Software Testing Guidelines

Company Proprietary

Copyright © 2017

All Rights Reserved

Template Version: 1.00

Table of Contents

[1 Document Objectives 4](#_Toc489015022)

[2 Intended Readers 4](#_Toc489015023)

[3 Software Testing Framework Overview 4](#_Toc489015024)

[4 Developer’s Perspective 5](#_Toc489015025)

[4.1 Unit Testing 5](#_Toc489015026)

[4.2 Component Integration Testing 6](#_Toc489015027)

[5 Tester’s Perspective 7](#_Toc489015028)

[5.1 Requirements Analysis 7](#_Toc489015029)

[5.2 Test Planning 7](#_Toc489015030)

[5.3 Test Case Development & Preparation 8](#_Toc489015031)

[5.4 Test Execution & Reporting 9](#_Toc489015032)

[5.5 Test Completion 11](#_Toc489015033)

[6 Business User’s Perspective 12](#_Toc489015034)

[6.1 User’s Acceptance Testing (UAT) 12](#_Toc489015035)

[7 Test Objectives vs. Stages Matrix 13](#_Toc489015036)

[8 Test Objectives vs. Test Techniques Matrix 14](#_Toc489015037)

[9 Defect Management 15](#_Toc489015038)

[9.1 Internal Defects Management Workflow 15](#_Toc489015039)

[9.2 External Defects Management Workflow 16](#_Toc489015040)

[9.3 Defect Log Record Structure 17](#_Toc489015041)

[10 Test Reports 23](#_Toc489015042)

[10.1 Test Execution 23](#_Toc489015043)

[10.1.1 Test Execution Progress 23](#_Toc489015044)

[10.1.2 Test Coverage Trend 24](#_Toc489015045)

[10.1.3 NotTested Distribution Trend 24](#_Toc489015046)

[10.1.4 Passed & Failed Ratios 25](#_Toc489015047)

[10.1.5 Defect By Severity 26](#_Toc489015048)

[10.1.6 Resolution Status 27](#_Toc489015049)

[10.2 Software Quality 28](#_Toc489015050)

[10.2.1 Defect Progress 28](#_Toc489015051)

[10.2.2 Defect Injected Distribution 28](#_Toc489015052)

[10.2.3 Defect Severity By Phase Injected 29](#_Toc489015053)

[10.2.4 Defect Distribution & Trend 29](#_Toc489015054)

[10.2.5 Defect Removal Efficiency 30](#_Toc489015055)

[10.2.6 Defect Severity 30](#_Toc489015056)

[10.2.7 Defect Severity by Phase Reported 31](#_Toc489015057)

[10.2.8 Defect Distribution by Modules 31](#_Toc489015058)

[11 Appendix – Test Objectives 32](#_Toc489015059)

[11.1 Acceptance/Qualification Testing 32](#_Toc489015060)

[11.2 Alpha & Beta Testing 32](#_Toc489015061)

[11.3 Configuration Testing (i.e. Compatibility, Portability) 32](#_Toc489015062)

[11.4 Functional Testing 32](#_Toc489015063)

[11.5 Interface Testing (i.e. Data Exchange) 33](#_Toc489015064)

[11.6 Installation Testing (Smoke, Sanity) 33](#_Toc489015065)

[11.7 Negative Testing (Error Handling) 34](#_Toc489015066)

[11.8 Parallel Testing (Back to Back) 34](#_Toc489015067)

[11.9 Performance Testing 34](#_Toc489015068)

[11.10 Recovery Testing 35](#_Toc489015069)

[11.11 Regression Testing 35](#_Toc489015070)

[11.12 Reliability Achievement & Evaluation 35](#_Toc489015071)

[11.13 Risk-Based Testing 35](#_Toc489015072)

[11.14 Security Testing 36](#_Toc489015073)

[11.15 Stress Testing (Load, Scalability, Volume) 36](#_Toc489015074)

[11.16 Usability &Human Computer Interaction Testing 36](#_Toc489015075)

[12 Definition of Terms 38](#_Toc489015076)

[13 Revision History 38](#_Toc489015077)

# Document Objectives

* To provide a testing framework & methodology for developers, testers (i.e. in-house tester and/or independent test service provider) and, business users in planning, preparation, execution, and reporting software testing activities under a project development environment.

# Intended Readers

* Project Managers
* Developers
* In-house Testers
* Test Service Providers
* Business Users

# Software Testing Framework Overview

Software testing consists of the dynamic verification that a software program provides expected behaviors on a finite set of test cases, suitably selected from the usually infinite execution domain. Dynamic means that testing always implies executing the software program on selected inputs.

Software Testing is performed by various project roles at varying incremental stages/phases. The table below shows us the project roles typical mapping against the different software testing stages/phases.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Project Roles | Unit Testing | Component Integration Testing | Quality Control (Analysis, Planning, Preparation & Testing) | | User’s Acceptance Testing |
| **System**  **Testing** | **System Integration Testing** |
| Developers | √ | √ |  |  |  |
| In-house Tester  and/or  Test Service Provider |  |  | √ | √ | √ |
| Business Users |  |  |  |  | √ |

# Developer’s Perspective

The Development Team, represented by developers is responsible for development, enhancements, and maintenance of systems applications. A part from creating the solutions to meet business needs and requirements, the Development Team is also responsible for the ensuring that the created solutions are verified against requirements and validated by the business users. The software product goes through software testing stages incorporated into development life-cycle process. This section describes the basic stages/levels of software testing before the software work products are released outside the development team.

## Unit Testing

**Objective**

The Unit Testing, also known as Component Testing is the narrowest scope of testing that must be performed by the developer. Each unit test concentrates on a single program, form, report, flex field, database trigger, or other custom object. When performed thoroughly, Unit Testing is one of the biggest contributors to a stable application system and will significantly reduce all downstream testing efforts. Unit Testing is a repetitive task; developers execute each Unit Test numerous times using different combinations of test data as specified in the data profile.

The purpose of the Unit Testing is to find defects in the smallest unit of software before it is logically linked to larger units. As such, Unit Test concentrates on testing application extension components on an individual basis to verify that the inputs, outputs, and processing logic of each application extension component functions without defects.

**Readiness Criteria & Input**

|  |  |
| --- | --- |
| READINESS CRITERIA | INPUTS |
| Completed Unit Source Code | Source Codes |
| Baselined V&V Strategic Matrix | V&V Strategic Matrix |
|  |  |

**Completion Criteria & Output**

|  |  |
| --- | --- |
| COMPLETION CRITERIA | OUTPUTS |
| Corrected Source Codes | Source Codes |
| All defects have been closed | Unit Test Cases & Results |
|  |  |

## Component Integration Testing

**Objective**

Component Integration Testing is the process of verifying the interactions among software components. Classical integration testing strategies, such as top-down and bottom-up are usually performed based on the hierarchical structure of the software being built.

**Readiness Criteria & Input**

| READINESS CRITERIA | INPUTS |
| --- | --- |
| Baselined V&V Strategic Matrix | V&V Strategic Matrix |
| Source Codes (raw & run-time format) |
| Unit Test Cases & Results |

**Completion Criteria & Output**

|  |  |
| --- | --- |
| COMPLETION CRITERIA | OUTPUTS |
| Corrected Source Codes & Copied to a Controlled Repository | Source Codes (raw and run-time format) |
| All defects have been closed | CI Test Cases & Results |
| Completed Compiling Scripts & Installation Scripts | Compiling Scripts, Installation Scripts |
| Submitted Hand-over Information | Hand-over Information |

# Tester’s Perspective

## Requirements Analysis

**Objective**

Requirements Analysis stage aims to consolidate all requirements/specification documents to be used for testing. This stage/phase will be used by the testers to gain knowledge on the software to be tested according to the requirements/specification document. This will also serve as the initial stage/phase where the testers can highlight and/or raise inconsistencies, and/or clarifications to gain understanding of the requirements. Requirement also intends to itemize and group the requirements so this can be easily tracked and controlled before performing software test related activities.

**Readiness Criteria & Input**

|  |  |
| --- | --- |
| READINESS CRITERIA | INPUTS |
| Collected Business Requirements Document | Business Requirements Document |
|  |  |

**Completion Criteria & Output**

|  |  |
| --- | --- |
| COMPLETION CRITERIA | OUTPUTS |
| Approved Requirements List | Requirements List |
|  |  |

## Test Planning

**Objective**

Test Planning stage aims to define the over-all test plans that will guide the quality control activities and the basis for monitoring these activities. The test plans may include but not limited to the following items:

* V&V Strategic Matrix
* Test Estimates
* Test Schedule
* Master Test Plan

The Test Manager is responsible for defining and integrating all the relevant plans while the test team members are primarily responsible for providing the test estimates which forms the basis of test scheduling and work assignments.

**Readiness Criteria & Input**

|  |  |
| --- | --- |
| READINESS CRITERIA | INPUTS |
| Drafted Requirements List | Requirements List |
|  |  |

**Completion Criteria & Output**

|  |  |
| --- | --- |
| COMPLETION CRITERIA | OUTPUTS |
| Submitted V&V Strategic Matrix | V&V Strategic Matrix |
| Baselined Master Test Plan | Master Test Plan |
| Baselined Test Estimates & Schedule | Test Estimates, Test Schedule |
|  | Project Repository, Project Roster |

## Test Case Development & Preparation

**Objective**

The Test Case Development stage aims to define and develop the test cases needed to verify and validate if the test cases are traceable to the system requirements and satisfy the system requirements during test execution. This phase also include activities to ensure the testing environment and systems are installed, configured, available, and verified against specifications before executing the test cases.

**Readiness Criteria & Input**

|  |  |
| --- | --- |
| READINESS CRITERIA | INPUTS |
| Baselined Requirements List | Requirements List |
| Baselined Test Estimates & Schedule | Test Estimates & Schedule |
| Drafted V&V Strategic Matrix | V&V Strategic Matrix |
|  |  |

**Completion Criteria & Output**

| COMPLETION CRITERIA | OUTPUTS |
| --- | --- |
| Baselined QC Test Cases & Results | QC Test Cases & Results |
| Allocated & Configured Test Environment | Test Environment |
| Baselined UAT Cases & Results | UAT Cases & Results |

## Test Execution & Reporting

**Objective**

Test Execution & Reporting stage aims to execute the target system guided by the test cases to ensure the system requirements are satisfied. The results of the testing are reported to relevant stakeholders to communicate the quality of the software and the test activity progress thru a clearly defined metrics.

The Quality Control testing can be sub-divided into three (3) major testing stages namely;

* Installation, Smoke and/or Sanity Testing
* System Testing
* System Integration Testing

**Installation, Smoke and/or Sanity Testing**

The first stage of quality control testing is to perform installation, sanity and/or smoke testing. The objective of performing the test initially is to ensure the correct environment has been verified; the system being is adequately verified as stable (i.e. Sanity and/or Smoke testing) to ensure continuity of performing the test cycles before under-going a more rigorous functional/non-functional and regression testing.

Smoke testing is applied to initial software builds when the software is relatively unstable while Sanity testing is performed for relatively stable builds after multiple rounds of regression tests.

**System Testing**

System Testing is concerned with testing the behavior of an entire system being built. System Testing is usually considered appropriate for assessing the non-functional system requirements (e.g. performance, stress, security, etc.). Though functional system requirements verification can be included in this stage, it is more of test scenarios that were not covered prior to this stage and/or it can only be done upon completion and availability of the entire system.

In System testing, the test environment should correspond to the final target or production environment as near as possible in order to minimize the risk of environment-specific failures not being detected.

**System Integration Testing**

In some situations, the target system needs to communicate/interface with other external systems (i.e. outside the target system). The verification process needs to perform System Integration Testing to ensure the data exchange/interface requirements are satisfied. Not all projects are required to perform System Integration Testing especially for stand-alone system.

**Readiness Criteria & Input**

|  |  |
| --- | --- |
| READINESS CRITERIA | INPUTS |
| Baselined QC Test Cases & Results | QC Test Cases & Results |
| Allocated & Configured Test Environment | Test Environment |
| Submitted Unit Test Cases & Results | Unit Test Cases & Results |
| Submitted CI Test Cases & Results, if applicable | CI Test Cases & Results |
| Submitted Hand-over Information | Hand-over Information |
| Corrected Source Codes & Copied to a Controlled Repository | Source Codes (raw format) |
| Completed Compiling Scripts & Installation Scripts | Compiling Scripts, Installation Scripts |

**Completion Criteria & Output**

| COMPLETION CRITERIA | OUTPUTS |
| --- | --- |
| Corrected Source Codes & Copied to a Controlled Repository | Source Codes (raw & run-time format) |
| Corrected Compiling/Installation Scripts & Copied to a Controlled Repository | Compiling Scripts, Installation Scripts |
| All defects have been closed | QC Test Cases & Results |
| Defect Log |
|  | Handover Information |

## Test Completion

**Objective**

The Test Completion stage aims to provide an over-all assessment report on the quality of the software as well as the quality of testing activities that have been performed and completed. The report also includes risk/issues based on the results, lessons learned, and best practices. This stage also initiate to officially release and endorse the system for UAT testing and/or deployment in the production environment.

**Readiness Criteria & Input**

|  |  |
| --- | --- |
| READINESS CRITERIA | INPUTS |
| All planned test cases has been executed | QC Test Cases & Results |
| All defects have been closed | Defect Log |
|  |  |

**Completion Criteria & Output**

| COMPLETION CRITERIA | OUTPUTS |
| --- | --- |
| Submitted Quality Report | Quality Report |
| Submitted Release Bulletin | Release Bulletin |
| E-mail UAT Endorsement | UAT Endorsement |
|  |  |

# Business User’s Perspective

## User’s Acceptance Testing (UAT)

**Objective**

User’s Acceptance testing is the prime responsibility of the business users of the system and may include other stakeholders (e.g. developers, testers, other business unit, etc.) to support the acceptance testing activities. There are some instances that the in-house testers and/or Test Service Providers may represent in behalf of the business users. The goal of acceptance testing is to establish confidence in the function of the system, in whole or in parts and may include specific non-functional requirements that are critical in the operations and use of the system. Completion of User’s Acceptance testing evaluates the system’s readiness for deployment and production use.

**Readiness Criteria & Input**

|  |  |
| --- | --- |
| READINESS CRITERIA | INPUTS |
| Baselined UAT Test Cases & Results | UAT Test Cases & Results |
| Submitted Hand-over Information | Hand-over Information |
| Submitted UAT Endorsement | UAT Endorsement |
| All previous defects have been closed | QC Test Cases & Results |
| Defect Log |
|  | Quality Report |

**Completion Criteria & Output**

| COMPLETION CRITERIA | OUTPUTS |
| --- | --- |
| Acknowledged UAT Acceptance Memo | UAT Acceptance Memo |
| All reported defects have been closed | Defect Log |
|  | UAT Cases & Results |
|  | Hand-over Information |
|  | Compiling Scripts, Installation Scripts |
|  | Source Codes (raw & run-time format) |

# Test Objectives vs. Stages Matrix

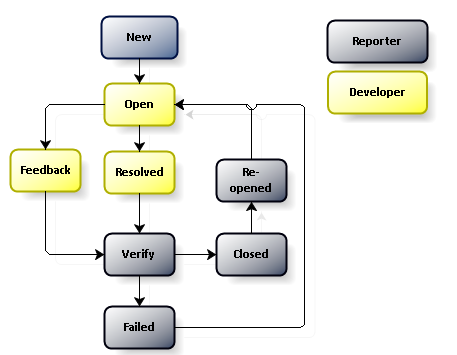
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Objectives | Design/  Prototype | Unit Testing | CI Testing | Quality Control | | UAT Testing |
| **System Testing** | **System Integration Testing** |
| Acceptance/Qualification Testing |  | √ | √ | √ | √ | √ |
| Alpha & Beta Testing |  |  |  | √ | √ |  |
| Configuration Testing  (i.e. Compatibility, Portability) | √ |  |  | √ |  | √ |
| Functional Testing | √ | √ | √ | √ | √ | √ |
| Interface Testing |  |  | √ | √ | √ | √ |
| Installation Testing (Smoke, Sanity) |  |  | √ | √ | √ | √ |
| Negative Testing |  | √ | √ | √ |  | √ |
| Parallel Testing (Back to Back) |  |  |  | √ | √ | √ |
| Performance Testing | √ |  |  | √ | √ | √ |
| Recovery Testing |  |  | √ | √ | √ | √ |
| Regression Testing |  | √ | √ | √ | √ | √ |
| Reliability Achievement & Evaluation |  |  |  | √ |  | √ |
| Risk-Based Testing |  |  |  | √ | √ | √ |
| Security Testing | √ | √ | √ | √ | √ | √ |
| Stress Testing | √ |  |  | √ |  |  |
| Usability & Human Computer Interaction Testing | √ |  |  | √ |  | √ |

# Test Objectives vs. Test Techniques Matrix

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Test Techniques | | | | | |
| Test Objectives | Experience | Input-Domain | Code Based | Fault Based | Usage Base | Model Based |
| Acceptance/Qualification Testing |  | EP, BVA,  Random |  |  | Operational Profile | Decision Table,  Workflow |
| Alpha &Beta Testing | Adhoc,  Exploratory |  |  |  | Operational Profile |  |
| Configuration Testing (i.e. Compatibility, Portability) |  | EP,  Pairwise,  Random |  |  |  |  |
| Functional Testing |  | EP, BVA, Pairwise  Random |  | Error Guessing | Operational Profile | Decision Table,  Workflow |
| Interface Testing |  | EP, BVA |  | Error Guessing |  |  |
| Installation Testing (Smoke, Sanity) | Exploratory | Random |  | Error Guessing | Operational Profile |  |
| Negative Testing | Adhoc | EP, BVA | Control Flow,  Data Flow | Error Guessing |  |  |
| Parallel Testing (Back to Back) |  | EP, BVA, Pairwise  Random |  | Error Guessing | Operational Profile | Decision Table,  Workflow |
| Performance Testing |  | EP, BVA |  |  | Operational Profile |  |
| Recovery Testing |  | EP, BVA | Control Flow,  Data Flow |  | Operational Profile | State Transition |
| Regression Testing |  | EP, BVA, Pairwise  Random | Control Flow,  Data Flow | Error Guessing | Operational Profile | Decision Table,  Workflow |
| Reliability Achievement & Evaluation |  |  |  |  | Operational Profile |  |
| Risk-Based Testing |  | EP, BVA, Pairwise  Random |  | Error Guessing | Operational Profile | Decision Table,  Workflow |
| Security Testing |  | EP, BVA,  Random | Control Flow,  Data Flow |  | Operational Profile |  |
| Stress Testing |  | EP, BVA |  |  | Operational Profile |  |
| Usability & Human Computer Interaction Testing |  |  |  |  | Operational Profile | Workflow |

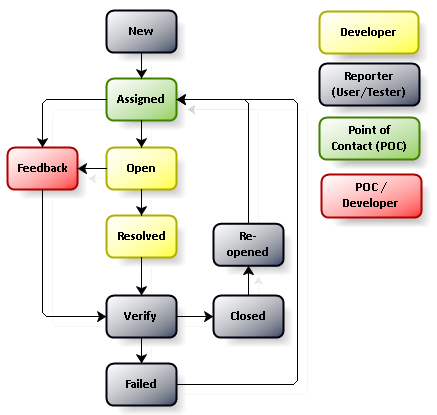
# Defect Management

## Internal Defects Management Workflow



| Status | Description | Performed By |
| --- | --- | --- |
| New | Once a defect is detected, it will be recorded in the defect log and the initial status of the defect is marked as New. | Reporter |
| Open | The developer can start to analyze and resolve the defect. | Developer |
| Feedback | If the defect is evaluated as invalid, the defect is marked as feedback, remarks are provided, and assigned back to the reporter for clarifications and further discussions. | Developer |
| Resolved | The developer modifies the work product (e.g. code, document, etc.) and verifies if the defect has been removed. Once verified, the defect record is marked as resolved. | Developer |
| Verify | The reporter verifies if the changes made on the work product resolves the defect. | Reporter |
| Failed | If the defect re-occurs during the retest, the defect is assigned back to the developer for resolution. | Reporter |
| Closed | The defect is marked as closed, once the final reason for closure has been determined. The possible reason for closure are as follows:  Complied  Duplicate  Deferred  Cannot Simulate  Solution Constraints  Not A Defect | Reporter |
| Re-opened | A closed defect can be re-opened if the defect recurred that was previously resolved in the previous test/review event. The defect is re-opened to go through the defect life-cycle workflow again. | Reporter |

## External Defects Management Workflow



| Status | Description | Performed By |
| --- | --- | --- |
| New | Once a defect is detected, it will be recorded in the defect log and the initial status of the defect is marked as New. | Reporter (User/Tester) |
| Assigned | The recorded defects is analyzed and assigned to the appropriate developer. | Point of Contact (POC) |
| Feedback | If the defect is evaluated as invalid, the defect is marked as feedback, remarks are provided, and assigned back to the reporter for clarifications and further discussions. | POC/Developer |
| Open | The developer can start to analyze and resolve the defect. | Developer |
| Resolved | The developer modifies the work product (e.g. code, document, etc.) and verifies if the defect has been removed. Once verified, the defect record is marked as resolved. | Developer |
| Verify | The reporter verifies if the changes made on the work product resolves the defect. | Reporter (User/Tester) |
| Failed | If the defect re-occurs during the retest, the defect is assigned back to the developer for resolution. | Reporter (User/Tester) |
| Closed | The defect is marked as closed, once the final reason for closure has been determined. The possible reason for closure are as follows:  Complied  Duplicate  Deferred  Cannot Simulate  Solution Constraints  Not A Defect  Recurred | Reporter (User/Tester) |
| Re-opened | A closed defect can be re-opened if the defect recurred that was previously resolved in the previous test/review event. The defect is re-opened to go through the defect life-cycle workflow again. | Reporter (User/Tester) |

## Defect Log Record Structure

| Field Name | Description | Sample Information |
| --- | --- | --- |
| Project | The name of the project. | Online Banking |
| Tracker | The name of the tracker where the bug was recorded. The expected values are as follows:  Internal – defects are reported within the development team  External – defects are reported outside the development team | Internal |
| Private | Set the bug to private making it visible to members of the project only. | No |
| Defect ID | Unique identifier automatically assigned to the defect. | 12345 |
| Subject | Summary of the defect. | Enrollment of new employee failed upon click of save button |
| Description | Detailed description of the defect. | Expected: Enrollment of new employee will be recorded in database upon click of save button Actual: Enrollment page crashed upon click of save button  1) Log in to SQME Payroll Software 2) Navigate to Payroll Page 3) Select Enrollment 4) Fulfill the required fields 5) Click save button |
| Raised By | The name of the person who reported the defect. | J. San Miguel |
| Status | The current status of the defect. The expected values are as follows:  New  Assigned  Open  Feedback  Resolved  Verify  Failed  Closed  Re-opened  Refer to the Defect Management Workflow section for the definition of the above status. | New |
| Priority | Defines the order on which bug should be resolved first. Acceptable values are as follows:  Immediate – the defect must be resolved as soon as possible because it is affecting the application or the product severely. The system or the component cannot be used until the defect is fixed.  Normal – the defect should be resolved in the normal course of development activities. It can wait until a new build or version is created.  Low – the defect is an irritant that should be repaired but can be deferred until after a more serious defect has been fixed. | Normal |
| Assignee | The name of the person that the defect is currently assigned. | J. Capsa |
| Category | The application name where the defect can be categorized. | Application 1 |
| Severity | The extent to which the defect affects the system. Acceptable values are as follows:  Critical – the defect results in termination of system or one or more component. The failed function is unusable and there is no acceptable alternative method.  Major – the defect results in termination of the system or one or more component. The failed function is unusable but there exists an acceptable alternative method.  Moderate – the defect does not result in termination but causes the system to produce incorrect, incomplete or inconsistent results.  Minor – the defect does not result in termination and does not damage the usability of the system, the desired results can be easily obtained by working around it and/or is related to the look and feel of the application. | Moderate |
| Phase Reported | Phase where the defect was detected. Expected values are as follows:  Unit Testing CI Testing Quality Control Testing User’s Acceptance Testing | Integration Testing |
| Phase Injected | Phase where the defect was introduced. Expected values are as follows:  Requirements Design Construction | Requirements |
| Verification Method | Type of verification activity where the defect was detected. Expected values are as follows:  Testing – Dynamic testing  Reviews – Static testing of document through inspections and reviews | Testing |
| Actual Session ID | The session id where the actual defect is detected based on test/review event (i.e. test cycle & test case number, review cycle, etc.) information. | Test Cycle A – Test Case 002 |
| Defect Category | The source format/media category where the defect is detected. Expected values are as follows:  Software Hardware Documents | Software |
| Defect Type | Requirements Defects:  R010 – Completeness;  R020 – Correctness;  R030 - Non-Functionality; R040 – Traceability  Design Defects:  D010 – Format;  D020 - Missing/Incorrect Data;  D030 - Misplaced data; D040 - Missing reference;  D050 - Incorrect reference;  D060 – Standards; D070 - Requirement Error;  D080 - Design Error;  D090 - Interface Error  Construction Defects:  C010 - Data Reference;  C020 - Data Declaration;  C030 – Computational; C040 – Comparison;  C050 - Control Flow;  C060 – Interface;  C070 – Input/Output;  C090 - Build, Package;  C090 – Environment;  C100 – Documentation;  C110 – Portability;  C120 – Reusability;  C130 – Performance;  C140 - Coding Standards | D090 - Interface Error |
| Start Date | The date when the bug was logged. | 05/20/17 |
| Work Product | The module where the defect was detected. | Module X |
| Source File | Source file name where the defect was detected. | Login.php |
| Findings | Specify the result of investigation of the assigned developer. Valid values are as follows:  Defective – if the reported defect is confirmed as defective  Not Defective – if the reported defect is not considered as defective | Defective |
| Reason | Reason for closure or re-opening the defect. Valid values are as follows:  Complied – satisfy the expected condition  Duplicate – duplicate defect. Previous related defect is used to track the defect  Deferred – resolution of defect is suspended  Cannot Simulate – Cannot reproduce the defect  Solution Constraints – defect cannot be fixed due to inherent solution constraints  Not A Defect – behavior is not considered as defect  Recurred –the defect have re-occurred | Complied |
| Recurrence# | The number of times the defect has recurred after it has been closed or failed a retest. Increment value starting from 1 if the defect was detected after it has been closed or has failed a retest. | 1 |
| Plan Verification Session | The plan session id where the defect can be verified as resolved based on test/review event (i.e. test cycle & test case number, review cycle, etc.) information. | Test Cycle C – Test Case 002 |
| Last Status Update | Date of the last status update. | 05/24/17 |
| Remarks | Any additional comments on the defect. | 5/25/2017 - Retest Failed, issue not fixed. |

# Test Reports

## Test Execution

Test Execution Reports provides metrics information related in executing the test activities. Below are some of the sample reports.

### Test Execution Progress

The Test Execution Progress Report displays the percentage of cumulative test cases executed against the planned total test cases across time by periods.

### Test Coverage Trend

The Test Coverage Trend report shows the progress of executed test cases against planned test cases across test cycles.

### NotTested Distribution Trend

The NotTested Distribution Trend report shows the progress of planned test cases that not been tested across the test cycles due to the following reasons namely; a) test item cannot be verified due to dependencies on other test cases that have pending issues, and b) test item pending for test execution.

### Passed & Failed Ratios

The Passed & Failed Ratios report shows the percentages of passed and failed test cases against total test cases executed. The report shows both over-all passed & failed ratios and passed & failed ratios trend across test cycles.

### Defect By Severity

The Defect By Severity report shows the severity distribution of executed test cases. The report shows both the over-all defect by severity percentage distribution and defect by severity trend (i.e. frequency count) across test cycles.

### Resolution Status

The Resolution Status report shows the frequency distribution (i.e. count and percentage) of the status defects in terms of state of resolution (i.e. open, re-open, resolved, cancelled, and deferred). The report shows both the over-all resolution status percentage distribution and resolution trend (i.e. frequency count) across test cycles.

## Software Quality

The Software Quality Reports provides metric information regarding the degree of quality of the software. The primary measure of software quality is through the defect counts classified by various attributes (e.g. by time periods, by severity, by modules, etc.). Below are some of the sample reports.

### Defect Progress

The Defect Progress report shows us the cumulative active defects versus the cumulative closed defect by periods.

### Defect Injected Distribution

The Defect Injected Distribution report shows the percentage defect distribution and frequency count of defect that was injected in the early phases of the project life-cycle.

### Defect Severity By Phase Injected

The Defect Severity By Phase Injected report shows the frequency count distribution of defect severity (i.e. critical, major, moderate, minor) that was injected in the early phases of the project life-cycle.

### Defect Distribution & Trend

The Defect Distribution & Trend report shows percentage distribution of defects detected and reported at various test stages performed at the later stage of the project life-cycle as displayed in the bar chart. The report also shows the defect trend which is the cumulative defects detected & reported at end of each major testing stage as displayed in the line chart.

### Defect Removal Efficiency

The Defect Removal Efficiency report shows how efficient are we detecting and resolving the defects at the completion of each of the major testing stages before the UAT stage.

### Defect Severity

The Defect Severity report shows the over-all percentage distribution (i.e. shown as bar chart) and frequency count (i.e. shown as line chart) of defects categorized by severity (i.e. critical, major, moderate, and minor).

### Defect Severity by Phase Reported

The Defect Severity by Phase Reported report shows the frequency count distribution of defect detected and reported at various test stages categorized by severity.

### Defect Distribution by Modules

The Defect Distribution by Modules report is an example of Pareto chart that shows us the defect count by modules sorted by highest to lowest values as shown by the bar chart. The line chart shows us the percentage cumulative defects against the total defects reported. The report highlights the most common sources of defects (i.e. by modules).

# Appendix – Test Objectives

## Acceptance/Qualification Testing

Acceptance / qualification testing determines whether a system satisfies its acceptance criteria, usually by checking desired system behaviors against the customer’s requirements. The customer or a customer’s representative thus specifies or directly undertakes activities to check that their requirements have been met.

The variants of acceptance/qualification testing can be associated to:

* Contract Acceptance Testing – It is performed against the contract’s acceptance criteria for producing custom developed software. Acceptance should be formally defined when the contract is agreed.
* Operational Acceptance Testing - Also known as production acceptance test that validates whether the system meets the requirements for operation.
* Confirmation Testing - When a test fails because of a defect, a new version of the software is expected to fix the defect. In this case we need to execute the test again to confirm that the defect was actually fixed.

## Alpha & Beta Testing

Before software is released, it is sometimes given to a small, selected group of potential users for trial use (alpha testing) and/or to a larger set of representative users (beta testing). These users report problems with the product. Alpha and beta testing are often uncontrolled and are not always referred to in a test plan.

## Configuration Testing (i.e. Compatibility, Portability)

In cases where software is built to serve different users, configuration testing verifies the software under different specified configurations.

The variant of configuration testing can be associated to:

* Compatibility Testing - In compatibility testing, the application is being checked on how well it performs in a particular hardware, software, operating system, and/or network environment and different combinations of above.
* Portability Testing - It refers to the process of testing the ease in which a computer software component or application can be moved from one environment to another(e.g. moving of any application from Windows 2000 to Windows 10). This is usually measured in terms of the maximum amount of effort permitted. Results are measured in terms of the time required to move the software and be able to run the software into the new platform/environment.

## Functional Testing

Functional testing is a software testing process used within software development in which software is tested to ensure that it conforms to all requirements. Functional testing is a way of checking software to ensure that it has all the required functionality that's specified within its functional requirements.

For Example:

| Test Specification | Test Case |
| --- | --- |
| The system should prompt for passbook printing when the user execute a “with book transaction”. | The user enters any savings account transactions with the with-book parameter enabled.   * The system should prompt for passbook printing   The user enters any savings account with the no-book parameter enabled.   * The system should not prompt for passbook printing |

## Interface Testing (Data Exchange)

Interface defects are common in complex systems. Interface testing aims at verifying whether the components interface correctly to provide the correct exchange of data and control information. Usually the test cases are generated from the interface specification. A specific objective of interface testing is to simulate the use of APIs by end-user applications. This involves the generation of parameters of the API calls, the setting of external environment conditions, and the definition of internal data that affect the API.

For Example:

|  |  |
| --- | --- |
| Test Specification | Test Case |
| The Cafeteria Ordering System (COS) totals should reconcile with the Payroll System totals. | The user retrieves the COS totals per user and compares the results of the generated totals per employee in the Payroll System.  Both reports should have the same values for each employee. |

## Installation Testing (Smoke, Sanity)

Often, after completion of the entire system, the software is verified upon installation in the target environment. Installation testing can be viewed as system testing conducted in the operational environment of hardware configurations and other operational constraints. Installation procedures may also be verified.

The variants of installation testing can be associated to:

* Smoke Testing - Smoke Testing is a kind of software testing performed after software build to ascertain that the critical functionalities of the program is working fine (e.g. application starts successfully). It is executed "before" any detailed functional or regression tests are executed on the software build. The test is applied to initial software builds when the software is relatively unstable.
* Sanity Testing –Sanity testing is a kind of software testing performed after receiving a software build, with minor changes in code, or functionality, to ascertain that the bugs have been fixed and no further issues are introduced due to these changes (e.g. verify new functionality, bug fixes in the build). The test is applied for relatively stable builds after multiple rounds of regression tests.

## Negative Testing (Error Handling)

Negative Testing is a variant of testing that can be performed on the system by providing invalid data as input. It checks whether an application behaves as expected with the negative input. This is to test the application that does not do anything that it is not supposed to do so.

For Example:

| Test Specification | Test Case |
| --- | --- |
| The system should display invalid account number when the check-digit is erroneously entered | Perform data entry on the following account numbers with erroneous check-digit numbers (i.e. x, y, z). The system should display invalid account numbers for wrong check-digit numbers. |

## Parallel Testing (Back to Back)

In parallel testing, the old and new system is being run simultaneously. Results are being used to detect unplanned differences between the two.

For Example:

|  |  |
| --- | --- |
| Test Specification | Test Case |
| All transaction posted in the Mainframe platform (i.e. old system) should tally with all transactions posted in the Client-Server platform (i.e. new system). | The user enters the transactions both in the Mainframe platform and Client-Server platform. The user generates the local journals, local totals, host journal and host totals of both systems and verifies that the reports tally with each other. |

## Performance Testing

Performance testing verifies that the software meets the specified performance requirements and assesses performance characteristics—for instance, capacity and response time.

For Example:

|  |  |
| --- | --- |
| Module | Test Case |
| Cash Deposit – PESO | Create 500 cash deposit transaction at the same time. Record the time needed to complete all transactions. Validate that all transactions were completed as expected. |

## Recovery Testing

Recovery testing is aimed at verifying software restart capabilities after a system crash or other “disaster.” Recovery testing initiates forced failure of the system and determine its behavior after a forced failure.

For Example:

|  |  |
| --- | --- |
| Module | Test Case |
| Cash Deposit – PESO | Perform a cash withdrawal transaction then turn off the connection that will process the transaction. Turn the connection; verify the behavior of the system on how it recover from connection failure. |

## Regression Testing

Regression testing is the “selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still complies with its specified requirements.” Typically, automation tools are used for this test type since a system is difficult to wholly cover in regression testing.

For Example:

|  |  |
| --- | --- |
| Module | Scenario |
| Core Banking | A defect was fixed in the cash deposit transaction.  Test Team would then need to do a regression testing of not just the cash deposit transaction but other related transactions having the same source file of the cash deposit transactions as well as transactions having dependencies with cash deposit transaction based on the impact of change make sure no other functions was unintentionally changed. |

## Reliability Achievement & Evaluation

Testing improves reliability by identifying and correcting faults. In addition, statistical measures of reliability can be derived by randomly generating test cases according to the operational profile of the software. The test can also be done the most commonly used modules/codes that were fixed due to defects.

## Risk-Based Testing

Risk based testing is basically a testing done for the project based on risks. Risk based testing uses risk to prioritize (i.e. based on impact and likelihood of occurrence) and emphasize the appropriate tests during test execution. It involves assessing the risk based on the complexity, business criticality, usage frequency, visible areas, defect prone areas, etc. Since there might not be sufficient time to test all functionality, Risk based testing involves testing the functionality which has the highest impact and probability of failure.

## Security Testing

Security testing is focused on the verification that the software is protected from external attacks. In particular, security testing verifies the confidentiality, integrity, and availability of the systems and its data. Usually, security testing includes verification against misuse and abuse of the software or system (negative testing).

For Example:

|  |  |  |
| --- | --- | --- |
| Module | Specifications | Test Case |
| System Login | The system will lock-out the user after failing to enter the correct password after three (3) re-tries | Perform login transaction and enter the incorrect password more than three (3) retries. |

## Stress Testing (Load, Scalability, Volume)

Stress testing exercises software at the maximum design load, as well as beyond it, with the goal of determining the behavioral limits, and to test defense mechanisms in critical systems.

The variants of stress testing can be associated to:

* Load Testing - Load testing is performed to determine a system’s behavior under both normal and at peak conditions. The primary goal of load testing is to define the maximum amount of work a system can handle without significant performance degradation
* Scalability Testing - Scalability testing tests the ability of a system, a network, or a process to continue to function well, when it is changed in size or volume in order to meet a growing need.
* Volume Testing - The purpose of volume testing is to determine system capacity and efficiency with increasing volumes of data.

For Example:

|  |  |  |
| --- | --- | --- |
| Module | Specifications | Test Case |
| Cash Deposit – PESO | The cash deposit module can handle 10,000 simultaneous transactions. | 1. Transact 10,001 simultaneous cash deposit transactions, observe the system behavior. 2. Transact 11,000 simultaneous cash deposit transactions, observe the system behavior. |

## Usability &Human Computer Interaction Testing

The main task of usability and human computer interaction testing is to evaluate how easy it is for end users to learn and to use the software. In general, it may involve testing the software functions that supports user tasks (i.e. Manual Support), documentation that aids users (i.e. Documentation Testing), and the ability of the system to recover from user errors

|  |  |
| --- | --- |
| Definition of Terms | |
| Terms | Description |
| [List the terms used in the document that needs to be defined.] |  |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Revision History | | | |
| Version No. | Date | Description of Changes | Author |
| 1.00 | 28-Jun-2017 | Initial Baseline | RBL |
|  |  |  |  |