I. Planning Your Test Effort

The goal of a software tester is to find bugs, find them as early as possible, and make sure they get fixed.

2. The Goal of Test Planning (Question: Explain the goal of test planning in software testing = 4 marks)

1. Performing testing tasks would be very difficult if the programmers wrote their code without telling what it does, how it works, or when it will be complete.

2. Software testers don't communicate what tester plans to test, what resources you need, and what your schedule is, your project will have little chance of succeeding.

3. The software test plan is the primary means by which software testers communicate to the product development team what they intend to do.

4. The test plan is simply a by-product of the detailed planning process that's undertaken to create it. It's the planning process that matters, not the resulting document.

5. The ultimate goal of the test planning process is communicating (not recording) the software test team's intent, its expectations, and its understanding of the testing that's to be performed.

3. Test Planning Topics

1. The problem with test planning approach is that it makes it too easy to put the emphasis on the document, not the planning process.

2. Test leads and managers of large software projects have been known to take an electronic copy of a test plan template or an existing test plan, spend a few hours cutting, copying, pasting, searching, and replacing, and turn out a "unique" test plan for their project.

3. They felt they had done a great thing, creating in a few hours what other testers had taken weeks or months to create.

4. They missed the point, though, and their project showed it when no one on the product team knew what the heck the testers were doing or why.

b) High Level Expectations

1. What's the purpose of the test planning process and the software test plan?

2. What product is being tested?

3. What are the quality and reliability goals of the product?

c) People, Places and Things

1. Test planning needs to identify the people working on the project, what they do, and how to contact them.

2. If it's a small project this may seem unnecessary, but even small projects can have team members scattered across long distances or undergo personnel changes that make tracking who does what difficult.

3. A large team might have dozens or hundreds of points of contact. The test team will likely work with all of them and knowing who they are and how to contact them is very important.

4. The test plan should include names, titles, addresses, phone numbers, email addresses, and areas of responsibility for all key people on the project.

d) Definitions

1. The software doesn't do something that the product specification says it should do.

2. The software does something that the product specification says it shouldn't do.

3. The software does something that the product specification doesn't mention.

4. The software doesn't do something that the product specification doesn't mention but should build.

The test plan should define the frequency of builds (daily, weekly) and the expected quality level.

- Test release document (TRD).A document that the programmers release with each build stating what's new, different, fixed, and ready for testing.
- Alpha release. A very early build intended for limited distribution to a few key customers and to marketing for demonstration purposes. It's not intended to be used in a real-world situation. The exact contents and quality level must be understood by everyone who will use the alpha release.
- Beta release. The formal build intended for widespread distribution to potential customers. e)

Inter-Group Responsibilities (Question: Explain inter group responsibilities in software testing = 4 marks)

1. Inter-group responsibilities identify tasks and deliverables that potentially affect the test effort. T 2. The test team's work is driven by many other functional groups—programmers, project managers, technical writers, and so on.

3. If the responsibilities aren't planned out, the project— specifically the testing— can become a comedy show of "I've got it, no, you take it, didn't you handle, no, I thought you did," resulting in important tasks being forgotten.

f) What will and won't be tested?

1. There may be components of the software that were previously released and have already been tested. Content may be taken as is from another software company.

2. An outsourcing company may supply pre-tested portions of the product.

g) Test Schedule

1. The test schedule takes all the information presented so far and maps it into the overall project schedule.

2. This stage is often critical in the test planning effort because a few highly desired features that were thought to be easy to design and code may turn out to be very time consuming to test.

3. An example would be a program that does no printing except in one limited, obscure area.

4. No one may realize the testing impact that printing has, but keeping that feature in the product could result in weeks of printer configuration testing time.

5. Completing a test schedule as part of test planning will provide the product team and project manager with the information needed to better schedule the overall

project.

6. They may even decide, based on the testing schedule, to cut certain features from the product or postpone them to a later release.

h) Test Cases

1. The test planning process will decide what approach will be used to write them, where the test cases will be stored, and how they'll be used and maintained.

i) Bug Reporting

1. The possibilities range from shouting over a cubical wall to sticky notes to complex bug-tracking databases.

2. Exactly what process will be used to manage the bugs needs to be planned so that each and every bug is tracked from when it's found to when it's fixed—and never, ever forgotten.

j) Metrics and Statistics

1. Metrics and statistics are the means by which the progress and the success of the project, and the testing, are tracked.

2. The test planning process should identify exactly what information will be gathered, what decisions will be made with them, and who will be responsible for collecting them.

Examples of test metrics that might be useful are

- Total bugs found daily over the course of the project.
- List of bugs that still need to be fixed.
- Current bugs ranked by how severe they are.

k) Risk and Issues

1. A common and very useful part of test planning is to identify potential problem or risky areas of the project—ones that could have an impact on the test effort.

II. Writing and Tracking Test Cases:

2. The Goal of Test Case Planning (Question: Explain the goal of test case planning in software testing = 4 marks)

1. Organization: Even on small software projects it's possible to have many thousands of test cases. The cases may have been created by several testers over the course of several months or even years. Proper planning will organize them so that all the testers and other project team members can review and use them effectively.

2. Repeatability: It's necessary over the course of a project to run the same tests several times to look for new bugs and to make sure that old ones have been fixed. Without proper planning, it would be impossible to know what test cases were last run and exactly how they were run so that you could repeat the exact tests.

3. Tracking: Similarly, you need to answer important questions over the course of a project. How many test cases did you plan to run? How many did you run on the last software release? How many passed and how many failed? Were any test cases skipped? And so on. If no planning went into the test cases, it would be impossible to answer these questions.

4. Proof of testing (or not testing): In a few high-risk industries, the software test team must prove that it did indeed run the tests that it planned to run. It could actually be illegal, and dangerous, to release software in which a few test cases were skipped. Proper test case planning and tracking provides a means for proving what was tested.

3. Test Case Planning Overview (Question: Explain the test case planning overview in software testing Each test planning 2 marks, diagram 2 marks = 8 marks)

1. The test planning comprises of the various stages such as test design specifications, test case specification and the test procedure specifications as shown in Figure 1.

a) Test Design

1. The overall project test plan is written at a very high level.

2. It breaks out the software into specific features and testable items and assigns them to individual testers, but it doesn't specify exactly how those features will be tested.

3. There may be a general mention of using automation or black-box or white-box

testing, but the test plan doesn't get into the details of exactly where and how they will be used.

4. This next level of detail that defines the testing approach for individual software features is the test design specification.

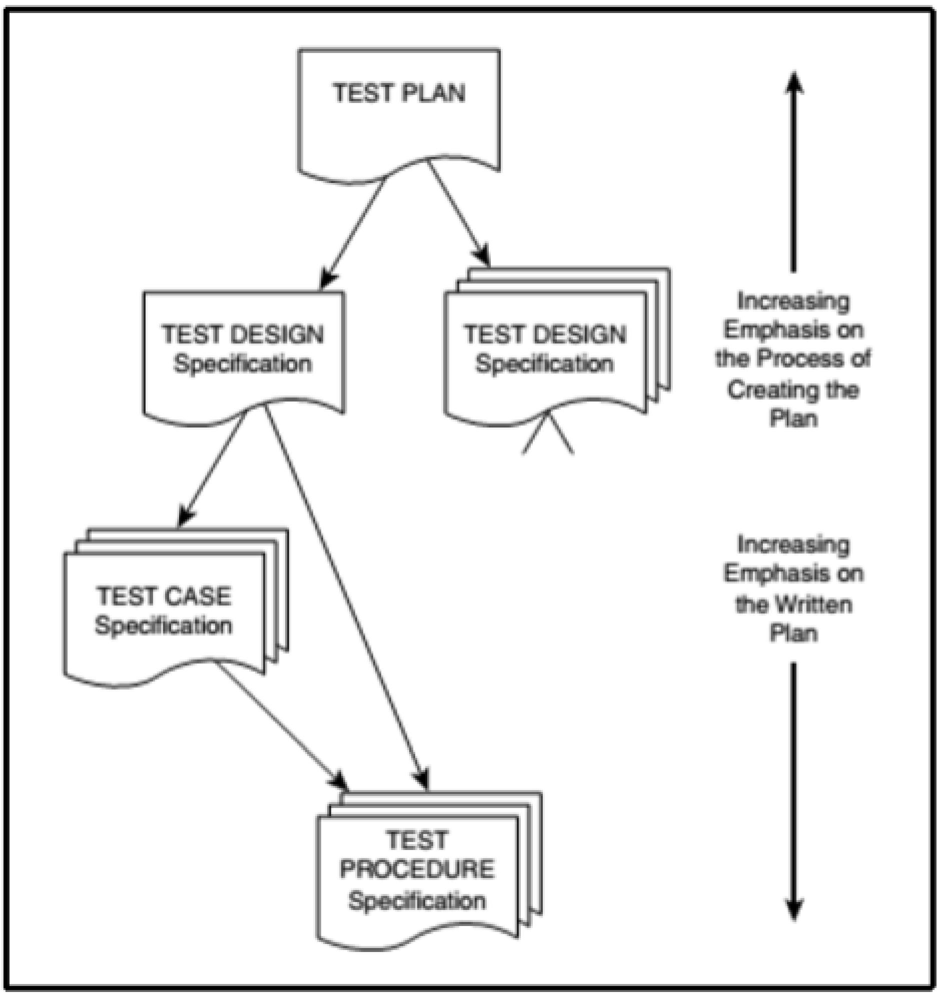
TEST PLAN
TEST DESIGN Specification
TEST DESIGN Specification
TEST CASE Specification
TEST PROCEDURE Specification
Increasing Emphasis on the Process of Creating the Plan
Increasing Emphasis on the Written Plan

Figure 1

b) Test Cases

1. Identifiers: A unique identifier that can be used to reference and locate the test design spec. The spec should also reference the overall test plan and contain pointers to any other plans or specs that it references.

2. Features to be tested: A description of the software feature covered by the test design spec—for example, "the addition function of Calculator," "font size selection and display in WordPad," and "video card configuration testing of QuickTime." For example, "Although not the target of this plan, the UI of the file open dialog box will be indirectly tested in the process of testing the load and save functionality."

3. Approach: A description of the general approach that will be used to test the features. It should expand on the approach, if any, listed in the test plan, describe the technique to be used, and explain how the results will be verified.

4. Test case identification: A high-level description and references to the specific test cases that will be used to check the feature. It should list the selected equivalence partitions and provide references to the test cases and test procedures used to run them.

5. Pass/fail criteria: Describes exactly what constitutes a pass and a fail of the tested feature. This may be very simple and clear—a pass is when all the test cases are run without finding a bug. It can also be fuzzy—a failure is when 10 percent or more of the test cases fail. There should be no doubt, though, what constitutes a pass or a fail of the feature.

c) Test Procedures

1. The test procedure or test script specification defines the step-by-step details of exactly how to perform the test cases.

Here's the information that needs to be defined:

- Identifier: A unique identifier that ties the test procedure to the associated test cases and test design.
- Purpose: The purpose of the procedure and reference to the test cases that it will execute.
- Special requirements: Other procedures, special testing skills, or special equipment needed to run the procedure.
- Procedure steps: Detailed description of how the tests are to be run:

4. Test Case Organization & Tracking

1. One consideration that you should take into account when creating the test case documentation is how the information will be organized and tracked.

2. The questions that a tester or the test team should be able to answer:

- Which test cases do you plan to run?
- How many test cases do you plan to run? How long will it take to run them?
- Can you pick and choose test suites (groups of related test cases) to run on

particular features or areas of the software?

- When you run the cases, will you be able to record which ones pass and which ones fail?
- Of the ones that failed, which ones also failed the last time you ran them?
- What percentage of the cases passed the last time you ran them?