# An Empirical Study About the Instability and Uncertainty of Non-functional Requirements

Luiz Viviani[1], Eduardo Guerra[2], Jorge Melegati[2(✉)], and Xiaofeng Wang[2]

[1] Institute for Technological Research, São Paulo, Brazil
[2] Free University of Bozen-Bolzano, Bolzano, Italy
{eduardo.guerra,jorge.melegati,xiaofeng.wang}@unibz.it

**Abstract.** Managing non-functional requirements (NFRs) has been a challenge in software development for many years. These requirements are typically used to make important architectural decisions early in the project, which can be problematic if they are uncertain or unstable. When this uncertainty is not considered when designing the software architecture, changes are often costly and sometimes even unfeasible. Some empirical studies on the subject have already been carried out, but few have focused on the perspective of professionals with extensive experience on the changes and uncertainties of NFRs. This work aims to expand the understanding about the management, clarity and validation of NFRs to fill this gap in the literature. To achieve this goal, a survey was carried out with professionals to find out how NFRs were managed and validated. For the research design, instead of generic questions, the questionnaire focused on some specific types of NFRs to induce participants to recall and report concrete situations. As a result, 40 valid responses were obtained, most from professionals with more than 10 years of experience. The results reveal that a significant number of NFRs were defined after the delivery of software increments (more than 30%) and that revision and change occurred in about a third of the NFRs. Hence, this study presents evidence that NFRs, as the functional ones, can also be uncertain and change frequently, requiring agile approaches and techniques to evolve the software architecture to consider this uncertainty.

**Keywords:** Non-functional requirements · quality attributes · software maintenance · software evolution · requirements engineering · empirical study

## 1 Introduction

Non-functional requirements (NFRs) are considered an important factor in software architecture decisions. However, the most important architectural decisions are usually made in the beginning of the project, even on agile projects [38], when some NFRs are uncertain. Since changes in software architecture are often expensive when its structure is not prepared to handle them, a late definition of these requirements might represent a big risk for the project [24].


C. J. Stettina et al. (Eds.): XP 2023, LNBIP 475, pp. 77–93, 2023.
https://doi.org/10.1007/978-3-031-33976-9_6

The uncertainty of functional requirements (FRs) is handled by agile methods by using techniques that allow the code to be more easily changed, such as TDD and refactoring [17]. During the early phases of software development, it is common to see FRs being prioritized while NFRs are ignored or neglected until later stages [6]. However, agile techniques, such as TDD and refactoring, might not be enough for changes in NFRs, which can crosscut functionalities and affect the entire system. Changing or introducing an NFR that is not compatible with the current architecture might turn a software project into a failure [24].

Most NFRs are often handled ad-hoc during system testing, and software engineers pay more attention to functional requirements that satisfies business needs [29]. This neglect or late focus might affect the final product, as both FRs and NFRs are necessary for the systems' success [35]. Instability, uncertainty and shallow focus on NFRs can result in high long-term maintenance costs [9]. Although the relevance of NFRs is widely accepted, the discourse and approaches on how to deal with them are still dominated by how to differentiate them from functional requirements [11,16]. Although more effort has been invested in satisfying NFRs [5], it seems that there is still an uneven emphasis on the importance given to the system's features and its quality requirements [15]. Understanding when NFRs are defined and how they change during the project can provide valuable evidence to guide software development, in particular, how the solution design and architecture accommodate these changes.

Some empirical studies on NFRs are present in the literature, but few focusing on professionals with large experience and high involvement in NFR management. The studies define or evaluate artifacts for the documentation of NFRs [32], compare methodologies and practices of traditional and agile requirements engineering [1] or even propose new artifacts or automation in document generation [30,32]. Other studies [20,24,26,34] aimed at identifying flaws in NFRs definition and management, but they do not discuss the validation of these NFRs, nor the impacts and difficulties caused by changes during development.

This study aims to gather evidence on the late definition of NFRs, either due to a lack of identification at the project start or due to later changes. That evidence highlights the need for more agile approaches to deal with changes and late identification of NFRs, since the ones used to handle changes in the functional requirements do not consider that it might have a huge impact in the software architecture. To achieve this goal, we conducted a survey with professionals with experience managing NFRs. The questionnaire focused on specific NFR types, stimulating participants to base their answers on concrete situations rather than on general impressions. As a result, evidence was identified of a relevant number of NFRs being defined after the start of the development and changed in late-stages of development.

## 2 Background and Related Work

NFRs refer to system restrictions and the quality of their development and implementation, which is always related and connected to FRs [28]. The IEEE defines

a non-functional requirement as "a software requirement that describes not what the software will do, but how the software will do it" [19]. In other words, NFRs are related to quality standards, as they describe the characteristics and needs of software so that it is efficient and fits its purpose [22]. The International Organization for Standardization (ISO) defines software quality as the level of satisfaction with the stated and implied needs of the user [18]. Software quality can be divided into two views: the first being that of "meeting specifications" and the second being "satisfying the customer", the latter means satisfying the customer's usage and consumption expectations, regardless of any measurable characteristic [18]. NFRs can be related to both observable attributes, such as performance, availability, and reliability of the system, as well as relative internal characteristics, for example, maintainability and portability [14,16].

Since NFRs express objectives that may conflict with each other, these requirements must be described as flexible goals. These conflicting objectives lead to a need of refinement and negotiation to reach acceptable solutions [13]. According to Berntsson et al. [9], the perception of the importance and purpose of an NFR may differ according to the professional's role in software development.

Several studies present the importance of NFRs to software project success. For instance, when NFRs are incorrectly managed, software projects have a failure rate of 60% or more [6]. NFRs play a critical role during software life cycle, and if defined or managed incorrectly, affect software maintainability, significantly impacting deadlines and costs [13]. NFRs generally influence the system architecture more than FRs [4,31], being key to ensuring a proper architecture.

Albeit their importance, NFRs receive less attention than their functional counterpart, causing customer dissatisfaction and rework and impacting time and cost [34]. This issue happens especially in projects employing agile methods [30], given these practices emphasis on functional aspects and lesser in documentation. The neglection of NFRs is a known issue for agile requirements engineering [20]. This issue is linked to agile methods' main characteristics of accepting and absorbing changes during the project and a lack of information at the beginning of the project for a proper upfront architecture [24]. However, other agile approaches such as Scaled Agile Framework (SAFe) and Disciplined Agile Delivery (DAD) have steps and procedures that aim to anticipate and mitigate the risks associated with RNFs and their volatility [2,27].

Given the possibility of architectural changes, early decisions on architecture in agile projects are encouraged [3,38,40]. This early definition contrasts with the recurring changes in agile projects, especially in the NFRs, which mostly cause impacts or revision in the architecture [24]. For instance, Ameller et. al [4] reinforce the importance of continuous management of the NFRs, stating that the list of NFRs of the project could never be considered complete even after the completion of development tasks. According to this work, this list should be evolved and negotiated during all project development and maintenance phases. The emphasis on continuously managing NFRs, reviewing and adapting the architecture to cope with possible impacts can be identified in the Sprint Zero of the

DAD [2] approach and in the Architectural Runway phases of the SAFe framework [27], as well as all necessary adjustments for the proper testing strategy for these requirements [2].

Some studies present evidence of the awareness about the importance of NFRs, but without adequate focus on their management [10]. Other studies evidence that lower importance is given to NFRs compared to FRs during software development [36], which explains the significant deficits in the management of NFRs, and how they are treated and prioritized. Only 40% of the NFRs are measurably specified [4,31], which makes hard its verification and assessment. The unified management of FRs and NFRs is defended by some studies, e.g., [8], which describe them as orthogonal. Consequently, architectural decisions that accommodate an FR may conflict with an NFR or vice versa.

## 3 Research Design

There is a lack of studies in the literature on the stability of NFRs, in particular, on when and how often they are defined or modified. Thus, as the objective of this study, we collected evidence from real projects on the definition and possible modification of NFRs. For example, we tried to understand in which project steps NFRs are defined and if they are modified in stages that lead to greater impact. To guide the research, we proposed the following research questions:

**RQ1** - How are NFRs defined in the software development process?
**RQ2** - How do NFRs change and are updated during a software project?
**RQ3** - How much does the approach for handling NFRs vary based on its type?

### 3.1 Methodology

To answer the research questions, we conducted a survey with professionals consisting of questions about the occurrence of specific types of requirements in real projects. The survey was designed to be answered by professionals with experience managing NFRs in software development projects. When answering the questions, participants should report if they recall the presence of a particular type of requirement in a project they had participated. In the case of a positive answer, a series of questions about this occurrence is presented, focusing on understanding when it had been specified, if it had changed, and why it changed.

As a first validation step, a first version of the questionnaire was created and reviewed by the authors and other invited collaborators. The survey was refined and sent to five professionals as a pilot study. Based on feedback from respondents, the demographic questions were improved, new options were added, and examples of NFRs were used to make them easier to understand. The result was the final version of the survey applied in the study. The participants from the pilot study were invited to answer the questionnaire again.

As a dissemination strategy, the authors sent invitations for participation to personal contacts, to software development email discussion groups and shared them on social media. It was intended for more experienced professionals and

that the estimated time to respond was 20 min, but not many spontaneous responses were received. Most participants were professionals with the target audience's profile, recruited through a direct invitation from one of the authors. The described approach for convenience sampling aligns with some guidelines from Rainer and Wohlin [33] to recruit credible participants.

We acknowledge that our sampling approach does not provide a representative sample (c.f. [7]). However, our goal could be stated as exploratory, i.e., looking for evidence of problems regarding NFR management and not to describe, for instance, how prevalent these problems are. We read and interpreted the answers to the open questions considering the answers given to the multiple choice questions, searching for any relevant complementary information.

### 3.2 Survey Design

Kitchenham and Pfleeger [23] suggest that the survey design must relate to research objectives so that the data obtained and the analysis can answer the questions defined for the study. We adopted a descriptive research design to conduct a retrospective study, where participants report past events and circumstances to support the understanding and explanation of current phenomena [23].

The questionnaire is available online[1] and was organized into eleven steps: (1) explanation of the research goals, presentation of the information about data confidentiality, and request of participant's consent; (2) presentation of the NFR concept adopted in the study and a request to choose a project in which the respondent participated as a reference for the following answers; (3) questions about participant background and the company in which the project used as a reference took place; (4) questions about the adopted practices for requirements engineering; (5–10) for each particular NFR type, a question regarding the existence of that particular type, and, in case of a positive answer, additional questions about it; (11) an optional open question about how NFR uncertainty was handled in the reference project.

The second step was included to make sure that the participant chooses a project as a reference for gathering data about a concrete experience. The participant can only proceed after confirming that the choice was made. To avoid misconceptions about the term "non-functional requirement", a short description was presented to the participant to agree for continuing.

In the third step, about the company, we asked the business segment and the number of employees. About the participants, the survey asked which role they played in the last five years (being able to select more than one option), years of experience in software development, and self-evaluation about the knowledge on NFRs. About the self-evaluation, five levels were defined (novice, advanced beginner, competent, proficient, and expert) with a description describing the expected expertise and knowledge for each. The fourth step assessed some information about how the reference project handled NFRs. The participant answered

[1] https://www.jmelegati.com/files/Survey-NFRs.pdf.

about the level of influence different roles have in defining NFRs, and how often these requirements were described in a more quantitative and measurable way or using a more qualitative and abstract approach.

In steps 5–10, each section focused on a particular NFR type. The decision to ask about specific NFRs was for the participants to recall a concrete past experience in which they could answer the questions more precisely. Limiting to six types enabled the exploration of different types of NFR without doing the survey very long. The chosen NFRs were: response time, number of simultaneous requests, scalability, flexibility, portability, and robustness. For each type, a definition and several concrete examples were provided to avoid misunderstandings.

If the respondent confirmed the presence of that type of NFR in the reference project, the following multiple-choice questions were presented: (a) approach used for NFR elicitation; (b) during each software development activity it was elicited; (c) if it was reviewed or modified during the project; (d) in case of a positive answer in the previous question, the reason it was reviewed or modified; (e) the current adherence of the NFR to the current project state. Finalizing each step, an optional open question asks details about the reported NFR.

The choice of these six specific types of NFRs can be considered a decision in the study scope and was motivated by the authors' individual experiences in industry in the field of software architecture. In the discussions conducted during the survey design, they pointed out that they had observed recurrent changes and instability on these types of NFRs. We recognize that this choice might introduce a bias in the study and we understand that the results should be interpreted as restricted to them. However, the six NFR types represent different kinds of quality attributes and cover relevant types of NFRs. Future surveys might replicate this study with other NFRs and compare the results.

## 4 Results

The survey received 40 answers. The pilot was executed in December 2021, and the survey received responses from April to June 2022. The following sections describe the obtained results.

### 4.1 Participant and Company Profile

Among the 40 professionals who answered the survey, more than 50% said they had worked in the last five years as a software architect or software engineer, and almost 40% as a developer. Other roles receiving a significant number of answers were researcher (20%), product manager (17%), and requirements analyst (17%).

The respondents fit the target profile based on the answers assessing their professional experience. Figure 1 presents the distribution of the number of participants according to their years of experience and their self-evaluation of their expertise in handling NFRs in projects. Most answers came from experienced professionals in the field.

The results revealed that projects used as a reference were developed in companies from diverse segments, emphasizing the financial and e-commerce

| NFR Experience | 1 to 3 years | 4 to 6 years | 7 to 9 years | 10 to 12 years | 13 to 15 years | 15 years or more |
|---|---|---|---|---|---|---|
| Novice | 0 | 1 | 0 | 0 | 0 | 0 |
| Advanced Beginner | 0 | 0 | 0 | 0 | 0 | 1 |
| Intermediate | 1 | 2 | 1 | 5 | 1 | 6 |
| Proficient | 0 | 0 | 2 | 1 | 4 | 6 |
| Specialist | 0 | 0 | 1 | 1 | 0 | 7 |

**Fig. 1.** Number of participants by years of experience and self-evaluated knowledge in handling NFRs

branches, which together represent 43% of the companies. However, we also received answers from 16 other segments, which received from one to three answers (several informed in the "Other" option). About the company size, the ones with more than 5000 employees add up to more than 56%. Since companies with a range from 500 to 2000 employees were declared by 18%.

Figure 2 presents the distribution of the answers about the influence of different roles on NFRs definition. Architects, product managers, and developers are among the roles with more significant influence. On the other hand, the ones less influential were director/c-level, tester/QA, and final users. In the question in which the respondents needed to classify from 1 to 5 how often they adopt a qualitative versus a quantitative approach for NFR specification (where one was always qualitative and five was always quantitative), the result was the following distribution: 1 (10%), 2 (40%), 3 (23%), 4 (23%), and 5 (4%).

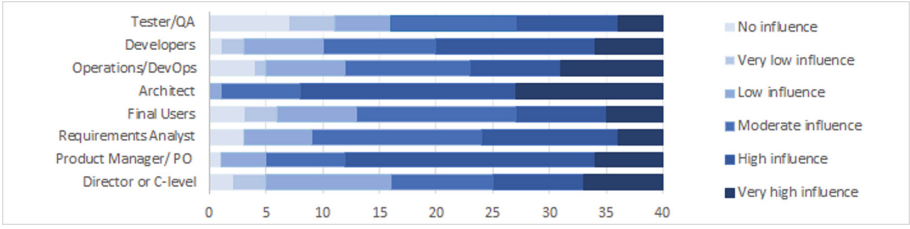


**Fig. 2.** Influence of Different Roles on NFRs

### 4.2 Answers About NFRs

As justified in the survey design, to obtain answers based on concrete situations from real projects, the questions concentrated on six types of NFRs. Figure 3 presents the percentage of answers reported to have each NFR type present in the reference project.

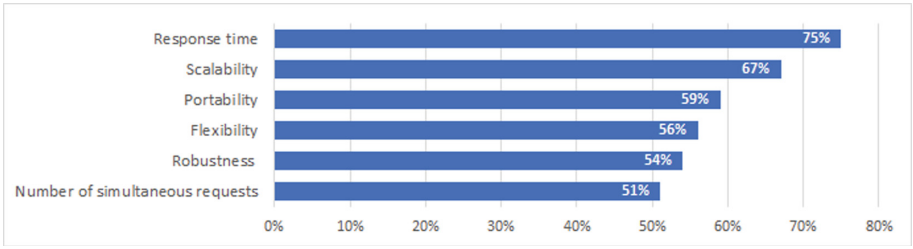


**Fig. 3.** Proportion of the projects in which the type of NFR was declared

For each NFR declared as present in the reference project, the survey asks for information about the method used for elicitation, time of definition, constancy during development, possible change motivators, and validation of project adherence to the defined NFR. A general analysis of the NFRs present in the study presents relevant information about the treatment and management of the NFRs. To perform this analysis, each requirement reported independent was considered of its type, which resulted in 144 instances. To answer RQ1 and RQ2, we considered these 144 instances together, and for RQ3 we compare the differences between them.

**Regarding the Elicitation Approach, as Shown in Figure 4,** the elicitation with stakeholders was the most mentioned (63%), which is a positive point since user involvement is among the main success factors for projects [21]. The reference from other systems (39%), restrictions imposed by legacy systems (33%), and regulation restrictions (28%) were also mentioned with significant frequency. The second most cited approach was "based on professional experience" (47%), even if systematically, this approach may be less suitable for requirements such as response time and flexibility, as they would be based on experience, with possible incompatibility with the actual requirements of the system. Other answers, such as metrics collected in production (33%), problems found on the project (19%), and complaints from users (32%), reveal that the elicitation was performed in later stages.

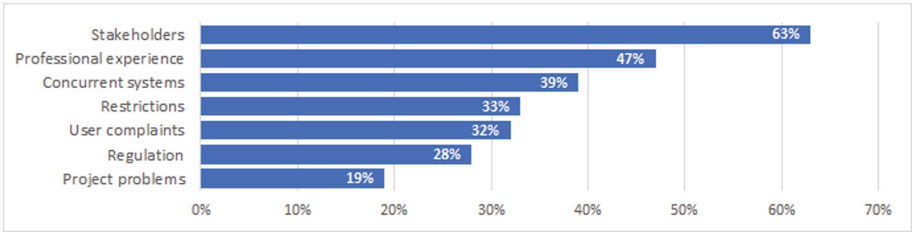


**Fig. 4.** NFRs elicitation approach

Figure 5 presents the answers during each activity the participants declared that the NFR was defined. As expected, in most cases, the requirement was elicited in the initial stages of design and analysis, however there are also answers reporting moments after the software delivery.

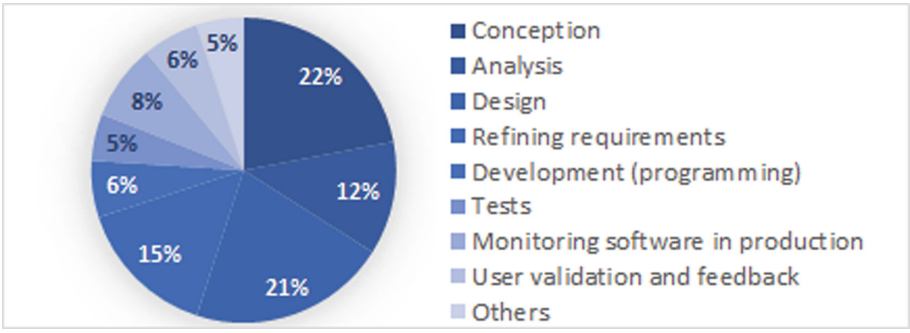


**Fig. 5.** Activities in which NFRs were defined

When asked if the requirement changed or was reviewed during the project, 25% answered that it was reevaluated but not modified and 32% that it was modified. Figure 6 presents the distribution of the answers about the drivers of change in NFRs, in the instances in which a review or change was reported in the previous question. In 87% of the cases, more than one reason was reported. With a higher percentage, there were options revealing that requirement changes were a reaction to something that happened in the system, such as changes in architecture, behavior or tests in production. The options "technical evaluation" and "regular review of requirements" represent more systematic approaches and received a significant number of answers (25% and 24%, respectively).

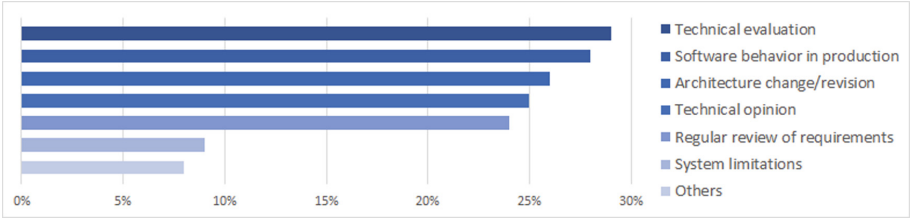


**Fig. 6.** Driver of changes in NFRs

The final question about NFRs was if the current requirement state is according to the system needs. To this question, 25% said that the requirement is outdated, and 9% did not know.

# 5 Discussion

## 5.1 RQ1 - How are Non-functional Requirements Defined in the Software Development Process?

Figure 5 presents the distribution of software development activities where the NFRs were defined. The results show that 22% were determined during development, and, for around 19%, their definition was reported to have happened in activities typical in later stages, such as based on user feedback, monitoring software in production, or testing. This result confirms that some requirements are neglected during the initial phases of the project, being only introduced when some event in the project points to that need. The literature reinforces the importance of this definition at the beginning of the project for technology selection and choice of patterns [25]. One participant reported for response time: *"the requirement was not idealized in the conception or design, a performance was expected, and during the tests we realized the need to include the requirement"*.

Some participants mentioned bad consequences of a late identification of requirements. For portability, one respondent mentioned that *"due to its late entry, it was adjusted and limited for having only a few capabilities"*. This limitation on implementing an NFR identified later was also reported regarding flexibility. The current literature supports this result [1,34], highlighting possible negative consequences of NFR late definition.

**Finding 1**: *A significant number, almost 1/3, of NFRs are not identified at the beginning of the project.*

The results show a great diversity of methods and techniques to elicit NFRs, often in combination. The fact that in almost half (47%) of the cases professional experience was reported as the approach used to define the requirement indicates the absence of a systematic approach for NFR elicitation. The number of answers that said the identification was based on problems found or user complaints also evidence that this later identification brought bad consequences to the project. For example, a respondent said *"problems that emerge only at deploy time on the client"*. The work of Lauesen [26] points out this lack of fulfillment of the requirements as a cause for project failures.

**Finding 2**: *Almost 1/3 of NFRs are defined in response to problems found on the project.*

This elicitation deficit is also reflected in the number of NFRs that changed later or are outdated to the system's needs in production. According to the results obtained, 34% of the requirements reported are not up to date with the system in production, or its consistency is unknown. That is evidence that is common to have NFRs neglected until the production stage. Indeed, a recent study [37] pointed out not-clear and not-measurable NFRs as a problem in some agile processes.

This inconsistency with the system in production can also be explained by unnecessary NFRs included at the beginning of the project, which were ignored for not being important, but were not removed from the system requirements. One participant stated that sometimes that is noticed only *"when the technical proof of its infeasibility or that it is something unnecessary"*. In the same testimony, it was also mentioned that *"the costs presented to end users often make them rethink whether something is really necessary"*.

**Finding 3**: *More than 1/3 of NFRs are identified in the project are outdated or have an unknown state to the system in production.*

The answers about the qualitative versus quantitative approach for defining NFRs might indicate one of the reasons for this deficit, revealing that a more qualitative and non-objective way to define the requirements is frequently used. That was confirmed by a developer whose answer to the survey stated that the specification is deficient *"often not knowing how to elaborate the artifacts as well as the structure of the requirement text without objectivity"*. Another participant gave some examples of this lack of objectivity, mentioning *"good response time"*, and *"low cost for evolution"*.

### 5.2 RQ2 - How Non-functional Requirements Change and are Updated During a Software Project?

The responses to the survey revealed that around one-third of the NFRs changed during the project. Considering the approach used for the requirement elicitation described in the previous section, there are factors that could contribute to changes in the NFRs due to flaws in its specification, like the usage of non-systematic approaches and a qualitative and non-precise specification. Lack of prioritization of NFRs is associated with rework and customer complaints [34].

**Finding 4**: *The difficulty in dealing with the uncertainty and volatility of NFRs results in reactive actions.*

The reasons for changes or revisions of the NFRs are diverse, but there is a strong relationship between the changes and the absence of a deeper analysis of the real needs of the software. Absence or ineffective processes of continuous management of NFRs, and strategies to anticipate and accommodate possible changes [2,27] result in reactive actions and with impacts to the project [12]. Of the five most cited factors, only two result from a systematic assessment of requirements, while the other three result from the demand for changes in the architecture and behavior of the system in production. In addition to NFRs' late elicitation, these results about the motivation of changes also evidence a late validation of NFRs. Around 54% of the mentioned modifications occurred only after analyzing the behavior or tests of the system in production, which can add significant costs [12], and 35% required or were caused by architecture review or system limitation.

**Finding 5**: *A significant occurrence of changes on NFRs occurred after analyzing the behavior of the system in production.*

Several statements spontaneously written by the respondents point to the impact of the changes of NFRs on the solution's design and architecture, even not having any question about it. As explicitly stated by one of them, changes in NFRs *"fundamentally change the design and architecture of a system"*. One respondent mentioned that the lack of visibility of the flexibility NFRs led to a revision of the architecture to support a higher number of customers. Another one mentioned that the system did not scale as it should, but it was in production without many complaints from customers. About system reliability, one mentioned that *"due to the addition of new features and new error situations, there came the need to change the requirements"*. Because of that, the architecture was modified, allowing more monitoring and failure recovery mechanisms. Since architectural decisions are usually made in the beginning of the project [40] and their implementation can be so interwoven in the code, changes might not be feasible [38].

In the other direction, changes in the architecture might also cause a review in NFRs. One respondent reported that after a change in the architecture *"portability requirements (servers, platforms and database) have been revised"*. The cost to implement the NFR was also mentioned by another participant as relevant to reconsider the requirement, revealing that some NFRs might be abandoned due to trade-off analysis that give them up in exchange of some other factor.

**Finding 6**: *Absence of a testing strategy makes it difficult to accommodate and validate changes that cross-impact NFRs and architecture.*

The close relationship between NFR and architecture results in managing and planning properly to avoid major impacts [24]. In this way, it is essential to pay attention to an adequate project testing strategy, anticipating and monitoring changes to mitigate them and be prepared to accommodate them [2].

### 5.3 RQ3 - How Much the Approach for Handling NFRs Vary Based on its Type?

For the previous research questions, we considered all the 144 reports from all six NFR types targeted in the survey. To answer this one, we analyzed the main differences between the answers given for each type. The first point to be highlighted is the number of participants reporting each requirement. As shown in Fig. 3, response time received the highest number of answers (75%), and the simultaneous number of requests received the lowest (51%).

Regarding elicitation approaches, the two most mentioned in all types of requirements were "elicitation with stakeholders" and "professional experience." Scalability and portability did not have any other elicitation method to be highlighted, but they were the ones with the higher number of answers that reported

the elicitation with stakeholders, respectively 76% and 69%. For response time and amount of simultaneous requests, which are easier to measure and observe than the others, the reference from concurrent systems (respectively 54% and 47%) and measurements (respectively 46% and 58%) were mentioned more. Complaints from users received a significantly higher number of occurrences for response time (68%), which was more than double the overall percentage. Finally, regulations and standards were mentioned more frequently for flexibility (38%) and restrictions from legacy systems for robustness (50%).

The following activities can be highlighted as being higher than the overall percentage: conception for portability (36%), analysis for scalability (20%), design for flexibility (29%) and robustness (30%), and refinement for the amount of simultaneous requests (26%). For response time, no activity alone received a high number of answers, but it received most answers in activities more typical of late project stages, such as development, tests, and user validation.

**Finding 7**: *NFR identification might happen for different reasons depending on its type.*

Regarding requirements changes, the NFR type that received less reviews or changes was robustness (60% reporting no changes) while on the other extreme were portability (36%) and amount of simultaneous requests (30%). Portability was the most common to be reviewed but not changed (41%) and amount of simultaneous requests the most changed one (45%).

The NFR types most affected by changes or revisions in the architecture are scalability (45%), portability (44%) and flexibility (44%). While for response time and robustness, there is not a specific driver that stands out, for the number of simultaneous requests, the software behavior in production (41%) and experiments/evaluations made by the technical team (47%) were the main drivers.

**Finding 8**: *The most common drivers for change might be different for each kind of NFR.*

Although the survey revealed some common points to handling NFRs, each type of requirement has its particularities. Based on that, suitable techniques for elicitation and evaluating its suitability during the project should be employed based on the target requirement. This result is aligned with another study [34] that state that there no unique solution that will solve all requirements engineering needs.

### 5.4 Threats to Validity

Empirical studies face certain validity threats which can be of construct, internal validity, external validity and reliability [39]. Below, we describe the threats for this study according to this classification and how we mitigated them.

**Construct validity** refers to ensuring correct operational measures for the concepts (constructs) under study [39]. Since most of the questions regarded to the description of real experiences from the respondents, threats in this regard are limited. However, there is still a threat that respondents will not understand the responses in the same way as the authors. To mitigate this issue, we carried out a pilot to get feedback and perform some adjustments. Also regarding this threat, the survey ask participants to answer based on a specific project, instead of reporting a general experience.

**Internal validity** is associated with causal inference and happens when it is believed that a certain condition is a consequence from a second condition when, in reality, they are consequence of a third condition not considered in the study [39]. Since this study is exploratory and it is not claiming the existence of causal-effect relationships, this threat is reduced.

**External validity** is about generalizing the results to a large population [39]. As discussed in Sect. 3, this study has an exploratory goal and does not aim to have a generalizable description of how NFRs are handled. The goal was to show evidence of potential issues. Another point reinforcing external validity was the fact that the study was directed to professionals with a specific profile without previous definitions or suggestions, because the survey does not influence the project chosen by professionals at the time of the answer. Besides that, as mentioned earlier, this study is restricted to six types of NFRs. We acknowledge this limited scope however, by focusing on limited facets, we were able to collect more precise answers from the respondents. Consequently, even though the results should be understood as restricted to these types, they suggest the value of further investigation of this research problem.

**Reliability** is related to the dependence of the data and results on the researchers who performed the study [39]. To improve this aspect, we defined a protocol, including a questionnaire and data analysis procedures. We also made public the questionnaire to allow the replication of the study.

## 6 Conclusions

NFRs are essential aspects impacting the success of a software project. Although some research focused on them, there was a lack of evidence related to changes and their late definition. This paper presents a survey with experienced professionals on how NFRs were elicited and modified in concrete software projects to fulfill this gap. We found a lack of a well-defined step for evaluating and reviewing the NFRs, and a prevalence of changes in NFRs even in the late stages of development. This evidence highlights the need for agile techniques to deal with changes in NFRs, which can significantly impact the architecture if this uncertainty is not considered to make it ready to change.

The limitations of this study include investigations of specific NFRs, chosen to obtain more precise answers.

This limitation gives space for future studies directed to other NFRs. The information obtained in the survey helped to understand existing issues in the

management of NFRs in real projects, it was out of its scope to find the reasons behind these issues. Future works could also delve into the relationship between subjectivity in the definition of NFRs with the changes and real need for NFRs in projects and investigate whether the project management model influences the management quality of non-functional requirements during software development.

## References


1. Alsaqaf, W., Daneva, M., Wieringa, R.: Quality requirements in large-scale distributed agile projects – a systematic literature review. In: Grünbacher, P., Perini, A. (eds.) REFSQ 2017. LNCS, vol. 10153, pp. 219–234. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-54045-0_17
2. Ambler, S., Lines, M.: Introduction to Disciplined Agile Delivery - Second Edition. Project Management Institute, July 2020
3. Ambler, S.W.: Agile modeling. Wiley, Nashville (2002)
4. Ameller, D., Ayala, C., Cabot, J., Franch, X.: How do software architects consider non-functional requirements: an exploratory study. In: IEEE (2012)
5. Ameller, D., Franch, X., Cabot, J.: Dealing with non-functional requirements in model-driven development (2010)
6. Bajpai, V., Gorthi, R.P.: On non-functional requirements: a survey (2012)
7. Baltes, S., Ralph, P.: Sampling in software engineering research: a critical review and guidelines (2022)
8. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice. SEI series in software engineering, Addison-Wesley Educational, Boston, MA (2003)
9. Berntsson-Svensson, R., Gorschek, T., Regnell, B.: Quality requirements in practice: an interview study in requirements engineering for embedded systems. In: REFSQ (2009)
10. Borg, A., Yong, A., Carlshamre, P., Sandahl, K.: The bad conscience of requirements engineering: an investigation in real-world treatment of non-functional requirements. In: Third Conference on Software Engineering Research and Practice in Sweden (SERPS 2003), Lund (2003)
11. Broy, M.: Rethinking functional requirements: A novel approach categorizing system and software requirements, pp. 155–187. John Wiley & Sons, Insc., September 2018
12. Budiardjo, E.K., Wibowo, W.C., et al.: Non-functional requirements (NFR) identification method using FR characters based on ISO/IEC 25023. Int. J. Adv. Comput. Sci. Appl. (2021)
13. Chung, L., Nixon, B., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Springer, New York (2012). https://doi.org/10.1007/978-1-4615-5269-7
14. Dörr, J., Kerkow, D., Koenig, T., Olsson, T., Suzuki, T.: Non-functional requirements in industry - three case studies adopting an experience-based NFR method. In: 13th IEEE International Conference on Requirements Engineering (RE'05) (2005)
15. Eckhardt, J., Vogelsang, A., Fernández, D.M.: Are "non-functional" requirements really non-functional? In: ACM (2016)
16. Glinz, M.: On non-functional requirements. In: 15th IEEE International Requirements Engineering Conference. IEEE, October 2007

17. Guerra, E., Aniche, M.: Achieving quality on software design through test-driven development. Elsevier (2016)
18. Hoyer, R., Hoyer, B.: What is quality? Quality Progress (2001)
19. IEEE: Standard glossary of software engineering terminology. Std 610.12 (1990)
20. Jarzebowicz, A., Weichbroth, P.: A qualitative study on non-functional requirements in agile software development. IEEE Access **9**, 40458–40475 (2021)
21. Johnson, J., Mulder, H.: Factors of succes 2015 (2015)
22. Kirner, T.G., Davis, A.M.: Nonfunctional requirements of real-time systems. In: Advances in Computers (1996)
23. Kitchenham, B.A., Pfleeger, S.L.: Principles of survey research part 2. In: ACM SIGSOFT Software Engineering Notes (2002)
24. Knauss, E., Liebel, G., Schneider, K., Horkoff, J., Kasauli, R.: Quality requirements in agile as a knowledge management problem: more than just-in-time (2017)
25. Kumar, D., Kumar, A., Singh, L.: Non-functional requirements elicitation in agile base models. In: Webology (2022)
26. Lauesen, S.: IT project failures, causes and cures. In: IEEE Access (2020)
27. Leffingwell, D.: SAFe 4.5 reference guide. Addison-Wesley Educational, Boston, MA, 2 edn., July 2018
28. Mijanur Rahman, M., Ripon, S.: Elicitation and modeling non-functional requirements - A POS Case Study. arXiv e-prints (2014)
29. Nguyen, Q.L.: Non-functional requirements analysis modeling for software product lines. In: IEEE (2009)
30. Oriol, M., et al.: Data-driven and tool-supported elicitation of quality requirements in agile companies (2020)
31. Pohl, K., Rupp, C.: Requirements engineering fundamentals. Rocky Nook (2015)
32. Rahy, S., Bass, J.M.: Managing non-functional requirements in agile software development. IET Softw. **16**(1), 60–72 (2021)
33. Rainer, A., Wohlin, C.: Recruiting credible participants for field studies in software engineering research. Inf. Softw. Technol. **151**, 107002 (2022)
34. Sherif, E., Helmy, W., Galal-Edeen, G.H.: Managing non-functional requirements in agile software development. In: Gervasi, O., Murgante, B., Hendrix, E.M.T., Taniar, D., Apduhan, B.O. (eds.) Computational Science and Its Applications – ICCSA 2022. ICCSA 2022. LNCS, vol. 13376, pp 205–216. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-10450-3_16
35. Slankas, J., Williams, L.A.: Automated extraction of non-functional requirements in available documentation. In: 1st International Workshop on Natural Language Analysis in Software Engineering (NaturaLiSE) (2013)
36. Svensson, R.B., et al.: Prioritization of quality requirements: State of practice in eleven companies. In: IEEE (2011)
37. Wagner, S., Fernández, D.M., Felderer, M., Kalinowski, M.: Requirements engineering practice and problems in agile projects: results from an survey (2016)
38. Waterman, M., Noble, J., Allan, G.: How much up-front? a grounded theory of agile architecture (2015)
39. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Planning. In: Experimentation in Software Engineering, vol. 9783642290, pp. 89–116. Springer, Berlin, Heidelberg (2012)
40. Yang, C., Liang, P., Avgeriou, P.: A systematic study on the combination of software architecture and agile development. J. Syst. Softw. **11**, 157–184 (2016)