FSW QA Testing Levels Definitions

1. Overview

This document is used to help determine the amount *and* quality of testing (or its *scope*) that is planned for or has been performed on a project. This analysis results in the testing effort being assigned to a particular *Testing Level* category.

Categorizing the quality and thoroughness of the testing that has been performed is useful when analyzing the metrics for the project. For example if only minimum testing was performed, how come so many person-hours were spent testing? Or if the maximum amount of testing was performed, how come there are so many trouble calls coming into the help desk?

2. Testing Level Categories

This section provides the basic definitions of the six categories of testing levels. The definitions of the categories are intentionally vague and high level. The Testing Level Matrix in the next section provides a more detailed definition of what testing tasks are typically performed in each category.

The specific testing tasks assigned to each category are defined in a separate matrix from the basic category definitions so that the categories can easily be re-defined (for example because of QA policy changes or a particular project's scope - patch release versus new product development).

If it is decided that a particular category of testing is required on a project, but a testing task defined for completion in that category is not performed, it will be noted in the Test Plan (if it exists) and in the Testing Summary Report.

Category	Basic Definition
0 - None	No testing was performed for this release of the product.
1 - Minimal	Risk based ad hoc testing focused on testing new features and defects that were fixed. Little or no testing documents are created.
2 - Moderate	Risk based ad hoc testing focused on testing new features, bug fixes, and augmented with ad hoc functional testing of the product's pre-existing functionality that has been designated as "business critical". A Test Plan document and *possibly* Test Case documents are typically created, but no Test Procedure documents are created.
3 - Complete Functional	All of the product's functionality (new and pre-existing) is tested using a formal testing process that typically includes the creation of a Test Plan, Test Cases, and Test Procedures for all Test Cases.
4 - Extensive	Complete Functional plus systems type testing (interoperability, platform, end-to-end, etc.) is performed. In addition, all internal (for example, Requirements document) and external deliverables (for example, User's Guide) are reviewed.
5 - Comprehensive	Extensive testing plus all other required testing and quality assurance functions (performance, stress, internationalization, etc.). In addition, test procedures judged to provide enough return on investment are automated.

Table 1 - Testing Levels Categories

3. Testing Levels Matrix

This section contains the matrix of which specific testing tasks are assigned to which testing level categories.

	Category					
QA Tasks	0	1	2	3	4	5
Installation / Uninstallation testing		✓	✓	✓	✓	✓
Perform boundary testing		✓	✓	✓	✓	✓
Functionality testing		✓	✓	✓	✓	✓
Virus Check		✓	✓	✓	✓	✓
Scenario testing			✓	✓	✓	✓
Perform database testing				✓	✓	✓
Perform end-to-end testing				✓	✓	✓
Field Entry testing				✓	✓	✓
Interface testing				✓	✓	✓
Regression testing				✓	✓	✓
Perform application interoperability testing				✓	✓	✓
Perform OS interoperability testing				✓	✓	✓
Perform Hardware, Network compatibility testing				✓	✓	✓
Architecture reviews					✓	✓
Code reviews					✓	✓
Dependencies					✓	✓
Functional and Test Spec Review					✓	✓
Requirements, Design docs Review					✓	✓
Usability testing					✓	✓
Specification based testing					✓	✓
Test Progress and Summary Review					✓	✓
User Documentation Review					✓	✓
International/Localization					✓	✓
Training Documentation Review					✓	✓
Perform error handling testing/Recovery testing						✓
Performance testing						✓
Submit testing summary report						✓
Reliability testing						✓
Automated Testing						✓
Stress testing						✓

Table 2 - Testing Levels Matrix

4. QA Task Definitions

4.1 Reviews

4.1.1 Project review

A formal review of a project's scope and application. QA's role is to assess the project and influence the establishment of initial requirements with respect to reliability and testing.

4.1.2 Architecture review

A formal review of a project's architecture, network interfaces, and data flow. QA's role is to influence the architecture towards reliable design, and to gain understanding of it in order to formulate the test design strategy.

4.1.3 Functional review

A formal review of a project's functional design. QA's role is to assure the project's requirements have been satisfied, and to gain further understanding of it in order to block out key testing requirements.

4.1.4 Design review

A detailed design review is presented by the developer of a specific aspect of the project's overall design, and allows for the opportunity to assess the design with respect to the input requirements. QA's role is to assure requirements have been addressed, that the design will produce a reliable implementation, and to further detail the developing test procedure.

4.1.5 Design documentation review

There are a number of documents that are produced during a project. QA's role in reviewing these documents is to help assure clarity, consistency, and completeness. Typical documents include the Software Requirements Specification (SRS), architectural spec, functional spec, design spec, and comments inserted into the actual code.

4.1.6 Code reviews

A code review is a detailed investigation into the developing, or developed, code generated from the SRS. QA's role is to help assure the code meets acceptable coding standards, and that the implementation of the requirements have been robustly met.

4.1.7 Test specification review

A test spec review is presented by the QA analyst for the purpose of reviewing it against the project requirements and design specification. Key goals are to show adequate test scope, coverage, and methodologies.

4.1.8 Test documentation review

Test documentation reviews are presented by the QA analyst for the purpose of reviewing the test plan and procedures to show they are in line with project deliverables.

4.1.9 User documentation review

A review of the user documentation helps to assure that information being sent to the customer is correct, accurate, clear, and concise.

4.1.10 Training documentation review

A review of the training documentation helps to assure that information being sent to the trainers and trainees is correct, accurate, clear, and concise.

4.2 Documentation Output

4.2.1 Test Plan

The Test Plan is a high level planning document created by the QA analyst in the early stages of a project. Key topics are: Scope, Staffing, Resources, Strategy, Schedule.

4.2.2 Test Procedure

The Test Procedure is a detailed document which defines and guides the testing, and is written by the QA analyst. Individual scripts and scenarios make up the bulk of this document.

4.2.3 Test Progress status reports

During active testing, periodic status reports (generally, weekly) are generated by the QA analysis to provide test status information to the project team.

4.2.4 Defect Reports and Bug Scrubs

Defects are verified and logged into the Clear Quest database. Periodic “bug scrubs”, or defect review meetings, are conducted by the QA analyst with the project team in order to share and understanding of each defect, to assign ownership, and corrective action.

4.2.5 Test Summary Report

A test summary report is generated by the QA analyst for each project release meeting. This report summarizes testing status to date, including identification and resolution of defects. During an application investigation, a Software Quality Analyst (SQA) must interact with a number of team members, with the developers and Product Manager being the primary interfaces. As a service to them, a Summary Report of defects is maintained and distributed by the SQA. This tool allows, in an easily digestible format, the communication of key issues to the “information consumers”. Developers can more readily focus on the issues important to the project, and the Product Manger can more easily assess the project risk at any one moment. A Summary Report must be compiled and brought to current project status at project milestones where testing has been active.

4.3 Types of Testing

4.3.1 Installation / Un-installation testing

An **installation test** investigates the effects on the application and on the target execution system when installed and executed on the target system. Original installation on a clean target system, update installation on a pre-existing installation, and re-installation options are explored.

The **de-installation test** investigates the effects on the target system when the previously installed application is removed.

4.3.2 Virus Check

A **virus check** helps to ensure against the propagation of a virus or worm from distribution on media which will be mass produced.

4.3.3 Functional testing

Functional testing explores the application’s key features and functions to specific requirements. It is generally a higher level test to demonstrate overall compliance to key requirements. It is concerned with what works, not how or why it works.

4.3.4 Performance testing

Performance Testing investigates the speed or response of an application to some desirable/specified amount.

4.3.5 Compatibility testing

Compatibility testing examines the software under test to a pre-existing software or hardware environment. Corruption or malfunction should not be observed in either the application nor the software environment. The application should be tested against the

hardware environment to ensure compatibility with networks (routing, permissions, security, latency, etc…). Further testing should examine the application's operational effects on common software packages likely to coexist within an environment. This is the same thing as what is sometimes called Application and Operating System interoperability testing.

4.3.6 Dependency testing

Dependence testing examines the application's requirements for pre-existing software, initial states, and configuration in order to maintain proper functionality.

4.3.7 Usability testing

Usability testing analyzes user and usage needs. It checks whether the product is useful, easy to use and learn. All in all, it must satisfy users needs.

4.3.8 Regression testing

Regression testing is the selective re-testing to detect faults introduced during modification of a system or system component, to verify that modifications have not caused unintended adverse effects, and to verify that a modified system or system component still meets its specified requirements.

4.3.9 Stress testing

Stress Testing, or sometimes called Volume Testing, investigates the ability of an application to process data or commands at rates significantly different than "normal" usage. Rates are usually higher than normal, but may also be lower than normal in some applications. Complex applications that may have various modes or components come into play based upon some set of circumstances, may have to be set up and configured to get the application into a higher risk state condition.

4.3.10 Scenario testing

Scenario Testing employs a specific series of steps, or conditions, that individually may not reveal a defect, but together may do so. It is a sequence of operations that establish initial conditions for a defect to be revealed.

4.3.11 Reliability testing

Reliability testing is how well (robustly) a product handles failure, data integrity, safety and security. It measures if the product will result in expected and consistent results with the same input. It explores the probability of failure-free operation for a specified period.

4.3.12 Specification based testing

A **Specification based test** restricts a software investigation to the exact scope of the documented specifications and requirements.

4.3.13 Error handling and recovery testing

An error and recover test explores an application when specific errors are introduced and the behavior of the application is observed.

4.3.14 Boundary testing

Boundary value testing ensures the proper function to work at the boundary (or edges) of allowable data input. Boundary values include maximum, minimum, just inside/outside boundary, typical values, and error values.

4.3.15 Field Entry testing

Field entry testing examines data that is entered in screen forms intended for database input. Testing includes data values, ranges, invalid input, GUI controls, error recovery, and any processing that might occur as a result of data input.

4.3.16 Database testing

Database testing validates that data written to, and read from, a database is correct. Any processing, conversion, and data recovery situations should be explored.

4.3.17 End-to-end testing

End-to-End Testing investigates the final expected results with data, or process, flow from the very start of a process to the very end. In complex systems, in terms of both hardware networks and software processes, there can be more than one path taken that may need to be investigated.

4.3.18 Interface testing

Interface testing examines protocols, handshaking, data and control exchanges between software components that share or pass information.

4.3.19 International / Localization

International and Localization of software generally means customization of the GUI to meet different language and usage requirements. Generally, the underlying functions remain the same. Testing should assure that all possible user messages (e.g. the GUI, dialog boxes, help, and error messages) are properly managed to the specific configuration requirements.

4.3.20 Automated Testing

Automated testing utilizes tools and techniques that minimize or do not require operator input, analysis, or evaluation in order to perform a software inspection.

