Critical Logic
INTEGRATED QUALITY MANAGEMENT

Washington State Software Quality Checklist for IT Projects

WaTech's Office of the CIO (OCIO) issued a memorandum for "Software Quality Best Practices" that provides a standards-based approach to achieving Software Quality.

This memo was issued to ensure IT Projects don't fall into the trap of focusing solely on measures of "project quality" like project budget, schedule, scope, stakeholder communication, etc.

Successful projects rely on other signs of quality – focused on the development process itself, how the code is written, the correctness of the behavior of the software and even the results of the software being used.

On large-scale projects for complex systems, the quality of the software being developed is the primary determinant of the satisfaction of stakeholders with the project's scope, schedule and budget. Measures of scope, schedule and budget are really driven by the underlying quality of the software and the work required to develop it.

Want to achieve OCIO compliance, but not sure whether you're covering all the quality bases? This checklist helps you identify actionable and specific items around Process Quality and Software Product Quality.

***Note*:** *Items in **orange bold** are directly from the 2023 OCIO Memorandum on Software Quality*

Process Quality: *Is your project set up for quality success?*

Follow a defined software development process that encompasses requirements for gathering, designing, coding, testing and deployment.

- ☐ Have you chosen the right methodologies for the project? Factors include project size, nature of the system being developed, complexity of the domain, volatility of requirements, number of stakeholders and stakeholder groups, project team capabilities, organizational characteristics and project business goals. No "one-size-fits-all," and no "silver bullet." A blend of approaches from different "methodologies" is usually appropriate.
- ☐ Is the Software Development Life Cycle fully defined in terms of roles, functions and processes?
- ☐ Does the Project Plan include well-defined processes for collaboration across working groups within the project team, to prevent "siloing"?
- ☐ Are there built-in iterations and mechanisms for measuring process performance and fostering improvement along the way?

Critical Logic
INTEGRATED QUALITY MANAGEMENT

Maintain comprehensive and up-to-date documentation, including requirements specifications, architectural designs, user manuals and test plans.

Define the Business Analysis approach for the project.

- [ ] Has the Stakeholder Analysis been conducted and a Stakeholder Engagement Plan developed?
- [ ] Are the types of requirements, levels of detail and documentation approach fully defined?
- [ ] Are processes for defining and maintaining Business Requirements, Stakeholder Requirements and Solution Design Requirements established throughout the entire life of the project?
- [ ] Is the process for establishing Quality Requirements defined?
- [ ] Are the BA information items, requirement viewpoints and Requirement Traceability relationships defined?
- [ ] Are the expectations for requirements of reusability clearly established (across project phases and/or throughout the life of the system, which may span multiple projects)?
- [ ] Are the objectives of each type of requirement deliverable defined so evaluators can verify completeness based in part on whether they meet the defined objectives for that type of deliverable? For an example of how Critical Logic defines completeness criteria, see Appendix B.

Define the Software Testing approach for the project.

- [ ] Has a Master Test Plan been developed?
- [ ] Does the Master Test Plan define the types of testing to be performed and the rationale (e.g., as a function of assessed risk)?
- [ ] Are test coverage standards defined for each Type of Testing defined in the Master Test Plan?
- [ ] Does the Test Strategy identify Test Stages (e.g., Integration, System, Acceptance, etc.), and which Types of Testing will be performed by whom in each?
- [ ] Does the Master Test Plan identify the lower-level Test Plans that will need to be developed together later with the elaboration of scope and Solution Design Requirements, including what the lower-level Test Plans will define (e.g., test groups, test environments, test data requirements, etc.)?

Critical Logic
INTEGRATED QUALITY MANAGEMENT

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

Are your Quality Requirements identified, captured and validated based on the SQuaRE Quality Model established in the ISO/IEC 25010 standard?

A complete list of the Quality Characteristics that Quality Requirements can be written for in the standard is found in Appendix A.

- ☐ Have you identified, defined and validated Quality Requirements and measures for **Process Quality**?
- ☐ Do your Quality Requirements satisfy external compliance criteria (e.g., IV&V requirements for Process Quality imposed by a federal system/program)?
- ☐ Have you identified, defined and validated Quality Requirements and measures for **Code Quality?**
- ☐ Do your Quality Requirements satisfy external compliance criteria (e.g., specific coding practices required for systems that demand high security like voting systems)?
- ☐ Have you identified, defined and validated Quality Requirements and measures for **Software Quality**.
- ☐ Do your Quality Requirements satisfy external compliance criteria (e.g., functional test coverage standards for critical systems with zero defect tolerance)?
- ☐ Have you identified, defined and validated Quality Requirements and measures for **Quality In-Use?**
- ☐ Are your Quality Requirements sufficiently measurable to ensure the Software In-Use satisfies the business case for the project (e.g., S.M.A.R.T. (Specific, Measurable, Achievable/Assignable, Realistic, Time-bound)) goals?
- ☐ Have you defined the process for verifying that the defined Quality Requirements are being met throughout the life of the project (frequency/timing of verification, environment, specific measures per Quality Characteristic, evaluated by whom, etc.) to foster a culture of continuous improvement?

Internal Product (Code) Quality: Does the technical work being done satisfy your criteria for reusability, scalability, durability and others?

- ☐ Have you established further Quality Requirements based on ISO/IEC 5055?
- ☐ Do the system architecture and code itself satisfy the code Quality Requirements established during planning using the ISO/IEC standards?
- ☐ **Do the system architecture and code itself follow the specific ISO/IEC 5055 recommendations set forth by the OCIO?**
- ☐ **Are your developers and other technical resources following the additional "Code Quality and Standards" guidelines expressly set forth by the OCIO?**

Critical Logic
INTEGRATED QUALITY MANAGEMENT

Software Quality: Does the software product work right?

Planning & Policy

- [ ] Is the Master Test Plan current and validated based on discoveries and elaboration of Stakeholder Requirements, Solution Design Requirements and scope definition throughout the life of the project to date?
- [ ] Have you defined the **Stage Test Plan(s)**, which define more specifically the who, why, when, what and how of each Test Stage (Unit, Integration, System, Acceptance, etc.), with traceability to established Quality Requirements?
- [ ] Are Test Stage entry and exit criteria established (e.g., percent of tests successfully executed once, number of severe defects, etc.)?
- [ ] Have you defined the **Group Test Plan(s)**, which define more specifically functionality groups that require unique treatment in terms of environmental considerations, stakeholders, project staff, test data management, Test Types/approaches, test automation, test coverage standards, etc.?

Test Management

- [ ] Has the tool for **Test Case tracking** ("Test Management System") been identified, installed and configured to adhere to the stipulations of the Master Test Plan?
- [ ] Has the tool for **Test Result and defect tracking** ("Test Management System") been identified, installed, and configured to adhere to the stipulations of the Master Test Plan?
- [ ] Are the "black box" tools for test execution of all the different functional layers of the system identified and implemented (e.g., web browsers, data interface inspection tools, database querying, etc.)?
- [ ] Has a rationale for and usage of Test Automation been established?
- [ ] Has the tool for Test Case tracking been identified, installed and configured to adhere to the stipulations of the Master Test Plan?
- [ ] Has the process for defect triage, management and escalation been established and validated?
- [ ] Has the process for test execution status reporting been established?
- [ ] Has product and project risk been identified?

The OCIO stipulates the following general guidelines for Software Testing, which are expanded in this section into a more detailed checklist.

- **Conduct thorough testing throughout the software development lifecycle, including unit testing, integration testing, system testing and user acceptance testing.**
- **Use appropriate testing techniques and methodologies to address the identified quality characteristics and sub-characteristics.**
- **Consider the use of automated testing tools and frameworks to improve efficiency and effectiveness.**
- **Utilize a robust defect tracking system to capture and manage software defects and issues effectively.**
- **Prioritize and address reported defects promptly, ensuring appropriate communication and resolution within the defined timelines.**
- **Establish a process for root cause analysis to identify and address underlying issues contributing to recurring defects.**

Test Execution

- [ ] Have the requirements deliverables to be verified (specs, user stories, etc.) been validated to confirm they meet the criteria for completeness set forth and agreed to in the Business Analysis plan and finalized with any necessary updates?
- [ ] Are all the necessary environments and related permissions in place to execute the necessary tests?
- [ ] Are all the necessary stakeholders prepared to engage at the appropriate time for the appropriate purpose according to the Test Plans?
- [ ] Is the plan for managing test data agreed upon and ready to execute? (e.g., procedures for restoring data to its original condition after data "burn" during initial tests)?
- [ ] Are the necessary Test Cases authored, validated and approved for their adherence to the testing depth and breadth standards defined in the Test Plans?
- [ ] Are all the test-case-specific "pre-conditions" in place or available to be put in place to execute the necessary tests?
- [ ] Have all the necessary tests been executed at least once (even if they've returned an unexpected result)?
- [ ] Is regression testing complete to ensure no unexpected changes were introduced by the latest release?
- [ ] Are the test execution results recorded completely and correctly?
- [ ] Have all the defects and issues been triaged and dispositioned appropriately in the Test Management System?
- [ ] Have the criteria for exiting the current Test Stage been met, as defined in the Stage Test Plan?
- [ ] Have the criteria for entering the next Test Stage been met, as defined in the Stage Test Plan?
- [ ] Is the test automation work for the current iteration complete and ready for the next iteration?

Critical Logic
INTEGRATED QUALITY MANAGEMENT

In-Use Quality: Are you meeting the goals you set when you made the business case for the project?

- ☐ Define/refine your Quality In-Use Requirements for each of the applicable Quality Characteristics from the ISO/IEC 25000 Standards Family (please reference Appendix A– ISO/IEC 25000 Standards Family" for a complete list of the Quality Characteristics)
- ☐ Are business goals and objectives defined using the S.M.A.R.T. framework so that specific measures can be assessed?
- ☐ Have you checked for overlap between business goals and quality in-use requirements (e.g., "Achieve 20% increase in enrollments through the new system in the first year" might be a function of measures of the Interaction Capability characteristics of the system – usability, learnability, etc.)?

Got questions? Contact us.

Critical Logic's specialized team of Software Quality Assurance professionals knows what it takes to ensure your software investment is set up for success, stays on track and delivers the highest value to all stakeholders. We'd love to discuss your next project with you.

Let's talk

Critical Logic
INTEGRATED QUALITY MANAGEMENT

Appendices

Appendix A – ISO/IEC 25000 Standards Family

Quality Characteristics

The following diagram, from the ISO/IEC 25000 family of standards for Software Quality, shows the "Quality Characteristics" menu for writing Quality Requirements in the SQuaRE model. The Requirements per characteristic and measures used per characteristic can vary depending on which of the four levels are being addressed. The four sections in this Checklist are Project Quality, Code Quality and Product/Software Quality, or Quality in-Use.

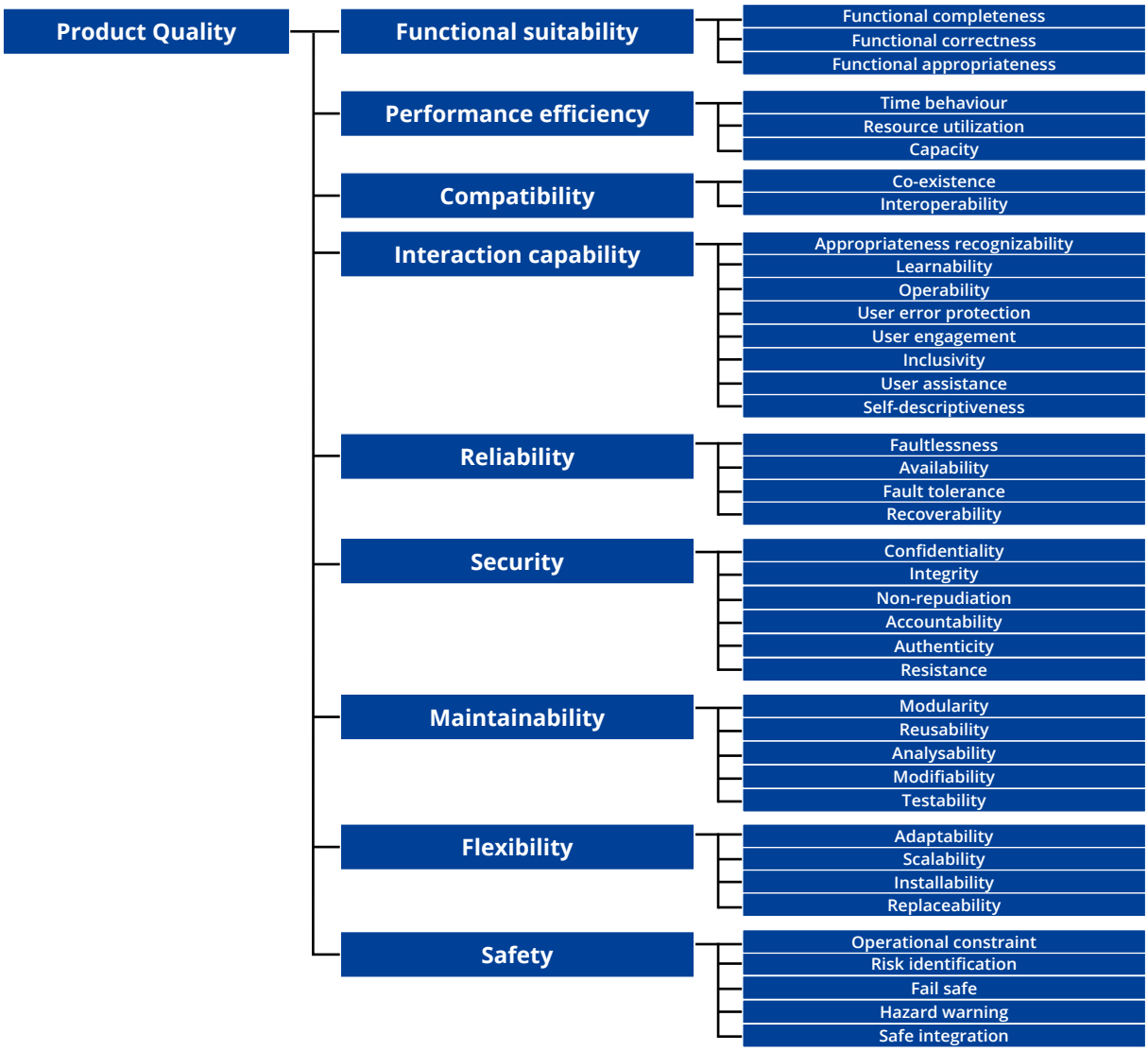

Figure 1 - Software Product Quality Model Characteristics (from ISO/IEC 25010:2023)

Quality Measures

The following diagram, from ISO/IEC 25023 "SQuaRE – Measurement of System and Software Product Quality," shows the relationship between the different types of quality measures. The four types of quality measures align with the four sections of this Checklist.

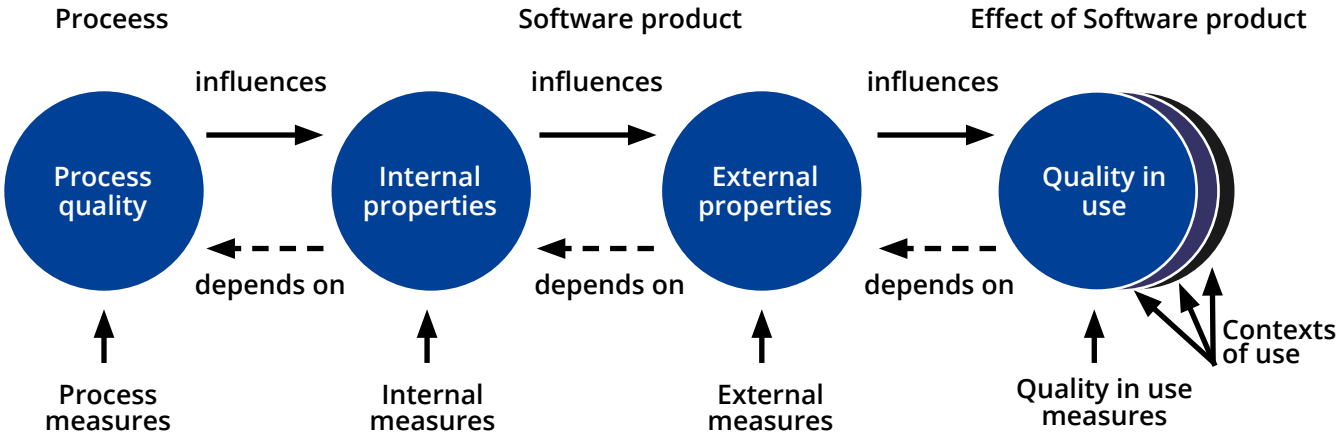

Figure 2 - Relationship Between Types of Quality Measures (from ISO/IEC 25023:2016)

Appendix B – Sample "Completeness Criteria" for software product Solution Design Documentation

The following is an example of the "Completeness Criteria" Critical Logic develops for a specific type of requirements deliverable to be tested. In this case, the deliverable is the document containing the functional solution design requirements for a data interface between two systems, sometimes also known as an "interface spec."

Critical Logic Completeness Validation Criteria: Data Interface Specifications

Problem Statement

A Data Interface Specification (DIS) should give the solution specialists (developers, vendor configuration experts, etc.) everything they need to know to formulate the solution.

All record and field-level information pertinent to the interface processing logic should be captured in the DIS. This "data mapping" includes all the necessary business rules to perform the mapping correctly, including selection criteria and data transform logic.

Business scenarios and test cases that exercise functionality defined in the DIS should provide clear traceability to the DIS for leverage when reporting failures and for coverage validation.

Business scenarios should provide sufficient breadth and depth to satisfy the business team's requirements for validating the data interface processing, including any unusual cases they may have in mind.

Detailed test cases should provide 100% functional coverage of all logic defined in the DIS.

Checklist

The following details should be defined in the IS for implementation and testing:

1. **Triggering Event Rules** –sometimes multiple events
2. **Selection Criteria** – are certain records excluded?
3. **Default Values** – defaults for target fields with no source or other exceptions
4. **Passthroughs** – fields for which the value is passed from source to target as-is
5. **Transformations** – rules for transformed or derived values
6. **Data Cleansing** – standard rules for target field formats, data types, field lengths

DIS Author/Analyst and Software Tester Guidelines

- What business information is being processed (not just data, but understanding what it means to the business)?
- Learn the logical structure of the data being converted.
  - Are there related tables? How are they related? Processing logic for entities in a many-to-many relationship, for example, may need to account for more possibilities than in a one-to-many relationship.
  - Is the source structure different than the target structure? If so, what are the mapping rules?
- Record Selection Criteria.
  - Sometimes, there is business logic to only send or receive certain kinds of records. This logic must be validated to ensure a given record that should be sent/received is, and a given record that should not be sent/received is not. (Note: The boundaries for the negative test are not always obvious and should be clearly documented before test design.)
- Defaults
  - For each target field: What is the default value if there is no source field?
  - For each target field: Does the system provide a default or simply pass blank if the source is blank?
- Passthroughs. Every passthrough must be validated. (Verify that Value A in the source field is passed through as Value A in the target field.)
- Transforms. Every transform must be fully exercised, including boundaries, etc.
- Derivations/lookups
  - For each target field whose value is derived on the fly (e.g., lookup “Vendor Name” based on “Vendor ID” in the source record), what is the expected behavior if a corresponding value is not found? Pass null? Reject the record? Stop the job?
  - For each derived value, what if more than one corresponding value is returned? Should the system just take the first one? Store both? Reject the record? (Note: In many cases this may be impossible, in which case it would not be a valid test. For example, if the lookup table is indexed by the field you’re doing the lookup with, then by definition there will never be more than one match.)

Business Scenario and Test Case Levels of Detail

Data Interfaces are not typically direct targets of business scenario testing but rather are invoked indirectly through the scenarios. Business scenarios written to test a DIS are likely to be at a high level of detail and may require additional collaboration between the business team and dev/test to ensure correct inputs and outputs.

The detailed test cases written to exercise a Data Interface Specification must incorporate all the record and field-level processing logic, plus the detailed steps to initiate the data interface and navigate to a form, screen or data file to verify the results.

The instructions to the tester detailing how to start the processing and where to go to perform the verifications already described in the test cases must accompany the test cases. Ideally, this will be in the database and/or the data container that transports the data to/from another system.

Note: *The steps to perform the necessary data setup for a given test script are not considered part of the test script. Critical Logic will provide a report detailing the data conditions necessary to start each test script (Preconditions Report).*

Summary

The Data Interface Specification and its corresponding test cases must incorporate all record and field-level information pertinent to the conversion processing logic and provide 100% functional coverage.

Early identification of this detail during spec writing reduces surprises during implementation. Strong traceability between tests and design requirements is maintained. Disputes over scope and intent are reduced.

The completely defined functional Data Interface Specification ensures accurate, efficient development and delivery. Detailed test cases also provide a basis for regression testing across cycles and release cycles.