<TRANSPOND>
TEST PLAN

Table of Contents

1 Introduction

This test approach document describes the appropriate strategies, process, workflows and methodologies used to plan, organize, execute and manage testing of software projects within caBIG.

1.1 Scope

Describe the current test approach scope based on your role and project objectives.

1.1.1 In Scope

The caBIG <workspace name> <system name> *Test Plan* defines the unit, integration, system, regression, and Client Acceptance testing approach. The test scope includes the following:

- Testing of all functional, application performance, security and use cases requirements listed in the *Use Case* document.
- Quality requirements and fit metrics<system name>.
- End-to-end testing and testing of interfaces of all systems that interact with the <system name>.

1.1.2 Out of Scope

The following are considered out of scope for caBIG <workspace name> <system name> system Test Plan and testing scope:

- Functional requirements testing for systems outside <application name>
- Testing of Business SOPs, disaster recovery and Business Continuity Plan.

1.2 Quality Objective

1.2.1 Primary Objective

A primary objective of testing application systems is to: ***assure that the system meets the full requirements, including quality requirements (AKA: Non-functional requirements) and fit metrics for each quality requirement and satisfies the use case scenarios and maintain the quality of the product.*** At the end of the project development cycle, the user should find that the project has met or exceeded all of their expectations as detailed in the requirements.

Any changes, additions, or deletions to the requirements document, Functional Specification, or Design Specification will be documented and tested at the highest level of quality allowed within the remaining time of the project and within the ability of the test team.

1.2.2 Secondary Objective

The secondary objective of testing application systems will be to: ***identify and expose all issues and associated risks, communicate all known issues to the project team, and ensure that all issues are addressed in an appropriate matter before release.*** As an objective, this requires careful and methodical testing of the application to first ensure all areas of the system are scrutinized and, consequently, all issues (bugs) found are dealt with appropriately.

-

1.3 Roles and Responsibilities

Roles and responsibilities may differ based on the actual SOW. Below listed functions are for testing phase.

1.3.1 Developer

An NCI-designated Cancer Center selected and funded by NCICB to participate in a specific Workspace to undertake software or solution development activities. Responsible to:

(a) Develop the system/application

(b) Develop Use cases and requirements in collaboration with the Adopters

(c) Conduct Unit, system, regression and integration testing

(d) Support user acceptance testing

1.3.2 Adopter

An NCI-designated Cancer Center selected and funded by NCICB to undertake formal adoption, testing, validation, and application of products or solutions developed by Workspace Developers. Responsible to:

(a) Contribute to Use case, requirement development through review

(b) Contribute to develop and execution of the development test scripts through review

(c) Conduct Full User Acceptance, regression, and end-to-end testing; this includes identifying testing scenarios, building the test scripts, executing scripts and reporting test results

1.3.3 Testing Process Management Team

Include NCI, BAH and Cancer Center Leads allocated to the <workspace name>. Group responsible to manage the entire testing process, workflow and quality management with activities and responsibilities to:

(a) Monitor and manage testing integrity and Support testing activities

(b) Coordinate activities across cancer centers

Add more as appropriate to testing scope

1.4 Assumptions for Test Execution

Below are some minimum assumptions (in black) that has be completed with some examples (in red). Any example may be used if deemed appropriate for the particular project. New assumptions may also be added that are reasoned to be suitable to the project.

- For User Acceptance testing, the Developer team has completed unit, system and integration testing and met all the Requirement's (including quality requirements) based on Requirement Traceability Matrix.
- User Acceptance testing will be conducted by End-users
- Test results will be reported on daily basis using Gforge. Failed scripts and defect list from Gforge with evidence will be sent to Developer directly.
- Use cases have been developed by Adopters for User Acceptance testing. Use cases are approved by test lead.
- Test scripts are developed and approved.
- Test Team will support and provide appropriate guidance to Adopters and Developers to conduct testing
- Major dependencies should be reported immediately after the testing kickoff meeting.

1.5 Constraints for Test Execution

Below are some minimum assumptions (in black) followed by example constraints (red). Any example may be used if deemed appropriate for the particular project. New constraints may also be added that are reasoned to be suitable to the project.

- Adopters should clearly understand on test procedures and recording a defect or enhancement. Testing Process Management Team will schedule a teleconference with Developers and Adopters to train and address any testing related issues.
- Developer will receive consolidated list of request for test environment set up, user accounts set up, data set (actual and mock data), defect list, etc. through **GForge** after the initial Adopter testing kick off meeting.
- Developer will support ongoing testing activities based on priorities
-
- Test scripts must be approved by Test Lead prior test execution
- Test scripts, test environment and dependencies should be addressed during testing kickoff meeting in presence of a SME and request list should be submitted within 3 days of the kickoff meeting

- The Developer cannot execute the User Acceptance and End to End test scripts. After debugging, the developer can conduct their internal test, but no results from that test can be recorded / reported.
- Adopters are responsible to identify dependencies between test scripts and submit clear request to set up test environment

1.6 Definitions

Bugs: Any error or defect that cause the software/application or hardware to malfunction. That is also included in the requirements and does not meet the required workflow, process or function point.

Enhancement:
1) Any alteration or modification to the existing system for better workflow and process.
2) An error or defect that causes the software/application or hardware to malfunction.
Where 1) and 2) is NOT included in the requirements can be categorized as an enhancement.

Enhancement can be added as a new requirement after appropriate Change Management process.

2 Test Methodology

2.1 Purpose

2.1.1 Overview

The below list is not intended to limit the extent of the test plan and can be modified to become suitable for the particular project.

The purpose of the Test Plan is to achieve the following:

- Define testing strategies for each area and sub-area to include all the functional and quality (non-functional) requirements.
- Divide Design Spec into testable areas and sub-areas (do not confuse with more detailed test spec). Be sure to also identify and include areas that are to be omitted (not tested) also.
- Define bug-tracking procedures.
- Identify testing risks.
- Identify required resources and related information.
- Provide testing Schedule.

2.1.2 Usability Testing

The purpose of usability testing is to ensure that the new components and features will function in a manner that is acceptable to the customer.

Development will typically create a non-functioning prototype of the UI components to evaluate the proposed design. Usability testing can be coordinated by testing, but actual testing must be performed by non-testers (**as close to end-users as possible).** Testing will review the findings and provide the project team with its evaluation of the impact these changes will have on the testing process and to the project as a whole.

2.1.3 Unit Testing (Multiple)

Unit Testing is conducted by the Developer during code development process to ensure that proper functionality and code coverage have been achieved by each developer both during coding and in preparation for acceptance into iterations testing.

The following are the example areas of the project must be unit-tested and signed-off before being passed on to regression Testing:

- Databases, Stored Procedures, Triggers, Tables, and Indexes
- NT Services
- Database conversion
- .OCX, .DLL, .EXE and other binary formatted executables

2.1.4 Iteration/Regression Testing

During the repeated cycles of identifying bugs and taking receipt of new builds (containing bug fix code changes), there are several processes which are common to this phase across all projects. These include the various types of tests: functionality, performance, stress, configuration, etc. There is also the process of communicating results from testing and ensuring that new drops/iterations contain stable fixes (regression). The project should plan for a minimum of 2-3 cycles of testing (drops/iterations of new builds).

At each iteration, a debriefing should be held. Specifically, the report must show that to the best degree achievable during the iteration testing phase, all identified severity 1 and severity 2 bugs have been communicated and addressed. At a minimum, all priority 1 and priority 2 bugs should be resolved prior to entering the beta phase.

Below are examples. Any example may be used if deemed appropriate for the particular project. New content may also be added that are reasoned to be suitable to the project. Important deliverables required for acceptance into Final Release testing include:

- Application SETUP.EXE
- Installation instructions
- All documentation (beta test scripts, manuals or training guides, etc.)

2.1.5 Final release Testing

Testing team with end-users participates in this milestone process as well by providing confirmation feedback on new issues uncovered, and input based on identical or similar issues detected earlier. The intention is to verify that the product is ready for distribution, acceptable to the customer and iron out potential operational issues.

Assuming critical bugs are resolved during previous iterations testing- Throughout the Final Release test cycle, bug fixes will be focused on minor and trivial bugs (severity 3 and 4).

Testing will continue its process of verifying the stability of the application through regression testing (existing known bugs, as well as existing test cases).

The milestone target of this phase is to establish that the application under test has reached a level of stability, appropriate for its usage (number users, etc.), that it can be released to the end users and caBIG community.

2.1.6 Testing completeness Criteria

Release for production can occur only after the successful completion of the application under test throughout all of the phases and milestones previously discussed above.

The milestone target is to place the release/app (build) into production after it has been shown that the app has reached a level of stability that meets or exceeds the client expectations as defined in the Requirements, Functional Spec., and caBIG Production Standards.

2.2 Test Levels

Testing of an application can be broken down into three primary categories and several sub-levels. The three primary categories include tests conducted every build (Build Tests), tests conducted every major milestone (Milestone Tests), and tests conducted at least once every project release cycle (Release Tests). The test categories and test levels are defined below:

2.2.1 Build Tests

2.2.1.1 Level 1 - Build Acceptance Tests

Build Acceptance Tests should take less than 2-3 hours to complete (15 minutes is typical). These test cases simply ensure that the application can be built and installed successfully. Other related test cases ensure that adopters received the proper Development Release Document plus other build related information (drop point, etc.). The objective is to determine if further testing is possible. If any Level 1 test case fails, the build is returned to developers un-tested.

2.2.1.2 Level 2 - Smoke Tests

Smoke Tests should be automated and take less than 2-3 hours (20 minutes is typical). These tests cases verify the major functionality a high level.
The objective is to determine if further testing is possible. These test cases should emphasize breadth more than depth. All components should be touched, and every major feature should be tested briefly by the Smoke Test. If any Level 2 test case fails, the build is returned to developers un-tested.

2.2.1.3 Level 2a - Bug Regression Testing

Every bug that was “Open” during the previous build, but marked as “Fixed, Needs Re-Testing” for the current build under test, will need to be regressed, or re-tested. Once the smoke test is completed, all resolved bugs need to be regressed. It should take between 5 minutes to 1 hour to regress most bugs.

2.2.2 Milestone Tests

2.2.2.1 Level 3 - Critical Path Tests

Critical Path test cases are targeted on features and functionality that the user will see and use every day.
Critical Path test cases must pass by the end of every 2-3 Build Test Cycles. They do not need to be tested every drop, but must be tested at least once per milestone. Thus, the Critical Path test cases must all be executed at least once during the Iteration cycle, and once during the Final Release cycle.

2.2.3 Release Tests

2.2.3.1 Level 4 - Standard Tests

Test Cases that need to be run at least once during the entire test cycle for this release. These cases are run once, not repeated as are the test cases in previous levels. Functional Testing and Detailed Design Testing (Functional Spec and Design Spec Test Cases, respectively). These can be tested multiple times for each Milestone Test Cycle (Iteration, Final Release, etc.).
Standard test cases usually include Installation, Data, GUI, and other test areas.

2.2.3.2 Level 5 - Suggested Test

These are Test Cases that would be nice to execute, but may be omitted due to time constraints.
Most Performance and Stress Test Cases are classic examples of Suggested test cases (although some should be considered standard test cases). Other examples of suggested test cases include WAN, LAN, Network, and Load testing.

2.3 Bug Regression

Bug Regression will be a central tenant throughout all testing phases.
All bugs that are resolved as “Fixed, Needs Re-Testing” will be regressed when Testing team is notified of the new drop containing the fixes. When a bug passes regression it will be considered “Closed, Fixed”. If a bug fails regression, adopters testing team will notify development team by entering notes into GForge. **When a Severity 1 bug fails regression, adopters Testing team should also put out an immediate email to development.** The Test Lead will be responsible for tracking and reporting to development and product management the status of regression testing.

2.4 Bug Triage

Bug Triages will be held throughout all phases of the development cycle. Bug triages will be the responsibility of the Test Lead. Triages will be held on a regular basis with the time frame being determined by the bug find rate and project schedules.
Thus, it would be typical to hold few triages during the **Planning phase**, then maybe one triage per week during the **Design phase**, ramping up to twice per week during the latter **stages of the Development phase**. Then, the Stabilization phase should see a substantial reduction in

the number of new bugs found, thus a few triages per week would be the maximum (to deal with status on existing bugs).

The Test Lead, Product Manager, and Development Lead should all be involved in these triage meetings. The Test Lead will provide required documentation and reports on bugs for all attendees. The purpose of the triage is to determine the type of resolution for each bug and to prioritize and determine a schedule for all "To Be Fixed Bugs'. Development will then assign the bugs to the appropriate person for fixing and report the resolution of each bug back into the GForge bug tracker system. The Test Lead will be responsible for tracking and reporting on the status of all bug resolutions.

2.5 Suspension Criteria and Resumption Requirements

This section should be defined to list criteria's and resumption requirements should certain degree and pre-defined levels of test objectives and goals are not met.
Please see example below:
- Testing will be suspended on the affected software module when Smoke Test (Level 1) or Critical Path (Level 2) test case bugs are discovered after the 3rd iteration.
- Testing will be suspended if there is critical scope change that impacts the Critical Path

A bug report should be filed by Development team. After fixing the bug, Development team will follow the drop criteria (described above) to provide its latest drop for additional Testing. At that time, adopters will regress the bug, and if passes, continue testing the module.

2.6 Test Completeness

Testing will be considered complete when the following conditions have been met:

2.6.1 Standard Conditions:

- When Adopters and Developers, agree that testing is complete, the app is stable, and agree that the application meets functional requirements.
- Script execution of all test cases in all areas have passed.
- Automated test cases have in all areas have passed.
- All priority 1 and 2 bugs have been resolved and closed.
- NCI approves the test completion
- Each test area has been signed off as completed by the Test Lead.
- 50% of all resolved severity 1 and 2 bugs have been successfully re-regressed as final validation.
- Ad hoc testing in all areas has been completed.

2.6.2 Bug Reporting & Triage Conditions:

Please add Bug reporting and triage conditions that will be submitted and evaluated to measure the current status.

- Bug find rate indicates a decreasing trend prior to Zero Bug Rate (no new Sev. 1/2/3 bugs found).

- Bug find rate remains at 0 new bugs found (Severity 1/2/3) despite a constant test effort across 3 or more days.
- Bug severity distribution has changed to a steady decrease in Sev. 1 and 2 bugs discovered.
- No 'Must Fix' bugs remaining prior despite sustained testing.

3 Test Deliverables

The following page contains a matrix depicting all of the deliverables that Testing will use.

3.1 Deliverables Matrix

Below is the list of artifacts that are process driven and should be produced during the testing lifecycle. Certain deliverables should be delivered as part of test validation, you may add to the below list of deliverables that support the overall objectives and to maintain the quality.
This matrix should be updated routinely throughout the project development cycle in you project specific Test Plan.

Deliverable
Documents
Test Approach
→ Test Plan
→ Test Schedule
→ Test Specifications
Test Case / Bug Write-Ups
Test Cases / Results
Test Coverage Reports
GForge Bug tracker for bug reporting
Reports
Test results report
Test Final Report - Sign-Off

3.2 Documents

3.2.1 Test Approach Document

The Test Approach document is derived from the Project Plan, Requirements and Functional Specification documents. This document defines the overall test approach to be taken for the project. The Standard Test Approach document that you are currently reading is a boilerplate from which the more specific project Test Approach document can be extracted.

When this document is completed, the Test Lead will distribute it to the Product Manager, Development Lead, User Representative, Program Manager, and others as needed for review and sign-off.

3.2.2 Test Plan

The Test Plan is derived from the Test Approach, Requirements, Functional Specs, and detailed Design Specs. The Test Plan identifies the details of the test approach, identifying the associated test case areas within the specific product for this release cycle.
The purpose of the Test Plan document is to:

- Specify the approach that Testing will use to test the product, and the deliverables (extract from the Test Approach).
- Break the product down into distinct areas and identify features of the product that are to be tested.
- Specify the procedures to be used for testing sign-off and product release.
- Indicate the tools used to test the product.
- List the resource and scheduling plans.
- Indicate the contact persons responsible for various areas of the project.
- Identify risks and contingency plans that may impact the testing of the product.
- Specify bug management procedures for the project.
- Specify criteria for acceptance of development drops to testing (of builds).

3.2.3 Test Schedule

This section is not vital to the document as a whole and can be modified or deleted if needed by the author.
The Test Schedule is the responsibility of the Test Lead (or Department Scheduler, if one exists) and will be based on information from the Project Scheduler (done by Product Manager). The project specific Test Schedule may be done in MS Project.

3.2.4 Test Specifications

A Test Specification document is derived from the Test Plan as well as the Requirements, Functional Spec., and Design Spec documents. It provides specifications for the construction of Test Cases and includes list(s) of test case areas and test objectives for each of the components to be tested as identified in the project's Test Plan.

3.2.5 Requirements Traceability Matrix

A Requirements Traceability Matrix (RTM) which is used to link the test scenarios to the requirements and use cases is a required part of the Test Plan documentation for all projects. Requirements traceability is defined as the ability to describe and follow the life of a requirement, in both a forward and backward direction (i.e. from its origins, through its development and specification, to its subsequent deployment and use, and through periods of ongoing refinement and iteration in any of these phases). [1]

1

Attached is a sample basic RTM which could provide a starting point for this documentation. The important thing is to choose a template or document basis that achieves thorough traceability throughout the life of the project.

3.3 Reports

The Test Lead will be responsible for writing and disseminating the following reports to appropriate project personnel as required.

3.3.1 Testing status reports

A weekly or bi-weekly status report will be provided by the Test Lead to project personnel. This report will summarize weekly testing activities, issues, risks, bug counts, test case coverage, and other relevant metrics.

3.3.2 Phase Completion Reports

When each phase of testing is completed, the Test Lead will distribute a Phase Completion Report to the Product manager, Development Lead, and Program Manager for review and sign-off.
The below bullets illustrates an example of what the document may include.
The document must contain the following metrics:

- Total Test Cases, Number Executed, Number Passes / Fails, Number Yet to Execute
- Number of Bugs Found to Date, Number Resolved, and Number still Open
- Breakdown of Bugs by Severity / Priority Matrix
- Discussion of Unresolved Risks
- Discussion of Schedule Progress (are we where we are supposed to be?)

3.3.3 Test Final Report - Sign-Off

A Final Test Report will be issued by the Test Lead. It will certify as to the extent to which testing has actually completed (test case coverage report suggested), and an assessment of the product's readiness for Release to Production.

3.4 Responsibility Matrix

Insert testing resource matrix – who is involved and what is the role.

4 Resource & Environment Needs

4.1 Testing Tools

4.1.1 Tracking Tools

GForge bug tracker is used by caBIG to enter and track all bugs and project issues. The Test Lead is responsible for maintaining the GForge database.

4.1.1.1 Configuration Management

Please include your configuration management plan

4.2 Test Environment

4.2.1 Hardware

Include the minimum hardware requirements that will be used to test the Application. Testing will have access control to one or more application/database servers separate from any used by non-test members of the project team. Testing will also have access control to an adequate number of variously configured PC workstations to assure testing a range from the minimum to the recommended client hardware configurations listed in the project's Requirements, Functional Specification and Design Specification documents.

4.2.2 Software

In addition to the application and any other customer specified software, the following list of software should be considered a minimum:

- NT Workstation 4.0 or Windows 95.
- MS Office 95 Professional
- MS Exchange
- TCM (Testing Tool Server)
- Task Manager (Testing Tool Server)

4.3 Bug Severity and Priority Definition

Bug Severity and Priority fields are both very important for categorizing bugs and prioritizing if and when the bugs will be fixed. The bug Severity and Priority levels will be defined as outlined in the following tables below. Testing will assign a severity level to all bugs. The Test Lead will be responsible to see that a correct severity level is assigned to each bug.

The Test Lead, Development Lead and Program Manager will participate in bug review meetings to assign the priority of all currently active bugs. This meeting will be known as "Bug Triage Meetings". The Test Lead is responsible for setting up these meetings on a routine basis to address the current set of new and existing but unresolved bugs.

4.3.1 Severity List

The tester entering a bug into GForge is also responsible for entering the bug Severity.

Severity ID	Severity	Severity Description
1	Critical	The module/product crashes or the bug causes non-recoverable conditions. System crashes, GP Faults, or database or file corruption, or potential data loss, program hangs requiring reboot are all examples of a Sev. 1 bug.
2	High	Major system component unusable due to failure or incorrect functionality. Sev. 2 bugs cause serious problems such as a lack of functionality, or insufficient or unclear error messages that can have a major impact to the user, prevents other areas of the app from being tested, etc. Sev. 2 bugs can have a work around, but the work around is inconvenient or difficult.
3	Medium	Incorrect functionality of component or process. There is a simple work around for the bug if it is Sev. 3.
4	Minor	Documentation errors or signed off severity 3 bugs.

4.3.2 Priority List

Priority	Priority Level	Priority Description
5	Must Fix	This bug must be fixed immediately; the product cannot ship with this bug.
4	Should Fix	These are important problems that should be fixed as soon as possible. It would be an embarrassment to the company if this bug shipped.
3	Fix When Have Time	The problem should be fixed within the time available. If the bug does not delay shipping date, then fix it.
2	Low Priority	It is not important (at this time) that these bugs be addressed. Fix these bugs after all other bugs have been fixed.
1	Trivial	Enhancements/ Good to have features incorporated- just are out of the current scope.

4.4 Bug Reporting

Testing team recognizes that the bug reporting process is a critical communication tool within the testing process. Without effective communication of bug information and other issues, the development and release process will be negatively impacted.

The Test Lead will be responsible for managing the bug reporting process. Testing's standard bug reporting tools and processes will be used. GForge is the company-wide standard Bug Logging / Tracking tool. Testing and development will enter their data into the GForge database following the field entry definitions defined below.

5 Terms/Acronyms

The below terms are used as examples, please add/remove any terms relevant to the document.

TERM/ACRONYM	DEFINITION

TERM/ACRONYM	DEFINITION
API	Application Program Interface
BAH	Booz Allen Hamilton
BCP	Business Continuity Plan
CAT	Client Acceptance Testing
End-to End Testing	Tests user scenarios and various path conditions by verifying that the system runs and performs tasks accurately with the same set of data from beginning to end, as intended.
ER;ES	Electronic Records; Electronic Signatures
N/A	Not Applicable
NCI	National Cancer Institute
QA	Quality Assurance
RTM	Requirements Traceability Matrix
SME	Subject Matter Expert
SOP	Standard Operating Procedure
TBPT	Tissue Bank and Pathology Tools
TBD	To Be Determined
TSR	Test Summary Report