Physical Education, Health and Social Sciences

https://journal-of-social-education.org

E-ISSN: 2958-5996

P-ISSN: 2958-5988

Analysis towards Non-functional Requirements: Current Issues and Approaches: (A Survey)

Fareed Ullah [1]

[1]Institute of Computing and Information Technology, Gomal University, Dera Ismail Khan, Pakistan Email: fareedkamran44@yahoo.com
Corresponding Author: fareedkamran44@yahoo.com

DOI: https://doi.org/10.63163/jpehss.v3i4.979

Abstract:
Non-functional requirements (NFRs) are crucial for software quality and project success, but in reality, they are frequently overlooked or poorly handled. The existing problems and difficulties with NFR elicitation, documentation, prioritization, and management in Pakistani software companies are examined in this study using a survey-based methodology. Software specialists from various organizations were given standardized questionnaires to complete in order to gather data. Significant deficiencies are revealed by the data, such as poor customer interaction, little understanding of NFRs, a lack of supporting tools, poor documentation procedures, and insufficient prioritization strategies. These flaws often lead to higher development costs, longer development times, and lower-quality software. This study makes insightful recommendations based on the empirical results, including the usage of checklists, standardized templates, agile methods, enhanced communication channels, and early NFR validation. The study's findings provide empirical evidence to the limited regional research on non-functional requirements management and offer practitioners practical insights.

Keywords: Non-Functional Requirements, Requirements Collection, Requirements management, Software Engineering

Introduction

Background: The scope, quality and success of software systems are determined by software requirements engineering, which is a crucial stage in the software development life cycle. Functional requirements, which specify system behaviour, and non-functional requirements (NFRs), which specify quality aspects including performance, security, usability, reliability, and maintainability, are the two basic categories of requirements. Non-functional requirements govern how well a system carries out its intended functions and are crucial for user happiness, system resilience, and long-term maintainability, whereas functional requirements define what a system should accomplish. Non-functional requirements govern how well a system carries out its intended functions and are crucial for user happiness, system resilience and long-term maintainability, whereas functional requirements define what a system should accomplish.

Problem Statement

However, in reality a lot of software companies' priorities functional requirements, treating non-functional requirements as secondary or addressing them later in the development process. Ambiguity, architectural mismatches, performance bottlenecks and system failures are frequently caused by this ad hoc processing of NFRs. The issue is especially noticeable in developing nations like Pakistan, where software companies must deal with issues such a lack of specialized tools for

handling non-functional requirements, time restrictions, insufficient client interaction and limited experience in requirements engineering.

Research Gap

A large portion of the previous research is conceptual, tool-oriented, or carried out in settings outside of developing software markets, despite the literature's emphasis on the significance of non-functional needs. Studies that examine the understanding, elicitation, documentation, and management of non-functional requirements in actual industrial settings in Pakistan are limited. This gap hinders the availability of evidence-based recommendations for enhancing NFR procedures in regional software companies.

Objectives

This paper conducts a survey-based analysis and makes practical recommendations based on industry input in order to address the lack of empirical knowledge regarding non-functional requirements practices in Pakistani software organizations. This study's primary goals are to:

(i) examine the methods currently in use for managing non-functional requirements in software projects;

(ii) highlight the main issues and dangers connected to improper NFR handling;

(iii) propose workable solutions to enhance the elicitation, documentation, prioritization and management of non-functional requirements.

This paper structured as follow. Section II presents related work, section III describes the research methodology and findings, Section IV discuss recommendation and section V conclude the paper with future directions.

Related Work:

IEEE and software engineering journals about Non-Functional requirements issues have already published many research papers and challenges. Nowadays for any company or organization, identification of NFR for software development is a key job. Nonfunctional requirements play a dynamic role in the in the development of a successful software product and having an important concept in requirement engineering. Organizations usually handled nonfunctional requirements in an ad-hoc approach throughout the testing phase of software development such as consistency, usability, safety, security, performance, etc. If nonfunctional requirements not considered properly at the early level of software development, the complexity, consuming time will be increase and will be difficult to handle them later on [7].

Requirements engineering is the first phase in software development life cycle and it plays one of the most important and critical roles. Requirement document mainly contains both functional requirements and non-functional requirements. Non-functional requirements are significant to describe the properties and constraints of the system. Early identification of Non-functional requirement has direct impact on the system architecture and initial design decision. Requirements Engineering (RE) can be defined as "a set of activities for exploring, evaluating, documenting, consolidating, revising and adapting the objectives, capabilities, qualities, constraints and assumptions that the system-to-be should meet based on problems raised by the system-as-is and opportunities provided by new technologies" [8].

Managing Non-Functional Requirements (NFRs) in software projects is challenging, and projects that adopt Model-Driven Development (MDD) a format to write and implement software quickly, effectively and at minimum cost. Although several methods and techniques proposed to face this challenge. Non-functional requirements (NFRs) are one of the main targets of research in the Requirements Engineering community. Disregarding NFRs will usually provoke that the generated system does not completely satisfy some of the stakeholders' expectations represented

by NFRs. [9] Generally industry followed functional requirement as primarily focus and on other hand the significant concepts such as NFR are not pondered till the design and development stage which create some serious flaws. Irregularity and ambiguity in nonfunctional requirements often case the failure of the system [15]. In the sense that attempt nonfunctional requirement can help the getting of other nonfunctional requirements at a definite of functionality [6]. NFRs are often ignored and inadequately specified during software development [4]. Because of the significance and criticality of nonfunctional requirements, I find out the difficult of modeling nonfunctional requirements for Software Product Lines, which adds yet a further aspect of complication [17]. Building a shared understanding of non-functional requirements (NFRs) is a known but understudied challenge in requirements engineering, especially in organizations that adopt continuous software engineering (CSE) practices. During the peak of the COVID-19 pandemic, many CSE organizations obeyed with working remotely due to the imposed health restrictions and implemented business processes to facilitate team communication and productivity. In remote CSE organizations, managing NFRs becomes more challenging due to the limitations to team communication. While the previous research has identified the factors that lead to a lack of shared understanding of NFRs in CSE, we still have a significant gap in understanding how CSE organizations, particularly in remote work, build a shared understanding of NFRs [12]. From the perspective of quality and tradeoff, a design pattern analyzed for investigating its potential for handling nonfunctional and conflicting requirements. Requirement changing are an issue in software development life cycle [14].

NFRs are stretch crosswise the entire service-oriented system and or within separate services and cannot be assigned to one specific system /service artifact. It makes them often more complicated to manage than functional requirements [11]. In this practice; it is very significant to assess the quality of the RFP to make sure that fundamental user requirement written adequate. Especially, non-functional NFRs are significant since the system architecture importantly depends on the NFRs such as response time and security matters. It is a hard challenge to fixing the amount to which any specific software system fulfills such requirements and integrating such considerations into the software design process. Intra conflicts of the nonfunctional requirements and functional requirements and inter conflicts among nonfunctional requirements [13]. Changing requirements is an issue in the life cycle of software development, which often originates from an incomplete knowledge of the domain of interest. A metamodel which clearly captures non-functional requirements and their relations, and which is independent from any programming standard [5]. The modification of the nonfunctional requirements framework which allows for the finding of a set of system functionalities that optimally satisfied a given set of nonfunctional requirements. According to the existing available NFRs literature, most of the work focus on NFRs but it still needs more work.

I conduct a survey that what types of problems and challenges are in non-functional requirements elicitation at initial phase during development software. What types of risks associated with these if non-functional requirements not implemented at initial phase? the industry of software raising it insist for not only essential functionality but the need of NFRs too. Here we surveyed from different software houses through questionnaires that what are the challenges and obstacles about Non functional software requirements to our software houses which consequences the failure of software.

Research Methodology

Participants and sampling

This study examined non-functional requirements practices in Pakistani software companies using a survey-based research methodology. Software developers, system analysts, project managers, and quality assurance staff who actively participated in requirements engineering activities were among the participants. Convenience sampling was used to get responses from several organizations of various sizes based on participant availability and industry involvement.

Survey Tool

In accordance with earlier research on requirements engineering procedures and non-functional requirements, a structured questionnaire was created. The questionnaire was divided into four sections: the impact of NFRs on software projects, organizational views on the significance of NFRs, organizational procedures for managing NFRs, and difficulties in eliciting and prioritizing NFRs.A five-point Likert scale, ranging from "Strongly Disagree" to "Strongly Agree," was used to score the majority of closed-ended items.

Data collection and analysis

The survey circulated both manually and electronically. Descriptive statistical methods were used to analyses the gathered replies. To find trends and patterns, percentages were computed using the total number of valid answers for each question. To improve the interpretation of the results, graphical representations were employed.

Validity Threats

This study has significant restrictions. First, the survey's focus on Pakistani software companies may have limited its applicability. Second, because the information is self-reported, respondent bias may exist. Lastly, the sample size can be increased in subsequent research for more robust statistical validation, although it still supports exploratory analysis.

Results and Finding.

The survey results are presented and examined in this section in order to identify the main problems, difficulties and shortcomings associated with non-functional requirements management in Pakistani software companies. The results demonstrate how poor management of non-functional requirements impacts software quality, raises development costs and causes schedule delays in software projects.

Practices used by organization:

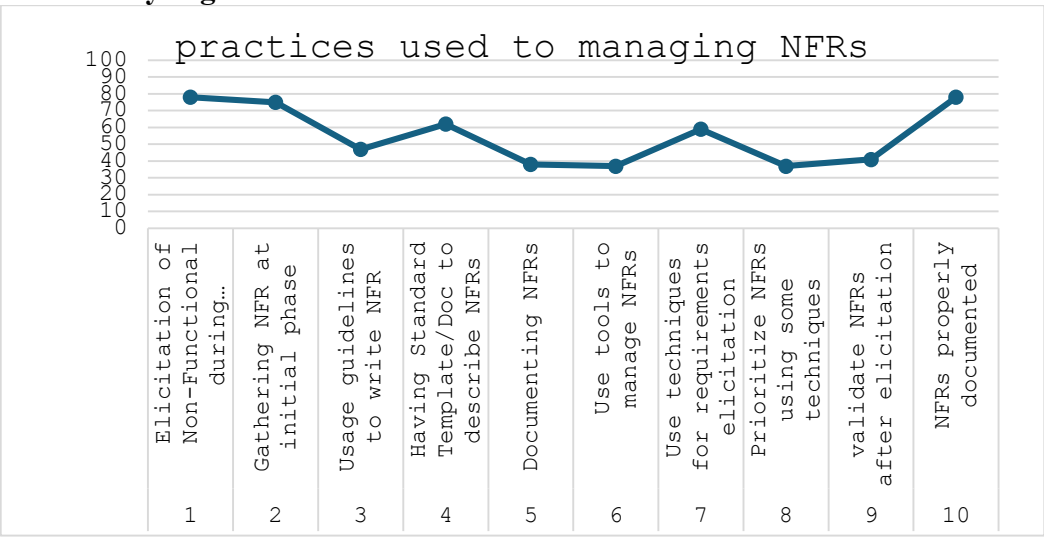

Figure 1. Practices used by organizations to manage non-functional requirements

The figure 1: show that the feedback comes from different organization using practices to manage NFRs is not satisfactory which lead to project failure.22% of organization do not Elicited NFRs at initial stage. Mostly organizations do not follow guidelines nor do they use standard template to manage NFRs. The figure show that 62% did not document NFRs nor do they use tools to manage. These figure show that we do not concentrate or not consider NFRs as primary requirements, which lead to project failure. If the above points do not cover it will increase the response time and end user will be compelled to wait until completion of the task and long response will decrease the efficiency and it impact on quality of software, its schedule and cost.

The findings indicate that in the early phases of software development, non-functional specifications are not consistently considered to be primary requirements. The lack of standardised templates, rules and tool support suggests that NFR management is primarily informal, which raises the risk of rework, schedule delays and quality degradation in subsequent stages of development.

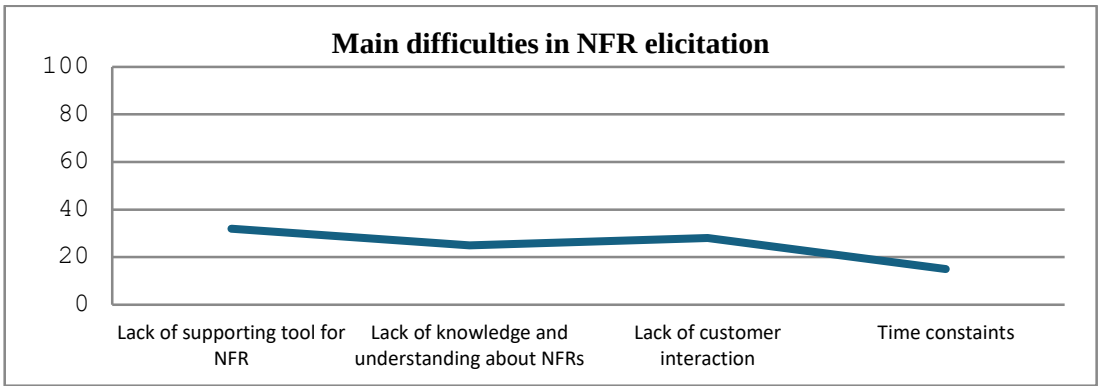

Figure 2.

In the figure 2 diagram show that our organization facing main difficulties in NFR elicitation are. 32 % contribution is lack of supporting tool for NFRs, 25% contribution is lack of knowledge, understanding about non-functional requirement, 28% is lack of customer interaction, and 15% contribution is time constraints out of 100 percent, which is not a good sign for a quality product. These results show that the main obstacles to NFR elicitation in organizations are a lack of supporting technologies and a lack of understanding of non-functional requirements. These problems are made worse by the lack of client interaction, which results in unclear or insufficient quality criteria. Similar difficulties have been documented in earlier research by Umar and Khan [7] and Ullah et al. [16], who discovered that poor elicitation techniques and a lack of knowledge of NFR ideas had a substantial impact on software quality and project success.

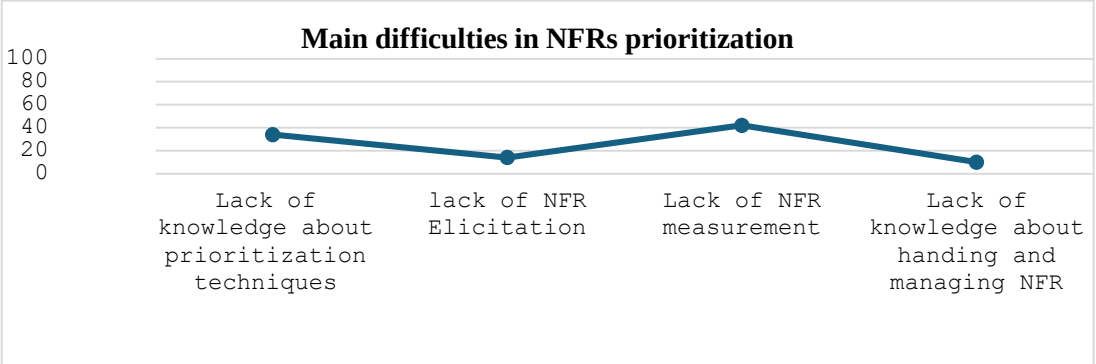

Figure 3.

In figure 3 show that during Non functional requirements prioritization, 81% organization facing difficulties during nonfunctional prioritization because in our survey report show that there is lack of knowledge about prioritization technique, lack of non-functional requirement elicitation, lack of knowledge about handling and managing nonfunctional requirements etc in our organization which causes difficulties in NFRs prioritization.

A significant flaw in the decision-making process during requirements engineering is highlighted by the large proportion of organizations that struggle to priorities non-functional requirements. Important quality characteristics like performance, security and dependability could be disregarded in the absence of efficient prioritization strategies. This result is consistent with past studies showing that inadequate NFR prioritization raises project risk and stakeholder dissatisfaction [7,15].

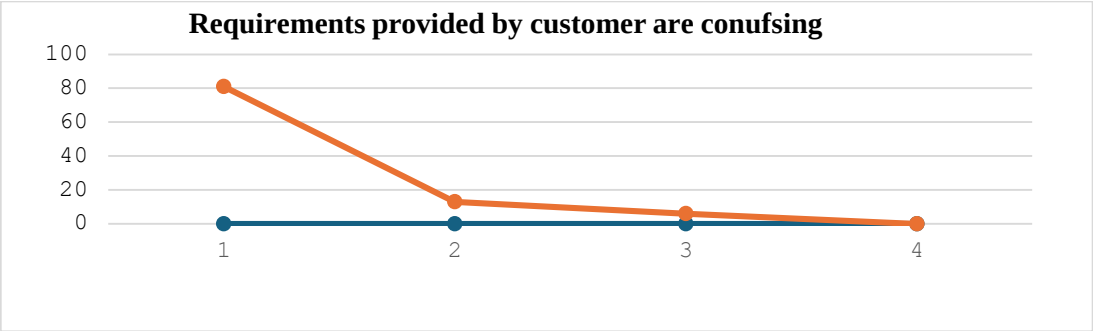

Figure 4

Figure 4 show that 81% organization, agree that the requirements provided by customer are confusing. Only 13% do not agree with that. In our survey report show that most of the customer not well known about non-functional requirements and are not well educated which causes ambiguous requirement.

These findings show that a major obstacle to efficient NFR management is confusing and ambiguous client expectations. This uncertainty frequently results from stakeholders' lack of technical expertise, which leads to a misunderstanding of quality expectations. As a result, software teams are compelled to make decisions that could have an adverse impact on system architecture and overall program quality.

- Some other results are also important in evaluating successful handling Non-functional requirements.
  - 96.87 % are agrees that NFRs play a significant role in the success of a software project and the 3.12 % did not answer.
  - Only 43.75% prioritize NFRs.
  - 40.62 % of organization use checklist for identification of NFRs.
  - 75.0 % of organization agrees that NFRs should mapped with FRs for better project management.
  - 34.37 % identify NFRs manually from the initial draft of gathering requirements.
  - 50 % of organization use guidelines to write NFRs.

Issues/ challenges.

The survey's findings point to a number of persistent difficulties in handling non-functional requirements. These difficulties fall into four main categories:

(i) low customer and developer knowledge and awareness of NFR concepts;

(ii) poor customer-developer communication leading to unclear requirements;

(iii)a lack of templates, tools and prioritization strategies;
(iv)time constraints and insufficient attention to NFRs in the early stages of development. These challenges significantly increase Project risk, cost overruns and negatively affect overall software quality.

Recommendation and Solution

Communication-Based Approaches

Effective management of non-functional requirements requires better customer-development team communication. During requirements elicitation, methods including prototyping, use cases, UML models, and agile approaches can assist stakeholders in expressing quality expectations in a clear and concise manner.

Process-Based approaches

Early in the software development process, non-functional requirements should be gathered and evaluated immediately. Traceability and consistency can be enhanced by mapping non-functional requirements with functional requirements. To reduce project risks, experienced developers or software architects should prioritise and oversee high-risk NFRs.

Tools and Documentation

To manage non-functional requirements, software companies should implement standardised templates, checklists and documentation rules. To cut down on unnecessary complexity and documentation costs, only the NFRs that are relevant to the particular system should be documented.

Awareness and Training

To increase developers, system analysts, and customers understanding on non-functional requirements, regular training sessions, workshops and awareness programs should be held. Increased awareness can greatly enhance NFR validation, prioritisation and elicitation procedures.

Conclusion and Recommendation for Future Work

The management of non-functional requirements (NFRs), which are essential for software quality and project success, presents considerable hurdles for Pakistani software companies. One of the main problems is a lack of experience in setting priorities, recording, managing, validating, and eliciting NFRs; this results in slower reaction times, decreased productivity, and compromised software quality, schedule, and cost. According to surveys, many firms do not have dedicated requirements engineering teams and instead rely on developers to gather requirements, which leads to subpar practices. Inadequate communication between customers and stakeholders makes these issues worse by leading to imprecise project scopes and ambiguous needs. To overcome these obstacles, it is advised to designate key developers to supervise important and high-risk needs, making sure that priority, validation, and documentation are all clear. Use cases, UML, agile approaches, and prototyping are some strategies that help improve requirement clarity and close communication gaps. Along with using application simulation technologies to improve communication between IT teams and business users, firms should also engage in training and seminars to increase awareness and comprehension of NFRs. Software houses can enhance their management of NFRs and achieve better software development results by implementing these tactics.

This study has some shortcomings despite its value. Due to its reliance on self-reported data and restriction to Pakistani software companies, the poll may have introduced bias. Through utilising bigger datasets, cross-national comparisons, and lifelong investigations, future study can expand on this work. Furthermore, non-functional needs elicitation and prioritisation using tool-based automation and artificial intelligence techniques are possible areas for future research.

Funding: No funding obtained for this study.
Conflicts of Interest: The author(s) declare(s) that there are no conflicts of interest regarding the publication of this paper.
Availability of Data and Materials: All the data and materials provided in the manuscript.

Reference

[1] Nazim, M., Mohammad, C. W., & Sadiq, M. (2022). A comparison between fuzzy AHP and fuzzy TOPSIS methods to software requirements selection. Alexandria Engineering Journal, 61(12), 10851-10870.
[2] Pargaonkar, S. (2023). Enhancing Software Quality in Architecture Design: A Survey-Based Approach. International Journal of Scientific and Research Publications (IJSRP), 13(08).
[3] Pargaonkar, S. (2023). A Comprehensive Review of Performance Testing Methodologies and Best Practices: Software Quality Engineering. International Journal of Science and Research (IJSR), 12(8), 2008-2014.
[4] Affleck, A., & Krishna, A. (2012, June). Supporting quantitative reasoning of non-functional requirements: A process-oriented approach. In Proceedings of the International Conference on Software and System Process (pp. 88-92). IEEE Press.
[5] Emadi, S., & Shams, F. (2008, July). An approach to non-functional requirements analysis at software architecture level. In Computer and Information Technology, 2008. CIT 2008. 8th IEEE International Conference on (pp. 736-741). IEEE
[6] Kassab, M., Ormandjieva, O., & Daneva, M. (2009, March). A metamodel for tracing non-functional requirements. In Computer Science and Information Engineering, 2009 WRI World Congress on (Vol. 7, pp. 687-694). IEEE.
[7] Umar, M., & Khan, N. A. (2011, July). Analyzing non-functional requirements (nfrs) for software development. In Software Engineering and Service Science (ICSESS), 2011 IEEE 2nd International Conference on (pp. 675-678). IEEE.
[8] Shreda, Q. A., & Hanani, A. A. (2021). Identifying non-functional requirements from unconstrained documents using natural language processing and machine learning approaches. IEEE Access.
[9] Ameller, D., Franch, X., Gómez, C., Martínez-Fernández, S., Araújo, J., Biffl, S., ... & Berardinelli, L. (2019). Dealing with non-functional requirements in model-driven development: A survey. IEEE Transactions on Software Engineering, 47(4), 818-835.
[10] Jung, H. T., & Lee, G. H. (2010, November). A systematic software development process for non-functional requirements. In Information and Communication Technology Convergence (ICTC), 2010 International Conference on (pp. 431-436). IEEE.
[11] Galster, M., & Bucherer, E. (2008, July). A taxonomy for identifying and specifying non-functional requirements in service-oriented development. In Services-Part I, 2008. IEEE Congress on (pp. 345-352). IEEE.
[12] Okpara, L., Werner, C., Murray, A., & Damian, D. (2022, August). A case study of building shared understanding of non-functional requirements in a remote software organization. In 2022 IEEE 30th International Requirements Engineering Conference (RE) (pp. 1-13). IEEE.

[13] Saito, Y., Monden, A., & Matsumoto, K. (2012, October). Evaluation of non functional requirements in a request for proposal (RFP). In Software Measurement and the 2012 Seventh International Conference on Software Process and Product Measurement (IWSM-MENSURA), 2012 Joint Conference of the 22nd International Workshop on (pp. 106-111). IEEE.
[14] Nguyen, Q. L. (2009, May). Non-functional requirements analysis modeling for software product lines. In Proceedings of the 2009 ICSE Workshop on Modeling in Software Engineering (pp. 56-61). IEEE Computer Society.
[15] Kassab, M., Ormandjieva, O., & Daneva, M. (2008, August). A traceability metamodel for change management of non-functional requirements. In Software Engineering Research, Management and Applications, 2008. SERA'08. Sixth International Conference on (pp. 245-254). IEEE.
[16 Ullah, S., Iqbal, M., & Khan, A. M. (2011, July). A survey on issues in non-functional requirements elicitation. In Computer Networks and Information Technology (ICCNIT), 2011 International Conference on (pp. 333-340). IEEE.
[17] Saadatmand, M., Cicchetti, A., & Sjödin, M. (2012, September). Toward model-based trade-off analysis of non-functional requirements. In Software Engineering and Advanced Applications (SEAA), 2012 38th EUROMICRO Conference on (pp. 142-149). IEEE.