UNIT-V

Test Planning, Management

1. TEST PLANNING

1.1 Preparing a Test Plan:

Testing is like any project should be written by a plan.

- What needs to be tested-the scope of testing, including clear identification of what will be tested and what will not be tested.
- How the testing is going to be performed-breaking down the testing into small and manageable tasks and identifying the strategies to be used for carrying out the tasks.
- What resources are needed for testing?

1.2 Scope management: Deciding features to be tested/Not tested.

One single test plan can be prepared to cover all phases and all teams or there can be separate plans for each phase or for each type of testing.

Scope management pertains to specifying the scope of a project. For testing, scope management entails.

1. Understanding what constitutes a release of a product.
2. Breaking down the release into features.
3. Prioritizing the features for testing.
4. Deciding which features will be tested and which will not be
5. Gathering details to prepare for estimation of resources for testing.

The following factors drive the choice and prioritization of features to be tested.

- **Features that are new and critical for the release:**
  - The new features of a release set the expectations of the customers and must perform properly.
  - These new features result in new program code and thus have a higher susceptibility and exposure to defects.
  - Hence it makes sense to put these features on top of the priority list to be tested.
- **Features whose failures can be catastrophic (Terrible):**
  - Regardless of whether a feature is new or not, any feature the failure of which can be catastrophic or produce adverse business impact has to be high on the list of features to be tested.
- **Features that are expected to be complex to test**
  - Early participation by the testing team can help identify that are difficult to test. This can help in starting the work.
- **Features which are extension of earlier features that have been defect prone(given, procurement)**
  - Certain areas of code tend to be defect prone and such areas need very thorough testing so that old defects do not creep in again.

1. 3 Deciding Test Approach/ Strategy

1. What type of testing would you use for testing the functionality
2. What are the configurations or scenarios for testing the features? What integration testing would you do to ensure these features work together?
3. What localization validations would be needed?
4. What "non-functional" tests would you need to do?

The test strategy or approach should result in identifying the right type of test for each of the features or combinations. There should also be objective

criteria for measuring the success of test.

1. 4 Setting Up Criteria for Testing

1. The entry criteria for test specify threshold criteria for each phase or type of test.
2. There may also be entry criteria for the entire testing activity to start.
3. A test cycle or a test activity will not be an isolated, continuous activity that can be carried out at one go.

1. 5 Identifying Responsibilities, Staffing, and Training Needs

1. Scope Management identifies what needs to be tested. The test strategy outlines how to do it. The next aspect of planning is the who part of it.
2. A testing project requires different people to play different roles.
3. The different role definition should be Ensure there is clear accountability for a given task, so that everyone knows how his or her work fits into the entire project.
4. Complement each other ensuring no one steps on an other's toes.

Staffing is done based on estimation of effort involved and the availability of time for release.
People are assigned to tasks that achieve the best possible fit between the requirements of the job and skills and experience levels needed to perform that function.

1. 6 Identifying Test Deliverables

1. The test plan itself (master test plan, and various other test plans

for the project).
2. Test case design specifications.
3. Test cases, including any automation that is specified in the plan.
4. Test logs produced by running the tests.
5. Test summary reports.

1. 7 Identifying Test Deliverables

1. The test plan itself (master test plan, and various other test plans for the project).
2. Test case design specifications.
3. Test cases, including any automation that is specified in the plan.
4. Test logs produced by running the tests.
5. Test summary reports.

1.8 Testing tasks : Size and Effort Estimation

Estimation happens broadly in three phases.

1. Size Estimation.
2. Effort Estimation.
3. Schedule Estimation.

Size estimate quantifies the actual amount of testing that needs to be done.

I. Size of the product under test:

This obviously determines the amount of testing that needs to be done. The larger product in general greater is the size of testing to be done. Effort estimation is given in person days, person months, or person years. The effort estimate is then translated is the translated to a schedule estimate.

Some of the measures of size of product under test are as follows.

1. Lines Of Code(LOC) :

- LOC is somewhat controversial measure as it depends on the language, style of programming, compactness.

2. A Function Point:

- A function point is a popular method to estimate the size of the application. Function point provide a representation of application size, independent of programming language.

3. Application Size:

- Simpler representation of application size is the number of screens, reports or transactions.

II. Extent of automation required

When the automation is involved the size of the work to be done for testing increases.

III. Number of platforms and inter-operability environments to be tested.

- If a particular product is to be tested under several different platforms or under several different configurations, then the size of the testing task increases.

WBS(Work Breakdown Structure)

: The work to be done is broken down into smaller and more manageable parts called Work Breakdown Structure units.

Size estimate is expressed in terms of any of the following.

1. Number of test cases.
2. Number test scenarios.
3. Number of configurations to be tested.

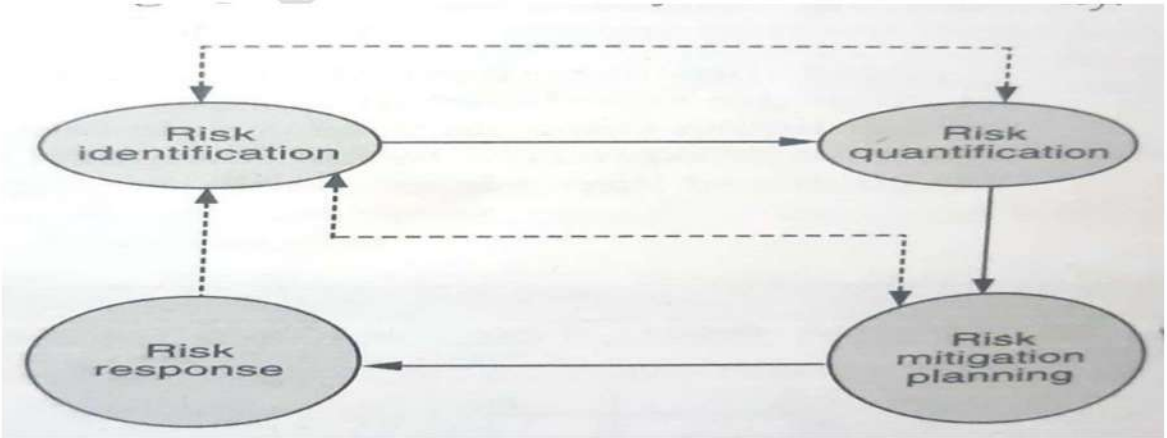
1. 9 Activity Breakdown and Scheduling

1. Identifying external and internal dependencies among the activities. Sequencing the activities, based on the expected duration as well as on the dependencies.
2. Identifying the time required for each of the WBS activities, taking into account the above two factors.
3. Monitoring the progress in terms of time and effort.
4. Rebalancing schedules and resources as necessary.

1.11 Risk Management

Risks are events that could potentially affect a projects outcome. These events are normally beyond the control of the project manager.

1. Identifying the possible risks.
2. Quantifying the risks.
3. Planning how to mitigate the risks and
4. Responding to risks when they become a reality.



Risk Identification:

Risk identification consists of identifying the possible risks that may hit a project. Although there could potentially be many risks that can hit a project, the risk identification step should focus on those risks that are more likely to happen.

Use of Checklists:

Over time, an organization may find new gleanings on testing that can be captured in the form of checklist.

Use of organizational history and metrics

When an organization collects and analyzes the various metrics, the information can provide valuable insights into what possible risks can hit a project.

Over time, an organization may find new gleanings on testing that can be captured in the form of checklist.

Information networking across the industry

The informal networking across the industry can help in identifying risks that other organization have encountered.

Risk Quantification

Risk quantification deals with expressing the risk in numerical terms. There are two components to the quantification of risk. One is the probability of the risk happening and the other is the impact of the risk, If the risk happens.

Risk Mitigation

Risk mitigation planning deals with identifying alternative strategies to combat a risk event, should that risk materialize. For example, a couple of mitigation strategies for the risk of attrition are to spread the knowledge to multiple people and to introduce organization wide processes and standards.

The following are some of the common risks encountered in testing projects and their characteristics.

Unclear requirements:

When the requirements to be satisfied by a product are not clearly documented, there is ambiguity in how to interpret the result of a test. This could result in wrong defects being reported or in the real defects being missed out.

Schedule dependencies:

The schedule of the testing team depends significantly on the schedules of the development team. Thus, it becomes difficult for the testing team to line up resources properly at the high time.

Insufficient time for testing:

System testing and performance testing can happen only after the entire product is ready and close to the release date. Usually tests are resource intensive for the testing team and in addition the defects that these tests uncover are challenging for the developers to fix.

Show stopper defects:

When the testing team reports defects, the development team has to fix them. Certain defects which are show stoppers may prevent the testing team has to fix them.

Availability of skilled and motivated people for testing

This is especially important for testing functions because of the tendency of people to look for development positions

3. TEST MANAGEMENT:

3.1 Choice of Standards

Standards comprise an important part of planning in any organization standards are of two types-external standards and internal standards.

External standards are standards that a product should comply with an externally visible, and are usually stipulated by external consortia.

Internal standards are standards formulated by a testing organization to bring in consistency and predictability. They standardize the process and method of working within the organization.

Some of the internal standards are:

1. Naming and Storage convention for test artifacts.
2. Document Standards.
3. Test coding Standards.
4. Test Reporting Standards.

Naming and Storage convention for test artifacts.

1. East identification of the product functionality that a set of tests are intended.

2. Remove mapping to identify the functionality corresponding toa given set of tests.

Documentation Standards.

1. In the case of manual testing documentation standards correspond to specifying the user and the system at the right level of detail that is consistent with the skill level of the tester.
2. While naming and directory standards specify how a test case represented externally, documentation standards specify how to generate information about the tests within the test scripts themselves.
3. Appropriate header level comments at the beginning of the file.
4. Sufficient in-line comments spread throughout the file explains the functions served by the various parts.
5. Up-to-date Change history information regarding all the changes made to the test file.

Test Coding Standards.

1. Enforce the right type of initialization and clean-up that and the result should be independent of other tests.
2. Encourage reusability of test artifacts.
3. Stipulate ways of naming variables within the scripts to make that the reader understands consistently the purpose of a variable.
4. Provide standard interfaces to external entities like operating system, hardware, and soon.

Test Reporting Standards.

1. All the stakeholders must get a consistent and timely view of the progress of tests. Test reporting standards address this issue.
2. It provides guidelines on the level of detail that should be present in the test reports.

3. Their standard formats and contents, recipients of the report and so on.

3.2 Test Infrastructure Management

1. A test case database (TCDB)
2. A defect repository.
3. Configuration Management repository and tool.

- A test case database captures all the relevant information about the test cases in an organization.
- A defect repository captures all the relevant details of defects reported fora product.
- Software configuration management (SCM) repository.
- An SCM repository also known as CM repository keeps track of change control and version control of all the files/Entities.

Information in a defect repository.

Entity	Purpose	Attributes
Defect details	Records all the "static" information about the tests	• Defect ID • Defect priority/severity • Defect description • Affected product(s) • Any relevant version information • Environmental information (for example, OS version) • Customers who encountered the problem (could be reported by the internal testing team also) • Date and time of defect occurrence
Defect test details	Provides details of test cases for a given defect. Cross-references the TCDB	• Defect ID • Test case ID
Fix details	Provides details of fixes for a given defect; cross-references the configuration management repository	• Defect ID • Fix details (file changed, fix release information)
Communication	Captures all the details of the communication that transpired for this defect among the various stakeholders. These could include communication between the testing team and development team, customer communication, and so on. Provides insights into effectiveness of communication	• Test case ID • Defect reference # • Details of communication

3.3 Integrating with Product Release

- The success of the product depends on the effectiveness of integration of the development and testing activities.
- When integration testing could start, when system testing could start and so on. These are governed by objective entry criteria for each phase of testing (to be satisfied by development).

- Service level agreements between development and testing as to how long it would take for the testing team to complete the testing.
- The purpose of the testing team is to identify the defects in the product and the risks that could be faced by releasing the product with the existing defects.

Software Test Automation:

Developing software to test the software is called automation. Test automations can help address several problems.

- Automation saves time as software can execute test cases faster than human do.
- The automation can free the test engineers from mundane tasks and make them focus on more creative tasks.
- Automated tests can be more reliable: When an engineer executes a particular test case many times manually, there is a chance for human error or a bias because of which some of the defects may get missed out.
- Automation helps in immediate testing: Automation reduces the time gap between development and testing as scripts can be executed as soon as the product build is ready.
- Automation can protect an organization against attrition of test engineers: Automation can also be used as a knowledge transfer tool to train test engineer on the product as it has a repository of different tests for the product.
- Certain type of testing cannot be executed without automation: Test case for certain types testing such as reliability testing, stress testing, load and performance testing, and cannot be executed without automation.

Terms Used in Automation:

- Test case is a set of sequential steps to execute a test operating on a set of predefined inputs to produce certain expected outputs.
- There are two types of test cases. Automated and manual test case.
- A manual test case is executed manually while an automated test case is

executed using automation.

- A test case can be represented in much form it can be documented as a set of simple steps or it could be an assertion statement or a set of assertions.
- Some test cases are repeated several times during a product release because the product is built several times.
- If we closely go through the above table one can observe that there are two important dimensions. "What operations have to be tested", "how the operation has to be tested", "what an operation has to do".

Skills Need for Automation:

First generation-Record and playback:
Record and playback avoid the repetitive nature of executing tests. Almost all the test tools available in the market have the record and playback feature. The scripts may contain hard-coded values.

Second generation-Data driven:
The method helps in developing test scripts that generate the set of input conditions and corresponding expected outputs. This generation of automation focuses on input and output conditions using the black box testing approach.

Third generation-Action driven:
All actions that appear on the application are automatically tested, based on a generic set of controls defined for automation.

The user needs to specify only the operations (such as log in, download, and so on).

Scope of Automation:

Certain types of tests automatically lend themselves of automation.
Stress, reliability, scalability, and performance testing: These types of testing require the test cases to be run from large number of different machines for an

extended period of time. Such as 24 hours, 48 hours

Regression tests: Regression tests are repetitive in nature. These test cases are executed multiple times during the product development phases. Automation will save significant time and effort in the long run.

Functional tests: This kind of test may require a complex set up and thus require specialized skill, which may not be available on an ongoing basis. Automating these once, using the expert skill sets, can enable using less-skilled people to run these tests on an ongoing basis.

Automating Area Less Prone to Change:

User interfaces normally go through significant changes during a project. To avoid an network on automated test cases, proper analysis has to be done to find out the areas of changes to user interfaces, and automate only those areas that will go through relatively less change.

Automating Tests that Pertain to Standards:

Automating for standards provides a dual advantage. Test suites developed for standards are not only used for product testing but can also be sold as test tools for the market. A large number of tools available in the commercial market were internally developed for **in-house usage**. Hence automating for standards create new opportunities for them to be sold as commercial tools.