blog.codacy.com

An Exploration of the ISO/IEC 25010 Software Quality Model

Codacy

12~15분

Ensuring software quality today is paramount. From seamless user experiences to robust functionality, software quality directly impacts user satisfaction, organizational efficiency, and even safety critical systems.

There are many software quality models and frameworks for measuring software quality. One of the most commonly adhered to is the ISO/IEC 25010 Software Quality Model, which offers a comprehensive framework to evaluate and improve software product quality.

Let’s take a deep dive into the product quality model ISO/IEC

25010, unraveling its significance and shedding light on how your software development team can use it to unlock excellence and uphold code quality standards.

Understanding ISO/IEC 25010 Components

The ISO/IEC 25010 Software Quality Model, developed by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC), provides a systematic approach to assessing and measuring software quality.

Originally derived from the earlier ISO/IEC 9126 standard, ISO/IEC 25010 offers a more comprehensive and updated framework for evaluating software characteristics and sub-characteristics.

ISO/IEC 25010 organizes software quality into two dimensions: product quality and quality in use. By considering both product quality and quality in use, organizations can adopt a holistic approach to software quality assurance, ensuring that their products meet technical specifications and deliver value and satisfaction to end-users.

Product Quality

Product quality refers to the inherent characteristics of the software product itself. It encompasses functionality, reliability, usability, efficiency, maintainability, and portability. These characteristics are evaluated based on predefined criteria and metrics to assess how well the software meets its intended quality requirements and objectives.

By examining product quality characteristics, organizations can better understand their software's strengths and weaknesses,

enabling them to make informed decisions about enhancements, optimizations, and future development efforts.

Quality in Use

Quality in use, on the other hand, moves focus from the inherent characteristics of the software product to its effectiveness and satisfaction in real-world usage scenarios. When interacting with the software in its intended environment, it considers user satisfaction, productivity, efficiency, and safety factors.

Unlike product quality, which is evaluated based on predefined criteria, quality in use is subjective and context-dependent. It requires gathering feedback from users and stakeholders to understand their experiences, preferences, and needs.

Product Quality Characteristics

Product quality relates to the static and dynamic properties of the software proper. It is divided into eight characteristics.

Functional Suitability

Functional Suitability pertains to the capability system or computer program to deliver functions that precisely address both explicit and implicit user requirements.

- **Functional Completeness** evaluates the inclusivity of functions covering all designated tasks and user objectives without omission.
- **Functional Correctness** evaluates the product's accuracy in delivering precise outcomes, aligning precisely with the required level of precision.
- **Functional Appropriateness** examines the effectiveness of

functions in accomplishing designated tasks and objectives within the intended context.

Reliability

Reliability focuses on the dependability of a system, product, or component in executing predefined functions under stipulated conditions.

- **Maturity** evaluates the readiness of a system, product, or component to meet reliability needs satisfactorily.
- **Availability** assesses the operational state and accessibility of a system, product, or component.
- **Fault Tolerance** gauges the system's operational continuity despite potential hardware or software faults.
- **Recoverability** evaluates the system's capability to retrieve data following interruptions or failures.

Performance Efficiency

Performance Efficiency involves the optimization of resource utilization concerning the performance output of a system or product.

- **Time Behavior** focuses on the system's response, processing times, and throughput rates during operational phases.
- **Resource Utilization** concerns the effective utilization of resources, such as CPU, memory, and network bandwidth, during system operation.
- **Capacity** evaluates the system's maximum limits concerning parameters and its ability to meet them adequately.

Usability

Usability assesses the ease and effectiveness users can achieve predefined goals using a product or system.

- **Appropriateness Recognizability** examines the user's ability to discern the product's suitability for their requirements.
- **Learnability** evaluates the ease of learning to use the product or system effectively, particularly in emergencies.
- **Operability** measures the ease of operation and control of the product or system.
- **User Error Protection** gauges the system's safeguards against user errors to minimize their occurrence and impact.
- **User Interface Aesthetics** evaluates the aesthetic appeal of the user interface and its impact on user engagement.
- **Accessibility** evaluates the product's usability across various user characteristics and capabilities.

Security

Security refers to protecting information and data from potential security vulnerabilities.

- **Confidentiality** focuses on ensuring that data remains accessible only to authorized individuals.
- **Integrity** evaluates the system's capability to prevent unauthorized access or modification to data and programs.
- **Non-repudiation** ensures that actions or events can be irrefutably proven to have occurred.
- **Accountability** refers to the traceability of unauthorized actions back to their originator.
- **Authenticity** concerns the verification of a subject or resource's identity.

Compatibility

Compatibility assesses a product, system, or component's ability to exchange information and perform its functions seamlessly within a shared hardware or software environment.

- **Co-existence** evaluates a product's ability to operate efficiently alongside other products without adverse effects.
- **Interoperability** examines the seamless exchange of information and its utilization across multiple systems and software components.

Maintainability

Maintainability evaluates a product or system's ease of modification to enhance, correct, or adapt to environmental or requirement changes.

- **Modularity** assesses the extent to which system components can be altered with minimal impact on others.
- **Reusability** concerns the potential for assets to be utilized across multiple systems.
- **Analysability** evaluates the effectiveness of impact assessments on planned changes and the system's diagnosability for deficiencies.
- **Modifiability** examines the ease of system modification without compromising quality.
- **Testability** concerns the effectiveness of establishing test criteria and conducting tests to ascertain compliance.

Portability

Portability evaluates a system, product, or component's ease of

transfer between different environments.

- **Adaptability** examines the system's ability to adapt to diverse or evolving hardware, software, and usage environments.
- **Installability** evaluates the system's success in installation and uninstallation processes.
- **Replaceability** gauges a product's potential to substitute another comparable product effectively.

Quality in Use Characteristics

Quality in use relates to the outcome of human interaction with the software. It is divided into five characteristics.

Effectiveness

Effectiveness refers to how software enables users to achieve specific goals wholly and accurately within a given context. It focuses on the software's ability to facilitate successful task completion and attain desired outcomes, ultimately contributing to user productivity and satisfaction.

- **Task Completion** ensures that users can perform intended actions accurately and efficiently to achieve desired results, enhancing overall system effectiveness.
- **Accuracy and Precision** involves delivering results that align precisely with user expectations and requirements, minimizing errors and discrepancies to enhance user confidence and trust in the software's capabilities.
- **Goal Achievement** involves providing necessary functionalities and support to enable users to accomplish tasks and achieve desired outcomes, ensuring user satisfaction and system success.

Efficiency

Efficiency pertains to optimizing resources and effort expended by users in accomplishing tasks with the software. It emphasizes minimizing the time, energy, and cognitive load required to achieve desired outcomes, enhancing user productivity and overall system performance.

- **Resource Utilization** evaluates how effectively the software utilizes system resources such as CPU, memory, and network bandwidth during task execution. Optimizing resource usage (CPU, memory, and network bandwidth ensures efficient operation and prevents wastage, contributing to overall system efficiency.
- **Time Optimization i**nvolves streamlining workflows, reducing latency, and enhancing system responsiveness to facilitate faster task execution and improve user efficiency.
- **Workflow Efficiency** assesses the smoothness and effectiveness of user interactions within the software. It involves eliminating unnecessary steps, reducing user effort, and providing intuitive navigation to enhance workflow efficiency and user productivity.

Satisfaction

Satisfaction reflects users' subjective perceptions and feelings regarding their interaction with the software. It encompasses usability, aesthetics, and emotional responses, influencing user engagement, loyalty, and overall satisfaction with the software experience.

- **Usefulness** evaluates their satisfaction with their perceived achievement of pragmatic goals, including the results and

consequences of use.

- **Trust** assesses how confident the user is that a product will behave as intended.
- **Pleasure** assesses the degree to which a user obtains pleasure from fulfilling their personal needs.
- **Comfort** assesses the degree to which the user is satisfied with physical comfort.

Freedom from Risk

Freedom from risk refers to how well software mitigates or eliminates potential hazards, errors, or adverse consequences arising from its use. It ensures user safety, data integrity, and protection against security threats, enhancing user trust and confidence in the software's reliability and security.

- **Economic Risk Mitigation** measures the degree to which a product or system mitigates the potential risk to financial status, efficient operation, commercial property, reputation, or other resources in the intended contexts of use.
- **Health and Safety Risk Mitigation** assesses the degree to which a product or system mitigates the potential risk to people in the intended contexts of use.
- **Environmental Risk Mitigation** measures the degree to which a product or system mitigates the potential risk to property or the environment in the intended contexts of use.

Context Coverage

Context coverage evaluates the software's suitability and adaptability across various usage contexts and environmental conditions. It considers factors such as the diversity of users,

tasks, and operating environments, ensuring that the software remains practical and usable across different scenarios and user groups.

- **Context Completeness** assesses the effectiveness, efficiency, freedom from risk, and satisfaction in all contexts of use.
- **Flexibility** assesses the effectiveness, efficiency, freedom from risk, and satisfaction in contexts not originally specified in the requirements.

ISO 25010 in Practice

ISO 25010 is an excellent addition for enterprise software teams who want a framework evaluating software product quality. By breaking down quality characteristics into sub-characteristics, developers can go on to define software metrics that make sense for their projects.

It does not provide a comprehensive and systematic mapping of sub-characteristics to software metrics. That's because individual circumstances matter most. For example, code quality for automated high-frequency trading software has an entirely different range of consequences than code quality for embarked software used on aircrafts.

The stakes in each development project are different and call for different priorities, metrics, and contingency plans.

Maintaining coding standards can be tough without the right tools. Static code analysis tools like linters can help you automate the code review and analysis process, making it easier for you to stick to established coding standards.

Codacy allows you to automate the static code analysis process by creating coding standards within the platform to ensure that

groups of repositories follow the same security rules or coding conventions, for example. To see how it works, start your free 14-day Codacy trial today.