REVIEW OF MANUAL TESTING

Audit: This is an inspection/assessment activity that verifies compliance with plans, policies and procedures and ensures that resources are conserved.

Brainstorming: An individual but usually group process for generating creative and diverse ideas.

Desk Check: A verification technique conducted by the author of the artifact to verify the completeness of their own work.

Baseline: A quantitative measure of the current level of performance.

Benchmarking: Comparing your company's products, services or processes against best practices or competitive practices, to help define superior performance of a product, service or support processes.

Functional Testing: Testing that ensures all functional requirements are met without regard to the final program structure.

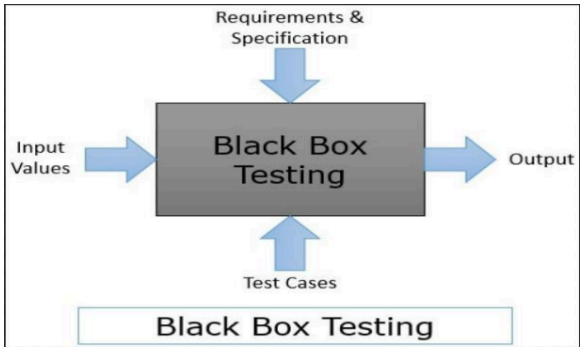
Common Functional Testing Techniques

- Smoke testing
- Sanity testing
- Black box testing
- Integration testing

Black-box Testing: A test technique that focuses on testing the functionality of the program component or application against its specifications without knowledge of how the system is constructed.

Types of Black-box Testing

- Functional testing
- Non-functional
- testing Regression testing



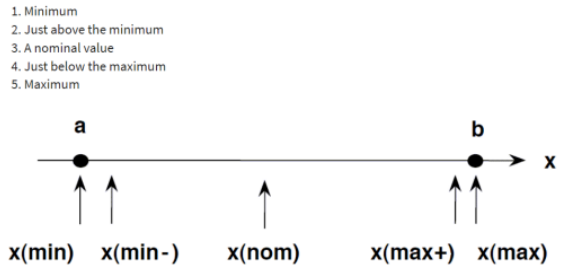
Black-box Testing Techniques

- Equivalence Partitioning
- Boundary Value Analysis
- Decision Table Testing
- State Transition Testing
- Use Case Testing Error Guessing

Equivalence Partitioning: A test technique that utilizes a subset of data that is representative of a larger class.

Boundary value analysis: A data selection technique in which test data is chosen from the "boundaries" of the input or output domain classes, data structures and procedure parameters. Choices often include the actual minimum and maximum boundary values, the maximum value plus or minus one and the minimum value plus

or minus one.



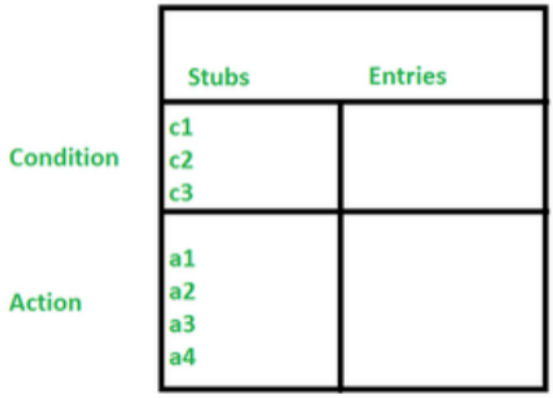
Decision Table: A tool for documenting the unique combinations of conditions and associated results in order to derive unique test cases for validation testing.

Types of Decision Table

1. Limited Entry
2. Extended Entry

4 Parts of Decision Tables

1. Condition Stubs
2. Action Stubs
3. Condition Entries
4. Action Entries



State Transition Testing: State Transition Testing exhibits the various states of scenario/system and possible transition between them.

Use Case Testing: A black box testing technique in which changes made in input conditions cause state changes or output changes in the Application Under Test(AUT).

Use Case Testing Rules

- The name of an actor or a use case must be meaningful and relevant to the system.
- Interaction of an actor with the use case must be defined clearly and in an understandable way

USE CASE TESTING

Technique	Model	Typical defects that can be found using this technique
Equivalence partitioning	Domain model	Wrong handling of data/domain values
Boundary value analysis	Domain model	Wrong handling of data on the domain boundaries
Decision table testing	Decision logic model	Wrong handling of the business rules
State transition testing	Behavioral model	Wrong handling of transitions between states
Use case testing	Business process model	Wrong handling of the business processing

Error Guessing: Test data selection techniques for picking values that seem likely to cause defects. This technique is based upon the theory that test cases and test data can be developed based on intuition and experience of the tester.

Integration Testing: This test begins after two or more programs or application components have been successfully unit tested. It is conducted by the development team to validate the interaction or communication/flow of information between the individual components which will be integrated.

Types of Integration Testing

- Big Bang Approach
- Incremental Approach
    - Top Down Approach
    - Bottom Up Approach
    - Sandwich Approach

Non-Functional Testing: A type of software testing that verifies non functional aspects of the product, such as performance, stability, and usability.

White-box testing: The opposite of black-box testing. It focuses on the internal structure of a test object, rather than just analyzing the inputs and outputs.

Techniques of White Box Testing

- Statement based
    - Static
    - Dynamic
- Decision-based
    - Static
    - Dynamic

Statement Coverage: The statement based technique focuses on the individual lines of code to be delivered. The idea is to analyze the actual lines of code in the software program and have tests that verify these lines.

Decision Coverage: A white-box testing technique that measures the number of – or percentage – of decision directions executed by the test case designed. 100% Decision coverage would indicate that all decision directions had been executed at least once during testing.

Static vs Dynamic

Whether you use statement or decision-based techniques there is another choice to make.

Static Analysis: Static is a manual approach. This relies on the developer going through the code and eyeballing the code to see obvious errors. This can be done by the creator of the code.

Dynamic Analysis: Analysis performed by executing the program code.

Entrance Criteria: Required conditions and standards for work product quality that must be present or met for entry into the next stage of the software development process.

Exit criteria: Standards for work product quality which block the promotion of incomplete or defective work products to subsequent stages of the software development process.

Force Field Analysis: A group technique used to identify both driving and restraining forces that influence a current situation.

Formal Analysis: Technique that uses rigorous mathematical techniques to analyze the algorithms of a solution for numerical properties, efficiency, and correctness.

Inspection: A formal assessment of a work product conducted by one or more qualified independent reviewers to detect defects, violations of

development standards, and other problems.

Life Cycle Testing: The process of verifying the consistency, completeness, and correctness of software at each stage of the development life cycle.

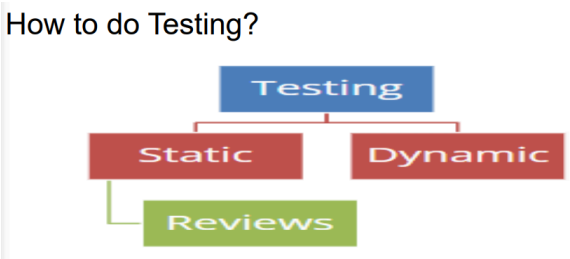
Pass/Fail Criteria: Decision rules used to determine whether a software item or feature passes or fails a test.

What are the scopes of testing?

Software Testing: A way to assess the quality of the software and to reduce the risks of software failure in operation.

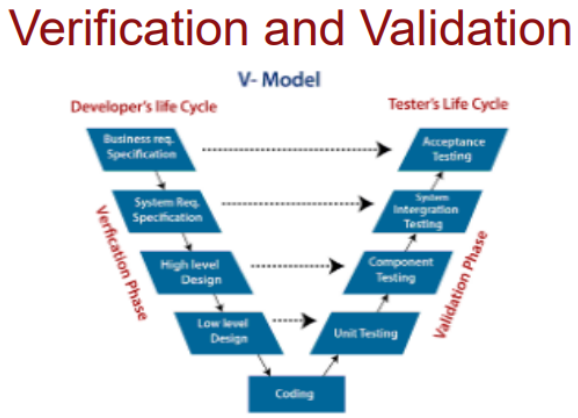
Software Testing Inclusions:

- Test planning
- Analyzing
- Designing
- Implementing tests
- Reporting test progress and results
- Evaluating the quality of a test object



Static testing: Does not involve the execution of the component
Ex. Code reviews, Requirements review, design review

Dynamic Testing: The execution of the component or system being tested
Ex. Accepting invalid values, Run time errors.



Testing Objectives

1. To prevent the defects
2. To verify all the specified requirements have been fulfilled
3. To check whether the test object is complete
4. To build confidence
5. To provide sufficient information
6. To comply with contractual, legal, or regulatory requirements

Component Testing: To find as many failures as possible so that the underlying defects are identified and fixed early.

Acceptance Testing: To confirm that the system works as expected and satisfies requirements.

Testing Scope: A list of product features, product parts, or product-related integrations that must

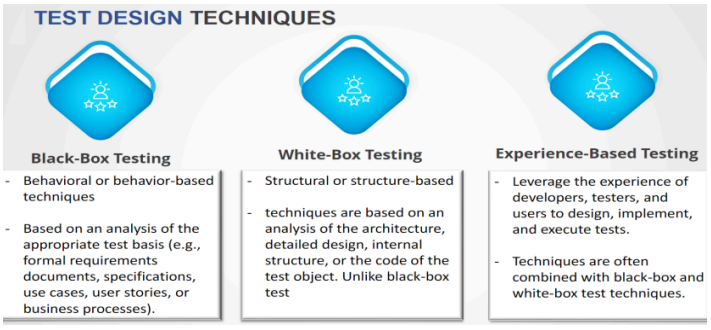
be tested in order to build a reliable assessment of a product's quality.

Tips to Determine the Scope of Software Testing

Tip #1: Clearly Outline the Purpose of the Software
Tip #2: Define a Testing Timeline
Tip #3: Nail Down a Budget
Tip #4: Know When to Stand and When to Swim

Different Test Design Techniques

Static Test Design Techniques: Include reviews, walkthroughs, Formal reviews etc.
Dynamic Test Design Techniques: Include equivalence partitioning (EP), boundary value analysis (BVA), decision table testing, state transition testing etc.



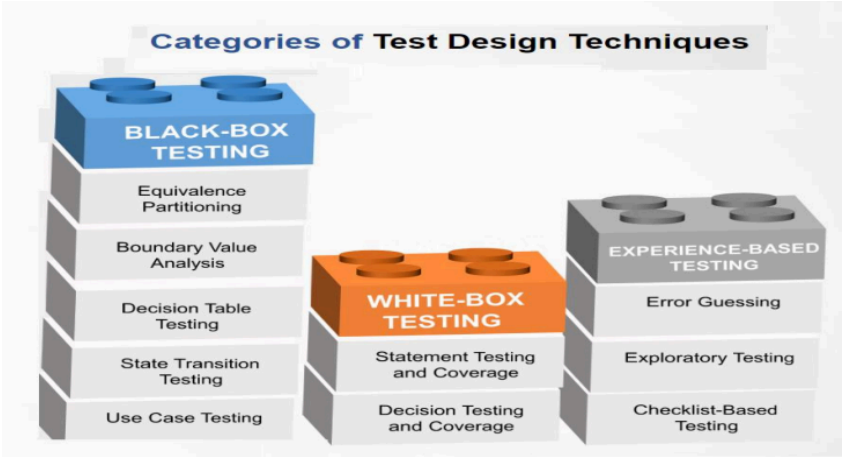
EQUIVALENCE VALUE PARTITIONING
A technique that divides the input data of a software unit into partitions of equivalent data from which test cases can be derived.

BOUNDARY VALUE ANALYSIS
Boundary testing is the process of testing between extreme ends or boundaries between partitions of the input values.

DECISION TABLE
❏ software testing methodology used to test system behavior for various input combinations
❏ represented in tabular form
❏ also called a Cause-Effect table

STATE TRANSITION DIAGRAM

- Shows all valid transitions and potentially invalid transitions between states, as well as the events, and resulting actions for valid transitions.
- A state is just what it says it is: the system is 'static', in a stable condition from which it will only change if it is stimulated by an event of some kind changing from one state to another.



Software
a collection of computer programs that helps us to perform tasks

TYPES OF SOFTWARE

1. System Software
2. Programming Software
3. Application Software

Software Testing
An activity to detect and identify the defects in the software

Software Quality

- Bug-free
- Delivered on time
- Within budget
- Meets requirements and/or expectations
- Maintainable

Project VS. Product

There is a software application that is developed for specific customers based on the requirements.
There is software application that is developed for multiple customers based on market requirements

Reason for Testing: To deliver quality products.

Quality Assurance (QA): Deals with 'prevention' of defects in the product being developed

Quality Control (QC): Its focus is defect detection and removal. Testing is a quality control activity.

Quality Improvement: To change a production process so that the rate at which defective products (defects) are produced is reduced.

ERROR, BUG/DEFECT VS. FAILURE

Error

- human mistakes
- characteristics of human actions
- indirect human action

Error or Defect

1. A discrepancy between a computed, observed or measured value or condition and the true, specified or theoretically correct value or condition
2. Human action that results in software containing a fault

Bugs/Defects: Related to an application which shows deviation from expected actual behavior.

Failure: Deviation that was identified by the end-user after sometime using the app.

Backlog: A list of tasks required to support a larger strategic plan. For example, a product development context contains a prioritized list of items. The product team agrees to work on these projects next.

Commercial Off-The-Shelf (COTS)
Synonyms: off-the-shelf software
A software product that is developed for the general market, i.e. for a large

number of customers, and that is delivered to many customers in identical format

Component integration testing
Testing performed to expose defects in the interfaces and interaction between integrated components

Component Testing
The testing of individual software components

Data-driven Testing: A scripting technique that stores test input and expected results in a table or spreadsheet, so that a single control script can execute all of the tests in the table.

Defect density
Synonyms: fault density
The number of defects identified in a component or system divided by the size of the component or system (expressed in standard measurement terms, e.g., lines-of code, number of classes or function points)

Epic: Allow you to keep track of large, loosely defined ideas in your backlog without the need to overpopulate your backlog with multiple items.

Exhaustive testing
Synonyms: complete testing
A test approach in which the test suite comprises all combinations of input values and preconditions.

Factory acceptance testing
Acceptance testing conducted at the site at which the product is developed and performed by employees of the supplier organization, to determine whether or not a component or system satisfies the requirements, normally including hardware

High-level test case
A test case without concrete (implementation level) values for input data and expected results. Logical operators are used: instances of the actual values are not yet defined and/or available.

Incremental Testing: Testing where components or systems are integrated and tested one or some at a time, until all the components or systems are integrated and tested

Level test plan: A test plan that typically addresses one test level

Load testing
A type of performance testing conducted to evaluate the behavior of a component or system with increasing load, e.g., numbers of parallel users and/or numbers of transactions, to determine what load can be handled by the component or system.

Maintenance testing: Testing the changes to an operational system or the

impact of a changed environment to an operational system.

Master test plan
A test plan that typically addresses multiple test levels.

Memory leak: A memory access failure due to a defect in a program's dynamic store allocation logic that causes it to fail to release memory after it has finished using it, eventually causing the program and/or 16 other concurrent processes to fail due to lack of memory.

Non-functional testing: Testing the attributes of a component or system that do not relate to functionality, e.g., reliability, efficiency, usability, maintainability and portability.

Operational testing: Testing conducted to evaluate a component or system in its operational environment.

Peer review: A review of a software work product by colleagues of the producer of the product for the purpose of identifying defects and improvements. Examples are inspection, technical review and walkthrough.

Post-execution comparison
Comparison of actual and expected results, performed after the software has finished running.

Postcondition: Environmental and state conditions that must be fulfilled after the execution of a test or test procedure.

Precondition: Environmental and state conditions that must be fulfilled before the component or system can be executed with a particular test or test procedure.

Requirements-based testing
An approach to testing in which test cases are designed based on test objectives and test conditions derived from requirements, e.g., tests that exercise specific functions or probe non-functional attributes such as reliability or usability.

Review: An evaluation of a product or project status to ascertain discrepancies from planned results and to recommend improvements. Examples include management review, informal review, technical review, inspection, and walkthrough.

Root cause: A source of a defect such that if it is removed, the occurrence of the defect type is decreased or removed.

Scribe
The person who records each defect mentioned and any suggestions for process improvement during a review meeting, on a logging form. The scribe should ensure that the logging form is readable and understandable.

Sprint: A set amount of time that a development team has to complete a specific amount of work.

Standard-compliant testing
Testing that complies to a set of requirements defined by a standard, e.g., an industry testing standard or a standard for testing safety-critical systems

State transition testing: A black-box test design technique in which test cases are designed to execute valid and invalid state transitions.

Statement testing: A white-box test design technique in which test cases are designed to execute statements.

Static testing: Testing of a software development artifact, e.g., requirements, design or code, without execution of these artifacts, e.g., reviews or static analysis.

Stress testing: A type of performance testing conducted to evaluate a system or component at or beyond the limits of its anticipated or specified workloads, or with reduced availability of resources such as access to memory or servers.

Structural coverage: Coverage measures based on the internal structure of a component or system.

Stub: A skeletal or special-purpose implementation of a software component, used to develop or test a component that calls or is otherwise dependent on it. It replaces a called component.

System testing: Testing an integrated system to verify that it meets specified requirements.

System Under Test (SUT): test item

Test approach: The implementation of the test strategy for a specific project.

Test automation: The use of software to perform or support test activities, e.g., test management, test design, test execution and results checking.

Test basis: All documents from which the requirements of a component or system can be inferred. The documentation on which the test cases are based. If a document can be amended only by way of formal amendment procedure, then the test basis is called a frozen test basis.

Test case: A set of input values, execution preconditions, expected results and execution post-conditions, developed for a particular objective or test condition, such as to exercise a particular program path or to verify compliance with a specific requirement.

Test closure: During the test closure phase of a test process data is collected from completed activities to consolidate experience, test ware, facts and numbers.

Test condition: An item or event of a component or system that could be verified by one or more test cases, e.g.,

a function, transaction, feature, quality attribute, or structural element.

Test data: Data that exists (for example, in a database) before a test is executed, and that affects or is affected by the component or system under test

Test design: The process of transforming general test objectives into tangible test conditions and test cases.

Test design tool: A tool that supports the test design activity by generating test inputs from a specification that may be held in a CASE tool repository, e.g., requirements management tool, from specified test 30 conditions held in the tool itself, or from code

Test item: The individual element to be tested. There usually is one test object and many test items.

Test log: A chronological record of relevant details about the execution of tests.

Test implementation: The process of developing and prioritizing test procedures, creating test data and, optionally, preparing test harnesses and writing automated test scripts.

Test plan: A document describing the scope, approach, resources and schedule of intended test activities.

Test schedule: A list of activities, tasks or events of the test process, identifying their intended start and finish dates and/or times, and interdependencies.

Test script: Commonly used to refer to a test procedure specification, especially an automated one.

Test specification: A document that consists of a test design specification, test case specification and/or test procedure specification.

Test strategy: A high-level description of the test levels to be performed and the testing within those levels for an organization or program (one or more projects)

Test suite: A set of several test cases for a component or system under test, where the post condition of one test is often used as the precondition for the next one

Test-driven development (TDD): A way of developing software where the test cases are developed, and often automated, before the software is developed to run those test cases.

Testware: Artifacts produced during the test process required to plan, design, and execute tests, such as documentation, scripts, inputs, expected results, set-up and clear-up procedures, files, databases, environment, and any additional software or utilities used in testing.

Use case:A sequence of transactions in a dialogue between an actor and a

component or system with a tangible result, where an actor can be a user or anything that can exchange information with the system.

Walkthrough: A step-by-step presentation by the author of a document in order to gather information and to establish a common understanding of its content

White-box test design technique
Procedure to derive and/or select test cases based on an analysis of the internal structure of a component or test design technique.

User acceptance testing
Acceptance testing carried out by future users in a (simulated) operational environment focusing on user requirements and needs.

5 Whys Software has Bugs

- Miscommunication or No communication
- Software Complexity
- Programming Errors
- Changing requirements
- Lack of skilled Testers

Myth: Anyone can do manual testing
Fact: Testing requires many skill sets

Myth: Testing ensures 100% Defect free product
Fact: Testing attempts to find as many defects as possible. Identifying all possible defects is impossible.

Myth: Automated testing is more powerful than manual testing
Fact: 100% test automation cannot be done. Manual Software Testing is also essential

Myth: Testing is easy
Fact: Testing can be extremely challenging. Testing an application for possible use cases with minimum test cases requires high analytical skills.

Manual Testing	Automated Testing
Manual testing requires human intervention for test execution.	Automation Testing is use of tools to execute test cases
Manual testing will require skilled labour, long time & will imply high costs.	Automation Testing saves time, cost and manpower. Once recorded, it's easier to run an automated test suite
Any type of application can be tested manually, certain testing types like ad-hoc and monkey testing are more suited for manual execution.	Automated testing is recommended only for stable systems and is mostly used for Regression Testing
Manual testing can become repetitive and boring.	The boring part of executing same test cases time and again is handled by automation software in Automation Testing.