I. J. Mathematical Sciences and Computing, **2026, 2, 51-65**
Published Online on June 8, 2026 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijmsc.2026.02.04

Comparative Study of Functional vs. Non-Functional Requirement Defects in Practice

S. M. Ahsan Habib
American International University-Bangladesh/Computer Science, Dhaka, 1229, Bangladesh
E-mail: smahsanhabib2019@gmail.com
ORCID iD: https://orcid.org/0009-0001-4076-1686

Md. Shariful Islam
American International University-Bangladesh/Computer Science, Dhaka, 1229, Bangladesh
E-mail: sharifulislam91168@gmail.com
ORCID iD: https://orcid.org/0009-0009-4692-6574

Md. Ashraful Islam
American International University-Bangladesh/Computer Science, Dhaka, 1229, Bangladesh
E-mail: 22-46479-1@student.aiub.edu
ORCID iD: https://orcid.org/0009-0002-2678-3026

Jannatul Hoque Samy
American International University-Bangladesh/Computer Science, Dhaka, 1229, Bangladesh
E-mail: hoquesamy25@gmail.com
ORCID iD: https://orcid.org/0009-0008-2410-4070

Jubayer Ahamed*
American International University-Bangladesh/Computer Science, Dhaka, 1229, Bangladesh
E-mail: jubayer@aiub.edu
ORCID ID: http://orcid.org/0000-0003-2076-9194
*Corresponding author

Received: January 30, 2026; Revised: February 27, 2026; Accepted: March 17, 2026; Published: June 8, 2026

Abstract: The success of a software system mostly depends on how effectively and carefully the requirements are understood, prioritized, and handled. Defects, cost escalation and project failure often occur because of misconception of requirements. While current requirement engineering methods have tried to solve these problems, still there are many that struggle to classify perfectly and prioritize needs. This is especially applied for non-functional requirements, where systematic validation remains as a notable gap in the process. This study suggests a structured FR/NFR Defect Decision Ontology Framework that combines ontology-based thinking including machine learning techniques. The framework is performed on a dataset consisting of 6,086 requirements (3,964 FRs and 2,122 NFRs). Defects are automatically identified by using ontology-driven rules, which leads to 458 defective occurrences. Evaluation has been done with a layered 80/20 train–test split with 10-fold cross-validation. FR/NFR classification, defect detection, defect type classification, and severity classification are four classification tasks which are performed by using models such as Naive Bayes, Support Vector Machine (SVM), Logistic Regression, and Random Forest. The result shows strong performance with the highest accuracy, 87.68% for FR/NFR classification, 97.29% for defect detection, 88.76% for defect type classification, and 82.02% for severity classification. The findings indicate that NFR defects are more complex and less traceable than FR defects. The framework will help to improve both accuracy and understandability that supports more effective requirement analysis and decision-making in software engineering.

Index Terms: Functional Requirement, Non-Functional Requirement, Classification, Comparative Analysis, Decision Rules, Prioritization, Defect

This work is open access and licensed under the Creative Commons CC BY 4.0 License.

1. Introduction

The quality of a software system in software engineering is highly affected by the accuracy and completeness of what was collected by the requirements as an initial phase in software development. For design, implementation, and testing requirements work as the model. Therefore, the defects introduced at this phase can spread to the entire software lifecycle which can lead to significant cost implications and project failure [1]. Despite ongoing development requirement engineering still experience challenges to manage complexity, ensure quality, and adapt to developing system needs [2].

The requirements are generally divided into Functional Requirements (FRs) and Non-Functional Requirements (NFRs) whereas FRs describe the behavior and features of system and NFRs define the quality attributes, including the performance, security, usability, maintainability, and reliability [3, 4]. Although both categories are important for a successful software product, but two categories differ in their inherent nature, documentation practices, and validation methodologies considerably.

This difference results in different defects patterns. In non-functional requirements, faults may affect system performance, reliability, or user satisfaction, Functional requirement errors may occur commonly which cause wrong or missing features that have direct impact on system behavior. Non-functional defects have a big impact on how effectively a software product is observed, even though they are less obvious and usually harder to find and address. In case these issues occur during testing or after deployment, it becomes significantly more costly to address and solve [1, 4]. So, enhancing software quality assurance procedures needs an understanding of the distribution and nature of requirement errors, both functional and non-functional. Studies have shown that despite having critical role in overall system quality non-functional requirements are frequently overlooked [5]. However, the critical influence of these concepts dictates the need to offer a single framework on comparative defect analysis.

Existing research focuses on functional and non-functional requirements separately but does not provide a combined mechanism to compare and analyze defects [5]. Some studies focus on classifying FRs or NFRs [1, 4], while others look at identifying specific types of defects, and they do not offer a clear and integrated way to compare defect characteristics across both functional and non-functional domains. Also, many current methods generally face a lack of transparency and strong empirical support. Manual classification of requirements is time consuming, expansive, and can lead to human errors, incomplete identification, and contradicting features [3]. Despite being more accurate, deep learning models sometimes face challenges due to their small size of datasets and 'black box' problem and that makes it difficult for them to clarify their decisions [2, 3]. To overcome these limitations, this paper suggests a layered model to compare and study defects in both functional and non-functional requirements in a practical manner.

The proposed research will compare and analyze the functional and non-functional requirement defects in practice and focus on their differences in patterns of their occurrence, detectability, and the impact on the quality of the software. The analysis is done on the report of defects and real project data to realize how these types of defects affect the overall performance of the system and reliability. In this paper, the researcher suggests a systematic and evidence-based framework, which will allow making a systematic comparison between FR and NFR defects and reduce current limitations.

The proposed framework not only detects and analyzes defects but also will help to go through better strategies of early-stage validation and defect prevention. The results of this study are expected to help project managers, testers, and software practitioners to improve requirement engineering process and make better decisions in software development.

In Section 2, we discuss the previous studies on functional and non-functional requirements that are related to our work. Section 3 is the description of the proposed solution along with a well-structured framework diagram. Section 4 contains the result and discussion part and in Section 5, we discuss the conclusion part and future scopes.

2. Related Works

The difference between Functional Requirements (FR) and Non-Functional Requirements (NFR) plays a crucial role in quality of the software, reliability, and maintainability of the system. FRs specify functionality of the system, whereas NFRs define quality features of the system like performance, security, usability, and scalability. Requirement defects have significant impact on software cost, software reliability, and project success, especially at initial stages of development. The recent research highlights that modern systems, especially AI-driven applications, bring additional challenges in non-functional requirements, such as explainability, transparency, and equity [2, 3].

Although a lot of research has been done in the requirements engineering domain, most research continues to examine both the functional and non-functional requirements separately, which makes it difficult to clearly compare their defect characteristics. This section systematically analyzes existing work across four major dimensions, and these are requirement identification challenges, machine learning-based classification approaches, elicitation and prioritization practices, and domain-specific requirement analysis. Overall, this literature shows the importance of comparing the functional and non-functional requirement defects to understand their influence on software quality, maintainability, and project success.

2.1. FR and NFR Identification and Defect Challenges using Automated Class

Several studies identify fundamental challenges in the identification of FRs and NFRs as well as their management. inconsistencieseals the lack of structured methods of reasoning in the mapping of architectural decisions which results in inconsistencies in the design of the system [6]. In Agile environments, a lack of standardization is also a factor that weakens the integration between FRs and NFRs [7].

Empirical studies reveal that NFRs such as usability, reliability and maintainability are usually under-specified with respect to performance-related requirements [8]. In addition, NFR categories like fairness and explainability are introduced by emerging systems, lacks formal validation approaches [2, 9].

Ambiguity in the understanding of NFR concepts is also a contributor to inconsistencies in the requirement specification and validation [10]. The lack of quantifiable validation criteria for modern NFR attributes is a critical problem [11, 12]. Furthermore, although it is critical to integrate the concepts of FR and NFR to reach system level objectives, current approaches do not have a unified evaluation mechanism [13].

To overcome such problems, machine learning techniques have already been used extensively in requirement classification automation. Deep learning models show better performance in detecting NFRs [14], and studies with comparative results revealed that classification accuracy depends much on feature engineering and quality of a dataset [15, 16]. Neural network-based approaches give promising results and often suffer from overfitting [17].

Pattern based and structured requirements modeling approaches contribute to classification and standardization but have no integration with defect analysis methods [18, 19]. Recent advancements also further demonstrate success of deep learning and hybrid methods for FR/NFR classification [3, 4], and automated completeness checking and defect prediction techniques [1]. But most of the approaches are primarily focused on classification accuracy and do not focus on defect reasoning and explainability. Overall, these methods are specific in terms of dataset and are not applicable across domains.

Synthesis:

Existing research demonstrates significant progress in requirement identification and classification; however, key limitations remain:

- Lack of structured defect identification mechanisms
- Absence of explainable reasoning for classification decisions
- High dependency on dataset characteristics
- No support for comparative defect analysis between FRs and NFRs

2.2. Requirement Elicitation, Prioritization, and Domain-Specific Challenges

Elicitation and prioritization of requirements play an important role in influencing software quality and occurrence of defects. Goal-oriented models provide structured modeling methodology that can't be tasted empirically in different domains [20]. The systematic literature reviews highlight issues of knowledge acquisition regarding the domain and the lack of automated tool support [21, 22].

The agile-based studies focus on the coordination of stakeholders, but the methods of prioritization mostly depend on subjective decision-making processes [23, 24, 25]. Multi-criteria decision-making strategies enhance the accuracy of priorities but have limitations due to the absence of progressive adaptation and interfacing with automated systems [26].

Recent studies emphasize that requirement defects occur frequently due to inconsistency and the lack of involvement of stakeholders especially at the initial stages of development [5]. Also, requirement management practices identify issues while maintaining consistency and traceability of the evolving system requirements [27].

Empirical studies provide inconsistency in the comprehension of the NFR categories like usability, reliability, and maintainability in certain domains [28, 29]. The model-driven methods try to align with NFR reasoning in the system architecture but have not sufficient empirical validation and tool support [30].

Architectural and system-level research focus on the significance of integrating NFRs in the design phases but does not provide a structured framework to detect comparison defects [31]. On the same way, security-centered studies identify important quality features like safety and reliability and have lack of mechanisms of datasets validation and comparative defect analysis mechanisms [32]. Decision-making approaches for software architecture design also highlight the role of FR and NFR evaluation; however, they do not support ontology-based comparative defect reasoning [33].

Recent research also emphasizes the need for explainable and transparent requirement analysis frameworks, especially through AI-driven systems [2, 5].

Synthesis:

Despite advancements in elicitation, prioritization, and domain-specific analysis, several limitations persist:

- Heavy reliance on subjective decision-making

- Lack of automation and tool-supported validation
- Limited empirical validation across real-world datasets
- Absence of structured frameworks for comparative FR/NFR defect analysis
- Poor integration between requirement processes and defect detection mechanisms

2.3. Problem Classification

The classification of research problems is vital for improving the field of software requirements engineering. This literature review identifies and categorizes significant gaps in managing Non-Functional Requirements (NFRs), from their elicitation and prioritization in Agile environments. This review shows a common problem across all these studies: the proposed methods and tools are rarely tested in real companies. Some ideas that might look good on paper, but without the validation from actual industry projects, may not work properly in the complex software development environment.

Although the above works are dedicated to the FR and NFR management separately, there are not many which provide more logical and structured comparative information on the differences in the practice of their defects. We address this gap by our work suggesting a framework through which FR and NFR defects will be explicitly compared based on ontological, empirical, and tool-driven analysis.

Table 1. Identified Gaps and Research Opportunities.

Description	Identified Gaps
Systematic studies on NFR architectural mapping [6, 13].	Lack of formal reasoning, structured methodology, and semantic clarity in mapping NFRs to architecture
NFR studies in Agile development lifecycle [7, 27].	Lack of integrated frameworks and limited empirical validation
Machine learning-based requirement classification and NFR analysis in ML systems [3, 4, 9, 14, 15, 16, 17].	Lack of explainability mechanisms, dataset dependency, and limited tool support
Single-method prioritization approaches [23, 24, 26].	Limited methodological diversity and lack of dynamic prioritization
Conceptual NFR frameworks without implementation [11, 12].	Lack of measurable evaluation metrics and absence of real-world validation
Requirement elicitation studies [21, 22].	Lack of automated tools and structured elicitation processes
Stakeholder-driven prioritization approaches [24, 25].	Limited stakeholder participation and reliance on subjective prioritization
Security and quality requirement studies [32].	Limited empirical validation and lack of experimental evaluation

In summary, managing non-functional requirements mostly depends on informal methods, which ends up with significant lack of industrial validation. Automated tools that are more practical can be useful in real development processes.

3. Methodology

The methodology of this research is well-structured, multilayered, and integrates conceptual reasoning, empirical validation, and tool-based automation to identify the limitations of the existing techniques of requirement analysis. This section explains how the proposed FR/NFR Defect Decision Ontology Framework is developed and applied in practice. The methodology begins with Section 3.1: Framework Description and Research Foundation that contains and describes the design of the proposed framework, its layers, research questions, and identified gaps of previous studies. Each layer of the framework contributes unique functions like conceptual reasoning, empirical validation, and automated decision support which work sequentially to ensure transparency, accuracy and explainability in requirement defect analysis. Section 3.2 Gap-to-Solution mapping of the Proposed Framework presents how each limitation has been addressed within the proposed FR/NFR Defect Decision Ontology Framework. This system provides transparency and traceability to create a reusable and practical method to determine priority requirements that also gives a clear link between the issue of research and the solution of methodology.

3.1. Framework Description and Research Foundations

In this part, our proposed framework has been provided. The framework has two main phases, which are the Initial layer, which is identified as the Define problem, and the FR/NFR Defect Decision Ontology Framework. In defined problem phase, the framework will collect data from the user. After that collected data will be evaluated and validated through the framework. FR/NFR Defect Decision Ontology Framework is divided into 3 layers: Conceptual Layer, Empirical Layer, and Tool-Based Layer. Every layer contains several sub-layers that work consecutively to assure the exact classification of the requirement, real-life validation, and automated defect detection and analysis. When the first phase is completed, the process transfer to the next phrase, which allows the refinement constant enhancement in the flow of work. A detailed explanation of each phase and its sub-phases is provided in Table 2.

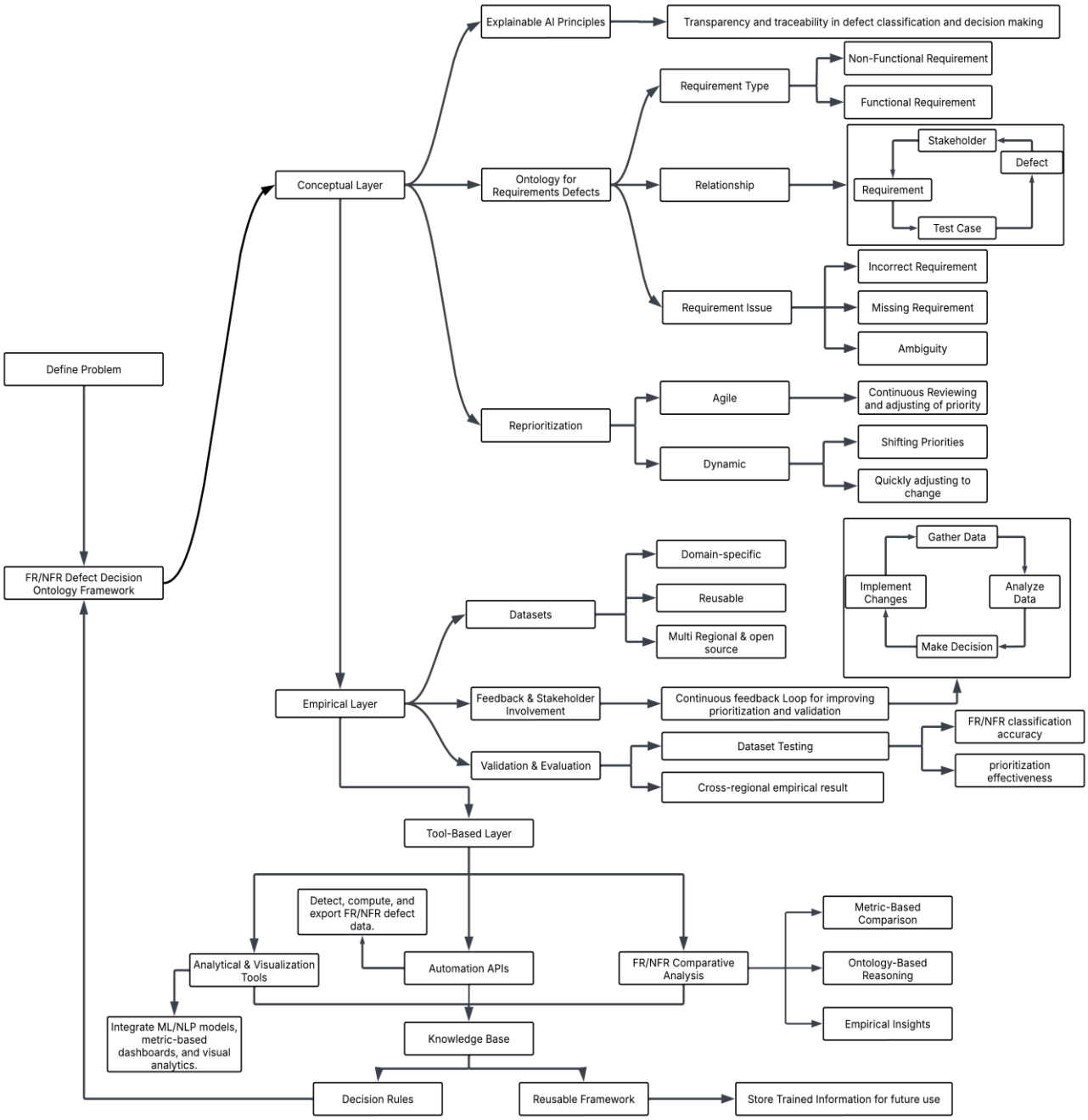


Fig. 1. Block diagram of our proposed framework.

In the study, our proposed framework will work for handling requirement analysis in software engineering. The framework will work across three interconnected layers: the Conceptual Layer, the Empirical Layer, and the Tool-Based Layer. Each layer will play a specific role, and the results produced at one stage will guide the next, ensuring progression throughout the decision-making process.

The “Conceptual Layer” will work as the base layer of the framework. Along with all possible defects of functional and non-functional requirements, this layer will focus on building a clear understanding. Explainable AI principles are added to provide transparency and traceability on making decisions of classification. The relationships that are defined in the ontology are Requirement, Stakeholder, Test case, and Defect. Requirements are listed in Functional Requirements and Non-Functional Requirements based on their characteristics. The framework also identifies common requirement issues like wrong requirements, missing requirements, and ambiguity. Agile and dynamic decision processes support reprioritization mechanism which allows continuous adjustment of requirement priorities according to the evolving needs of stakeholders.

Table 2. Description of proposed multi-layer framework for FR/NFR defect analysis.

Layer 1: Conceptual Layer	Layer 2: Empirical Layer	Layer 3: Tool-Based Layer
Sub-Layer: a) **Explainable AI Principles**: This layer will ensure that all decisions related to classification or prioritization can be explained clearly. The reasoning behind FR/NFR identification or defect detection will be transparent, addressing explainability and traceability.	a) **Datasets**: This layer will collect data from different projects and domains so which will help the framework to become more flexible and useful for wider situations.	a) **Automation APIs**: This layer will use automated detection for checking and identifying FR/NFR defects and calculating defect severity scores. It will also generate analytical reports automatically that will reduce manual analysis and enhance scalability.
b) **Ontology for Requirement Defects**: This layer will build a systematic ontology that describes both functional and non-functional requirements along with their defect patterns. The ontology will connect stakeholders, requirement categories, test cases, and common issues such as incompleteness or ambiguity. By structuring these relationships, the framework will enable conceptual reasoning and a clear understanding of how requirement defects appear and circulate. The organization will make sure that requirement validation stays consistent throughout the process.	b) **Feedback & Stakeholder Involvement**: This level highlights the importance of continuous interaction with stakeholders through planned meetings and output checks. This approach will simplify the analysis results and make sure that they meet stakeholders' expectations. Including this kind of interaction in the framework will promote transparency, flexibility, and consistency in strategy.	b) **Analytical & Visualization Tools**: This level will include analytical and visual elements to display the results from the analysis of FR and NFR defects. Visual dashboards will show metrics such as defect intensity, occurrence, and defect traceability. These tools will give stakeholders data-backed insights to uncover complex and hidden relationships and patterns among different types of requirements and their related defects. This level will offer visual analytics to enhance understanding and help stakeholders make informed decisions.
c) **Reprioritization**: This layer adds a system that keeps priorities up to date, so requirements can adjust as the project progresses. When new risks, dependencies, or feedback appear, priorities are reviewed and updated to stay aligned with the current goals. This approach makes the framework more flexible and better able to support agile, iterative workflows, while keeping both functional and non-functional aspects balanced and carefully monitored throughout the project.	c) **Validation & Evaluation**: This layer will be verified and tested through both quantitative and qualitative methods to check how reliable the classification and prioritization are. Real-world datasets will be used to measure the accuracy, precision, and reliability of FR/NFR classification. The reviews and feedback of stakeholders will be collected to acquire practical applicability. Finally, performance and improvements will be evaluated through comparison with existing approaches.	c) **Knowledge Base**: This layer will store all the validated reasoning patterns and classification rules. The knowledge base will serve as a centralized repository for reusable analytical models and will ensure that all lessons learned and defect identification patterns will be usable for future projects. The structure contains empirical data, and analytical results will help this layer promote organizational learning and help minimize unnecessary analytical work in the same area.
		FR/NFR Comparative Analysis: This sub-layer will perform metric-based, ontological, and empirical comparisons between FR and NFR. It relates to the Tool-Based Layer via a dotted line. This connection uses analytical and visualization tools for computation and representation. The outcomes will be represented through the dashboard and stored in the Knowledge Base for continuous development.

Once the conceptual understanding is built, the framework will progress to the "Empirical Layer", where real-world data will be introduced. This layer allows validates conceptual model with the assistance of dataset gathered from multiple sources in both domains, reusable, and multi-regional open dataset. The feedback of stakeholders is combined in form of a continuous feedback loop to increase the requirement prioritization and validation processes. Testing of datasets and cross-regional empirical analysis to examine the validity of classification and the success of prioritization is also performed. The continuous procedure of the empirical work is based on the iterative process that includes data collection, data examination, and decision formation as well as the introduction of the improvements.

Finally, the framework will reach the "Tool-Based Layer", where the focus will be on automation and efficient implementation. This layer allows automatically detecting, computing, and exporting FR and NFR defect information by applying analytical and visualization tools. Both machine learning and natural language processing can be combined to assist automated classification and analysis based on metrics. APIs of automation allow the communication between modules and the processed requirement information are stored in knowledge base and reusable framework components are supported.

Every output of conceptual, empirical, and tool-based layers is passed to the Decision Rules module which is the main rationale mechanism of the framework. Ontology reasoning outcomes, database validation outputs, stakeholder

feedback information, calculated analytical measures, and stored knowledge patterns are combined by the Decision Rules to generate the final framework determinations of FR and NFR defect characteristics.

In addition, a "FR/NFR Comparative Analysis", which can be called a "Cross-Layer Component", will be included. FR/NFR comparative analysis performs as a part of decision analyzing process, where defect attributes of functional and non-functional requirements are undertaken systematically. The decision outcome is executed with the support of classification results, defect recognition patterns, result of empirical validation, and comparison based on metrics. The result of examined decision is sent back to the first stage of the framework to support ongoing improvement of ontology links, requirement prioritization methods, and precision of detection.

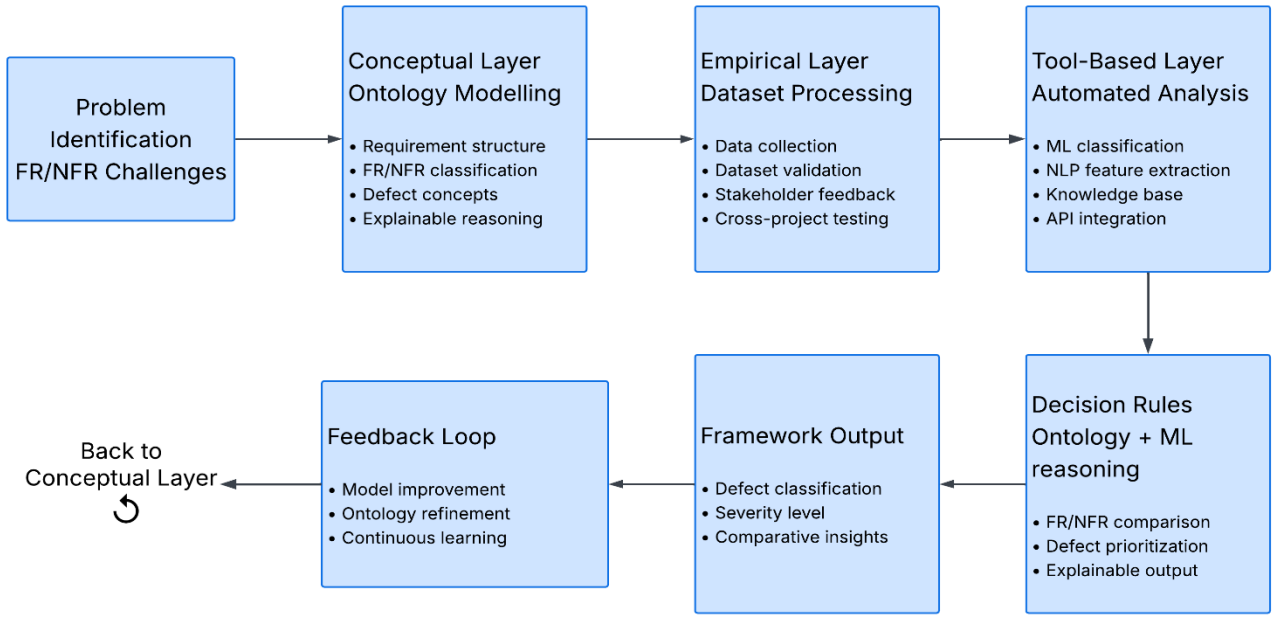


Fig. 2. Workflow diagram of our proposed framework.

Our proposed framework will assist researchers to take the correct decisions in requirement handling. Using this framework, the researchers will find answers to the following questions:

1. How they prioritize and classify requirements.
2. How they detect and handle requirement defects.
3. How they receive inputs of stakeholder.
4. How they use automation tools for decision-making.
5. How they compare functional and non-functional requirements.

Overall, the application of this framework will support clear communication, more accurate requirement prioritization and improved transparency in decision-making. By going through each layer, the development process will become more systemic and dependable. As a result, teams will be better at making decisions regarding requirement handling, leading to improved software quality and performance of the software system.

Ontology Concepts and Relationships:

The formally specified ontology structure describes the defects of requirement to enhance the clarity in technology. The ontology represents the conceptual relationships among the entities of requirement and activates rule-based reasoning to identify defects.

Requirement → General representation of system requirement statements.
Functional Requirement (FR) → Determines what to do within the system.
Non-Functional Requirement (NFR) → describes quality attributes of the system.
Defect → Reflects the problems of requirement quality.
Ambiguity → Vague or unclear requirement description.
Incorrect Requirement → Logically inconsistent or conflicting requirement.
Missing Requirement → Incomplete or insufficiently specified requirement.
Stakeholder → Source of requirement information.
Test Case → Validation artifact linked to requirements.

Ontology Relations:

Requirement → hasType → FR or NFR
Requirement → mayContain → Defect
Defect → hasCategory → Ambiguity, Incorrect, Missing
Requirement → validatedBy → Test Case
Stakeholder → provides → Requirement

Reasoning rules provide the opportunity to identify defects using semantic patterns and linguistic indicators. The formalization provides the traceability of decisions in classification, as well as enhances the interpretability of the result of defect detection.

The reasoning logic establishes the process of identifying requirement types, defects, and severity levels with the help of ontology and analytical modules. The decision-making in classifications is transparent and traceable due to the rules of reasoning.

Requirement Type and Requirement Indicator

Functional Requirements and Non-Functional Requirements are differentiated by using concept-based attributes derived from requirement text patterns.

Table 3. Requirement type with indicator.

Requirement Type and decisions	Indicators
Functional Requirement (FR)	System shall, user can, allow users, process data, generate reports, perform operation
Non-functional Requirement (NFR)	Performance, usability, reliability, security, maintainability, response time, availability

Defect Identification Logic

Defects are detected using ontology relationships and linguistic pattern analysis.

Table 4. Defect type with identification logic.

Defect Type	Identification Logic
Ambiguity	presence of vague or subjective terms (fast, secure, user-friendly, efficient, flexible)
Incorrect	logical inconsistency or conflicting condition
Missing	incomplete specification/description or missing information

Severity Estimation Logic

Severity level is estimated using impact-based reasoning derived from requirement importance and affected quality attributes.

Table 5. Severity level with impact criteria.

Severity Level	Impact Criteria
High	affects system performance, reliability, or security
Medium	affects usability or maintainability
Low	minor clarity or formatting issue

Dataset Description

The dataset is collected from Mendeley Data repository and consists of:

- Total requirements: 6,117
- Valid requirements: 6,086

Data Splitting

- Training Set: 80% (4,868 requirements)
- Test Set: 20% (1,218 requirements)
- Functional Requirements (FR): 3,964 (Training – 3171 and Testing – 793)
- Non-Functional Requirements (NFR): 2,122 (Training – 1697 and Testing – 425)

To further improve reliability and reduce overfitting, K-Fold Cross-Validation (k = 10) was applied during model training.

Machine Learning Models

The following models are used:

- Naïve Bayes
- Support Vector Machine (LinearSVC)
- Logistic Regression
- Random Forest

Evaluation Metrics

The framework is evaluated using:

- Accuracy → $\frac{TP+TN}{TP+TN+FP+FN}$
- Precision → $\frac{TP}{TP+FP}$
- Recall → $\frac{TP}{TP+FN}$
- F1-score → $\frac{2 \times Precision \times Recall}{Precision+Recall}$

The stability and the overall ability to generalize the model are measured through cross-validation. The values of Standard deviation and mean are obtained between validation folds so that performance estimates are reliable. The metrics can be used to compare the effectiveness of various machine learning algorithms objectively.

Tasks Performed

1. FR/NFR Classification
2. Defect Detection
3. Defect Type Classification
4. Severity Prediction

Tool-Based Layer Implementation

To ensure practical applicability, the framework is implemented using:

1. Programming Language: Python
2. Libraries: Scikit-learn, Pandas, NumPy
3. Text Processing: TF-IDF Vectorization
4. API Integration: REST-based processing pipeline

Comparative Analysis Metrics

To compare FR and NFR defects, the following formal metrics are defined:
Defect Density

$$DefectDensity = \frac{Number\ of\ Defects}{Total\ Requirements} \quad (1)$$

Severity Index

$$SeverityIndex = \frac{\sum SeverityWeights}{Total\ Defects} \quad (2)$$

Comparative Difference

$$\Delta = Metric_{FR} - Metric_{NFR} \quad (3)$$

These metrics enable quantitative comparison between FR and NFR defect characteristics across datasets.

3.2. Gap-to-Solution Mapping of the Proposed Framework

In this part, we will discuss how our proposed framework will solve the problems mentioned in Table 1.

The gaps found from previous studies contain limitations in requirement analysis, prioritization, and defect-handling practices. The proposed framework highlights these issues by combining conceptual clarity, empirical validation and tool-based automation layer. Gaps related to the lack of structured reasoning, semantic clarity, and explainability are reduced through ontology-based modeling with the help of requirement analysis and prioritization methods, and explainable AI principles, which ensure transparency and traceability of FR/NFR defect classifications. Limitations associated with the weak empirical grounding, lack of sufficient datasets, and poor geographic validation

are mitigated with the assistance of the Empirical Layer, which combines dataset testing, feedback and stakeholder involvement, and cross-regional evaluations and validations to enhance the accuracy of decision.

Table 6. How to solve the identified gaps using our proposed framework.

Identified Gaps	Solve the gaps using our model
1. Lack of formal reasoning, structured methodology, and semantic clarity in mapping NFRs to architecture [6, 13].	The model addresses a multi-layered FR/NFR Defect Decision Framework that provides the missing formal structure and semantic clarity for classifying defects and making architecture-related decisions based on NFRs.
2. Lack of integrated frameworks and limited empirical validation [7, 27].	The framework is designed to be Dynamic, Domain-specific, and Reusable, which contains a Feedback Loop and generates Cross-regional empirical results, ensuring validation, domain adaptability, and agile scalability through continuous review.
3. Lack of explainability mechanisms, dataset dependency, and limited tool support [3, 4, 9, 14, 15, 16, 17].	The framework is built based on Explainable AI Principles, which ensures transparency. It conducts dataset insufficiency through Dataset Testing and Cross-regional empirical result gathering. It solves insufficient tool support by combining automation APIs, analytical and visualization tools, and defect-detection services to classify, prioritize, and export FR/NFR defect data.
4. Limited methodological diversity and lack of dynamic prioritization [23, 24, 26].	The proposed solution utilizes open-source software and is also multi-regional with Continuous Feedback Loop to enhance refinement and validation. The solution integrates several instruments, including APIs and dashboards, further allowing Modification and Rapid Shifting of Priorities.
5. Lack of measurable evaluation metrics and absence of real-world validation [11, 12].	The framework is built with Automation APIs and Analytical & Visualization Tools, which move from theoretical application to practical application. It focuses on actions such as gathering and analyzing data, Implementing Changes, and Making Decisions.
6. Lack of automated tools and structured elicitation processes [21, 22].	The framework uses automation APIs and tools to detect, compute, and export FR/NFR defect data. The ontology considers Incorrect, Missing, and Ambiguous requirements, hence negating the Issues.
7. Limited stakeholder participation and reliance on subjective prioritization [24, 25].	The design incorporates a vital element that revolves around feedback and Stakeholders. The design utilizes metric-based alternative analysis for objective stakeholder consideration and decision-making.
8. Limited empirical validation and lack of experimental evaluation [32].	The Framework introduces a sub-layer that focuses on validation and evaluation grounded on dataset testing and cross-regional empirical evidence to enhance its recommendations with real-world data that is validated.

Problems with the lack of automation, dependence on single-technique, and lack of dynamic reprioritization are overcome with the assistance of Tool-Based Layer, which provides automated APIs, visualization dashboards, and continuous mechanism of reprioritizing. The framework also addresses issues associated with inadequate stakeholder involvement by implying the feedback loops for perfect prioritization and validation decisions. Moreover, theoretical concepts are transformed into practical, working processes with the assistance of automated defect detection, the comparison based on metrics, and empirical understanding.

Finally, the framework also assures reusability and consistent implementation across projects by upholding a knowledge base and decision rule repository. Overall, all the layers provide a comprehensive solution that directly responds to each identified research gap.

4. Result and Discussion

In the analysis, the experimental evaluation of the proposed framework was aimed at determining its efficiency in categorizing requirements and detecting defects on several analytical dimensions. The assessment encompasses four major tasks, which include FR/NFR classification, defect detection, defect type classification, and severity prediction. Several machine learning models such as Naive Bayesian, Support Vector Machine (LinearSVC), Logistic Regression, and random forest have been implemented to make it robust and generalizable. Data was split into train and test set 80:20, cross-validation was performed using a K-fold (k=10) cross-validation approach in the attempt to reduce bias in performance and increase the reliability of the model.

To analyze the performance standard classification metrics like accuracy, precision, recall, F1-score, cross-validation mean and standard deviation were used. These measures give a clear and thorough explanation of both the predictive performance and the stability of a model. The findings also prove that the effectiveness of models varies among tasks because of the variety of classification complexity and character of requirement data. When dealing with a textual classification task, the linear models tend to perform well because they can cope with high-dimensional sparse features whereas ensemble methods exhibit benefits in pattern recognition tasks. The results of every task in detail are provided in the following tables.

Table 7. FR/NFR classification result.

Model	Accuracy	Precision	Recall	F1-score	CV Mean	CV Std
Naïve Bayes	0.8612	0.8599	0.8612	0.8599	0.8841	0.0156
SVM (LinearSVC)	0.8768	0.8763	0.8768	0.8765	0.8798	0.0231
Logistic Regression	0.8695	0.8693	0.8695	0.8694	0.8809	0.0171
Random Forest	0.8588	0.8604	0.8588	0.8543	0.8496	0.0144

Best performance: SVM achieved 87.68% accuracy for requirement classification.

Table 8. Defect classification result.

Model	Accuracy	Precision	Recall	F1-score	CV Mean	CV Std
Naïve Bayes	0.9466	0.9397	0.9466	0.9391	0.9449	0.006
SVM (LinearSVC)	0.9688	0.9671	0.9688	0.9668	0.9634	0.0068
Logistic Regression	0.9622	0.9597	0.9622	0.9603	0.9596	0.0094
Random Forest	0.9729	0.9732	0.9729	0.9702	0.9705	0.0078

Best performance: Random Forest achieved 97.29% accuracy in defect detection.

Table 9. Defect type classification result.

Model	Accuracy	Precision	Recall	F1-score	CV Mean	CV Std
Naïve Bayes	0.8539	0.855	0.8539	0.8357	0.9906	0.0121
SVM (LinearSVC)	0.8876	0.8954	0.8876	0.877	0.9973	0.0053
Logistic Regression	0.8876	0.8954	0.8876	0.877	0.9973	0.0053
Random Forest	0.7753	0.8562	0.7753	0.7873	0.9651	0.015

Best performance: SVM and Logistic Regression achieved 88.76% accuracy.

Table 10. Severity classification result.

Model	Accuracy	Precision	Recall	F1-score	CV Mean	CV Std
Naïve Bayes	0.7528	0.7472	0.7528	0.7476	0.9556	0.0113
SVM (LinearSVC)	0.8202	0.8206	0.8202	0.813	0.9796	0.0148
Logistic Regression	0.8202	0.8206	0.8202	0.813	0.9795	0.0169
Random Forest	0.6854	0.7546	0.6854	0.6871	0.9507	0.0243

Best performance: SVM and Logistic Regression achieved 82.02% accuracy for severity prediction.

The experimental results confirm that the proposed framework supports comparative analysis between FR and NFR defects by integrating ontology reasoning with empirical learning models. Compared to existing approaches that focus only on requirement classification, the proposed framework provides a multi-stage analytical pipeline consisting of:

- FR/NFR classification
- Defect detection
- Defect type identification
- Severity estimation
- Comparative defect analysis

This integrated capability improves explainability and supports better requirement validation decisions during early software development phases.

5. Conclusion and Future Work

Requirements and Non-Functional Requirements in the form of an ontology. The framework combines semantic modelling, empirical validation, and machine learning methods to help explain the recognition and categorization of requirement defects. The results of experiment confirmed that the proposed strategy has consistent performance in the four tasks during the analysis. SVM performance was the best in requirement classification and severity prediction, but the prediction of defects had the best accuracy in the Random Forest. The outcomes show that the organization based on ontology enhances semantic consistency and makes it interpretable in classification. The framework is a progression of the research as it provides a single methodology to examine the published research, even though it provides a wider form of categoric approach in the decomposition of requirements, the nature of defects and the intensity of defects. The potentiality of the comparative analysis enables seeing the trends of defects in FR and NFR groups with better accuracy, which allows for making a better decision regarding the validity of requirements and their prioritization. Generally, the proposed framework contributes to the systematically replicated system of quality enhancement of requirements and the minimization of spreading defects in the process of creating software.

Future work can extend the structure of ontology by engaging additional defect characteristics like inconsistency and redundancy. Combination of advanced deep learning models such as language models based on transformers may enhance semantic understanding of complex requirements. On big-scale industrial data, the evaluation can enhance the generalizability of the framework. Also, it is possible to develop automated visualization dashboards and real time end

requirement validation tools that could develop practical applicability of the proposed framework within agile software development setting.

All the Declarations and Statements

Author Contributions Statement

S. M. Ahsan Habib — Responsible for conceptualization, methodology, implementation, result analysis and drafting the manuscript, conducted the implementation of experiment, prepared the result, contributed to the literature review, and formatted and drafted the whole manuscript.
Md. Shariful Islam — Supported to write literature review and contributed and assisted to the conclusion section of drafting section.
Md. Ashraful Islam – Writing – Introduction and Literature Review Contribution: Prepared the introduction section and supported literature review through identification and analysis of relevant studies.
Jannatul Hoque Samy – Literature Review Contribution and Writing – Conclusion and Future Work: Contributed to literature review development and prepared the conclusion and future research directions.
Jubayer Ahamed – Supervised the research work, reviewed the manuscript thoroughly, and gave constructive guidance to improve the findings easier to follow and increase the quality of research.

All authors have read and agreed to the published version of the manuscript.

Conflict of Interest Statement

The authors declare that there is no conflict of interest regarding the publication of this paper.

Funding Declaration

No outside funding was received for conducting this research.

Data Availability Statement

The dataset used in this study is taken from publicly available sources and stored in a public GitHub repository, so anyone can check and reproduce the work if needed. They can be accessed at:
=> https://github.com/ahsan-habib7/FR-NFR_Defect_Classification

Ethical Declarations

The research does not include any human or animal subjects, nor does it use personal data requiring ethical approval. Therefore, ethical approval was not required for this research.

Acknowledgments

The authors thank their supervisor for consistent guidance, meaningful feedback and encouragement during this research. They also acknowledge the publicly available datasets that made the empirical validation of the proposed framework possible.

Declaration of Generative AI in Scholarly Writing

The authors made use of AI-assisted tools during the writing process to improve clarity, structure, and language. The authors ensured that all content was critically reviewed and approved for accuracy and originality.

Abbreviations

The following abbreviations are used in this manuscript:

AI – Artificial Intelligence
ML – Machine Learning
FR – Functional Requirement
NFR – Non-Functional Requirement
SVM – Support Vector Machine
API – Application Programming Interface

References

[1] Ruslan Bahrii, Tetiana Skrypnyk, Bohdan Romanov, Elena Zaitseva, Houda El Bouhissi, and Volodymyr Lytvynenko. An approach to identifying functional and non-functional requirements in IT-project management using deep learning models. 2025.

[2] Umm-e-Habiba, Markus Haug, Justus Bogner, and Stefan Wagner. How mature is requirements engineering for AI-based systems? A systematic mapping study on practices, challenges, and future research directions. Requirements Engineering, 29(4):567–600, 2024.

[3] Kiramat Rahman, Anwar Ghani, Sanjay Misra, and Arif Ur Rahman. A deep learning framework for non-functional requirement classification. Scientific Reports, 14(1):3216, 2024.

[4] Abdur Rahman, Abu Nayem, and Saeed Siddik. Non-functional requirements classification using machine learning algorithms. International Journal of Intelligent Systems and Applications, 15(3):56–69, 2023.

[5] Cyrille Dongmo. A review of non-functional requirements analysis throughout the SDLC. Computers, 13(12):308, 2024.

[6] Muhammad Nouman, Muhammad Azam, Ashraf Mousa Saleh, Abdullah Alsaeedi, and Hayfa Yousef Abuaddous. A systematic review of non-functional requirements mapping into architectural styles. Bulletin of Electrical Engineering and Informatics, 12(2):1226–1236, 2023.

[7] Devendra Kumar, Anil Kumar, and Laxman Singh. Non-functional requirements elicitation in agile base models. Webology, 19(1):1992 2018, 2022.

[8] Misael C Júnior, Domenico Amalfitano, Lina Garcés, Anna Rita Fasolino, Stevão A Andrade, and Márcio Delamaro. Dynamic testing techniques of non-functional requirements in mobile apps: A systematic mapping study. ACM Computing Surveys (CSUR), 54(10s):1–38, 2022.

[9] Khan Mohammad Habibullah, Gregory Gay, and Jennifer Horkoff. Non-functional requirements for machine learning: Understanding current use and challenges among practitioners. Requirements Engineering, 28(2):283–316, 2023.

[10] Anderson Oliveira, João Correia, Wesley KG Assunção, Juliana Alves Pereira, Rafael de Mello, Daniel Coutinho, Caio Barbosa, Paulo Libório, and Alessandro Garcia. Understanding developers' discussions and perceptions on non-functional requirements: The case of the spring ecosystem. Proceedings of the ACM on Software Engineering, 1(FSE):517–538, 2024.

[11] Maximilian A Köhl, Kevin Baum, Markus Langer, Daniel Oster, Timo Speith, and Dimitri Bohlender. Explainability as a non-functional requirement. In 2019 IEEE 27th international requirements engineering conference (RE), pages 363–368. IEEE, 2019.

[12] Larissa Chazette and Kurt Schneider. Explainability as a non-functional requirement: challenges and recommendations. Requirements Engineering, 25(4):493–514, 2020.

[13] Pablo Becker, Guido Tebes, Denis Peppino, and Luis Antonio Olsina Santos. Applying an improving strategy that embeds functional and non-functional requirements concepts. Journal of Computer Science & Technology, 19, 2019.

[14] Md Abdur Rahman, Md Ariful Haque, Md Nurul Ahad Tawhid, and MdSaeedSiddik. Classifying non-functional requirements using rnn variants for quality software development. In Proceedings of the 3rd ACM SIGSOFT International Workshop on Machine Learning Techniques for Software Quality Evaluation, pages 25–30, 2019.

[15] Vajeeha Mir Khatian, Qasim Ali Arain, Mamdouh Alenezi, Muhammad Owais Raza, Fariha Shaikh, and Isma Farah. Comparative analysis for predicting non-functional requirements using supervised machine learning. In 2021 1st International Conference on Artificial Intelligence and Data Analytics (CAIDA), pages 7–12. IEEE, 2021.

[16] László Tóth and László Vidács. Comparative study of the performance of various classifiers in labeling non-functional requirements. Information Technology and Control, 48(3):432–445, 2019.

[17] Cody Baker, Lin Deng, Suranjan Chakraborty, and Josh Dehlinger. Automatic multi-class non-functional software requirements classification using neural networks. In 2019 IEEE 43rd annual computer software and applications conference (COMPSAC), volume 2, pages 610–615. IEEE, 2019.

[18] Methinee Amorndettawin and Twittie Senivongse. Non-functional requirement patterns for agile software development. In Proceedings of the 20193rdInternational Conference on Software and e-Business, pages 66–74, 2019.

[19] LuizMarcioCysneirosandJulioCesarSampaiodoPradoLeite. Non-functional requirements orienting the development of socially responsible software. In International Conference on Business Process Modeling, DevelopmentandSupport, pages335–342.Springer,2020.

[20] Jennifer Horkoff, Fatma Başak Aydemir, Evellin Cardoso, Tong Li, Alejandro Maté, Elda Paja, Mattia Salnitri, Luca Piras, John Mylopoulos, and Paolo Giorgini. Goal-oriented requirements engineering: an extended systematic mapping study. Requirements engineering, 24(2):133–160, 2019.

[21] Lenis RWong,DavidSMauricio,GlenDRodriguez,etal. A systematic literature review about software requirements elicitation. Journal of Engineering Science and Technology, 12(2):296–317, 2017.

[22] Eva-Maria Schön, Jörg Thomaschewski, and María José Escalona. Agile requirements engineering: A systematic literature review. Computer standards & interfaces, 49:79–91, 2017.

[23] M Younas, DNA Jawawi, I Ghani, and R Kazmi. Non-functional requirements elicitation guideline for agile methods. Journal of Telecommunication, Electronic and Computer Engineering (JTEC), 9(3-4):137–142, 2017.

[24] Richard RMaitiandFrankJMitropoulos. Prioritizing non-functional requirements in agile software engineering. In Proceedings of the 2017 ACM Southeast Conference, pages 212–214, 2017.

[25] Aleksander Jarzębowicz and Paweł Weichbroth. A qualitative study on non-functional requirements in agile software development. IEEE Access, 9:40458–40475, 2021.

[26] Abdulhafeez Muhammad, Ansar Siddique, Muhammad Mubasher, Amjad Aldweesh, and Quadri Noorulhasan Naveed. Prioritizing non-functional requirements in agile process using multi criteria decision making analysis. IEEE Access, 11:24631–24654, 2023.

[27] Scarlet Rahy and Julian M Bass. Managing non-functional requirements in agile software development. IET software, 16(1):60–72, 2022.

[28] Jie Zou, Ling Xu, Mengning Yang, Xiaohong Zhang, and Dan Yang. Towardscomprehendingthenon-functional requirements through developers' eyes: An exploration of stack overflow using topic analysis. Information and Software Technology, 84:19–32, 2017.
[29] Arshad Ahmad, Chong Feng, Kan Li, Syed Mohammad Asim, and Tingting Sun. Toward empirically investigating non-functional requirements of ios developers on stack overflow. IEEE Access, 7:61145–61169, 2019.
[30] David Ameller, Xavier Burgués, Dolors Costal, Carles Farré, and Xavier Franch. Non-functional requirements in model-driven development of service-oriented architectures. Science of computer programming, 168:18–37, 2018.
[31] Mohammad Gulam Lorgat and Hugo Paredes. Handling non-functional requirements in software architecture. Architecture, 30:31, 2021.
[32] Rafiq Ahmad Khan, Siffat Ullah Khan, Habib Ullah Khan, and Muhammad Ilyas. Systematic mapping study on security approaches in secure software engineering. Ieee Access, 9:19139–19160, 2021.

Authors' Profiles



S. M. Ahsan Habib was born in Dhaka, Bangladesh. He is currently pursuing a B.Sc. degree in Computer Science and Engineering at American International University–Bangladesh (AIUB), Dhaka, Bangladesh. His major field of study is software engineering. His research interests include software engineering, requirements engineering, and the application of machine learning in software quality analysis.



Md. Shariful Islam was born on October 1, 2001, in Sirajganj, Bangladesh. He is pursuing a Bachelor of Science in Computer Science and Engineering at American International University–Bangladesh (AIUB). He is currently working as a Data Analyst at Noba Global in Dhaka, Bangladesh. Previously, he worked in ICT Operations at Rupali Bank Limited, Motijheel Head Office, Dhaka. His research interest includes Software Engineering.



Md. Ashraful Islam is currently pursuing his B.Sc. in Computer Science at American International University-Bangladesh (AIUB), Dhaka, Bangladesh. He is an undergraduate student in Computer Science from AIUB. Currently, He is preparing for his future academic and professional career in the field of Computer Science. His research interest includes Software Engineering.



Jannatul Hoque Samy is currently pursuing her B.Sc. in Computer Science at American International University-Bangladesh (AIUB), Dhaka, Bangladesh. She is an undergraduate student in Computer Science from AIUB. Currently, she is preparing for her future academic and professional career in the field of Computer Science.



Jubayer Ahamed completed his B.Sc. in Computer Science from American International University-Bangladesh, Dhaka, Bangladesh. He obtained his MSc from American International University-Bangladesh, Dhaka, Bangladesh. Currently, Jubayer Ahamed is working as a Lecturer in the Department of Computer Science (CS) at American International University- Bangladesh (AIUB). He is also associated with Artificial Intelligence Research & Innovation Lab (AIRIL) as a research associate. His research interests include Software Engineering and Machine Learning.

How to cite this paper:

S. M. Ahsan Habib, Md. Shariful Islam, Md. Ashraful Islam, Jannatul Hoque Samy, Jubayer Ahamed, "Comparative Study of Functional vs. Non-Functional Requirement Defects in Practice", International Journal of Mathematical Sciences and Computing(IJMSC), Vol.12, No.2, pp. 51-65, 2026. DOI: 10.5815/ijmsc.2026.02.04