Software Testing

Unit - 5

Test Planning

→ important to provide efficient resource control monitoring

Test plan

→ prepared by test manager
→ document containing all future testing-related activities
→ defines work products to be tested, how will they be tested and test type distribution among the testers

Importance

→ quick guide for the testing process
→ avoid out-of-scope functionalities
→ determines time, cost, effort
→ provides a schedule
→ resource requirement
→ understand test details
→ determine quality of software applications

Test Plan Guidelines

→ avoid overlapping & repetition
→ avoid lengthy paragraphs
→ use lists and tables
→ update plan
→ don't use outdated documents

Types of test plan

- **Master Test Plan** — multiple strategies & multiple levels of testing
- **Phase Test Plan** — emphasize on one phase of testing
- **Specific Test Plan** — only specific types of testing especially non-functional

Test Plan Attributes

Objective → Test Strategy (→ calculate test effort & cost; ← in scope - modules tested rigorously, out scope - modules not " ") → Test methodology → Approach (→ flow of application) → Assumptions → Risk → Backup/Mitigation Plan → Roles and Responsibilities → Scheduling → Defect Tracking → Test environment → Entry and exit criteria → Test automation → Deliverables (→ outcome) → Templates

If no plan

→ misunderstanding roles & responsibilities
→ no clear objective
→ no surety when the process ends
→ undefined test scope may be confusing

Test Management

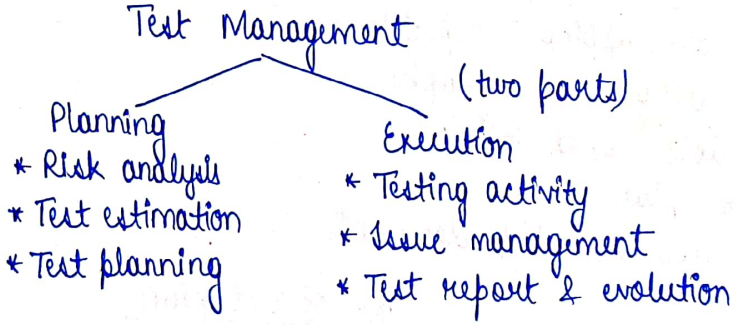
process where testing activities are managed to ensure high quality and high end testing of software applications.

→ consists of tracking, organization, controlling process, that is the visibility of the testing process in order to deliver a high quality software application.

"planning, controlling, tracking & monitoring facilities"

Responsibilities

1) Works in collaboration with test analyst & technical test analyst to customize templates and establish standards
2) provides facilities to track and control testing throughout.
3) gives clear understanding of the testing activity of the upcoming project & also past ones.



Test Execution

→ process of executing the code and comparing the expected and actual results.

Factors that need to be considered

1) In this phase, the QA team performs actual validation of AUT based on prepared test cases & compares the stepwise result with the expected one.

2) The entry criteria of this phase marks the completion of Test Plan and Test Case Development Phase.

3) The validation of Test environment setup is recommended through smoke testing before test execution.

4) Exit criteria requires successful completion & validation of all Test cases, defects should be closed.

Activities during Test Execution

1) System Integration Testing

→ real validation of product / AUT starts here.

→ It is a black box technique that evaluates the system's compliance against specified requirements

→ performed on subset of system with min usage of testing tools

→ interactions and behavior of each data field is tested.

After integration, 3 main states of data flow

a) Data state within the integration layer
b) " " " " database "
c) " " " " application "

2) Defect Reporting

→ Bug arises when the expected result doesn't match the actual one.

→ arise from errors & mistakes by developers

→ The errors need to be reported to the concerned team member

→ Members take action & fix them.

→ Advantage: to ease out the tracking of status of defects

Defect Reporting is a process of finding defects in application during testing or recording feedback & making new versions that fix these defects.

3) Defect Mapping

→ Once defect is reported & logged, it should be mapped with the concerned failed/blocked test cases.

→ done by Defect Reporter

→ helps in making a proper defect report and analyzing the impathness in product.

4) Re-testing

→ means executing a previously failed test against AUT to check whether the problem is resolved.

→ testers look for granular details at the changed area of functionality.

5) Regression Testing

→ Once all defects are closed, deferred or rejected, it is said that system integration testing is completed and then one quick testing is required to ensure none of the functionality is broken due to code changes.

→ Regression Testing is a black box testing that consists of re-executing those tests that had an impact due to code changes

Final Regression Tests
↓
validates the build that has not undergone a change over a period of time

Regression Tests
↓
normal regression tests to verify if the build has NOT broken any other parts of appl^n by recent code changes

Test Automation

→ makes use of specialized tools to control the execution of tests and compare the results against the expected result.

→ Testing tools help to automate data set up generation, product installation, GUI interaction etc.

→ used in both func^l & non-func^l testing → requires money & resources

Scope

I) Functional Testing with the help of Automation

→ mainly used in regression test case execution

→ advisable to keep 60 to 70% of regression test cases to be automated
→ not possible to execute all test cases manually

II Performance Testing using Automation
→ not possible manually
→ process of evaluating app^n performance.

III Load & Stress Testing
→ to test these large no. of concurrent users is needed and multiple computers are required
→ manually not possible
→ so, tools generate no. of virtual users on a single computer

Test Automation Framework

→ It is scaffolding that is laid to provide an execution environment for the automation test scripts
→ Frameworks provide various benefits to develop, execute and report the automation test scripts efficiently.

Advantage

1) Reusability of code
2) Maximum coverage
3) Recovery scenario
4) Low-cost maintenance
5) Minimal manual intervention
6) Easy reporting

Disadvantage

1) requires lot of effort in selecting the tool
2) Tools might generate unexpected errors
3) Not every method or process might be automated.

Types of Automated Testing Frameworks

1) Linear Automation Framework (Record-and-playback framework)
→ testers don't need to write code to create functions
→ steps written in sequential order
→ tester records each step such as navigation, user input & then plays the script back automatically to conduct the test

Adv

1) no need to write custom code
2) fastest ways to generate test scripts
3) test workflow easier to understand

Disadv

1) scripts developed aren't reusable
2) Maintenance is a hassle.

2) Modular Based Testing Framework

→ testers divide the application under test (AUT) into separate units, functions or sections each tested in isolation

→ After this, a test script is created to combine to build larger tests in a hierarchical fashion.

Adv

1) changes in appl^n need only to be addressed to the associated modules,
2) Creating test cases is easier & reusable

Disadv

1) Data is hard-coded into test script
2) Programming knowledge is required

3) Library Architecture Testing Framework

→ Instead of dividing the application under test into various scripts, similar tasks within the scripts are identified and later grouped by func^c so the app^n is broken down by common objectives.

Adv

1) high level of modularization, so easy maintenance & scalability
2) higher degree of reusability

Disadv

1) Test data is hard coded
2) Technical knowledge is needed

4) Data Driven Framework

→ allows users to store and pass the I/O parameters to test scripts from an external data source like Excel sheets, Text files, csv files.

→ These test scripts then read & populate necessary data when needed.

Adv

1) Hard coding data is avoided
2) Tests can be executed with multiple data sets.
3) Time saving by running more test cases faster.

Disadv

1) need an experienced tester who has knowledge of programming lang
2) Setting up environment takes time.

5) Keyword-Driven Framework

→ Each fun^c of the AUT is laid out in a table with a series of instructions in consecutive order that needs to be run.

→ Keywords are also stored in an external data table making them independent from the automated testing tool being used to execute the tests.

→ After the table, testers have to write the code that will prompt the necessary action based on the keywords.

Adv

1) Minimal scripting knowledge is required
2) Single keyword can be used across so the code is reusable
3) Test scripts can be built independent of the AUT

Disadv

1) Initial cost of setting up the framework is high
2) Keywords can be a hassle to maintain.
3) need good automation skills

6) Hybrid Test Automation Framework

→ combination of any previous frameworks to leverage the advantages

→ give the best results possible.

Manual Testing	Automated Testing
→ less reliable, not accurate all times	→ performed by third party tools, more reliable
→ heavy investment for resources	→ lesser investment for tools
→ time consuming	→ faster
→ Initial investment is lower which in turn lowers the returns	→ higher initial investment, hence higher rate of return

Automation Testing Process

Test tool selection → Define Scope of Automation → Planning, Design, Development → Test Execution → Maintenance

Generic Requirements for test tool / framework

1) No hard coding in the test suite
2) Adding new test cases should not affect the others
3) Should be reusable code
4) Automatic setup and clean
5) Insulating test case during execution
6) Coding standard and directory structure
7) Parallel execution of test cases
8) Test case execution based on previous result

Test Tool Selection

1) Meeting Requirements
   - → should improve productivity
   - → should include detailed information reports related to testing process
   - → should be robust in release management tracking ability (Devops)
   - → must provide proper test report

2) Agile Support
   - → creating stories
   - → estimation
   - → sprint backlog planning
   - → Scrum & Kanban

3) External Integration
   - → must support continuous integration
   - → bug tracking tools integration
   - → support desk integration

4) Mobile

→ Test management device should support mobile device & tablets
→ must have full feature set available on mobile

5) support

→ customer focused
→ user friendly
→ advanced search facility
→ accomodate multiple languages & time zones

Testing in Object Oriented Systems

→ In object-oriented systems, there are certain additional dependencies that are not covered in conventional testing

1) Class to class dependencies
2) Class to method dependencies
3) Class to message dependencies
4) Class to variable "
5) Method to variable "
6) Method to message "
7) Method to method "

→ There is lack of sequential control flow within a class so it requires different approach for testing.

Techniques

1) Fault based Testing
2) Class Testing based on Method Testing
3) Random Testing
4) Partition Testing
5) Scenario based Testing