

# TEST PLAN OUTLINE (IEEE 829 FORMAT)

- 1) Test Plan Identifier
- 2) References
- 3) Introduction
- 4) Test Items
- 5) Software Risk Issues
- 6) Features to be Tested
- 7) Features not to be Tested
- 8) Approach
- 9) Item Pass/Fail Criteria
- 10) Suspension Criteria and Resumption Requirements
- 11) Test Deliverables
- 12) Remaining Test Tasks
- 13) Environmental Needs
- 14) Staffing and Training Needs
- 15) Responsibilities
- 16) Schedule
- 17) Planning Risks and Contingencies
- 18) Approvals
- 19) Glossary

# IEEE TEST PLAN TEMPLATE

## 1 TEST PLAN IDENTIFIER

Some type of unique company generated number to identify this test plan, its level and the level of software that it is related to. Preferably the test plan level will be the same as the related software level. The number may also identify whether the test plan is a Master plan, a Level plan, an integration plan or whichever plan level it represents. This is to assist in coordinating software and testware versions within configuration management.

Keep in mind that test plans are like other software documentation, they are dynamic in nature and must be kept up to date. Therefore, they will have revision numbers.

You may want to include author and contact information including the revision history information as part of either the identifier section or as part of the introduction.

## 2 REFERENCES

List all documents that support this test plan. Refer to the actual version/release number of the document as stored in the configuration management system. Do not duplicate the text from other documents as this will reduce the viability of this document and increase the maintenance effort. Documents that can be referenced include:

- Project Plan
- Requirements specifications
- High Level design document
- Detail design document
- Development and Test process standards
- Methodology guidelines and examples
- Corporate standards and guidelines

## 3 INTRODUCTION

State the purpose of the Plan, possibly identifying the level of the plan (master etc.). This is essentially the executive summary part of the plan.

You may want to include any references to other plans, documents or items that contain information relevant to this project/process. If preferable, you can create a references section to contain all reference documents.

Identify the Scope of the plan in relation to the Software Project plan that it relates to. Other items may include, resource and budget constraints, scope of the testing effort, how testing relates to other evaluation activities (Analysis & Reviews), and possibly the process to be used for change control and communication and coordination of key activities.

As this is the “Executive Summary” keep information brief and to the point.

## 4 TEST ITEMS (FUNCTIONS)

These are things you intend to test within the scope of this test plan. Essentially, something you will test, a list of what is to be tested. This can be developed from the software application inventories as well as other sources of documentation and information.

This can be controlled and defined by your local Configuration Management (CM) process if you have one. This information includes version numbers, configuration requirements where needed, (especially if multiple versions of the product are supported). It may also include key delivery schedule issues for critical elements.

Remember, what you are testing is what you intend to deliver to the Client.

This section can be oriented to the level of the test plan. For higher levels it may be by application or functional area, for lower levels it may be by program, unit, module or build.

## **5 SOFTWARE RISK ISSUES**

Identify what software is to be tested and what the critical areas are, such as:

- A. Delivery of a third party product.
- B. New version of interfacing software
- C. Ability to use and understand a new package/tool, etc.
- D. Extremely complex functions
- E. Modifications to components with a past history of failure
- F. Poorly documented modules or change requests

There are some inherent software risks such as complexity; these need to be identified.

- A. Safety
- B. Multiple interfaces
- C. Impacts on Client
- D. Government regulations and rules

Another key area of risk is a misunderstanding of the original requirements. This can occur at the management, user and developer levels. Be aware of vague or unclear requirements and requirements that cannot be tested.

The past history of defects (bugs) discovered during Unit testing will help identify potential areas within the software that are risky. If the unit testing discovered a large number of defects or a tendency towards defects in a particular area of the software, this is an indication of potential future problems. It is the nature of defects to cluster and clump together. If it was defect ridden earlier, it will most likely continue to be defect prone.

One good approach to define where the risks are is to have several brainstorming sessions.

- Start with ideas, such as, what worries me about this project/application.

## **6 FEATURES TO BE TESTED**

This is a listing of what is to be tested from the **USERS** viewpoint of what the system does. This is not a technical description of the software, but a USERS view of the functions.

Set the level of risk for each feature. Use a simple rating scale such as (H, M, L): High, Medium and Low. These types of levels are understandable to a User. You should be prepared to discuss why a particular level was chosen.

It should be noted that Section 4 and Section 6 are very similar. The only true difference is the point of view. Section 4 is a technical type description including version numbers and other technical information and Section 6 is from the User's viewpoint. Users do not understand

technical software terminology; they understand functions and processes as they relate to their jobs.

### 7 FEATURES NOT TO BE TESTED

This is a listing of what is NOT to be tested from both the Users viewpoint of what the system does and a configuration management/version control view. This is not a technical description of the software, but a USERS view of the functions.

Identify WHY the feature is not to be tested, there can be any number of reasons.

- Not to be included in this release of the Software.
- Low risk, has been used before and is considered stable.
- Will be released but not tested or documented as a functional part of the release of this version of the software.

Sections 6 and 7 are directly related to Sections 5 and 17. What will and will not be tested are directly affected by the **levels of acceptable risk** within the project, and what does not get tested affects the **level of risk** of the project.

### 8 APPROACH (STRATEGY)

This is your overall test strategy for this test plan; it should be appropriate to the level of the plan (master, acceptance, etc.) and should be in agreement with all higher and lower levels of plans. Overall rules and processes should be identified.

- Are any special tools to be used and what are they?
- Will the tool require special training?
- What metrics will be collected?
- Which level is each metric to be collected at?
- How is Configuration Management to be handled?
- How many different configurations will be tested?
- Hardware
- Software
- Combinations of HW, SW and other vendor packages
- What levels of regression testing will be done and how much at each test level?
- Will regression testing be based on severity of defects detected?
- How will elements in the requirements and design that do not make sense or are untestable be processed?

If this is a master test plan the overall project testing approach and coverage requirements must also be identified.

Specify if there are special requirements for the testing.

- Only the full component will be tested.
- A specified segment of grouping of features/components must be tested together.

Other information that may be useful in setting the approach are:

- MTBF, Mean Time Between Failures - if this is a valid measurement for the test involved and if the data is available.
- SRE, Software Reliability Engineering - if this methodology is in use and if the information is available.

How will meetings and other organizational processes be handled?

### 9 ITEM PASS/FAIL CRITERIA

What are the Completion criteria for this plan? This is a critical aspect of any test plan and should be appropriate to the level of the plan.

- At the Unit test level this could be items such as:
  - All test cases completed.
  - A specified percentage of cases completed with a percentage containing some number of minor defects.
  - Code coverage tool indicates all code covered.
- At the Master test plan level this could be items such as:
  - All lower level plans completed.
  - A specified number of plans completed without errors and a percentage with minor defects.

This could be an individual test case level criterion or a unit level plan or it can be general functional requirements for higher level plans.

What is the number and severity of defects located?

- Is it possible to compare this to the total number of defects? This may be impossible, as some defects are never detected.
  - A defect is something that **may** cause a failure, and may be acceptable to leave in the application.
  - A failure is the result of a defect as seen by the User, the system crashes, etc.

### 10 SUSPENSION CRITERIA AND RESUMPTION REQUIREMENTS

Know when to pause in a series of tests.

- If the number or type of defects reaches a point where the follow on testing has no value, it makes no sense to continue the test; you are just wasting resources.

Specify what constitutes stoppage for a test or series of tests and what is the acceptable level of defects that will allow the testing to proceed past the defects.

Testing after a truly fatal error will generate conditions that may be identified as defects but are in fact ghost errors caused by the earlier defects that were ignored.

### 11 TEST DELIVERABLES

What is to be delivered as part of this plan?

- Test plan document.
- Test cases.
- Test design specifications.
- Tools and their outputs.
- Simulators.
- Static and dynamic generators.
- Error logs and execution logs.
- Problem reports and corrective actions.

One thing that is not a test deliverable is the software itself that is listed under test items and is delivered by development.

### **12 REMAINING TEST TASKS**

If this is a multi-phase process or if the application is to be released in increments there may be parts of the application that this plan does not address. These areas need to be identified to avoid any confusion should defects be reported back on those future functions. This will also allow the users and testers to avoid incomplete functions and prevent waste of resources chasing non-defects.

If the project is being developed as a multi-party process, this plan may only cover a portion of the total functions/features. This status needs to be identified so that those other areas have plans developed for them and to avoid wasting resources tracking defects that do not relate to this plan.

When a third party is developing the software, this section may contain descriptions of those test tasks belonging to both the internal groups and the external groups.

### **13 ENVIRONMENTAL NEEDS**

Are there any special requirements for this test plan, such as:

- Special hardware such as simulators, static generators etc.
- How will test data be provided. Are there special collection requirements or specific ranges of data that must be provided?
- How much testing will be done on each component of a multi-part feature?
- Special power requirements.
- Specific versions of other supporting software.
- Restricted use of the system during testing.

### **14 STAFFING AND TRAINING NEEDS**

Training on the application/system.

Training for any test tools to be used.

Section 4 and Section 15 also affect this section. What is to be tested and who is responsible for the testing and training.

### **15 RESPONSIBILITIES**

Who is in charge?

This issue includes all areas of the plan. Here are some examples:

- Setting risks.
- Selecting features to be tested and not tested.
- Setting overall strategy for this level of plan.
- Ensuring all required elements are in place for testing.
- Providing for resolution of scheduling conflicts, especially, if testing is done on the production system.
- Who provides the required training?
- Who makes the critical go/no go decisions for items not covered in the test plans?

### **16 SCHEDULE**

Should be based on realistic and validated estimates. If the estimates for the development of the application are inaccurate, the entire project plan will slip and the testing is part of the overall project plan.

- As we all know, the first area of a project plan to get cut when it comes to crunch time at the end of a project is the testing. It usually comes down to the decision, 'Let's put something out even if it does not really work all that well'. And, as we all know, this is usually the worst possible decision.

How slippage in the schedule will to be handled should also be addressed here.

- If the users know in advance that a slippage in the development will cause a slippage in the test and the overall delivery of the system, they just may be a little more tolerant, if they know it's in their interest to get a better tested application.
- By spelling out the effects here you have a chance to discuss them in advance of their actual occurrence. You may even get the users to agree to a few defects in advance, if the schedule slips.

At this point, all relevant milestones should be identified with their relationship to the development process identified. This will also help in identifying and tracking potential slippage in the schedule caused by the test process.

It is always best to tie all test dates directly to their related development activity dates. This prevents the test team from being perceived as the cause of a delay. For example, if system testing is to begin after delivery of the final build, then system testing begins the day after delivery. If the delivery is late, system testing starts from the day of delivery, not on a specific date. This is called dependent or relative dating.

### **17 PLANNING RISKS AND CONTINGENCIES**

What are the overall risks to the project with an emphasis on the testing process?

- Lack of personnel resources when testing is to begin.
- Lack of availability of required hardware, software, data or tools.
- Late delivery of the software, hardware or tools.
- Delays in training on the application and/or tools.
- Changes to the original requirements or designs.

Specify what will be done for various events, for example:

- Requirements definition will be complete by January 1, 19XX, and, if the requirements change after that date, the following actions will be taken.
  - The test schedule and development schedule will move out an appropriate number of days. This rarely occurs, as most projects tend to have fixed delivery dates.
  - The number of test performed will be reduced.
  - The number of acceptable defects will be increased.
    - These two items could lower the overall quality of the delivered product.
  - Resources will be added to the test team.
  - The test team will work overtime.
    - This could affect team morale.
  - The scope of the plan may be changed.
  - There may be some optimization of resources. This should be avoided, if possible, for obvious reasons.

- You could just QUIT. A rather extreme option to say the least.

Management is usually reluctant to accept scenarios such as the one above even though they have seen it happen in the past.

The important thing to remember is that, if you do nothing at all, the usual result is that testing is cut back or omitted completely, neither of which should be an acceptable option.

## **18 APPROVALS**

Who can approve the process as complete and allow the project to proceed to the next level (depending on the level of the plan)?

At the master test plan level, this may be all involved parties.

When determining the approval process, keep in mind who the audience is.

- The audience for a unit test level plan is different than that of an integration, system or master level plan.
- The levels and type of knowledge at the various levels will be different as well.
- Programmers are very technical but may not have a clear understanding of the overall business process driving the project.
- Users may have varying levels of business acumen and very little technical skills.
- Always be wary of users who claim high levels of technical skills and programmers that claim to fully understand the business process. These types of individuals can cause more harm than good if they do not have the skills they believe they possess.

## **19 GLOSSARY**

Used to define terms and acronyms used in the document, and testing in general, to eliminate confusion and promote consistent communications.