LOGISTICS SYSTEMS DIGITALISATION AND SOFTWARE QUALITY: WHY IT'S IMPORTANT AND HOW IT'S RELATED TO ISO/IEC 25010

Borut Jereb
University of Maribor, Slovenia
E-mail: borut.jereb@um.si

Milena Kajba
University of Maribor, Slovenia
E-mail: milena.kajba1@um.si

Received: August 11, 2023
Received revised: September 25, 2023
Accepted for publishing: September 26, 2023

Abstract

Software quality is critical in supply chains, ensuring that software products meet user expectations and deliver value by digitalising their business. Organisations rely on established standards and frameworks to achieve and assess software quality effectively. The ISO/IEC 25000 family of standards, also known as the SQuaRE (Software Quality Requirements and Evaluation) series, provides a comprehensive and internationally recognised set of guidelines for software quality management. The ISO/IEC 25000 series consists of standards and technical reports covering various aspects of software quality characteristics, including quality models, evaluation processes, and measurement methods. ISO/IEC 25010 serves as the core standard in the series, defining a comprehensive quality model and a set of quality characteristics. Quality characteristics encompass functionality, reliability, usability, efficiency, maintainability, and portability. By following these characteristics, organisations can evaluate, measure, and improve the quality of software products. It also provides a systematic approach enabling organisations to set quality requirements, define evaluation criteria, and select appropriate evaluation techniques. These standards promote consistency and comparability in software quality evaluation, facilitating effective communication among stakeholders, improve decision-making processes, enable effective risk management, enhance customer satisfaction, and promote continuous improvement. By aligning with these standards, supply chain organisations can achieve greater transparency, reliability, and interoperability in their digitalised business processes. This paper explores the significance of software quality and its relationship to the ISO/IEC 25010 standard in a supply chain, a complex logistics system consisting of logistics processes and facilities that transform raw materials into finished products, which are later distributed to end consumers. This paper delves into the details of the ISO/IEC 25010, which provides key guidelines for ensuring software quality. Practical examples of standards usage will be presented by

Software Quality Measurement (SQM) model and evaluation of vehicle telematics to better understand how standards can enhance the quality of technological solutions.

Keywords: supply chain, logistics processes, digitalisation, software, software quality

1. INTRODUCTION

Understanding "quality" is subjective since different definitions of quality are appropriate for different people, areas of study or work, and circumstances. If we search for the word in dictionaries, many different meanings will arise, such as: "how good or bad something is" (Cambridge Dictionary, 2023); "a degree of excellence" (Merriam-Webster dictionary, 2023); an essential or distinctive characteristic, property, or attribute (Dictionary.com, 2023); and "performance upon expectations" or even "fit for functions " (TQP, 2020).

When speaking of quality in companies, the word can be used to describe different things, like quality of work, production, service, product, software, people, process, system, management, and many others. Regarding the latest, the ISO organisation presents quality management principles (QMPs) in ISO 9000 standards. The principles are meant to be a foundation that guides organisations to better performance (ISO, 2015). Their importance varies from company to company and can be used as needed. The quality management principles are Customer focus, Leadership, Engagement of people, Process approach, Improvement, Evidence-based decision-making, and Relationship management (Nahil, 2020).

The digitalisation of supply chains is essential since the utilisation of digital supply chains provides insights for enhanced efficiency to assist and create an increased economy (Menon & Shah, 2019). Supply chain technologies validate the performance of companies with better plans, designs, and management of the flow of goods, products and services (Nasir et al., 2017). The digitalisation of logistics and supply chains represents an essential tool for resilience (Gupta et al., 2022), where it plays a pivotal role in enabling sustainable and environmentally friendly supply chains (Chauhan et al., 2023). Based on digitalisation capabilities for supply chains, companies can harness ICT resources to transition their physical operations into digital domains, seamlessly integrating them to optimise resource efficiency and foster productivity enhancements. This comprehensive approach encompasses physical and digital activities, facilitating reduced resource consumption, heightened network visibility, and real-time feedback mechanisms. Additionally, it covers a variety of specialised tools for tailored production and cooperative relationships with suppliers throughout all network stages. (Queiroz et al., 2019)

Thus, software quality plays a pivotal role in determining the success of any project. Inadequately constructed code can give rise to a litany of problems, including bugs, system crashes, security vulnerabilities, software malfunctions, and performance bottlenecks, all of which can lead to considerable challenges in the future. Moreover, suboptimal software quality can precipitate elevated expenses, extended development timelines, and an escalated likelihood of project failure.

(Quality Gurus, 2021) This is where software quality standards and frameworks come into play. Standards and frameworks encompass a compendium of best practices, guidelines, and principles, all geared towards guaranteeing the dependability, usability, maintainability, and security of software products and processes. Adherence to these standards and frameworks becomes imperative, facilitating the delivery of top-tier software that aligns with the expectations and demands of the company, stakeholders, customers and end-users. (Software Project Management, n.d.)

In contemporary supply chain management, information and communication technologies (ICT) are pivotal (Buxmann et al., 2004), where software can immensely improve operations performance. Enhanced coordination is frequently achieved by implementing a unified software platform capable of aggregating, processing, and transmitting information from diverse ICT systems spanning the supply chain. A diverse offer of supply chain management software (SCMS) is available through software packages that offer a spectrum of services encompassing material requirements planning (MRP), warehouse management system (WMS), enterprise resource planning (ERP), and workforce management (WfMs) solutions. The scope of these services extends from facilitating operations related to logistics, inventory management, planning, forecasting, sales, cash flow, and acquisition, among others. (Haulder et al., 2019) Software solutions also enable shipment monitoring at every stage of the supply chain journey (Mohsen, 2023) and data-driven decision-making (Di Vaio & Varriale, 2020). Supply chain management methods are made feasible through ERP software, which monitors and records business processes, providing various departments with productivity insights about each transaction conducted within the company (Aroba & Prinavin, 2023). Thus, software quality plays a critical role in supply chains, ensuring that software products meet user expectations (Kitchenham & Pfleeger, 1996) and deliver value by digitalising their business (Krasner, 2021).

This paper explores the significance of software quality and its relationship to the ISO/IEC 25010 standard in a supply chain, a complex logistics system consisting of logistics processes and facilities that transform raw materials into finished products, which are later distributed to end consumers.

2. FRAMEWORKS FOR SOFTWARE QUALITY

Software quality is essential for the success of any software project. Developers can deliver valuable, dependable software products that meet user expectations and business requirements. Software quality is measured by (ISO, 2011):

1. **Testing**: helps identify defects and ensures the software functions as expected (unit testing, integration testing, and user acceptance testing).
2. **Code Reviews**: helps to identify code issues and improve overall code quality.
3. **User Feedback**: allows developers to address pain points and make necessary improvements.
4. **Quality Metrics**: help track and measure software quality over time (defect density, response time, uptime).

These measures ensure ongoing improvement and maintain high software quality standards and can bring us the following (ISO, 2011):

1. **Customer Satisfaction**: high-quality software leads to satisfied customers, positive reviews, and increased user retention.
2. **Cost-Effectiveness**: fixing defects early in the development process is less expensive than addressing them later or after deployment.
3. **Reputation and Trust**: high-quality software builds trust among users and establishes a good reputation for the organisation.
4. **Competitive Advantage**: quality software differentiates a company's products from competitors and attracts more users.
5. **Reduced Risks**: quality software is less prone to crashes, security breaches, and data loss, reducing risks to the organisation and its users.

However, several software industry frameworks and methodologies are used to ensure and improve software quality. These frameworks provide guidelines and best practices for managing software quality throughout the development and maintenance processes. Here are some prominent frameworks and standards for software quality management:

1. **ISO/IEC 25000 (SQuaRE - Software Product Quality Requirements and Evaluation)**: a comprehensive standard defining a series of international software product quality evaluation standards. It includes models and metrics to assess various aspects of software quality (functionality, reliability, usability, efficiency, maintainability, and portability).
2. **ISO/IEC 9126 (replaced by ISO 25010)**: defined software quality characteristics and sub-characteristics, providing a structured approach for evaluating software quality attributes.
3. **ISO/IEC 25010 (System and Software Quality Models): the successor to ISO/IEC 9126 defines the quality model, a set of quality characteristics and sub-characteristics to assess software quality.**
4. **ISO 9000 series:** family of standards focuses on quality management systems and provides general guidelines for quality management systems. **While not exclusively for software, ISO 9001:2015** sets the requirements for a quality management system that an organisation can use to enhance customer satisfaction and consistently provide products and services that meet regulatory requirements. It also **includes specific requirements for quality management that can be applied to software development.**
5. **Capability Maturity Model Integration (CMMI)**: is a process improvement framework that helps organisations optimise their processes and improve software development quality. It provides a set of best practices for developing, maintaining, and acquiring products and services.
6. **Six Sigma**: is a data-driven methodology aimed at improving the quality of process outputs by identifying and removing the causes of defects and minimising variability in business and manufacturing processes.
7. **Agile Development**: agile methodologies, such as Scrum and Kanban, prioritise delivering working software in short iterations. Frequent feedback, collaboration, and continuous improvement are integral to Agile approaches, ensuring that software meets user needs effectively.

8. **Test-Driven Development (TDD)**: is a software development approach where developers write automated tests before writing the actual code. It helps ensure that the software meets requirements and that tests cover any changes to the code.
9. **Behaviour-Driven Development (BDD)**: is an extension of TDD that focuses on collaboration between developers, testers, and business stakeholders to define and verify the behaviour of the software in natural language.
10. **Continuous Integration and Continuous Delivery (CI/CD)**: automate integrating code changes, running tests, and delivering software to production. This ensures frequent feedback, early detection of defects, and faster delivery of high-quality software.
11. **Risk-Based Testing**: prioritizes testing efforts based on identified risks to the software, ensuring that critical areas are thoroughly tested, and any potential issues are addressed early.
12. **Open Web Application Security Project** (OWASP): provides a list of the most critical security risks of web applications. Developers can use this list to focus on securing their software against common vulnerabilities.
13. **IEEE 730 Standard**: defines software quality assurance processes, including planning, implementation, and evaluation.
14. **IEEE 829**: outlines the software test documentation format, helping maintain consistent and comprehensive testing practices.
15. **Information Technology Infrastructure Library (ITIL)**: is a set of best practices for IT service management that includes practices related to software quality management within the broader context of IT service delivery.
16. **Project Management Body of Knowledge (PMBOK)**: includes guidelines for project management, which indirectly influences software quality management by addressing project planning, risk management, and quality assurance processes.
17. **ASTM E2500**: focuses on risk-based and science-based verification and validation of software, providing guidelines for software development and testing.

When selecting a framework or standard for software quality management, organisations should consider their specific needs, the complexity of their projects, industry requirements, and the maturity of their development processes. Integration of multiple frameworks might be necessary to cover different aspects of software quality and align with organisational goals and industry standards.

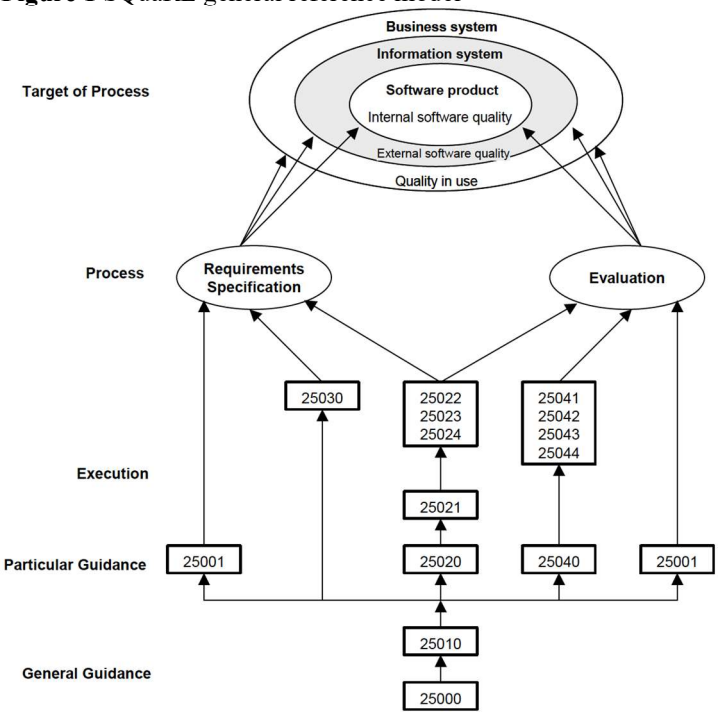
ISO standards are internationally agreed upon by experts and provide a common set of guidelines and best practices that are recognised and accepted globally. They are the distilled wisdom of people with expertise in their subject matter who know the needs of the organisations they represent. This helps ensure consistency and uniformity in various industries and sectors across different countries. In the remaining part of this paper, we will describe the ISO/IEC 25000 family of standards that provide a comprehensive and internationally recognised set of guidelines for software quality management. (ISO, 2014)

3. ISO/IEC 25000 AND 25010

ISO/IEC 25000, also known as Software Product Quality Requirements and Evaluation (SQuaRE), is an international standard that provides a comprehensive framework for evaluating and managing software product quality. SQuaRE was developed by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) to address the need for a structured and consistent approach to software quality assessment to help organisations understand, define, and evaluate the quality of their software products, as well as make informed decisions about improvements and optimisations (ISO, 2011).

This standard comprises a collection of documents that establish quality models, delineate quality attributes, and outline quality requirements for software products. These models and attributes serve as valuable tools for evaluating and gauging various facets of software quality. They find utility among software development organizations, clients, and other concerned parties for the purposes of evaluating, communicating, and enhancing the quality of software products. By adhering to the guidelines and principles set forth in this standard, organizations can elevate the overall quality of their software products, ultimately resulting in heightened customer satisfaction and successful software deployments. The foundational reference model of SQuaRE can be found in Figure 1.

Figure 1 SQuaRE general reference model



Source: ISO (2014)

ISO/IEC 25010 provides a detailed quality model with specific quality characteristics and sub-characteristics for evaluating software product quality. SO/IEC 25000 and ISO/IEC 25010 are related standards but serve different purposes within the broader SQuaRE series. (ISO, 2011) The key differences between ISO/IEC 25000 and ISO/IEC 25010 are presented in Table 1.

Table 1 Key differences between ISO/IEC 25000 and ISO/IEC 25010

	ISO/IEC 25000	**ISO/IEC 25010**
Scope and Purpose	- the overarching standard in the SQuaRE series; - provides a framework for software product quality requirements and evaluation; - defines the general concepts, terms, and principles related to software quality management; - introduces the quality models and metrics used to assess software quality and guides users to other specific standards in the series, including ISO/IEC 25010.	- a specific standard within the SQuaRE series that defines a comprehensive quality model; - outlines the quality characteristics and sub-characteristics that can be used to evaluate and measure software product quality; - dives deeper into the various dimensions of software quality, providing specific criteria for assessing software products.
Content	- an informative standard that introduces the software product quality framework, the overall structure of quality characteristics and sub-characteristics; - it acts as a guiding document for understanding software quality evaluation in the context of the SQuaRE series.	- a normative standard containing specific requirements and is intended to be used directly for software quality evaluation; - defines the eight primary quality characteristics and their sub-characteristics (accuracy, compliance, efficiency).
Usage	- primarily used to provide an overview of software quality management and direct users to other relevant standards in the SQuaRE series; - sets the context and terminology for the quality models and metrics for software quality assessment.	- used directly for evaluating software product quality; - serves as a reference for practitioners who want to assess and measure the quality attributes of a software product; - guides the selection of relevant quality characteristics and sub-characteristics to be evaluated based on the specific needs and goals of the evaluation.

Source: ISO (2011); ISO (2014)

In summary, ISO/IEC 25000 is an informative standard that introduces broader concepts and structures for software quality evaluation. At the same time, ISO/IEC 25010 is a normative standard that explicitly defines the quality model and criteria for assessing software product quality.

3.1. ISO/IEC 25010 System and software quality models

ISO/IEC 25010 Systems and Software Quality Models consists of (ISO, 2011):

1. **Quality Model:** covers eight main quality characteristics, where each represents a critical aspect of software quality:
   - Functional Suitability: the extent to which software provides the necessary functions to meet specified needs;
   - Performance Efficiency: the software's ability to perform well concerning resource usage, response time, and throughput.
   - Compatibility: the software's capability to operate with other systems, software, or hardware components.
   - Usability: the ease of software use and the user experience.
   - Reliability: the software's ability to maintain its level of performance under specific conditions for a given period.
   - Security: the software's ability to protect data and functionalities from unauthorised access and harm.
   - Maintainability: the effort required to make modifications, correct defects, or adapt the software to changes.
   - Portability: the ease with which the software can be transferred from one environment to another.
2. **Sub-characteristics:** each quality characteristic is broken down into specific sub-characteristics, making it easier to assess and focus on specific areas of quality evaluation.
3. **Quality Requirements:** includes a set of quality requirements that can specify the desired level of each quality characteristic and sub-characteristic for a particular software product.
4. **Quality in Use:** emphasizes the importance of evaluating how end-users perceive the software quality during actual usage.

Table 2 represents quality characteristics and their subcharachetiristics.

Table 2 ISO/IEC 25010 set of quality characteristics with their subcharacteristics

(Sub)Characteristics			
Functional suitability		**Reliability**	
	- Functional completeness; - Functional correctness; - Functional appropriateness.		- Maturity; - Availability; - Fault tolerance; - Recovery.
Performance efficiency		**Security**	
	- Time behaviour; - Resource utilization; - Capacity.		- Confidentiality; - Integrity; - Non-repudiation; - Accountability; - Authenticity.
Compatibility		**Maintainability**	
	- Co-existence; - Interoperability.		- Modularity; - Reusability; - Analysability; - Modifiability; - Testability.
Usability		**Portability**	
	- Appropriateness recognizability; - Learnability; - Operability; - User error protection; - User interface aesthetics; - Accessibility.		- Adaptability; - Installability; - Replaceability.

Source: ISO (2011)

3.2. Software Quality measurement model (SQM)

A concise explanation of how software quality is measured and quantified by can be provided by (ISO, 2011):

1. **Quality Properties**: These are aspects or attributes of software that determine its overall quality. Examples include performance, reliability, and security.
2. **Measurement Method**: This is a systematic set of steps used to assess and quantify these quality properties. It's like a structured process for evaluating specific aspects of the software.
3. **Quality Measure Element**: This is the result obtained when a measurement method is applied. It represents a specific quantified value related to a quality property.

4. **Measurement Functions**: These are algorithms or formulas used to combine multiple quality measure elements. They help in aggregating or summarizing the individual measurements into a single value.
5. **Software Quality Measure**: This is the outcome of applying a measurement function. It represents a comprehensive measurement of a particular quality characteristic or subcharacteristic of the software.

Table 3 represents Quality in use characteristics and subcharacteristics.

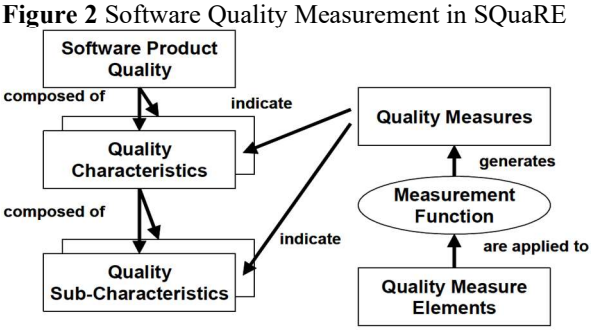
Table 3 Quality in use

Effectiveness	
Efficiency	
Satisfaction	
	- Usefulness; - Trust; - Pleasure; - Comfort.
Freedom from risk	
	- Economic risk mitigation; - Health ans safety risk mitigation; - Envirnmental rist mitigation.
Context coverage	
	- Context completeness; - Flexibility.

Source: ISO (2011)

In essence, this process allows to express the quality of software in a quantifiable manner by breaking it down into measurable components and then aggregating those measurements using appropriate algorithms. This approach provides a structured way to assess and improve the quality of software products. (ISO, 2011) Figure 2 represents Software Quality Measurement (SQM) in SQuaRE.

Figure 2 Software Quality Measurement in SQuaRE



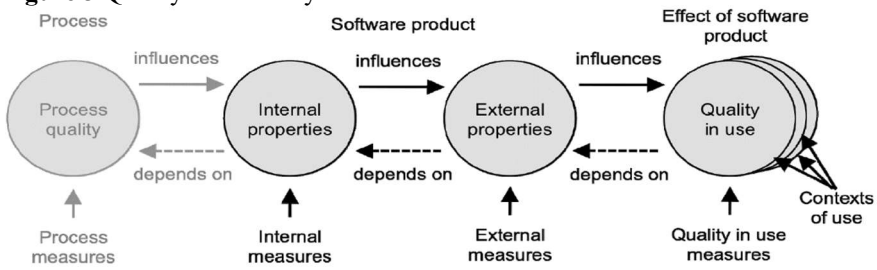
Source: ISO (2014)

User requirements for quality encompass the criteria for system performance in specific usage scenarios (ISO, 2011), as illustrated in Figure 1. These identified requirements serve as valuable inputs when defining both external and internal quality metrics using software product quality characteristics and subcharacteristics.

3.2.1. Approaches to quality

The quality of a software product can be assessed through various approaches. This evaluation can involve measuring internal characteristics, which usually consist of static measurements of intermediate development stages. Another evaluation measure external attributes by evaluating the behaviour of the code during execution. Another approach is to assess quality in use, which involves evaluating the product's performance in real or simulated usage scenarios. See Figure 1 and Figure 3.

Figure 3 Quality in the lifecycle



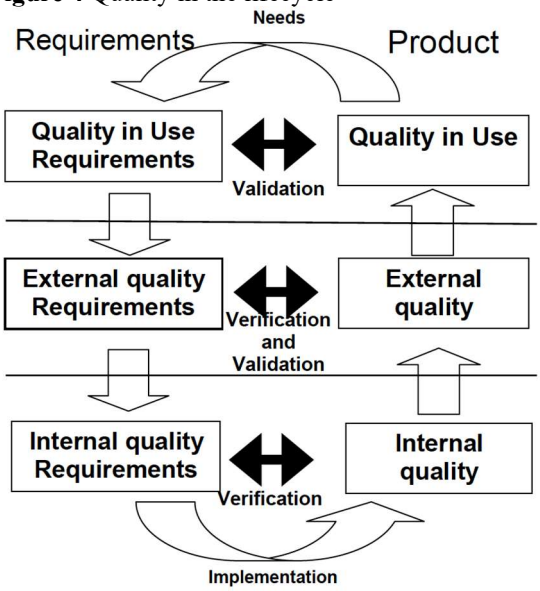
Source: ISO (2014)

3.2.2. Quality life cycle model

The quality life cycle model, as depicted in Figure 4, focuses on ensuring software quality in three key phases of the software product's life cycle (ISO, 2011):

1. During the product under **development phase**, internal measures are used to assess software quality.
2. In the **product testing phase**, external measures are employed to evaluate software quality.
3. In the **product in use phase**, the focus shifts to quality in use assessments.

Figure 4 Quality in the lifecycle



Source: ISO (2014)

Quality in use requirements define the expected quality levels from the user's perspective. These requirements are derived from the needs of various stakeholders, including users, software developers, system integrators, acquirers, or owners. Quality in use requirements serve as the benchmark for user validation of the software product. These requirements for quality in use characteristics should be clearly documented in the quality requirements specification, outlining the criteria for quality measurements used during product evaluation. (ISO, 2011)

Requirements for **external measures of computer system quality** establish the expected quality levels as observed from an external perspective. These requirements encompass criteria derived from stakeholder quality demands, which may include quality-in-use requirements. External software quality requirements serve as the focal point for the technical verification and validation of the software product. To ensure clarity and precision, requirements for external quality measures should be quantitatively defined in the quality requirements specification, outlining the criteria for external assessments used during product evaluation. (ISO, 2011)

Requirements for i**nternal software quality measures** define the desired level of quality from an internal perspective of the product. These requirements encompass criteria derived from external quality requirements. Internal software quality requirements play a crucial role in specifying the characteristics of intermediate software products, such as specifications and source code. They may also extend to non-executable software products like documentation and manuals. These internal software quality requirements can serve as targets for verification at different stages of development. Furthermore, they are valuable in shaping development strategies and

establishing criteria for evaluation and verification throughout the development process. (ISO, 2011)

4. EXAMPLES OF USE

4.1. Software Quality measurement model (SQM)

Imagine you are a software quality assurance engineer working for a company that develops web applications. Your team has recently developed a new web application, and you want to ensure that it meets the required quality standards before it is released to the public. You use ISO/IEC 25010 to evaluate the web application's quality to achieve this.

Step 1: Identify Quality Characteristics: the first step is to identify the relevant quality characteristics for the web application. Based on ISO/IEC 25010, you select the following characteristics as most important for your evaluation:

1. **Functionality**: to assess whether the application meets the functional requirements and provides the necessary features.
2. **Usability**: to evaluate how easy and efficient the application is for its intended users.
3. **Performance Efficiency**: to measure the application's responsiveness, resource usage, and throughput.
4. **Security**: to assess the level of protection against unauthorised access, data breaches, and other security threats.

Step 2: Identify Subcharacteristics: for each of the selected characteristics, you identify the relevant subcharacteristics to dive deeper into the evaluation process. For example:

1. **Usability**:
   - **Learnability**: how easy it is for new users to understand and navigate the application.
   - **Operability**: how easy it is for users to operate and control the application.
   - **User Error Protection**: how well the application prevents and handles user errors.
2. **Performance Efficiency**:
   - **Time Behavior:** how fast the application responds to user actions.
   - **Resource Utilization:** How efficiently the application uses system resources like memory and CPU.

Step 3: Define Evaluation Criteria: for each subcharacteristic, you define specific evaluation criteria. For instance:

1. **Usability – Learnability: evaluation criteria**: the application should have clear and concise onboarding instructions, intuitive navigation, and interactive tutorials to help new users get started easily.
2. **Performance Efficiency - Time Behavior: evaluation criteria**: the application should load its main page within 2 seconds for a standard internet connection.

Step 4: Evaluation Process: you evaluate by using various methods such as user testing, performance testing, security testing, and expert reviews. Each evaluation method corresponds to the relevant quality characteristic or subcharacteristic.

Step 5: Analyze Results and Improve: once the evaluation is complete, you analyse the results to identify areas of improvement. For example, if the application's loading time is slower than expected, you may work on optimising its performance. If security vulnerabilities are identified, you will take appropriate measures to address them.

4.2 Evaluating vehicle telematics (includes tachograph data) m-application

In this scenario, let us consider a case where you are a software quality assurance analyst working for vehicle telematics for an easy and efficient fleet management telematics, including tachograph data that develops mobile applications drivers. Your company has recently developed a new fleet m-telematics app designed to help drivers and their companies track their daily activities, monitor their drive metrics, and provide personalised driver recommendations. To ensure the app meets the highest quality standards, use ISO/IEC 25010 for evaluation.

Step 1: Identify Quality Characteristics: Based on the nature of the m-telematics application and its intended use, you identify the following quality characteristics to evaluate:

1. **Functionality**: to assess whether the app includes all the necessary features for tracking physical activity, recording health metrics, and offering personalised health insights.
2. **Reliability**: to measure the app's ability to perform consistently and reliably under various conditions, ensuring it accurately tracks and stores user data.
3. **Usability**: evaluate the application is user-friendly, considering the target audience, which might include individuals of different age groups and technical expertise.
4. **Privacy and Security**: assess the level of protection for sensitive health data and ensure compliance with privacy regulations.

Step 2: Identify Subcharacteristics: for each quality characteristic, you identify the relevant subcharacteristics that are critical for the evaluation:

1. **Functionality**:
   - Completeness: how well the app covers all the necessary functions, such as step tracking, heart rate monitoring, sleep analysis, etc.
2. **Reliability:**
   - Availability**:** the app should be accessible and functional whenever users need it.
   - Data Integrity**:** ensuring the user's health data is accurately recorded and stored without any loss or corruption.
3. **Usability:**
   - Accessibility**:** the app should be usable by individuals with disabilities or special needs.

- User Interface (UI) Design**:** the app should have an intuitive and visually appealing UI.

4. **Privacy and Security:**
   - Confidentiality**:** ensuring that user health data is kept private and only accessible by authorised personnel.
   - Data Encryption**:** the app should use encryption techniques to protect data during transmission and storage.

Step 3: Define Evaluation Criteria: for each subcharacteristic, you define specific evaluation criteria. For example:

1. **Usability - Accessibility: Evaluation Criteria**: the app should adhere to accessibility guidelines, allowing users with visual impairments to use assistive technologies like screen readers.
1. **Privacy and Security - Data Encryption: Evaluation Criteria**: all health data transmitted between the app and the server should be encrypted using SSL/TLS protocols.

Step 4: Evaluation Process: evaluate by combining various methods, including functional testing, usability testing with representative users, security assessments, and reliability testing. Each evaluation method aligns with the relevant quality characteristic or subcharacteristic.

Step 5: Analyze Results and Improve: after the evaluation, you analyse the results to identify areas that require improvement. For instance, if the app's UI is not intuitive for certain users during usability testing, you might redesign some elements to enhance the user experience. If any security vulnerabilities are found, they are promptly addressed to safeguard user data.

5. CONCLUSION

The software quality has revealed its far-reaching implications, from customer satisfaction and cost-effectiveness to reputation building and risk reduction. Utilizing software quality standards and frameworks is imperative in delivering high-quality software products aligned with the expectations of various stakeholders. Moreover, within the dynamic supply chain management landscape, software solutions are indispensable for achieving enhanced coordination, monitoring, and data-driven decision-making. ISO standards are critically important in supply chains and logistics due to their capacity to establish a universal framework of quality, safety, and efficiency criteria. Organisations turn to ISO standards because they provide a structured and internationally recognised basis for evaluating and benchmarking processes, products, and services, ultimately enhancing interoperability, reducing operational risks, and ensuring compliance with regulatory requirements. In the context of supply chains and logistics, adherence to ISO standards fosters consistent and reliable practices, bolsters organisational credibility, facilitates global trade, and contributes to optimising resource utilisation, leading to heightened competitiveness and improved overall performance.

This paper described the vital relationship between software quality, logistics, and ISO/IEC 25000 standards. We emphasised the critical role of software quality in modern supply chains, particularly in facilitating the seamless digitalisation of logistics through ICT. The ISO/IEC 25010 standard, part of the SQuaRE series, defines a comprehensive quality model that serves as a basis for evaluating the quality of software products. This model is organised into two main categories: system and software product quality. Each category has several quality characteristics and subcharacteristics contributing to the overall quality assessment. It has been introduced as a robust framework for comprehensively assessing and managing software quality, covering essential characteristics such as functionality, reliability, usability, efficiency, maintainability, and portability.

Looking ahead, it becomes evident that the collaboration between software quality and logistics systems will continue to evolve and deepen. The application of ISO/IEC 25010 standard in supply chains will likely become more pervasive, fostering greater transparency, reliability, and interoperability. The integration of emerging technologies, such as artificial intelligence, blockchain and digital twins (Kajba et al., 2023), may further enhance the digitalisation of logistics, necessitating even more robust software quality practices. In today's ever-evolving world, organisations prioritising software quality and embracing ISO/IEC 25000 standards are poised to thrive, offering enhanced efficiency and value to their customers and stakeholders in the complex world of supply chain management.

6. REFERENCES

Aroba, O.J. & Prinavin, G. 2023. Improving supply chain management processes through digital transformation of ERP systems in the oil and gas industry. Interdisciplinary Journal of Economics and Business Law, 12(3): 20.

Buxmann, P., Ahsen, A. von, Díaz, L.M. & Wolf, K. 2004. Usage and evaluation of Supply Chain Management Software – results of an empirical study in the European automotive industry. Information Systems Journal, 14(3): 295–309.

Cambridge Dictionary. 2023. Quality. Cambridge Dictionary. https://dictionary.cambridge.org/dictionary/english/quality 8 September 2023.

Chauhan, S., Singh, R., Gehlot, A., Akram, S.V., Twala, B. & Priyadarshi, N. 2023. Digitalization of Supply Chain Management with Industry 4.0 Enabling Technologies: A Sustainable Perspective. Processes, 11(1): 96.

Di Vaio, A. & Varriale, L. 2020. Blockchain technology in supply chain management for sustainable performance: Evidence from the airport industry. International Journal of Information Management, 52: 102014.

Dictionary.com. 2023. Quality Definition & Meaning. Dictionary.com. https://www.dictionary.com/browse/quality 8 September 2023.

Gupta, H., Yadav, A.K., Kusi-Sarpong, S., Khan, S.A. & Sharma, S.C. 2022. Strategies to overcome barriers to innovative digitalisation technologies for supply chain logistics resilience during pandemic. Technology in Society, 69: 101970.

Haulder, N., Kumar, A. & Shiwakoti, N. 2019. An analysis of core functions offered by software packages aimed at the supply chain management software market. Computers & Industrial Engineering, 138: 106116.

ISO. 2014. ISO/IEC 25000:2014 Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - Guide to SQuaRE. https://www.iso.org/standard/64764.html 8 September 2023.

ISO. 2011. ISO/IEC 25010:2011 Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models. https://iso25000.com/index.php/en/iso-25000-standards/iso-25010 8 September 2023.

ISO. 2015. Quality management principles.

Kajba, M., Jereb, B. & Obrecht, M. 2023. Considering IT Trends for Modelling Investments in Supply Chains by Prioritising Digital Twins. Processes, 11(1): 262.

Menon, S. & Shah, S. 2019. An Overview of Digitalisation in Conventional Supply Chain Management. MATEC Web of Conferences, 292: 01013.

Merriam-Webster dictionary. 2023. Definition of QUALITY. https://www.merriam-webster.com/dictionary/quality 8 September 2023.

Mohsen, B.M. 2023. Developments of Digital Technologies Related to Supply Chain Management. Procedia Computer Science, 220: 788–795.

Nahil, C. 2020. 7 Quality Management Principles Under ISO 9001:2015. ETQ. https://www.etq.com/blog/7-quality-management-principles-under-iso-90012015/ 8 September 2023.

Nasir, M.H.A., Genovese, A., Acquaye, A.A., Koh, S.C.L. & Yamoah, F. 2017. Comparing linear and circular supply chains: A case study from the construction industry. International Journal of Production Economics, 183: 443–457.

Quality Gurus. 2021. Causes of Poor Software Quality. Quality Gurus. https://www.qualitygurus.com/causes-of-poor-software-quality/ 8 September 2023.

Queiroz, M.M., Pereira, S.C.F., Telles, R. & Machado, M.C. 2019. Industry 4.0 and digital supply chain capabilities: A framework for understanding digitalisation challenges and opportunities. Benchmarking: An International Journal, 28(5): 1761–1782.

Software Project Management. How do you follow software quality standards and frameworks? LinkedIn. https://www.linkedin.com/advice/1/how-do-you-follow-software-quality 8 September 2023.

TQP. 2020. What is Quality? Quality Definition | Quality Meaning. Tech Quality Pedia. https://techqualitypedia.com/quality/ 8 September 2023.