

IMPLEMENTATION OF ISO/IEC 25022: MEASUREMENT OF SOFTWARE QUALITY-IN-USE

A Dissertation by

Bassam Ali Jaradat

Master of Management Information Systems, Friends University, 2000

Bachelor of Science in Aeronautical Engineering, Wichita State University, 1991

Submitted to the Department of Industrial, Systems, and Manufacturing Engineering
and the faculty of the Graduate School of
Wichita State University,
in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

May 2024

© Copyright 2024 by Bassam Jaradat

All Rights Reserved.

IMPLEMENTATION OF ISO/IEC 25022: MEASUREMENT OF SOFTWARE QUALITY-IN-USE

The following faculty members have examined the final copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy with a major in Industrial Engineering.

Gamal S. Weheba, Committee Chair

Sue Abdinnour, Committee Member

Krishna Krishnan, Committee Member

Deepak Gupta, Committee Member

Wajun Si, Committee Member

Accepted for the College of Engineering

Anthony Muscat, Dean

Accepted for the Graduate School

Coleen Pugh, Dean

DEDICATION

I dedicate this dissertation to my lovely wife, Fathieh Jaradat; our children, Malik, Laila, Lieth, Sura, Shaddin; our son and daughter in-law Michael and McKenzie, and our grandchildren, Lucy and Ada. Their unwavering support and guidance have profoundly shaped my academic journey, standing by me through every challenge and triumph. I am immensely grateful for their love and dedication, which have empowered me to reach this milestone.

I also extend my heartfelt gratitude to my beloved sister, Basema Jaradat, whose constant encouragement and belief in my endeavors have strengthened me. Her unwavering support has been invaluable throughout this research journey.

May this doctoral achievement serve as a beacon of inspiration for my children, motivating them to embrace the value of education and pursue their goals with diligence and perseverance.

This dissertation is a testament to my family's love, encouragement, and support, without whom this endeavor would not have been possible.

ACKNOWLEDGMENTS

With profound gratitude, I express my sincere thanks to those who have contributed to the success of this endeavor. I greatly appreciate my advisor, Dr. Gamal S. Weheba, whose invaluable assistance and steadfast guidance have been instrumental throughout this research expedition. Furthermore, I am indebted to my esteemed committee members for their insightful comments, invaluable suggestions, and vast expertise, which have enriched my dissertation and academic pursuits.

To my family and friends, mere words fail to convey the depth of my gratitude for your unwavering love, prayers, encouragement, and steadfast support throughout my educational odyssey. Your presence has been my most significant source of fortitude and inspiration, and I am profoundly grateful for your unwavering faith in me.

I also extend my heartfelt gratitude to the Department of Industrial, Systems, and Manufacturing Engineering faculty and staff at Wichita State University. Your dedication to providing exemplary education and arming students with the requisite tools and skills for their professional journeys has been pivotal in my academic and personal growth.

I offer my most profound appreciation to everyone who has played a role in this expedition. Your contributions have been immeasurable, and I am profoundly thankful for your unwavering encouragement and support. Above all, I thank God for the guidance and assistance throughout this beautiful, challenging, and blessed journey.

ABSTRACT

Software quality encompasses various features and characteristics of a software product or service. These features and characteristics determine the software's ability to meet users' needs and expectations. Software quality must be considered in sensitive systems, such as those used in aerospace, healthcare, or finance. Poor software quality in these domains can have severe consequences, including loss of life, injuries, financial losses, or mission failures. Therefore, maintaining high-quality standards is paramount to mitigate risks and ensure safety and reliability.

Various quality models have been developed to achieve and maintain software quality. These models provide frameworks and guidelines for assessing, measuring, and improving software quality. Each model offers different perspectives and methodologies for assessing software quality. This involves identifying areas for enhancement, addressing deficiencies, and incorporating feedback from users' experience (UX), which plays a crucial role in software quality. A software product may meet all functional requirements but still needs to deliver satisfactory user experience. Therefore, it is essential to consider software quality-in-use measures.

This research investigated the challenges in measuring software Quality in-use following the ISO/IEC 25022:2016 standard. By pinpointing these implementation challenges, the research proposed an implementation guide to simplify the evaluation process and recommended adjustments for future standard iterations. These recommendations are geared towards enhancing the precision of software quality measurement practices.

TABLE OF CONTENTS

Chapter	Page
1. Introduction.....	1
2. Literature Review	4
2.1 Software Quality Models	4
2.1.1 McCall’s Model.....	5
2.1.2 Boehm’s Model.....	7
2.1.3 The FURPS Model.....	7
2.1.4 Dromeys’ Model	9
2.1.5 ISO 9126 Model.....	10
2.1.6 ISO/IEC 25022: 2016 Model	16
3. Discussion and Implementation Guide	23
3.1 Effectiveness Measures	26
3.2 Efficiency Measures.....	28
3.3 General Satisfaction Measures.....	30
3.4 Economic Risk Mitigation Measures	31
3.5 Implementation Guide to ISO/IEC 25022:2016 (E).....	33
4. Conclusions and Future Research.....	37
4.1 Conclusions	37
4.2 Future Research.....	39
REFERENCES.....	40

LIST OF FIGURES

Figure	Page
1. Quality Models.....	5
2. McCall’s Quality Model.....	6
3. Boehm’s Model.....	7
4. The FURPS Model.....	8
5. Dromeys’ Model.....	9
6. ISO 9126 quality model for external and internal quality Model.....	11
7. ISO 9126 quality model for quality in-use (characteristics) Model.....	11
8. Quality in the lifecycle [ISO, 2001].....	12
9. Organization of the SQuaRE series of International Standards.....	13
10. Structure of the Quality Measurement Division.....	15
3.1 Quality in-use Model.....	23
3.2 Effectiveness measure Table 1: ISO/IEC 25022: 2016 (E).....	24
3.3 Effectiveness measure Table 1: ISO/IEC 25022: 2016 (E).....	26
3.4 Effectiveness measure Table 1: ISO/IEC 25022: 2016 (E).....	27
3.5 Effectiveness measure Table 1: ISO/IEC 25022: 2016 (E).....	28
3.6 Effectiveness measure Table 1: ISO/IEC 25022: 2016 (E).....	28
3.7 Effectiveness measure Table 1: ISO/IEC 25022: 2016 (E).....	29
3.8 Effectiveness measure Table 1: ISO/IEC 25022: 2016 (E).....	30
3.9 Effectiveness measure Table 1: ISO/IEC 25022: 2016 (E).....	31
3.10 Economic risk mitigation measures Table 9: ISO/IEC 25022: 2016 (E).....	32
3.11 Economic risk mitigation measures Table 9: ISO/IEC 25022: 2016 (E).....	33
3.12 Task cannot be partially completed.....	34

LIST OF FIGURES (continued)

Figure	Page
3.13 Task can be partially completed.....	36

CHAPTER 1

INTRODUCTION

Quality is pivotal in achieving success and acquiring market share within the software industry. Recent trends indicate a notable escalation in the competition among industry participants to enhance the quality standards of their offerings. Quality is an essential component of plans for companies striving to elevate themselves above their competitors. After all, quality must be present in the day-to-day operations of any organization. Once the concept of quality is understood, it becomes easier to identify the different aspects of quality.

Juran (1988) identified three aspects of quality: quality of design, conformance, and performance quality. The quality of design is the ability to translate customer's needs and expectations into requirements and specifications. Quality of conformance is the ability to achieve design requirements. Quality of performance relates to product performance during use.

According to <https://www.statista.com/> website, revenue in the Software market is projected to reach US\$698.80bn in 2024, and enterprise Software dominates the market with a projected market volume of US\$292.00bn in 2024. Revenue is expected to show an annual growth rate (CAGR 2024-2028) of 5.27%, resulting in a market volume of US\$858.10bn by 2028. Moreover, in global comparison, most revenue will be generated in the United States (US\$353.50bn in 2024). The packaged software market continues to thrive, driven by standardization, evolving tech solutions, and the changing landscape of digital consumption. As businesses and individuals rely more on technology, the demand for bundled software solutions remains robust.

Software standards ensure acceptable quality in both software products and processes. Software quality has evolved over time, initially defined as the adherence to a standard or specification. The International Organization for Standardization (ISO) introduced ISO 9126 (Software Product Evaluation) in 1991, establishing it as the standard framework for assessing software quality. Per this

standard, quality represents a comprehensive product or service assessment, considering various features and characteristics that collectively contribute to its ability to meet specified needs. This definition emphasizes that quality is not limited to one aspect but encompasses a range of attributes. This definition underscores the comprehensive nature of quality, emphasizing its relevance to meeting specific requirements and satisfying user expectations.

This definition emphasizes the comprehensive nature of quality, encompassing various features and attributes that contribute to meeting user requirements and expectations.

ISO 9126 has undergone expansion, resulting in the formulation of one International Standard along with three technical reports. ISO/IEC TR 9126-1:2001 introduces the Software product quality model, outlining the fundamental framework for assessing software quality. Meanwhile, ISO/IEC TR 9126-2:2003 and ISO/IEC 9126-3:2003 offer specific metrics tailored to evaluate software products' external and internal quality attributes, respectively. These standards collectively provide a structured approach to measure and improve software quality across various dimensions. Concurrently, ISO/IEC TR 9126-4:2004 concentrates on quality-in-use metrics, furnishing directives for evaluating software quality from end-user perspectives.

ISO/IEC 25010:2011, a part of the Systems and Software Quality Requirements and Evaluation (SQuaRE) series, represents a significant evolution from its predecessors, ISO 9126 and ISO 9126-1. This updated standard offers a more comprehensive and refined framework for evaluating software quality, incorporating advancements in software engineering practices, and addressing emerging challenges in the field. It provides a structured approach to assess various quality characteristics, including functionality, reliability, usability, efficiency, maintainability, and portability.

In parallel, ISO 9126-2 and 9126-3 have succeeded ISO/IEC 25023:2016, focusing on enhancing system and software product quality measurement. This updated standard offers revised guidelines and metrics for evaluating various quality attributes, ensuring alignment with contemporary

industry practices and requirements.

Additionally, ISO 9126-4 has been revised into ISO/IEC 25022:2016, focusing on quality-in-use within the SQuaRE series of standards. These revisions aim to enhance the effectiveness and relevance of quality assessment frameworks in software evaluation processes.

This research investigated the ISO/IEC 25022:2016 standard and identified implementation challenges in measuring software quality in-use. It proposed an implementation guide to overcome these challenges and provided recommendations for future standard revisions.

Chapter 2 reviews the literature on software quality models; this review aims to investigate how software quality models have evolved significantly in measuring quality. Chapter 3 presents a discussion highlighting the implementation challenges in measuring software quality-in-use following ISO/IEC 25022:2016 standards from the perspective of effectiveness, efficiency, satisfaction, and economic risk mitigation measures. Conclusion and future research are provided in Chapter 4.

CHAPTER 2

LITERATURE REVIEW

2.1. Software Quality Models

Research on software quality has been ongoing since the early days of software development, driven by the need for error-free programs and efficient functionality. Users' demand for higher-quality software has fueled further research in this area, leading to the recognition of software quality as an engineering discipline. According to the IEEE Standard Glossary of Software Engineering Terminology, software quality is defined as “the extent to which a system, component, or process meets specified requirements and satisfies user needs or expectations.” This definition emphasizes the importance of aligning software development efforts with user requirements and expectations to ensure that the end product effectively fulfills its intended purpose.

Over time, significant developments have occurred in software quality models. They fall into two main categories: basic models, which span from 1977 to 2001, and many tailored quality models, which emerged from 2001 onwards.

The basic models follow a hierarchical structure and are adaptable to various software products, focusing on evaluation and improvement. Tailored quality models emerged around 2001, characterized by their specific applications, where the importance of features may vary compared to a general model. These models address the need within organizations and the software industry for specialized quality assessment of individual components. They are derived from Basic Models, notably ISO 9126, by adding or modifying sub-factors to cater to specific domain needs or specialized applications.

The six most significant models shown in Figure 1 are as follows: These include McCall et al.'s (1977) model, Boehm et al.'s (1978) model, the FURPS Model introduced in (1992), Dromey's (1995)

model, and the ISO 9126 standard originating in (1991), later updated to ISO 9126-1 in (2001).

Additionally, subsequent iterations, such as ISO/IEC 9126-2 (2003), focused on external metrics, ISO/IEC 9126-3 explore software's internal characteristics, examining factors such as code maintainability and scalability. ISO/IEC 9126-4 provides quality in use metrics for measuring the attributes defined in ISO/IEC 9126-1 considering effectiveness, productivity, safety, and user satisfaction. These standards offer a holistic framework for analyzing and improving software quality across various dimensions.

Dromey (1995) presented a framework for a software product model that takes a different approach than the two previously presented.

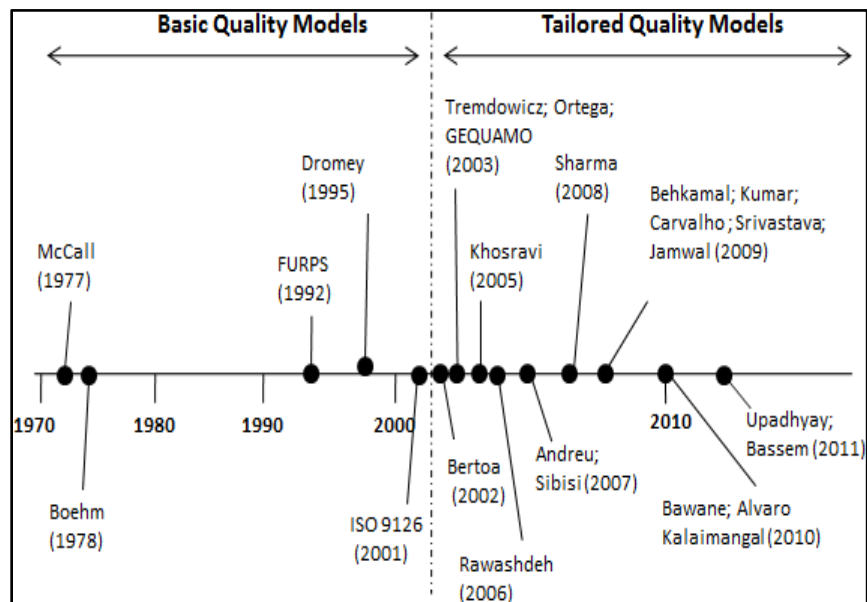


Figure 1: Quality Models

2.1.1 McCall's Model

McCall et al. (1977) introduced a model known as the General Electric Model, which emerged from the US military and was specifically developed for the US Air Force. This model is primarily

geared towards system developers and the system development process, providing a structured framework for evaluating software quality attributes and guiding development efforts to meet the rigorous standards and requirements of military applications.

The model defined product quality using different aspects, categorized into three perspectives: product review (maintenance, flexibility, and testing), product operation (correctness, reliability, efficiency, integrity, and usability), and product transition (portability, reusability, and interoperability).

The critical contribution of the McCall method was recognizing the relationships between quality features and metrics. This model served as the foundation for creating other quality models. However, a significant drawback of the McCall model is its limited accuracy in measuring quality since it relies on simple Yes or No responses. Additionally, it overlooks functionality, which diminishes the user's perspective. Figure 2 shows all three significant perspectives, 11 quality factors, and 22 McCall Quality Model quality criteria.

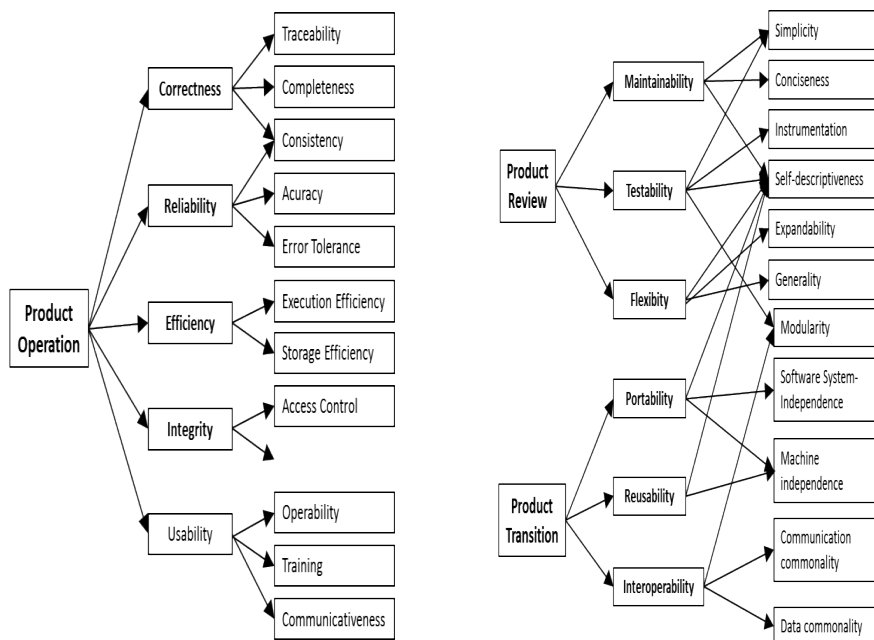


Figure 2: McCall’s Quality Model

2.1.2 Boehm's Model

Boehm, B. W., Brown, H., Lipow, M. (1978) introduced significant enhancements to the McCall model by incorporating large-scale characteristics encompassing various levels. These high-level factors include a) Utility, which assesses the ease, reliability, and efficiency of using a software product; b) Maintainability, covering aspects such as modification, testability, and comprehension; and c) Portability, indicating the ability to continue usage despite changes in the environment as shown in Figure 3.

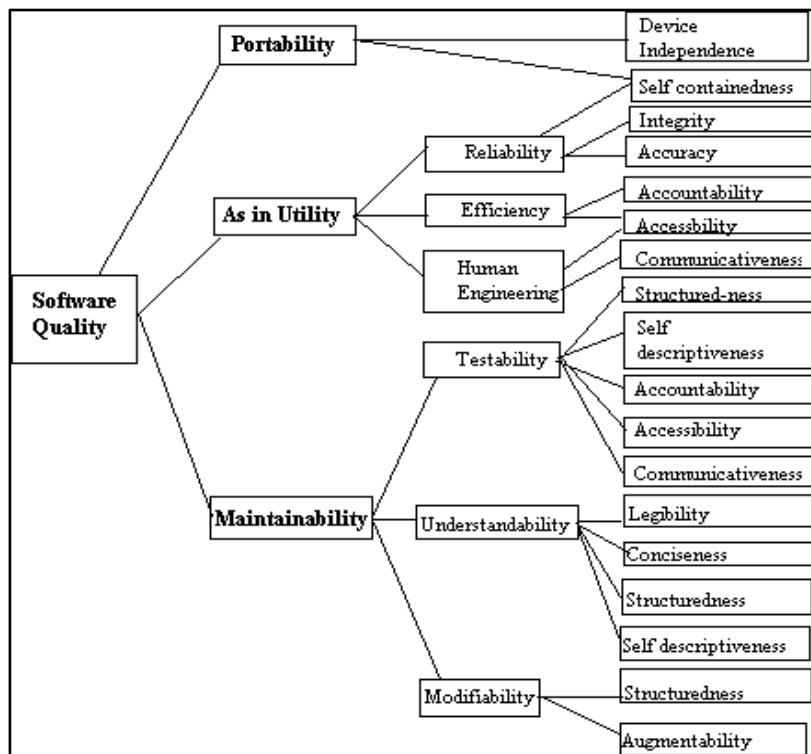


Figure 3: Boehms' Model (1978)

2.1.3 The FURPS Model

Robert Grady (1987) introduced the FURPS model. This model breaks down software characteristics into two main categories of requirements: functional and nonfunctional. Functionality

pertains to the specific features and requirements of the software identified during the early phases of development. On the other hand, nonfunctional, called URPS, encompasses features that support the functional requirements of the software, such as its performance, reliability, usability, and security. These nonfunctional aspects ensure the software meets users' broader goals and expectations beyond its basic functionality.

However, the drawback of the FURPS model is that it overlooks the software product's portability, as shown in Figure 4.

Two key steps are considered when using the FURPS model: prioritizing and defining measurable quality attributes. Grady and Caswell (1987) emphasize the significance of setting priorities because there is often a trade-off involved; achieving one quality characteristic might come at the cost of another. Therefore, it is essential to determine which aspects are most important and where compromises can be made.

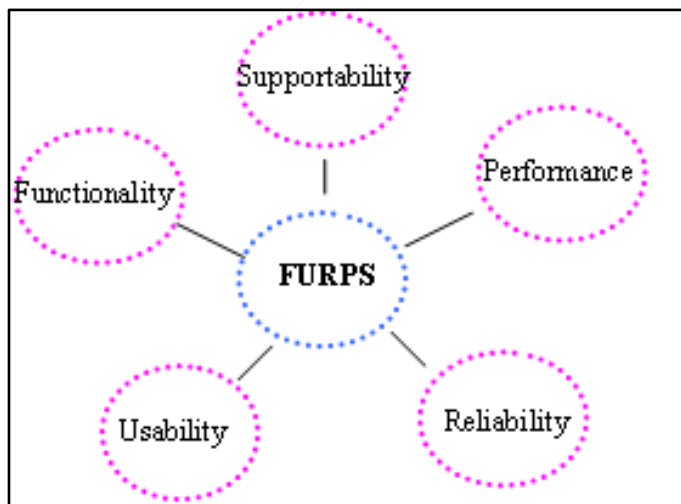


Figure 4: The FURPS Model

2.1.4 Dromey's Model

Dromey (1995) introduced a framework for a software product model, employing a distinct approach from the two previously discussed models. Dromey stated that,

"What must be recognized in any attempt to build a quality model is that software does not directly manifest quality attributes. Instead, it exhibits product characteristics that imply or contribute to quality attributes and other characteristics (product defects) that detract from the quality attributes of a product. Most software quality models fail to deal adequately with the product characteristics side of the problem, and they also fail to make direct links between quality attributes and corresponding product characteristics."

The Dromey model adopts a product quality perspective, resulting in unique evaluations for each product and establishing a dynamic evaluation approach. According to this model, a high-quality product requires all its components to meet quality standards. However, the implementation of this concept needs to be discussed, and the model primarily serves as a theoretical framework for designing more specific models, as shown in Figure 5.

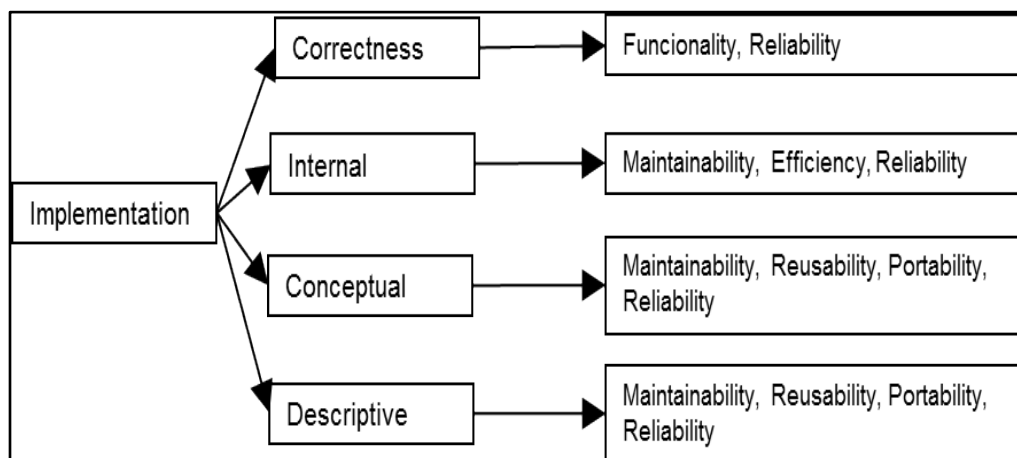


Figure 5: Dromeys' Model (1995)

2.1.5 ISO 9126 Model

The International Organization for Standardization proposed this model in 1991. This standard was based on McCall and Boehm's models and structured similarly. It is divided into four parts, which address the following subjects: quality model, internal metrics, external metrics, and quality-in-use measures.

The primary objective of ISO 9126 is to mitigate any potential misunderstandings between purchasers and suppliers. A key emphasis lies in recognizing that quality characteristics are the highest level within a hierarchical structure of attributes; each characteristic can be broken down into quality sub-characteristics and attributes. At the bottom of the hierarchy, those attribute values are measured using appropriate measuring metrics.

From 2001 to 2004, the ISO published an expanded four-part version containing both the ISO quality models and proposed measures. The ISO 9126 series consists of one International Standard and three technical reports: ISO/IEC TR 9126-1:2001 pertains to the Software Product Quality Model, ISO/IEC TR 9126-2:2003 addresses Software Product External Quality Metrics, ISO/IEC 9126-3:2003 focuses on Software Product Internal Quality Metrics and ISO/IEC TR 9126-4:2004 deals with Software Product Quality-in-use Metrics.

These standards collectively provide a structured framework for evaluating different dimensions of software quality, encompassing both internal characteristics and the product's usability in real-world contexts.

The first part of the two-part quality model lists six characteristics subdivided into twenty-seven sub-characteristics, as seen in Figure 6 below. The second part of the two-part model lists four qualities in-use characteristics, as seen in Figure 7.

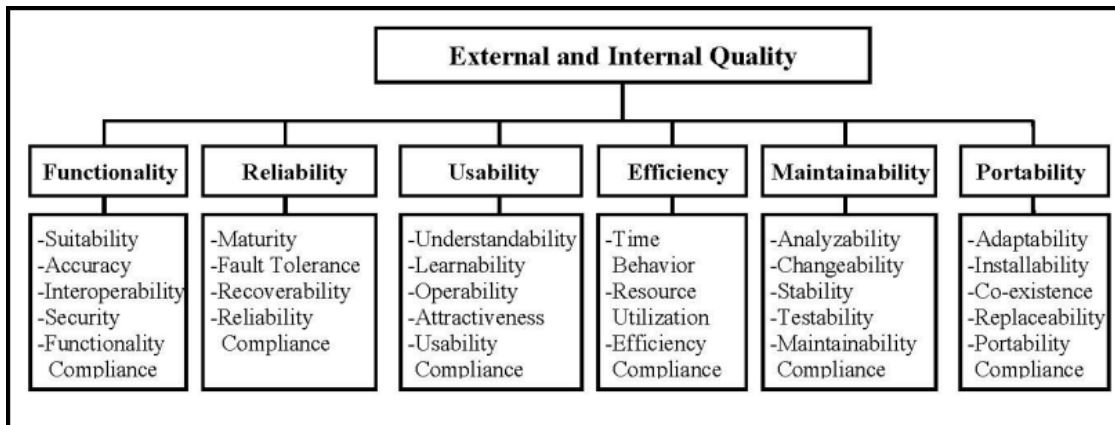


Figure 6: ISO 9126 quality model for external and internal quality (characteristics and sub-characteristics).

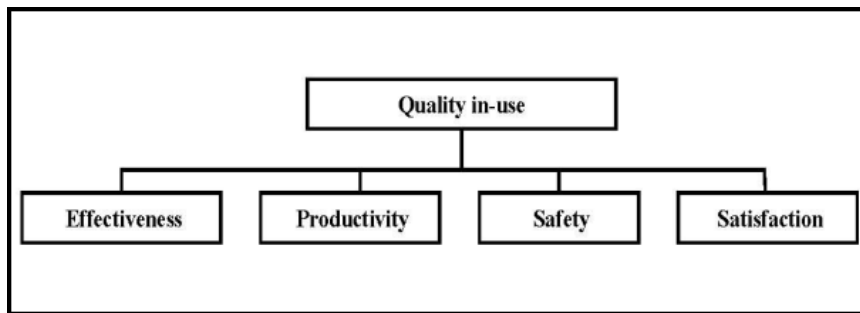


Figure 7: ISO 9126 quality model for quality in-use (characteristics)

The ISO standard illustrates the interrelationship between internal, external, and Quality-in-use attributes of software in Figure 8. Internal quality attributes affect external quality attributes, which, in turn, influence the quality in use. Furthermore, Quality-in-use depends on external quality attributes, while external quality attributes are impacted by internal quality attributes. This interconnectedness emphasizes the significance of considering internal and external aspects of software quality to ensure satisfactory user experiences and overall software effectiveness.

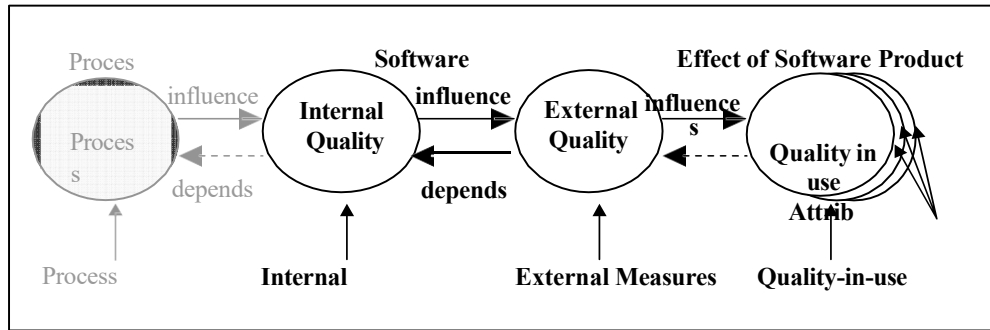


Figure 8: Quality in the lifecycle (ISO 9126)

Al-Kilidar, H., Cox, K., & Kitchenham, B. (2005) discussed the application of ISO/IEC 9126 to assess the quality of software design documents in an empirical study, detailing experiences and findings on an experiment involving 158 final-year participants as part of the Software Project Management course at the University of New South Wales. They were tasked with producing design outputs and planning and tracking their design activities using Microsoft Projects.

The finding highlighted several issues with ISO/IEC 9126 for assessing design quality, including ambiguous concept definitions, overlapping definitions leading to multiple counting in metrics construction, and omitting essential characteristics like reliability and maintainability. It also noted that some measures require inaccessible information or are unsuitable for design documents and the lack of guidelines for overall evaluation.

The ISO/IEC 25000 standard, initially introduced in 2005 as an evolution of the ISO/IEC 9126 standard, which laid the groundwork for software quality assessment. Revised in 2014, ISO/IEC 25000 is part of the SQuaRE (System and Software Quality Requirements and Evaluation) series, providing an updated version with enhancements and adjustments. It served as a framework for organizing the SQuaRE series, depicting families of standards and divisions within.

The primary objective of ISO/IEC 25000 is to steer the development of software products by specifying quality requirements and facilitating the evaluation of quality characteristics, aiming to

ensure higher-quality software products. as shown in Figure 9.

These revisions aimed at keeping the standards relevant and effective in addressing the evolving needs and challenges in software quality management and evaluation. Figure 9 (adapted from ISO/IEC 25000) illustrates the organization of the SQaRE series, which represents families of standards, further called divisions.

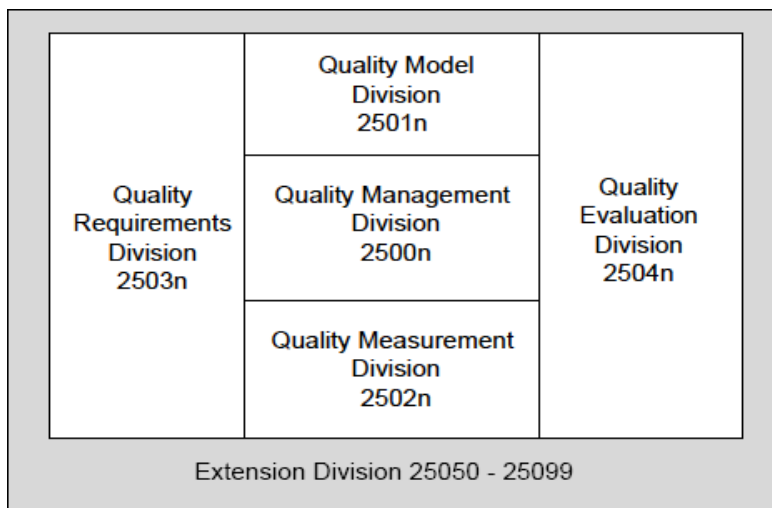


Figure 9: Organization of SQaRE series of International Standards

The ISO/IEC 2502n Quality Measurement Division is a set of standards focused on quality measurement in systems and software engineering, as shown in Figure 10. It is designed to help organizations assess and improve the quality of their software products and services. These standards cover a variety of quality dimensions critical for assessing software comprehensively. They include functionality, which evaluates if the software performs its tasks effectively.

Reliability assesses the software's consistency over time and under various conditions. Usability examines how user-friendly the software interface is. Efficiency measures resource usage, while maintainability evaluates how easily the software can be modified. Portability assesses its ability to function across different platforms. Together, they ensure the software meets high standards of

performance and usability.

Each aspect plays a critical role in assessing and enhancing the overall quality of a product or service, ensuring that it meets the needs and expectations of users while maintaining high standards of performance and usability. Within this standard, the quality measurements division focuses on defining and implementing metrics and measures to evaluate different aspects of software quality. This division involves several key components:

1. **Quality Metrics Definition:** This involves defining specific metrics to assess different quality characteristics of software systems quantitatively. For example, metrics could include measures of reliability (e.g., mean time between failures), performance (e.g., response time), or usability (e.g., user satisfaction ratings).
2. **Measurement Methods:** These standards guide selecting appropriate measurement methods to collect data for quality metrics. This may include techniques such as surveys, observations, or automated tools for collecting metrics from software systems.
3. **Data Analysis:** Once data is collected using the defined metrics and measurement methods, ISO/IEC 2502n outlines approaches for analyzing the data to derive meaningful insights into the software system's quality. This may involve statistical analysis, trend analysis, or comparisons against predefined quality benchmarks.
4. **Reporting and Interpretation:** The standard also addresses effectively communicating quality measurement results to stakeholders. This includes creating reports summarizing the findings, interpreting the data's implications for decision-making, and providing recommendations for improvement.
5. **Continuous Improvement:** These standards emphasize the importance of using quality measurements in a continuous improvement process. Organizations are encouraged to regularly assess and monitor software quality using established metrics, identify areas for

improvement, and implement corrective actions to enhance overall quality over time.

Overall, the quality measurements division within ISO 2502n provides a structured approach to evaluating and improving software quality, helping organizations deliver more reliable, efficient, and user-friendly software products and services.

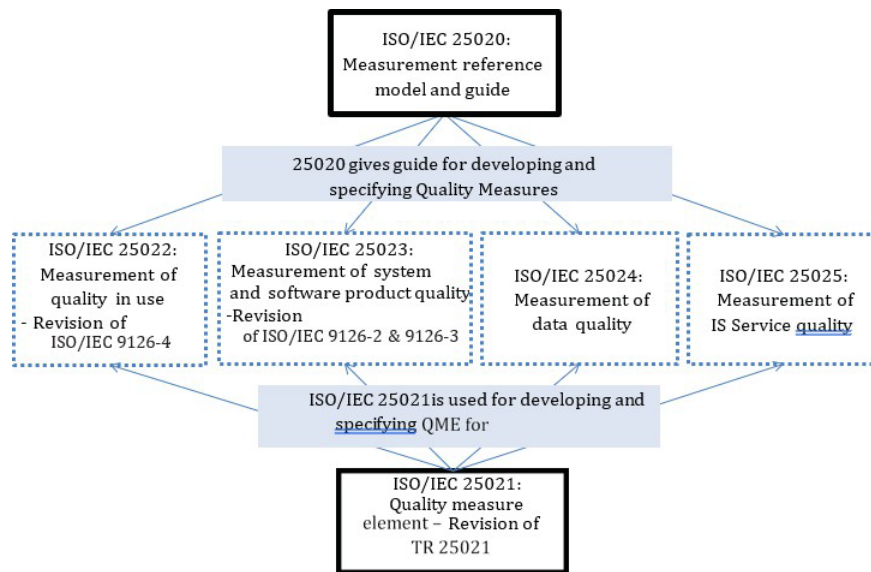


Figure 10: Structure of the Quality Measurement Division

Over the past two decades, companies have used commercial off-the-shelf software (COTS), while others have preferred to develop theirs in-house. The quality of these software systems will need to be evaluated. Several software quality models were presented earlier that can be used to evaluate these software systems. Most of these models share hierarchical characteristics of efficiency, functionality, maintainability, portability, reliability, and usability.

Al-Qutaish (2009) noted that many metrics in ISO 9126 produced results falling within the 0 to 1 range, making it easier to convert to percentage values. However, he proposed a viewpoint

suggesting that the results could be better understood if expressed in qualitative terms. For example, assigning qualitative labels based on percentage results could enhance clarity in the context of task completion. A 100% result could be denoted as "excellent" task completion, while an 80% result might be categorized as "very good," providing a more user-friendly interpretation of the metrics.

Al-Qutaish (2010) conducted an analytical and comparative study on five quality models, highlighting their strengths and weaknesses. McCall's model was found to rely on subjective measurements, potentially introducing variability in assessment due to the judgment of individuals answering 'yes' or 'no' questions. In contrast, the ISO 9126-1 quality model was praised for its comprehensive framework, incorporating three characteristics as sub-characteristics derived from others.

The FURPS model, developed for IBM Rational Software Company, was noted for its specialization in specific organizational needs. However, the study identified a common issue across McCall's, Boehm's, Doromey's, and FURPS models: the need for more precise definitions and connections between lower-level metrics and upper-level criteria. This necessitates better alignment for practical evaluation, particularly in McCall's model, where metrics should be linked to corresponding quality criteria.

2.1.6 ISO/IEC 25022:2016 Quality-in-use Model

The ISO/IEC 25022:2016 standard known as the Software Quality Model or SQuaRE (Software Quality Requirements and Evaluation) quality-in-use measures superseded and expanded upon ISO/IEC TR 9126-4:2004 by offering a more structured and modular approach to software quality evaluation shown in chapter 3 Figure 3.1.

Quality-in-use refers to how particular users can utilize a product or system to fulfill their requirements and accomplish predefined objectives with effectiveness, efficiency, satisfaction, and minimal risk within specific usage contexts.

Alrawashdeh, T. A., Muhairat, M. I., & Althunibat, A. (2013) conducted a literature review on evaluating the quality of software in ERP systems and presented a model proposed for assessing the quality of ERP systems within higher education institutions. They used the ISO 9126 standard to define quality characteristics and sub-characteristics. They highlighted two main contributions: a comparative analysis of existing quality models and identifying specific quality characteristics for ERP systems in this context.

The review stressed the importance of extending the study to rank these quality characteristics, offering insights into their interrelations and impact on overall system quality. Their research promised to enhance understanding and provide practical guidance for improving ERP systems in higher education settings.

Miguel, J.P., Mauricio, D., & Rodriguez, G.D. (2014) pointed out that while general models are available for assessing software quality, their application to specific cases can be challenging. Tailored quality models, developed because of basic models and focused on specific domains, offered more specificity but were limited in scope. The ISO 9126 model, updated to ISO 25010 in 2011, expanded the fundamental characteristics from six to eight, serving as a reference point for future model development.

None of the studied models incorporated communication as a quality factor, despite its critical importance in complex systems and the Internet era. Moreover, their review highlighted the need to consider user communities, particularly in free/open-source software, as a high-level feature that influences both construction and acceptance.

Haboush et al. (2014) explored the evolution of software product quality models, focusing on maintainability. They noted the transition from ISO 9126 to ISO 25010, which brings about

modifications, including changes to maintainability. However, the study needed more clarity in defining maintainability and its sub-characteristics within both models.

Despite revisions, ISO 25010 needed to provide a standardized definition or clear evidence of modifications to maintainability compared to ISO 9126. The analysis emphasized the necessity for a more transparent framework to assess software maintainability across various quality models.

Rochimah, S., Rahmani, H. I., & Yuhana, U. L. (2015) explored the usability evaluation of the administration module within an academic information system (AIS), specifically focusing on the usability characteristics outlined in ISO/IEC 9126. These characteristics encompass understandability, learnability, operability, attractiveness, and usability compliance, each with corresponding metrics for evaluation. Software attributes such as feature, interface, message, operation, input, and regulation derive parameter values for the metrics.

Jaradat, B., Weheba, G., & Kanan, M. (2015) conducted a literature review on traditional cost of quality models and current software industry trends, particularly in evaluating the cost of software quality based on software quality in-use post-release performance. They recognized the need for new research to estimate these costs from users' perspectives. To address this gap, they developed a cost model that considers deviations from performance targets set by ISO 9126-4 quality in-se standard.

This model incorporated results from a survey instrument designed to gauge users' perceptions of software quality. Discrepancies between standard performance targets and users' perceptions indicated the cost of poor quality. The model aimed to assess the return on investment for software replacement, upgrade, or continued use decisions by focusing on users' viewpoints.

Gong, Lu, and Cia (2016) indicated that the inconsistency in measurement units for quality measures might pose a challenge. Different measurement units within the same characteristic and sub-characteristic can create difficulties during the evaluation phase. They also stated that the units of the quality measures could be more consistent in both ISO/IEC 25022:2016 and ISO/IEC 25023:2016.

The measurement results have different variation tendencies. Notably, in the performance efficiency measures in ISO/IEC 25023:2016, for example, the results for time behavior sub-characteristics are considered better when smaller. In contrast, the results for the capacity sub-characteristics are considered better when larger. This discrepancy may lead to confusion for users of the standards when interpreting the results.

Santos et al. (2016) highlighted the critical role of User Interface (UI) quality in facilitating effective communication and interaction within software systems. They pointed out developers' efforts to implement strategies for creating user-friendly designs. Usability, as defined by ISO 9126, covers attributes related to ease of use and user judgment. Usability, however, encompasses multiple components, including learnability, efficiency, error handling, and satisfaction, underscoring its subjective nature in software engineering.

Despite the importance of metrics in assessing quality characteristics, their relationship often needs to be more adequately defined, as noted by previous research. ISO 9126 outlines quality features and sub-features in its initial part. However, subsequent sections lack clear connections between metrics and their measured attributes, leading to ambiguity in assessing software quality.

Zhao, Y., Gong, J., Hu, Y., Liu, Z., & Cai, L. (2017) analyzed quality evaluation based on ISO/IEC SQuaRE series standards and its considerations and pointed out that the Quality in-use model defines five characteristics - effectiveness, efficiency, satisfaction, freedom from risk, and context coverage - which are essential in assessing software from the perspective of its users, with each characteristic relevant to various stakeholders' activities, such as maintenance for developers.

They pointed out that the measurement functions in ISO/IEC 25022:2016 are linear functions, such as $X=A/B$ or $X=1-A/B$. However, linear functions may need to represent the quality measure adequately in some cases. In such cases, it becomes necessary to consider non-linear functions, especially in the context of value perception. Unlike linear functions that exhibit uniform adjustments

with variations in A, non-linear functions introduce changes in the resulting variable X and offer the advantage of mapping the independent variable into a recognizable range or level, which can be beneficial in certain situations.

Pradanita, W. R., Ni'Mah, A. T., Rochimah, S., & Adiputra, F. (2019) examined the evaluation of usability characteristics for the administration module of an academic information system (AIS), focusing on ISO/IEC 9126 standards. The evaluation aims to assess usability behavior within the AIS administration module, considering six sub-characteristics: understandability, learnability, operability, attractiveness, and usability compliance. Metrics are employed for each sub-characteristic to gauge evaluation results, with parameters derived from software documentation. Software attributes such as feature, interface, message, operation, input, and regulation are analyzed.

The evaluation revealed that the learnability characteristic yielded a zero value, indicating a lack of user assistance features within the AIS student administration module. While other sub-characteristics yield non-zero values, they are not deemed perfect. The overall results suggest a need for evolution within the AIS to enhance its quality.

Pradanita and Rochimah (2020) completed an assessment based on the ISO/IEC 25022:2016 Standard. Their research aimed at demonstrating the impact of effectiveness, efficiency, satisfaction, and freedom from risk on the overall quality-in-use of digital wallet products. They concluded that satisfaction demonstrated the strongest correlation (0.939), whereas freedom from risk exhibited the weakest correlation (0.730). This explained the high level of user satisfaction with the product despite the presence of potential risks. They developed a conceptual framework for explaining observed correlations, making it more straightforward to develop hypotheses.

Hashim, N. L., Yusof, N., Hussain, A., & Ibrahim, M. (2022) completed a study investigating the User Experience (UX) dimensions of e-procurement systems. Guided by the SQuaRE standards, specifically, ISO 25022:2016 and ISO 25023:2016, the study identified UX dimensions by referencing

their characteristics and definitions to categorize them within the literature.

The analysis findings revealed that several critical user experience (UX) dimensions, including satisfaction, security, transparency, efficiency, and reliability, are consistently emphasized in e-procurement studies. These dimensions emerged as prominent factors influencing the user's perception and interaction with e-procurement systems.

Additionally, the findings underscore the importance of measuring UX dimensions like satisfaction, security, transparency, efficiency, and reliability for e-procurement and similar systems like e-commerce. User satisfaction is paramount due to its direct impact on overall user experience. Organizations can better understand and address user needs by prioritizing the measurement of these UX dimensions, ultimately enhancing the usability, functionality, and overall quality of e-procurement and e-commerce systems.

Burgos Robles, L. J., & Huanca Torres, F. A. (2023) presented a quality assessment model for human talent management systems based on ISO 25022, employing the Engineering Cycle methodology. The model provides a comprehensive overview of the system's performance and user satisfaction by evaluating Efficiency, Effectiveness, and Satisfaction.

Through a case study, the model's ability is demonstrated, highlighting its utility in identifying areas for improvement and ensuring compliance with user perspectives. The study concluded that the ISO 25022-based model effectively gauges the quality in the use of human talent management systems, offering valuable insights for ongoing enhancement and adherence to quality standards in software development.

Robles and Torres (2023) evaluated the quality-in-use of a human talent management information system based on the ISO/IEC 25022:2016 standard. A specialized quality evaluation model, structured and applied following the standard, was utilized as a practical instrument to identify shortcomings and initiate continual improvements in the organization's processes. They evaluated

efficiency, effectiveness, and satisfaction. They calculated the proportion of tasks completed and the time efficiency. The evaluation indicated that the levels achieved for each characteristic were enough to determine how well the system aligns with the user's perspective and measure the overall quality.

CHAPTER 3

DISCUSSION AND IMPLEMENTATION GUIDE

Software quality-in-use is a key factor in the success of any software from the users' perspective based on their preferences. With the large amount of software published online and the development of mobile applications, users need to find the most suitable software to fulfill their needs.

The ISO/IEC 25022:2016(E) Standard is a Software Quality Requirements and Evaluation (SQuaRE) series component. This standard describes measures associated with quality-in-use and addresses users' interactions with the software. The standard characterizes Quality-in-use as the extent to which a product or system enables specific users to fulfill their requirements and attain predefined objectives with effectiveness, efficiency, satisfaction, and devoid of risk within designated usage contexts.

The standard provides sub-characteristics for satisfaction, freedom from risk, and context coverage. Satisfaction includes usefulness, trust, and pleasure (user experience). Freedom from risk includes mitigation of economic risks, health and safety risks, and environmental risks. Context coverage includes context completeness and flexibility, as shown in Figure 3.1.

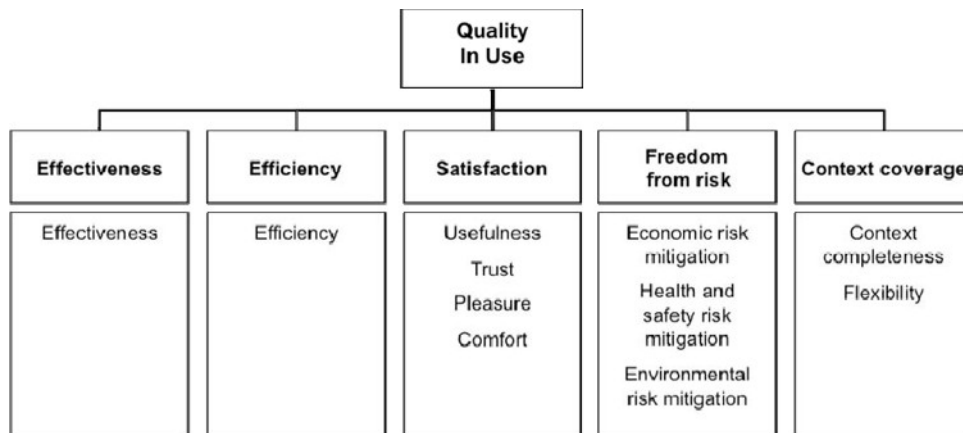


Figure 3.1: Quality-in-use model

The standard contains 13 tables and provides a total of 36 measures (24 general and 12 special). General measures labeled with the letter G are generally applicable and could be used in a wide range of situations. Special measures are labeled with the letter S and used for specific needs. The format for each table is based on Clause 8 of the standard. It provides the identification code (ID), name of the quality measure, a description of the information provided, measurement function, and the method that can be used to obtain its value shown in Figure 3.2, an example of Table 1: Effectiveness measures.

Per the standard, usability has been defined as quality-in-use, focuses on users' perceptions, and refers to the subset composed of effectiveness, efficiency, satisfaction, and context coverage.

ID	Name	Description	Measurement function	Method
Ef-1-G	Tasks completed	The proportion of the tasks that are completed correctly without assistance	$X = A/B$ A = Number of unique tasks completed B = Total number of unique tasks attempted	Measure user performance
NOTE 1 This measure can be measured for one user or a group of users. NOTE 2 If tasks can be partially completed, the Objectives achieved measure is more appropriate. NOTE 3 If the tasks are of different complexity, weighted tasks could be used in the formula: $X = \sum_{(i=1..n)} W_i \times A_i/B$ where i is the number of the task, and W_i represents the difficulty of that task where the total sum of $W_i = 1.0$. NOTE 4 This could be applied either to the tasks identified in the requirements or to the tasks attempted by the user.				
Ef-2-S	Objectives achieved	The proportion of the objectives of the task that are achieved correctly without assistance	$\{X = 1 - \sum A_i \mid X \geq 0\}$ $A_i =$ Proportional value of each missing or incorrect objective in the task output (maximum value = 1)	Measure user performance
Ef-4-G	Tasks with errors	Proportion of tasks where errors were made by the user	$X = A/B$ A = Number of tasks with errors B = Total number of tasks	Measure user performance

Figure 3.2: Effectiveness measure Table 1: ISO/IEC 25022: 2016 (E).

The standard avoids subjective assessments by proposing objective measures for certain quality attributes. However, effectiveness, efficiency, and satisfaction are highly correlated, as noted in Note

1 of Table 2 on page 12 of the standard, and can lead to redundancy in measurement, as shown in Figure 3.7.

Despite this correlation, the standard does not explicitly encourage organizations to quantify the extent of these relationships. The standard considers general satisfaction as an independent measure, not a direct correlation of software quality, which can reflect users' overall perception and feeling towards the software.

Furthermore, the standard lacks clear guidelines or methods for identifying tasks and errors that are independent versus those that are dependent on each other. With this distinction, it could be more accessible to prioritize tasks or understand the impact of errors on the overall software assessment.

The standard is complex and challenging to understand, especially for individuals and organizations without prior experience in software quality evaluation. In addition, it conducts thorough evaluations according to the standard demands of substantial resources, including time, expertise, and possibly financial investment. This could be a barrier, particularly for small organizations with limited resources, as they may need help allocating the necessary resources to adhere to the standard effectively.

According to the standard, quality-in-use is determined by the inherent quality of the software or computer system and the specific conditions and environment in which the product is utilized. This encompasses factors such as user characteristics, task complexity, organizational processes, and environmental constraints, all of which can significantly influence the effectiveness, efficiency, satisfaction, and safety of the product during its use. Therefore, comparisons of the quality-in-use of a software product or system are only valid when the measures are made in the same context of use.

This author has identified the following challenges in measuring user performance:

3.1. Effectiveness Measures:

Table 1, page 11 in the ISO/IEC 25022:2016 (E) standard defines effectiveness as the “accuracy and completeness with which users achieve specified goals.” This table contains four general measures and one special measure. The following are identified difficulties that could be faced in utilizing these measures shown in Figure 3.3:

ID	Name	Description	Measurement function	Method
Ef-1-G	Tasks completed	The proportion of the tasks that are completed correctly without assistance	$X = A/B$ A = Number of unique tasks completed B = Total number of unique tasks attempted	Measure user performance
Ef-2-S	Objectives achieved	The proportion of the objectives of the task that are achieved correctly without assistance	$\{X = 1 - \sum A_i \mid X \geq 0\}$ A _i = Proportional value of each missing or incorrect objective in the task output (maximum value = 1)	Measure user performance
NOTE: Each potential missing or incomplete component is given a weight A _i based on the extent to which it detracts from the value of the output to the business or user. (If the sum of the weights exceeds 1, the quality measure is normally set to 0, although this can indicate negative outcomes and excessive risks.) The scoring scheme is refined iteratively by applying it to a series of task outputs and adjusting the weights until the results obtained are repeatable, reproducible, and meaningful.				

Figure 3.3: Effectiveness measures Table 1: ISO/IEC 25022: 2016 (E).

- 3.1.1) Using accurately as a synonym for correctly may present a challenge in measuring effectiveness in some contexts. For example, in data processing software, if an output of 0.529 represents the accurate outcome, an output of 0.53 may be considered incorrect. In this scenario, a threshold or tolerance may be needed to determine how close the output is to the actual value (accurate) to be considered a correct output. For consistency, these two measures should have been described as the proportion of tasks/objectives that are completed accurately.
- 3.1.2) Accomplishing a task or objective correctly without assistance suggests that the users may not receive any help or assistance. According to Clause 6, page 8 of the standard, it is

emphasized that users should receive assistance and support consistent with what they would encounter in the actual operational environment. Internal help functions are typically available to users during normal operations. However, the standard does not distinguish between users utilizing internal help functions and users asking for external assistance from the evaluation administrator or technical personnel. It would be more appropriate to describe these measures as the proportion of tasks/objectives that are accurately completed without external assistance.

3.1.3) While the standard allows for assigning different weights to tasks based on their complexity, it does not account for the impact of users' experience correctly (accurately) completing the tasks. No guidelines accounted for the users' skills in calculating the measures. The standard could have suggested a similar weighing scheme for users' skills and familiarity with the system.

3.1.4) When counting the number of errors in a task in Ef-3-G, the standard does not distinguish between corrected and uncorrected errors, as shown in Figure 3.4.

ID	Name	Description	Measurement function	Method
Ef-3-G	Errors in a task	The number of errors made by the user during a task	$X = A$ A = Number of errors made by the user during a task	Measure user performance
NOTE 1 The number of errors the user makes can include all errors, only uncorrected errors, or only errors that result in the task not being completed correctly.				
NOTE 2 Measures of counts of errors can be used to make comparisons between the same task carried out in different circumstances, for example, when comparing different versions of a system under development.				
NOTE 3 To compare errors made in different tasks, the number of errors could be related to the number of actions in each task.				
NOTE 4 It is only appropriate to make comparisons if errors have equal importance or are weighted.				
NOTE 5 Errors can be analyzed using a user-by-problem matrix indicating how many users had which problem and in which combination.				
Ef-4-G	Tasks with errors	Proportion of tasks where errors were made by the user	$X = A/B$ A = Number of tasks with errors B = Total number of tasks	Measure user performance

Figure 3.4: Effectiveness measures Table 1: ISO/IEC 25022: 2016 (E).

It will be challenging to identify the number of errors made in situations where the user continues to perform the task using erroneous inputs. The errors are counted without consideration of their severity or impact. If errors have different severity, a seriousness classification may be applied. Serious errors that can go undetected by the system should be penalized the most. Measure EF-5-G, shown in Figure 3.5, accounts only for the frequency at which a specific error is made by the users.

ID	Name	Description	Measurement function	Method
NOTE: The notes to Ef-3-G apply.				
Ef-5-G	Task error intensity	Proportion of users making an error	$X = A/B$ A = Number of users making an error B Total number of users performing the task	Measure user performance
NOTE: The notes to Ef-3-G apply.				

Figure 3.5: Effectiveness measures Table 1: ISO/IEC 25022: 2016 (E).

3.2. Efficiency Measures:

Table 2, page 12 in the standard, represents all efficiency measures as recommended in Claus 8 of the ISO/IEC 25022:2016. Efficiency is defined by the "resources expended about the accuracy and completeness with which users achieve goals." This set of measures consists of one general and five special measures. The following difficulties have been identified:

3.2.1) In measuring the task time Ey-1G, the standard was used successfully as a synonym for accurately describing the measure, as shown in Figure 3.6.

ID	Name	Description	Measurement function	Method
Ey-1-G	Task time	The time taken to successfully complete a task	$X = T$ T = Task time	Measure user performance
NOTE Learnability (see ISO/IEC 25023) can be measured by the time taken by a normal user to complete a task in comparison with the time taken by an expert, and how these changes with repeated usage.				

Figure 3.6: Efficiency measures Table 2: ISO/IEC 25022: 2016 (E).

This may represent a challenge in calculating the measure. It would have been more appropriate to describe the measure as the time taken to complete a task accurately.

3.2.2) According to the standard, time efficiency (Ey-2-S), shown in Figure 3.7, Note 1 acknowledged the correlation between efficiency, effectiveness, and task time.

ID	Name	Description	Measurement function	Method
Ey-2-S	Time efficiency	The efficiency with which users achieve their objectives over time when using the system	$X = A/T$ A = Number of objectives achieved T = Time	Measure user performance
NOTE 1: Time efficiency is a measure of productivity: the number of objectives achieved for every unit of time. Efficiency increases with increasing effectiveness and reducing task time. It enables comparisons, for example, between fast, error-prone interfaces and slow, easy interfaces. NOTE 2 If Ef-1-G tasks completed have been measured, time efficiency can be measured as tasks completed/ time. This measures the proportion of tasks completed successfully for every unit of time. A high value indicates a high proportion of successful tasks in a small amount of time. NOTE 3 The time efficiency could be compared with that of an expert, with time efficiency for a different product or version, or with completing the task manually. NOTE 4 If the objectives achieved have different values, they could be weighted.				

Figure 3.7: Efficiency measures Table 2: ISO/IEC 25022: 2016 (E).

This indicates redundancy in measuring user performance. It would have been more appropriate for the standard to specify one measure (Task time) only or encourage users to evaluate the strength of the correlation between these measures and their impact on overall user satisfaction.

3.2.3) Fatigue consequences (Ey-6-S) is one of the efficiency measures shown in Figure 3.8.

A combination of user performance methods and automated data collection can be used.

ID	Name	Description	Measurement function	Method
Ey-6-S	Consequences of fatigue	The decrease in human performance after continuous use	$X = 1 - A/B$ A = Current performance B = Initial performance	Measure user performance or automated data collection

<p>NOTE 1 Applies to continuous use by an experienced user.</p> <p>NOTE 2 Performance refers to any appropriate measure of effectiveness or efficiency (if necessary, normalized so that a larger number is better).</p> <p>NOTE 3 Physiological measures can be used to assess the effects of fatigue.</p> <p>NOTE 4 Personal assessment of fatigue can be measured using a questionnaire.</p> <p>NOTE 5 Measures of fatigue are only appropriate for experienced users carrying out repetitive tasks. NOTE 6 Computer systems and working practices can be designed to reduce fatigue.</p> <p>NOTE 7: Closer to 0 is better.</p>

Figure 3.8: Efficiency measures Table 2: ISO/IEC 25022: 2016 (E).

According to Note 1, the measure applies to continuous use by an experienced user only. However, the standard does not suggest a period for the assessment. Multiple task times could have been suggested to facilitate the assessment.

3.3. General Satisfaction Measures.

Per the standard, general satisfaction is defined as "the degree to which user needs are satisfied when a product or system is used in a specified context of use and noted to encompass users who do not directly interact with a system, only accomplishing their purpose and building trust in the system are relevant factors, and categorizes users into primary, secondary, and indirect, clarifying their roles and interactions with the system. In addition, users' needs include their desires and expectations associated with using a product, system, or service. Exceeding desires and expectations is a means of significantly increasing satisfaction and improving the user experience".

3.3.1) The standard ignores the need to report the reliability of questionnaires when evaluating satisfaction measures (SUs-1-G) in Table 3, page 14 of the standard shown in Figure 3.9. Failure to address the necessity of reporting the reliability of questionnaires when assessing satisfaction measures is a significant oversight. To rectify the oversight, the proposed solution will be to determine which reliability metrics are most appropriate for assessing satisfaction measures through questionnaires. Standard metrics include

internal consistency (e.g., Cronbach's alpha), test-retest reliability, and inter-rater reliability.

ID	Name	Description	Measurement function	Method
SUs-1-G	Overall satisfaction	The overall satisfaction of the user	$X = \sum A_i$ $A_i = \text{Response to a question}$	Questionnaire
NOTE Examples of overall measures of satisfaction are the Net Promoter Score [18] and Single Ease Question [20].				

Figure 3.9: General satisfaction measures Table 3: ISO/IEC 25022: 2016 (E)

3.4. Economic Risk Mitigation Measures.

Economic risk mitigation measures per the standard to "assess the impact of quality on economic objectives related to financial status, efficient operation, commercial property, reputation, or other resources that could be at risk or provide opportunities." This set of measures listed in Table 9, pages 18 and 19 of the standards consists of five general measures and three specials. Business analytics is the method proposed for calculating the measures. However, the following difficulties have been identified:

- 3.4.1) The standard's misuse of the term "additional benefits" in the return on investment (ROI) calculation to measure (REc-1-G) which deviates from conventional business practice and terminology. In the context of (ROI), "additional benefits" refers to the extra value or advantages gained beyond the initial investment. To align with established practices, it would be more appropriate for the standard to define (ROI) as a financial metric assessing the profitability of an investment relative to its cost. This would involve calculating (ROI) by dividing the net profit generated, derived from total return or total revenue minus the initial cost, by the initial cost of the investment and expressing it as a percentage rather than a ratio. By refraining from such misuse, clarity and consistency can be maintained in interpreting and applying (ROI shown in Figure 3.10).

3.4.2) In the calculation of the time to achieve a return on investment (REc-2-G), as shown in Figure 3.10, the standard relies on the concept of expected return on investment, defined in Note 2 under the 8.5.2 economic risk mitigation measures section as "the expected values of the measures can be estimated, based on actual values from historical data." This time measure lacks a benchmark for a desirable or acceptable (ROI) value from a business perspective pertaining to the software's use post-deployment. Instead, it would have been more appropriate for the standard to adopt measures such as time to achieve payback or time to achieve zero return on investment (the breakeven point) and incorporate values calculated from REc-1-G. This will clarify the investment's profitability and align with common business practices for assessing (ROI).

ID	Name	Description	Measurement function	Method
REc-1-G	Return on investment (ROI)	The return on investment	$X = (A - B)/B$ A = Additional benefits obtained B = Invested amount	Business analytics
NOTE:	Examples of benefits obtained can include a reduction in personnel expenses, shrinkage of inventory assets, reduction of stock, or reduction of material cost through concentrated purchase.			
REc-2-G	Time to achieve a return on investment	The time taken to achieve the expected return on investment	$X = T$ T = Time to achieve ROI	Business analytics
NOTE: This can be compared with an acceptable time to achieve ROI.				

Figure 3.10: Economic risk mitigation measures Table 9: ISO/IEC 25022: 2016 (E)

3.4.3) Calculating business performance (REc-3-G) based on profitability or sales compared to a target can be very difficult in large organizations due to the scale of operations and diverse business units, as shown in Figure 3.11.

It will be challenging to pinpoint the impact of all users' collective contribution to the software quality-in-use assessment on both sales and profit as a ratio of actual to target sales. The more appropriate path to measure such business performance will be total profit or

net cost savings. For example, net cost savings due to a software deployment could be reductions in labor costs due to increased productivity, savings from automating manual processes, lower expenses associated with system maintenance, or avoided costs from errors or inefficiencies that the software helps to prevent.

ID	Name	Description	Measurement function	Method
REc - 3- G	Business per- formance	Profitability or sales compared. to a target	$X = Aa/At$ A = Profitability or sales of the company (a = actual, t = target)	Business analytics
NOTE This can be compared with the IT investment amount or sales of another company for comparison.				

Figure 3.11: Economic risk mitigation measures Table 9: ISO/IEC 25022: 2016 (E)

3.5. Implementation Guide to ISO/IEC 25022:2016 (E)

To streamline the implementation and utilization of the ISO/IEC 25022:2016 standard for assessing software quality-in-use, particularly in measuring effectiveness, efficiency, and satisfaction, the following Implementation guidelines are proposed. To accurately assess how well a product performs during use, it is vital to detail every aspect of its use. This means understanding the users, their aim, and the setting in which they interact with the product.

The evaluation should include representative user groups and tasks. Consider whether the tasks selected are dependent or independent.

If selected tasks are independent and cannot be partially completed, follow the steps shown in Figure 3.12. Under these conditions, different weights may be assigned to each task depending on its complexity.

The standard does not mention nor address task dependencies, which can impact task completion, effectiveness, and efficiency by influencing the sequence and timing of user interactions with the software.

In this proposed Implementation guide, it is assumed that all tasks are independent. However, if dependencies exist among tasks, the first step is to thoroughly analyze the evaluation requirements and break down tasks into manageable units. This analysis helps identify dependencies between tasks, such as those requiring the completion of predecessor tasks before they can start. Once dependencies are identified, tasks are sequenced appropriately to ensure that dependent tasks are scheduled and executed separately in the correct order. This approach ensures that task dependencies are managed effectively, enabling smooth assessment execution and successful measure outcomes.

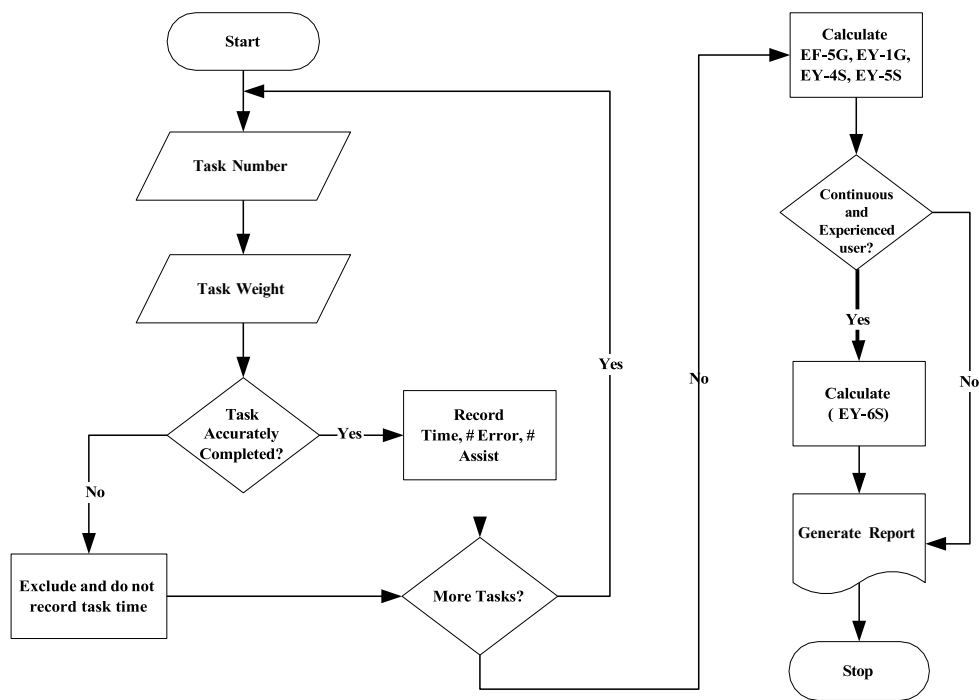


Figure 3.12: Task cannot be partially completed.

After successfully completing a task, the results should be checked for correctness. Only tasks completed correctly without assistance should be included in calculating the number of unique tasks completed for obtaining (Ef-1-G). Both the task completion time (Ey-1-G) and the number of errors (assisted or unassisted) encountered by the user during the task (Ef-3-G) need to be recorded. In instances where a user cannot correctly complete a task, the task should be excluded from the

assessment. Users will then proceed to work on more tasks until all are completed. The number of errors encountered should be recorded. The error resolution process requires considering both administrative times spent assisting the user and the time users spent searching the software help menu.

Upon the completion of all tasks and based on the collected values, calculations will be performed to determine task error intensity (Ef-5-G), task time (Ey-1-G), productive time ratio (Ey-4-S), and unnecessary actions (Ey-5-G). If an experienced user is included in the user group, consider calculating the consequences of fatigue by comparing the user's initial performance to their current performance over the evaluation period (Ey-6-S), where applicable. Subsequently, a report following the ISO/IEC 25061 recommendations can be prepared to capture the measured values.

Figure 3.13 presents the steps to follow when selected tasks can be partially completed. Here, each potential missing or incomplete objective is given a weight. Following the successful completion of an objective, the results should be checked for correctness (accuracy). Only objectives completed correctly without assistance should be included in calculating the proportion of objectives achieved (Ef-2-S) and time efficiency (Ey-2-S). For all objectives, the number of errors encountered should be recorded.

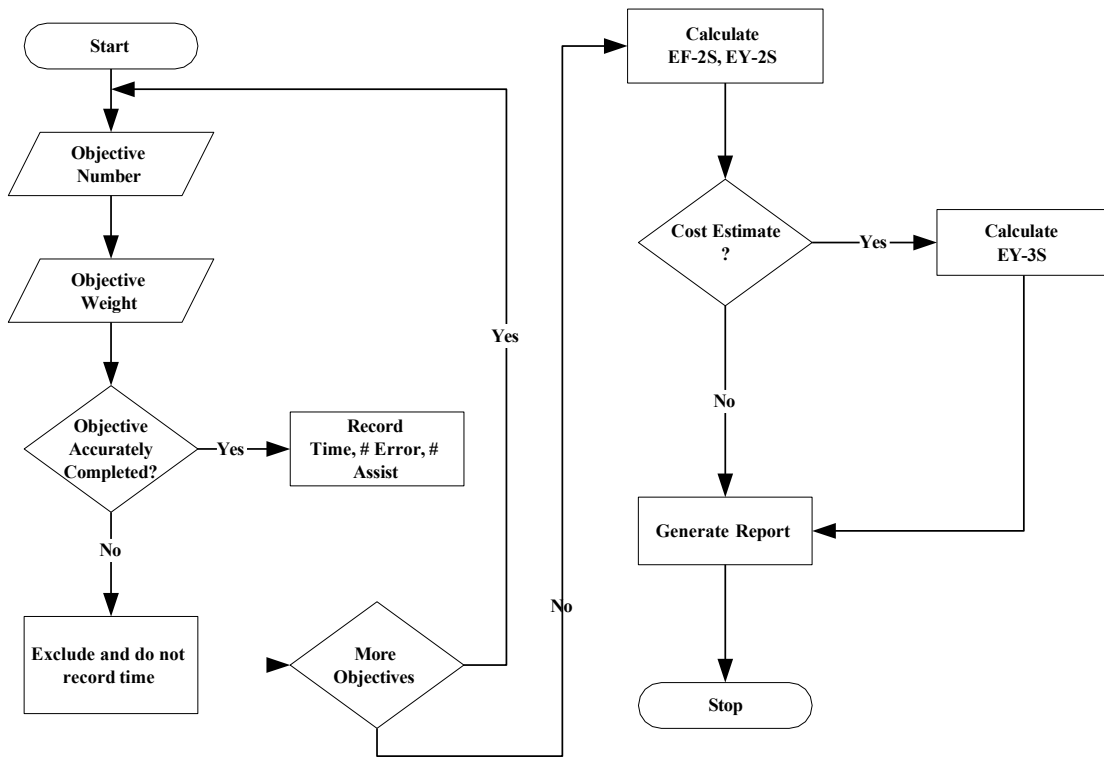


Figure 3.13: Task can be partially completed.

The error resolution process requires considering both administrative time spent providing help to the user and the time users spend searching the software help menu. In instances where a user cannot complete an objective, the objective will be excluded from the assessment, and no associated completion time and error count will be recorded; users will then proceed to work on more objectives until all are completed. The cost-effectiveness (EY-3-S) measure may be obtained if cost estimates are available. Subsequently, a report capturing measured values can be prepared.

CHAPTER 4
CONCLUSIONS AND FUTURE
RESEARCH

4.1 Conclusions

This chapter concludes the challenges identified in measuring user performance associated with quality-in-use according to the ISO/IEC 25022:2016 standard. While the standard offers a framework for evaluating the quality-in-use of software products, various challenges persist in its implementation. Chapter 3 proposes an implementation guide to streamline the implementation and utilization of the standard, mainly focusing on measuring effectiveness, efficiency, and satisfaction; the following is summarized as follows:

4.1.1) The proposed implementation guide aims to enhance understanding of the standard, identify relevant measures, and establish clear measurement goals for assessing software quality-in-use. One significant challenge addressed in the chapter is the ambiguity in language within the standard, particularly regarding effectiveness measures. The recommendation is to remove ambiguity and ensure alignment with intended meanings. This involves ongoing refinement and revision of the standard to reflect evolving industry best practices, ultimately enhancing the effectiveness and usability of software quality evaluation.

4.1.2) Incorporating a weighting scheme into the standard evaluation process to accommodate users' proficiency and familiarity with the system is a significant improvement. By factoring in users' skills, the evaluation becomes more comprehensive

and accurately reflects real-world usage scenarios. Additionally, the standard should specify the duration of continuous use required to calculate the consequences of fatigue.

4.1.3) The standard should offer clear instructions on defining and measuring return on investment (ROI) and business performance measures effectively, using methods that businesses commonly use. With clear and standardized definitions, the standard can make ROI assessments easier to understand and more consistent and provide a clearer understanding of the direct financial benefits derived from investing in software quality improvements.

4.1.4) Understanding the dependencies between tasks and objectives is essential for the effective planning and execution of software quality evaluation activities. It ensures that tasks are performed in the correct sequence and that any prerequisites are met before proceeding to subsequent tasks.

4.1.5) The implementation guide outlined in Chapter 3 will be useful for evaluating software quality in real-world usage scenarios, aiming to overcome the identified challenges. It is important to note that this guide may evolve, especially as the standard is due for an update to a newer version. Therefore, it is subject to change to ensure alignment with the latest developments and best practices in the field of software quality evaluation.

4.1.6) The proposed implementation guide and the identified challenges within the standard can be used as the foundation for developing a web-based application to allow for automated data collection and analysis and aid administrators and users in assessing software quality-in-use by the standard focusing on evaluation of effectiveness, efficiency, and satisfaction measures.

By addressing considerations and ambiguity, tasks, and objectives dependency, incorporating user proficiency considerations, and providing clear instructions for ROI assessment to be addressed by the Quality Measurement Division 2502n and considered in the upcoming revisions, the standard becomes more robust and applicable to real-world scenarios.

4.2 Future Research

This work will continue to build upon enhancing software quality assessment practices by extending the scope of the investigation to encompass forthcoming revisions of the standard. This will involve a thorough evaluation of new and altered metrics about software quality in use. By undertaking this endeavor to contribute to the ongoing evolution and enhancement of methodologies for ensuring software quality.

Future research may target incorporating Artificial Intelligence (AI) in testing. User feedback analysis can automate the testing process by generating test cases, executing tests, analyzing results, and analyzing user feedback from various sources, such as reviews, social media, and support tickets, to assess user satisfaction and identify areas for improvement. AI algorithms can optimize software quality metrics by identifying the most relevant indicators for a particular project or organization. Through iterative learning and experimentation, AI systems can continuously refine quality assessment models to better align with stakeholders' priorities and goals.

REFERENCES

REFERENCES

- Al-Kilidar, H., Cox, K., & Kitchenham, B. (2005, November). The use and usefulness of the ISO/IEC 9126 quality standard. In 2005 International Symposium on Empirical Software Engineering, 2005. (pp. 7-pp). IEEE.DOI: 10.1109/ISESE.2005.1541821.
- Al-Qutaish, R.E, (2009). An investigation of the weaknesses of the ISO 9126 international standard. Second International Conference on Computer and Electrical Engineering. DOI 10.1109/ice.2009.83.
- Al-Qutaish, R. E. (2010). Quality models in software engineering literature: an analytical and comparative study. *Journal of American Science*, 6(3), 166-175.
- Alrawashdeh, T. A., Muhairat, M., & Althunibat, A. (2013). Evaluating the quality of software in ERP systems using the ISO 9126 model. *International Journal of Ambient Systems and Applications (IJASA)*, 1(1), 1-9.
- Boehm, B. W., Brown, H., Lipow, M. (1978) "Quantitative Evaluation of Software Quality," TRW Systems and Energy Group, 1978.
- Dromey, R. G. (1995). A model for software product quality. *IEEE Transactions on Software Engineering*, 21,146-162.
- Burgos Robles, L. J., & Huanca Torres, F. A. (2023, April). Quality in Use Evaluation Model Based on ISO/IEC 25022 for Human Talent Management Information Systems. In *Computer Science On-line Conference* (pp. 817-830). Cham: Springer International Publishing.
- Gong, J., Lu, J., & Cai, L. (2016, September). An induction to the development of software quality model standards. In 2016 third International Conference on trustworthy systems and their applications (TSA) (pp. 117-122). IEEE.
- Grady, R. B., & Caswell, D. (1987). *Software metrics: Establishing a company-wide program*. Englewood Cliffs: Prentice-Hall.
- Haboush, A., Alnabhan, M., Al-Badareen, A., Al-Nawayseh, M., & El-Zaghmouri, B. (2014). Investigating software maintainability development: A case for ISO 9126. *International Journal of Computer Science Issues (IJCSI)*, 11(2), 18.
- Hashim, N. L., Yusof, N., Hussain, A., & Ibrahim, M. (2022). User Experience Dimensions for E-procurement: A Systematic Review. *Journal of Information and Communication Technology*, 21(4), 465-494.

- IEEE. (1990). IEEE Std 610.12-1990 (1990)- “IEEE Standard Glossary of Software Engineering Terminology,”
- ISO/IEC IS 9126: Software Product Evaluation - Quality Characteristics and Guidelines for Their Use, International Organization for Standardization, Geneva, Switzerland, 1991.
- ISO/IEC IS 9126-1: Software Engineering - Product Quality- Part 1: Quality Model, International Organization for Standardization, Switzerland, 2001.
- ISO/IEC TR 9126-2. (2003). Software Engineering - Product Quality - Part 2: External Metrics. International Organization for Standardization, Geneva, Switzerland.
- ISO/IEC TR 9126-3. (2003): Software Engineering - Product Quality - Part 3: Internal Metrics, International Organization for Standardization, Geneva, Switzerland.
- ISO/IEC TR 9126-4. (2004): Software Engineering - Product Quality - Part 4: Quality-in-use Metrics. International Organization for Standardization, Geneva, Switzerland.
- ISO/IEC 25010:2011 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models.
- ISO/IEC 25000:2014 Systems and software engineering Systems and software Quality Requirements and Evaluation (SQuaRE).
- ISO/IEC 25022:2016 Systems and software engineering — Systems and software quality requirements and evaluation (SQuaRE) — Measurement of quality-in-use.
- ISO/IEC 25023:2016 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Measurement of system and software product quality.
- Jaradat, B., Weheba, G., & Kanan, M. (2015). Cost of Software Quality: A Literature Review. *Journal of Management & Engineering Integration*, 8(1).
- Juran, J.M., (1988)." Juran's Quality Control Handbook,"4th Ed, New York; McGraw Hill Book Co.
- McCall, J. A., Richards, P. K., Walters, G. F. (1977). Factors in Software Quality, Volumes I, II, and III. *US Rome Air Development Center Reports*, US Department of Commerce, USA.
- Miguel, J. P., Mauricio, D., & Rodríguez, G. (2014). A review of software quality models for the evaluation of software products. arXiv preprint arXiv:1412.2977.
- Pradanita, W. R., Ni'Mah, A. T., Rochimah, S., & Adiputra, F. (2019, July). Assessment of Academic Information System Quality from Two Perspectives: Product Quality and Quality in Use. In 2019 12th International Conference on Information & Communication Technology and System (ICTS) (pp. 1-6). IEEE . DOI: 10.1109/ICTS.2019.8850933

- Pradanita, W. R., & Rochimah, S. (2020, October). Quality-in-use of Digital Wallet based on ISO/IEC 25022. In 2020 7th International Conference on Electrical Engineering, Computer Sciences, and Informatics (EECSI) (pp. 282-286). IEEE. DOI: 10.23919/EECSI50503.2020.9251300.
- Rochimah, S., Rahmani, H. I., & Yuhana, U. L. (2015, May). Usability characteristic evaluation on administration module of Academic Information System using ISO/IEC 9126 quality model. In 2015 International Seminar on Intelligent Technology and Its Applications (ISITIA) (pp. 363-368). IEEE.
- Santos, C., Novais, T., Ferreira, M., Albuquerque, C., de Farias, I. H., & Furtado, A. P. C. (2016, June). Metrics focused on usability ISO 9126 based. In 2016 11th Iberian Conference on Information Systems and Technologies (CISTI) (pp. 1-3). IEEE. DOI: 10.1109/CISTI.2016.7521437.
- Zhao, Y., Gong, J., Hu, Y., Liu, Z., & Cai, L. (2017, May). "Analysis of quality evaluation based on ISO/IEC SQuaRE series standards and its considerations," 2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS), Wuhan, China, 2017, pp. 245-250, doi: 10.1109/ICIS.2017.7960001.