I.J. Information Technology and Computer Science, **2025, 4, 38-48**
Published Online on August 8, 2025 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2025.04.04

Software Quality Attributes in Requirements Engineering

Denys Gobov*
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, 03056, Ukraine
E-mail: d.gobov@kpi.ua
ORCID iD: https://orcid.org/0000-0001-9964-0339
*Corresponding author

Oleksandra Zuieva
CS ltd, Kharkiv, 61072, Ukraine
E-mail: olekszueva@gmail.com
ORCID iD: https://orcid.org/0000-0001-9661-9657

Received: 09 February 2025; Revised: 11 April 2025; Accepted: 23 May 2025; Published: 08 August 2025

Abstract: As software systems continue to grow more complex, evaluating software quality becomes increasingly critical. This study analyzes existing software quality models, including McCall, Boehm, FURPS, and ISO Systems and Software Engineering – Systems and Software Quality Requirements and Evaluation (SQuaRE), with a specific focus on the ISO/IEC 25010:2023 standard. The research aims to assess the completeness of these models and explore interdependencies among key quality attributes relevant to software requirements engineering. The paper identifies key characteristics and associated metrics based on ISO/IEC standards using comparative analysis and a literature review. Findings show that ISO/IEC 25010:2023 provides the most comprehensive structure, with Functional Suitability and Compatibility identified as essential due to their universally recommended metrics. Survey data from 328 practicing analysts in Ukraine and internationally demonstrate a gap between theoretical models and real-world requirements documentation practices, particularly for non-functional requirements. The identified dependencies between quality attributes enable a more integrated and structured approach to identifying and analyzing non-functional requirements in IT projects. The study emphasizes that software quality models must be tailored to project-specific goals and constraints, with attention to trade-offs and stakeholder needs during the requirements specification, prioritization, and validation processes. The findings support the adaptation of quality models to specific project constraints and emphasize the business analyst's role in tailoring quality criteria for practical use in software development.

Index Terms: Software Quality Model, Software Requirements, Quality Attribute.

1. Introduction

Modern society relies heavily on software systems integrated into daily life. As the functionality of most software products has advanced to address critical user needs and applications grow more complex, ensuring high quality becomes crucial. High-quality software ensures user satisfaction and loyalty, providing a significant competitive advantage in a competitive market.

In the article [1], five key characteristics of software quality were defined, emphasizing that these characteristics describe both the subjective aspect – the user's perception of the product under certain conditions – and the objective aspect – conformance with requirements and the absence of defects. During the software development process, user needs are transformed through business analysis techniques into requirements, which are then used to create the software. These requirements are the fixed and measurable needs of an average user, which allows them to be used as a basis for assessing the final product's conformity to the needs of end users.

Thus, the subjective aspect of software quality - how well the implementation meets the user's needs and expectations - is a derivative of two main factors: how well the documented requirements align with the user's real needs (requirements quality) and how well the final implementation fulfills these requirements.

This article focuses on the quality of software assessed under the documented requirements; the quality of the requirements is not considered here. Quality assurance and the application of quality management methods primarily rely on evaluating the quality level. Quality models are used for evaluation, in which separate components of quality

characteristics, sub-characteristics (attributes), and specific criteria and metrics for measurement are distinguished. Quality measurement and evaluation are key aspects of delivering software products to users. As a result, researchers have adopted diverse methods of quantifying software quality. Software quality models serve as frameworks that provide a systematic approach to classifying and evaluating various characteristics, collectively defining the overall quality of a software product.

These models provide a standardized and structured approach for understanding and effectively managing the multifaceted nature of software quality. The first quality models appeared in the late 1970s. The papers [2, 3] review the evolution from basic models, such as those by McCall [4] and Boehm [5], to tailored quality models that are specific to a particular domain of application. It should be noted that [6] conducted an in-depth study based on the then-existing models of 31 authors and 55 different quality attributes and criteria, emphasizing the high importance of the issue and the significant differences of opinion among researchers. It is also worth noting that most tailored models are based on the ISO 9126 standard [7].

2. Related Works

Given the increasing demands on the quality of IT systems and their increasing complexity, research continues into quality models and the possibilities of their practical application within the software development process. In [8], existing software quality models were comprehensively reviewed, with a focus on their applicability to commercial off-the-shelf (COTS) and open-source software components. The findings highlight that while traditional models, such as those by McCall and Boehm, provide a foundation, they often lack standardized approaches for integration into diverse IT systems. This lack of standardization consistently hinders the assessment and comparison of software quality across different development paradigms. In [9], the authors conducted a systematic literature review of existing software quality meta-models. 75% of the analyzed models were proposed for generic purposes and to evaluate custom software products. ISO 9126 was previously considered the leading model, but ISO/IEC 25010 has since superseded it.

The article [10] explores the evolution of perspectives on software quality by comparing viewpoints from academia and industry. It reveals that while researchers traditionally emphasize abstract, model-driven definitions of quality (such as ISO standards or theoretical frameworks), practitioners often focus on tangible outcomes like maintainability, performance, and customer satisfaction. The authors emphasized the need for adaptable and context-aware models that better align with both theoretical and practical concerns. It is worth noting that several studies have been devoted to the influence of the project context and approaches to constructing quality assessment models. In [11], the impact of project context and business analysis techniques on the structure of documents, including non-functional requirements, is examined. The study [12] examined how cultural factors influence the design, application, and interpretation of software quality assessment models. It finds that national, organizational, and team cultures can significantly impact the prioritization and evaluation of quality attributes. The paper advocates for developing culturally adaptive quality models and encourages researchers to incorporate sociotechnical dimensions into software quality assessments. In [13] examines which classes of non-functional requirements are most often used by practicing business analysts and requirements engineers in IT projects. It also examines how the project context (subject area, company type, team size, etc.) influences the choice of non-functional requirements for solution quality assessment. The study [14] emphasizes the significant impact of development team members on producing high-quality software products. It proposes a qualitative model that focuses on team characteristics, such as team experience, cohesion, and communication, which are crucial in determining software quality. The study's findings were obtained through interviews with seven experts and a survey. The statistical reliability of the results was not assessed due to the small number of respondents. Article [15] confirms the significant role of quality requirements on project success and team satisfaction. In [16], a software quality forensics framework was proposed for forensic investigation into software quality. Implementing this framework has reportedly improved the efficiency and effectiveness of investigating and managing incidents related to software quality. The study acknowledges that the proposed framework requires further empirical validation. The study [17] presents a recommendation system that leverages machine learning to suggest appropriate software quality characteristics tailored to the dynamic requirements of software projects. While the proposed model shows promise, its effectiveness is demonstrated only through simulations. Additionally, the model's adaptability to diverse software development environments and varying requirements remains to be thoroughly evaluated.

An analysis of works on studying quality models of IT solutions reveals that defining a quality model remains a relevant approach. A universal quality model is also necessary for creating narrowly focused quality models adapted to specific classes of information systems.

However, the issues of practical application of quality models and the use of quality attributes in requirements within the context of specific projects remain open, although this forms the foundation for providing software quality in general.

3. Software Quality Models and Characteristics

To conduct a structured comparative analysis of Software Quality Models, we selected a representative set of such

models based on four key criteria: historical significance, industry adoption, architectural perspective, and standardization status:

- McCall's [4] and Boehm's [5] models were included due to their foundational role in defining quality frameworks.
- FURPS represents a widely used industry model embedded in the Rational Unified Process (RUP) and other engineering practices.
- Dromey's [18] model and the arc42 [19] Quality Model were selected for their emphasis on the architectural link between quality attributes and system components.
- The BABOK [20] model was included to reflect the current practices of business analysts in using quality characteristics in requirement specifications.
- ISO/IEC 25010:2023 [21] and ISO/IEC 25019:2023 [22] were used as benchmarks due to their status as current international standards, offering the most comprehensive and structured taxonomy of quality characteristics, sub-characteristics, and associated metrics. It is worth noting that the ISO SQuaRE series of standards comprises four models: the Product Quality Model, the Service Quality Model, the Data Quality Model, and the Quality-in-Use Model. The Product quality and Quality-in-use models directly relate to software properties, while the Data quality and Service quality models are complementary to each other.

Table 1 compares the basic quality models based on their main characteristics. It was constructed using key characteristics and sub-characteristics of various models [4, 5, 18, 19, 20, 21, 22]. Considering that [21] provides for using 39 sub-characteristics combined into nine main characteristics, the table uses these characteristics. Italicized characteristics include sub-characteristics listed in other models for comparison purposes. This approach highlights the comprehensive nature of the ISO/IEC 25010:2023 model while illustrating how different models emphasize various aspects of software quality.

Table 1. Comparison of the factors in software quality models

McCall's	Boehm's	FURPS	Dromey's	arc42	BABOK. IIBA	ISO/IEC 25010:2023+25019:2023
Reliability	Reliability	Reliability	Reliability	Reliable	Reliability	Reliability
Correctness		Functionality	Functionability	Suitable	Functionality	Functional suitability
Efficiency	Efficiency	Performance	Efficiency	Efficient	Performance Efficiency	Performance Efficiency
Integrity				Secure	Security	Security
Usability	Human engineering	Usability	Usability	Usable	Usability	Interaction capability
Maintainability	Maintainability	Supportability	Maintainability	Operable	Maintainability	Maintainability
Testability	Testability					Maintainability (Testability)
Flexibility	Modifiability			Flexible		Flexibility
Portability	Portability		Portability		Portability	Flexibility (Adaptability)
Reusability			Reusability			Maintainability (Reusability)
Interoperability					Compatibility	Compatibility
	Understandability					Appropriateness Recognizability (Interaction)
				Safe		Safety
					Availability	Reliability (Availability)
					Scalability	Flexibility (Scalability)
					Localization	Interaction capability (inclusivity)

As shown in Table 1, the quality characteristics and sub-characteristics presented in the ISO SQuaRE series cover all the quality properties highlighted in other models. It should be noted that the characteristics of Certification and Compliance, as highlighted in the BABOK guide, can be aligned with the acceptability characteristic of the Quality in Use model and its sub-characteristics—Trustworthiness and Compliance, respectively, according to [22]. Also, according to [23], assigned properties are not quality characteristics. Service Level Agreements, highlighted in the BABOK guide, are assigned properties; therefore, they're not part of software quality models, as they can be changed without altering the software. Therefore, the ISO standards provide the most comprehensive coverage of software properties; consequently, this article's further analysis will be based on this approach. According to [24], the SQuaRE quality model provides a structured approach to representing these characteristics by dividing them into sub-characteristics, each of which can be measured using specific quality metrics. Based on stakeholder requirements, these measures enable the quantification of ICT product quality.

Furthermore, [25] provides essential guidance on selecting and applying quality measures for each characteristic and its corresponding sub-characteristics. The measures may be classified as internal or external, depending on whether they are used during development or operation. Additionally, they can be categorized as either generic or specific,

meaning they apply broadly or in particular situations.

The standard also defines recommendation levels – Highly Recommended (HR), Recommended (R), and Used at User's Discretion (UD) – to assist in the selection and use of these measures. Each application or business domain encounters a unique set of software quality challenges. Therefore, software quality should be tailored for each specific context to ensure that software meets the specific needs of a particular business or domain. Figure 1 illustrates the distribution of quality characteristics proposed in the [21] standard, categorized by the metrics included in these characteristics according to the ISO/IEC 25023:2016 standard [25].

As shown in Figure 1, every metric for Functional Suitability is rated as Highly Recommended, generally applicable, and covers both internal and external contexts. It suggests that organizations should prioritize thorough assessments of functional completeness, correctness, and appropriateness when establishing quality evaluation processes. Compatibility is similar to Functional Suitability, which includes the most general and highly recommended metrics. Unlike Functional Suitability, Compatibility incorporates 25% specific and 25% external metrics. It indicates that while it primarily adheres to broad interoperability standards, it also requires context-specific assessments in specific settings. Consequently, organizations should evaluate Compatibility with the same rigor as Functional Suitability and conduct operational testing to address its external metric components. Performance Efficiency also mainly uses generally applicable metrics (83%). Notably, 42% of its metrics target external aspects, emphasizing operational relevance. This distribution highlights a sustained focus on performance efficiency from development through maintenance. Therefore, organizations should prioritize metrics that align with their operational constraints and specific performance needs, ensuring optimal resource utilization and user satisfaction regarding system responsiveness. Interaction Capability is distinguished by a high proportion of specific metrics (74%) and User Discretion metrics (48%), highlighting its dependence on contextual nuances. Unlike Functional Suitability and Performance Efficiency, which rely on general criteria, Interaction Capability requires a tailored evaluation approach. Moreover, Interaction Capability metrics maintain relevance throughout the development lifecycle, as they apply equally to internal and external contexts. Organizations should develop customized evaluations for Interaction Capability characteristics that align with their business context and user experience objectives rather than relying solely on standardized quality metrics.

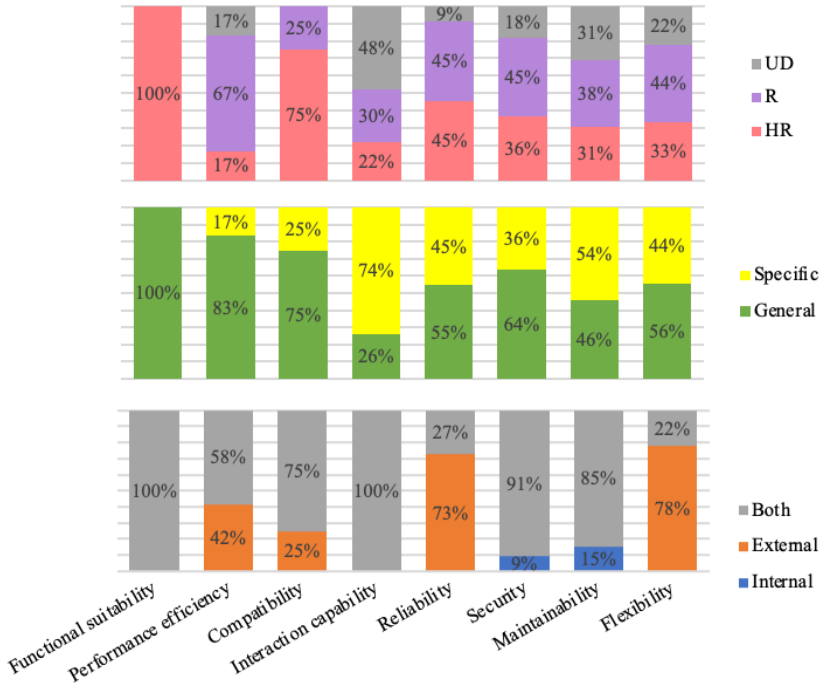

Fig.1. Quality characteristics distribution by metric types

Reliability and Flexibility display similar distributions of metrics while representing contrasting approaches to quality assessment from a business perspective. Each maintains a balanced mix of general and specific metrics, reflecting comparable contextual adaptability, and both emphasize external metrics to highlight operational significance. The similarity in these patterns suggests that despite their contrasting business objectives, both characteristics require balanced assessment frameworks. Consequently, organizations should implement complementary measurement strategies that acknowledge their operational focus. Maintainability and Security are categories that include the recommended internal metrics, making them essential for early-stage evaluation. In contrast, Reliability and Flexibility are typically evaluated later. Security mainly includes general metrics (64%), while Maintainability is more specific (54%).

Thus, the requirements for Functional Suitability and Compatibility should serve as the basis for developing a quality assessment system, as they encompass the most significant number of generally recommended metrics. When creating software quality requirements, businesses should pay special attention to characteristics with a significant percentage of specific metrics, such as Interaction Capability, as well as characteristics evaluated in an external context, including Performance Efficiency, Reliability, and Flexibility. Requirements for these characteristics can be developed in conjunction with user requirements, unlike Maintainability and Security requirements, which necessitate an internal assessment. This implies that the requirements for these characteristics depend on the requirements for external characteristics.

Therefore, ISO standards represent the most comprehensive approach to covering software properties. Based on the analysis of quality metrics distribution according to ISO/IEC 25023:2016, it can be concluded that not all quality attributes have equal importance and applicability in the process of software requirements formation within the context of a specific project. The structure of metrics and their classification provide clear guidelines for prioritizing quality characteristics.

Functional Suitability and Compatibility should be included in requirements as foundational characteristics, since all their metrics are highly recommended and universally applicable. These attributes form the foundation of the quality assessment system. Performance Efficiency, Reliability, and Flexibility, which are characterized by a significant proportion of external metrics, should be developed at the initial stages in parallel with user requirements and tested in the operational environment. Requirements for Interaction Capability should be formulated in consideration of the specific business context and user experience objectives. Maintainability and Security, which include predominantly internal metrics, should be specified at an early development stage as requirements for architecture and internal system structure.

Thus, it is recommended to distribute quality requirements documentation in phases: first, basic and internal attributes (Functional Suitability, Compatibility, Maintainability, Security), then operational and context-dependent ones (Performance Efficiency, Reliability, Flexibility, Interaction Capability).

However, the universality and completeness of the standard do not enable the determination of the project context's influence on the selection of the most critical quality attributes, which may be necessary due to the project's time and budget constraints. The domain can also influence the choice of a set of quality attributes. Projects in safety-critical domains (e.g., healthcare, aerospace) or highly regulated industries (e.g., finance) often require domain-specific attributes or advanced interpretations (e.g., security, compliance) that are not well represented in the base model.

To address this problem, updated frameworks adapted for Agile and DevOps environments are being developed. For example, ISO/IEC 29110 [26] provides guidance for very small entities with simplified quality processes tailored for agile-compatible workflows. DORA (DevOps Research and Assessment) metrics [27] are used to evaluate the performance of software development teams and measure the effectiveness of their DevOps practices, which in turn impact software quality. Furthermore, emerging approaches using machine learning and AI-driven tools [28] for quality prediction and anomaly detection are being explored, particularly in the context of continuous integration and continuous deployment. While these models are not as formalized as ISO standards, they reflect the growing need for context-aware and automation-friendly quality evaluation. At the same time, it is worth noting that despite the ISO/IEC 25010:2023 providing a comprehensive and structured taxonomy of software quality attributes, particular difficulties and limitations arise regarding its practical application in software development.

4. Software Quality Attributes Interdependencies

Software quality attributes are interdependent and often have complex relationships. Optimizing one attribute may impact others, making it difficult to achieve an ideal balance across all quality factors simultaneously. So, compromises are an integral aspect of the software development process. Understanding these interrelationships is crucial for making informed design decisions and prioritizing quality attributes in accordance with project requirements. Several studies have been devoted to studying the relationships between quality criteria [29, 30], as well as in the annexes to the ISO/IEC 25030:2019 standard. An example of these relationships is given [31]. Table 2 provides a general picture of the identified relationships between quality criteria. If the quality attribute from the first column is enhanced, then quality attributes from the "Influence+" column will also be enhanced, and quality attributes from the "Influence-" column will be degraded.

Table 2 illustrates the interdependencies between the eight Product quality attributes of software, as outlined in the ISO/IEC 25010:2023 standard [20]. Given that the characteristic "Safety" was added only in the latest revision and the current ISO/IEC 25030:2019 standard does not include it [31], this characteristic is not listed in the table. However, its relationship with other characteristics can be determined logically and experimentally while developing a quality assessment system for a specific project.

Table 2. Software quality attributes interdependencies

Quality attribute	Influences +	Influences -
Functional suitability	Interaction capability, Reliability, Maintainability, Flexibility	Performance efficiency, Security, Reliability, Compatibility, Maintainability, Flexibility
Performance efficiency	Interaction capability	Compatibility, Interaction capability, Security, Maintainability, Flexibility, Reliability
Compatibility	Flexibility, Maintainability (partially), Functional suitability	Security, Maintainability (partially), Performance efficiency
Interaction capability (Usability)	Functional suitability, Reliability, Performance efficiency (partially), Maintainability (partially)	Flexibility, Performance efficiency (partially), Maintainability (partially), Security
Reliability	Functional suitability, Interaction capability, Maintainability (partially), Security	Maintainability (partially), Flexibility
Security	Reliability	Performance efficiency, Compatibility, Interaction capability, Flexibility, Functional suitability
Maintainability	Functional suitability, Compatibility, Flexibility, Reliability (partially), Security (partially)	Performance efficiency, Reliability (partially), Security (partially)

As shown in Table 2, improving Functional suitability positively influences interaction capability, which is mutually positive. However, the impact on Reliability, Maintainability, and Flexibility is ambiguous and may worsen Performance Efficiency, Security, and Compatibility. Functional suitability primarily affects the quality-in-use for primary users, for whom implementing functions that meet their needs may include increasing software usability. At the same time, expanding functionality to more fully cover requirements can reduce the overall performance of the software, complicate its maintenance, and negatively impact the ability to interact with other products. Improving Performance efficiency also undoubtedly enhances the user experience. However, optimizing for speed and resource usage may require compromising particular aspects of security, flexibility, or reliability. Enhancing Compatibility and Reliability can positively influence the system's functional suitability to user needs and the software's ability to operate in different environments. The impact on Maintainability is ambiguous. To ensure compatibility with other systems, the product may become more complex and require constant support and updates to maintain these interactions.

On the other hand, integrations can expand the product's modification capabilities. Interaction with other products requires a certain level of openness, so increasing Compatibility may negatively affect Security. Increasing Reliability improves user experience and functional suitability to needs, even if such needs are not explicitly stated. Reliability also increases the level of Security, unlike other categories where improvements may require lowering the level of security. However, ensuring stable operation can complicate software maintenance and adaptability to different environments.

Ensuring software security is often a stumbling block in product development and can hinder the improvement of other qualities. Therefore, increasing the level of Security negatively affects most other quality characteristics, and conversely, improving other characteristics may increase the likelihood of vulnerabilities. Maintainability shows favorable relationships with many attributes, suggesting that high-quality support contributes to the possibilities of functional expansion and integrations with other systems. However, modifications can sometimes lead to a decrease in the product's security level or performance.

Increasing Flexibility, as the software's ability to adapt to changes in the environment and usage contexts, undoubtedly positively influences the ability to interact with other systems, increases the number of integrations, etc. However, it may harm performance and security.

Therefore, the conducted analysis demonstrates that ensuring software quality requires a systematic approach to prioritizing quality characteristics and considering their interdependencies. Table 2 illustrates the potential impacts of specific quality characteristics on others, which should be considered when developing metrics, a quality assessment system, and software requirements, taking into account the specific context and business features of a particular project. Requirements elicitation processes should incorporate trade-off analysis techniques to identify potential conflicts between quality characteristics specific to a project. Requirements specifications should clearly delineate the desired quality attributes, their relative importance, and the acceptable compromise thresholds. Furthermore, validation metrics should be designed to evaluate individual quality attributes and their systemic effects on other characteristics, acknowledging that optimization in one dimension may precipitate degradation in others. The requirements elicitation processes should include trade-off analysis techniques and a dependency matrix to predict potential conflicts between quality attributes. Requirements specifications should clearly outline the desired quality attributes, their relative importance, and acceptable compromise thresholds. The business analyst plays a central role in implementing these principles, being responsible for adapting theoretical quality models to the specific project conditions. This includes identifying relevant quality attributes, forming a balanced set of criteria considering interdependencies, correctly formulating requirements with specific metrics and threshold values, ensuring traceability between quality requirements and functional requirements, and documenting acceptable compromises between conflicting attributes.

5. Practices of Working with Quality Criteria at the Stage of Defining Requirements in IT Projects

To collect information on the practices of working with quality criteria when identifying and analyzing requirements for IT solutions, as well as the influence of project context, a survey was conducted. The survey involved 328 practicing business analysts and requirements engineers from Ukrainian and international companies. The structure

of the questionnaire, in terms of information about the project context, was based on the NAPIRE initiative questionnaire [11]. Survey participants were asked to indicate current work practices, namely:

- What types of classes of non-functional requirements do they document in software requirements?
- What sections are present in their documents?
- What problems do they encounter in their work?

The problem of missing or ignoring non-functional requirements was also identified as one of the most important, according to the survey results. Twenty-four percent of respondents indicated it as the most important, along with «Moving targets (changing goals, business processes and/or requirements» (43%), «Incomplete or hidden requirements» (41%), «Weak access to customer needs and/or (internal) business information» (33%).

According to the survey results illustrated in Figure 2, only nine percent of respondents do not specify non-functional requirements. The current practices of business analysts and requirements engineers show a clear prioritization pattern that partially aligns with recommended approaches but reveals significant gaps in coverage of critical quality attributes. Most often, business analysts and requirements engineers capture requirements for usability (64%), security (60%), and performance (57%).

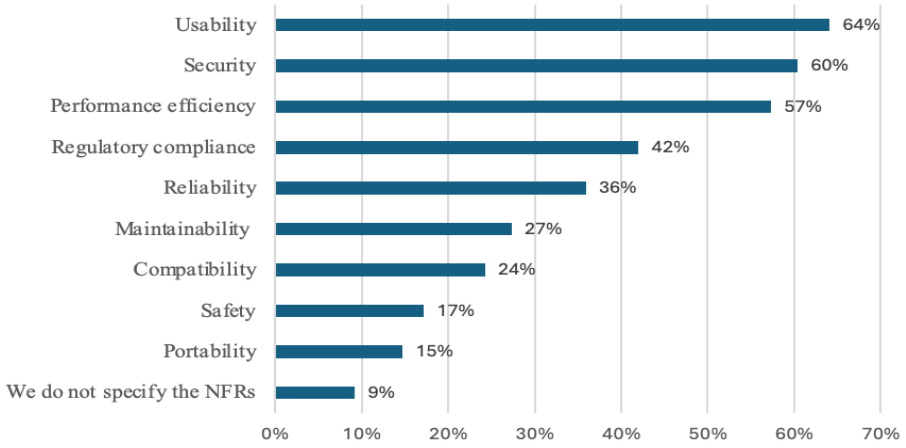

Fig.2. Types of Non-functional requirements in software requirement documents

It can be concluded that analysts intuitively give the highest priority to the user and operational quality characteristics reflected in the ISO model, which have immediate business impact. At the same time, characteristics related to maintenance and system architecture (e.g., Maintainability, Portability, Compatibility) are documented much less frequently. This practice-based prioritization creates a concerning disconnect with the recommended systematic approach to documenting quality requirements. This can be explained by:

- focus on short-term tasks and functionality;
- insufficient involvement of architects in the analysis phases;
- lack of tools for assessing these qualities at early stages.

While Security is appropriately prioritized and aligns with the recommendation to specify it as an early-stage internal architectural requirement, the relatively low documentation rates for Maintainability (27%) and Compatibility (24%) represent significant risks. They can lead to technical debt and operational problems. According to the recommended phased approach, both Maintainability and Compatibility should be treated as foundational characteristics requiring early specification. The data reveal that Reliability (36%) receives moderate attention, although it should be developed in parallel with user requirements at the initial stages, according to best practices. Most concerning is the minimal documentation of Portability (15%) and Safety (17%), despite their potential critical importance in specific project contexts. Therefore, in practice, the recommended step-by-step distribution, which includes starting with the main and internal attributes before moving to operational and context-dependent, is not systematically performed, underestimating the long-term effects of architectural and operational characteristics.

Figure 3 illustrates the most often used section in requirements documents. Sixty percent of respondents indicated that they describe non-functional requirements in a separate section of requirements documents. Thus, this section is among the five most frequently used, along with functional and business requirements, glossary, and user interface

requirements. Note that user interface requirements are also closely related to non-functional requirements and receive special attention. The fact that functional requirements are a key deliverable of the business analyst's work supports the focus on Functional Suitability as a key quality attribute. This indicates recognition of the importance of NFRs and their formalization in the requirements structure (at least at the level of a dedicated section).

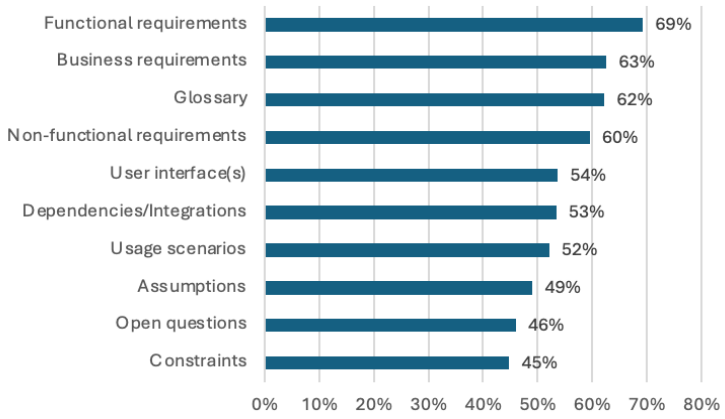

Fig.3. Top 10 most common sections in requirements documents

The analysis of the influence of the project context on the work of business analysts in IT projects, carried out in works [11, 13, 32], allowed us to identify the following dependencies:

- Having a dedicated business analyst role on the team increases the likelihood of identifying and documenting non-functional requirements;
- Using techniques such as user stories, use cases, and acceptance criteria to document functional requirements increases the likelihood of having a separate section for non-functional requirements
- The problem of the lack of non-functional requirements often occurs in projects with a team size of more than 30 people. This can be explained by the fact that in large teams, there is an increased need for a detailed description of requirements, including non-functional ones, for the transfer of information, the long-term use of information, as well as the increased complexity of the solutions being created.
- The use of document templates increases the likelihood of describing non-functional requirements. Implementing such templates at the company or project portfolio level should be recognized as a recommended practice that improves the completeness and clarity of requirements for IT solutions.
- There is a direct correlation between the number of years of experience of a business analyst and the presence of a separate section on non-functional requirements. If for analysts with up to three years of experience, the frequency of creating such a section is 45%, then for analysts with over 5 years of experience, the probability is already 71%. From this, we can conclude that it is advisable to include the study of approaches to working with non-functional requirements in training programs for young specialists. Another solution may be to use templates of requirements documents adapted to specific types of projects created by more experienced analysts and requirements engineers.
- The project category also influences how non-functional requirements are handled. In from-scratch development projects, a dedicated section for non-functional requirements (NFRs) is present in 69% of cases, indicating a systematic focus on specifying performance, security, usability, and reliability parameters at early development stages. For user interface (UI) engineering projects, where the primary goal is redesigning the front-end layer of existing systems, NFRs are documented separately in 61% of instances, reflecting sustained attention to quality attributes despite a UI-centric scope. In reengineering projects, the proportion of separate NFR sections decreases to 57%. This reduction may be attributed to the focus on restoring or improving functional aspects, assumptions regarding existing non-functional requirements (NFRs) in legacy systems, and resource limitations. In such contexts, NFRs are often embedded within functional requirements or addressed through informal communication channels. The lowest separation rate (42%) is observed in product or platform customization projects, where NFRs are frequently integrated into general documentation or implicitly derived from platform standards. These differences suggest that the inclusion and visibility of NFRs vary depending on the type of project, with greenfield initiatives necessitating greater formalization, while customization projects leverage pre-established frameworks.

The analysis revealed a diverse approach to working with quality criteria in IT projects, as well as a significant difference between current practices and recommended methodological approaches. Although 91% of respondents work with non-functional requirements, their priorities are primarily determined by their understanding of the significance of user and operational features, resulting in an underestimation of the architectural and internal attributes of quality. The importance of professional competence in transforming theoretical quality models into practical tools is confirmed by the growing emphasis on documenting non-functional requirements and the experience of a business analyst. The completeness of quality criteria specification is greatly impacted by contextual factors such as team size, project type, and the use of document templates, which suggests the need to adapt approaches to specific project conditions. The findings indicate the importance of standardizing approaches to working with quality attributes by implementing structured methodologies, creating context-specific documentation templates, and training business analysts in software

quality models.

6. Conclusions

This study investigated the applicability of established software quality models from the perspective of IT business analysis and requirements engineering, with particular emphasis on ISO/IEC 25010:2023. By combining comparative analysis and empirical data from a survey of practitioners from Ukrainian and international companies, the research demonstrated that although software quality is widely recognized as a critical success factor, the practical use of structured models remains inconsistent.

The ISO/IEC 25010 model was found to offer the most comprehensive taxonomy of quality characteristics, including clearly defined metrics suitable for both internal and external evaluation. The analysis of metric distribution revealed that quality attributes have varying levels of importance and applicability in the formation of requirements. Functional Suitability and Compatibility emerged as foundational characteristics forming the basis for quality assessment systems. Performance Efficiency, Reliability, and Flexibility require operational testing and should be developed in parallel with user requirements. Interaction Capability demands tailored evaluation approaches that align with specific business contexts and user experience objectives.

The investigation of interdependencies between quality attributes demonstrated that software quality characteristics are interconnected and often have complex relationships. Ensuring software quality requires making balanced decisions regarding the prioritization of quality characteristics, considering their mutual influences. Requirements elicitation and validation processes should include trade-off analysis, and requirements specifications should define not only the desired attributes but also their relative importance and acceptable compromise levels for a specific project. Validation metrics should evaluate individual attributes and their systemic impact on other characteristics. Requirements specifications should define not only the desired attributes but also their relative importance and acceptable compromise levels. The study established that quality requirements documentation should follow a phased approach: initially focusing on foundational and internal attributes (Functional Suitability, Compatibility, Maintainability, Security), followed by operational and context-dependent characteristics (Performance Efficiency, Reliability, Flexibility, Interaction Capability).

Empirical results revealed that non-functional requirements are analyzed and specified in most cases (only 9% of practitioners do not work with them) and are more likely to be documented separately in large, from-scratch projects and by experienced analysts. Analysis shows a significant gap between current IT project practices for handling quality criteria and recommended methodologies. Practice-based prioritization, which often skips a systematic, step-by-step approach, can result in long-term technical debt and operational issues. Templates and organizational practices were found to have a positive influence on the completeness of requirements regarding quality attributes. These findings underscore the importance of institutional support, methodological guidance, and training in implementing a structured approach to handling quality within the requirements elicitation, analysis, and specification phases of IT projects. Business analysts play a pivotal role in tailoring theoretical quality models to project-specific contexts by: selecting domain- and business-relevant quality attributes; composing balanced, interdependency-aware criteria sets; specifying measurable non-functional requirements with precise metrics, thresholds, and acceptance conditions; maintaining traceability between quality and functional requirements; and documenting justified trade-offs among conflicting quality attributes.

Future research should explore the integration of agile-oriented standards (e.g., ISO/IEC 29110), operational metrics (e.g., DORA), and AI-based tools into a unified quality assessment approach that is adaptable to iterative and dynamic development environments.

Overall, this study contributes to the discourse on bridging formal quality models and real-world business analysis practices, emphasizing the need for contextualized, practical frameworks that strike a balance between completeness, project limitations, stakeholder preferences, and alignment with business requirements.

References

[1] D. Gobov and O. Zuieva, "Examining software quality concept: business analysis perspective", Bull. Nat. Tech. Univ. "KhPI"., Ser. *Syst.* Anal. *Control Inf. Technol.*, no. 2(10), 2023: 9–14. DOI: 10.20998/2079-0023.2023.02.02

[2] D. Gobov and O. Zuieva, "Identifying the dependencies between IT project context and business analysis document content", *Innovative Technologies and Scientific Solutions for Industries*, no. 2(24), 2023: 39–53. DOI: 10.30837/itssi.2023.24.039

[3] J. P. Miguel, D. Mauricio, and G. Rodríguez, "A review of software quality models for the evaluation of software products", *International Journal of Software Engineering & Applications*, vol. 5, no. 6, 2014: 31–53. DOI: 10.5121/ijsea.2014.5603

[4] J. A. McCall, P. K. Richards, and G. F. Walters, Factors in Software Quality. Volume I. *Concepts and Definitions of Software Quality*. Sunnyvale, CA: GE Co., 1978.

[5] B. W. Boehm, J. R. Brown, and H. Kaspar, *Characteristics of Software Quality*. Amsterdam: North-Holland Publishing Co., 1978.

[6] M. S. Hemayati and H. Rashidi, "Software quality models: A comprehensive review and analysis", *Journal of Electrical and Computer Engineering Innovations*, vol. 6, no. 1, 2017: 59–76. DOI: 10.22061/jecei.2019.1076

[7] ISO/IEC 9126-1:2001, Software engineering – Product quality – Part 1: Quality model [S]. Geneva, Switzerland: ISO, 2001.

[8] G. L. Saini, D. Panwar, S. Kumar, and V. Singh, "A systematic literature review and comparative study of different software quality models", *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 23, no. 2, 2020: 585–593. DOI: 10.1080/09720529.2020.1747188
[9] N. Yılmaz and A. K. Tarhan, "Meta-models for software quality and its evaluation: a systematic literature review", in *Proc. Int. Workshop on Software Measurement and the 15th Int. Conf. on Software Process and Product Measurement*, Mexico, 2020.
[10] I. G. Ndukwe, S. A. Licorish, A. Tahir, and S. G. MacDonell, "How have views on software quality differed over time? Research and practice viewpoints", *Journal of Systems and Software*, vol. 195, 2023: 111524. DOI: 10.1016/j.jss.2022.111524
[11] D. Gobov and N. Sokolovskiy, "An association rule mining for selection requirement elicitation and analysis techniques in IT projects", *Lecture Notes in Business Information Processing*, vol. 499, 50–65, 2024. DOI: 10.1007/978-3-031-51075-5_4
[12] N. N. U. Januhari, A. Setvanto, and E. Utami, "The influence of cultural factors in software quality assessment models: A systematic literature review", in *Proc. 6th Int. Conf. on Cybern. Intell. Syst. (ICORIS)*, Nov. 2024: 1–6. DOI: 10.1109/icoris63540.2024.10903754
[13] D. Gobov, "Practical study on software requirements specification and modelling techniques*", International Journal of Computing*, vol. 22, no. 1, 2023: 78–86. DOI: 10.47839/ijc.22.1.2882
[14] H. Asfa and T. J. Gandomani, "Software quality model based on development team characteristics", *International Journal of Electrical and Computer Engineering*, vol. 13, no. 1, 2023. DOI: 10.11591/ijece.v13i1.pp859-871
[15] D. Gobov, D. and O. Titlova, "Towards identifying challenges in business analysis on IT projects–a practical study", *Radioelectronic and Computer Systems*, (2), 193-206, 2023. DOI: 10.32620/reks.2023.2.16
[16] S. K. Sharma and M. Khaliq, "Design and development of software quality forensics framework and model", *Multidisciplinary Science Journal*, vol. 6, no. 7, 2024: e2024111. DOI: 10.31893/multiscience.2024111
[17] K. Borana, M. Sharma, and D. Abhyankar, "A novel software quality characteristic recommendation model to handle the dynamic requirements of software projects that improves service quality and cost", *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 7, 2023. DOI: 10.14569/ijacsa.2023.0140762
[18] I. Jacobson, G. Booch, and J. Rumbaugh, *The Unified Software Development Process. Reading*, MA: Addison-Wesley, 1999.
[19] arc42 Quality Model [Online]. Available: https://quality.arc42.org/articles/arc42-quality-model
[20] International Institute of Business Analysis (IIBA), *A Guide to the Business Analysis Body of Knowledge (BABOK Guide)*, Version 3.0. Toronto, Canada: IIBA, 2015.
[21] ISO/IEC 25010:2023, Syst. and soft. engin. – Sys. and soft. Quality Requirements and Evaluation (SQuaRE) – Product quality model. Geneva, Switzerland: ISO, 2023.
[22] ISO/IEC 25019:2023, Syst. and soft. engin. – Sys. and soft. Quality Requirements and Evaluation (SQuaRE) – Quality measure elements. Geneva, Switzerland: ISO, 2023.
[23] ISO 9000:2015, *Quality management systems – Fundamentals and vocabulary*. Geneva, Switzerland: ISO, 2015.
[24] ISO/IEC 25002:2024, Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Quality measurement reference model and guide. Geneva, Switzerland: ISO, 2024.
[25] ISO/IEC 25023:2016, Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Measurement of system and software product quality. Geneva, Switzerland: ISO, 2016.
[26] A. R. García-Borgoñón, M. Mejía, and J. Ospina, "A compliance analysis of Agile methodologies with the ISO/IEC 29110 standard", *Procedia Computer Science*, vol. 60, 2015: 1131–1138. DOI: 10.1016/j.procs.2015.08.179
[27] M. Wiedemann, P. Zimmermann, and S. Wagner, "DevOps and software quality: A systematic mapping, *Computer Science Review*, vol. 38, 2020: 100286. DOI: 10.1016/j.cosrev.2020.100286
[28] A. Trockman, D. Xiong, J. Huang, and L. Zhang, "AICodeReview: Advancing code quality with AI-enhanced reviews", *Journal of Systems and Software*, vol. 205, 2024: 111740. DOI: 10.1016/j.jss.2024.111740
[29] B. Naqvi, A. Seffah, and A. Abran, "Framework for examination of software quality characteristics in conflict: A security and usability exemplar", *Cogent Engineering*, vol. 7, no. 1, 2020: 1788308. DOI: 10.1080/23311916.2020.1788308
[30] A. M. Alashqar, "A survey on the relationships between quality attributes of the ISO9126 model", *International Journal of Engineering and Information Systems*, vol. 5, no. 3, 2021: 29–34.
[31] ISO/IEC 25030:2019, Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Quality requirements framework. Geneva, Switzerland: ISO, 2019.
[32] D. Gobov, N. Sokolovskiy, "An Association Rule Mining for Requirement Document Structure in IT Projects." *Lecture Notes on Data Engineering and Communications Technologies*, vol 242, 2025. DOI: 10.1007/978-3-031-84228-3_27

Authors' Profiles

Dr. Denys Gobov received a master's degree in computer science from the National Technical University of Ukraine "Kyiv Polytechnic Institute" and a Ph.D. in mathematical modeling and computational methods from V.M. Glushkov Institute of Cybernetics of the National Academy of Sciences of Ukraine in 2010. He is currently working as an Associate Professor of the Department of Computer Science and Software Engineering of the National Technical University of Ukraine "Ihor Sikorskyi Kyiv Polytechnic Institute", Kyiv, Ukraine. Denys Gobov has over 20 years of extensive experience in business and system analysis. He has published more than 20 research articles in National and International Journals. Denys Gobov is a vice president on professional development in the Ukraine IIBA chapter, the leading professional association of business analysts. His research area includes business analysis and requirement engineering.

Dr. Oleksandra Zuieva holds a Ph.D. in economics and currently works as an Analytics Consultant at CS Ltd., Kharkiv, Ukraine. She has over six years of experience in business and system analysis and more than six years of experience as a university teacher. Her research area includes business analysis and requirement engineering.

How to cite this paper: Denys Gobov, Oleksandra Zuieva, "Software Quality Attributes in Requirements Engineering", International Journal of Information Technology and Computer Science(IJITCS), Vol.17, No.4, pp.38-48, 2025. DOI:10.5815/ijitcs.2025.04.04