Test Plan

Instructions

Version 1.2 ● 14 JAN 2008

Version History

This and other Framework Extension tools are available on the Framework Web site.

Release Date	Description
14-Jan-2008	Version 1.2 released. Modified “Using this Template” section of the Template and italicized all section instructions to align with the Framework and Change Request (CR) #34. CR #34 was recommended by the Framework Change Advisory Board (CAB) and approved by DIR.
13-Mar-2007	Version 1.1 released. Made minor modifications to indicate Framework Extension.
24-May-2006	Version 1.0 – Instructions and Template Released.

Contents

Introduction 1
Use of the Test Plan 1
Section 1. Overview 2
Section 2. Test Methodology 2
2.1 Elements of Testing 2
2.2 Types of Testing 3
2.3 Phases of Testing 7
Section 3. Test Schedule 10
Section 4. Test Monitoring and Reporting 11
4.1 Monitoring 11
4.2 Reporting 11
Section 5. References 11
Section 6. Glossary 11
Section 7. Revision History 12
Section 8. Appendices 12

Introduction

The Test Plan is included within the within the System Development Life Cycle (SDLC) Extension of the Texas Project Delivery Framework (Framework) to establish a consistent method for planning and executing testing of information technology (IT) systems. The Test Plan describes the overall planning efforts and test approach for all testing during a project. The Test Plan is used to manage the test effort for information resource projects.

Testing is the process used to identify the correctness, completeness**,** and quality of an IT system. It involves any activity performed to evaluate an attribute or capability of a system or component of a system and determine if it meets its expected and required results. Test planning reduces project risk. A well-planned and executed test effort can reduce project risk by reducing uncertainty in implementation of the system or system component.

Use of the Test Plan

Within the Framework, the Test Plan is initiated in the Project Planning Review Gate and completed, reviewed, and approved at a project level, and executed in the Implementation Review Gate.

The Test Plan Template should be used to develop a Test Plan for each project. The format of the Test Plan Template serves as a basis for creating an actual project document.

A Test Plan is used to:

- Identify and describe the test phases to be executed
- Identify the general objective, focus, test types, staffing, environment, and entry, suspension, and exit criteria for each test phase
- Describe the methodology for each test phase including testing types to be executed and the elements of testing to be used to determine correctness of the system or system component
- Define the schedule of test activities, deliverables, and milestones
- Define the monitoring activities and milestones required to evaluate actual progress to plan
- Define the reports that will be produced to communicate the progress of test execution and test results
- Manage test preparation and execution

The Test Plan should be developed in coordination with and be accessible by appropriate project team and stakeholder entities. In addition, all schedule and work plan activities and roles and responsibilities required for the execution of the Test Plan should be integrated into the Project Plan. All information in the Test Plan should be consistent with the Project Plan and the related project documents.

The Test Plan should be updated to maintain consistency with other project documents throughout the life of the project and be assessed periodically for effectiveness.

Section 1. Overview

Describe the high-level approach to testing for the project. Describe the test objectives, test scope, and approach to testing adopted by the project team.

Section 2. Test Methodology

In the following subsections, describe a test methodology that addresses both the types of tests to be performed and the phases of testing.

Testing ensures that a particular feature or functionality in a work product is correct. Testing requires both a standard for correctness and a means of determining correctness. The standard for correctness is typically a requirement. A requirement may include a specific description of functionality, a defined level of service, or a prescribed format. The means of determining correctness is described below.

2.1 Elements of Testing

Describe the elements of testing for the project.

According to accepted best practices, four specific elements of testing are used to determine correctness: inspection, analysis, demonstrations, and test.

- **Inspection** – Inspection is the direct perception of the correctness of the item under test. Examples include viewing a graphical user interface-based entry form to ensure that the field labels are correct, viewing a printed report to ensure that the format and content are correct, listening to an audio output of an accessibility-enabled user interface to ensure that the audio messages are correct, or other examples where the immediate perception of the item under test will reveal whether it is correct.

- **Analysis** – Analysis is the assessment, calculation, or breakdown of an item under test to determine that the perceived information is correct. Analysis is the process of taking the immediate data or presentation and either decomposing it or combining it with other information to ensure that the item under test is correct. Examples of analysis include the manual calculation of a result displayed by the item under test, using the source data used by the test item to ensure that the item under test performs the calculation correctly, the investigation to determine whether a business rule has been applied correctly, or other examples which validate that information provided by or functions performed by the item under test are correctly derived.

- **Demonstration** – Demonstration is the simple execution of a function in the item under test. In demonstration, the item under test initially resides in one state and is triggered to perform some function that either results in an output and/or causes the item under test to move to a new state. Examples of demonstration include accepting a form, generating a report, posting

to an account, saving a file, or other actions that are the direct result of triggering an event. Like inspection, the correctness of a demonstration can be immediately perceived; but unlike inspection, an event must occur to cause the item under test to perform its function.

- **Test** – Test is the methodical (and typically documented) combination of a set of inspections, analyses, and demonstrations that provide a well-defined test procedure to ensure the correctness of some functionality in the item under test. Usually, a complex function cannot be assessed for correctness by a single inspection, analysis, or demonstration. Most commonly, a series of steps must be performed that include some ordered set of demonstrations with inspections and analysis to verify functionality.

The elements of testing are applicable to each of the types of tests described below.

2.2 Types of Testing

In the following subsections describe the types of testing which correlate to the various types of requirements for the system or system component. Identify test tools to be used in the testing. Include a description of each tool and an explanation of the intended use of each tool.

2.2.1 Functional Testing

Describe the functional testing for the project.

Functional testing ensures that the system or system component correctly performs its intended function. In addition to a description of the function to be performed, functional requirements may include requirements for allowable or unallowable inputs and outputs and specific operations to be performed in satisfying the function.

If functionality will be tested against use cases, the actor, pre-conditions, post-conditions, scenarios, and alternate scenarios should be validated in testing to ensure that the functionality satisfies the use case.

Functional testing should address these factors for each function to be tested. Each functional test should have a unique identifier and each functional test should trace to the function or design element (and its unique identifier) to be tested.

2.2.2 Data Testing

Describe the data testing for the project.

Testing of data includes both the testing of the data contained within the system and the testing of the data used by the system.

Internal data may be in the form of constants, rules, or other data that is either static or seldom changes during the production use of the system (e.g., an IP address for a web-based application).

External data includes both data that is accepted by the system and data that is generated by the system, such as user input, communications with other applications, database sources, constructed data, and reports.

Testing of data ensures that the item under test accepts or delivers all and only the data intended for that item, in the form, format, and frequency that is correct for that item.

Each data test should have a unique identifier, and each data test should trace to the function (and its unique identifier) that describes the data to be tested.

2.2.3 User Testing

Describe the user testing for the project.

User testing ensures that the item under test meets usability, accessibility, and user documentation requirements.

User testing for usability and accessibility addresses the required behavior of the user interfaces. These should be documented in a set of specific requirements that ensure the testability of the user interface. The ultimate success of systems that are heavily dependent upon user interfaces often rests upon the clarity of the requirements for the user interface and the success of the system in meeting those user interface requirements.

User interface testing should ensure that the user interface meets the quantified and specific look and feel requirements, including any performance or ease-of-use requirements that are documented in specific, testable terms (e.g., requiring a system to be easy to use is a highly subjective and therefore less testable requirement than requiring a system to provide a display response time of less than or equal to one second.).

Testing of user documentation includes testing of online documentation, help, and other support text to ensure it is correct and adequately supports the needs of the typical users. User documentation testing also includes testing of the system according to any written documentation to ensure that the documentation correctly describes the component or system functionality (e.g., installation instructions for the system should be tested by performing an installation according to the instructions).

Each user test should have a unique identifier, and each user test should trace to the user requirement (and its unique identifier) that describes the user function to be tested.

2.2.4 Non-functional (Systems Requirements) Testing

In the following subsections, describe the non-functional or systems requirements testing for the project.

Non-functional or systems requirements testing validates required qualities or properties of the system that address how well some behavioral or structural aspect of the system should be

accomplished. Non-functional tests include: performance testing, quality testing, and interface testing, among others.

2.2.4.1 Performance Testing

Describe the performance testing for the project.

Performance testing ensures that the item under test meets specified requirements for throughput, number of users, response times, maximum workloads, and other performance characteristics of the item.

Performance testing may include load, stress, and availability testing. Load testing determines the ability of the item under test to support the performance requirements while under increased use up to the stated bounds of its capability. Stress testing determines the ability of the item under test to deal with situations when loads exceed the bounds of the stated capability (e.g., whether the item locks up, fails, processes in a degraded fashion, or otherwise changes performance). Availability testing determines whether the item under test is available according to its stated availability requirements (e.g., 24x7 ability to log in to the application).

Load and stress typically have a direct impact on availability and performance testing and typically addresses each individually as well as in combination. For example, a performance test may test an item under load with transient stresses while looking simultaneously at the effect of load and stress on availability.

Each performance test should have a unique identifier and each should trace to the performance requirement (and its unique identifier) that describes the performance requirement to be tested.

2.2.4.2 Quality Testing

Describe the quality testing for the project.

Quality validation includes the following subcategories:

- **Correctness** – Correctness is the extent to which specifications are satisfied and mission objectives are fulfilled.
- **Efficiency** – Efficiency is the relationship between the level of performance of the product and the amount of resources used, under stated conditions.
- **Flexibility** – Flexibility is the effort required to modify operational product.
- **Integrity/Security** – Integrity/Security is the extent to which access to the system or data by unauthorized personnel can be controlled.
- **Interoperability** – Interoperability is the effort needed to couple one system with another.

- **Maintainability** – Maintainability is the effort required to locate and correct an error during operation.
- **Portability** – Portability is the effort needed to transfer from one hardware or software environment to another.
- **Reliability** – Reliability is the extent to which the system performs with required precision and robustly responds to reliability challenges (e.g., through fail over or degraded operation).
- **Reusability** – Reusability is the extent to which the system and associated artifacts can be reused in another application.
- **Testability** – Testability is the effort needed to test to ensure software performs as intended.
- **Usability** – Usability is the effort required to learn, operate, prepare input for, and interpret output from the system.

Though many quality requirements for a system may be general to the system, some may be specific to particular function(s) or component(s) of the system. Any associated testing for these quality requirements must address the scope and limitations of the particular requirement. For example, the entire system may have a general security requirement that prohibits unauthorized users from accessing the system. Conversely, a portability requirement may be applicable only to a specific user interface of a system.

Specific quality requirements may also be mutually limiting or may compete with performance or functional requirements. For example, maintainability and security may conflict at times, since making a system more secure may also decrease its maintainability. Similarly, portability is often at odds with systems that require extremely high performance, since tuning a system for performance may make it less portable. Unless care is taken in test planning and design, these issues tend toward less objective criteria for testing quality requirements.

For each quality requirement that is satisfied through testing (as opposed to other forms of validation), the test should have a unique identifier, and each should trace to the quality requirement (and its unique identifier) to be tested.

2.2.4.3 Interface Testing

Describe the interface testing for the project.

Interface testing determines the correctness of the defined interfaces for the system. These interfaces may be internal to the system or with interfaces with other systems. The interfaces may be between software and hardware components or software and software components. The interface typically has a specific format, message set, and protocol. Testing the interface

according to the requirement will necessitate testing the interface features and any performance and error detection/recovery mechanisms for the interface.

Each interface test should have a unique identifier, and each should trace to the interface definition and requirement and its unique identifier.

2.2.4.4 Other Testing

Describe the other testing for the project that is not covered in the types of tests described in previous sections.

Other testing includes testing of features or requirements that are deemed to require testing that are not covered in the types of tests described above. All other tests should have a unique identifier and each test should trace to the definition, requirement, and its unique identifier.

2.3 Phases of Testing

Provide an overview of the test effort for the project by completing the Test Phase Chart provided in the template. Identify the planned test phases and provide the general objective, focus, test types, staffing, environment, entry criteria, suspension criteria, and exit criteria for each phase.

Test Phase Chart

Test Phases	Unit	Integration	System	Acceptance	X Phase
Objective	To test units of code that are considered complete	To ensure that aggregates of units perform accurately together	To ensure that the system performs according to documented requirements and the customer's expectations	To ensure that the completed system performs according to documented requirements and the customer's expectations	To …
Focus	Correctness of specific functionality of a unit and its input, outputs, and primary and fault handling	Correctness of the aggregate with regard to its associated requirements	Correctness of the system and that it conforms to stated requirements	Correctness of all functionality of the system	Correct-ness of…
Test Types/ Subtypes	Functional (low-level), Data, Performance, Integrity/Security, Interface	Functional, Data, Performance, Reliability, Integrity/Security, Interface, Usability	Functional, Data, Performance, Reliability, Integrity/Security, Interface, Usability	Functional, Data, Performance, Reliability, Integrity/Security, Interface, Usability	…

Test Phases	Unit	Integration	System	Acceptance	X Phase
Staffing	Development Team (author of a unit should not be the tester)	Test Team, independent of the development team (member of the integration test team should not perform revisions to code he is testing)	Test Team, independent of the development team (member of the system test team should not perform revisions to code he is testing)	Customers or end users, can be lead by testers independent of the development team (member of the acceptance test team should not perform revisions to code he is testing)	…
Environ-ment	Development	Should be performed in an environment segregated from development and controlled by a group independent of the development team	Should be performed in an environment segregated from development and controlled by a group independent of the development team	• Should be performed in an environment segregated from development and controlled by a group independent of the development team • Should be identical to the product-ion environment or as close as possible	...

Test Phases	Unit	Integration	System	Acceptance	X Phase
Entry Criteria	• Code complete • Test plan approved • Unit test procedure and scenarios approved • Unit test data approved	• Code complete • Test plan approved • Unit test successful • Integration test procedure and scenarios approved • Integration test data approved • Integration test exit criteria, including allowable errors and functional discrepancies, specified	• Code and documentation baselined • Test plan approved • Integration test successful • System test procedure and scenarios approved • System test data approved • System test exit criteria, including allowable errors and functional discrepancies, specified • Unit and Integration test reports complete • Requirements Traceability Matrix complete • User documentation complete • Test Readiness Review minutes complete	• Code and documentation baselined • Test plan approved • Integration test successful • Acceptance test procedure and scenarios approved • Acceptance test data approved • Acceptance test exit criteria, including allowable errors and functional discrepancies, specified • Unit, Integration, and System test reports complete	…

Test Phases	Unit	Integration	System	Acceptance	X Phase
Suspen-sion Criteria	• Environment Issues • Design rework • Development rework • Requirements changes	• Unit development not complete • Unit test not complete, unsuccessful, or inadequate • Environment issues • Design rework • Development rework • Requirement changes • Defects encountered > X • Fault with major feature that prevents significant functionality from being tested	• Environment issues • Design rework • Development rework • Requirement changes • Defects encountered > X • Installation problems • Fault with major feature that prevents significant functionality from being tested	• Environment issues • Design rework • Development rework • Requirement changes • Defects encountered > X • Installation problems • Fault with major feature that prevents significant functionality from being tested	…
Exit Criteria	• Test logs approved • Unit test scenarios passed • Unit is baselined	• Test logs approved • Allowable errors and functional discrepancies • Allowable errors and functional discrepancies do not exceed thresholds	• Test logs approved • Allowable errors and functional discrepancies do not exceed thresholds • System is baselined	• Test logs approved • Stakeholder acceptance	…

Section 3. Test Schedule

Provide a reference to the location of the project test schedule information or specify the project test schedule information that establishes the sequence and coordination for the project's test activities. State the sequence and dependencies among all test activities and the relationship of key test activities to project milestones or events. Cover the duration of the Test Plan and include all major milestones of the project related to test activities. Include resources, prerequisites, and start/completion dates for each activity, deliverable, and milestone.

Section 4. Test Monitoring and Reporting

4.1 Monitoring

Describe monitoring activities and milestones that will be used to evaluate actual progress to plan. Execution of the test plan must be monitored to recognize deviations from plan. Testing progress and success depends on realistic, detailed test planning and frequent, interim milestones at which actual progress can be compared with the plan to identify deviations.

Commonly monitored test metrics include percent of system tested, test cases executed/passed/failed/not executed, number and severity of problems reported, number of problems closed/resolved/retested, and cost of test effort (estimated versus actual).

Examination and evaluation of these metrics should include trend analysis. Trend analysis of metrics provides the ability to make comparisons between the project's metrics from other test phases or cycles and between metrics from other projects. Trend analysis can provide substantial information about the quality of the product or product segment.

4.2 Reporting

Describe reports that will be used during testing. Each test phase should have one or more reports to document test execution and results. The reports will:

- Identify the items tested
- Summarize the evaluation of the planned test items (expected versus actual, including the impact of variances)
- Indicate the version/revision level of the software tested
- Indicate the environment in which the testing took place
- Contain references to the test plan, test scenario, test procedure, test log, and problem reports, if they exist
- Specify metrics that were monitored during the testing effort, including any trend analysis compiled
- Contain a comprehensive test evaluation summary, including conclusions regarding the quality and stability of the product.

Section 5. References

Identify the information sources referenced and utilized in developing the Test Plan. Include for each the document number, title, date, and author.

Section 6. Glossary

Define all terms and acronyms required to interpret the Test Plan properly.

Section 7. Revision History

Identify changes to the Test Plan.

Section 8. Appendices

Include any relevant appendices.