaman, C., Guo, Y. (2011). Chapter 2 - measuring and monitoring technical debt. *Advances in Computers82*, 25–46

Shull, F., Falessi, D., Seaman, C., Diep, M., Layman, L. (2013). Technical debt: Showing the way for better transfer of empirical results. In: Perspectives on the Future of Software Engineering, pp. 179–190. Springer, ???

Soares, H.F., Alves, N.S.R., Mendes, T.S., Mendonça, M., Spínola, R.O. (2015). 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. https://doi.org/10.1109/ITNG.2015.68

Summers, G. F., & Hammonds, A. D. (1969). Toward a paradigm for respondent bias in survey research. *The Sociological Quarterly, 10*(1), 113–121. https://doi.org/10.1111/j.1533-8525.1969.tb02066.x

Wagner, S., Méndez Fernández, D., Felderer, M., Kalinowski, M. (2017). Requirements Engineering Practice and Problems in Agile Projects; Results from an international survey. In: 2017 Iberoamerican Conference on Software Engineering (CiBSE 2017), pp. 85–98

Wagner, S., Mendez, D., Felderer, M., Graziotin, D., Kalinowski, M. (2019) Challenges in survey research. arXiv:1908.05899

Wagner, S., Méndez-Fernández, D., Kalinowski, M., Felderer, M. (2018) Agile requirements engineering in practice: Status quo and critical problems. *CLEI Electronic Journal21*(1)

Wagner, S., Fernández, D. M., Felderer, M., Vetrò, A., Kalinowski, M., Wieringa, R., Pfahl, D., Conte, T., Christiansson, M.-T., Greer, D., et al. (2019). Status quo in requirements engineering: A theory and a global family of surveys. *ACM Transactions on Software Engineering and Methodology (TOSEM), 28*(2), 9–1948.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A. (2012). Experimentation in Software Engineering. Springer, ???

Womack, J.P., Jones, D.T., Roos, D. (1990). The Machine that Changed the World, p. 339. Rawson Associates, ???

Yli-Huumo, J., Maglyas, A., Smolander, K. (2014). The Sources and Approaches to Management of Technical Debt: A Case Study of Two Product Lines in a Middle-Size Finnish Software Company. In: Jedlitschka, A., Kuvaja, P., Kuhrmann, M., Männistö, T., Münch, J., Raatikainen, M. (eds.) Product-Focused Software Process Improvement. *Lecture Notes in Computer Science*, pp. 93–107. Springer, Cham. https://doi.org/10.1007/978-3-319-13835-0_7

Authors and Affiliations

Ezequiel Scott[1] · Gabriela Robiolo[2,3] · Santiago Matalonga[4] · Michael Felderer[5,6] · Dietmar Pfahl[1]

✉ Ezequiel Scott
ezequielscott@gmail.com

Gabriela Robiolo
grobiolo@austral.edu.ar

Santiago Matalonga
santiago.matalonga@uws.ac.uk

Michael Felderer
Michael.Felderer@dlr.de

Dietmar Pfahl
dietmar.pfahl@ut.ee

1 Institute of Computer Science, University of Tartu, Narva mnt 18, Tartu 51009, Tartumaa, Estonia

2 Facultad de Ingenieria, Universidad Austral, Mariano Acosta 1611, Pilar B1630FHB, Buenos Aires, Argentina

3 LIDTUA (CIC), Mariano Acosta 1611, Pilar B1630FHB, Buenos Aires, Argentina

4 The University of the West of Scotland, High Street, Paisley PA1 2BE, Paisley, United Kingdom

5 Institute of Software Technology, German Aerospace Center (DLR), Muenchener Str. 20, Wessling 82234, Germany

6 Department of Mathematics and Computer Science, University of Cologne, Albertus-Magnus-Platz, Cologne 50923, Germany

Terms and Conditions

Springer Nature journal content, brought to you courtesy of Springer Nature Customer Service Center GmbH ("Springer Nature").

Springer Nature supports a reasonable amount of sharing of research papers by authors, subscribers and authorised users ("Users"), for small-scale personal, non-commercial use provided that all copyright, trade and service marks and other proprietary notices are maintained. By accessing, sharing, receiving or otherwise using the Springer Nature journal content you agree to these terms of use ("Terms"). For these purposes, Springer Nature considers academic use (by researchers and students) to be non-commercial.

These Terms are supplementary and will apply in addition to any applicable website terms and conditions, a relevant site licence or a personal subscription. These Terms will prevail over any conflict or ambiguity with regards to the relevant terms, a site licence or a personal subscription (to the extent of the conflict or ambiguity only). For Creative Commons-licensed articles, the terms of the Creative Commons license used will apply.

We collect and use personal data to provide access to the Springer Nature journal content. We may also use these personal data internally within ResearchGate and Springer Nature and as agreed share it, in an anonymised way, for purposes of tracking, analysis and reporting. We will not otherwise disclose your personal data outside the ResearchGate or the Springer Nature group of companies unless we have your permission as detailed in the Privacy Policy.

While Users may use the Springer Nature journal content for small scale, personal non-commercial use, it is important to note that Users may not:

1. use such content for the purpose of providing other users with access on a regular or large scale basis or as a means to circumvent access control;
2. use such content where to do so would be considered a criminal or statutory offence in any jurisdiction, or gives rise to civil liability, or is otherwise unlawful;
3. falsely or misleadingly imply or suggest endorsement, approval , sponsorship, or association unless explicitly agreed to by Springer Nature in writing;
4. use bots or other automated methods to access the content or redirect messages
5. override any security feature or exclusionary protocol; or
6. share the content in order to create substitute for Springer Nature products or services or a systematic database of Springer Nature journal content.

In line with the restriction against commercial use, Springer Nature does not permit the creation of a product or service that creates revenue, royalties, rent or income from our content or its inclusion as part of a paid for service or for other commercial gain. Springer Nature journal content cannot be used for inter-library loans and librarians may not upload Springer Nature journal content on a large scale into their, or any other, institutional repository.

These terms of use are reviewed regularly and may be amended at any time. Springer Nature is not obligated to publish any information or content on this website and may remove it or features or functionality at our sole discretion, at any time with or without notice. Springer Nature may revoke this licence to you at any time and remove access to any copies of the Springer Nature journal content which have been saved.

To the fullest extent permitted by law, Springer Nature makes no warranties, representations or guarantees to Users, either express or implied with respect to the Springer nature journal content and all parties disclaim and waive any implied warranties or warranties imposed by law, including merchantability or fitness for any particular purpose.

Please note that these rights do not automatically extend to content, data or other material published by Springer Nature that may be licensed from third parties.

If you would like to use or distribute our Springer Nature journal content to a wider audience or on a regular basis or in any other manner not expressly permitted by these Terms, please contact Springer Nature at

onlineservice@springernature.com