Non-functional Requirements Elicitation in Agile Base Models

Article *in* Webology · January 2022

DOI: 10.14704/WEB/V19I1/WEB19135

3 authors, including:

Laxman Singh
Noida Institute of Engineering and Technology
62 PUBLICATIONS **203** CITATIONS

SEE PROFILE

Non-functional Requirements Elicitation in Agile Base Models

Devendra Kumar*
Research Scholar, Dr. A.P.J. Abdul Kalam Technical University, Lucknow, U.P., India.
Email: kdevbali@gmail.com

Anil Kumar
Department of Computer Science, B.I.E.T. Jhansi, U.P., India.
E-mail: solankibiet13@gmail.com

Laxman Singh
Department of Electronics & Communication, Noida Institute of Engineering and Technology, Greater Noida, U.P., India.
E-mail: Laxman.mehlawat2@gmail.com

Received September 06, 2021; Accepted December 07, 2021
ISSN: 1735-188X
DOI: 10.14704/WEB/V19I1/WEB19135

Abstract

The elicitation of non-functional and functional needs is one of the most critical jobs of a requirement engineer. This scenario involves the imposition of limits on non-functional needs, whereas functional requirements call for the operation of a system in order to carry out functionality. Over the last few years, agile software development approaches have gained widespread acceptance in the software industry as a problem-solving paradigm. Non-functional requirements (NFRs) are frequently cited as a point of contention in non-functional requirements (NFR) approaches. As well as functional requirements like speed and efficiency, security is desired, amongst a host of other things. Aspects like usability, security, and privacy must all be taken into account. Functional needs must be treated as though they were first-class under the current industry standard of practice. Functional requirements are distinguished from non-functional requirements by the fact that only implemented requirements can be evaluated. To give an example, this method attracts the attention of the system's end users to a critical defect in its architecture. Projects of this type frequently fail because to dissatisfaction among the target audience. If you'd like a great demonstration, consider the London Ambulance System. When dealing with non-compliance to the necessary degree of detail, it is feasible to raise the likelihood of software success this is the first study of its kind in its sector to bring attention to the most critical NFR issues. The problems that arise during the elicitation stage of requirement engineering in agile base models. It also outlines the techniques and strategies that are being considered. Proposed in the literature as a means of dealing with these problems.

Keywords

NFR, Agile Base Models, FNR, ASD, Reliability, Functional Requirements, Quality of Service (QOS), NERV.

Introduction

Enhanced customer satisfaction, demand changes every stage of development, frequent software modules delivery, and direct contacts between customers are all causes for the increasing popularity of agile approaches. The software requirements fluctuate in terms of the development of the project when agile methodologies are used. Non-functional requirements of the industry (NFR) are overlooked or only processed in phases of design and execution, whilst the principal functional requirements of the industry (FNR) are considered. Due to the user's unintelligible NFR approaches, they tend to overlook (Younas, M., et al., 2017). And the nature of agile techniques. NFR is not well defined in the development of agile software. It is often challenging to incorporate NFR into various stages of the software development process. The system's failure is frequently due to the inability to recognize NFR the London Ambulance System. Another research revealed the inability to define the NFR because of the organization's lack of technical and financial capabilities to comply with NFR. The NFRs are significant in the early stage of development, as they identify technology selection, hardware allocation and standards in the creation of software. In addition, NFR helps decide the process for software security, licensing and distribution. Problems in the domain of agile requirements have been identified, in particular with regard to absence of elicitation guidelines. In software development approaches, the approach for incorporating NFR is lacking in depth (Jarzębowicz, A., & Weichbroth, P., 2021).

Agile software development (ASD) is a method of developing software products that is iterative in nature. The word "agility" refers to the capacity to adjust to be flexible and to work in close cooperation with the client. An Agile method is predicated on rational values like as trust (Behutiye W., et al., 2017), responsibility (Behutiye W., et al., 2017), and loyalty (Aljallabi, B.M., & Mansour, A. 2015), among others. A little more than half of companies (Jarzębowicz A., Weichbroth P., 2021) are now using agile methods for change and transformation management and have been doing so for more than three years. Furthermore, according to the findings of a study performed in 2018 among software industry practitioners, 97 percent of those who responded said that they used agile techniques (Amorndettawin, M., &Senivongse, T., 2019). In reality, many studies (Kopczyńska, S., Ochodek, M., &Nawrocki, J., 2020) have shown the advantages of

adopting Agile methods, including an increase in team productivity, motivation, and discipline, as well as an improvement in overall software quality, to mention a few.

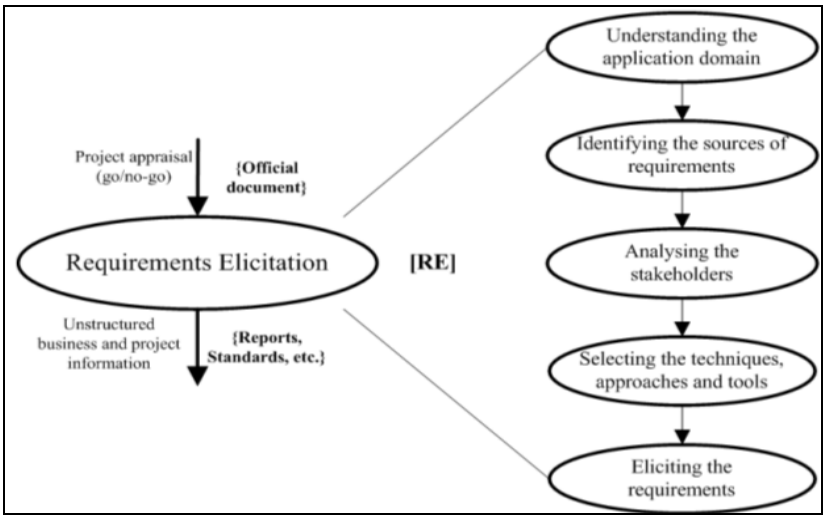

Figure 1 Requirements elicitation process (Ahmed M., Khan S.U.R., Alam K.A. (2021)

Indeed, software quality is an essential element to address throughout the software lifecycle (Asghar, A.R., et al., 2017), (Aziz, Y., et al., 2017), and it is often described in terms of high-level characteristics rather than lower-level attributes. Another option is to place extra restrictions on the system's ability to behave in a certain way. In other words, the necessary characteristics (attributes and constraints) are defined as non-functional requirements (hereinafter, NFRs), in addition to the functional requirements, in addition to the functional requirements (FRs). Since the inception of software development as a profession, non-functional requirements (NFRs) have been recognized as important elements influencing the adoption and usage of products by end users (Binkhonain, M., & Zhao, L., 2019). In fact, active user participation in ASD is essential in order to reduce the possibility of users' unhappiness as a result of misunderstanding or ignoring their expectations and requirements (Ramos, F.B.A., et al., (2019). The inevitable issue is whether this increased user involvement entails additional risks, and if the development team will be required to strike a balance between risks and rewards.

Since the publication of the Agile Manifesto (Ferreira Martins, H., et al., 2019), there has undoubtedly been a great deal of study conducted in an attempt to find an answer to this issue. In spite of this, few studies offer an evidence-based review and analysis on the topic of adopting NFRs, in particular on the problems that may emerge as ASD progresses, and on the techniques that have been recorded as effective facilitators. The ideals and concepts adhered to in ASD result in techniques that are distinct from those employed in more conventional software development approaches, such as agile development. For example, requirements engineering methods (Younas, M., et al., 2019) that presuppose ongoing

close collaboration with the customer (Gondal, M.; et al., 2020), place more emphasis on face-to-face communication and utilize less formal techniques such as collaborative games (Maiti, R.R., et al., 2018) are also included.

In order to do this, the purpose of this research is to examine and assess the current studies and their results while also providing a summary of the documented problems and applicable methods, to the degree of NFR identification and implementation, in the context of autism spectrum disorder. In order to offer evidence-based and up-to-date responses to the questions above, we performed a systematic literature review using the most recent available data.

Non- Functional Requirements

Non-functional (NFR) requirements, often known as quality requirements, are important to the software's success. It is extremely crucial to strike the correct balance between both functional and quality requirements. If these requirements are not met, the poor quality of the product is unavoidable. The NFR may be subjective and relative, which allows many persons to intersect and assess in various ways. They may also be interactive in nature, where attempts to attain one NFR might have good or negative effects on the aims of another NFR. It is quite crucial to deal successfully with them. These system characteristics are distinct and many academics have tried to categories them and establish several groups of such traits. Collective information on these qualities is given with a full discussion of what each means (Ramos, F.B.A., et al., 2018). A map of these links is also provided in Figure 1. The phrases Non-functional systems features, Non-functional requirements, and quality requirements are used interchangeably throughout this paper and their interplay with software development in Table 1.

Table 1 Non- Functional Requirements Interpretation

Non-Functional Property	Interpretation
Usability	Nonfunctional Requirements (NFRs) are qualities of a system that specify characteristics such as security, dependability, performance, maintainability, scalability, and usability, among others.
Integrity	The degree to which the intentional and unauthorized access to the system is restricted.
Efficiency	It refers to the effective usage of all the available resources.
Reliability	It is the extent to which the system works without failure
Maintainability	It is the ease of maintaining the systems throughout its life cycle.
Re-usability	It is the extent of reusing several components of the system.
Portability	It can be the extent to which the system can be moved in different environments or being able to run the software on different platforms.
Testability	It denotes the extent to which the system under test reacts to the testing process of creating and executing successful tests.
Interoperability	It is the degree to which the system or the individual components being interconnected for different operations.
Security	"It is the degree to which malicious harm to a valuable asset is prevented, detected and re- acted to." (Younas, M., et al., 2019).
Performance	Different timing characters of the software like Jitter, Throughput, Response Time, etc. are referred using this property.
Quality of Service (QOS)	It is extent to which the system offers its services with good quality being rendered to the user.
Availability	It is level to which the system is always up for the users to use.
Scalability	It is level with which the current system can be modified to a enhance the current capabilities.

Agile Software Development

Agile is capable of creating and responding to change. It's a strategy to cope with an unpredictable and tumultuous environment and finally flourish. For this overall notion, the creators of the Agile Manifesto picked 'Agile' because it symbolized adaptability and reaction to change, which was so fundamental to their methodology. It truly means contemplating how you can grasp what's happening in the environment, what uncertainty you face, and how you can adjust to it as you go along (Younas, M., Jawawi, D.N., Ghani, I., & Shah, M.A., 2020); (D.S. Gupta and G.P. Biswas, 2018).

Agile is a collection of beliefs and concepts. Agile is a way of thinking. Agile is a method to think and to behave. Agile involves short cycles, iterative and progressive delivery, quick failure, feedback, early and personally business value, collaboration and engagement. Agile is an attitude that concerns openness, examination and adjustment. Agile consists, however, of no roles, events or objects (Symeonidis, I., et al., 2019). It's a way of thinking. Scrum is for instance one of the frequently used frameworks under the Agile umbrella, which may assist you to become agile, but many other frameworks exist in the Agile movement, such Kanban, XP, Crystal and many more as the figure below shows.

According to the currently available research, the majority of studies focusing on NFR are still limited and incomplete. Even after all these years, integrating NFR into the software development process is still a challenge. As a result, researchers face numerous difficulties in their work, including a wide range of NFRs, detailed descriptions and characterizations of demands, the subjective nature of NFRs, and the resolution of differences among NFR models (Roy, M., Deb, N., Cortesi, A. et al., 2021).

Integration of NFR with FR

Despite the fact that non-functional requirements (NFRs) are widely recognized as the key success metric for project success, they get less attention in industry practices of software development than functional requirements do in academia. There are numerous recommendations available for sketching and modelling functional requirements, such as UML views such as use cases that may be found online. There is more work to be done in the future in terms of dealing explicitly with NFR and clearly specifying NFR in conjunction with FR (Werner, C., Li, Z.S., Ernst, N., & Damian, D., 2020). As a result of this informal approach, NFRs are often represented as "special needs" in use cases, which

can cause a variety of issues during the later stages of software development, including the inability to trace requirements back.

In contrast to functional requirements, non-functional needs should not be regarded the same way. When it comes to security and usability, for example, they shouldn't be approached together; instead, they should be considered separately. Numerous communities dealing with NFRs and other issues insist on it. A single approach or technique is difficult to adhere to when dealing with numerous NFRs at once.

1. Conflict with Requirements

Everyone involved in eliciting functional or non-functional needs has varied or even contradictory views and preferences most of the time. The NFR received from diverse stakeholders in a system will most likely differ. A new concept known as partial NFR satisfaction (satisfice), which describes solutions that are considered great despite the fact that they are not optimal (Rahy, S., & Bass, J., 2021), has evolved since it is difficult to tell whether an NFR has been met in practice. As a result, all of the derived dependencies, as well as the conflicts, should be identified. Furthermore, choices on how to resolve disputes should be made in accordance with the hierarchy of priorities.

NFR Techniques

According to current industry practice, only functional requirements are specified at the initial stages of software development, with non-functional needs being specified at the implementation level. As a result, the early stages (analysis and design) of a software system may not accurately represent the NFRs, resulting in a failing product. In order to address this issue, international standards suggest that NFRs be taken into account throughout the development process, starting with the phase in which requirements are defined in detail. Various methods, ranging from unstructured and casual language to very rigorous mathematical procedures, have been suggested. The method used in each instance is determined by the project's objectives as well as the available resources.

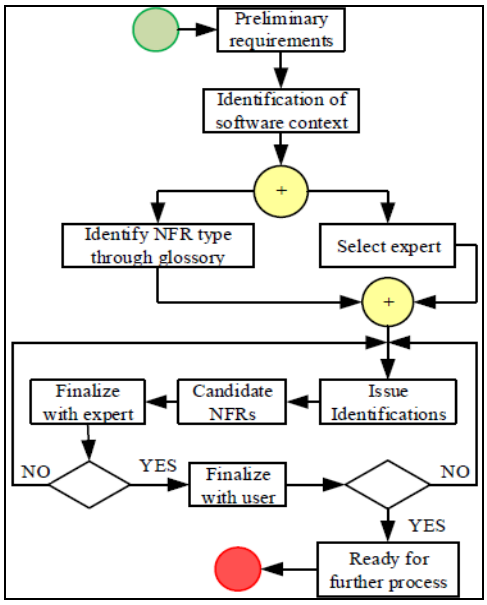

Figure 2 Requirements Elicitation Process (Kopczyńska, S., Ochodek, M., &Nawrocki, J., 2020)

Existing research (Sabir, M., Chrysoulas, C., &Banissi, E., 2020) have shown that this information may be used to determine NFR kinds. Additionally, ISO/IEC 9126 quality standard includes software engineering artefacts and mentions international standards for software engineering artefacts, such as ISO 25010.The quality standard contains the description of the quality criterion, but it does not include instructions on how to fulfil it. As depicted in Figure 2, a Software quality tree was identified by (Jha, N., & Mahmoud, A., 2019) Quality attribute classification depicts the tree. In addition, there is no defined standard that dictates their textual content. NFR elicitation is often conducted using Agile methodologies in a number of research. Some studies specifically concentrate on the elicitation of Agile requirements but not others. Furthermore, there are six other research that support the NFR use in non-agile techniques.

1. Automated Approaches

Scrum and Extreme Programming have gained popularity in recent years due to the popularity of agile software development (XP). Non-functional requirements (NFRs) in Agile methods seem to be either neglected or poorly specified. Although both functional requirements (FRs) and non-functional requirements (NFRs) must be considered, it is widely agreed that ignoring non-functional requirements is a leading cause of software project failure. For this reason, it is critical to spend extra emphasis on NFRs in Agile software development. It is vital to do requirements elicitation, implementing the software design (Younas, M., et al., 2020), developing, and validating it. User stories are a kind of

functional requirements that may be obtained by using excellent agile methods. Agile approaches have devoted little attention to elicitation of NFRs. Furthermore, validating and reasoning for NFRs has been scarce. “The proposed (Ameller, D., et al., 2019) methodology for NERV is called the ‘Non-functional requirements elicitation, reasoning, and validation in agile processes.' Currently, findings reveal that artefacts generated in this study might possibly assist firms using early Agile processes with managing non-functional requirements (NFRs).

NFRs are properties of software products that indicate how well the products fulfil their duties. These software features are fundamental and essential to any programmer in the market. They serve as the primary difference between competing software products, since all of them have comparable features. As a result of its growing popularity, agile methodology provides better software development practices, but it comes with restrictions when it comes to requirements analysis. Laxity about non-functional requirements is one of the greatest impediments to Agile project success. It offers no commonly acknowledged means for eliciting and managing needs that are considered non-functional. This research (Bhowmik, T., & Do, A.Q., 2019) proposes an upgraded technique for improved analysis of NFRs. Combining their strengths and overcoming their shortcomings, the new strategy is superior than the previous ones.

Functional requirements are given greater attention because agile development techniques see non-functional needs as being more significant. This is caused by the absence of established requirement elicitation techniques and the fact that the software development process used in agile software development is still in its formative stages. Focusing on too few solutions causes software projects to fail. According to the twelve agile principles, using cloud computing helps to practice the concepts of non0-functional demand elicitation. A non-functional need may be drawn from developing agile environments in addition to cloud computing environments. An NLP-based automated NFR extraction technique was utilized to rapidly go through the NFR elicitation process. In order to assess the approach, it is first applied to the eProcurement dataset. According to this research, the outcomes are enhanced by 8.77% and 1.76% in terms of “Successful” NFR. This drop is 7.02% and 1.75% as compared to prior research, in the “Partial success” category, and 1.76% to 0% in the “Failure” category.

Delivering and developing software solutions rapidly and effectively using an agile approach is called agile software development. Agile approach tends to focus on what we call functional requirements (FRs). It ignores non-functional requirements, which we will call non-functional requirements (NFRs) (NFRs). Neglecting NFRs may lead to worse

quality and greater rate of repairs in later stages of software development. This study is a part of the Capture, Elicit, and Prioritizing (CEP) technique. This article (Ahmed M., Khan S.U.R., Alam K.A., 2021) covers the Prioritizing phase of the methodology. The main aim of this study is to better priorities NFRs. The findings described in this research give guidelines for implementing an agile development process in the early phases of the agile development process. Using the αβγ- framework to priorities NFRs is an approach used by Capture-Elicit-Prioritize. The αβγ-structure offers novel possibilities to software developers. αβγ-interchangeable framework's qualities enable developers to be adaptable. Prioritization of NFRs is in agreement with the goal of rapidly producing agile software. This supports developers who are agile enough to organize their finances and schedule.

Agile software development has become popular for its ability to swiftly and effectively build software. However, this technique usually takes into consideration the FRs (Functional Requirements), as it's common for agile software development to include FRs, and it does not factor in the non-functional requirements (NFRs) (Yadav, S.P.. 2021). Ignoring NFRs leads to software products with worse quality and more costs to address bugs at a later point in the software development lifecycle. A recent study, presented in this article, calls for a research to conduct a pilot to collect NFRs information from artefacts such as papers and photos. To meet this goal, the criteria for identifying false positives will be expanded to encompass NFRs, as well as FRs, in the early phases of software requirements collection. Furthermore, this research will use historical patterns to forecast future NFRs that architects neglect and which might be included into agile software development earlier on. It is critical to stakeholders as well as software developers that NFRs be prioritized utilizing methods that are already in place for implementing high-quality software. This study is a follow-up to past research that focuses on new form factors in agile software development. This study attempts to expand on previous research on NFRs to aid in devising strategies for effectively identifying and predicting NFRs when agile software development is still in its early phases (Alflen, N.; Santos, L.; Prado, E. and Grotta, A., 2021).

The effective development and deployment of software necessitates identifying non-functional needs. Reaching agreement with the client on the software's non-functional needs is essential for their adoption of the product. To find these needs, we must identify all the non-functional needs that all stakeholders expect from our product. Among other things, few techniques are available for this purpose in the literature. From this, we conclude that we need to use a multi-layered strategy to find non-functional needs. It would have several benefits if we were to go with a layered strategy instead of a non-layered method. Other rules are recommended for usage in each layer as part of this

method. This methodology has already been successfully utilized on two case studies. Once the non-functional needs have been determined, they are verified by checking off a list and then a metric is used to measure the completion of those non-functional needs (Bouraga, S., et al., 2021).

In the modern day, software helps micro-companies to compete with larger firms. Everything from accounting software to point of sale software to e-commerce software and so on could be included depending on the application. Micro-businesses must provide the software developers with a detailed explanation of their needs before the programme can be completed. Micro-businesses must make some compromises in order to flourish due to a lack of resources and access to proprietary software. Using a pragmatic way of requirement analysis will be necessary if this is the case. This method must be low-cost, non-technical, and labor-free in order to be considered. Pentathlon Systems Resources Inc. applied an agile requirements elicitation technique to an arcade as part of a study (Trichaisri, S., 2021) and found it to be beneficial. A micro-software enterprise's requirements have been defined with the use of a variety of models and approaches such as goal models, business process models, patterns, and non-functional requirements.

2. Structured Elicitation of Non-functional Requirements

The benefit and cost of needs elicitation are not fully balanced, and as such it is still an open issue how to create a fair balance. One must have a good idea of what beneficial outcomes will be derived from using time and other resources for elicitation. We have an in-depth exploration of the Structured Elicitation of Non-functional Requirements (SENoR) technique in this article. A way to go about doing anything is to break it down into short, focused sessions where ideas are generated in a way that incorporates ISO 25010 quality standards. To help the process of elicitation, it employs Non-functional Requirements Templates (NoRTs) (Brataas, G., et al., 2021). Our exploratory case study on the cost and efficacy of SENoR and NoRTs included seven web application development projects that were specially crafted for the project. This experiment reveals that a set of two 1.5-hour elicitation workshops is capable of eliciting non-functional needs that represent a stable state level of 80%.

3. Modeling Non-Functional Requirements

Modeling Non-Functional Requirements (NFRs) is a challenge, mainly from the subjective nature of their elicitation. Modeling NFRs as collaborative approach may establish a consensus among stakeholders, combining. Modelling and elicitation in a learning cycle. However, the partnership provides difficulties, such as the various

perspectives, the impact of creativity, the modelling tool, the domain/scope of the target software system, and the basic elements of configuration management. This article presents our views on collaborating modelling a deliberate model for the management of fruits and vegetables n a futuristic kitchen. Agile software development focuses on rapid delivery and changing adaptability. Although agile methods are successful in providing acceptable functional requirements, they tend to overlook (Przybyłek, A., et al., 2021).

Non-functional needs until subsequent software phase's development. This study (Yadav, S.P., Zaidi, S., Mishra, A. et al., 2021) concentrates on the most popular Scrum. Agile approach and suggestion on non-functional needs Early support system for Scrum practitioners identification. The solution is built on the Scrum instrumentation Process for extracting valuable information and using collaborative filtering and the suggestion of the item. To assess the suggestions, we Data gathered from 12 Scrum were performed off-line trial Practitioners by use of an inquiry. The data were evaluated using 10- Cross-validation fold. This demonstrated our suggested solution A reminder rate of up to 81%, which shows that it is promising Approach to suggest a list of non-functional needs Functional needs specified by stakeholders in the project.

In creating software products, it is necessary to consider both functional and non-functional requirements (NFRs). Requirements engineering (RE) is the early phase in which software development activities create and manage system requirements. The RE process comprises requirements generation, analysis, documentation, validation and management. Most work in the RE sector nevertheless primarily focuses on functional requirements, but NFRs are frequently more important than functional needs. In this article, we will examine the state of the art on how NFRs are handled when the RE process is carried out. A set of criteria covering the key activities of the RE process is thus established as a systemic method for comparing various RE process models. The criterion list is compared with six existing RE process models followed by suggestions for the new NFR-supporting RE process technique (Ramkumar, R., & Mala, G.A., 2021).

Design engineering is extremely essential in the field of software engineering. Roughly 70-85% of the expense of project revisions is spent on identifying and correcting requirement flaws. Requirement engineering found it especially challenging to handle Non-Functional Requirements (NFRs). Failure to use NFRs leads to the most costly and complex mistakes that can only be fixed after the system is complete. Fast identification and recognition of NFR enhance the chances of success of the project. This article is an attempt to compare the different techniques for the extraction and classification of NFRs by machine. It also creates gaps in literature for future concerns.

4. Scrum Methodology

The needs of software are important. The quality of the requirements is defined by the interaction between functional and non-functional demands, and this, in turn, influences the quality of the programme at the end. Non-functional requirements (NFR) receive less attention during the Requirements Engineering (RE) process than functional requirements (FR) (Yadav, S.P., Yadav, S., 2019). You must know the characteristics and criteria used to determine the link between free radicals (FR) and net free radicals (NFR) for determining NFR (Functional Requirements). To calculate NFR through stakeholder interviews, many software developers are now using the Agile Methodology, which they refer to as the Scrum Methodology. NFR attribute mapping, how to collect NFR characteristics, and how to represent them closely with FR are all taken into consideration while considering the importance of NFR attributes. In this study, the methodology is based on the Prisma Framework. Results of this research develop domain NFR attributes, identify NFR attributes, classify NFR attributes, and represent NFR attributes in relation to FR in order to obtain application features.

The necessary automation in all fields makes software an important component of our life as shown in Table 2 for requirement's parameters. The development of a software requirement plays a major role. In requirement engineering, priority must be given to delivering the software system successfully. Numerous software prioritising methods exist, such as cumulative voting, cumulative hierarchical voting, planning games, MOSCOW etc. However, these methods are assessed to be inadequate to priority a large number of needs and the requirement engineer in most instances just prioritises the functional requirements and disregard non-functional requirements. The priority of Non-Functional Requirements (NFRs) is a challenge to justify the higher priority requirement and functional requirements in large software systems. In order to address this gap, we have suggested a new "PGAHP" method to priority requirements by fusing planning games with analytical hierarchy (AHP) ideas. The proposed PGAHP approach is validated using the case study of the library management system. Results indicate that the difficulty of prioritization is reduced with the use of "PGAHP" by a comparison of thirty-one (31) pairs with AHP that has a pair of (105) functional and non-functional requirement comparisons (Kumar D., Dhir S., 2022).

RE frequently struggles with difficulties like as ambiguity, incompleteness and inconsistent data towards an effective and high-quality software development process. In this respect, the objective of this study is to evaluate the contribution of RE combined methods in the course of information systems higher education (IS) to I functional

requirements elicitation and ii) non-functional requirements (NFR). Through a Systemic Literature Review (SLR) 39 papers were thoroughly examined and the RE list ultimately produced from the Scopus database that met the RE search parameters. In the IS course case study, 56 students were then supported by the top three REs (interviews, prototyping, and brainstorming). Combined FR and NFR methods have shown that RE isolated enhanced completeness and consistency in comparison with each single technique.

Table 2 Requirements parameters

Accessibility	There being an abundance of people available when they're needed.
Accuracy	Easygoingness is a personality trait that makes people like them (WordNet).
Auditability	An assessment of the degree of correctness or the lack of mistakes in anything. IEC and IEEE A metric used to assess the degree of an error.
Availability	It is called availability the degree to which a system or component is operational and accessible when it is required to be used. Most of the time, it's shown as a percentage (IEEE 610).
Compliance	A software product's compliance ability is a measure of how well it follows a product's specifications match the requirements of a certain standard (ISO 9126).
Confidentiality	A characteristic associated with the informative field Keeping confidential information safe means making sure it's not exposed to or divulged to anyone who shouldn't have access to it. Individuals and processes are both considered entities when using the term "entities" (ISO 27001).
Configuration	Component or system composition can be described by the number, nature, and connections between its constituent elements (IEEE).
Documentation	A group of documents devoted to a single subject; a library. Data that describes, identifies, specifies or reports on an event and is certified by a third-party data collection agency (IEEE).
Efficiency	Capability refers to a software product's ability to produce enough performance for the number of resources it consumes under specific conditions (ISO 9126).

Cloud computing is rapidly evolving as an alternative in industrial practice to formulate cost-effective and needs-oriented information systems. Although cloud computing in the business is becoming accepted, many key issues remain unresolved or handled only partly. In addition to questions about optimal designs, legal problems and price models, cloud-based solutions providers also confront the challenge of proper requirements engineering. This implies an optimal knowledge of the customer's needs and the implementation of the relevant solutions. This paper analyses chosen known requirement engineering techniques to assess the degree to which the particular needs of cloud-based solutions may be addressed. It also provides a comparison framework with cloud computing capabilities. This comparison framework is applicable to four known requirement engineering process models. Recommendations for a cloud-adapted requirements engineering system are developed.

5. Capturing Prioritizing

Non-functional requirements (NFR) are ignored, while functional requirements (FRs) in the development of agile software are typically more preferred. Neglecting NFRs has detrimental effects that result in poorer software quality and costly processes later in the

development of software (Jarzębowicz A., Weichbroth P., 2021). Developed CEP "Capturing Prioritizing" technique for retrieving Non-Functional Requirements from the document and diagram/image requirements in the documents. This study has been validated using 26 European Union eProcument requirements documents. The baseline statistics are based on the NORMAP technique as prior studies. To compare the data, the NERV technique was utilized. In order to enhance the results of earlier research, the CEP approach connected NFR Metadata (NFRM) with pictures and figures discovered in requirements documents to comparable needs.

In early phases of software development, FRs are prioritized in the current plan, but NFRs can only be considered in the latter stages of software development because they cannot meet user needs right now. To merge NFRs and FRs, a great deal of work has been done since the beginning of software development. These methods include goal-driven approaches and Chung's NFR framework, pattern-based approaches, UML, and visualization techniques, as well as research to show that NFRs are important in relation to FRs. This research examines NFR methods in the following categories: goal-driven approaches and research to show that NFRs are important in relation to FR. Optical Character Recognition (OCR) and historical pattern recognition, along with an OCR recognition component, round out the other features (Aljallabi, B.M., & Mansour, A. 2015). Methodologies presented in the literature may overlap, making it difficult to classify them precisely.

Although performance, usability, stability, security, and scalability are all critical criteria for business-oriented software, they are usually addressed ad hoc during system test phases later in the development process. It's possible that NFRs will be required across the SPL when a comparable request appears across all product lines and the needs change; this is when the parameterized or alternative feature will be implemented. NFRs will be required. NFRs may be required in the future. NFR analysis frameworks were created to address interdependencies between non-functional requirements and functional requirements, as well as application engineering to select and characterize non-functional and functional requirements, in accordance with the strategy. A requirement engineer could model both non-functional and functional requirements has shown in Table 3. All needs are treated as objectives in goal-based modelling, the modelling technique employed. NFRs are considered soft targets and two AND/OR trees can be constructed to display targets for NFRs, one for FRs and NFR. The connection is shown as a straight diagram, with nodes as objectives, soft goals in the target nodes and rims as + or + characters.

Table 3 Non-Functional Requirements List

Accessibility	Access Control	Accuracy	Adaptability
Adjustability	Affordability	Agility	Auditability
Audit	Availability	Buffer Space Performance	Capability
Capacity Clarity	Code-Space Performance	Cohesiveness	Commonality
Communication Cost	Communication Time	Compatibility	Completeness
Component Integration Cost	Component Integration Time	Comprehensibility	Conceptuality
Conciseness	Confidentiality	Configurability	Consistency
Controllability	Coordination Cost	Correctness Cost	Coupling
Customer Loyalty	Customizability	Data-Space Performance	Decomposability
Domain Analysis Time	Dependability	Development Cost	Degradation of Service
Development Time	Diversity	Domain Analysis Cost	Efficiency
Elasticity	Execution Cost	Extensibility	External Consistency
Fault-Tolerance	Feasibility	Flexibility	Formality
Generality	Guidance	Hardware Cost	Impact Analyzability
Independence	Inspection Cost	Inspection Time	Integrity
Inter-Operability	Internal Consistency	Intuitiveness	Learnability
Legal	Look & Feel	Main-Memory Performance	Maintainability
Maintenance	Maintenance Cost	Maintenance Time	Maturity
Mean Performance	Measurability	Mobility	Modifiability
Modularity	Naturalness	Observability	Off-Peak-Period Performance
Operability	Operational	Operating Cost	Peak-Period Performance
Performance	Planning Cost	Planning Time	Plasticity
Portability	Precision	Predictability	Process Management Time
Productivity	Project Stability	Project Tracking Cost	Promptness
Privacy	Proto typing Cost	Proto typing Time	Re-Configurability

Nodes in the AND tree represent the following: the feature or NFR; the priority node is included in the requirement's requirements; if a feature has no impact on NFR, the priority of the feature is nil, and its child priority is equal to or lower than its parent priority. Nodes in the AND tree represent the following. As part of the enhanced UML approach, PLUS adds performance criteria throughout several SPL modelling processes. Because NFR performance is now the primary focus, advocated that the PLUS technique be used to other NFRs in addition to those listed above. Provide ratings for NFRs that are categorized as high, medium, or low for rating purposes, including discrete satisfaction rates The safety NFRs can also be improved to ensure protection levels in line with the NIST standard's acknowledgement of the data protection level stated. There are three new stereotypes to support NFR that presented as an alternative: common, optional, and NFR-parameterization (Ferreira Martins, H., et al., 2019).

Although this strategy is effective, it has the drawback of being narrowly focused on one particular NFR. This is because NFRs can vary widely in terms of their efficacy. In addition, one NFR may have an impact on another NFR or even on the FR. This technique has a lot of potential benefits because it combines NFRs with current design tools like UML. Developers and software architects are familiar with these technologies, and using similar tools will make integrating NFRs with FRs easier. Expanded PLUS method, according to, had only one NFR and its performance was the only limitation. There's a hole in this research because not all NFRs are created equal; not all NFRs are created equal. After that, similar studies are evaluated.

6. Sami Automated Approaches

Non-formal software requirements (NFR) are not defined by the IEEE or the ISO, while formal software requirements (FR) are formal requirements for a long time, controlling NFRs was a difficult task, and it was widely criticized for its inconsistency, since one NFR may help functioning while another NFR may hamper it. It's critical to keep in mind that NFRs are purely arbitrary. Users and system engineers may have different ideas about how fast a system should respond. As a result of NFRs' subjective nature, tools and strategies for managing NFRs are evidently employed. Using artificial intelligence, Chung's NFR framework manages NFRs. This is a process-driven solution he devised. Chung's NFR framework portrays NFRs as softer goals rather than explicit ones NFR interdependencies and system functionalities are connected in graphical form in Chung's NFR Framework's core structure, which is represented by soft objectives that satisfy and operationalize systems functions. Using Chung's NFR Framework, system functions are operationalized and soft targets are satisfied. It provides system functionality and ensures that software functions in line with software operation labels thanks to the Soft goal Interdependency Graph (SIG). It is necessary to meet the following NFR goals. Many other terms have been coined to characterize people who aren't totally content, such as Satisfied, Weakly Satisfied, and Satisfied Plus. Other terms include Weakly Denied, Unknown, and Conflict Process-driven adaptation can be used to determine the optimal set of system functions that meet an arbitrary number of NFRs when applied to Chung's NFR framework, as demonstrated. It is a novel technique called Soft goal Interdependency Rule Set (SIRGs) that reflects the NFR System's functions and automatic optimization connections.

There is a chance that SIRG will get more features in the future, such NFR priority levels at the Top-Level and total volume v weight as NFR priority levels. A wide range of resource characteristics, including development time and cost, maintenance and development obstacles, and dangers encountered during the development process, might reflect soft aims. Unfortunately, all NFRs are treated the same, and thus creates a gap where NFRs aren't exactly the same, which might lead to confusion. While this technique had flaws, it did have one strength: it assigned a cost, a priority, and weight to NFRs. Therefore, the dangers of classifying NFRs separately are considered. The Chung NFR framework is then employed in conjunction with the approach being evaluated. Many projects fall short because of a lack of NFR management competence. There are issues with using NFRs in the late design phase or early design phase. The NFR framework for Chung was developed to overcome this problem. To bridge the gap between NFR and design needs, Chung created an NFR framework. Although the Chung NFR paradigm was

elaborated upon, no quantitative evidence was found to support or refute it in the investigation. Expansion of the NFR Chung will require the following actions: identification of soft targets, decomposition of soft targets, assignment of leaf target weights, identification of operationalization and calculations, identification of operationalization, calculation of operationalization scores, calculation of soft target scores and calculation of success. The simulation criteria or decisions made by the developers during the simulation satisfied the differences discovered in the expanded Chung NFR framework. The extended Chung NFR framework's differences (Kumar D., Dhir S., 2022). A similar approach can be implemented to every system; however, in this case, it was not essential because individual mapping of goals and operationalization did not benefit from the extension's influence on decisions and developers benefited from SIG's more compromises than other systems because of this (Przybyłek, A., et al., 2021). In order to maximise the leaf objective, soft target, and achievement scores, the operationalization selection technique can be tweaked to enhance the amount of effort, time, and money spent on a successful software project. It shows the strength of the methodology that Chung's expanded NFR framework may be applied to any system studied by However, it is unclear from the study if any additional or specific NFRs have been explored in the past, indicating that not all NFRs are created equal and hence cannot be treated in the same way. The identification and classification of certain NFRs appears to be the study difficulty in many research enquiries that follow a similar pattern. This study's findings are being considered and evaluated in light of future work.

More comprehensive NFR research has to be done to include many kinds of NFRs, including safety. The trends in these methods are based on the goal-based NFR framework of Chung, where each author extends each approach from the NFR framework of Chung. The current tendency for each of these methods was to detect and categories each NFR in which NFRs seem to be wide or selective in each of the given strategies. Next, the NFRs are analysed using a pattern-based method. Dealing with NFRs needs a wide range of NFR expertise.

Security and confidence are frequently referred to as "soft objectives" because of their subjective nature. In general, there are four different kinds of patterns: pattern selection pattern, design pattern for problem, alternative pattern, and goal-oriented pattern. a visual representation of a typical NFR knowledge is used to express an NFR pattern in visual form. If the stakeholder defines an NFR pattern as subjective, objective modelling can be used to record numerous NFR definitions visually and unambiguously, where the stakeholder can reuse the NFR pattern by using the specific soft goal defined in objective modelling. In order to capture soft problem information, the issue pattern is utilized. To

deal with the subjective and contradictory NFRs that develop when attempting to fulfil the goal of a soft issue, it is possible to depict it in a different manner. Using these patterns, selection systems may be captured that facilitate more automated and systematic decision-making Connection specialism, relationship component, and relationship occurrence are provided in this order: A section of the specialization relationship is used to integrate smaller patterns with bigger pieces of information when using the specialization connection. One or more soft objectives or soft concerns are handled during the pattern operation. "Soft" and "soft difficulty" are also used in the procedure. There is a soft-goal or a soft-problem known as NFR that needs to be honed into a specific pattern during the application process. The specialized procedure permits visible model expansions in a tool environment based on models. One or more pre-existing patterns must be used in the composition process in order to build a new composite pattern. The instantiate operation accepts a model instance and uses the model to bind requirements. An addition to StarUML that lets users see NFR patterns, including an objective pattern for defining the NFR's purpose, a problem pattern for identifying barriers that must be avoided, alternative patterns for recording solutions, and selection patterns for picking the best option. It also enables the visualization of inter-pattern relationships, including specialization, composition, and instantiation. To collect refining rules, an NFR pattern visualization framework is employed. These rules are then applied to the reused target model. Establishing pattern connection integrity restrictions is accomplished using refinement rules. The refinement rules are then compiled using the framework.

To make matters more complicated, as the number of NFR patterns increases, the NFR pattern will become increasingly complex. There are other NFR patterns that rely on this one. One of the most important strengths is the ability to reuse patterns in software development and design. Stakeholders and developers can better understand NFR requirements thanks to the patterns' visual portrayal. For example, point out that the Chung model fails to consider other phases of software development, such as architecture and design. Other phases of software development would be taken into account by this methodology. Similar studies are then conducted as a result (Siakas E., et al., 2021).

Because most software engineers and stakeholders are already familiar with UML, it's easy to use. Unified Modeling Language (UML) SysML may be developed to incorporate complicated systems engineering demand modelling applications This study widened the SysML meta-scope models to include NFRs, and it also placed a heavy emphasis on NFRs' effects on FRs. Afterwards, a SysML subclass, which is a subclass of the meta-classes, accumulates information on non-functional goals.

Fundamental or abstract non-functional aims are both examples of non-functional objectives because the NFG is more subjective than the FG, finding and developing a solution to meet the basic NFG idea of contribution will be necessary to effectively complete the refinement process. With regard to contribution features, there are two types: explicit or induced, which are defined by the contribution feature association; positive or negative contributions, which are defined by the contributions type. The contribution features are established via the Contribution-Feature Association.

User evaluations collected from mobile application (app) stores typically contain technical information that might be valuable to app developers when designing mobile applications. Information mining and classification have become increasingly important in software maintenance requests like bug reports and functional feature request. Recent research has focused on these topics. There has been very little work done towards identifying the Non-Functional Requirements (NFRs) and extracting and synthesizing them. When it comes to software quality, no requirement should fail. NFRs are a set of high-level quality requirements that a software system must meet (e.g., security, performance, usability, and dependability). It's vital to meet these criteria if you want to acquire customer pleasure and succeed in the mobile app industry. A two-phase study that would mine NFRs from user evaluations available on mobile app marketplaces will be proposed in this paper in order to close the gap. 6,000 user reviews from a variety of iOS app categories were used for the first stage of our qualitative research process. 40 percent of the reviews included in our dataset have at least one sort of NFR implicated in them, according to our findings. The findings also indicate that users in different app categories prefer to raise different types of NFRs, which is consistent with the findings. To collect NFRs from user evaluations automatically, a better dictionary-based multi-label classification system is created in the second stage.

Assessments of the Review

Current research blends agile project management with other project management facets. This research gaps. NFR methods that can be changed on the fly. Its Efficacy the North American Requirements Plan is working on a brand-new risk quality project management metric (NORPLAN). A risk-based algorithm created by the NFR integration and prioritization team can be used by project managers and SCRUM teams. The NORMAP approach also considers technological risks and project threats (Siakas E., et al., 2021). NORMAP and ISO 9001 have similar quality requirements. Calculating risk is essential in agile planning. The case study's scope was restricted to the UK's procurement system.

Table 4 Comparison of Techniques vs Proposed Method

Sr	Title	Research Objective	Research Question	Proposed Method
1	Eliciting Efficiency Requirements with Use Cases	It is proposed that efficiency requirements be elicited via the use of use case scenarios. (Kopczyńska, S., Ochodek, M., & Nawrocki, J., 2020)	• A very high degree of connection between FR, AOs, and NFR was elicited and established. • Separation of concerns amongst stakeholders in the elicitation of NFRs and FRs; compact description of the solution space; tractability between FRs, NFRs, and AOs'	1. To identify broad information about NFRs, quality models and attribute types may be used to identify general knowledge about NFRs, while a template can be used to identify particular needs for NFRs. 2. Checklists may be used to elicit non-functional requirements (NFRs) related with the system's design and use cases.
2	A Framework for Incorporating Non-Functional Requirements into Conceptual Models is shown here.	A methodology for integrating NFR into Object Oriented and Entity Relationship models is presented in this paper. (Amorndettawin, M., & Senivongse, T., 2019).	It is very difficult to detect NFRs since no one at either end of the process (stakeholders or requirement engineers) is familiar with dealing with such issues.	A proposed No-Functional model is based on a mix of ethnography and LEL and is described in detail below. Using LEL to catalogue the knowledge of NFRs; using a knowledge base in conjunction with appropriate elicitation techniques for the elicitation of NFRs; and representing NFRs by modifying NFRs are the approaches used.
3	**The NERV approach is a technique that**	According to the literature, functionality is the main emphasis of Agile processes, while non-functional requirements (NFRs) are either neglected or poorly specified in these processes.	Requirements It is critical to understand the activities involved in software development, including elicitation, design, development, and validation. Agile methods are effective at eliciting functional requirements, which are then recorded as user stories. However, in Agile processes, the elicitation of NFRs has not received nearly enough consideration. Furthermore, there has been a deficiency in the rationale and validation of NFRs.	According to this study, the "NERV Methodology: Non-functional Requirements Elicitation, Reasoning, and Validation in Agile Processes" should be implemented. According to the current findings, the artefacts created in this study have the potential to assist software development companies in addressing NFRs during the early stages of Agile processes.
4	Guideline for the Elicitation of Non-Functional Requirements in Agile Methodologies	elicitation rules for non-functional requirements in agile techniques.	The Non-Functional Requirement has received less attention than the Functional Requirement, according to the majority of research (NFR). The user and developer's lack of understanding of NFR elicitation is the cause of their carelessness with NFR elicitation.	In this research, we offer elicitation recommendations for non-functional requirements in agile methodologies. This guideline will benefit both developers and users who work with agile methodologies. With the assistance of elicitation guidelines, a case study is performed on a group of master students in order to elicit non-verbal feedback (NFR). Additional findings were acquired by extracting NFRs from an eProcurement document that included requirements for important European Union projects, which was used to generate the original results. The outcome of the case study is good and encouraging for new developers and users who are less familiar with NFRs as a consequence of the research. The research also discusses the importance of cloud computing in agile techniques, namely in the elicitation activity. Additional findings include:
5	QA-Extractor is a Quality Attributes Extraction Framework for	Many software projects fail as a result of a lack of knowledge of the quality of the programmer at the early phases of its development. Non-functional	How to elicited NFR using technique which make it easy?	For the extraction of essential quality characteristics from functional requirements, a framework called QAExtractor has been created (i.e. user stories in Agile-based Software

	Agile-Based Software Development that was developed by the Agile Software Development Foundation.	requirements have a significant impact on the overall quality of the programmed. During the requirements elicitation stage of the software development life cycle, however, the requirements elicitation team pays less attention to non-functional needs than they do to functional requirements. In agile-based software development, the requirements elicitation team is only concerned with the user stories that are being developed (i.e. functional requirements in an agile-based software development context). The non-functional needs are not taken into consideration, and the team is left with just the user stories to work with. Because they are holding user stories in their hands, it is very difficult for them to make quality choices (Aziz, Y., et al., 2017).		Development context). The Natural Language Processing (NLP) framework lies at the heart of this approach. Regulated expressions, which generalise user stories for a particular quality factor and attribute, are the foundation of the suggested method. The quality factor, which is security, provides the context of the user narrative, while the quality attribute, which is integrity, indicates the quality aspect, which is security. An empirical case study is used to demonstrate the efficacy of the suggested method.
6	In order to detect evolving non-functional requirements, a multi-layered desires-based framework has been developed.	For the purpose of requirements elicitation, combined techniques are used.	The requirements elicitation (RE) phase is one of the most challenging stages to complete in the requirements engineering domain. In the pursuit of a successful and high-quality software development process, RE often encounters information problems like as ambiguity, incompleteness, and inconsistent data, among others.	At an Information Systems Higher Education (IS) course, this study will examine the contribution of RE combined methods to both I the elicitation of functional requirements (FR) and ii) the elicitation of non-functional needs (NFR) in the context of information systems. A systematic literature review (RSL) was conducted, in which 61 articles crawled from the Scopus database that met the RE search criteria were thoroughly examined, and the final list of REs was produced.
7	In Agile development, elicitation of non-functional requirements is accomplished via the use of a cloud computing environment.	Non-functional needs are given less attention since functional requirements are seen as more essential in the context of agile development methods than non-functional requirements. This is owing to a lack of established requirement elicitation methods, as well as the nature of the agile software development process used in software development. Because of the lack of attention, there were few solutions in the area, which resulted in the collapse of the software project (Przybyłek, A., et al., 2021).	How to create a semi-automated approach which elicited NFR?	As a result of the findings of this research, a semi-automated technique for eliciting non-functional needs in agile development and cloud computing environments has been suggested. To expedite the elicitation of NFRs, the technique used an NLP-based automated NFR extraction strategy to accelerate the elicitation process. The technique is tested by applying it to a dataset from the eProcurement system. The results have improved by 8.77 percent in terms of "Successful" NFR and by 1.76 percent in terms of "Failed" NFR. When compared to previous research, it has dropped by 7.02 percent and 1.75 percent in terms of "Partial success," and by 1.76 percent and 0.0 percent in terms of "Failure." It has also declined by 1.76 percent and 0.0 percent in terms of "Failure."

It is vital to include non-functional requirements (NFRs) including security, reliability, and performance when trying to determine if a system is successful or not. An additional structure and foundation were offered for the construction of an NFR design level by which decomposes distributed software architecture at an early stage of design (Binkhonain, M., & Zhao, L., 2019). Construction can begin at any point of design development from conceptual design to final product design. High-level abstraction is the first step in decomposition. Then come decomposition layers of functionality and lastly technical abstractions. Architectural models and a common vocabulary for distributed systems can be developed using high-level modelling abstractions found at the framework's top; middle and second-level frameworks are distributed, and the second level is made up of low levels of abstract functionality attached to decomposed layers. The researchers explained how to put the framework into action using safety criteria and the fundamental safety analysis procedure. Dependability and performance NFRs can be processed through the framework as well. The framework can enable dependability and performance in terms of both quality and quantity. None of these points were discussed in the framework presentation. Developers without extensive safety experience should be provided with a predetermined attack list as part of the security scope. Those who don't have a strong background in safety would benefit most from this. Although reliability and performance have been supported, there has been some criticism voiced about the Chung NFR framework's lack of testing. Because various NFRs differ and are interdependent on one another, they should not be treated the same manner. As a result of potential conflicts with NFRs and FRs, a precise categorization of NFRs must be established. The level of security is increasing with each passing day.

Gaps and Gainsays in NFR Elicitation in Agile Base Models

Using Ruby on Rails, NFR data is shown in the web framework and workbench. The ease with which web frameworks may be built was a major factor in choosing the Ruby development platform. Using the NFR Locator categories outlined above, as well as additional NFRs groups utilized in NORMAP and NERV programmes, Table 4 illustrates the NFR information (NFRM) that was collected and visually categorized. There are 18 different sorts of NFR Locators to consider while using this methodology. The data came from the NFR's metadata database, which may be found online. In Table 4, you'll find a list of NFR metadata groups, along with descriptions and priority. Like we said before, the system uses the NFR Locater to classify words into various NFR categories by parsing natural language text into an internal representation within the NFR metadata.

Security for any software is now almost unavoidable. To this end, the software's safety requirements should be efficiently modelled. However, current modelling languages such as the Unified Modeling Language have limits in non-functional modelling needs such as safety. Most current software is hosted on the internet or in the cloud with a significant flow of important information between many users. Security is an obvious requirement in this context. This article offers a technique for generating safety requirements for all stakeholders involved, assessing the degree of safety necessary for all software assets and presenting this security evaluation via simple yet effective graphics.

While UML or Unified Modeling Language is a popular language for software modelling needs, it is mainly helpful for functional requirements and offers little help for non-functional requirements, such as security. In the current situation, where the usage of internet and cloud-based apps is growing, these criteria are even more important. This article proposes Rank Diagrams that follow a well-structured process to get needs and to ranking, and represent these non-functional requirements to lead to improved system design.

Conclusion

It is an important topic in requirements engineering since it is critical to the success or failure of a system. Non-functional requirements are requirements that do not have any functional needs. Unfortunately, NFR issues are often addressed during the design process and at the level of execution, and this approach results in the failure of the majority of the systems It was necessary to identify concerns, problems, and obstacles while also generating responses. NFRs. This is the first attempt of its type to conduct a literature survey look for NFR problems and possible solutions during the conference. The elicitation level of requirement engineering is a subset of requirement engineering. Author has what we need. Highlighted a number of important problems such as conflicts of interest unclear requirements, integration of NFR with FR, and integration of NFR with Specification of the system's characteristics. Some of them have been discovered by us the answers to the issues mentioned above that are based on the material that is now accessible. The NFR framework (Binkhonain, M., & Zhao, L., 2019) serves as the foundation for the majority of the literature as opposed to a more formal method, and there is a need to provide a formal structure taking into consideration the points raised the answer presented in this article is based on the literature currently accessible. In the future, we would want to rebuild the framework and implement new technologies remove the problems that have been mentioned.

References

Younas, M., Jawawi, D.N.A., Ghani, I., & Kazmi, R. (2017). Non-Functional Requirements Elicitation Guideline for Agile Methods. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC), 9*(3-4), 137–142. https://jtec.utem.edu.my/jtec/article/view/2933

Jarzębowicz, A., & Weichbroth, P. (2021). A Qualitative Study on Non-functional Requirements in Agile Software Development. *IEEE Access, 9,* 40458-40475. https://doi.org/10.1109/access.2021.3064424

Behutiye W., Karhapää P., Costal D., Oivo M., Franch X. (2017). Non-functional Requirements Documentation in Agile Software Development: Challenges and Solution Proposal. In: Felderer M., Méndez Fernández D., Turhan B., Kalinowski M., Sarro F., Winkler D. (eds) Product-Focused Software Process Improvement. *PROFES 2017.* Lecture Notes in Computer Science, 10611. Springer, Cham. https://doi.org/10.1007/978-3-319-69926-4_41.

Aljallabi, B.M., & Mansour, A. (2015). Enhancement approach for non-functional requirements analysis in Agile environment. *International Conference on Computing, Control, Networking, Electronics and Embedded Systems Engineering (ICCNEEE),* 428-433.

Jarzębowicz, A., & Weichbroth P. (2021). A Systematic Literature Review on Implementing Non-functional Requirements in Agile Software Development: Issues and Facilitating Practices. In: Przybyłek A., Miler J., Poth A., Riel A. (eds). Lean and Agile Software Development. LASD 2021. *Lecture Notes in Business Information Processing, 408.* https://doi.org/10.1007/978-3-030-67084-9_6

Amorndettawin, M., &Senivongse, T. (2019). Non-functional Requirement Patterns for Agile Software Development. *In Proceedings of the 3rd International Conference on Software and e-Business,* 66-74. https://doi.org/10.1145/3374549.3374561

Kopczyńska, S., Ochodek, M., &Nawrocki, J. (2020). On importance of non-functional requirements in agile software projects—a survey. *In Integrating Research and Practice Software Engineering, Springer, Cham,* 145-158.

Asghar, A.R., Tabassum, A., Bhatti, S.N., &Jadi, A.M. (2017). Impact and challenges of requirements elicitation & prioritization in quality to agile process: Scrum as a case scenario. *In International Conference on Communication Technologies ComTech,* 50-55.

Aziz, Y., Aziz, T., Malik, M.I., Baig, M.K., Ali, M.Z., &Baqer, M. (2017). Non Functional Requirement in Agile Software Development. *University of Engineering and Technology Taxila. Technical Journal, 22*(1), 107-112.

Silva, A., Pinheiro, P.R., Albuquerque, A., & Barroso, J. (2017). Evaluation of an approach to define elicitation guides of non-functional requirements. *IET Software, 11*(5), 221-228. https://doi.org/10.1049/iet-sen.2016.0302.

Binkhonain, M., & Zhao, L. (2019). A review of machine learning algorithms for dentification and classification of non-functional requirements. *Expert Systems with Applications: X, 1.* https://doi.org/10.1016/j.eswax.2019.100001

Gupta, D.S., & Biswas, G.P. (2018). Design of lattice-based ElGamal encryption and signature schemes using SIS problem. *Transactions on Emerging Telecommunications Technologies, 29*(6).

Ramos, F.B.A., Pedro, A., Cesar, M., Costa, A.A.M., Perkusich, M., de Almeida, H.O., & Perkusich, A. (2019). Evaluating Software Developers' Acceptance of a Tool for Supporting Agile Non-Functional Requirement Elicitation. *In SEKE,* 26-42.

Ferreira Martins, H., Carvalho de Oliveira Junior, A., Dias Canedo, E., Dias Kosloski, R.A., Ávila Paldês, R., & Costa Oliveira, E. (2019). Design thinking: Challenges for software requirements elicitation. *Information, 10*(12). https://doi.org/10.3390/info10120371

Younas, M., Wakil, K., Jawawi, D.N., Shah, M.A., & Mustafa, A. (2019). An automated approach for identification of non-functional requirements using Word2Vec model. *International Journal of Advanced Computer Science and Applications, 10*(8), 539-547. http://dx.doi.org/10.14569/IJACSA.2019.0100871.

Gondal, M.A., Qureshi, N.A., Mukhtar, H., & Ahmed, H.F. (2020). An Engineering Approach to Integrate Non-Functional Requirements (NFR) to Achieve High Quality Software Process. *In Proceedings of the 22nd International Conference on Enterprise Information Systems, 2,* 377-384. http://doi.org/10.5220/0009568503770384

Maiti, R.R., Krasnov, A., &Wilborne, D.M. (2018). Agile software engineering & the future of non-functional requirements. *Journal of Software Engineering Practice, 2*(1), 1-8.

Ramos, F.B.A., Costa, A.A.M., Perkusich, M., Almeida, H.O., & Perkusich, A. (2018). A Non-Functional Requirements Recommendation System for Scrum-based Projects. *In SEKE,* 149-148. http://doi.org/10.18293/SEKE2018-107

Younas, M., Jawawi, D.N., Ghani, I., & Shah, M.A. (2020). Extraction of non-functional requirement using semantic similarity distance. *Neural Computing and Applications, 32*(11), 7383-7397. http://doi.org/10.1007/s00521-019-04226-5

Werner, C., Li, Z.S., Ernst, N., & Damian, D. (2020). The Lack of Shared Understanding of Non-Functional Requirements in Continuous Software Engineering: Accidental or Essential? *In IEEE 28th International Requirements Engineering Conference (RE),* 90-101. http://doi.org/10.1109/RE48521.2020.00021

Rahy, S., & Bass, J. (2021). Managing non-functional requirements in agile software development. *IET Software*. https://doi.org/10.1049/sfw2.12037

Sabir, M., Chrysoulas, C., & Banissi, E. (2020). Multi-label classifier to deal with misclassification in non-functional requirements. *In World Conference on Information Systems and Technologies, Springer, Cham,* 486-493.

Jha, N., & Mahmoud, A. (2019). Mining non-functional requirements from app store reviews. *Empirical Software Engineering, 24*(6), 3659-3695. https://doi.org/10.1007/s10664-019-09716-7

Younas, M., Jawawi, D.N.A., Shah, M.A., Mustafa, A., Awais, M., Ishfaq, M.K., & Wakil, K. (2020). Elicitation of nonfunctional requirements in agile development using cloud computing environment. *IEEE 34 Access, 8,* 209153-209162.

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. http://doi.org/10.1109/TSE.2019.2904476

Bhowmik, T., & Do, A.Q. (2019). Refinement and resolution of just-in-time requirements in open source software and a closer look into non-functional requirements. *Journal of Industrial Information Integration, 14*, 24-33. https://doi.org/10.1016/j.jii.2018.03.001

Symeonidis, I., Schroers, J., Mustafa, M.A., &Biczók, G. (2019). Towards systematic specification of non-functional requirements for sharing economy systems. *In 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 423-429. http://doi.org/10.1109/DCOSS.2019.00086

Ahmed M., Khan S.U.R., Alam K.A. (2021). QAExtractor: A Quality Attributes Extraction Framework in Agile-Based Software Development. In: Anwar S., Rauf A. (eds) *Proceedings of the First International Workshop on Intelligent Software Automation. Advances in Intelligent Systems and Computing, 1347.* https://doi.org/10.1007/978-981-16-1045-5_2

Alflen, N.C., Santos, L.C., Prado, E.P., & Grotta, A. (2021). Using Combined Techniques for Requirements Elicitation: A Brazilian Case Study. *In Proceedings of the 23rd International Conference on Enterprise Information Systems, 2,* 241-248. https://doi.org/10.5220/0010432702410248

Bouraga, S., Burnay, C., Jureta, I., & Faulkner, S. (2021). Requirements Elicitation for Applications Running on a Blockchain: Preliminary Results. *In International Conference on Advanced Information Systems Engineering,* 38-46. https://doi.org/10.1007/978-3-030-79108-7_5

Trichaisri, S. (2021). Intention to Adopt Non-Functional Requirements: Determinants and Consequences of Perceived Value and Perceived Risk. *Science & Technology Asia, 26*(1), 127-141. https://ph02.tci-thaijo.org/index.php/SciTechAsia/article/view/240464

Roy, M., Deb, N., Cortesi, A., Chaki, R., & Chaki, N. (2021). Requirement-oriented risk management for incremental software development. *Innovations in Systems and Software Engineering, 17,* 1-18. https://doi.org/10.1007/s11334-021-00406-6.

Przybyłek, A., Miler, J., Poth, A., & Riel, A. (2021). Lean and Agile Software Development. *5th International Conference, LASD 2021, 408.* https://doi.org/10.1007/978-3-030-67084-9.

Yadav, S.P., Zaidi, S., Mishra, A., & Yadav, V. (2021). Survey on Machine Learning in Speech Emotion Recognition and Vision Systems Using a Recurrent Neural Network (RNN). *Archives of Computational Methods in Engineering,* 1-18. https://doi.org/10.1007/s11831-021-09647-x

Ramkumar, R., & Mala, G.A. (2021). Non functional requirement based software architecture scheme with security requirement using hybrid group search optimization and genetic algorithm. *Journal of Ambient Intelligence and Humanized Computing, 12*(5), 4863-4876.

Yadav, S.P. (2021), Emotion recognition model based on facial expressions. *Multimedia Tools and Applications, 80,* 26357–26379. https://doi.org/10.1007/s11042-021-10962-5

Brataas, G., Martini, A., Hanssen, G.K., & Ræder, G. (2021). Agile elicitation of scalability requirements for open systems: A case study. *Journal of Systems and Software, 182*. https://doi.org/10.1016/j.jss.2021.111064.

Kumar, D., & Dhir, S. (2022) Requirement Barriers to Implement the Software Projects in Agile Development. In: Aggarwal A.G., Tandon A., Pham H. (eds). *Optimization Models in Software Reliability Springer Series in Reliability Engineering.* Springer, Cham. https://doi.org/10.1007/978-3-030-78919-0_15

Yadav, S.P., & Yadav, S., (2019). Mathematical Implementation of Fusion of Medical Images in Continuous Wavelet Domain. *Journal of Advanced Research in dynamical and control system, 10*(10), 45-54.

Siakas, E., Rahanu, H., Georgiadou, E., & Siakas, K. (2021). Towards Reducing Communication Gaps in Multicultural and Global Requirements Elicitation. In: Yilmaz M., Clarke P., Messnarz R., Reiner M. (eds). Systems, Software and Services Process Improvement. EuroSPI 2021. *Communications in Computer and Information Science, 1442*. https://doi.org/10.1007/978-3-030-85521-5_17