Detailed Test Plan Contents

|  |  |
| --- | --- |
| **Test Planning Area** | **Key planning concepts to consider** |
| Work Products to be tested | What work products (devices, functions, systems, applications, etc.) will be verified during the execution of this test plan? |
| Work Products not to be tested | What will be excluded from the testing? |
| Utilities, tools, and software | List testing tools and needed utilities, such as simulators or comparison utilities. List other supporting applications (including versions) needed for testing (e.g., upstream applications to create test data, downstream applications to verify test results). |
| Transition milestones | Identify the milestones to be completed prior to transition to production. Suggested milestones:* Test results and issues summary review
* Deliverables accepted by users
* Checkpoint - decide whether to move into production
* Implementation verification complete
 |
| **Testing Overview** |   |
| **Test Sets** |   |
| Test Set status | Include the results summary or provide a link to the detailed and summary results. The summary should include pass/fail status and responsible testers for every test set, along with relevant details, such as unresolved defects. |
| **Test Results Summary** |   |
| Test milestones | Identify the milestones to be completed during actual test execution. For example:* Installation/Conversion tests
* Configuration Acceptance test
* Test Sets (sub-milestones for each test set - see [#Test Sets](https://wiki.cac.washington.edu/display/TEGPM/Test%2BPlanning%2BChecklist#TestPlanningChecklist-TestSets), below)
 |
| **Test Environment** |   |
| Test approach | Outline how the tests, test environment and deliverables under test will be managed. Include how tests will be reviewed and prioritized, types of tests (e.g., parallel, automated), and expected phases (such as unit, integration, system, alpha, beta, pilot). |
| System configuration | For each test environment (unit, integration, beta, etc.), list the system components on all platforms that make up the environment (the 'system baseline'). Include servers and other hardware, databases, MCP usercodes, as well as which versions of operating systems, database systems, and other supporting applications, such as ASTRA, are needed. Keep in mind that the test environment should look as much like the production environment as possible, in order to provide the highest level of confidence in the verification of the changes. If the environments are too different, changes that appear to function well in test may fail in production. |
| **Strategy / Approach** |   |
| Status of testing issues | Track issues in the test plan or provide a link to testing issues with their status. This may be a subset of the [project issues list](https://wiki.cac.washington.edu/display/TEGPM/Issue%2BAction%2BDecision%2BLog). |
| **Staffing** |   |
| Staff requirements / roles and responsibilities | List the roles, qualifications and testing responsibilities for the testing staff, including definition of acceptance criteria and final acceptance of the results. Possible roles include:* Project manager
* Test manager/coordinator
* Development team lead
* Developer
* Tester
* Acceptance test lead
* System subject matter expert
* Business subject matter expert
* Acceptor
 |
| Specific tests, grouped into functional or other areas. | Describe the sets of tests to be run. Include the specific test cases, or provide an overview plus a link to the document(s) describing the test cases. Possible test sets:* Regression tests
* Functional tests (e.g., business logic scenarios, data handling tests)
* Security tests
* Report tests
* User interface and usability tests
* System interface tests
* Performance tests
* Volume and stress tests
* Error recovery tests (including backup and recovery)
* Fault retesting and regression (verifying that fixes applied during tests didn't break functionality)
* Documentation/help verification
* User acceptance tests
* Implementation verification tests
 |
| Requirements coverage strategy | Explain how requirements will be traced to tests and how the project team will know which requirements have had at least one associated test case executed. Include a link to the requirements/testing (test coverage) traceability matrix. Include a strategy for identifying tests that meet 'implied requirements' (it doesn't break). |
| Purpose and scope of the plan | Provide a brief statement of what systems and test phases (e.g., integration, system, alpha, beta) are addressed in the test plan. Reference the requirements which will be verified by this testing. |
| **Product Changes** |   |
| Problem reporting and test tracking procedures | Describe how tests will be tracked (executed, failed, associated defects) and the problem reporting procedures, including how problems will be prioritized. Identify any tools to be used (e.g., RT, TestTrack). |
| Pre-test milestones | Identify the milestones to be completed prior to starting test execution. Suggested milestones:* Components to be tested identified
* Risk assessment complete (new or changed components have been tagged as high, medium or low-risk, based on risk of failure in production)
* Test planning complete
* Resources obtained (system, personnel, data, other)
* Preliminary tester training complete
* Test environment ready
* Pre-Installation data preparation
 |
| Overview | What's the general description of the changes that will be tested? Provide an overview or a link. |
| Other limitations or assumptions | Are there work products that can't be tested? Are there test methods that can't be used? |
| New, changed, obsolete, affected items | List or link to the specific work products (Project Items) to be tested. Include affected items - e.g., programs that use changed components or data, 'shadow' systems, downstream applications. Each deliverable or affected item should have an associated risk (if it fails in production), needed environmental components, and summary of the changes. This list would be used to determine test priorities, track test status, and to verify production readiness. |
| **Milestones / Work Plan** | Prepare a testing work plan with major testing milestones, including the high-level functions or areas to be tested. Include estimated hours or time lines where known. |
| Maintenance - resolved defects/enhancements | List or link to bugs or enhancements targeted to be included in this release, if any. |
| **Issues** |   |
| Dependencies, risks and risk management | List any items or actions upon which the execution of the tests is dependent. List any risks if tests are not executed or fail. Reference the deliverables risk assessment.. |
| Data requirements | List the data needed for testing (e.g., a full copy of a production database from a given financial cycle). Include any specific test data requirements, such as certain transaction or employee types. |
| Contingency plan | Prepare a plan to mitigate any risks that arise if required test environment components are not available (contingency, test remediation, risks). |
| Configuration management (version control) and environment migration | Describe how the test environment will be controlled, and how the changes will be stored and migrated from the development environment into the final test environment. Identify any tools to be used (e.g., Visual SourceSafe or Visual Studio Team System Source Control for web software, SURE for Unisys software, Concurrent Versions System (CVS) for Unix software). |
| Acceptance criteria | Identify the tasks, milestones, deliverables or quality levels that must reach a given state in order for the testing phase(s) to be declared complete. This can consist of both entrance & exit criteria if the test plan covers multiple test phases. Some examples of acceptance criteria:* All high-priority defects have been corrected and any of the associated tests rerun successfully
* All outstanding (unresolved) defects have been documented and include workarounds where required
* Requirements coverage is 100% (100% of test cases addressing requirements have been executed) or discrepancies documented and acceptable
* Code coverage (the percentage of code tested) is at least 95%
* The success rate (test cases passed) is at least 95%; the failure rate is documented and acceptable
* Acceptor has signed off on test results and outstanding issues.
 |