NUMBER:
Effective: June 28, 2023

Replaces:
N/A

WaTech
Washington Technology Solutions

SOFTWARE QUALITY BEST PRACTICES GUIDELINE

See Also:
RCW 43.105.054 OCIO Governance
RCW 43.105.205 (3) Higher Ed
RCW 43.105.020 (22) “State agency”

Contents

Introduction

This document outlines the software quality standards and best practices to be followed by development teams at Washington state. These standards aim to ensure the delivery of high-quality software products that meet customer expectations and align with ISO/IEC 25010 and ISO/IEC 25023.

Scope

Agencies are encouraged to follow these guidelines. For gated projects subject to section 701 of the Washington State legislative budget, sections 1-4 are given priority review for the gating process.

1. Process and Documentation
    a. Follow a defined software development process that encompasses requirements for gathering, designing, coding, testing, and deployment.
    b. Maintain comprehensive and up-to-date documentation, including requirements specifications, architectural designs, user manuals, and test plans.
2. Code Quality and Standards
    a. Adhere to a set of coding standards that define naming conventions, code structure, indentation, and formatting to enhance code readability and maintainability. Ensure that coding standards are easily accessible and well-documented for all developers.
    b. Follow a modular and component-based approach to design and development, promoting reusability and maintainability of code.
    c. Use meaningful and self-explanatory variable, function, and class names to enhance code readability. Avoid ambiguous or cryptic names.
    d. Write code that is concise and follows the principle of "Don't Repeat Yourself" (DRY) to minimize duplication and improve maintainability.
    e. Break down complex logic into smaller, manageable functions or methods, each responsible for a specific task or functionality.
    f. Implement error handling and exception management mechanisms to handle unexpected situations and ensure graceful degradation.
    g. Strive for code simplicity and clarity by avoiding unnecessary complexity or convoluted logic. Favor straightforward solutions over overly clever or complex ones.
    h. Ensure that code is well-commented, providing explanations for non-obvious sections, complex algorithms, or important business rules. Follow consistent commenting styles and use descriptive comments.
    i. Regularly refactor code to improve its structure and enhance readability. Utilize code analysis tools or linters to identify potential issues or violations of coding standards.

j. Write unit tests to validate individual components or functions and ensure their correctness. Aim for comprehensive test coverage, including both positive and negative test cases.

k. Maintain a clean and organized codebase, adhering to directory structure conventions and separation of concerns principles.

l. Continuously review and optimize code performance, identifying and addressing bottlenecks or inefficiencies. Consider using profiling tools to identify areas that require optimization.

m. Stay updated with new language features, best practices, and emerging coding standards relevant to the programming languages and frameworks used in your project.

n. Foster a collaborative coding culture where team members can provide feedback, conduct code reviews, and suggest improvements. Encourage open communication and constructive criticism.

3. Measurement and Evaluation- ISO/IEC 5055 based guideline

ISO/IEC 5055:2021 is an international standard for measuring the quality and integrity of a software system by analyzing its internal construction to detect several structural weaknesses.

These weaknesses are related to four business-critical factors:

- Security
- Reliability
- Performance Efficiency
- Maintainability

The standard provides a set of automated source code quality measures that can be used to evaluate and improve the software product during development and maintenance.

The following guidelines are based on the ISO/IEC 5055 standard and aim to help software developers and managers achieve high-quality software products that are trustworthy, dependable, and resilient.

a. Use static analysis tools that implement the ISO 5055 standard to detect, report, and measure the weaknesses across the entire technology stack and its interconnections. The tools should provide a comprehensive coverage of the weaknesses specified in the standard and support the languages and platforms used in the software system.

b. Establish quality goals and thresholds for each of the four factors (Security, Reliability, Performance Efficiency, and Maintainability) based on the business requirements and risks of the software system. The quality goals should be expressed in terms of the ISO 5055 measures, such as the number

or density of weaknesses per factor or the sigma level achieved by the product.

c. Monitor and track the quality of the software system throughout its life cycle using the ISO 5055 measures. Compare the actual quality measures with the quality goals and thresholds and identify any gaps or deviations. Use dashboards and reports to communicate the quality status and trends to all stakeholders.

d. Prioritize and remediate the weaknesses detected by the ISO 5055 measures according to their severity and impact on the four factors. Focus on eliminating the most dangerous weaknesses that could result in unacceptable operational risks or excessive costs. For example, 'Ban Unintended Paths' is an architectural weakness that violates security and data protection controls by allowing a path from the user interface directly to the database without passing through user authentication routines.

e. Apply good architectural and coding practices to prevent or reduce the introduction of new weaknesses in the software system by following the coding standards and guidelines that are consistent with the ISO 5055 measures to avoid common pitfalls and anti-patterns that could compromise the quality of the software system.

f. Use code reviews and testing to verify and validate the quality of the software system.

4. Testing and Quality Assurance

a. Conduct thorough testing throughout the software development lifecycle, including unit testing, integration testing, system testing, and user acceptance testing.

b. Use appropriate testing techniques and methodologies to address the identified quality characteristics and sub characteristics.

c. Consider the use of automated testing tools and frameworks to improve efficiency and effectiveness.

5. Defect and Issue Management

a. Utilize a robust defect tracking system to capture and manage software defects and issues effectively.

b. Prioritize and address reported defects promptly, ensuring appropriate communication and resolution within the defined timelines.

c. Establish a process for root cause analysis to identify and address underlying issues contributing to recurring defects.

6. Security and Compliance

a. Implement secure coding practices to prevent common vulnerabilities, such as input validation errors, injection attacks, and insecure access control.

b. Stay up to date with industry security standards and guidelines to ensure compliance with relevant regulations, such as CJIS, IRS Pub 1075, and HIPAA.

c. Conduct regular security assessments and penetration testing to identify and mitigate potential security risks.

7. Continuous Improvement

   a. Foster a culture of continuous improvement by encouraging feedback and suggestions from team members.

   b. Conduct periodic retrospectives to reflect on the software development process and identify areas for improvement.

   c. Stay informed about advancements in software development practices, tools, and technologies to enhance software quality and efficiency.

8. Training and Knowledge Sharing

   a. Provide training and resources to ensure that team members are equipped with the necessary skills and knowledge to meet software quality standards.

   b. Encourage knowledge sharing within the team through technical presentations, workshops, and collaborative platforms.

9. Compliance and Audit

   a. Regularly review and assess compliance with software quality standards, both internally and through external audits if required.

   b. Maintain records and evidence of compliance to demonstrate adherence to established standards.

REFERENCES

1. SB 5187-S.S.L Section 701 (2) and Section 701(9)
2. ISO/IEC 25010:2011 - Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models
3. ISO/IEC 25023:2016 - Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Measurement of system and software product quality
4. ISO/IEC 5055:2021 - Information technology — Software measurement — Software quality measurement — Automated source code quality measures