Requirements Engineering in Agile Projects

Research Paper Submitted for fulfillment
Of Semester Project
For

Software Requirement Engineering

Focusing

"Requirement Engineering Process For Non-Functional Requirements In Agile Projects"

by

BSEF19M004 - Hurmain Javaid
BSEF19M007 - Anns Shahbaz
BSEF19M010 - Nimra Haq
BSEF19M012 - Iqra Sarwar

To

Ma'am Amna Mirza

Faculty Of Computing and Information Technology

Faculty of Computing & Information Technology

CONTENTS

Abstract

1 Introduction

Requirement Engineering

Elicitation

Analysis

Documentation

Validation & Verification

Management

Agile

Non-functional Requirements

2 Background

3 Motivation

Related Work

Summary Of related work

5 Results

Elicitation

Analysis

Documentation

Validation & Verification

Management

6 Future Directions

7 Conclusion

ARTICLE INFO

Keywords:
Agile,
Non-functional requirements,
State Of Agile,
Functional requirements,
Principles,
Practices,
Requirement Engineering,
Software Development,
Literature review,

ABSTRACT

Context: Agile Principles and Practices are used in 86% software development companies according to the 15th state of agile. Among the major challenges faced in implementation of agile,insufficient attention to Non-functional requirements (NFRs) is very common as the primary focus in agile lies on functional requirements. This is due to lack and inconsistency of practices among teams to deal with NFRs.

Objective: We aimed at presenting the recommendations and guidelines for the requirement engineering(RE) model for NFRs in agile software development.

Method: We conducted a literature review in which we reviewed and analyzed the existing studies and papers concerning the topic to deduce the best practices and recommendations for various activities of RE.

Results: The review revealed a number of practices and guidelines to be followed. Elicitation turns out to be better if dedicated NFR elicitation sessions are conducted with the help of e-procs and experts. Data-driven approach to elicitation seems another effective and emerging solution. 10 recommendations for management of NFRs are also revealed.

Conclusions: We concluded with a complete guideline to requirement engineering of non-functional requirements. These are the collection of guidelines that turns out better according to existing studies, analysis and experiments.

CHAPTER 1

INTRODUCTION

F.Brook says that “The hardest single part of building a software system is deciding precisely what to build.” To develop a worthwhile project requirement engineering is the building stone.

1.1 REQUIREMENT ENGINEERING

According to Ian Sommerville and Pete Sawyer (1997) “Requirements are the description of what should be implemented. These are the specifications of how the system should behave, or what properties or attributes the system should exhibit. These may be a constraint on the development process of the system.” Requirements drive the software development process and play a primary role in the success of the system. Due to the undeniable importance of requirements, the IT community developed a systematic approach to develop the requirements. The systematic way to establish and manage requirements is called requirement engineering.
*“*Requirements engineering comprises all the life-cycle activities related to systemizing the requirements. This includes gathering, analyzing, documenting and managing requirements. With the growing awareness of the significance of requirements in the software process, requirements engineering increasingly becomes an area of focus in software engineering research.”Major activities involved in requirement engineering are shown in Figure 1.1. This process is divided into 5 steps shown below. Here is their brief introduction.

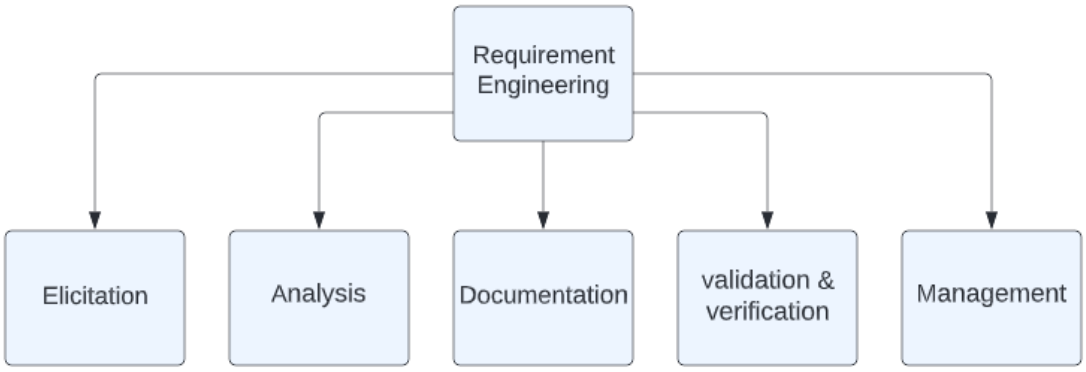

Figure 1.1 Activities involved in Requirement Engineering

Elicitation

Requirements Elicitation is the first and foremost step in requirement engineering. “It is the process of identifying the needs and constraints of the various stakeholders for a software system.” Elicitation is not the same as “gathering requirements.” Nor is it a simple matter of transcribing exactly what users say. Elicitation is a collaborative and analytical process that includes activities to collect, discover, extract, and define requirements. Elicitation is used to discover business, user, functional, and nonfunctional requirements, along with other types of information. Requirements elicitation is perhaps the most challenging, critical, error-prone, and communication-intensive aspect of software development.

Analysis

“Requirements analysis is the process of determining user expectations for a new or modified product - studying and analyzing the customer and the user needs to arrive at a definition of the problem domain and system requirements.” Requirements analysis involves frequent communication with system users to determine specific feature expectations, resolution of conflict or ambiguity in requirements as demanded by the various users or groups of users, avoidance of feature creep and documentation of all aspects of the project development process from start to finish. Energy should be directed towards ensuring that the final system or product conforms to client needs rather than attempting to mold user expectations to fit the requirements. Requirements analysis is a team effort that demands a combination of hardware, software and human factors engineering expertise as well as skills in dealing with people.

Documentation

After analysis of the requirements, documenting them is a very important part of the requirements engineering process as proper documentation provides accurate information about all the requirements and they can easily be verified by all kinds of stakeholders when in written form. The result of requirements development is a documented agreement among stakeholders about the product to be built. The requirements are written in an SRS(Software Requirements Specification) Document.Customers,Marketing Department, Sales Staff, Project Managers, Software development teams, Testers, Maintenance and support staff, Documentation writers, Training personnel, Legal staff and subcontractors, all rely on the SRS. It’s important to organize and write the SRS so that the diverse stakeholders can understand it: Requirements should be labeled, sequence numbers and hierarchical numbering should be used, visual emphasis should be kept in mind, table of contents should be made.

Validation & Verification

Validation and Verification are two different activities in software development. Verification determines whether the product of some development activity meets its requirements (doing the thing right). Validation assesses whether a product satisfies customer needs (doing the right thing). Extending these definitions to requirements, verification determines whether you have written the requirements right: your requirements have the desirable properties. Validation of requirements assesses whether you have written the right requirements: they trace back to business objectives.Validating requirements allows teams to build a correct solution that meets the stated business objectives. Requirements validation activities attempt to ensure that:

- Software requirements accurately describe the intended capabilities and characteristics of the system that meet the needs of different stakeholders.
- Software requirements are properly derived from business requirements, system requirements, business rules, and other sources.
- The specifications are complete, realizable and verifiable.
- All requirements are required and the full set is sufficient to achieve the business objectives.
- All representations of requirements are mutually consistent.The requirements form a solid basis for the further development of design and construction.

But we should keep in mind that we can Validate only documented requirements, not implicit requirements that exist only in someone's head.

Management

Requirements evolve as the project progresses. Due to a lot of changes in scope and objectives we need to maintain requirements in consistent, traceable and unambiguous form. Hence, we have to manage requirements.”Requirements management includes all activities that maintain the integrity, accuracy, and currency of requirements agreements throughout the project.” There are several different methods involved in the management of requirements. They include requirement traceability, version control, change control, status tracking. It also includes the storing requirements,effective use of requirement management tools, considering the effect of plans and commitments on the requirements.

“Agile development refers to a set of software development methods that encourage continuous collaboration among stakeholders and rapid and frequent delivery of small increments of useful functionality”.[KwJb13]

According to the 15th annual report of “state of agile”, agile adoption within software development teams, increased from 37% in 2020 to 86% in 2021. [Soar15]

Requirements engineering involves gathering requirements; the traditional approach is to elicit stakeholder needs and desires and develop them into an agreed-upon set of detailed requirements that can serve as a basis for all development activities.

The agile approach however introduces its own values and practices which are different than those used in more traditional approaches, which also includes somewhat different requirements engineering practices - thus the term of “agile requirements engineering” (ARE) was coined and now is in common use. **[2] ARE** is reported to reduce several requirements-related problems *e.g.* coping with changing requirements, and inadequate communication with the customer, but at the same time it introduces its own challenges that could and should be addressed in order to continuously improve software development processes. Some of the most frequently quoted ARE challenges concern non-functional requirements (NFRs), also known as quality requirements.

NFRs judge the software system based on responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system.Research study and academic research study suggest that agile methodology only considers functional requirements important and non-functional requirements have been ignored [4].

In particular, the **major reported problems** are **: [2]**

- neglecting NFR, while focusing on functionality **[7]–[9]**
- too-minimal requirements documentation to capture NFRs **[6], [8]**
- insufficiency of available ARE techniques to deal with NFRs **[8], [10], [12]**

Such challenges could result in a major obstacle in technology acceptance and its further use, since there are numerous examples showing that insufficient system quality leads to its abandonment by the users and NFRs are the main way of expressing quality expectations. [2] Not considering NFRs in the starting of agile software development has resulted in a failure of at least 60% into models and resolving conflicts among NFRs. [4]

In the industry, functional requirements are treated as primary requirements while **Non Functional requirements (NFR) are ignored** or only catered at design and implementation level.[3] Many researchers claimed that NFRs were typically dealt with as an “afterthought” process and not treated as first-class artifacts during the software requirement phase. [4]. A survey was conducted on Importance of Non-functional Requirements In Agile Software Projects, which concluded that, out of 118 participants, “For over 77% of respondents the

practice of defining NFRs is at least important;for 30% of respondents it is critical for Agile project." [6]

In Agile methods, NFR is ignored due to unawareness of the user about NFR and the nature of agile methods. NFR is ill-defined in agile software development. [3]

Research goal and questions to be addressed: The point of interest in our research is to deal with the issue of negligence of NFR's in ARE; the Enhancement of Non Functional Requirements in Agile Software Development.

Thesis statement (basically the entire idea in a straightforward way):

Outline (basically a road map, explaining sections of our research ppr)

Chapter 2
Background

And the most significant barriers to adopting and scaling Agile practices is "Inconsistent processes and practices across teams ".

CHAPTER 3

MOTIVATION

Related Work

1. M. Younas. conducted a study to present elicitation guidelines for Non-Functional Requirement (NFR). They used eProcurement documents for initial results. Their case study also describes the role of cloud computing in agile methods. They presented guidelines leading towards NFR story cards. Using the preliminary requirements presented by the user, they identify the type of software and the type of NFRs that best suits that particular category of the software. According to the type of NFRs experts are selected and questions are prepared. Users answer the questions and results from those answers are validated by the experts according to the quality standards. The finalized results are validated by the users and stored in the NFR story cards. Although their research presented an effective method for elicitation of the NFRs they don't shed any light on how to analyze, document and review the requirements and what can be the possible or best ways to do that. Also their study is silent about the methods used for preparing the list of questions and validation techniques of the NFRs. Hence they are unable to present a complete and effective method for requirement engineering of NFRs. **[YMjDgIkR17]**
2. Y.Aziz, T Aziz, Malik and Baqer conducted a thorough study about NFRs in agile software development. Initially, they presented the traditional requirement engineering method and agile development briefly. Afterwards, they put forward the core principles and values of agile development. They also discussed the tools for requirement description and the issues faced in agile requirements. They also reviewed the requirement gathering approaches in their literature review. They proposed a methodology for consisting of several phases including inception phase, feature's list identification, feature grouping, group prioritization, NFRs identification, architecture envisioning, task identification and task development. At last, they presented an analysis of NORMAP, MEDoV, and Dual Application Model for combining FRs and NFRs in agile projects. Their study is a worthy contribution as it lays stress on combined models for requirements in agile. But it doesn't serve the purpose of being a complete guide to NFRs in agile software development.**[YaTaMi17]**
3. M Younas presented a study about elicitation of NFRs in agile using cloud computing. They reviewed the existing methodologies like NERV, NORMAP, and CEP. They presented a method for elicitation of NFRs by using cloud computing. Based on the requirements gathered

from users and the cloud services like docs, Dropbox, and pivot tracker they establish preliminary requirements which they use for the identification of software context. According to context, NRFs are identified using extraction and glossary. These NRFs are consulted with experts, issues and candidate NFRs are identified which are then finalized with experts and users. Their methodology identifies NFRs based on the keywords. This methodology was tested by applying on the e procurement document and its results turned out better than the existing methodologies. The only focus of this study is the elicitation of NFRs.**[YMsMaJDi20]**

4. Krogsite's data driven elicitation, assessment and documentation for NFRs is the revolutionary study that steps forward to innovate the requirement management according to the tools and techniques of the modern world. They presented the drawbacks of the traditional elicitation methods as the explicit feedback and interviews can be ambiguous, irrational or biased. These techniques don't cope up with the changing data and requirements too. They presented the Q-rapids method for NFRs. They focused on the time to improve quality concerns, the presentation of QRs, consequences of undertaking QRs, and documentation to add them to backlog. In Q-rapid method data is mined from several sources like user-feedback, software repositories, and monitoring systems. Quality model is formed by data gathered and quality alerts are generated, the QRs pattern catalog then suggest the candidate QRs to decision makers. Accepted QRs are added to backlog for further processing. **[KJR18]**
5. Marc Oriol worked on the data driven approach presented by Krogstie. They elaborated the process model of QR generation and other artifacts involved. They conducted workshops for Q-rapids methodology, gathered results and performed analysis on their basis. In workshops they presented the workflow of Q-rapids and collected feedback from several companies. Companies suggested to have top-down and bottom-up traceability, and easy naming conventions. They suggested to add the facility to dynamically expand QR catalog for missing alerts. They also demanded for the better documentation techniques and compatibility to add QRs in backlog for different documentation patterns. Their study is highly useful for NFRs in agile software paradigm.**[PMR19]**
6. Kulsoom Iftikhar, Saqib Ali And Md Asri Ngadi addressed the problem of negligence of NFRs in agile software development in their research paper. They proposed a four-layered approach for handling NFR in agile software development. Their research also highlighted the importance of NFRs in producing high quality products and how NFRs are the most difficult requirements found on the customer side as well as on the project side. They conducted a survey in different software houses to test the improvement, in software development, occurred by their four-layered technique. The survey results proved their approach to be the most satisfying for finding NFRs, improving the quality of the software product. Statistical results showed that the proposed approach improves the **requirements analysis** process to 15 percent by considering NFRs in early phases of agile software development. Although their research presented a highly effective method of finding and handling NFRs and improved requirements analysis process, they didn’t focus on the documentation and review of the requirements, which is an important step in RE.**[KiSaMan16]**

7. Trupti Suryawanshi and Gauri Rao**,** in their research paper, addressed the problem that There is no proper NFR elicitation method available as process engineers are mainly focused only on functional requirements under the stress of deploying the software within a time, and NFR is also important element of the development process. They proposed a three-tier architectural model, to generate NFRs, in which FRs with NFRs are combined under a single tool and risk evaluation process and Scheduling, designing and visualization of the future plan becomes easy. Their research also focuses on risk driven evaluations and concludes the importance of NFR elicitation in successful software development. Although their research highlighted the importance of NFRs and presented an effective tool for their elicitation, they didn't focus on the other important steps of requirements engineering.**[TsGr16]**

8. Nomi Baruah presented management steps used in different agile development methodologies. According to his research scrum and extreme programming(XP) uses user stories. Other ways include UML diagrams, features list, user input on screens, and story cards for various methods. Although he presented management steps for a wide range of agile methodologies he didn't focus on the NRFs and despite being helpful for management this paper didn't serve the purpose of engineering of NFRs.**[Bn15]**

9. Franco's study analyzed the agile methods to see if they can be used to enhance the system engineering approach. His paper provides an insight to some crucial agile steps including decomposition, traceability, verification & validation, requirements baseline. Some viable results listed by him include continuous delivery pipeline, demos and DevOps.His paper didn't even touch the requirement engineering for NRFs.**[FCC18]**

10. Scarlet Rahy and Julian M. Bass presented their study on management of NRFs in the context of inter-team boundaries and agile software development. They conducted face to face interviews and then analyzed the results of interviews. They worked with two organizations developCo and HealthCo. DevelopeCo being a small and emerging company tries to implement 100% agile in their projects while HealthCo being an internationally established mature company applies 70 to 80 % agile and some waterfall methodologies. They analyzed both and presented their recommendations for documentation and safety critical work items. Their recommendations are viable but lack recommendations for others.**[RSBJm21]**

11. López presented best practices of quality requirements management in agile. They used assembly-based situational method engineering (SME) to find the best practices for quality requirements management.SME uses method chunks as building blocks. According to the proposed problem method requirements are deduced. Method chunks are specified as solutions to these requirements. Requirement map is formed by map process modeling formalism Using specified method requirements. This map includes intentions and multiple strategies to achieve these intentions. They applied the above explained SME to the management of NRFs. They

come up with the intentions of elicitation, specification, communication, validation & validation of Quality Requirements. They presented three strategies for specification that are by refinement, documentation, and prioritization. For prioritization they identify two concrete strategies of prioritization by urgency and prioritization based on value. Their study is a valuable contribution but it is in the initial stage.**[LBKRF17]**

CHAPTER 4

RESULTS

Requirement engineering isn't meant to be a proposed solution applicable to all projects, rather it is a collection of guidelines, recommendations or best practices to be tailored according to the needs of the project. This chapter contains the recommendations for requirement engineering of NFRs that are deduced from the literature review.

4.1 ELICITATION

Elicitation of NFRs must be started in earlier phases of the project as per the best practices for NFRs management. Today's software development is data driven in which development can’t be based on the initial or static data. Elicitation must be carried out as the objectives and requirements of the project changes. Following are few guidelines for the better elicitation NFRs in agile software development.

Dedicated NFRs Elicitation Sessions

Different elicitation techniques for agile projects are common for FRs and NFRs. These techniques include face-to-face and online meetings between customers and development team, interviews with stakeholders, Brainstorming, prototyping and workshops.Dedicated elicitation techniques for NFRs aren’t conductedcommonly. But it is recommended to arrange a few dedicated sessions for NFRs. These sessions can be dedicated workshops or quality attribute workshops.

Prepare Questionnaire

Before the elicitation session, spend some time to find out more about the proposed project. Previous studies and projects show that well prepared questions are a big help for better elicitation.

NFR Expert

NFR experts know better about the candidate questions, issues and improvements for NFRs. It is highly recommended to ask help from experts while preparing the questions, during the analysis and refining of NFRs.

Regional E-procurement Strategy

Different regions and countries have their strategy and roadmap for e-procurements which software companies can't refrain from. During the elicitation and finalizing of NFRs, consulting e-procurement strategy is helpful for starting development in the right direction.

Dedicated Structure

NFRs elicitation is conducted through user stories and traditional methods. Several studies recommend using a dedicated structure for NFRs. M Younas presented NRF cards as a dedicated structure. There exist several other structures that can be tailored according to the requirements of the project.

NFR Quality

Quality of NFRs elicited is a major issue. Eliciting and implementing requirements requires intensive communication. They become obsolete if they are not workable due to their poor quality. Requirements must be clear, unambiguous, verifiable and measurable.

Cloud Computing & Keyword Based Elicitation

Cloud computing tools are a big help these days for generating NFRs based on keywords. NFRs are recorded in these tools. These documents are then analyzed to locate keywords and then generate NFRs automatically. M younas study is a great resource for guidance in this regard. Tables of the cloud computing tools and keywords from their study are given below.

Table 4.1 Tools for sharing data and communication

No	Description	Cloud services, and tools
1	Sharing of Requirement documents	GoogleDocs, Drobox, Team Foundation Server (TFS), Pivotal Tracker
2	Team Communication	Cockpit,Mingle,Skype,Basecamp,Microsoft share point, Bootstrap today and Confluences

Table 4.2 NFR types and indicator keywords

Accessibility	administrator, add, choose, edit, lhcp, hcp, visit, privilege, read, office, representative, sort, name, personal, dlhcp, view, status,

	accessor, role, list
Auditability	authorship, trail, arise, worksheet, auditable, exclusion, reduction, deletion, examine, editing, stamp, non-repudiation, inclusion, id, alteration, finalize, disable, summarize, attestation, log
Availability	availability, addition, achieve, 24, 98, 99, available, hour, day, online, schedule, technical, year, transmit, integrity, maintenance, period, resource, %, confidentiality
Accuracy	accurate, consistent, time, correct, exact, definite, accuracy, certainty, correctness, definiteness, perfection, near true value, error free, correct, precision, standard
Compliance	require, compliance, submission, accede, bow, put, forth, acquiescence, biddability, compliancy, deference, obedience, collaboration, teamwork, prostration, adjust, adapt, custom, get used to, standards, conform to requirements, follow rule, act in accord with accepted standards, o official requirements, regulations,
Confidentiality	confidential, private, esoteric, hushed, intimate, privy, nonpublic, secret, common, open, public, shared, advertised, blazed, broadcast, declared, proclaimed, professed, promulgated, publicized, published, reporting, reported, widespread, discreetness, circumspection, prudence, data, protection, unauthorized, information, information protection,
Configuration	Configuration, architecture, armature, cadre, frame, edifice, fabric, framework, infrastructure, shell, skeleton, composition, material, matter, stuff, substance, assemble, piece, make, create, connect, tie, link up, reassemble, computer
Documentation	document, documentary, contract, documentation, confirmation, corroboration, proof, evidence, substantiation, testimonial, validation, voucher, witness, certificate, exhibit, illustration, authentication, identification, manifestation, verification, confirmation, information, substance, communication, reinforcement, corroboration, support, describe, define, specify, report information, certify activities, requirements, management, identify, acquire, process, store, disseminate
Efficiency	efficient, edge, effectiveness, productive, , economy, ualness, ratio
Interoperability	operability, interoperability, available, employable, exploitable, fit, functional, practicable, service, useful, impracticable, nonfunctional,

	unavailable, unemployable, unusable, ability, quality, adaptability, compatibility
Legal	legally, legal, court ordered, jural, ratified, sanctioned judicial, juristic, statutory, legislative, legislature, legislation, illegal, valid, invalid, lawful, legitimate, licit, allowable, authorized, noncriminal, permissible, justifiable, warrantable, constitutional, dejure, regulation, statutory, good, innocent, just, proper, right, illegitimate, illicit, lawless, unlawful, wrongful, , harm, scope, affect, derive, vocabulary, reuse
Multilingual	multilingual, multi languages, support, multiple, language, support, more than one language, multiple languages
Performance	perform, interpretation, account, reading, rendition, version, nonfulfillment, nonperformance, space, time, memory, response, time, throughput, second, minutes, hour, day, week, month, year, byte, kilobyte, megabyte, gigabyte, execution, instruction, execution, perform, efficiently, functional, operate, operational, fast, simultaneous, scale, capable, increase, peak, longer, average, acceptable, lead, handle, flow, response, capacity, maximum, cycle
Reliability	reliably, reliability, unreliability, unreliable, dependability, dependableness, responsibility, solidity, solidness, sureness, infallibility, reproducibility, duplicability, trustiness, accountability, answerability, availability, recoverability, MTBF, probability of availability, continual operation, perform, functions, dependent, validate, input, query, accept, loss, failure, operate, alert, laboratory, prevent, database, product, appropriate, event, application, capability, ability, time
Scalability	scalable, scalability, able, equal, fit, good, qualified, suitable, incompetent, inept, poor, quantifiability, measurability, ratability, simultaneous, second, scale, capable, increase, longer, average, acceptable, lead, handle, flow, response, capacity, maximum
Security	authority, authorities, security, invulnerable, impregnable, inviolable, secure, strong, unassailable, hazard, risk, threat, instability, precariousness, harm's way, exposure, liability, openness, violability, vulnerability, susceptibility, susceptibleness, danger, distress, endangerment, imperilment, jeopardy, peril, trouble, secureness, protection, shelter, safety,

Data Driven Elicitation

Data is the primary driving force in softwares these days. There is a continuous change in the scope, requirements, and objective of softwares. NRFs can't be elicited at once and relied upon in the whole project. They need rework as the functional requirements evolve. There is a need for a system that continuously monitors the quality, alerts when quality deteriorates and suggests new NFRs. This can be made possible by the use of project management tools, Github, JIRA and others. Krogstie and Oriol studies presented their study with Q-rapids technique that can be applied extensively.

NFRs and FRs Combined Model

Elicitation of NFRs and FRs can't be isolated. Methodologies that focus on the combined model of both exist too. NERV, NORMAP, CEP and NFRelicit are few of them. According to the software under development the suitable method should be adopted.

4.5 MANAGEMENT

Management of NFRs is an umbrella activity spanning the whole course of the development. Below are the few recommendations for managing NFRs in an agile paradigm that are considered as best practices and produce better results according to the analysis of existing literature.

No	Recommendation	References
MR1	Start working on NFRs early in the project.	[LBKRF17] [JAWP21] [ksOmN20]
MR2	Educate stakeholders, developers and other actors of the project about the importance of NFRs and consequences of overspecified NFRs.	[JAWP21] [ksOmN20]
MR3	Make an independent team for testing of NFRs and involve NFR specialists team to ensure proper implementation of NFRs.	[JAWP21] [ksOmN20]
MR4	Use a suitable version Control system to record and update the NFRs. Ensure that the whole team is working on the same updated version in each iteration.	[KwJb13]
MR5	Use continuous integration, continuous delivery(CI/CD) to utilize	[FCC18]

	automated testing, early and continuous monitoring by developers and stakeholders.	[JAWP21] [KwJb13]
MR6	Label requirements by proper status. Possible status includes Proposed, InProgress, Drafted, Approved, Implemented, Verified, Deferred, Deleted, and Rejected.	[KwJb13]
MR7	When a change is proposed, during analysis consider its impact on NFRs carefully and don't ignore consequences if it's a turnover for NFRs.	[KwJb13]
MR8	Divide sprint in different chunks including careful allocation to NFRs and CI/CD requirements along with FRs.	[JAWP21]
MR9	Arrange NFR based code reviews.	[JAWP21]
MR10	Use automated monitoring tools e.g. SONAR to monitor the quality of software systems under development.	[PMR19] [JAWP21] [LBKRF17]

BIBLIOGRAPHY

B

1. [Bn15] Baruah, N. (2015). **Requirement Management in Agile Software Environment.** *Procedia Computer Science*, *62*, 81–83. https://doi.org/10.1016/j.procs .2015.08.414

F

[FCC18] Franco (Frank) Curtolo, CEng **Requirements Management applied in an agile Project Management environment** Copyright © 2018 by Principal Consultant, Program Planning Professionals Ltd, Frank.Curtolo@pcubed.com.

J

[JAWP21] Jarzębowicz, Aleksander & Weichbroth, Paweł. (2021). **A Systematic Literature Review on Implementing Non-functional Requirements in Agile Software Development: Issues and Facilitating Practices.** 10.1007/978-3-030-67084-9_6.

K

[KwJb13] Karl Wiegers And Joy Beatty **Software Requirements**, Copyright © 2013 Karl Wiegers And Seilevel

[ksOmN20] Kopczyńska, S., Ochodek, M., Nawrocki, J., 2020. **On Importance Of Non-Functional Requirements In Agile Software Projects—A Survey, In:Studies In Computational Intelligence.** Pp. 145–158. Https://Doi.Org/10.1007/978-3-030-26574-8 _11

[KJR18] Franch, X. et al. (2018). Data-Driven Elicitation, Assessment and Documentation of Quality Requirements in Agile Software Development. In: Krogstie, J., Reijers, H. (eds) Advanced Information Systems Engineering. CAiSE 2018. Lecture Notes in Computer Science(), vol 10816. Springer, Cham. https://doi.org/10.1007/978-3-319-91563-0_36

L

[LBKRF17] López, L., Behutiye, W., Karhapää, P., Ralyté, J., Franch, X., Oivo, M., 2017. **Agile Quality Requirements Management Best Practices Portfolio: A Situational Method Engineering Approach**, Https://Doi.Org/10.1007/978-3-319-6992 6-4_45

P

[PMR19] Oriol, M. *et al.* (2019). **Data-Driven Elicitation of Quality Requirements in Agile Companies.** In: Piattini, M., Rupino da Cunha, P., García Rodríguez de Guzmán, I., Pérez-Castillo, R. (eds) Quality of Information and Communications Technology. QUATIC 2019. Communications in Computer and Information Science, vol 1010. Springer, Cham. https://doi.org/10.1007/978-3-030-2923 8-6_4

R

[RSBJm21] Rahy, S., & Bass, J. M. (2021). **Managing non‐functional requirements in agile software development**. *IET Software*, *16*(1), 60–72.https://doi.org/10.1049 /sfw2.12037

Y

[YMjDgIkR17] Younas, Muhammad & Jawawi, Dayang & Ghani, Imran & Kazmi, Rafaqat. (2017). **Non-Functional Requirements Elicitation Guideline For Agile Methods.** Journal Of Telecommunication, Electronic And Computer Engineering. 9. 137-142.

[YaTaMi17] Y. Aziz , T. Aziz , M. I. Malik , M. K. Baig , M. Z. Ali , M. Baqer Non Functional Requirement in Agile Software Development. Technical Journal, University of Engineering and Technology (UET) Taxila, Pakistan Vol. 22 No. I-2017

[YMsMaJDi20] Younas, Muhammad & Shah, Muhammad Arif & Jawawi, Dayang & Ishfaq, Muhammad & Awais, Muhammad & Wakil, Karzan & Mustafa, Ahmad. (2020). Elicitation Of Nonfunctional Requirements In Agile Development Using Cloud Computing Environment. Ieee Access. Pp. 1-1. 10.1109/Access.2020.3014381

Abstract
Statement

-intro to agile
-intro to non functional

Background
Topic content
Future prediction
Conclusion
https://tynerblain.com/blog/2009/02/10/agile-non-functional-reqs/
https://agilecheetah.com/handling-non-functional-requirements-in-agile-projects/

[2] **A. Jarzębowicz and P. Weichbroth, "A Qualitative Study on Non-Functional Requirements in Agile Software Development," in IEEE Access, vol. 9, pp. 40458-40475, 2021, doi: 10.1109/ACCESS.2021.3064424.**

[3] Non-Functional Requirements Elicitation Guideline for Agile Methods
https://www.researchgate.net/publication/321300052_Non-Functional_Requirements_Elicitation_Guideline_for_Agile_Methods

[4] NON-FUNCTIONAL REQUIREMENTS AS CONSTRAINTS AND THEIR VALUES IN SOFTWARE DEVELOPMENT: A REVIEW
http://www.jseis.org/Volumes/Vol5/V5N2-5.pdf

[5]
https://www.wrike.com/project-management-guide/faq/what-is-agile-methodology-in-project-management/

[Soar15] "15th annual state of agile report,"2021 (accessed: 13.06.2022) https://stateofagile.com/ .

[AaCw05] Aybüke Aurum · Claes Wohlin (Eds.) 123 Engineering and Managing Software Requirements

[KiSaMan16] KULSOOM IFTIKHAR, SAQIB ALI, MD ASRI NGADI, "**Enhancement of Non Functional Requirements in Agile Software Development**" International Journal of Computer Science and Information Security (IJCSIS), Vol. 14, No. 12, December 2016

[TsGr16] Trupti Suryawanshi, MTECH, Computer Engineering, BVDUCOE, Pune, India
&
Gauri Rao, Associate Professor, Computer Engineering, BVDUCOE,Pune, India,
"**Modeling of Nonfunctional Requirements for Agile Development Processes**"
Vol.04 Issue-02, (February, 2016)
IJITE - International Journal in IT and Engineering, Impact Factor- 5.343

[7] B. Ramesh, L. Cao, and R. Baskerville, ''Agile requirements engineering practices and challenges: An empirical study,'' Inf. Syst. J., vol. 20, no. 5, pp. 449–480, Nov. 2007.

[8] V. T. Heikkilä, D. Damian, C. Lassenius, and M. Paasivaara, ''A mapping study on requirements engineering in agile software development,'' in Proc. 41st Euromicro Conf. Softw. Eng. Adv. Appl., Aug. 2015, pp. 199–207.

[9] I. Inayat, S. S. Salim, S. Marczak, M. Daneva, and S. Shamshirband, ''A systematic literature review on agile requirements engineering practices and challenges,'' Comput. Hum. Behav., vol. 51, pp. 915–929, Oct. 2015.

[10] S. Wagner, D. Méndez Fernández, M. Kalinowski, and M. Felderer, ''Agile requirements engineering in practice: Status quo and critical problems,'' CLEI Electron. J., vol. 21, no. 1, p. 15, Apr. 2018.

[11] J. Medeiros, D. C. Alves, A. Vasconcelos, C. Silva, and E. Wanderley, ''Requirements engineering in agile projects: A systematic mapping based in evidences of industry,'' in Proc. Ibero-Amer. Conf. Softw. Eng. (CIBSE), 2015, pp. 460–476.

[12] W. Alsaqaf, M. Daneva, and R. Wieringa, ''Quality requirements in large-scale distributed agile projects—A systematic literature review,'' in Proc. Int. Working Conf. Requirement Eng., Found. Softw. Qual. Cham, Switzerland: Springer, 2017, pp. 219–234.