

mAster test plan

**VERSION 1.0**

This template was created to enable departments to more easily develop their project plans. The Department of Technology, Consulting and Planning Division, created this template based on its experiences. The template relies on industry best practices combined with decades of experience on California state information technology projects. The way it was structured is to enable a department to complete the information related to its project without having to write background information related to the discipline. A department may use as much or as little of the template as it wishes.

**Template Instructions:**

* ***Instructions for completing*** this template – written for the author of the project plan - are encased in **[ ]** and the text is ***italicized*** *and* ***bolded.***
* *Examples* are provided as a guideline to the type of sample information presented in each section and the text is *italicized*.
* Boilerplatestandard language for each section is written in the document font and may be used or modified, as necessary.
* A department’s project specific information goes within the brackets ***<< >>***.
* *Informational text is italicized* within square brackets [ ] for informational purposes to the person who has to create the plan and includes background information, explanation, rationale, etc.

APPROVAL SIGNATURES

| CONTRACTOR | | DATE |
| --- | --- | --- |
| *<<Deliverable Owner>>*  *<<John Doe, Manager>>* | *<<Signature>>* |  |
| *<<Vendor Project Manager>>* | *<<Signature>>* |  |
| *<<Vendor Managing Director>>* | *<<Signature>>* |  |

| **STATE PROJECT MANAGER** | | | | |
| --- | --- | --- | --- | --- |
| *Date:* | *<<Date>>* | *To:* | ***<<***PM Name***>>,*** *Project Manager*  ***<<*** Test Manager’s Name ***>>,*** *Test Manager* | |
|  | *I approve this deliverable and have no further questions or comments.* | | | |
|  | *I approve this deliverable conditionally, contingent on the review and approval of the following corrections (see comments).* | | | |
|  | *I reject this deliverable for the following reasons identified (see comments).* | | | |
|  | | | | |
| *<<SIGNATURE>>* | | | | *<<DATE>>* |
| *<<SIGNATURE>>*, Test Manager | | | | *<<DATE>>* |
| Comments | | | | |
|  | | | | |

DOCUMENT HISTORY

| **DOCUMENT APPROVAL HISTORY** | |
| --- | --- |
| Prepared By |  |
| Reviewed By |  |
| Approved By | <<Typically the Project Sponsor>> |

| **DOCUMENT REVISION HISTORY** | | | |
| --- | --- | --- | --- |
| Date | Document Version | Revision Description | Author |
| 1/28/2014 | 1.0 | Initial Version | J. Fong |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

TABLE OF CONTENTS

1. OVERVIEW 1

1.1. Introduction 1

1.2. Project Description 2

1.3. Testing stakeholder communication 3

1.4. Reference Material 4

1.5. Glossary 5

2. TESTING CONTEXT 5

2.1. Test Item 5

2.2. Test Scope 7

2.2.1. Features/Functions to be Tested or Retested 8

2.2.2. Features/Functions not to be Tested 9

2.3. Dependencies, Assumptions, and Constraints 9

2.4. Software Risk 11

2.4.1. Product Risks 11

2.4.2. Project Risks 12

3. TEST STRATEGY 13

3.1. Test Design Approach 13

3.2. Verification Approach 14

3.2.1. Requirements Review 14

3.2.2. Use Case Review 16

3.2.3. Design Review 17

3.2.4. Code Review 19

3.2.5. Test Document Review (all types below) 20

3.3. Validation Approach 22

3.3.1. Unit Test 22

3.3.2. Integration test 23

3.3.3. System Test 25

3.3.4. User acceptance Test 29

3.3.5. Documentation Test Approach (if applicable) 31

3.4. Defect Management 33

3.5. Metrics and Reporting 34

3.6. Test Data Requirements 35

3.7. Test Environment AND Facilities Requirements 36

3.8. Test Tools 39

3.9. Automation Plan 40

3.10. Hardware Allocation & Procurement Plan 41

3.11. Suspension and Resumption Criteria 41

4. TEST MANAGEMENT 43

4.1. Administration 43

4.2. Approval Authority 44

5. STAFFING 45

5.1. Roles & Responsibilities 45

5.1.1. Configuration and Release management resources 46

5.1.2. Unit Test Resources 47

5.1.3. Integration test resources 48

5.1.4. System Test Resources 48

5.1.5. user Acceptance Test Resources 49

5.2. Test Team Training Requirements 51

6. PROJECT SCHEDULE AND TEST ARTIFACT REPOSITORY 51

APPENDIX A. COTs, MOTS, and Custom implementation A-1

APPENDIX B. Glossary B-1

[The Master Test Plan is a critical foundational test document that describes the key components needed for any testing effort regardless of project size and complexity. Although this Plan is extensive, it is important that the components are carefully considered for effective test planning and preparation. This Plan is scalable to allow the author to provide the relevant level of content detail to fit the project size and test complexity. Hence, test details may be less for a small project while much greater for a large project.

For example, a smaller project may only require a handful of staff to support testing efforts. On the other hand, a larger project may require a significant number of coordinated personnel to support the multitude of test environments, external interfaces, software, hardware, test execution, and test management activities necessary to validate the system under test. In any case, the test resources and staffing needs must be clearly articulated in the Master Test Plan regardless of project size and test complexity.]

# 

# OVERVIEW

The Master Test Plan identifies the scope, strategy, approach, and details of the testing phases including the testing schedule, resources, environment requirements, and mechanism for documenting and tracking test results. Development of the Master Test Plan should start simultaneously with project initiation. The Master Test Plan is the most critical test plan document, as it serves as a communication tool to ensure that key project team members and stakeholders are in alignment with the overall test effort and strategy.

Test plans exist for all levels of testing. For larger projects, separate test plans should be developed for testing levels such as: unit test, integration test, system test, and acceptance test.

## Introduction

The Master Test Plan is a test planning and management tool for facilitating the successful planning, controlling, and management for all testing aspects for <<Project Name >>.

The Master Test Plan serves many important purposes. It lays the groundwork for software testing and assures that all aspects of testing have been considered. Also, the Master Test Plan serves as a communication mechanism to all project stakeholders describing what is involved, what the risks are, and what resources are required to successfully test the system. Further, a Master Test Plan establishes the expectations for testing the system including what is in scope, what is out of scope, and what the objectives and deliverables are for the system under test. The Master Test Plan assists in the preparation of the testing processes required to ensure the highest level of project success.

[Provide the purpose of the Master Test Plan relevant to your project. Depending on the size of your project, separate test plans may need to be created to detail the specifics for the particular level of testing (e.g., Integration Test, System Test, and User Acceptance Test).]

## Project Description

This section describes the scope for << Project Name >>.

[Provide an overview of the Project. Items that may be included in this section include:

* Project background – Is there a business problem that the project is trying to solve?
* Purpose of the project – Why is the project being undertaken?
* What is the scope of the project? (e.g., There will be four phases: Phase 1 - Data Conversion, Phase 2 - Gap Analysis of “As-Is” and “To-Be” Systems, Phase 3 – Complete Request for Proposal to obtain vendor resources, Phase 4 - Development and implementation of project’s solution. This Master Test Plan addresses Phase 4….). A high level diagram illustrating the project scope and components may be inserted in this section.
* What are the business objectives for the project? What does success look like? What are the critical success factors and key performance indicators?
* Who are the stakeholders?
* What are the high level project phases and timelines?
* What business benefits will the program area achieve when the project is implemented?]

*Example:*

*In 2012, the XYZ Project delivered a successful prototype of the ABC Company’s SOA Infrastructure model. In 2013, the XYZ Project will develop Phase I of the production version of the ABC Company’s SOA Infrastructure, and it is planned to be the basis of all future ABC Company’s IT development. Phase I is scheduled to be developed, tested, and delivered by October 2014.*

*For the development of the production version of the SOA Infrastructure, testing will be the cornerstone and driving force that ensures the success of the XYZ Project by delivering a quality SOA product. In the past, IT has had difficulty implementing enhancements to the current system due to the inflexible architectural structure. With the new solution, the business will benefit from the infrastructure improvements in that they will now have the flexibility to easily incorporate improvements to the system due to legislative changes and new business opportunities. Overall, the business goal is to enhance the business services to our customers.*

## Testing stakeholder communication

Successful testing efforts require collaboration and timely communication. Identifying the correct testing stakeholders is important to timely decision making. It is important that all project team members including, but not limited to the development personnel, test teams, and users, fully understand the channels for communication especially as it pertains to test coordination and test activities. For example, a project’s communication plan may describe weekly status meetings to unify the project and test teams to review test status, relative to open incidents and outstanding defects, and discuss prioritizing plans for fixing the problems.

For any issues surrounding the resolution of disagreements concerning the satisfactory completion of entrance or exit criteria, << Project Name >> will handle any conflicts via the project’s escalation process. For example, the Test Manager may be responsible for initially resolving the issue with the test team. If the Test Manager remains unsuccessful, he/she will escalate the issue to the Project Manager, who will be responsible for arranging a subsequent discussion with the pertinent parties to resolve the issue.

[The testing methods of communication can be expanded in the project’s Communication Plan, if the document exists. Specify in this Testing Stakeholder Communication section the specific areas of the Communication Plan where the audience may reference the communication methods for testing. If the project does not have a Communication Plan, list the stakeholders and describe the methods of communication between each stakeholder as it relates to testing.

Example: The authority for resolving issues that were raised as a result of the testing activities and the authority for approving the test products and processes is indicated in the following table of Roles and Responsibilities. The communication may be represented visually via picture, diagram, etc.]

## Reference Material

The following reference materials were used in the development of the Master Test Plan for << Project Name >>.

* ANSI/IEEE Standard 1008-1987: Software Unit Testing
* IEEE 829-2008 – IEEE Standard for Software and System Test Documentation
* IEEE Standard 1012-2012: System and Software Verification and Validation
* ISO/IEC 16085-2006: System and Software Engineering – Lifecycle Processes – Risk Management
* IEEE Standard for Adoption of ISO/IEC 26513:2009 Systems and Software Engineering—Requirements for Testers and Reviewers of Documentation
* ISO/IEC/IEEE 29119-1:2013 Software and Systems Engineering – Software Testing Part 1: Concepts and Definitions
* ISO/IEC/IEEE 29119-2:2013 Software and Systems Engineering – Software Testing Part 2: Test Processes
* ISO/IEC/IEEE 29119-3:2013 Software and Systems Engineering – Software Testing Part 3: Test Documentation
* International Software Testing Qualification Board (ISTQB) – Standard glossary of terms used in Software Testing Version 2.3 ( March 28, 2014)
* Pierre Bourque and Richard E. (Dick) Fairley, Software Engineering Body of Knowledge (SWEBOK) Guide V3.0 (2014), 978-0-7695-5166-1
* Statewide Information Management Manual (SIMM) – California Project Management Methodology (CA-PMM)

[List only the reference materials that are used within this document. This may be in the form of project documents or links to the project documents. The reference listing allows the audience to refer to a specific document should the person want more detailed information about a subject. The references listed above represent the sources used in the development of the Master Test Plan Template. For your project, delete any reference that is not relevant.]

Example:

* *Project Authorization*
* *Project Management Plan*
* *Requirements*
* *System Architecture Documents*
* *Design*
* *Risk Assessment Methodology*
* *Quality Assurance Plan*
* *Configuration Management Plan*
* *User Guide*
* *Installation Guide*

## Glossary

The glossary section is designed to provide the terms and definitions used in this Master Test Plan. See Appendix B.

[Provide a project specific list and definition of terms, abbreviations, and acronyms, if any, used in the document. Note: This section may refer to another document providing a project-specific glossary.]

# TESTING CONTEXT

The Testing Context describes the boundaries for testing including what functions will be tested and what functions will not be tested. It is important that the project (i.e., business, state, and vendor) understands the scope of the system under test and aligns with expected results as defined in the business requirements.

## Test Item

Test items represent individual elements or work products that are the object of testing. Test items may be comprised of a component, system, a requirement document, design specification, or user guide to be tested. It is important that the test item be clearly defined to ensure that the project understands the scope and objectives of what will be tested, not tested, and the criteria for fulfilling the test item via a pass/fail criteria. Criteria are a set of conditions for permitting the acceptance (pass) or rejection (failure) of the test item.

[Describe the objective of what is to be tested within the scope of the Master Test Plan (e.g., testing for the New ADD/UPDATE function(s), software unit, interfaces between units, subsystem). If applicable, differentiate between primary and secondary objective. (e.g., A primary objective may be to reduce the time for manual regression testing. A secondary objective may be to create an automated test library by automating 20% of the manual test scripts).

Each test item has pass/fail criteria in terms of: test cases passed and failed, number, type, severity, and location of bugs, usability, reliability, and/or stability. Describe the pass/fail criteria for each test item in this section.]

| Test Item | Pass/Fail Criteria |
| --- | --- |
| 1. | Description |
| 2. | Description |
| 3. | Description |

Example:

For ABC Web Application

| Test Item | Pass/Fail Criteria |
| --- | --- |
| 1. *Manage ABC Registration* | * *Pass* * *No open severity 1 (critical) and/or severity 2 (major) defects.* * *Users can successfully enter data, perform data validation, and send completed transaction information to the department for processing.* * *Response rate for the enrollment screen is less than or equal to 2 seconds.* * *Fail* * *Response rate for enrollment screen is greater than 2 seconds.* |
| 1. *ABC Batch Processes* | * *Pass* * *No open severity 1 (critical) and/or severity 2 (major) defects.*   + *The batch systems successfully process multiple daily data files and update the ABC System with current business registration information.*   + *Batch processing time concludes within the time allotment window (e.g., 1 AM – 5 AM).*   + *Batch Transaction system updates are completed by the start of the next business day (7:00 AM).* * *Fail*   + *Batch processing exceeds time allotment window (e.g., 1 AM – 5 AM).*   + *Batch Transaction system updates are not completed by the start of the next business day (7:00 AM).* |
| 1. *ABC Database and System Synchronization* | * *Pass* * *No open severity 1 (critical) and/or severity 2 (major) defects.*   + *The ABC System updates ancillary databases and sub-systems with current business registration information. Users are able to view the processed and updated business registration information in System 123 and System 456.* * *Fail* * *Mirror database and system synchronization do not accurately reflect production system data.* |

## Test Scope

It is important to define the scope of testing to ensure that planning is completed for all aspects of the functionality that will be tested and that the entire project team knows what will not be tested.

[Describe the overall scope of the project testing effort by identifying those features (e.g., system functions, software, documentation, hardware or other system components) which "will" and "will not" be tested. The list of features is intended to clearly specify the test boundaries for the project.]

The following section lists the features of the test item(s) to be tested.

### Features/Functions to be Tested or Retested

Features to be tested may be comprised of specific attributes of specific software, functions, interfaces, or business process. For << Project Name >>, the features/functions to be tested or retested include:

[List those features/functions that will be tested/re-tested during this project.]

Examples of system documentation that may assist a project in the definition of the features/functions to test or retest include, but are not limited to:

* Business process workflows
* State transition diagrams
* Requirements (functional and non-functional) documents
* Use cases
* Business rules
* User interface screens
* Interface documents
* Batch processes
* Architectural diagrams
* Design specifications

Additionally, a project may consider the core business functions that support operations including frequency and impact. A business activity or task with high frequency or high impact would likely be a strong candidate to be a feature/function to be tested and retested. For example, in the XYZ health application, a technician processes over 1000 transactions per day for adding a member to a health plan. The technician has a daily quota in order to meet business targets. The impact is high to the organization if the processed transactions fall short of the daily expectations. In this case, adding a member to a health plan would easily constitute a critical feature/function to be tested. Also, the volume of transactions processed would assist the test team in prioritizing the most important features of the system and which feature to test first or to more thoroughly test.

Example:

*Features/Functions that will be tested during this project include:*

* *Member enrollment into a health plan.*
* *Adding a member to a health plan.*
* *Modifying a member’s existing health plan.*
* *Deleting a member from an existing health plan.*

### Features/Functions not to be Tested

As with features/functions to be tested, it is important that the project identifies the features/functions that will not be tested or features that are outside the scope of testing. For << Project Name >>, the features/functions that will not to be tested or retested include:

[List the features/functions that will not be tested during this project. Provide the rationale for excluding the features/functions from testing.]

Example:

*Features/Functions that will not be tested during this project include:*

* *Enrollment of the member’s dependents into a health plan will not be tested. The system pertains to the enrollment of members only. Phase 2 of the project will cover the enrollment of the member’s dependents into a health plan.*
* *Payment of a member’s health premium when enrolling in a health plan is handled in an ancillary system and outside the scope of the project.*
* *Issuance of a member’s medical health card upon enrollment into a health plan is outside the scope of the project.*

## Dependencies, Assumptions, and Constraints

The identification of test dependencies, assumptions, and constraints allows the project to proactively plan and effectively manage testing efforts across the software lifecycle. The timely management of the dependencies, assumptions, and constraints is important to minimizing any potential issues from escalating into more serious problems which could negatively impact the project’s success.

[Describe any test dependencies, assumptions, and constraints for the test efforts covered by this plan. These may include regulatory standards, the requirements in the test policy or the organization’s test strategy, contractual requirements, project time and cost constraints, and availability of appropriately-skilled staff, tools, and/or environments.

***[List any test dependencies, if any, for your project. Include the owner, due date, and status.]***

|  |  |  |  |
| --- | --- | --- | --- |
| Dependencies | Owner | Due Date | Status |
| 1. |  |  |  |
| 2. |  |  |  |

*Example:*

| Dependencies | Owner | Due Date | Status |
| --- | --- | --- | --- |
| 1. *Project is dependent on the current system application changes to support SBXXXX legislative changes.* | *John Doe, XYZ Application Manager* | *10/31/14* | *In Progress* |

***[List any test assumptions, if any, for your project. Include the owner of the assumption and specify the impact (i.e., high, medium low***) ***to the project or business if the assumption was not timely managed.]***

| Assumptions | Owner | Impact | Status |
| --- | --- | --- | --- |
| 1. |  |  |  |
| 2. |  |  |  |

*Example:*

| Assumptions | Owner | Impact | Status |
| --- | --- | --- | --- |
| 1. *Funding has been approved to purchase software licenses for the test environment devices.* | *Mary Smith, Project Manager* | *Medium Impact - Project may delay the start of testing if the licenses have not been approved or purchased. The delay will impact the ability to get the test environment ready in time to begin testing.* | *In Progress* |

***[List any constraints associated with testing, if any, for your project. Include the owner of the constraint and specify the impact (i.e., high, medium, or low***) ***to the project or business if the constraint was not timely managed.]***

| Constraint | Owner | Impact | Status |
| --- | --- | --- | --- |
| 1. |  |  |  |
| 2. |  |  |  |

*Example:*

| Constraint | Owner | Impact | Status |
| --- | --- | --- | --- |
| 1. *The new application must be fully tested and implemented by 4/30/15.* | *John Smith, Business Division Manager* | *High Impact - Business will be penalized with payments of $XXX to the federal government for non-compliance if the system is not implemented by 6/1/15.* | *In Progress* |

## Software Risk

Risk is an uncertain event or condition that, if it occurs, has a positive or negative effect on the project objective (e.g., time, cost, scope). Product and project risk for   
**<<** Project Name **>>** are listed below.

[Identify the testing risks covered by the Master Test Plan.]

### Product Risks

Product risk is risk that is directly related to the system or component to be tested. Product risk may be associated with the quality of the product. For example, an unsatisfactory system might omit key functionality that the customers specified. Test related product risks may be documented in the project’s Risk Management Log.

[Identify test related product risks, severity of the risk if the risk were to occur, and a recommendation for mitigating the risk. Risks are graded by level of severity as follows: High, Medium, or Low. The test related product risk may be documented in the project’s Risk Management Log. See the CA-Project Management Methodology (PMM) within the State wide Information Management Manual (SIMM) for the methodology for rating risk.

If the test-related product risks are captured in a risk document (e.g., Risk Management Log), specify the document name and location of the test risk in this section.]

| ITEM Number | Risk | Severity | Contingency/Mitigation |
| --- | --- | --- | --- |
| 1. |  |  |  |
| 2. |  |  |  |

Example:

| ITEM Number | Risk | Severity | Contingency/Mitigation |
| --- | --- | --- | --- |
| 1. | *Critical defects in calculating the member’s account balance for the software system are identified late in the testing phase (e.g., user acceptance testing).* | *High* | *Plan early in the software lifecycle to ensure the requirement is consistent with business rules and process, ensure the requirement is testable, and perform test design and code reviews (e.g., test case traceability to the requirements, test case reviews).* |

### Project Risks

A project risk relates to the management and control of the project. For example, a project risk may be a lack of staffing or changing requirements. Test related project risks may be documented in the project’s Risk Management Log.

[Identify test-related project risks and provide recommendations to mitigate each risk. Risks are graded by level of severity as follows: High, Medium, or Low. If the test-related project risks are captured in a risk document (e.g., Risk Management Log), specify the document name and location of the test risk in this section.]

| ITEM Number | Risk | Severity | Contingency/Mitigation |
| --- | --- | --- | --- |
| 1. |  |  |  |
| 2. |  |  |  |

Example:

| ITEM Number | Risk | Severity | Contingency / Mitigation |
| --- | --- | --- | --- |
| 1. | *Skilled testing resources are unavailable during the software lifecycle, specifically during test execution, due to scheduling and resource constraints* | *Medium* | *Keep project manager informed of resource availability and revise project plans, as necessary. Seek temporary consulting resources to supplement test responsibilities*. |

# TEST STRATEGY

The test strategy describes how testing will be performed and explains any issues that may impact the success of the project. Each testing activity will consist of entry criteria, exit criteria, and deliverables. It is important that the test team thoughtfully analyze and thoroughly define in the Master Test Plan the entrance and exit criteria. Often the advancement to the next test level including production deployment rests on the successful attainment of the entry and exit criteria.

## Test Design Approach

Test design is the process of transforming general test objectives into tangible test conditions and test cases. Test design is divided into logical units (modules) based on the project’s features and functionality. Each module will have its own test design. The test designs will contain both verification and validation test objectives. Tests are written to coincide with the software lifecycle. As feature or functionality requirements are defined, the test team develops the test modules within the test phase (e.g., requirement phase, design phase) in preparation for testing the system in the test environment.

[Determine and document the Test Design Approach for the project and specify the entry criteria into Test Design, exit criteria leaving Test Design, deliverable produced, and due date.]

Entry/Exit Criteria

Entry criteria are the conditions which the product must meet in order to enter Test Design. Exit criteria are the conditions the product must meet to complete the Test Design.

| Criteria Type | Criteria Description |
| --- | --- |
| Entry | Description |
| Exit | Description |

Deliverable/due date

Test Activity/Deliverable is what needs to happen. Due date is when it must be completed.

| Test Activity/Deliverable | Due Date |
| --- | --- |
|  |  |
|  |  |

*Example*:

Entry/Exit Criteria

| Criteria Type | Criteria Description |
| --- | --- |
| *Entry* | * *Requirements are testable, documented, and have been approved and signed-off.* |
| *Exit* | * *There is one-to-many documented and validated test cases created to test the requirement.* * *The Test Case traces back to the requirements.* * *All severity 1 (critical) and severity 2 (major) defects have been fixed, fully tested, and meet system requirements.* |

Deliverable/due date

| Test Activity/Deliverable | Due Date |
| --- | --- |
| * *Test Case Document for Function XYZ* * *Test Procedure Document for Function XYZ* | *5/30/15* |

## Verification Approach

Verification is defined as the confirmation by examination and through provision of objective evidence that specified requirements have been fulfilled. The approach to verification may be formal or informal review or examination of the technical team’s work products without executing the software. The review areas may include, but are not limited to, requirements, design, testing, implementation, and maintenance.

The goal is to improve the quality of software work products by assisting the technical team in recognizing and fixing their own defects early in the software lifecycle. Industry practices have shown through studies that as a result of reviews, a significant increase in productivity and product quality can be achieved [Source: Gilb and Graham, 1993, van Veenendaal, 1999]. Reducing the number of defects early in the life cycle means that less time has to be spent in testing and maintenance.

### Requirements Review

During the requirements phase, requirements for a software product are defined and documented. Requirements review involves key project stakeholders (e.g., business analysts, technical team, business subject matter experts) meeting to understand and agree to the system’s requirements. These review sessions are important communication mechanisms to ensure project buy-in and system requirements approval signatures.

[Determine and document the approach to verify that the requirements (e.g., functional, non-functional) are clearly defined from a business perspective and that the requirements are testable.]

Example:

*Requirements Review sessions are conducted with project stakeholders including business subject matter experts and testers to ensure that all requirements have been clearly documented to meet the project’s business objectives. A requirements document is approved and has been signed by authorized stakeholders*.

Entry/Exit Criteria

To effectively fulfill the project’s objectives of Requirements Review, it is important that criteria be established to define the parameters of the review sessions and the outcome achieved at the conclusion of the review sessions. Entry criteria are the conditions which the product must meet in order to enter Requirements Review. Exit criteria are the conditions the product must meet to complete the Requirements Review.

[Specify the entry criteria into the Requirements Review, exit criteria leaving Requirements Review, deliverable produced, and due date.]

| Criteria Type | Criteria Description |
| --- | --- |
| Entry | Description |
| Exit | Description |

Deliverable/Due Date

| Test Activity/Deliverable | Due Date |
| --- | --- |
|  |  |
|  |  |

*Example*:

Entry/Exit Criteria

| Criteria Type | Criteria Description |
| --- | --- |
| *Entry* | * *Requirements have been documented according to IT standards (e.g., template) as defined by the project.* |
| *Exit* | * *Requirements are clearly defined, testable, and traceable.* |

Deliverable/Due Date

| Test Activity/Deliverable | Due Date |
| --- | --- |
| * *Traceability Matrix* * *Approved Requirements* | *5/30/15* |

### Use Case Review

Use cases represent a sequence of transactions in a dialogue between an actor and a component or system with a tangible result, where an actor can be a user or anything that can exchange information with the system. In a Use Case Review, documented use cases are reviewed with the key project stakeholders including the program area to ensure that all business workflows, processes, and rules are accurate and complete for the system being developed. As with requirements, use cases provide the foundation for test design and test execution. It is critical that the use cases are reviewed for accuracy to ensure the highest success during testing.

[Determine and document the approach to verify the Use Cases (e.g., functional, non-functional) are clearly defined with business rules and the requirements are testable.]

Example:

*Use Case Review sessions are conducted with the project stakeholders including business subject matter experts and testers to ensure that all use cases have been clearly documented to meet the project’s business objectives. The use case documents are approved and signed by authorized stakeholders.*.

Entry/Exit Criteria

To effectively fulfill the project’s objectives of the Use Case Review, it is important that criteria be established to define the parameters of the review sessions and the outcome achieved at the conclusion of the review sessions. Entry criteria are the conditions which the product must meet in order to enter Use Case Review. Exit criteria are the conditions the product must meet to complete the Use Case Review.

[Specify the entry criteria into the Use Case Review, exit criteria leaving Use Case Review, deliverable produced, and due date.]

| Criteria Type | Criteria Description |
| --- | --- |
| Entry | Description |
| Exit | Description |

Deliverable/due date

| Test Activity/Deliverable | Due Date |
| --- | --- |
|  |  |
|  |  |

*Example*:

Entry/Exit Criteria

| Criteria Type | Criteria Description |
| --- | --- |
| *Entry* | * *Use cases have been documented according to IT standards (e.g., template) as defined by the project.* |
| *Exit* | * *Use cases have been reviewed and approved by the project team (e.g., business, technical, testers).* |

Deliverable/Due date

| Test Activity/Deliverable | Due Date |
| --- | --- |
| * *Approved Use Cases* | *5/30/15* |

### Design Review

Design Review is the process in which the software design is evaluated against the requirements. Design Reviews may be conducted at a high level to check the overall architecture or at a lower level to check specifications. The importance of Design Review includes the assurance that the proposed design will meet the specified requirement, that the technical goals of the project will be met, and that the software design principles will be met according to standards.

Design Review sessions may include the presentation of planned software design at early stages of the lifecycle so that changes may be made easily. For example, a test design review may be comprised of meetings with the testing team to review the proposed test design documents such as test traceability. Test design reviews are important to validate that the contents of the designed tests will comprehensively test the system during Test Execution.

[Determine and document the approach to verify the Design documents meet test objectives and the test cases trace back to the requirements.]

Example:

*Design Review sessions are conducted with the test resources (e.g., Test Manager, Test Lead, and Test Executor) to:*

* *Verify the list of test objectives meets the project’s business objectives.*
* *Verify that the test cases are designed with traceability to the requirements and meet the test objectives.*
* *Verify the test procedures/scripts are clearly documented to align with the test cases*.

Entry/Exit Criteria

To effectively fulfill the project’s objectives of the Design Review, it is important that criteria be established to define the parameters of the review sessions and the outcome achieved at the conclusion of the review sessions. Entry criteria are the conditions which the product must meet in order to enter Design Review. Exit criteria are the conditions the product must meet to complete the Design Review.

[Specify the entry criteria into the Design Review, exit criteria leaving Design Review, deliverable produced, and due date.]

| Criteria Type | Criteria Description |
| --- | --- |
| Entry | Description |
| Exit | Description |

Deliverable/Due date

| Test Activity/Deliverable | Due Date |
| --- | --- |
|  |  |
|  |  |

*Example*:

Entry/Exit Criteria

| Criteria Type | Criteria Description |
| --- | --- |
| *Entry* | * *The design documents have been documented according to IT standards (e.g., template) as defined by the project.* |
| *Exit* | * *Design documents have been reviewed and approved by the project team (e.g., business, technical, testers).* |

Deliverable/due date

| Test Activity/Deliverable | Due Date |
| --- | --- |
| * *Approved Design Documents* | *5/30/15* |

### Code Review

Code is defined as computer instructions and data definitions expressed in a programming language or in a form output by an assembler, compiler, or other translator. Code Reviews typically are comprised of peer group discussions that focus on achieving consensus on the technical contents of a document. The technical experts involved in a technical review may include: architects, chief designers, and key users.

Code Review sessions are important where proactive defect-detection occurs. Correcting a defect in a conceptual technical specification is much cheaper than correcting a defect directly in a production environment. Also, Code Reviews ensure that processes and standards are consistently followed (e.g., coding standards), as this structure will facilitate the maintenance of the system during production operations.

[Determine and document the approach to verify that the code meets the requirements and/or standards.]

Example: *Code walkthrough inspections are conducted with technical peers (e.g., developers) to ensure the code follows standards and the program is efficiently constructed*.

Entry/Exit Criteria

To effectively fulfill the project’s objectives of a Code Review, it is important that criteria be established to define the parameters of the review sessions and the outcome achieved at the conclusion of the review sessions. Entry criteria are the conditions which the product must meet in order to enter Code Review. Exit criteria are the conditions the product must meet to complete the Code Review.

[Specify the entry criteria into the Code Review, exit criteria leaving Code Review, deliverable produced, and due date.]

| Criteria Type | Criteria Description |
| --- | --- |
| Entry | Description |
| Exit | Description |

Deliverable/due date

| **Test Activity/Deliverable** | **Due Date** |
| --- | --- |
|  |  |
|  |  |

*Example*:

Entry/Exit Criteria

| Criteria Type | Criteria Description |
| --- | --- |
| *Entry* | * *Source code has been written following IT standards as defined by the project.* * *Source code has been documented (e.g., purpose, processing instructions) according to IT standards as defined by the project.* |
| *Exit* | * *Source code review documents are approved by the appropriate project team (e.g., technical team).* |

Deliverable/due date

| Test Activity/Deliverable | Due Date |
| --- | --- |
| * *Baselined source code review documents* | *5/30/15* |

### Test Document Review (all types below)

Test Document Review consists of validating the quality of the documentation for the system. Examples of document review artifacts may include, but are not limited to: Master Test Plan, Unit Test Plan, Integration Test Plan, System Test Plan, User Acceptance Test Plan, Test Case Specification, Test Procedure Specifications, Test Incident Report, Test Log, Test Procedures, and Test Summary Report. Refer to *IEEE 829-2008 - IEEE Standard for Software and System Test Documentation* for more information on the test documents.

It is important that Documentation Reviews be conducted during the software lifecycle to ensure that test documentation for the system under test is accurate and complete. The documents are important throughout the software lifecycle and especially important during post production support including maintenance and operations.

[List the entry and exit criteria, completion deliverable, and due date. It may also be useful to include the standards and/or guidelines used to determine the test documents for review.]

Entry/Exit Criteria

To effectively fulfill the project’s objectives of Test Document Review, it is important that criteria be established to define the parameters of the review sessions and the outcome achieved at the conclusion of the review sessions. Entry criteria are the conditions which the product must meet in order to enter Test Document Review. Exit criteria are the conditions the product must meet to complete the Test Document Review.

| Criteria Type | Criteria Description |
| --- | --- |
| Entry | Description |
| Exit | Description |

Deliverable/due date

| Test Activity/Deliverable | Due Date |
| --- | --- |
|  |  |
|  |  |

*Example*:

Entry/Exit Criteria

| Criteria Type | Criteria Description |
| --- | --- |
| *Entry* | * *Test documents have been documented according to IT standards (e.g., template) as defined by the project.* |
| *Exit* | * *Test documents have been reviewed and approved by the project team (e.g., business, technical, testers).* |

Deliverable/Due date

| Test Activity/Deliverable | Due Date |
| --- | --- |
| * *Test documents that have been approved by authorized stakeholders (e.g., Master Test Plan, System Test Plan)* | *5/30/15* |

## Validation Approach

Validation is defined as the confirmation by examination and through provision of objective evidence that the requirements for a specific intended use or application have been fulfilled. Validation involves the actual testing of the software product. Validation is reactive as the emphasis is defect detection, where verification focuses on defect prevention.

Test validation is a critical component in the software lifecycle. The system under test is being validated to ensure that the system meets the requirements. Further, the results of test validation will assist the project in determining if business objectives for the system have been met.

[Describe the approach for testing the product and document the levels of testing conducted (e.g., unit testing, integration testing, system testing, user acceptance testing, and performance testing.]

### Unit Test

Unit Test is the activity of testing the smallest software element and testing the system’s internal structures, logic data paths, functions, and observable behavior. According to software development practices, the **<**< The group responsible for developing the software (indicate whether it is the Contractor or State) **>>** will perform Unit Test to discover discrepancies between the module’s design specifications and its actual behavior and to ensure the related module interfaces are tested.

It is important that Unit Test entrance and exit criteria are clearly defined and approved. The Unit Test entrance and exit criteria for **<<**Project Name**>>** are documented in the table below.

[List the entry and exit criteria, completion deliverable, and due date.]

Entry/Exit Criteria

Entry criteria are the conditions which the product must meet in order to enter Unit Test. Exit criteria are the conditions the product must meet to complete the Unit Test.

| Criteria Type | Criteria Description |
| --- | --- |
| Entry | Description |
| Exit | Description |

Deliverable/due date

| Test Activity/Deliverable | Due Date |
| --- | --- |
|  |  |
|  |  |

*Example*:

Entry/Exit Criteria

| Criteria Type | Criteria Description |
| --- | --- |
| *Entry* | * *Source code has been coded following IT standards as defined by the project.* |
| *Exit* | * *Source code has been successfully tested to meet system requirements.* * *No open severity 1 (critical) and/or severity 2 (major) defects.* |

Deliverable/due date

| Test Activity/Deliverable | Due Date |
| --- | --- |
| * *Level Test Summary Report* * *Approved unit test* * *Baselined source code* | *5/30/15* |

### Integration test

Integration testing is performed to expose defects in the interfaces and in the interactions between integrated components or systems (i.e. component integration testing, system integration testing). Component integration testing tests the interactions between software components. System integration testing tests the integration of systems and packages, testing the interfaces to external organizations (e.g., Electronic Data Interchange, Internet). The greater the scope of integration, the more difficult it is to isolate failures to specific interfaces, which may lead to an increased risk. Integration testing is often performed by the system integrator’s test team.

For << Project Name >>, Integration Test will be performed in accordance with the << Project’s Integration Test Process >>. Integration Test will be performed by the << Test group responsible for integration testing as stated in the contract (indicate whether it is the Contractor or State) >>.

Integration Test begins after all entrance criteria have been met. Integration Test will include regression tests, which retests previously tested features to ensure that the change or bug fix has not affected any part of the software application. Any defects found in regression test will need to be scheduled for retest.

Usability testing activities may be conducted during Integration Testing. Usability testing performed during Integration Testing will typically validate the interfaces between components and interactions to different parts of a system such as an operating system, file system and hardware, or interface between systems.

Usability testing is defined as testing to determine the extent to which the software product is understood, easy to learn, easy to operate, and attractive to the users under specified conditions. Integration Test is completed when the exit criteria has been met.

[Specify the conditions under which retest and regression test will be performed. This could include a description of the estimated number of test cycles.]

| Application Under Test \* | Initial List of Framework Components | Any Special Test Objectives |
| --- | --- | --- |
|  |  |  |

[\* There should be an Integration Test Plan for each application under test. Each plan should also include Usability Test. List the application to be tested, framework components, and any special test objectives. ]

| Applications needing Regression Test | Initial List of Regression Test Cases | Any Special Test Objective |
| --- | --- | --- |
|  |  |  |

[A Regression Test Case List should be created that contains a composite of all regression tests identified during Master Test planning and Integration Test planning. List the applications requiring regression test, regression test cases, and test objectives.]

[List the entry and exit criteria, completion deliverable, (i.e., Integration Test Plan(s) and Regression Test Case List), and due date.]

It is important that Integration Test entrance and exit criteria are clearly defined and approved. The Integration Test entrance and exit criteria for <<Project Name>> are documented in the table below.

Entry/Exit Criteria

Entry criteria are the conditions which the product must meet in order to enter Integration Test. Exit criteria are the conditions the product must meet to complete the Integration Test.

| Criteria Type | Criteria Description |
| --- | --- |
| Entry | Description |
| Exit | Description |

Deliverable/due date

| Test Activity/Deliverable | Due Date |
| --- | --- |
|  |  |
|  |  |

### 

*Example*:

Entry/Exit Criteria

| Criteria Type | Criteria Description |
| --- | --- |
| *Entry* | * *Evidence that unit test goals were completed successfully per the Unit Test Plan.* |
| *Exit* | * *Evidence that all severity 1 (critical) and severity 2 (major) integration test defects are closed. All remaining open defects have been either deferred or dispositions arranged per mutual agreement between the business and development teams.* |

Deliverable/due date

|  |  |
| --- | --- |
| Test Activity/Deliverable | Due Date |
| * *Level Test Summary Report* * *Approved Integration Test* | *5/30/15* |

### System Test

System Test is the process of demonstrating that a program or integrated system does or does not meet its original requirements and objectives as stated in the requirements specification. System Tests are designed by analyzing the requirements specification and created by analyzing the functional design specification or user documentation. For << Project Name >>, System Test will be performed in accordance with the << Project’s >> System Test Process. System Test commences and concludes when all the entrance and exit criteria have been fulfilled.

[List the System Test categories (e.g., Load, Security), whether the testing is required for the project, and the test objectives. Modify the System Test Category rows as it pertains to your project.]

| System Test Category | Required:  (Y)-Yes  (N)-No | Any Special Test Objective |
| --- | --- | --- |
| Volume Test |  |  |
| Load Test |  |  |
| Stress Test |  |  |
| Security Test |  |  |
| Usability Test |  |  |
| Performance Test |  |  |
| Resource Utilization Test |  |  |
| Configuration Test |  |  |
| Compatibility/Conversion Test |  |  |
| Install Ability Test |  |  |
| Failover/Recoverability Test |  |  |
| Serviceability/Maintainability Test |  |  |
| Reliability Test |  |  |
| Scenario Test |  |  |
| System Integration Test |  |  |
| Operational Test |  |  |

System Test will also include regression tests, which retest previously tested features to ensure that the change or bug fix has not affected any part of the software application. Any defects found in regression test will need to be scheduled for retest.

Usability Testing activities may be conducted during the System Testing phase. Usability Testing performed during System Testing will emphasize usability from a system integration perspective (i.e., collection of all the system features and functionality that comprise the integrated system). As stated, Usability Testing is defined as testing to determine the extent to which the software product is understood, easy to learn, easy to operate, and attractive to the users under specified conditions. In essence, Usability Testing is performed to ensure that the customer can easily use the system to perform business activities. System Test is completed when the exit criteria have been met.

Retest involves test cases that failed the last time they were run in order to verify the success of corrective actions. Regression Test is testing of a previously tested program following modification to ensure that defects have not been introduced or uncovered in unchanged area of the software as a result of the changes made. It is performed when the software or its environment is changed.

[Specify the conditions under which retest and regression test will be performed.

This may include a description of the estimated number of test cycles.]

| Application Under Test \* | Initial List of  Framework Components | Any Special  Test Objectives |
| --- | --- | --- |
|  |  |  |

[\* There should be a System Test Plan for each application under test. Each plan should also include Usability Test. List the application to be tested, framework components, and test objectives. ]

| Applications needing Regression Test | Initial List of  Regression Test Cases | Any Special  Test Objectives |
| --- | --- | --- |
|  |  |  |

[A Regression Test Case List should be created that contains a composite of all regression tests identified during Master Test planning and System Test planning. List the applications requiring regression test, regression test cases, and test objectives.]

[List the entry and exit criteria, completion deliverable (i.e., System Test Plan(s) and Regression Test Case List), and due date.]

It is important that System Test entrance and exit criteria are clearly defined and approved. The System Test entrance and exit criteria for <<Project Name>> are documented in the table below.

Entry/Exit Criteria

Entry criteria are the conditions which the product must meet in order to enter System Test. Exit criteria are the conditions the product must meet to complete the System Test.

| Criteria Type | Criteria Description |
| --- | --- |
| Entry | Description |
| Exit | Description |

Deliverable/due date

| Test Activity/Deliverable | Due Date |
| --- | --- |
|  |  |
|  |  |

*Example*:

Entry/Exit Criteria

| Criteria Type | Criteria Description |
| --- | --- |
| *Entry* | * *Evidence that integration test goals were completed successfully per the Integration Test Plan.* |
| *Exit* | * *Evidence that all severity 1 (critical) and severity 2 (major) system test defects are closed. All remaining open defects have been either deferred or dispositions made per mutual agreement between the business and development teams.* |

Deliverable/due date

| Test Activity/Deliverable | Due Date |
| --- | --- |
| * *Level Test Summary Report* * *Approved System Test* | *5/30/15* |

### User acceptance Test

User Acceptance Test (UAT) is the process of comparing the end product with the user requirements and demonstrating that those requirements have been satisfied. For << Project Name >>, the customers or end users will perform UAT to ensure the application meets the business requirements. Once the UAT entrance criteria have been met, UAT will begin with the customers or end users testing the software in a pre-production environment (e.g., Acceptance Environment) for a specified period of time. UAT testing will follow the << Project’s >> User Acceptance Test Process. User Acceptance Test will conclude once the UAT exit criteria have been met.

Also, User Acceptance Test will include regression tests, which retests previously tested features to ensure that the change or bug fix has not affected any parts of the software application. Any defects found in regression test will need to be scheduled for retest.

During User Acceptance Test, Usability Testing activities will be conducted. As mentioned, Usability Testing is defined as testing to determine the extent to which the software product is understood, easy to learn, easy to operate, and attractive to the users under specified conditions.

UAT Usability Testing emphasizes, from a user’s perspective, his or her ability to easily use the system to perform business activities. Often, users will test usability using normal daily transactions to validate whether they can easily use the system to perform critical business functions.

Retest and regression testing is also performed during User Acceptance Testing. As mentioned, retest involves testing test cases that failed the last time they were run in order to verify the success of corrective actions. Regression Test is testing of a previously tested program following modification to ensure that defects have not been introduced or uncovered in unchanged area of the software as a result of the changes made; it is performed when the software or its environment is changed.

[Specify the conditions under which retest and regression test will be performed. This may include a description of the estimated number of test cycles.]

| Application Under Test \* | Initial List of  Framework Components | Any Special  Test Objectives |
| --- | --- | --- |
|  |  |  |

[\* There should be a User Acceptance Test Plan for each application under test. Each plan should also include Usability Test. List the application to be tested, framework components, and test objectives. ]

| Applications needing Regression Test | Initial List of  Regression Test Cases | Any Special  Test Objectives |
| --- | --- | --- |
|  |  |  |

[A Regression Test Case List should be created that contains a composite of all regression tests identified during Master Test planning and User Acceptance Test planning. List the applications requiring regression test, regression test cases, and test objectives.]

[List the entry and exit criteria, completion deliverable (i.e., User Acceptance Test Plan(s) and Regression Test Case List), and due date.]

It is important that User Acceptance Test entrance and exit criteria are clearly defined and approved. The User Acceptance Test entrance and exit criteria for <<Project Name>> are documented in the table below.

Entry/Exit Criteria

Entry criteria are the conditions which the product must meet in order to enter User Acceptance Test. Exit criteria are the conditions the product must meet to complete the User Acceptance Test.

| Criteria Type | Criteria Description |
| --- | --- |
| Entry | Description |
| Exit | Description |

| Deliverable/Due dateTest Activity/Deliverable | Due Date |
| --- | --- |
|  |  |
|  |  |

*Example*:

Entry/Exit Criteria

| Criteria Type | Criteria Description |
| --- | --- |
| *Entry* | * *Evidence that system test goals were completed successfully per the System Test Plan.* * *Testers were trained on the system under test prior to UAT commencement.* |
| *Exit* | * *Evidence that all severity 1 (critical) and severity 2 (major) user acceptance test defects are closed. All remaining open defects have been either deferred or dispositions made per mutual agreement between the business and development teams.* |

Deliverable/due date

| Test Activity/Deliverable | Due Date |
| --- | --- |
| * *Test Summary Report* * *Approved User Acceptance Test* | *5/30/15* |

### Documentation Test Approach (if applicable)

The purpose of Documentation Test is to validate and verify documentation in conjunction with the software that it supports. Industry best practices specify documentation test for both printed documentation and on-screen documentation used in the work environment by the users of the system software and is applicable to user documentation for systems including hardware. This includes printed user manuals, online help, tutorials, and user reference documentation. In addition, best practice standards can be helpful for testing and reviewing the following types of documentation:

* Documentation of products other than software (e.g., hardware or devices)
* Multimedia systems using animation, video, and sound
* Computer-Based Training (CBT) packages and specialized course materials intended primarily for use in formal training programs
* Documentation produced for installers, computer operators, or system administrators who are not end users
* Maintenance documentation describing the internal operation of the system software

The roles involved in testing and the development of software and user documentation include project managers, usability experts and information developers, testers, and reviewers. The process for Documentation Test includes:

* Operational testing to ensure that the documentation performs correctly and responsively. For example, on-screen documentation is appropriately linked to the software topic and navigation operation is consistent and as expected. Also, the index in printed documentation has accurate references.
* Usability testing of documentation with the product to determine whether the intended audience can perform their tasks with the aid of the documentation.

More details regarding the process for the review and testing of documentation can be found in the *IEEE Standard for Adoption of ISO/IEC 26513:2009 Systems and Software Engineering—Requirements for Testers and Reviewers of User Documentation.*

<< For the Project, the project stakeholders may include business, technical, testers, and operations staff >> who will be responsible for performing documentation testing to uncover any errors in the system artifacts (e.g., inconsistencies between the online help screen and current graphical user interface, hyperlink within the online help not working).

[List the entry and exit criteria, completion deliverable (i.e., Document Test Plan), and due date.]

It is important that Documentation Test entrance and exit criteria are clearly defined and approved. The Documentation Test entrance and exit criteria for <<Project Name>> are documented in the table below.

Entry/Exit Criteria

Entry criteria are the conditions which the product must meet in order to enter Documentation Test. Exit criteria are the conditions the product must meet to complete Documentation Test.

| Criteria Type | Criteria Description |
| --- | --- |
| Entry | Description |
| Exit | Description |

Deliverable/due date

| Test Activity/Deliverable | Due Date |
| --- | --- |
|  |  |
|  |  |

*Example*:

Entry/Exit Criteria

| Criteria Type | Criteria Description |
| --- | --- |
| *Entry* | * *Evidence that system test and user acceptance testing were completed successfully and approved by authorized stakeholders.* |
| *Exit* | * *Evidence that all severity 1 (critical) and severity 2 (major) documentation test defects are closed. All remaining open defects have been either deferred or dispositions made per mutual agreement between the business and development teams.* |

Deliverable/due date

| Test Activity/Deliverable | Due Date |
| --- | --- |
| * *Test Summary Report* * *Approved Documentation Test* | *5/30/15* |

## Defect Management

The escalation process for handling issues and concerns regarding incidents and defects will be handled through << Project’s Name >> defect management escalation process. This process will document the framework including the flowchart, steps, resources, timeline, and escalation points (e.g., 1st Level – Test Team, 2nd Level – Project Management, 3rd Level – Executive Management/Sponsor). ***[Explain the escalation process for the Project here.]***

[Describe how defects are managed throughout the project. Information may include:

* Defect Management Process
* Defect Management Tool Used
* Defect Logging, Classification (e.g., Open, Closed, Withdrew, Deferred), and Reporting
* Defect Meeting (e.g., Triage), Defect Resolution, and Escalation Process
* Resources to manage defects]

## Metrics and Reporting

Metrics and reporting are used to monitor the activities related to the periodic checking of testing progress on a project. Reports are prepared that compare the actual results to that which was planned using metrics and other relevant information. Reports are used as a means to effectively communicate to project stakeholders the progress of testing. Reports may vary depending on factors such as the preference of the stakeholder, the needs and goals of the project, regulatory requirements, time and money constraints, and limitations of the tools available for test status reporting. Usually the Test Manager or Test Lead is tasked with monitoring the progress of test activities.

Test monitoring is important and serves the following purposes:

* Gives the test team and test manager feedback on how well the testing is progressing. This allows for opportunities to guide and improve the testing and the project.
* Provides the project team with visibility about the test results.
* Measures the status of the testing, test coverage, and test items against the exit criteria to determine whether the test work is finished.
* Gathers data for use in estimating future test efforts.

For smaller projects, test monitoring information may be gathered manually using documents, spreadsheets, and simple databases. For projects with large teams, distributed projects, and long-term test efforts, the use of automated tools may facilitate data collection.

For example, test progress may be gathered using the IEEE 829-2008 Test Log Template and reported using the IEEE 829-2008 Test Summary Report.

Determining what test data to collect and report should occur during test planning and the test preparation period. The actual collection of the specific metrics will occur during and at the end of a test period (e.g., system test, user acceptance test).

Questions to consider when determining what test metrics to capture for reporting purposes include:

* How will you assess the adequacy of the test objectives for a given test level and determine whether those objectives were achieved?
* How will you assess the adequacy of the test approaches taken and determine whether they support the achievement of the project’s testing goals?
* How will you assess the effectiveness of the testing with respect to those objectives and approaches?

For example, for requirements based testing, metrics may measure coverage in terms of requirements or functional areas rather than risk.

[Describe the metrics and reports for which values are to be collected during the test activities. Include sample metrics and reports in this plan.

Examples of Test Metrics:

Test Defects, Defects after Release, Open Problems, Open Issues, Test Results

* *Percentage of completed tests*
* *Number of tests that passed, failed, or were unable to progress because of blocking defects*
* *Number of outstanding defects (measured by severity and by component)*
* *Rate of defect discovery (measured by severity and by component)*
* *Rate of defects being fixed compared to the discovery rate*
* *Number of defect fixes requiring regression testing*]

## Test Data Requirements

Test data iscreated or selected to satisfy the input requirements for executing one or more test cases, which could be defined in a Test Plan (e.g., Master Test Plan, System Test Plan), test case, or test procedure. Test data may be stored within the product under test (e.g., in arrays, flat files, or a database), or could be available from or supplied by external sources, such as other systems, other system components, hardware devices, or human operators. The preparation of proper test data is important to the test set up, as without proper test data important test cases may be missed during testing due to unavailable data.

It is the responsibility of each tester to create his/her own test data according to testing needs. If it is not possible to create a new set of test data for each and every build, the next best option is to use standard production data. A technical support resource such as a database administrator may be needed to help facilitate the tester’s ability to gather the necessary data for testing. A good way to design test data is to use the existing sample test data or test bed (i.e., software and hardware requirements set using the predefined data values) and append the new test case data to the test bed.

[Describe the specific test data requirements for the project. Identify the test data source or origin where the specific test data is located, whether the data has to be disguised or masked for confidentiality reasons, and the roles/responsibilities for the test data.]

Example: *Test data originates from the legacy production system. Database Administration will copy the test data from the legacy production system to build a test bed. The test data needs to be masked and the test team will validate and certify the masked data in the target environment before testing begins. The test team will need a representative sampling of specific test data to cover test scenarios used during test execution cycles. After each test cycle, the data will need to be refreshed back to its original state. The Test Manager or Test Environment coordinator manages the test environment and test data***.**]

## Test Environment AND Facilities Requirements

A test environment contains hardware, instrumentation, simulators, software tools, and other support elements needed to conduct a test. It is important that the created software development test environments (e.g., development, test, user acceptance, staging, and training) mirror the target “To-Be” environment. Early planning during the lifecycle to define the test environment and facility (e.g., test lab, desktops) requirement is necessary to secure the appropriate configuration and environment set in preparation for test activities.

This section describes the test environment properties required to successfully test the system for << Project Name >>. This includes test environment requirements including, but not limited to, hardware (e.g., servers, desktops), software (e.g., Oracle eBusiness, Adobe Reader), testing tools (e.g., defect management tool), databases (e.g., Oracle), and personnel (e.g., system administrator, testing resources).

[Include a description of the generic test environment requirements and the approach to environment preparation. Test environment needs may include hardware, software, data, interfaces, facilities, extra power supply or air conditioning, publications, and/or security access.]

Examples:

*System Resource Requirements*

* *Web Server (e.g., Server Name, Network or Subnet)*
* *Application Server (e.g., Server Name, Network or Subnet)*
* *Database Server (e.g., Server Name, Network or Subnet)*
* *Test Repository (e.g., Server Name, Network or Subnet)*

[List the required System Resources, Quantity of Resources, Name/Classification Title (e.g., Senior Developer) of the System Resource, and the name of the designated IT Support Person responsible for the installation and management of the resources during the project.]

| System Resources |  |  |  |
| --- | --- | --- | --- |
| Resource | Quantity | Name and Title | IT Support Person |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

*Client Test PC Hardware configuration requirements*

* *Computer (e.g., Dell XYZ, 32 or 64 Bit Core Processor unit (CPU))*
* *Monitor (e.g., 24” monitor with a resolution of at least (nnn X mmm))*
* *Printer (e.g., Network printer)*

[List the required Hardware Type, detailed Hardware Specifics, and the name of the designated IT Support Person responsible for the installation and management of the hardware during the project.]

| Hardware type | Specifics | IT Support Person |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

*Software Requirements*

* *Windows Operating System*
* *Internet Browser (e.g., Internet Explorer X, Chrome, Firefox)*
* *Middleware*
* *Desktop Software (e.g., Adobe Reader, Microsoft Outlook)*
* *Virtualization Software*

[List the name of required Software, Version Number, Software Type, and the name of the designated IT Support Person responsible for the installation and management of the software during the project.]

| Software Name | Version | Type and Other Notes | IT Support Person |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

*Productivity and Support Tools used during the test process*

* *Test Management Tools (e.g., HP Quality Center)*
* *Requirements Management (e.g., IBM RequisitePro, Microsoft Word)*
* *Defect Tracking (e.g., IBM ClearQuest)*
* *Test Automation (e.g., HP LoadRunner, Quick Test Professional)*
* *Version Control (e.g., IBM ClearCase)*
* *Virtualization Tools*

[List the required Tool Category, Brand, Vendor Name, Version Number, and the name of the designated IT Support Person responsible for the installation and management of the tool during the project.]

| Tool Category or Type | Tool Brand Name | Vendor Name | VerSION Number | IT Support Person |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## Test Tools

Test tools are software products that support one or more test activities such as planning and control, specification, building initial files and data, test execution and test analysis. Test tools are important components of a project in that these tools help to efficiently manage testing activities such as test traceability, test progress, test case management, and defect management.

[List the test tools that will be used during testing. Include the Tool and Purpose, Responsible Party, Planned Date of Acquisition For New Test Tool, and Implementation Plans.]

*Example:*

*Test Tool*

* *Test Management Tools (e.g., HP Quality Center)*
* *Requirements Management (e.g., IBM RequisitePro, Microsoft Word)*
* *Defect Tracking (e.g., IBM ClearQuest)*
* *Test Automation (e.g., HP LoadRunner, Quick Test Professional)*
* *Version Control (e.g., IBM ClearCase)*
* *Virtualization Tools*

| Tool &  Purpose / Use | Responsible Party | Planned Date of Acquisition for New Test Tool | Implementation Plans |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Automation Plan

Test automation is the use of special [software](http://en.wikipedia.org/wiki/Software) (separate from the software being tested) to control the execution of tests and the comparison of actual outcomes with predicted outcomes. Test automation can automate some repetitive but necessary tasks in a formalized testing process already in place or add additional testing that would be difficult to perform manually. For example, extensive [regression testing](http://en.wikipedia.org/wiki/Regression_testing) can be laborious and time consuming to do manually. In addition, a manual approach might not always be effective in finding certain defects. Test automation may offer a possibility to perform these types of testing effectively. This section describes the test automation plans for <<Project Name>>.

| Automation Activity | Responsible Party | Automation REQUIREMENTS  (e.g., Tool, Data, Test Level, Environment) | Planned Date for automation |
| --- | --- | --- | --- |
| *Examples:*  *Define requirements and strategy for developing automated regression test scripts for system XYZ* | *John Doe, Automation Lead* | * *Test Level – System Test* * *HP QuickTest Professional* * *System Test (CorTEST Test Environment)* * *Test data source from XYZ of Test Cycle 1* | *5/15/15* |
| *Develop performance test scripts for load/stress test system XYZ* | *John Smith, Automation Engineer* | * *Test Level – System Test and User Acceptance Test* * *HP LoadRunner* * *System Test (CorTEST Test Environment) & User Acceptance (CorACC Acceptance Environment)* * *Test data source from XYZ of Test Cycles 1 & 2* | * *System Test Environment -8/1/15* * *User Acceptance Environment – 10/1/15* |

[Document test automation plans (e.g., Virtualization). Identify who is responsible for system and user acceptance test automation and associated data. The development group may also use automation for unit and integration testing.]

## Hardware Allocation & Procurement Plan

This section describes the allocation plans for hardware to support the project’s test activities. Hardware allocation may consist of existing resources (e.g., server) owned by the information technology department that is temporarily loaned to the project. Also, depending on the project requirements, new hardware may be required to support testing activities for the targeted system.

It is important that planning of hardware occurs early to ensure the timely delivery and installation of the hardware for testing. Ideally, planning should occur when the testing needs are defined in a Feasibility Study. This allows enough time to secure hardware and resource commitments with the IT department and allows the department to fiscally plan for the procurement of the new hardware. Early planning will ensure the timely delivery, installation, and validation of the hardware according to the project schedule’s testing timelines.

| hardware type | Hardware Source  (e.g., In-House, PRoCURED) | Responsible Party | Planned Date of hardware allocation | procurement Plan  (if required) |
| --- | --- | --- | --- | --- |
| *HP Server* | *Information Technology Department* | *Tom Smith, Infrastructure Manager* | *1/10/15* | *Not Applicable* |
|  |  |  |  |  |

[Describe how existing hardware resources will be allocated and how new hardware will be procured to support the project’s testing activities (e.g., servers, database, or service virtualization tools.]

## Suspension and Resumption Criteria

Suspension criteria identify the criteria used to temporarily stop all or a portion of the testing activities on the test item (e.g., component of the system to be tested). Resumption criteria specify the conditions used to restart all or a portion of the testing activities that were suspended previously.

[In the table below, enter the description of the criteria for suspending and resuming the test activity and the responsible party who has the authority to suspend and resume the test activities.]

suspension/resumption Criteria

| criteria | Criteria Description | responsible party for suspension/resumption |
| --- | --- | --- |
| Suspension | Description |  |
| Resumption | Description |  |

*Example*:

| criteria | Criteria Description | responsible party for suspension/resumption |
| --- | --- | --- |
| *Suspension* | * *If the selected tests in system test do not give the expected result, then the testing will be suspended until the bug is fixed.* * *The unavailability of external dependent system XYZ during system integration testing will result in testing suspension.* | *Jane Doe, Project Manager* |
| *Resumption* | * *Testing will begin only after the selected bug fixes have been corrected and yield the expected results.* * *The external dependent system XYZ is available during system integration testing during the timeframe 5/1/15 – 6/1/15.* | *Jane Doe, Project Manager* |

[Specify in the table below the resumption requirement for the test activity that may have to be repeated when testing is resumed, due date, and the responsible party to whom the activity is assigned for completion.]

Resumption Deliverable

| **resumption Test Activity/Deliverable** | **Assigned To** | **Due Date** |
| --- | --- | --- |
|  |  |  |
|  |  |  |

*Example:*

| **resumption Test Activity/Deliverable** | **Assigned To** | **Due Date** |
| --- | --- | --- |
| *Refresh the test environment to reflect the test bed data for release R1, dated 6/1/15.* | *John Doe, Data Conversion Support Lead* | *5/1/16* |
| *Review test cases and test data to verify external dependent system readiness. Coordinate testing resources at the external department to test system functionality.* | *Mary Doe, Test Manager* | *5/30/16* |

# TEST MANAGEMENT

Test Management is a critical project responsibility, where the individual has vested formal authority over the testing activities across the software lifecycle. The Test Manager is responsible for the overall coordination and orchestration of test planning, test design, test execution, test analysis, and defect management. This includes, but is not limited to: ensuring that test schedules align with the project’s schedule and budget, staffing the test efforts accordingly with skilled trained resources, motivating staff, managing the test environments, mitigating test issues and risks, and monitoring that entrance and exit criteria are attained within each test level.

## Administration

Test Management is the planning, estimating, monitoring and controlling of test activities which is typically carried out by the Test Manager. For << Project Name >> test management and administrative responsibilities for test execution will be performed by <<Test Manager Name >>. During Test Execution, the Test Manager will be responsible for managing the progress of testing through status reports and metrics, managing incidents and defects, managing testing resources, overseeing testing schedules and deliverables, and timely management of issues and risks. It is important that test activities are administered and tracked timely to ensure that testing activities complete within project timelines.

[Describe how test execution will be administered and tracked (e.g., status reports, summary report, issue log, incident report, and metrics) during the project. Identify any specific tools and techniques used. For generic forms and/or reports used across all levels of testing, identify the information with a sample of the form and/or reports, which can be referenced in the Appendix or accessed via a hyperlink.]

Example: *The Test Manager conducts all on-going test activity administration for the XYZ Project. These duties include test planning, test design, test execution, test analysis, and defect management. Tools used include PVCS Dimensions for defect tracking. The XYZ project further describes in detail the responsibilities defined in the test planning, test design, and test execution stages of the lifecycle…...*

## Approval Authority

Critical project test activities will require authorization to transition to the next test activity or test phase as defined in the entrance and exit criteria. Test phases include but are not limited to: unit test, integration test, system test, and user acceptance test. Approval of authorized stakeholders, as indicated in the table below, is required to ensure that test deliverables and milestones have been completed and that the system performs according to specifications. Failure to properly obtain approval signatures may result in a lack of governance and accountability or a product of inferior quality that does not meet customer expectations and requirements.

| **Test Activity requiring Approval** | **Authorized Approver** | **Document Location** |
| --- | --- | --- |
| *Test Design Approach* | *Jane Doe, Project Manager* | *Section X.X* |
| *Requirements Review* | *Mary Doe, Test Manager* | *Section X.X* |
| *Use Case Review* | *Mary Doe, Test Manager* | *Section X.X* |
| *Design Review* | *Mary Doe, Test Manager* | *Section X.X* |
| *Code Review* | *James Doe, Technical Manager* | *Section X.X* |
| *Test Document Review* | *Mary Doe, Test Manager* | *Section X.X* |
| *Unit Test* | *Mary Doe, Test Manager* | *Section X.X* |
| *Integration Test* | *Mary Doe, Test Manager* | *Section X.X* |
| *System Test* | *Jane Doe, Project Manager* | *Section X.X* |
| *User Acceptance Test* | *Jane Doe, Project Manager* | *Section X.X* |
| *Documentation Test* | *Mary Doe, Test Manager* | *Section X.X* |
| *Suspension Criteria* | *Jane Doe, Project Manager* | *Section X.X* |
| *Resumption Criteria* | *Jane Doe, Project Manager* | *Section X.X* |

[List the test activities requiring approval for your project, person authorized to sign-off on the test activity, and the section within the Master Test Plan which references the test activity. Table rows may be added or deleted, as necessary to reflect the project’s needs].

# STAFFING

Identifying the correct technical roles, resources, and staffing numbers is paramount to successfully managing and supporting testing activities. The Test Manager, in conjunction with the Project Manager, needs to ensure that operational level agreements are in place with management of the internal technical resources as early as possible. It is critical that the Test Manager and vendor clearly understand the statement of work expectations and deliverables for contracted technical staff. Failure to secure appropriate resources for testing may impact the ability to successfully complete testing responsibilities within schedule and budget.

[Describe the staffing requirements (e.g., development group, configuration management, customer support) for the tests covered by this plan.]

## Roles & Responsibilities

Successful test efforts require the management, coordination, and orchestration of resources assigned to various roles and responsibilities on a project. Testing resources may originate from different domain areas including, but not limited to: program area staff, information technology support staff (e.g., network, server, desktop support, database administration, data administration), operational staff, contractors, external interface partner personnel, and test organization resources (e.g., testers, test manager, test designer). To effectively and efficiently perform test responsibilities and duties according to project timelines, it is important that all members involved in testing efforts have clearly defined roles, responsibilities, and expectations.

A responsibility matrix is one mechanism where testing roles and responsibilities may be clearly defined as they relate to testing tasks or deliverables. A responsibility matrix, also known as a RACI matrix,describes the participation by various [roles](http://en.wikipedia.org/wiki/Role) in completing [tasks](http://en.wikipedia.org/wiki/Task_(project_management)) or [deliverables](http://en.wikipedia.org/wiki/Deliverable) for a [project](http://en.wikipedia.org/wiki/Project) or [business process](http://en.wikipedia.org/wiki/Business_process). It is especially useful in clarifying roles and responsibilities in cross-functional/departmental projects and processes. RACI is an [acronym](http://en.wikipedia.org/wiki/Acronym) derived from the four key responsibilities most typically used: Responsible, Accountable, Consulted, and Informed.

[Identify the roles and responsibilities required to participate in each level of the testing process. Determine the organization and manager for the test resource performing the test role. Describe the communication plan for defining the expectations and communications (e.g., meetings, emails) between the IT resources and Test Team. A responsibility matrix may be inserted in this section or referenced in the back of the document as an appendix.]

### Configuration and Release management resources

*Example:*

| Role | Organization | Manager | Responsibilities |
| --- | --- | --- | --- |
| *Deployment Manager* | *Information Technology Division* | *John Jones* | * *Plans deployment releases and provides timely assistance to the project team when deploying builds/releases into the test environments.* * *Responsible for the “go-live” phase of the project.* * *Ensures that hardware and software systems are fully deployed, implemented, and functioning.* * *Plans the roll-out process and the sequence of new systems and platform deployments.* * *Prepares plans, instructions, mapped IT system diagrams, and installation technical design packages.* * *Communicates with all affected internal and external stakeholders on the progress of the project.* |

### Unit Test Resources

*Example*:

| Role | Organization | Manager | Responsibilities |
| --- | --- | --- | --- |
| *Developer (may be state or contracted staff)* | *Information Technology Division* | *Jane Jones* | * *Responsible for the development, execution, and completion of unit test deliverables.* * *Ensures the timely delivery of unit test deliverables in accordance with the project schedule.* * *Interfaces with multi-disciplinary resources to understand requirements, business domain, design specifications, and test cases or scripts.* * *Ensures that the unit test deliverables adhere to the state’s processes and standards (e.g., code walkthrough, naming conventions)* * *Ensures the unit test application source code is maintained in the most current state through the project’s duration.* * *Collaborates with program staff during defect management triage discussions.* * *Consults in the development of the unit test plans.* * *Performs knowledge transfer of source code to designated resources.* * *Participates in baselining unit test source code artifacts in the test repository.* * *Participates in process improvements and lessons learned sessions with the test team.* |

## 

### Integration test resources

*Example:*

| Role | Organization | Manager | Responsibilities |
| --- | --- | --- | --- |
| *Integration Test Lead* | *Information Technology Division* | *Frank Smith* | * *Ensures that all system functions and interfaces (e.g., multiple components for a function) are successfully tested and that the integration of all functions meets requirement specifications.* * *Ensures that integration testing adheres to the project’s test policies and architectural practices.* * *Responsible for leading the development, execution, and completion of integration test deliverables.* * *Leads and ensures the timely delivery of integration test deliverables in accordance with the project schedule.* * *Interfaces with multi-disciplinary resources to understand requirements, business domain, designs specifications, and test cases or scripts.* * *Ensures that the integration test deliverables adhere to the state’s processes and standards (e.g., code walkthrough, naming conventions).* * *Ensures the application source code is maintained in the most current state through the project’s duration.* * *Collaborates with program staff and application staff during defect management triage discussions.* * *Consults in the development of the integration test plan.* * *Participates in knowledge transfer of source code to designated resources.* * *Participates in establishing the baseline of integration test source code artifacts in the test repository.* * *Leads process improvements and lessons learned sessions with the test team.* |

### System Test Resources

*Example:*

| **Role** | **Organization** | **Manager** | **Responsibilities** |
| --- | --- | --- | --- |
| *System Test Lead* | *Information Technology Division* | *Scott Green* | * *Ensures that entrance criteria have been fulfilled prior to starting system testing.* * *Ensures staff is resourced to execute test cases within the project timeframe.* * *Ensures that all system functions are successfully tested and that the system meets requirement specifications.* * *Ensures that system testing adheres to the project’s test policies and architectural practices.* * *Responsible for leading the development, execution, and completion of system test deliverables.* * *Leads and ensures the timely delivery of system test deliverables in accordance with the project schedule.* * *Interfaces with multi-disciplinary resources to understand requirements, business domain, design specifications, and test cases or scripts.* * *Ensures that the system test deliverables adhere with the state’s processes and standards.* * *Ensures the application source code is maintained in the most current state through the project’s duration.* * *Collaborates with program staff and application staff during defect management triage discussions.* * *Consults in the development of the system test plan.* * *Participates in knowledge transfer of source code to designated resources.* * *Participates in establishing baseline of integration test source code artifacts in the test repository.* * *Leads process improvements and lessons learned sessions with the test team.* |

### user Acceptance Test Resources

*Example:*

| Role | Organization | Manager | Responsibilities |
| --- | --- | --- | --- |
| *Subject Matter Expert (e.g., Business Lead)* | *Business Division* | *Julie Smith* | * *Consults with the Test Manager to ensure that the acceptance entrance criteria have been fulfilled prior to starting user acceptance testing.* * *Responsible for the development, execution, and completion of UAT deliverables.* * *Provides input regarding the definition of the UAT exit criteria.* * *Develops the business focused test cases for UAT.* * *Ensures that the business focused test cases have been executed to successfully test that the system meets requirement specifications.* * *Ensures the timely delivery of UAT deliverables in accordance with the project schedule.* * *Ensures that the UAT test deliverables adhere to the state’s processes and standards.* * *Ensures the UAT test cases are maintained in the most current state through the project’s duration.* * *Collaborates with the technical team during defect management triage discussions.* * *Consults in the development of the UAT test plans.* * *Participates in test data readiness verification.* * *Provides input to the Test Manager on issues and risks.* * *Provides test status updates to the Test Manager.* * *Assists in the development of training materials and education to program staff resources.* * *Participates in baselining UAT artifacts in the test repository.* * *Participates in process improvement and lessons learned sessions with test team.* |

## Test Team Training Requirements

To ensure a highly productive test team, resources may require training in order to successfully and competently perform the testing duties and responsibilities for a project. Training provides organizational benefits such that staff would enhance their skills, improve morale, promote retention, and successfully support test operations. Examples of test training may include: training on developing test cases, learning the new system under test, understanding the defect management tool, training on how to log defects, or learning new business workflow changes.

[It is important to prepare the test team, users, operations staff, project staff, and program staff for the new system, as these individuals will likely influence the success of the system and project via customer confidence through a positive hands-on user experience. The success of any project rests on the ability to smoothly transition a project to staff from development into production for maintenance and operational activities.]

[Specify test training needs by skill level and identify training options for providing the necessary skills for the staff including, but not limited to customer support, installation, and production operations.]

Role/Resource

| **Role/Resource** | **Number To Be Trained** | **Course Name** | **Training Due Date or Test Phase** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

# PROJECT SCHEDULE AND TEST ARTIFACT REPOSITORY

The project schedule serves as a roadmap for the test team and illustrates the dependencies, relationships, constraints, resources, and time estimates for each activity relative to the overall project phases. The project schedule should be designed to align test tasks, activities, and milestones with the test strategy for the system under test.

A test artifact repository/library is important for the purpose of storing critical project information such as: requirements, use cases, test cases, and test scripts. The artifacts will become important after project completion and after the system has transitioned to production under maintenance and operations. Application and test resources will access the test cases and test scripts for regression testing purposes. Operations support staff will use the test artifacts to gather release note information, known errors, workarounds, and process improvement recommendations.

[Identify test milestones defined in the project schedule and from the test strategy. Specify the schedule for each testing activity and test milestone based on the activity estimates, available resources, and other constraints. A graphic depiction of the project schedule with pertinent test information may be inserted in this section (i.e., Microsoft Project). Also, if the test schedule is located in the project files, specify the location of the project repository where the test artifact can be accessed.]

*Example*:

| **Milestones** | **Start Date** | **Completion Date** |
| --- | --- | --- |
| *Completed Testing Schedule* | *<< mm/dd/yy >>* | *<< mm/dd/yy >>* |
| *Completed Test Plan* | *<< mm/dd/yy >>* | *<< mm/dd/yy >>* |
| *Completed Test Objectives* | *<< mm/dd/yy >>* | *<< mm/dd/yy >>* |
| *Completed Test Cases* | *<< mm/dd/yy >>* | *<< mm/dd/yy >>* |
| *Executed Test Cases* | *<< mm/dd/yy >>* | *<< mm/dd/yy >>* |
| *Execute “Interface” and “Integration” testing* | *<< mm/dd/yy >>* | *<< mm/dd/yy >>* |
| *Finalized testing and restore database* | *<< mm/dd/yy >>* | *<< mm/dd/yy >>* |
| *Completed and approved testing documents* | *<< mm/dd/yy >>* | *<< mm/dd/yy >>* |

# 

1. COTs, MOTS, and Custom implementation

Appendix A describes the implementation considerations and test emphasis for a COTS, MOTS, or Custom application. Testing is a critical component to validating that a system meets requirements. It is critical that testing be conducted with the right technical resources. As such, with the exception of Unit Testing where the developer performs the testing, testing should be conducted by an independent test team that did not program the source code.

| **CONSIDERATIONS FOR COTS, MOTS, and CUSTOM IMPLEMENTATION** | |
| --- | --- |
| **COTS** | * Unit Test – Testing of the product capabilities is not necessary. The users do not have access to the source code. * System Test – Testing of business requirements relative to the COTS product is required. Testing will focus on how the COTS configuration changes meet business requirements. The testing should be conducted by an independent test team that did not configure the COTS product. * Performance Test – The testing focus is usually a set number of users (i.e., with an authorized license) aim to ensure the volume of users can be accommodated without degrading the system, unlike custom systems where the emphasis may be scaling a web application to handle a million users on the system. For example, performance testing may test how quickly data transactions process under “normal use” and the maximum number of transactions encountered before system degradation occurs (e.g., 1000 transactions need to be processed within 30 minutes). The testing may be conducted by a skilled tester with extensive knowledge in performance testing within the development group or by an independent test team. * Integration Test – Testing of the COTS product and how the system integrates with other systems is required. This includes testing such as: compatibility testing, performance testing, interoperability testing, and back-end testing (e.g., writing SQL statements to verify the data in the database). For example, the COTS product may send data or may interface with another legacy application where the data is processed and reported in an ancillary system. The testing should be conducted by an independent test team that did not configure the COTS product. * User Acceptance Test – The users will focus on testing the daily business transactions and whether the COTS product meets business requirements. Further, the users will test how the COTS product integrates with other ancillary systems through comprehensive business scenarios. The users may apply new or existing business processes to accompany User Acceptance Testing. |
| **MOTS** | * Unit Test – The modified source code pertinent to the COTS application will require unit testing. These changes are not configuration changes, but rather customized changes to the source code. The developer typically performs unit testing. * System Test - The COTS portion will be validated the same way as indicated above (See COTS Considerations). For the custom portion of the COTS solution where application source code changes occurred, as usual, testing will focus on the solution meeting business requirements. Also, testing should be conducted by an independent test team that did not program or modify the source code. * Integration Test - The tests focus on how well both the COTS and customization of the application meets business requirements. Further, if the MOTS solution interfaces with other systems, testing is required to ensure that the integration between the MOTS and other systems (e.g., legacy system) meets business requirements. The testing should be conducted by an independent test team that did not configure or write source code for the MOTS product. User Acceptance Test – The users will focus on testing the daily business transactions and whether the MOTS product meets their requirements. Also, the users may apply new or existing business processes to accompany User Acceptance Testing. NOTE: Depending on the amount of application customization in the MOTS solution, an organization may risk invalidation of the product warranty and the inability to perform regular application patch updates. |
| **CUSTOM** | * All test levels need to be conducted (e.g., Unit Test, Integration Test, System Test, System Integration Test, and User Acceptance Test). With the exception of Unit Testing, where the developer performs the testing, testing should be conducted by an independent test team that did not program the source code. |

1. Glossary

The following is a glossary of terms used in this Master Test Plan.

| TERM | DEFINITION |
| --- | --- |
| **Acceptance** | See acceptance testing. |
| **Acceptance Criteria** | The exit criteria that a component or system must satisfy in order to be accepted by a user, customer, or other authorized entity. |
| **Acceptance Testing** | Formal testing with respect to user needs, requirements, and business processes conducted to determine whether or not a system satisfies the acceptance criteria and to enable the user, customers, or other authorized entity to determine whether or not to accept the system. |
| **Availability** | The degree of which a component or system is operational and accessible when required for use. Often expressed as a percentage. |
| **Baseline** | A specification or software product that has been formally reviewed or agreed upon, that thereafter serves as the basis for further development, and that can be changed only through a formal change control process. |
| **Best Practice** | A superior method or innovative practice that contributes to the improved performance of an organization under given context usually recognized as ‘best’ by other peer organizations. |
| **Black Box Testing** | Testing, either functional or non-functional, without reference to the internal structure of the component or system. |
| **Bug** | See defect. |
| **Code** | Computer instructions and data definitions expressed in a programming language or in a form output by an assembler, compiler, or other translator. |
| **Commercial Off-The-Shelf Software** | See off-the-shelf software. |
| **Compatibility Testing** | See interoperability testing. |
| **Component** | A minimal software item that can be tested in isolation. |
| **Component Integration Testing** | Testing performed to expose defects in the interfaces and interaction between integrated components. |
| **Component Testing** | The testing of individual software components. |
| **Configuration Testing** | See portability testing. |
| **Confirmation Testing** | Testing that executes test cases that failed the last time they were run in order to verify the success of corrective actions. |
| **Conversion Testing** | Testing of software used to convert data from existing systems for use in replacement systems. |
| **COTS** | Acronym for Commercial Off-The-Shelf software. See off-the-shelf software. |
| **Coverage** | The degree, expressed as a percentage, to which a specified coverage item has been exercised by a test suite. |
| **Coverage Item** | An entity or party used as a basis for test coverage (e.g., equivalence partitions or code statements). |
| **Defect** | A flaw in a component or system that can cause the component or system to fail to perform its required function (e.g., an incorrect statement or data definition). A defect, if encountered during execution, may cause a failure of the component or system.  Examples:  Critical - The defect affects critical functionality or critical data. It does not have a workaround.  Major - The defect affects major functionality or major data. It has a difficult workaround that is not obvious to the user.  Minor - The defect affects minor functionality or non-critical data. It has a simple workaround.  Trivial - The defect does not affect functionality or data. It does not even need a workaround. It does not impact productivity or efficiency but is merely an inconvenience. |
| **Defect Management** | The process of recognizing, investigating, acting, and disposing of defects. This involves recording defects, classifying them, and identifying their impact. |
| **Defect Management Tool** | A tool that facilitates the recording and status tracking of defects and changes. The tool often has workflow-oriented facilities to: track and control the allocation, correction, re-testing of defects, and provide reporting facilities. See also incident management tool. |
| **Deliverable** | Any (work) product that must be delivered to someone other than the (work) product’s author. |
| **Deviation** | See incident. |
| **Documentation Testing** | Testing the quality of the documentation (e.g., user guide or installation guide). |
| **Effectiveness** | The capability of producing an intended result. See also efficiency. |
| **Efficiency** | The capability of the software product to provide appropriate performance relative to the amount of resources used under stated conditions. Also, the capability of a process to produce the intended outcome relative to the amount of resources used. |
| **Entry Criteria** | The set of generic and specific conditions for permitting a process to go forward with a defined task (e.g., test phase). The purpose of entry criteria is to prevent a task from starting which would entail more (wasted) effort compared to the effort needed to remove the failed entry criteria. |
| **Error** | A human action that produces an incorrect result. |
| **Exit Criteria** | The set of generic and specific conditions agreed upon with the stakeholders for permitting a process to be officially completed. The purpose of exit criteria is to prevent a task from being considered completed when there are still outstanding parts of the task which have not been finished. Exit criteria are used to report against and to plan when to stop testing. |
| **Expected Result** | The behavior predicted by the specification, or another source, of the component or system under specified conditions. |
| **Fail** | A test is deemed to fail if its actual result does not match its expected result. |
| **Failover Testing** | Testing by simulating failure modes or actually causing failures in a controlled environment. Following a failure, the failover mechanism is tested to ensure that data is not lost or corrupted and that any agreed service levels are maintained (e.g., function availability or response times). See also recoverability testing. |
| **Failure** | Deviation of the component or system from its expected delivery, service, or result. |
| **Fault** | See defect. |
| **Feature** | An attribute of a component or system specified or implied by requirements documentation (e.g., reliability, usability or design constraints). |
| **Feature Set** | Logical subset of the test item(s) that may be treated independently of other feature sets in the subsequent test design activities. |
| **Formal Review** | A review characterized by documented procedures and requirements (e.g., inspection). |
| **Functional Requirement** | A requirement that specifies a function that a component or system must perform. |
| **Functional Testing** | Testing based on an analysis of the specification of the functionality of a component or system. See also black box testing. |
| **Functionality** | The capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions. |
| **Functionality Testing** | The process of testing to determine the functionality of a software product. |
| **Horizontal Traceability** | The tracing of requirements for a test level through the layers of test documentation (e.g., test plan, test design specification, test case specification, and test procedure specification or test script). |
| **Hyperlink** | A pointer within a web page that leads to other web pages. |
| **Incident** | Any event occurring that requires investigation. |
| **Incident Management** | The process of recognizing, investigating, taking action, and disposing of incidents. It involves logging incidents, classifying them, and identifying the impact. |
| **Incident Management Tool** | A tool that facilitates the recording and status tracking of incidents. The tool often has workflow-oriented facilities to track and control the allocation, correction, and re-testing of incidents, and to provide reporting facilities. See also defect management tool. |
| **Informal Review** | A review not based on a formal (documented) procedure. |
| **Input** | A variable (whether stored within a component or outside) that is read by a component. |
| **Install ability** | The capability of the software product to be installed in a specified environment. See also portability. |
| **Install ability Testing** | The process of testing the install ability of a software product. See also portability testing. |
| **Installation Guide** | Supplied instructions on any suitable media which guides the installer through the installation process. This may be a manual guide, step-by-step procedure, installation wizard, or any other similar process description. |
| **Integration** | The process of combining components or systems into larger assemblies. |
| **Integration Testing** | Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems. See also component integration testing, system integration testing. |
| **Interoperability** | The capability of the software product to interact with one or more specified components or systems. See also functionality. |
| **Interoperability Testing** | The process of testing to determine the interoperability of a software product. See also functionality testing. |
| **Load Testing** | A type of performance testing conducted to evaluate the behavior of a component or system with increasing demand (e.g., numbers of parallel users and/or numbers of transactions) to determine what load can be handled by the component or system. See also performance testing, stress testing. |
| **Maintainability** | The ease with which a software product can be modified to correct defects, modified to meet new requirements, modified to make future maintenance easier, or adapted to a changed environment. |
| **Maintainability Testing** | The process of testing to determine the maintainability of a software product. |
| **Maintenance** | Modification of a software product after delivery to correct defects, to improve performance or other attributes, or to adapt the product to a modified environment. |
| **Master Test Plan** | A test plan that typically addresses multiple test levels. See also test plan. |
| **Metric** | A measurement scale and the method used for measurement. |
| **Milestone** | A point in time in a project at which defined (intermediate) deliverables and results should be ready. |
| **Monitor** | A software tool or hardware device that runs concurrently with the component or system under test and supervises, records, and/or analyzes the behavior of the component or system. |
| **Non-Functional Requirement** | A requirement that does not relate to functionality, but to attributes such as reliability, efficiency, usability, maintainability and portability. |
| **Off-The-Shelf Software** | A software product that is developed for the general market (i.e., for a large number of customers) and that is delivered to many customers in identical format. |
| **Operational Environment** | Hardware and software products installed at users’ or customers’ sites where the component or system under test will be used. The software may include operating systems, database management systems, and other applications. |
| **Operational Testing** | Testing conducted to evaluate a component or system in its operational environment. |
| **Output** | A variable (whether stored within a component or outside) that is written by a component. |
| **Pass** | A test is deemed to pass if its actual result matches its expected result. |
| **Pass/Fail Criteria** | Decision rules used to determine whether a test item (function) or feature has passed or failed a test. |
| **Performance** | The degree to which a system or component accomplishes its designated functions within given constraints regarding processing time and throughput rate. See also efficiency. |
| **Performance Testing** | The process of testing to determine the performance of a software product. See also efficiency testing. |
| **Portability** | The ease with which the software product can be transferred from one hardware or software environment to another. |
| **Portability Testing** | The process of testing to determine the transportability of a software product. |
| **Priority** | The level of (business) importance assigned to an item (e.g., defect). |
| **Problem** | See defect. |
| **Product Risk** | A risk directly related to the test object. See also risk. |
| **Project** | A project is a unique set of coordinated and controlled activities with start and finish dates undertaken to achieve an objective conforming to specific requirements, including the constraints of time, cost, and resources. |
| **Project Risk** | A risk related to management and control of the (test) project, (e.g., lack of staffing, strict deadlines, changing requirements, etc.). See also risk. |
| **RACI Matrix** | A matrix describing the participation by various roles in completing tasks or deliverables for a project or process. It is especially useful in clarifying roles and responsibilities. RACI is an acronym derived from the four key responsibilities most typically used: Responsible, Accountable, Consulted, and Informed. |
| **Recoverability** | The capability of the software product to re-establish a specified level of performance and recover the data directly affected in case of failure. See also reliability. |
| **Recoverability Testing** | The process of testing to determine the recoverability of a software product after interruption |
| **Regression Testing** | Testing of a previously tested program following modification to ensure that defects have not been introduced or uncovered in unchanged areas of the software as a result of the changes made. It is performed when the software or its environment is changed. |
| **Reliability** | The ability of the software product to perform its required functions under stated conditions for a specified period of time or for a specified number of operations. |
| **Reliability Testing** | The process of testing to determine the dependability of a software product. |
| **Requirement** | A condition or capability needed by a user to solve a problem or achieve an objective that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document. |
| **Requirements-Based Testing** | An approach to testing in which test cases are designed based on test objectives and test conditions derived from requirements (e.g., tests that exercise specific functions or probe non-functional attributes such as reliability or usability). |
| **Resource Utilization** | The capability of the software product to use appropriate amounts and types of resources (e.g., amounts of main and secondary memory used by the program and the sizes of required temporary or overflow files) when the software performs its function under stated conditions. See also efficiency. |
| **Resource Utilization Testing** | The process of testing to determine the resource-utilization of a software product. See also efficiency testing. |
| **Result** | The consequence/outcome of the execution of a test. It includes outputs to screens, changes to data, reports, and communications sent. See also actual result, expected result. |
| **Resumption Criteria** | The criteria used to restart all or a portion of the testing activities that were suspended previously. |
| **Resumption Requirements** | The defined set of testing activities that must be repeated when testing is re-started after a suspension. |
| **Re-Testing** | See confirmation testing. |
| **Review** | An evaluation of a product or project status to ascertain discrepancies from planned results and to recommend improvements. Examples include: management review, informal review, technical review, inspection, and walkthrough. |
| **Risk** | A factor that could result in future negative consequences usually expressed in terms of impact and likelihood. |
| **Risk Management** | Systematic application of procedures and practices to the tasks of identifying, analyzing, prioritizing, and controlling risk. |
| **Scenario Testing** | See use case testing. |
| **Security** | Attributes of software products that bear on its ability to prevent unauthorized access, whether accidental or deliberate, to programs and data. See also functionality. |
| **Security Testing** | Testing to determine the security of the software product. See also functionality testing. |
| **Serviceability Testing** | See maintainability testing. |
| **Severity** | The degree of impact that a defect has on the development or operation of a component or system. |
| **Software** | Computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system. |
| **Software Lifecycle** | The period of time that begins when a software product is conceived and ends when the software is no longer available for use. The software lifecycle typically includes a concept phase, requirements phase, design phase, implementation phase, test phase, installation, operation and maintenance phase, and sometimes, retirement phase. Note these phases may overlap or be performed iteratively. |
| **Suspension Criteria** | The criteria used to (temporarily) stop all or a portion of the testing activities on the test items. |
| **System** | A collection of components organized to accomplish a specific function or set of functions. |
| **System Integration Testing** | Testing the ability of systems to perform together, including packages and interfaces to external organizations (e.g., Electronic Data Interchange, Internet). |
| **System Testing** | The process of testing an integrated system to verify that it meets specified requirements. |
| **Test** | The activity of executing a set of one or more test cases and procedures. |
| **Test Approach** | The implementation of the test strategy for a specific project. It typically includes the decisions made based on the (test) project’s goal and the risk assessment carried out, starting points regarding the test process, the test design techniques to be applied, and exit criteria and test types to be performed. |
| **Test Bed** | See test environment. |
| **Test Case** | A set of input values, execution preconditions, expected results and execution post conditions, developed for a particular objective or test condition such as to exercise a particular program path or to verify compliance with a specific requirement. |
| **Test Condition** | An item or event of a component or system that may be verified by one or more test cases (e.g., a function, transaction, feature, quality attribute, or structural element). |
| **Test Coverage** | See coverage. |
| **Test Data** | Data that exists (e.g., database) before a test is executed and that affects or is affected by the component or system under test. |
| **Test Deliverable** | Any test (work) product that must be delivered to someone other than the test (work) product’s author. See also deliverable. |
| **Test Design** | The process of transforming general test objectives into tangible test conditions and test cases. See test design specification. |
| **Test Environment** | An environment containing hardware, instrumentation, simulators, software tools, and other support elements needed to conduct a test. |
| **Test Execution** | The process of running a test on the component or system under test producing actual result(s). |
| **Test Item** | The individual element to be tested. There usually is one test object and many test items. See also test object. |
| **Test Level** | A group of test activities that is organized and managed together. A test level is linked to the responsibilities in a project. Examples of test levels are component test, integration test, system test, and acceptance test. |
| **Test Log** | A chronological record of relevant details about the execution of tests. |
| **Test Management** | The planning, estimating, monitoring, and control of test activities typically carried out by a test manager. |
| **Test Manager** | The person responsible for management of project testing activities, resources, and evaluation of a test object. The individual that directs, controls, administers, plans, and regulates the evaluation of a test object. |
| **Test Object** | The component or system to be tested. See also test item. |
| **Test Phase** | A distinct set of test activities collected into a manageable phase of a project (e.g., the execution activities of a test level). |
| **Test Plan** | A document describing the scope, approach, resources, and schedule of intended test activities. It identifies, among other test items: the features to be tested, the testing tasks, the resource that will perform each task, the degree of tester independence, the test environment, the test design techniques, the entry and exit criteria, and any risks requiring contingency planning. It is a record of the test planning process. See also Master Test Plan. |
| **Test Planning** | The activity of establishing or updating a test plan. |
| **Test Result** | See result. |
| **Test Scenario** | See test procedure specification. |
| **Test Schedule** | A list of activities, tasks, or events of the test process identifying their intended start and finish dates and/or times and interdependencies. |
| **Test Script** | Commonly used to refer to a test procedure specification, especially an automated one. |
| **Test Set** | See test suite. |
| **Test Strategy** | A high-level description of the test levels to be performed and the testing within those levels for an organization or program (e.g., one or more projects). |
| **Test Suite** | A set of several test cases for a component or system under test where the post condition of one test is often used as the pre-condition for the next one. |
| **Test Summary Report** | A document summarizing testing activities and results. It also contains an evaluation of the corresponding test items against exit criteria. |
| **Testable Requirement** | A requirement that is stated in terms that permit establishment of test designs (and subsequently test cases) and execution of tests to determine whether the requirement has been met. |
| **Tester** | A skilled professional who is involved in the testing of a component or system. |
| **Testing** | The process consisting of all lifecycle activities including both static and dynamic. This process involves the planning, preparation, and evaluation of software and related work products to determine that specified requirements are satisfied, are fit for purpose, and that defects are detected. |
| **Traceability** | The ability to identify related items in documentation and software such as requirements with associated tests. See also horizontal traceability and vertical traceability. |
| **Traceability Matrix** | A two-dimensional table which correlates two entities (e.g., requirements and test cases). The table allows tracing back and forth the links of one entity to the other, thus enabling the determination of coverage achieved and the assessment of impact of proposed changes. |
| **Unit** | See component. |
| **Unit Testing** | See component testing. |
| **Usability** | The capability of the software to be understood, learned, used, and attractive to the user when used under specified conditions. |
| **Usability Testing** | Testing to determine the extent to which the software product is understood, easy to learn, and easy to operate and attractive to the users under specified conditions. |
| **Use Case** | A sequence of transactions in a dialogue between an actor and a component or system with a tangible result, where an actor can be a user or anything that can exchange information with the system. |
| **User Acceptance Testing** | See acceptance testing. |
| **Validation** | Confirmation by examination and through provision of objective evidence that the requirements for a specific intended use or application have been fulfilled. |
| **Verification** | Confirmation by examination and through provision of objective evidence that specified requirements have been fulfilled by the product and/or system. |
| **Vertical Traceability** | The tracing of requirements through the layers of development documentation to components. |
| **Volume Testing** | Testing where the system is subjected to large volumes of data. See also resource utilization testing. |
| **Walkthrough** | A step-by-step presentation by the author of a document in order to gather information and to establish a common understanding of its content. See also peer review. |