SOFTWARE ENGINEERING

Page No.:
Date : / /

Unit - 04

ONE SHOT + PYQ SOLUTION

⇒ Topics :

☆ Testing Objectives
- Unit testing
- Integration testing
- Acceptance testing
- Regression testing

☆ Testing for Functionality and Testing for Performance

☆ Top - Down and Bottom up

☆ Testing Strategies :
- Test Drivers and Test stubs
- Structural Testing (White Box Testing)
- Functional Testing (Black Box Testing)
- Test Data Suit Preparation
- Alpha and Beta Testing of Products

☆ Static Testing Strategies :
- Formal Technical Review (Peer Review)
- Walk Through
- Code inspection
- Compilance with Design
- coding standard

Page No.
Date / /

⇒ Testing:

- Testing is the process of executing a program to find errors
- To make our software perform well it should be error-free.
- If testing is done successfully it will remove all the errors from the software

⇒ Software Testing:

- Software Testing is a method to assess the functionality of the software program.
- The process check whether the actual software matches the expected requirement and ensure the software is bug-free
- The purpose of software testing is to identify the errors, faults, or missing requirements in contrast to actual requirement
- It mainly aims at measuring the specification, functionality, and performance of a software program or application.

- Software testing is a process of executing a program with the intention of finding bugs or fault in the code.

⇒ Testing objectives:

☆ The main objectives of software testing are:

1. Verify that the software system meets the specified requirements.
2. Identify defects, errors, and issues in the system

Page No.:
Date: / /

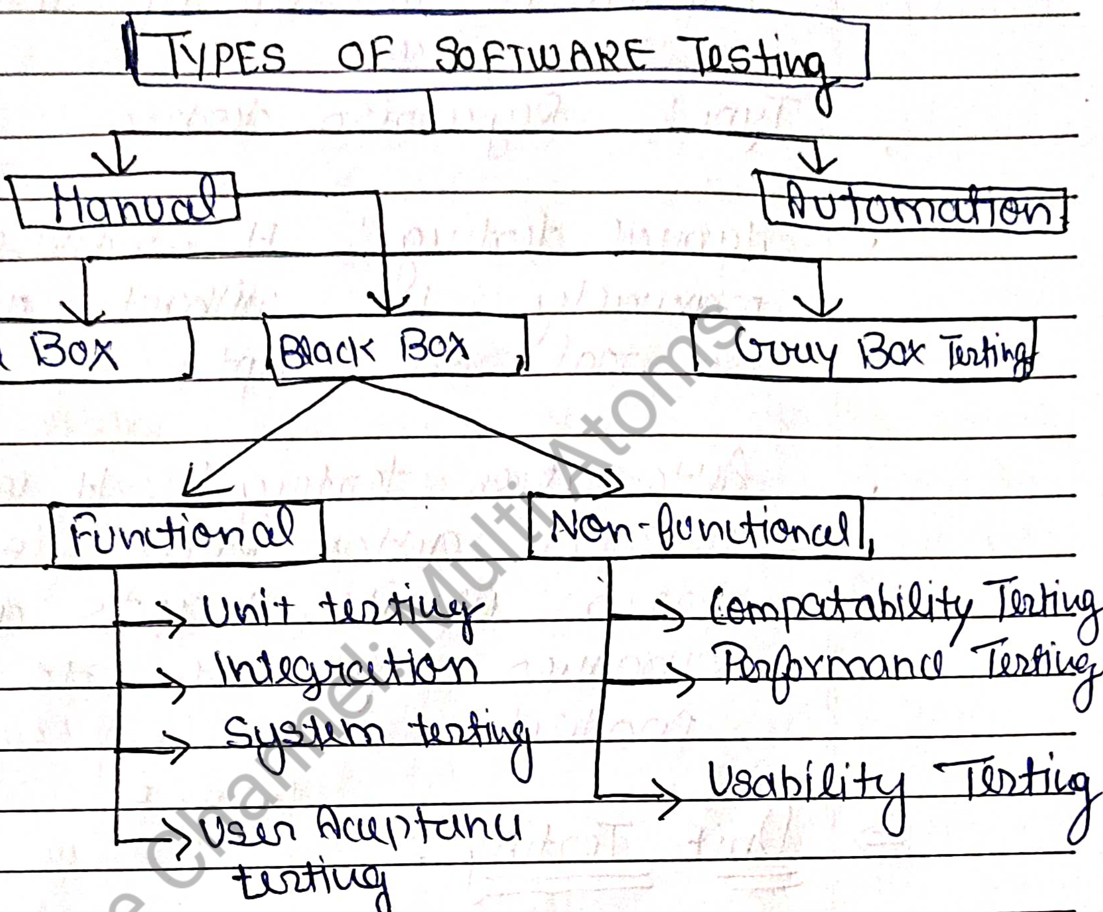
3. Validate the system's functionality and performance
4. Ensure the system is reliable, secure and user friendly.
5. Provide feedback and information for improving the software development process

Principles of Testing:

- All the tests should meet the customer's requirements.
- To make our software testing should be performed by a third party.
- Exhaustive testing is not possible. As we need the optimal amount of testing based on the risk assessment of the application.
- All the tests to be conducted should be planned before implementing it.
- Assign best person of the company during testing.
- We must perform both functional and non-functional testing
- Start testing with small parts and extend it to large parts
- Test time, resource cost is limited.

Importance of Software Testing:
- Defects can be identified early
- Improves quality of software
- Increased customer satisfaction

- Helps with scalability.
- Save time and money.



- Functional testing: It is a type of software testing that validates the software systems against the functional requirement.

→ It is performed to check whether the application is working as per the software's functional requirements or not. Types: ~~Performance, stress testing~~, Unit test, Integration test, smoke test, and so on

- Non-functional testing: It is a type of software testing that checks the application for non-functional requirement like performance, scalability, portability, stress etc.

Types: Stress testing, Usability testing and so on

Page No.:
Date: / /

- Maintenance testing: It is the process of changing, modifying and updating the Software to keep up with the customer needs
Type: Regression testing

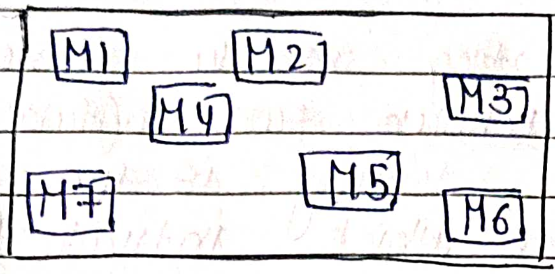
- Manual testing: It includes testing software manually i.e without using any automation tool or script.

- Automation testing: It is also known as Test automation is when the tester write scripts and uses another software to test the product.

⇒ **Unit Testing:**

- Instead of testing the whole program, testing the code at the class level, method level, etc is called Unit Testing.
- The code has to be split into separate classes and methods so that testing can be carried out easily at a unit level.
- It is a White Box technique that is usually performed by the developer.
- It is the lowest level of testing of an application.
- If proper unit testing is done in early development, then it saves time and money in the end.

- Unit testing increases the speed of development



Adv :-

- Unit testing allows the programmer to refine code and make sure the module works properly
- Unit testing enables testing parts of the project without waiting for others to be completed
- Early Detection of issues.
- Improved code quality.
- Increased confidence
- Faster development

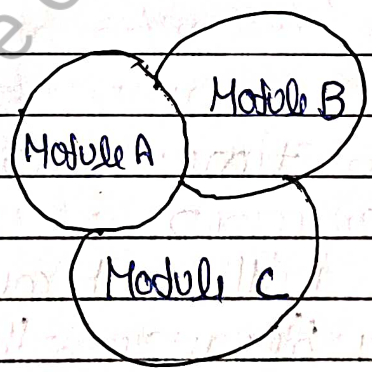
Disadv :-

- The process is time-consuming for writing the unit test cases
- Unit Testing will not cover all the errors in the module because there is a chance of having errors in the module while doing integration testing
- Unit testing is not efficient for checking the errors in the UI part of the module
- Time and effort.

⇒ Integration Testing :-

- Interface testing is the process of testing the interface between two software units or module
- Integration testing is the phase in software testing in which individual software module are combined and tested as a group
- A typical software project consist of multiple software modules, coded by different programme
- It occur after unit testing and before validation testing.
- The purpose of this level of testing is to expose defects in the interaction between software modules when they are integrated

Module A | Module B | Module C



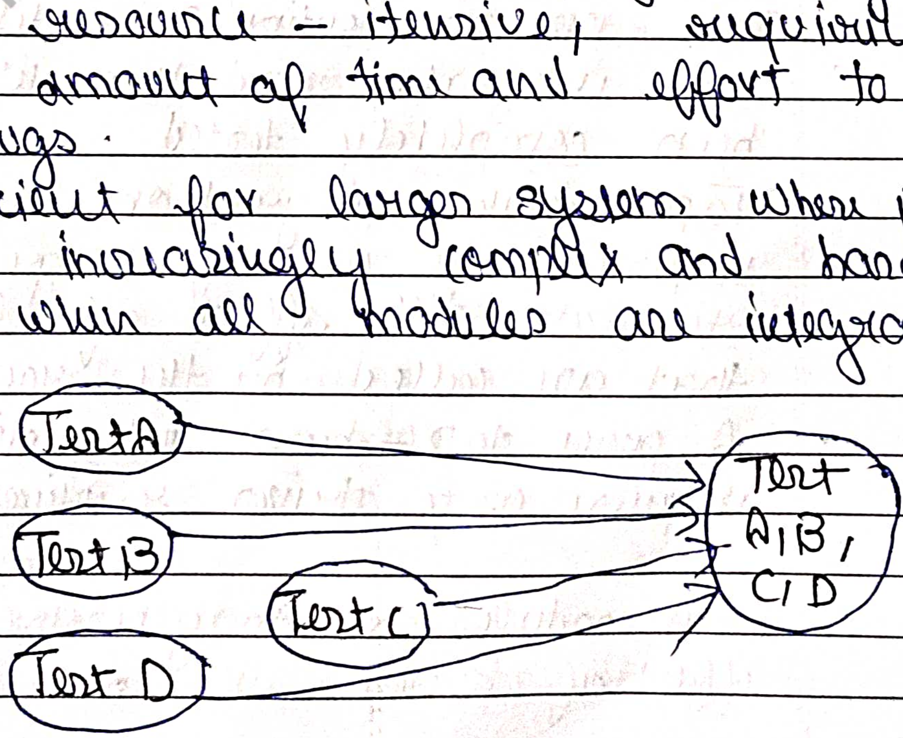
Union Integration testing.

⇒ Types of Integration Testing :

- Big Bang Integration Testing
- Top-Down Integration Testing
- Bottom-Up Integration Testing
- Sandwich / Hybrid Integration Testing

⇒ Big Bang Integration Testing :-

- Big Bang Integration testing is a testing approach where all components or modules are integrated and tested as a single unit.
- It is easier to set up because testing part starts only after all the modules have been integrated.
- It is better suited for smaller system where the modules are heavily interlinked
- It can potentially save time, as testing is conducted after the entire software has been developed and integrated.
- It can be challenging to isolate and fix bugs because of the high level of integration
- There's a high risk involved as any significant issues are only found late in the development process, which can lead to project delays.
- It can be resource - itensive, requiring a significant amount of time and effort to find and fix bugs.
- It's inefficient for larger system where problem can become increasingly complex and hard to identify when all modules are integrated at once



Page No.:
Date: / /

Adv:

- The simplest form of integration Testing
- Easy to implement
- Bugs can be identified at once
- Suitable for small project
- Save resources.
- Simplify the testing process.

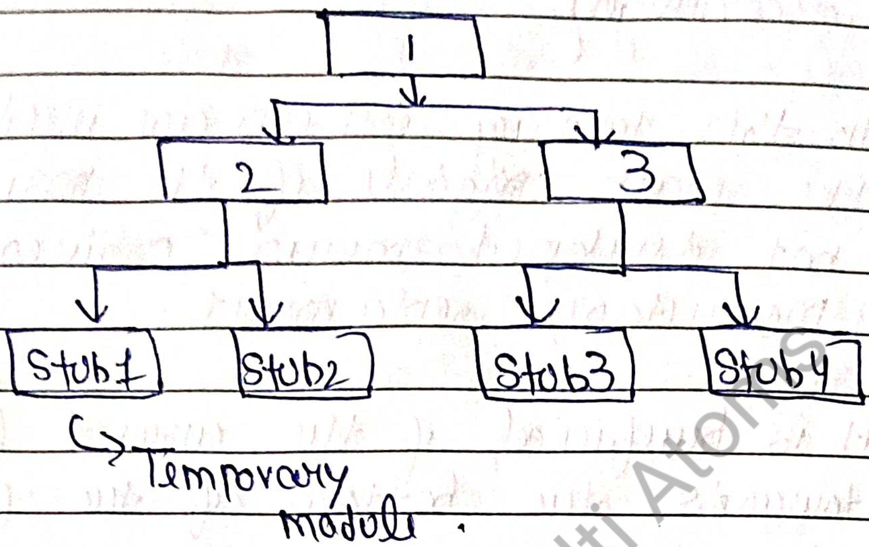
Disadv:-

- It can cause Delays
- Difficult to identify the root cause of error.
- Time consuming
- Not scalable.

Top-Down Integration Testing :-

- In this approach, testing process starts with testing the top most module/component in the hierarchy and move downwards
- This process continues until all component are integrated and then whole system has been completely tested
- Top-down integration testing approach requires the use of program stubs to stimulate the effect of lower-level routines that are called by the routines under test.
- A pure top-down integration does not require any driver routines.

- Stub modules are may be used to simulate the effect of lower-level modules that have not yet been integrated and are called by the routine under test.



Adv:
-> Early Defect Identification.
-> Facilitates progressive Testing
-> Supports Early Demonstration.

Disadv:
-> Stub development
-> Late Detection of lower-level Bugs
-> Difficulty in Test Management.

Bottom-Up Integration Testing:-

- Bottom up Integration testing is one of the approach of integration testing in which integration testing takes place from bottom to top means system integration begins with lowest level module

- In this testing the lower level modules are

Page No.:
Date: / /

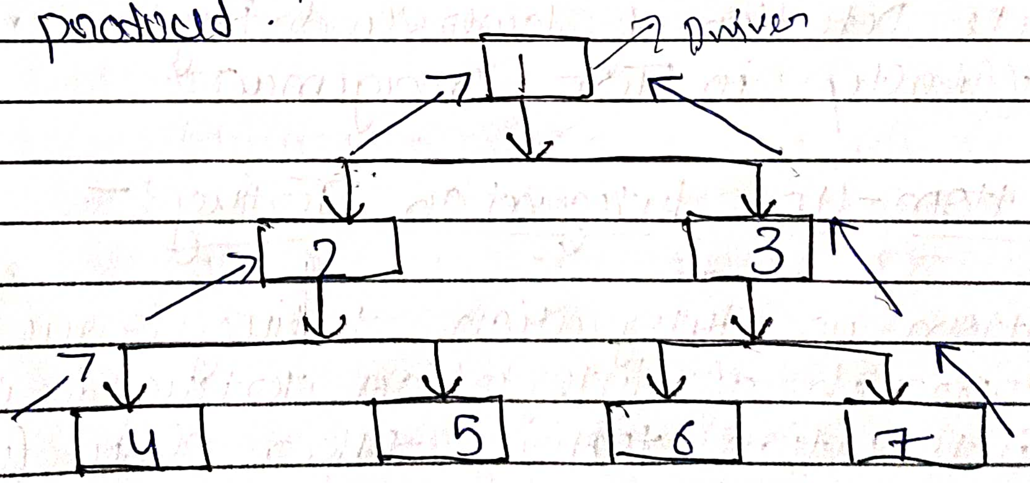
tested first and then the higher level modules are tested and then the modules are integrated accordingly.

- In this testing drivers are used for simulate the main module if the main module is not developed means Driver works as a momentary replacement.

- It is beneficial if the crucial flaws encounter towards the bottom of the program.

- In This approach different module are created first then these modules are integrated with the main function.

- It is implemented on OOP language.
- It works on small to big component.
- In this approach Driver module must be produced.



Adv
- Early Problem Detection
- No need for stubs.

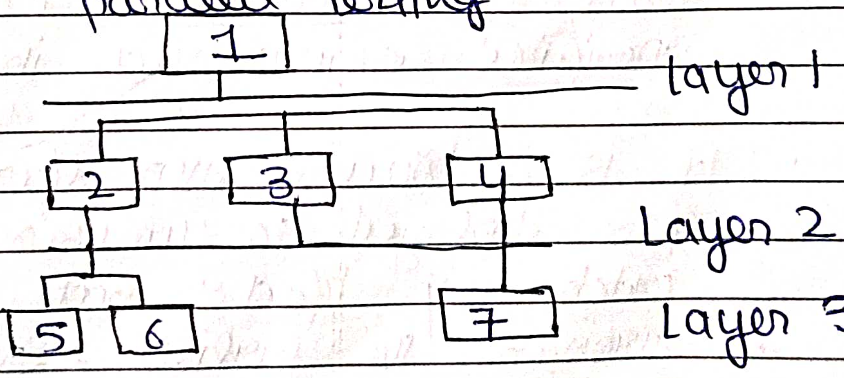
Page No.
Date: / /

- Simultaneous Development and Testing.

Disadv:
- Early Problem Detection.
- Need for Drivers
- Late Detection of Higher level Bugs
- Incomplete system overview.

Sandwich Integration testing

- The combination of top-down and bottom-up integration testing techniques is called sandwich integration
- It is called Hybrid Integration testing.
- This system is viewed as three layers just like a sandwhich
- It makes use of both stubs as well as drivers
- This upper layer of sandwhich use top down integration the lower layer of the sandwich integration use bottom up integeration and stubs and drivers are used to replace the missing modules in the middle layers
- It allows parallel testing.



- It is time saving approach.

Page No.:
Date : / /

Adv

- It allows parallel testing
- It is a time-saving approach.
- It can be used to test system with complex dependencies.
- It can be used to test system with complex algorithm.
- It is a well-defined process that can be easily repeated and documented

Disadv

- In sandwich testing the need for stubs and drivers is very high.
- It is expensive to set up and maintain the necessary infrastructure.

⟹ System Testing :-

- System Testing is a level of software testing where the complete and integrated software system as a whole is tested to evaluate its compliance with specified requirement in SRS.

- It is a crucial step before the software gets deployed to the user, aiming to catch any defects that might have slipped through the earlier stage of testing.

Page No.

Date / /

- System testing is performed in an environment that closely resembles the real-world or production environment.

- Generally, it is performed by an independent testers who haven't been involved in the development phase to ensure unbiased testing.

- It may include functional testing, usability testing, performance testing, security testing and compatibility testing.

⇒ **User Acceptance Testing :-**

- Acceptance testing is a level of software testing where a system is tested for acceptability.
- Testing done by users, customers or other authorized entities to determine application / software needs and business processes.
- Acceptance testing is the most important phase of testing as this decides whether the client approves the application / software or not.
- Acceptance testing is the last phase of software testing performed after system testing and before making the system available for actual use.

Page No.:
Date: / /

Steps to perform Acceptance Testing:

- Requirement Analysis
- Test plan Creation
- Test case Designing
- Test case Execution
- Confirmation of objectives

⟹ Regression Testing :-

- Regression testing is a type of software testing carried out to ensure that previously developed and tested software still functions as expected after making changes, such as updates or bug fixes

- The main goal is to identify any issues or defects introduced by changes in the software, and to ensure that the changes have not disrupted any existing functionality.

- Types of regression testing include unit regression, partial regression and complete regression testing.

- Regression testing generally involves re-running previously completed tests and verifying the program behaviour has not changed as a result of the newly introduced changes

Page No.
Date: / /

- Due to the repetitive nature of these tests, regression testing is often automated to improve efficiency and accuracy.

⇒ Need of Regression Testing :-

- It ensures that the fixed bugs and issues do not reoccur
- New features is added to the software
- Defect fixing
- Performance issue fixing.

Adv :

- It ensures that no new bugs have been introduced after adding new functionalities to the system
- It helps to maintain the quality of the source code

Disadv :

- It can be time and resource - consuming if automated tools are not used
- It is required even after very small changes in the code.

Page No.:
Date: / /

⇒ Difference between Performance testing and Functional testing:

⇒ Functional Testing:-

- Purpose :- Verifies that software function as intended and meets specified requirements.
- Focus :- Tests individual function or features to ensure correct behavior
- Scope: Includes unit testing, integration testing, system testing, etc.
- Testing Criteria: Validates functionality, user interface, data handling, etc.
- Examples: Unit testing, integration testing, system testing, acceptance testing, etc.
- Key Metrics: Pass / fail based on functional requirements.
- Users Perspective: Concerned with that the system does.
- Tools: Selenium, JUnit, TestNG, etc.

Page No.
Date. / /

⇒ Performance Testing :

- Purpose : Evaluates the system's performance under various conditions like load, stress, and scalability.

- Focus : Measures responsiveness, speed, and stability of the entire system.

- Scope : Involves load testing, stress testing, scalability testing, etc.

- Testing Criteria : Validates functionality, user interface data handling, etc.

- Examples : Load testing, stress testing, endurance testing, scalability testing, etc.

- Key Metrics : Response time, throughput, resource utilization, error rates, etc.

- Users Perspective : Concerned with how well the system performs under different conditions.

- Tools : Apache JMeter, LoadRunner, Gatling, etc.

Page No.:
Date: / /

Top-Down and Bottom-Up Testing strategies:

⇒ Structural testing (White Box testing)

- White Box Testing is a software testing method in which the internal structure/design/implementation of the item being tested is known to the tester.
- Code implementation is necessary for white box testing.
- It is mostly done by software ~~testers~~. developer.
- Knowledge of implementation is required.
- It is the inner of the internal software testing
- It is a structural test of the software.
- This type of testing of software is started after a detail design document.
- It is mandatory to have knowledge of programming.
- It is the logic testing of the software.
- It is generally applicable to the lower level of software testing.
- It is most time consuming.
- It is suitable for algorithm testing.
- Data domains along with inner or internal boundaries can be better tested.
- Example :- By input to check and verify loops.

Page No.
Date: / /

⇒ Black Box Testing :- (functional testing)

- Black Box Testing is a software testing method in which the internal structure / design of the item being tested is not known to the tester
- Only the external design and structure are tested.
- Implementation of code is not needed for black box testing.
- It is mostly done by software testers.
- No knowledge of implementation is needed.
- It can be referred to as outer or external software testing.
- It is a functional test of the software.
- This testing can be initiated based on the requirement specifications document.
- No knowledge of programming is required.
- It is the behaviour testing of the software
- It is applicable to the higher levels of testing of software.
- It is also called closed testing.
- It is least time consuming.
- It is not suitable or preferred for algorithm testing.
- Can be done by trial and error ways and methods.
- Example :- Search something on google by using keywords.

Page No.:
Date: / /

Test Driver and Test Stubs :-

- The stubs and Drivers are considered as elements which are equivalent to to-do modules that could be replaced if modules are in their developing stage, missing or not developed yet, so that necessity of such modules could be met.

- Drivers and stubs simulate features and functionalities, and have ability to serve features that a module can provide.

- This reduces useless delay in testing and makes the testing process faster

- Stubs are mainly used in Top-Down integration testing, thus ~~inc~~ while the Drivers are used in Bottom-up integration testing, thus increasing the efficiency of testing process

⇒ Stubs:

- Stubs are developed by software developers to use them in place of modules, if the respective modules aren't developed, missing in developing stage, or are unavailable currently while Top-down testing of modules

- A stub simulates module which has all the capabilities of the unavailable module.

- Stubs are used when the lower-level modules are needed but are unavailable currently.

⇒ Stubs are divided into four basic categories based on what they do:

- Shows the traced messages,
- Shows the displayed message if any,
- Returns the corresponding values that are utilized by modules,
- Returns the value of the chosen parameters (arguments) that were used by the testing modules

⇒ <u>Drivers</u> :-

- Drivers serve the same purpose as stubs, but drivers are used in Bottom-up integration testing and are also more complex than stubs.

- Drivers are also used when some module are missing and unavailable at time of testing of a specific module because of some unavoidable reasons, to act in absence of required module

- Drivers are used when high-level modules are missing and can also be used when lower-level modules are missing

Difference between Stubs and Drivers :-

Stubs :

- Stubs are used in Top-down Integration Testing.
- Stubs are basically Known as a "called program" and are used in the top-down integration testing.
- Stubs are similar to the modules of the software, that are under development process
- Stubs are basically used in the unavailability of low-level modules
- Stubs are taken into use to test the feature and functionality of the module.
- The stubs are taken into concern if testing of upper-levels of the modules are done and the lower-level of the modules are under developing process.
- Stubs are used when lower-level of modules are missing.

Drivers :-

- Drivers are used in Bottom-Up integration testing.
- while, drivers are the "calling program" and are used in bottom up integration testing.
- while drivers are used to invoking the component that needs to be tested.

Page No.
Date: / /

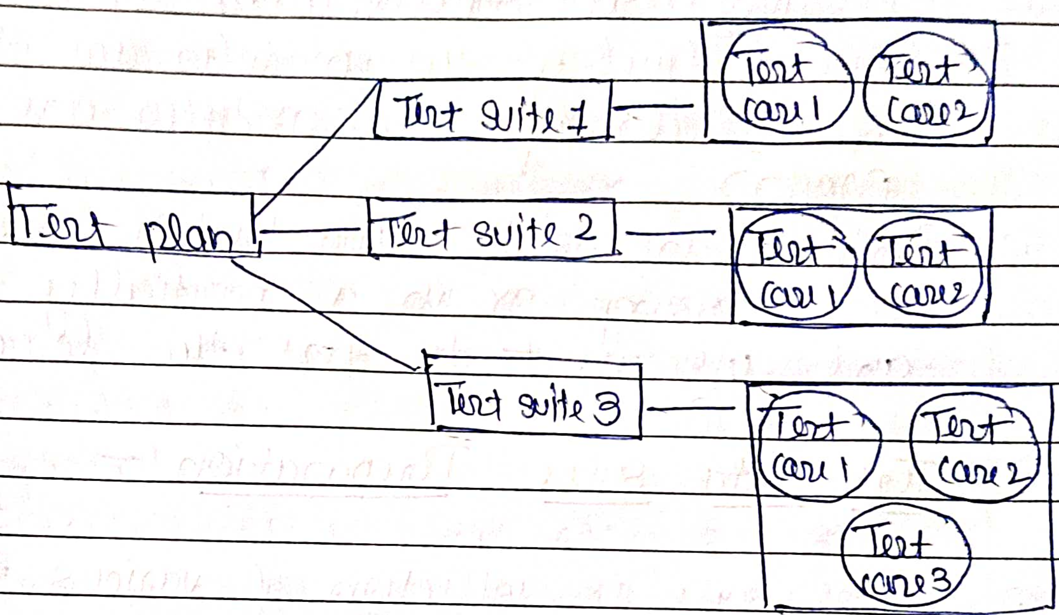
- while drivers are mainly used in place of high level modules and in some situation as well as for low-level modules
- whereas the drivers are used if the main module of the software isn't developed for testing
- The drivers are taken into concern if testing of lower-level of the modules are done and the upper-levels of the modules are under developing process
- Drivers are used when higher-level of modules are missing or in a partially developed phase, and we want to test the lower (sub)-module

⇒ **Test Data Suite Preparation:-**

- A test suite is collection of various test cases that are intended to test a software product or specific functionality / feature, once the software development is accomplished.
- Test suite is a container that has a set of tests which helps testers in executing and reporting the test execution status
- It can take any of the three states namely Active, Inprogress and completed.
- A test case can be added to multiple test suites and test plans. After creating a test plan, test suites are created which in turn can have any number of tests

(i) Test suites are created based on the cycle or based on the scope. It can contain any type of tests, viz- functional or Non-functional.

TEST SUITE DIAGRAM



characteristic of Test suite :-

- Test suites are created after the test plan
- Includes a number of tests and test cases.
- Describe the goals and objective of test cases
- Covers test parameters like application, environment, version etc

Template: The template for a suit case can either be predefined or can be created by the team, as per the requirement of their project.

→ Test Suite Summary
→ Test suite Design
→ Formal Review

Page No.
Date / /

- Pre condition and Post condition
- Expected Results
- Risk Assessment / Analysis
- Test cases
- Documents and Reports

Alpha and Beta Testing :-

⇒ Alpha Testing :-

- Alpha testing is one of the most common software testing strategies used in software development.

- It is specially used by product development organizations.
- This test takes place at the developer's site.
- Developer observe the users and note problem
- Alpha testing is testing of an application when development is about to complete.
- Minor design changes can still be made as a result of alpha testing.

Adv :

- Provides better view about the reliability of the software at an early stage.
- Helps simulate real time user behaviour and environment.

Disadv :

- In depth functionality of the software cannot be tested as it is still under development stage.

Page No.:
Date: / /

⇒ Beta Testing :-

- Beta testing is also known as field testing
- It takes place at a customer's site.
- It sends the system / software to users who install it and use it under real-world working conditions.
- Beta testing of a product is performed by "real users" of the software application in a "real environment" and can be considered as a form of external user Acceptance testing.
- Beta version of the software is released to a limited number of end-users of the product to obtain feedback on the product quality.
- Beta testing reduces product failure risks and provide increased quality of the product through customer validation.

Adv :
- Reduce product failure risk via customer validation
- Improves product quality via customer feedback.

Disadv :
- finding the right beta users and maintaining their participation could be a challenge.

Page No.
Date / /

⇒ Difference between Alpha and Beta Testing!

⇒ Alpha Testing :-

- Alpha testing involves both the white box and black box testing.
- Alpha testing is performed by testers who are usually internal employees of the organization
- Alpha testing is performed at the developer's site
- Reliability and security testing are not checked in aplha testing.
- Alpha testing ensures the quality of the product before forwarding to beta testing.
- Alpha testing requires a testing environment or a lab
- Alpha testing may require a long execution cycle
- Developers can immediately address the critical issues or fixes in alpha testing.
- Multiple test cycles are organized in alpha testing

⇒ Beta Testing !-

- Beta testing commonly uses black-box testing.
- Beta testing is performed by clients who are not part of the organization
- Beta testing is performed at the end user of the product
- Reliability, security and robustness are checked during beta testing

Page No.:
Date: / /

- Beta testing also concentrates on the quality of the product but collects users input on the product and ensures that the product is ready for real time users
- Beta testing doesn't require a testing environment or lab.
- Beta testing requires only a few weeks of execution
- Most of the issues or feedback collected from the beta testing will be implemented in future version of the product
- Only one or two test cycles are there in beta testing.

=> Static testing strategies :-

- Static testing is a type of software testing method which is performed to check the defects in software without actually executing the code of the software application

=> Formal Technical Review (Peer Reviews)

- A FTR is a structured, systematic and disciplined approach to examining and evaluating software artifacts, such as code, design documents, or requirements specifications, with the primary goal of identifying and addressing defects, inconsistencie

Page No.
Date / /

or areas for improvement

- FTRs are conducted by a team of peers, consisting of the artifact's author and other software professionals, who analyze the work product and provide constructive feedback to ensure high-quality software development.

- Formal Technical reviews play a crucial role in improving software quality, detecting issues early in the development lifecycle, promoting knowledge sharing, and fostering collaboration within software engineering teams.

⇒ Types of Formal Technical Reviews:
- Code review
- Walkthrough
- Inspection
- Pair programming

The formal Technical review process:

A. Planning
- Establish objectives
- Select Participants
- Set schedule

B. Preparation
- Distribute materials
- Review guidelines
- Allocate time for individual review

Page No.:
Date: / /

C. Conducting the review
- Discuss objectives
- Review findings
- Make decisions on action items

D. Post - review activities
- Document review results
- Implement action items
- Monitor follow-up actions

⇒ Role in FTR.
- Review leader
- Author
- Reviewers
- Recorder

⇒ Benefits of FTR.
- Improved code quality
- Early defect detection
- Knowledge sharing
- Increased team collaboration.

⇒ Challenges and Best Practices in FTR.
- Time management
- Maintaining a constructive environment
- Addressing bias
- Ensuring consistency in review standards

Page No.
Date / /

⇒ Walk Through :-

- In SE, a walkthrough is a type of informal review process in which the author of a software artifact, such as code, design, document or requirements specifications, present their work to a group of peers.

- The main purpose of a walkthrough is to identify defects, inconsistencies, or areas for improvement through a collaborative discussion.

- Walkthrough are typically less formal than other review methods, such as inspections, and focus on knowledge sharing, collaboration and training among team members.

⇒ Walk through Process:

A. Planning
- Define objectives
- Select participants
- Schedule the walkthrough

B. Preparation
- Distribute materials
- Review guidelines
- Allocate time for individual review

C. Conducting the walkthrough
- Present the work Product
- Discuss finding
- Record issues and suggestion

Page No.:
Date: / /

D. Post-walkthrough activities.
- Summarize findings
- Assign action items
- Monitor follow-up actions

Roles in walkthrough :-
- Presenter
- Reviewer
- Recorder.

Benefits of walkthroughs :-
- Early defect detection
- Knowledge sharing
- Team collaboration
- Training and mentoring

⇒ Code Inspection :-

- Code inspection is a systematic review process in which a team of developers evaluates a software product's source code for potential issues, such as errors, vulnerabilities, and deviations form coding standard

objectives :
- Improve code quality
- Detect and fix defects early in the development cycle
- Share knowledge and best practices among team members
- Enforce coding standards and guidelines

Page No.
Date / /

⇒ Process :

- Planning : Select the code to be inspected, define goals and assemble the inspection team
- Preparation : Team members review the code individually to identify potential issues
- Inspection Meeting : The team discuss the identified issues, and the moderator notes down agreed-upon action items
- Rework : The original developer addresses the identified issues and submits the revised code.
- Follow-up : The moderator verifies that all action items have been addressed and closes the inspection.

Inspection Team Roles :

- Author : The developer who wrote the code being inspected
- Moderator : The person who leads the inspection process and ensures it runs smoothly
- Reviewers : Other developers who provide insights the suggestions for improvements

Page No.:
Date : / /

- Recorder : The person responsible for documenting the issues found and decision made during the inspection

Benefits :

- Enhanced code quality and maintainability
- Reduced development costs and project risk
- Faster time-to-market due to early detection of defects
- Improved team collaboration and learning

Limitations :-

- Time-consuming process
- Possibility of human errors or oversights
- Potential for conflict among team members
- May not catch all types of defects, such as performance or concurrency issues

Compilance with Design and coding standard

- Compilance with design and coding standard refers to adhering to a set of rules, guidelines, and best practices that govern the process of designing and writing software code, ensuring consistent, high-quality, and maintainable software product.

Importance :-

- Improves code readability and maintainability

Page No.
Date. / /

- facilitates collaboration and communication among team members
- Minimizes the introduction of defects and vulnerabilities
- Reduces development time and cost.

Design Standards :-

- Architectural consistency: Ensuring that the overall structure of the software adheres to established pattern and principles.

- Component modularity :- Encouraging a modular design that promotes separation of concerns and code reusability

- Interface design :- Defining clear, consistent and easy-to-understand interface for component or modules

- Scalability and Performance :- Designing software to handle future growth and changing requirement without -vely impacting performance.

Compilance Enforcement :-

- Code reviews :- Conducting regular peer reviews to ensure adherence to design and coding standards.

Page No.:
Date: / /

- **Automated tools:** Using static code analysis and linear tools to identify deviations from established standards.

- **Continous integration:** Integrating and testing code change frequently to catch issues early in the development process

- **Training and education:-** Providing team member and with training and resources to stay up-to-date on best practices and standards

Coding standards:

- Indentation.

- Inline comment

- Use of global

- Structured programming

- Naming Conventions for gb, lb.

- Errors & Exception handling.

Page No.
Date / /

Coding Guidelines

- Line length
- Spacing
- Code is well documented
- The length of any function should not exceed 10 source line.
- Do not use goto statements
- Inline Comments
- Error Message.

Boundary Value Analysis [AKTU 2021-22]

- Boundary Value Analysis is a software testing technique that focuses on the values at the boundaries of the input domain.
- The theory behind BVA is that errors are more likely to occur at the extremes of an input domain rather than in the center.
- Hence it's generally more useful to focus on testing the boundary values.
- BVA is used for testing ranges and data

array elements in a software application

- In practice, BVA can be applied by identifying all the boundaries and then creating test cases for the boundary values and just above and below the boundary values.

- Example :- Let's consider a simple application that accepts an integer input from 1 to 100.

- The boundary values here would be 0 (just below the valid range), 1 (lower limit), 100 (upper limit), and 101 (just above the valid range)

- You would then create test cases to input these values and verify the system's behaviour

⇒ Significance :-

- Error Detection
- Efficiency.
- Test coverage.
- Quality Assurance.
- Compliance