JOURNAL OF SOFTWARE ENGINEERING & INTELLIGENT SYSTEMS
ISSN 2518-8739
31 August 2020, Volume 5, Issue 2, JSEIS, CAOMEI Copyright © 2016-2020
www.jseis.org

NON-FUNCTIONAL REQUIREMENTS AS CONSTRAINTS AND THEIR VALUES IN SOFTWARE DEVELOPMENT: A REVIEW

IQRA BATOOL [1], LAREB KOSAR [2], MARRIUM MEHMOOD [3]

[1,2,3] *Department of Software Engineering, Bahria University Islamabad, Pakistan*
Email: ms.iqrasatti@gmail.com [1], laraibkousar@gmail.com [2], mrymmhmood@gmail.com [3]

ABSTRACT

OPEN ACCESS

Academic Editor: Muhammad Imran Babar, Pakistan.
Published: 31 August, 2020
Funding: Authors have no funds.
Article Type: Review Article

Non-functional values specify the system's operational capabilities and constraints to enhance its functionality. The non-functional requirements (NFRs) may include security, reliability, performance etc. We may say that constraint is a line with quality expectations. NFR is always a constraint to other user stories. In some scenarios, NFRs are intangible that require human judgement. Agile Software Development (ASD) model is very popular worldwide for developing software. NFRs are not considered essential in ASD. In this paper, we discuss how non-functional values work as constraint and add value to NFR in agile methodology. NFRs ensure the reliability, usability and performance of the software system. NFRs are vital for the success in case of ASD. If they are not clearly addressed undesirable results may occur such as developer cannot satisfy the user and many other issues may arise in development. There are two types of constraints internal quality constraints and external quality. NFRs are also considered as backlog constraints.

Keywords: non-functional requirements; software development; agile software development; constraints; agile methodology; software quality;

1. INTRODUCTION

Broadly speaking requirement engineering is divided into two parts functional and non-functional requirements Non-functional requirements specifies the quality attributes of software system. They 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. As from the past papers we come to the point that NFRs are not given important like functional requirement. Research study and academic research study suggest that agile methodology only consider functional requirement important and non-functional requirement has been ignored [1, 2]. As with increase in complexity and scale of software it become essential that with functional requirement non-functional requirement are also important. Dealing with NFRs involves many activities like eliciting, modeling and analysing [3]. Non-functional requirement describes how any of the system works. The definition for non-functional requirements is essentially specifies how the system should behave and that is it is constraint on system behaviors. Non-functional requirements are also considered as quality attribute for a system. Non-functional requirements cover all the requirements which are not covered by the functional requirements. Like all other requirement NFRs must be quantified for the clarity to ensure stakeholders desires are clearly and concisely understood by everyone. Next, we have provided a small review of non-functional requirement as constraints and their values in software development.

There are certain ways to show non-functional requirement in agile mythology. The most common ways are backlog item and acceptance criteria [4, 5]. Non-functional requirement as backlog item: non-functional requirement is visible by creating independent backlog item. Non-functional requirements will be developed and tested before the backlog item is considered "done".

Agile method deals with unstable and volatile requirement by using number of techniques. The most noticeable are simple training, short iteration, earlier release and frequent customer feedback [6].

Non-functional requirement acceptance criteria: Another way to track non-function requirements is adding them into acceptance criteria. Acceptance criteria is condition of satisfaction by the stakeholders especially when the priorities are of stakeholders are involved [7, 8].

Researcher usually face numerous challenge which includes great diversity of NFRs, formal specification of requirements, NFRs subjective nature, embedding them. Research has shown by

Not considering NFRs in starting of agile software development has resulted in a failure in at least 60%into models and resolving conflicts among NFRs [9]. Most of the work is derived from Chung's NFR framework but it lacks the integration of NFR into conceptual mode [9]. Non-functional requirement are operationalizable for example security is first refined into the authenticate user and further refined to authorize user by user Id and password [4]. Agile methods for developing software is popular because of these characteristics' short deadline, smaller team, consistent change in requirements etc. Agility impacts not only design and coding but also concerns requirements engineering [10]. Adoption of agile methods leads to higher development quality and cheaper rate. This research study is aimed to focus on the values of NFRs in agile methods. Many researchers claimed that NFRs were typically dealt with as an "afterthought" process and no treated as first-class artifacts during the software requirement phase.

This paper is organized as follow section II contains related work. Section III present non-functional requirement as constraints. Section IV describes the values of non-functional requirement in agile methodology. Section V discuss the challenges and future work. Section VI summarizes the conclusion and section VII include the references.

2. RELATED WORK

Some of the software prioritization techniques, conflicting requirements, challenges and solution and interdependencies of non-functional requirements are discuss below.

2.1 Non-functional requirements in agile software development

Requirement engineering is complicated due to documentation. There are three major issues in agile requirements.

- **Missing requirement activities**:

For creation of activity specification document there is no technique

- **Missing Requirement Interface:**

As stakeholder does not know the whole requirements of system which cause missing requirement interface.

- **Non-functional Requirement Elicitation:**

The main issue in agile is that at every step product is handed over to customer to make sure his/her requirements met. In this paper researcher divide proposed methodology into several steps.

- **Non-Functional requirement identification:**

Consist of Elicitation, validation & prioritization.

Basic concept of NORMAP technique is to combine both NFR and FR into single model.

The main steps of NORMAP are as follow

Step 1: Select NFR and initial data collection
Step 2: Initial data processing
Step 3: Automatic parsing of requirements statements.
Step 4: Modeling agile use cases
Step 5: modeling agile loose cases
Step 6: Modeling agile choose cases
Step 7: Requirement implementation sequence planning

2.2 Non-functional Requirement challenges and solutions

ASD put less emphasize on documentation of NFR. In ASD, NFR are ill-defined and rarely documented and there is no legal acceptance test for NFR, as a result problem arises in later stages of development[11]. Cao and Ramesh identified that neglect of NFRs and minimal documentation as major challenges of agile requirement engineering. Minimal documentation and neglect of NFRs in ASD result in challenge of scalability of software [12]. Increase cost and time. Security is major issue.

Researcher propose guideline for documenting NFRs in ASD Researcher distinguish three different type of scope of NFRs.

Type 1: System wide for those who apply the entire system
Type 2: Group wide for those who apply for a setoff user stories.
Type 3: Local for those who apply a single user story.

The proposed guideline acknowledge diversity of NFRs and utilizing existing ASD artefact such as epic, user stories and acceptance criteria for documenting NFRs.

2.3 Software requirement prioritization based on non-functional requirements

Few approaches and their drawbacks

AHP incorporated consistency check

Cost value approach fails to assess interdependencies between prerequisite [13].

Numerical assignment (Grouping): classifying requirements into different and handling them over to each stakeholder.

In proposed approach, assign importance value to each non-functional requirement based on pairwise method. Assign importance value to each functional requirement based on non-functional requirements. Calculate prioritization by matrix multiplication [14].

2.4 Determining Interdependencies among NFRs in agile environment to reduce conflicts

The main limitation of UML based approach is that they only consider performance requirements, not all NFRs. NFRs hold a conflicting nature and may affect software positively or negatively.so it is very important to handle NFRs by using some tools and methodologies because of its subjective nature. For automatic optimization new methodology is introduced known as Softgoal interdependency rule set graph (SIRGs) [6]. As NFRs are not equal in nature and different from each other but this approach treats all NFRs equally which is the main drawback of this approach. In conventional requirement engineering, NFRs are not properly defined lack of considerations of NFRs in agile methodologies [15-17]. NFRs in early stages of development has great benefit but later stages it increases cost and complexity [18, 19].

After through study researcher introduce the approach for identifying and classifying NFRs along with FRs to reduce conflicts. Requirement elicitation: the purpose of this model is to elicit user requirement then prioritize them for next phase of development to minimize the development error [9]. In this phase requirements are decomposed into two steps by refining and by prioritizing.

Kano questionnaire: First component consists of questionnaire 1st question is about stakeholder reaction and second question is about feeling of stakeholder. Dealing with NFRs and their classifications. Dealing with dependencies. Identification and resolution of conflicted NFRs [20-22]. After using all of the above technique the project’s requirements are validate from customer.

2.5 Enhancement of Non-Functional Requirements:

For the development and designing of non-functional requirements at the early stage of software development four layered approach is introduced. This methodology enhances the process of non-functional-requirements in agile software development [23, 24]. The steps of this techniques are:

Step 1: Recognize the key partners of the framework
Step 2: Create the objectives from Stakeholders in view of designer's information and experience.
Step 3: Divide the objective into sub objectives.
Step 4: Recognize the Non-Functional Requirements for every sub objective.

The success rate of this technique is 15% [25-27].

2.6 Capturing, Eliciting, and Prioritizing (CEP) NFRs Methodology:

CEP "Capture Elicit Prioritize" methodology to retrieve Non-Functional Requirements from requirements documents and diagram/images contained in the docs. Validation of this research was done by using 26 requirements docs from European Union eProcurement. In comparison to NORMAP and NERV methodology CEP [16, 28, 29] technique shows better and accurate results but not 100% enhancements are still required in the future. The limitations of this techniques are that it is only restricted to United States and European Union documents, not scalable for large projects.

3. NON-FUNCTIONAL REQUIREMENTS AS CONSTRAINTS

3.1 Internal quality

- Rule is a "constraint" that sets a limit to comply during software construction
- Impose constraints using "Rules"
- Internal qualities such as maintainability and testability, that are barely visible by the stakeholders but simplify how to build the software

- Each user story is not Done until each rule is confirmed.
- It can be confirmed by doing peer reviews / inspection.

The internal qualities are mentioned in Table 1 and Table 2.

Table. 1 NFR simplicity and naming convention

Non-Functional Requirements	Rule
Simplicity	**Naming Convention:** Practices that ensure code is its own best documentation by allowing useful information, such as programming constructs, to be deduced from the names.

Table. 2 Definitions of internal qualities

Non-functional Requirements	Definition
Simplicity	Ease to understand or explain.
Maintainability	Ease to evolve with change and minimal effort.
Testability	Ease to confirm conformance by observing a reproducible behavior.
Portability	Ease to reuse for multiple platforms.
Extensibility	Ease to takes into consideration future growth.

3.2 External quality

- Restriction is a "constraint" that sets a limit to comply during software execution.
- External qualities such as performance, correctness, security and usability, that carry out the software's functions at run time, and as such, are not only visible by stakeholders but also highly desirable [14, 30]. For example: the user requires 2 BHK but it requires the rooms to be in good shape as well.
- It can be confirmed by test(s), automation or performance testing.
- Restriction is specific for one scenario.
- Restriction has a measurable quality objective.
- Restriction is confirmed by test(s).

The definitions of the external qualities are mentioned in Table 3.

Table. 3 Definition of external qualities

Non-functional Requirements	Definition
Correctness	Ability with which the software respects the specification.
Performance	Ease with which the software is doing the work it is supposed to do.
Usually	It is measured as a response time or a throughput.
Reliability	Ability with which the software performs its required functions in understated conditions for a specified period of time.
Robustness	Ability with which the software copes with errors during execution.
Scalability	Ability with which the software handles growing amounts of work in a graceful manner.
Security	Degree to which the software protects against threats.
Usability	Ease with which the software can be used by specified users to achieve specified goals.

3.3 Restriction (external quality constraints)

- Restriction is specific for one scenario.
- Restriction has a measurable quality objective.
- Restriction is confirmed by test(s).

Table 4 highlights the characteristics of the restrictions being imposed.

Table. 4 Characteristics of the restrictions

Characteristics	Description
Specific	It should target a piece of functionality that is small, consistent and simple.
Measurable	It imposes a limit that is measurable, otherwise how would you know when you've addressed it.

Attainable	It is recognized as achievable by the team.
Relevant	It is directly related, connected, and pertinent to the non-functional requirement.
Traceable	It is linked with a requirement and a target that justifies why it exists.

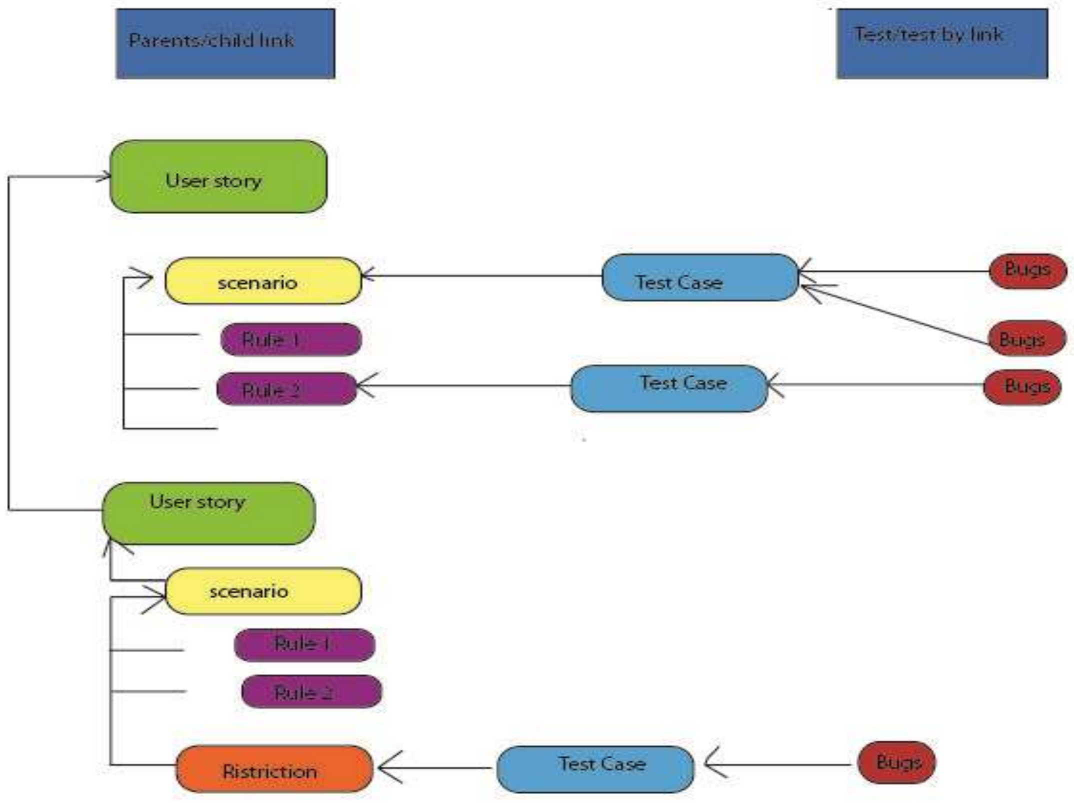

Figure. 1 Restriction confirmation by tests

4. NFRS AS BACKLOG CONSTRAINTS

NFRs are not themselves backlog item rather NFRs are backlog constraints. They are constraint on the development that limit some degree of design for those building systems. These constraints often defined in acceptance criteria for multiple backlog. NFRs are associated with back log through SAFe [20, 31]. Because non-functional requirement has significant attribute of solution that agile release train (ART) and value stream create, they most often influence the backlog of ART and solution trains. NFRs roll down to constraints team-level backlog item. NFRs may constrain any number of backlogs item.in order to know the system complies with constraints most NFRs requires one or most quality test [17, 29].

5. VALUES OF NFRS IN AGILE METHODS

It is necessary to define the non-functional requirements as they are critical to project success. Over specifying will pose a question about quality and price of the system. Agile neglects the non-functional requirements and only focuses on functional requirements [18, 28, 32].

It is important to focus on correcting non-functional requirements right so that the software runs well and is long-term sustainable. These contribute to the success of the software as much as the functional requirements do so they should not be overlooked [33]. This is because they will affect user experiences significantly. An example of an NFR would be “how quickly does a web page loads?” A page should load in less than 3 seconds, so this NFR allows you to lay down those law that must be compiled with. Four example of non-functional requirements groups: usability, reliability, performance and supportability as well.

5.1 Usability

Usability is the degree of ease with which the user will interact with your products to achieve required goals effectively and efficiently. Prioritize the important function of the system based on usage patterns. Frequently used functions should be tested for usability, as should complex and critical functions [12, 34]. Usability requirements for interface design should support the following from the perspective of its primary user.

Efficiency of use: Goals are easy to accomplish quickly with few or no user error.

Intuitiveness: The interface is easy to learn and navigate button, headings help/error messages are simple to understand [9, 35].

Low perceived workload: The interface appears easy to use, rather than intimidating demanding and frustrating.

5.2 Reliability

Reliability describes the trust in the system/devices that develops after a time of use it. It defines the likelihood of the software to work without fail over for a given time of period. The number of bugs in the code, hardware failures, and problems can reduce the reliability of the software. Your goal should be a long MTBF (mean time between failures) [3, 27, 29]. It is defined as the average period the system runs before failing. Create a requirement that data created in the system will be retained for several years without the data being changed by the system. It is also good idea to also include requirements which make monitoring system easier. The NFR data from the NORMAP and NERV methodologies along with the CEP methodology was used to determine the overall value of the NFRs based on Chung's NFR Framework [36, 37].

5.3 Performance

What should system/machine response time will be measured from any point under what circumstances? Is there particular peak period when the load on the network is exceptionally high? Think of stress times, at the end of the month or in the conjunction with payroll disbursements. The biggest problem with performance NFR is that it is treated as a qualitative measure rather than a quantitative one [38]. It is important to consider performance constraints from very initial stage and monitor them regular basis. NFRs such as security crosscut the different parts of the system which also improve the performance [39]. The below diagram shows the performance impact of NFRS.

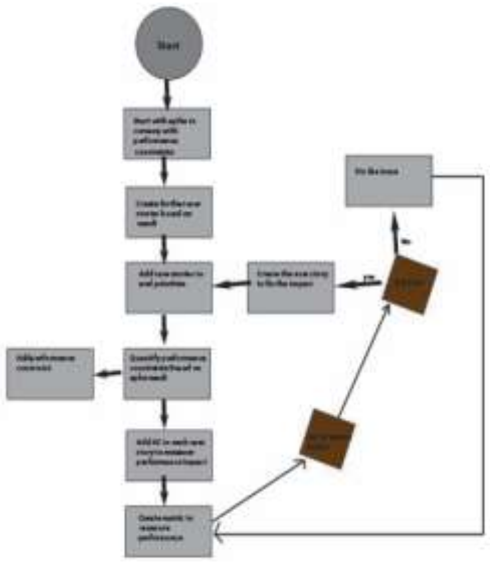

Figure. 2 Performance Impact of NFRs

5.4 Supportability

The system needs to be **cost-effective to maintain**. Maintainability requirements may cover diverse levels of documentation, such as system documentation, as well as test documentation [18, 40, 41], e.g. which test cases and test plans will system support? Supportability is highly vital in terms of NFRs both for industrialists and researchers [38, 42].

6. CONCLUSION

Non-functional requirements (NFR) have recently become the focus of industry and research. Non-functional requirement should be treated equally as functional requirement in software engineering. Neglecting non-functional requirements in agile is the biggest limitation. There has been lack of research in early stages of agile. This is a very attractive feature for agile team members to replace parts of the framework in an agile manner to suit their needs in prioritizing NFRs. Performance, reliability, security and supportability are the characteristic which make the system better. In this paper we focused on non-functional requirement as internal and external constrained and we also describe the values of non-functional requirement. Functional requirements are not stable without non-functional requirement. Thus, NFRs should be considered essential at every stage of the development. NFRs may constraint any number of backlog item. We should work on the values of non-functional requirement to make the development better.

REFERENCES

1. Farid, W.M. and F.J. Mitropoulos. *NORMATIC: A visual tool for modeling non-functional requirements in agile processes*. 2012: IEEE.
2. Glinz, M. *Rethinking the notion of non-functional requirements*. in *Proc. Third World Congress for Software Quality*. 2005.
3. Yin, B. and Z. Jin. *Extending the problem frames approach for capturing non-functional requirements*. 2012: IEEE.
4. Xu, L., et al., *An architectural pattern for non-functional dependability requirements.* Journal of Systems and Software, 2006. 79(10): p. 1370-1378 %@ 0164-1212.
5. Affleck, A., A. Krishna, and N.R.J.T.C.J. Achuthan, *Non-functional requirements framework: a mathematical programming approach.* 2015. 58(5): p. 1122-1139.
6. Huo, M., et al. *Software quality and agile methods*. 2004: IEEE.
7. Babar, M.I., et al., *Stakemeter: Value-based stakeholder identification and quantification framework for value-based software systems.* PloS one, 2015. 10(3): p. e0121344.
8. Babar, M.I., et al., *Stakeholder management in value-based software development: systematic review.* IET Software, 2014. 8(5): p. 219-231.
9. Ullah, S., M. Iqbal, and A.M. Khan. *A survey on issues in non-functional requirements elicitation*. 2011: IEEE.
10. Nawrocki, J., et al. *Agile requirements engineering: A research perspective*. in *International Conference on Current Trends in Theory and Practice of Informatics*. 2014: Springer.
11. Schön, E.-M., et al., *Agile Requirements Engineering: A systematic literature review.* 2017. 49: p. 79-91.
12. Ho, C.-W., et al. *On agile performance requirements specification and testing*. in *AGILE 2006 (AGILE'06)*. 2006: IEEE.
13. Butt, S.A. and T.J.P.o.A.P.J.o.M.R. Jamal, *Frequent change request from user to handle cost on project in agile model.* 2017. 5(2): p. 26-42.
14. Farid, W.M. and F.J. Mitropoulos. *NORPLAN: Non-functional Requirements Planning for agile processes*. in *2013 Proceedings of IEEE Southeastcon*. 2013: IEEE.
15. Domah, D. and F.J. Mitropoulos. *The NERV methodology: A lightweight process for addressing non-functional requirements in agile software development*. in *SoutheastCon 2015*. 2015: IEEE.
16. Franch, X. *Systematic formulation of non-functional characteristics of software*. in *Proceedings of IEEE International Symposium on Requirements Engineering: RE'98*. 1998: IEEE.
17. Chung, L., B.A. Nixon, and E. Yu. *Using non-functional requirements to systematically support change*. in *Proceedings of 1995 IEEE International Symposium on Requirements Engineering (RE'95)*. 1995: IEEE.
18. Aljallabi, B.M. and A. Mansour. *Enhancement approach for non-functional requirements analysis in Agile environment*. 2015: IEEE.
19. Glinz, M. *On non-functional requirements*. in *15th IEEE International Requirements Engineering Conference (RE 2007)*. 2007: IEEE.
20. Guan, Y. and A.K. Ghose. *Use constraint hierarchy for non-functional requirements analysis*. in *International Conference on Web Engineering*. 2005: Springer.
21. Jingbai, T., et al. *A context awareness non-functional requirements metamodel based on domain ontology*. in *IEEE International Workshop on Semantic Computing and Systems*. 2008: IEEE.
22. Cysneiros, L.M. and E. Yu, *Non-functional requirements elicitation*, in *Perspectives on software requirements*. 2004, Springer. p. 115-138.
23. Arbain, A.F., et al., *Non-Functional Requirement Traceability Process Model for Agile Software Development.* 2017. 9(3-5): p. 203-211.
24. Mahalakshmi, K., R.J.G.J.o.C.S. Prabhakar, and Technology, *Performance evaluation of non functional requirements.* 2013.
25. Farid, W.M. and F.J. Mitropoulos. *Visualization and scheduling of non-functional requirements for agile processes*. in *2013 Proceedings of IEEE Southeastcon*. 2013: IEEE.
26. Mahmoud, A. *An information theoretic approach for extracting and tracing non-functional requirements*. in *2015 IEEE 23rd International Requirements Engineering Conference (RE)*. 2015: IEEE.

27. Keramati, H. and S.-H. Mirian-Hosseinabadi. *Integrating software development security activities with agile methodologies*. in *2008 IEEE/ACS International Conference on Computer Systems and Applications*. 2008: IEEE.
28. Maiti, R.R. and F.J. Mitropoulos. *Capturing, eliciting, and prioritizing (CEP) NFRs in agile software engineering*. 2017: IEEE.
29. Dabbagh, M., S.P. Lee, and R.M.J.S.C. Parizi, *Functional and non-functional requirements prioritization: empirical evaluation of IPA, AHP-based, and HAM-based approaches.* 2016. 20(11): p. 4497-4520.
30. Asadi, M., et al., *Toward automated feature model configuration with optimizing non-functional requirements.* 2014. 56(9): p. 1144-1165.
31. Ebert, C.J.A.o.S.E., *Dealing with nonfunctional requirements in large software systems.* 1997. 3(1): p. 367-395.
32. Shah, T. and S.J.P.C.S. Patel, *A novel approach for specifying functional and non-functional requirements using rds (requirement description schema).* 2016. 79: p. 852-860.
33. Farid, W.M. *The Normap methodology: Lightweight engineering of non-functional requirements for agile processes*. in *2012 19th Asia-Pacific Software Engineering Conference*. 2012: IEEE.
34. Thakurta, R.J.S.Q.J., *A framework for prioritization of quality requirements for inclusion in a software project.* 2013. 21(4): p. 573-597.
35. Moreira, A., J. Araújo, and A. Rashid. *A concern-oriented requirements engineering model*. in *International Conference on Advanced Information Systems Engineering*. 2005: Springer.
36. Galster, M. and E. Bucherer. *A taxonomy for identifying and specifying non-functional requirements in service-oriented development*. in *2008 IEEE Congress on Services-Part I*. 2008: IEEE.
37. Hosono, S., et al. *Prioritizing service functions with non-functional requirements*. in *Proceedings of the 2nd CIRP IPS2 Conference 2010; 14-15 April; Linköping; Sweden*. 2012: Linköping University Electronic Press.
38. Sachdeva, V. and L. Chung. *Handling non-functional requirements for big data and IOT projects in Scrum*. 2017: IEEE.
39. Saadatmand, M., A. Cicchetti, and M. Sjödin. *Toward model-based trade-off analysis of non-functional requirements*. in *2012 38th Euromicro Conference on Software Engineering and Advanced Applications*. 2012: IEEE.
40. Badr, Y., et al. *Enhancing web service selection by user preferences of non-functional features*. in *2008 4th International Conference on Next Generation Web Services Practices*. 2008: IEEE.
41. Mylopoulos, J., L. Chung, and B.J.I.T.o.s.e. Nixon, *Representing and using nonfunctional requirements: A process-oriented approach.* 1992(6): p. 483-497.
42. Farhat, S., G. Simco, and F.J. Mitropoulos. *Refining and reasoning about nonfunctional requirements*. 2009.

AUTHORS PROFILE