Test Planning

by Josh Probert

1. Introduction

Software testing is a formal process carried out by a committed testing team in which a piece of software, parts of software or even multiple pieces of software are examined to detect differences between existing and required conditions. Testing is the single biggest Software Quality Assurance task with most projects allocating just under a quarter of their budget to this section alone. If successful, testing should find and eradicate any problems that arise with the relevant software. There many different types of software testing including many contrasting methodologies. Therefore, each software development project requires a substantial investment in planning. Therefore, test planning is essential in ensuring testing identifies and reveals as many errors in the software as possible, brings software to an acceptable level of quality and is efficient regarding budgetary and scheduling limitations. ANSI/IEEE Standard for Software Test Documentation defines Test Planning as “a document describing the scope, approach, resources and schedule of intended testing activities”. In this report, I will define what is involved in test planning, following the IEEE 829 Test Plan Standard.

Test planning is an ongoing process throughout the project lifecycle with test plans being developed for each phase of software development. These plans include Acceptance, Integration and Unit test plans. A software test plan enables the mapping of tests to the software requirements and defines the entry and exit criteria for each phase of testing. Test planning is one of the most important factors in successful software testing, yet it’s frequently omitted. Pressure to begin coding, or scheduling problems are often the villains behind test planning absence. Without detailed planning, issues are certain to arise when testing proceeds. Issues such as ignorance of software problems, breaching financial and scheduling limits and contrasts in expected quality and end quality. Effective test planning is very important in the software development lifecycle, “The plan is nothing, the planning is everything” Dwight D. Eisenhower

2. Level of Test Plan

The level of test plan defines what the test plan is being created for e.g. subsections of test planning. There are various levels of test planning that are specific to their own purposes. A Master Test Plan is an overall test plan document that is considered to be one the “major communication channels with all project participants” Mustafa Khan (2007). It is therefore written in a managerial style to define and explain the plan for the software testing. As there are many different types of software testing, such as Integration, Unit, Regression and Acceptance testing, the level of test plan is important to identify in terms of what the document will contain and hope to achieve. A specific test plan must exist for whatever level of testing is to proceed. The author(s) of a Test Plan differ depending on the level. For example, A Master Test plan may be co-written by all project participants, while a more technical test plan such as a Unit Test plan may be written by developers and testers only. The author(s) of test plans differ on the relevant knowledge needed for the specific content of that test plan.
3. Test Plan Structure

All test plans follow the same structure defined by the IEEE 829 standard, and differ only in content and detail. In this section the report will state and specify the typical structure of a test plan document, detailing all the information that applies to each category.

3.1. Test Plan Identifier

A test plan document will commence with a unique test plan identifier. This will be a company generated number to identify the individual test plan, its test level and the level of software it is related to. This unique identifier may also identify whether the test plan is a Master test plan, an Integration test plan, etc. The aim of the identifier is to assist in coordinating software and test ware versions. As a test plan is a software document, revision numbers of the test plan are also stated in this section.

3.2. Identifying Test Items

Identifying the test items is a section that basically specifies the things that are to be tested within the scope of this test plan, i.e. functions of the software in question. It is essentially a list of what is to be tested developed from the software requirements stated in the design section of the SDLC. Software and hardware needed for testing will also be listed here, along with other test materials and participating organizations.

3.3. Software Risk Issues

All risks associated with the software and its testing need to be identified in this section. This could include complex functions, new versions of cooperating software, etc. There are some inherit software risks such as complexity, that need to be identified. These include such risks as safety, client impact and government regulations. Misunderstanding of the original requirements is also a risk that should be considered with test planners being aware of vague, unclear or non-testable requirements.

3.4. Features to be Tested

This section identifies the features to be tested from a user’s point of view. It differs significantly in comparison to “Identifying Test Items” (3.2.) in the fact it is not a technical description of the software but a user’s view of the functions. A test planner should identify the level of risk for each function in an understandable manner i.e. High, Medium and Low. The test planner should also discuss why this level was chosen. This section in effect is a basic description of features that will be included in testing.

3.5. Features not to be Tested

This section lists the features not to be included in the testing process, identifying the reason behind its exclusion. This section is directly related to previous and future sections (3.3. and 3.13.), with what will and will not be tested being directly affected by levels of acceptable risk within the project. If a feature does not get tested it affects the level of risk of the project.
Reasons why a feature may be omitted from testing could be that it has been used before and deemed stable or that it may not be intended to be included in the release of the software.

3.6. Test Approach

This section identifies the strategy for this test plan, differing depending on the level of test plan. The approach stated should be appropriate and in agreement with all higher and lower levels of test plans. Overall rules and processes are identified here including such aspects as the use of automated testing tools. If this is the Master Test Plan, the overall testing approach and coverage requirements must also be identified. The level of detail of this section differs depending on the level of test plan. For example, a Unit test plan will go into much detail on individual unit tests and test data.

3.7. Test Pass/Fail Criteria

This section identifies the pass and fail criteria appropriate to this test plan. For a Unit test plan, this section would involve such criteria as:

- All test cases completed
- A specified percentage of cases completed with a percentage containing some number of minor defects.
- Automated testing tools indicated all lines of code covered.

A Master test plan may include:

- All lower level test plans completed
- A specified number of test plans completed without errors and a percentage with minor defects.

A successful test plan should give a clear understanding of when the project can or cannot proceed.

3.8. Suspension Criteria

A suspension criterion involves identifying when pausing during a series of tests is necessary. For example, if the number of defects reaches a point where the follow on testing has no value, it makes no sense to continue the test and waste resources. A test planner should specify what constitutes stoppage for a test and what is an acceptable number of defects to allow testing to continue. This role of go/no go decisions should be specified to a member of the testing team later in the test plan.

3.9. Test Deliverables

This section is used to specify what is to be delivered as part of this test plan. This section would identify such features as test plan documentation, test cases, testing tools and their outputs, simulators, error logs, problem reports and their corresponding corrective actions. One thing that is not a test deliverable is the software itself. The software is listed under test items and is delivered by development.
3.10. **Environmental Requirements**

Environmental requirements will state any special requirements for this test plan. This will include necessary hardware and software required for testing to proceed. Descriptions of how test data will be provided and the level of testing on each component should also be listed. Environmental requirements will describe everything needed for testing to proceed in some location. Documenting the physical components required for test execution helps to identify potential gaps in what is required and what actually exists. The test environment should be identified in detail including such as aspects as security and configurations.

3.11. **Staffing and Training Needs**

This section identifies all personnel and hierarchies relevant to the test plan. This includes such questions as who is in charge. This includes all areas of the plan such as setting risks, selecting testing and non-testing features, scheduling and most importantly critical go/no go decisions.

If necessary, testing teams are organised and stated in this section. For example, three teams involving a team leader and four testers per team. All relevant training needs are also stated in this section e.g. training needed to operate testing tools or handle special test data.

3.12. **Schedule of Test**

Scheduling should be based on realistic and validated estimates for software testing. Milestones should be identified with schedules being specified for each milestone. Scheduling should provide management with an explicit representation of all test activities. Key phases and milestones should be discussed in relation to quality assurance, commenting on standards and testing goals. Depending on the level of test, the size of this section will differ, e.g. Master test plan will involve all the test plan schedules below it making it fairly large.

It is always best to tie all test dates directly to their related development activity dates. This prevents the test team from being blamed for delays if they occur. This is called dependant/relative dating.

3.13. **Planning for Risks and Contingencies**

This section aims to identify the overall risks to the project with an emphasis on the testing process. This could be such problems as lack of personnel resources when testing is to begin. Another example could be the lack of availability of requirements, software, hardware, etc. The section should in turn identify how to plan for risks stated earlier in the test plan. This will involve procedures to counteract if possible problems to arise. For example, with regards to lack of personnel, a solution such as using a member of a test team as an administrative employee could solve this possible problem.

Approvals states who can consent a process as complete and allow the project to proceed to the next stage. This depends on the level of test plan and can differ from a test team leader to a more executive employee. At the Master test plan this may involve all parties connected to the project. When determining the approval process, a test planner must keep in mind who the audience is. The type of knowledge at each level of test plan differs significantly. For example, programmers may understand the technical side of software but not the managerial or business side.

4. Conclusion

Test planning is an essential phase of the SDLC providing detailed information about how to approach and conclude the testing of software. Testing is one of the most important phases of the SDLC, accounting for a large portion of the software’s budget. Without a detailed test plan, an end product can be bug ridden and in turn unsuccessful. Detailed test planning will ensure problems are dealt with if they arise, be it software problems or test problems. With levels of test plan providing different levels of information, developers can ensure successful testing of a product. Such a systematic approach as specified by the IEEE 829-1998 Test Plan Structure, provides developers with software specific categories to guarantee a successful software testing stage.
5. References


