Document Number: MQXTEST
Rev. 1.0.0, 02/2014

MQX Test Methodology

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

Document Number: MQXTEST
Rev. 1.0.0, 02/2014

1.0 Contents

2.0 Introduction

2.1 The purpose of testing

Purpose: The purpose of testing is to ensure that the Microcontrollers Platform SW deliverables have met the appropriate quality level and are delivered in a timely manner. Additionally, the purpose of testing is to provide a clear and concise feedback to the development teams.

2.2 Test scope and objective

The main objective for **MQX SW** testing is to detect and eliminate defects as early in the process as possible and to ensure that no new defects are introduced moment during the development cycle.

In addition to eliminating defects, testing also verifies the source code and ensures that the project deliverables meet **software requirement specifications** and customer expectations. Finally, testing validates the new feature development and ensures that bugs, which have been fixed in previous releases, are not reintroduced and that the existing functionality is not broken.

2.3 Quality objectives

The primary quality objective of testing is to ensure that all required features were developed in accordance with expectations and **software requirement specifications** without breaking backward compatibility or existing functionality.

Secondary quality objective is to identify defects, communicate issues, and ensure that all defects that affect the quality level of the software product functionality or end-user experience are addressed and fixed before the project is released.

2.4 Definitions, Acronyms, and Abbreviations

MQX SW is Freescale Real-Time Operating System (RTOS) providing real-time performance within a small and configurable footprint.

Product Requirements Document (PRD) defines the **MQX SW** and its requirements for new features. It is internal-facing and allows teams to understand what a product should do and how it should work.

Software Requirement Specification (**SRS**) describes functionality and behavior of the software product kernel and peripheral components. SRSs are used for the software and test design and development.

Test Strategy is a document describing the testing approach for the software development cycle. It is created to inform project managers, testers, and developers about the testing processes.

Test Plan is a document describing test schedule, test resources and testing environment.

White Box is a device, system, or object, with a known internal structure, processing, inputs, outputs, and relationship between each component.

Black box is a device, system, or object which can be viewed solely in terms of its input, output and transfer characteristics without any knowledge of its internal workings. In other words, its implementation is "opaque" (black).

3.0 Test levels

3.1 Unit testing

Unit testing, also known as component testing, verifies a specific section of the source code to find problems in the early development stage. Unit testing is performed during a new feature or a component development cycle. Unit tests are usually white box tests and may be done in isolation from the product. Tests are designed by the development team and executed during a new development phase.

3.2 Regression testing

Regression testing ensures that the software product meets all requirements and quality at any given time. Testing reveals defects during development and when implementation ends and new source code is integrated into the software product. Regression testing validates the stability of the product and ensures that the software product remains uncorrupted. Tests are executed either on a daily basis delivering continuous software quality measurements, or on demand validating the newly-implemented features or bug fixes before the integration.

Test cases are designed and implemented by the test team based on either requirements or business. Requirements-driven test cases involve software functionality and behavior requirements and business driven involve use-case scenarios and user stories. Test design usually follows black box approach but for special cases, when black box test is not able to reach a source code, white box approach can be used instead. Test code coverage is regularly analyzed; test gaps are identified and covered with new tests. If unit tests are reviewed and accepted by the test team, they can become part of the regression testing.

3.3 Compilation and build check

Compilation and build check ensures that the software product and all its sample applications can be compiled at any moment without an error. The check runs on a daily basis and verifies that the compilation is not broken.

3.4 Pre-certification testing

For certain software components, standardized test frameworks are available on the market or in a public community to validate standard software features or to enable a pre-certification test execution on-site.

Pre-certification testing is performed for the MQX RTCS IPv6 communication stack with two public domain test frameworks (TAHI IPv6 Conformance test and IPv6 Interoperability test).

3.5 Out of box testing

Out of box (OOB) testing ensures that the software installer package and all sample and demo applications work as specified in the user documentation.

3.6 Field test

Field test is a validation test cycle where the customer facing teams evaluate the software installer package. Testers ensure that the released software and documentation is accurate, consistent, complete, and that it meets customers’ requirements. The main objective of field testing is to provide an initial setup experience and validate the ease of use of the software product. The secondary objective is to ensure that the package and documentation issues are not reintroduced. Additional benefit of field testing is that it provides the supporting teams with an early access to software.

3.7 Legal check

The legal check looks for a possible open-source-software (OSS) contamination. It protects the deliverables against a possible “copy and paste” code from the web, optimizes the use of open source, ensures compliance, discovers vulnerabilities, identifies bugs and corrects poor programming practices. All software deliverables are analyzed to uncover potential risks early in the development cycle and identify due diligence-related concerns well in advance.

4.0 Test schedule

4.1 Software development cycle

Software development cycle contains four main phases: product definition, product development, manufacture, and support as shown this figure.

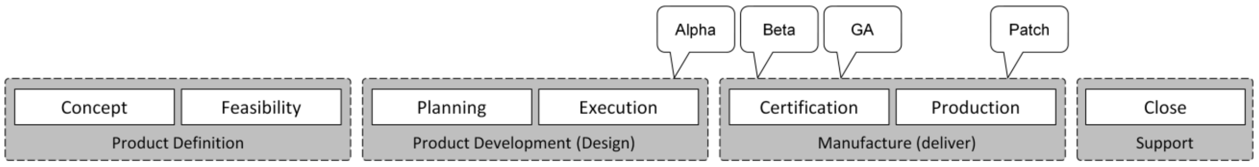

Figure 4-1. Software development cycle

A product definition phase consists of the concept and feasibility NPI phases where the product is conceptualized and its feasibility explored.

A product development phase consists of the planning and execution NPI phases. In planning, the product roll-out is prepared. In the execution phase, construction, validation, and release of the product occur.

A product manufacture and support phase involves maintenance of the product and ending production when the product has reached the end of life.

4.2 Software release development

There are four software releases such as Alpha, Beta, General Availability, and Patch releases.

The Alpha stage occurs at the beginning of every new software product version. During the Alpha stage, the developers focus on the new features or on the components development. Software can be unstable and could cause crashes or data loss. External availability of the Alpha software is uncommon. Alpha software release is often useful for demonstrations and previews within an organization. Some developers refer to this release as a preview, prototype, technical preview (TP), or early access release.

The Beta stage is the software phase which occurs after Alpha is released and after the software is feature-complete. The focus during Beta phase is to reduce the number of defects, stabilize, and optimize the software.

The General Availability (GA) stage occurs when the software product is mature for general customer availability and deemed sufficient to integrate into customers’ production products. The software package is released to the general market via freescale.com. GA software is stable, optimized, and bugs are either fixed or well documented.

A maintenance stage follows the GA release and involves discovering critical bugs, fixing them and delivering to customer as Patch releases.

4.3 Software deliverables types

The MQX SW is delivered as an installer package. The installer package includes all the source files, documentation, and examples. Apart from the main deliverable, there are two other types of deliverables such as standalone and add-on deliverable.

Standalone is a separate installer which can be installed without the main MQX release. Usually, it is used to support a single BSP released before the next MQX GA, e.g. MQX 4.0.1 TWRK20F100M BSP
Add-on is a separate installer and requires the main MQX release to be installed prior to the add-on. Add-on extends the MQX functionality, such as IPv6 support.

4.4 Software test cycle

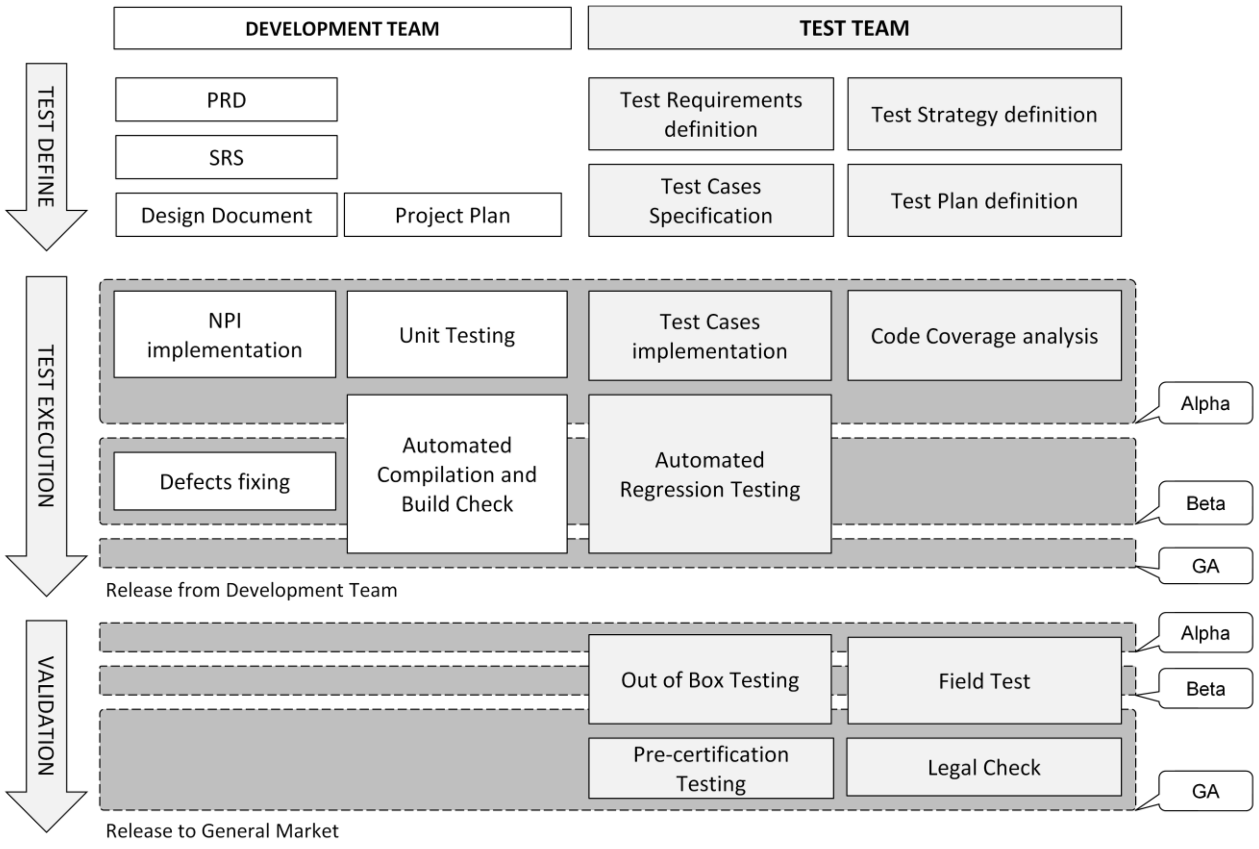

Figure 4-2. Software test cycle

The software test cycle is aligned with the software development cycle and can be described with three main phases: test definition, test execution, and validation. Test definition phase is aligned with the product definition phase and happens only once per software development cycle. Test execution and validation phase are repeated several times during the product development phase for every Alpha, Beta, GA, and Patch stage.

In the test definition phase, the test team focuses on defining the test strategy and the test plan to describe the future software testing. Test requirements are derived from the SRS and test cases are specified to cover the requirements.

In the test execution phase, all development and defect fixing is tested using the automated compilation and build check and regression testing for all stages. During the Alpha stage, the test team focuses on the new test cases development and validates the test implementation using the code coverage analysis and ensuring that the defined test-specification is met. Developers test the NPI implementation by using unit testing before their work is integrated.

In the validation phase, the test team receives the software installer package and validates the package by using out-of-box testing and field testing. In the GA stage, pre-certification testing and legal check is executed.

In the end of the every stage, test team checks that all exit criteria are met to ensure software release.

For post-GA stage, when critical defects are identified, compilation and build-check and regression testing is executed before the installer package is sent to out-of-box testing validating that the fix and the patch are released.

5.0 Test Exit Criteria

5.1 Defects clasification

All defects detected by test team or reported from customers are classified with a severity. The severity classification specifies the degree of the detrimental impact a defect has on the intended use of the product. Severity is used to identify important defects to estimate the quality of the product. These are the severity classifications:

1. Catastrophic (product unfit for use)
2. Serious (essential function degraded)
3. Moderate (apparent functional problem)
4. Minor (cosmetic or unapparent problem)
5. Trivial (transparent to customer/user)

If during regression testing and out-of-box testing a defect is found, it is reported with the appropriate severity classification. If any functionality which was working in the past is identified as broken, a regression defect is raised and catastrophic (S1) or serious (S2) classification is assigned based on the current development stage.

5.2 Exit criteria

There are the different exit criteria which are enforced depending on the type of the release:
Alpha:

1. All supported configurations must build.
2. Issues with severity S1 or S2 must be documented in the Release Notes and approved by marketing or the core team.

3. Documentation must clearly state the quality of the product.
4. Field test is optional.

Beta:

1. All supported configurations must build.
2. Issues with severity S1 must be closed.
3. Unresolved issues with severity S2 are approved by marketing and core team members.
4. Field test must pass including approval of the documentation.

GA:

1. All supported configurations must build.
2. All issues with severity S1 and S2 must be closed.
3. All unresolved issues with severity S3 must be approved by marketing and the core team members.
4. Field test must pass including the approval of the documentation.

6.0 Test team organization

From the organizational perspective, the test team is not a part of the development team and reports to the organization on the same level as the development team. Not being a part of the development team, enables the test team to define testing strategies and search for defects objectively, without the influence of the development team. Reporting to a senior management level allows the test team to effectively stop any upcoming release, until all issues are eliminated and release quality is met.
Even though test team is independent, it is always located next to development teams so that the test engineers can synchronize with the development engineers, review defects quickly, and communicate effectively.

7.0 Revision History

Version	**Date**	**Change Description**
1.0	01/2014	Initial version of the document.