2) Types of Testing

Page No. 12
Date

Introduction :-

There are many approaches available in software testing. Reviews, walkthroughs or inspections are referred to as static testing, whereas actually executing programmed code with a given set of test cases is referred to as dynamic testing.

Software testing types :-

1. White box testing
2. Black box testing

2.1 White box testing :- White box testing is the detailed investigation of internal logic and structure of the code. In order to find white box testing on an application, the tester needs to passes language of internal logic of code. The tester needs to have a look inside the source code and find out which unit of code is behaving inappropriately. White box testing includes the following :-

* Control flow testing
* Data flow testing
* Branch testing
* Statement coverage
* Decision coverage
* Modified condition / decision coverage
* Prime path testing
* Path testing.

White box testing is further classified as follows:-

- Static Testing
- Structural testing

- Static Testing :- Static testing of the software work products manually, or with the set of tools but they are not executed. It starts early in the Life cycle and so it is done during verification process. It does not need computer as the testing of program is done without executing the program.

Following are the methods to review Static Testing:

1) Inspection :- It is the most formal type of review. It is led by the trained group of moderators. During inspection, documents are prepared and checked thoroughly by the reviewers before the meeting. A seperate preparation is carried out during which the product is examined and the defects are found. The defects found are documented in a logging list or issue log.

The goals of inspection are :-

* It helps author to improve quality of document under inspection.
* It removes defects efficiently and as early as possible
* It improves the product quality
* It creates common understanding by exchanging information
* Learn from defects found and to prevent occurances of similar defects.

2] Structured Walkthroughs :- It is a static testing technique performed in an organised manner between a group of peers to review and discuss technical aspects of software development process. The main objective is to find defects in order to improve the quality of product. They are NOT usually used for technical discussions or to discuss the solution to be found; because the aim is to detect errors, not to correct errors. When a walkthrough is finished, author of the output is responsible for fixing the issues

Benefits :-

* Saves time and money as defects are found and rectified very early in the life cycle
* This provides value added comments from reviewers with different technical backgrounds and experience
* It notifies the project management team about the progress of development process.

Participants :-

* Author :- The author of the document under review
* Presenter :- The presenter usually develops the agenda for the walkthrough and presents the output being reviewed.

* Moderators:- The moderator facilitates the walkthrough session, ensures the walkthrough agenda is followed, and encourages all the reviewers to participate
* Reviewers:- The reviewer evaluate document under test to determine if it is technically accurate
* Scribe:- The scribe is the recorder of the structured walkthrough outcomes who records the issues identified and any other technical comments, suggestions and unresolved questions.

3] Technical Review:- It is less formal review. It is led by the trained moderator but can also be led by a technical expert. It is often performed as a peer review without management participation. Defects are found by the experts who focus on content of the document.

Goals of Technical review are:-
* To ensure that early stage technical concepts are used correctly
* To access the value of technical concepts and alternatives in the product.
* To have consistency in use and representation of technical concepts.
* To inform participants about the technical content of the document.

	Walkthroughs	Inspection
Participants ①	Author is leader and others are participants	Every participant has some decided role
Rigor ②	It is informal meet	It is formal meet
Training Required ③	No	Yes
purpose ④	To find bugs	To improve quality of product
Effectiveness ⑤	Low	High

Structural testing:-

It is often referred to as white box or glass box or clear box testing because in structural testing we are interested in what is happening inside the application.

In structural testing, testers are required to have the knowledge of internal implementations of the code.

1] Code functional testing:-

It involves debugging sort of activities. All the included activities include must require the code knowledge like loop iterations, conditions, input and corresponding output etc.

Following are some of the methods for code functional testing:-

① Being a developer, we can expect certain obvious input errors. We can perform those tests by passing required inputs and checking the outputs. These quick tests highlight the obvious mistakes. We can repeat above test many times to increase our confidence at next level.

② For complex logic and conditions, we can check correct working of loops and iterations by simply putting print statement at intermediate points. Once concluded that all defects are resolved, remove the print statements.

③ We can use different IDE or debugging tools to resolve early defects from product or module. These tools provides functions which allows us to run module statement by statement.

2] Code Coverage Testing :-

It is a form of testing which inspects code directly and is therefore a form of white box testing. Code coverage measurement simply determines those statements in the body which has been executed and those which have been not.

In general, the code coverage system collects information about the running program and then combines that with source information to generate a report of test suite's code coverage.

The purpose of code coverage analysis is :-
① Finding the areas in the program which has not been executed
② Creating additional test cases to increase coverage
③ Determining a quantitive measure of code coverage, which is an indirect measure of quality

Optional aspects of code coverage analysis is :-
① Identifying redudant test cases that do not increase coverage.
② You use coverage analysis to assure the quality of your set of tests, not the quality of your actual product.
③ It requires access to test program source code and often requires recompiling it with a special command.

Code coverage types :-
A] Program statements and Line Coverage
B] Branch coverage
C] Condition coverage

A] Program statements and line coverage :-
Statement coverage is a code coverage metric that tells you whether the flow of control reached every executable statement of the source code at least once. It identifies which statements in a method or class have been executed.

The ultimate benefit of statement coverage is the ability to identify which blocks of code have not been executed.

The problem with statement coverage, however, is that it does not identify bugs that arise from the control flow constructs in your source code, such as compound conditions or consecutive switch labels. This means you can have 100% coverage and still have glaring, uncaught bugs.

B] Branch Coverage :- It is the outcome of a decision, so branch coverage simply measures which decision outcomes have been tested.

This metric reports whether boolean expressions tested in control structures such as if statement and while statements evaluated to both true and false.

Also this metric includes coverage of switch statements cases, exception and interrupt handler.

C] Condition Coverage :- It reports the true or false outcome of each boolean sub-expressions, seperated by logical-and & logical-or if they occur. Condition coverage measures the sub-expression independently of each other.

Code complexity Testing

It is a type of testing through which system design and system coding is verified. Complexity testing performs the verification through reviews, walkthroughs, or inspection as per planned arrangement.

McCabe's Cyclomatic complexity

McCabe's complexity used to define minimum number of test cases required for a module and it is used during SDLC to quantify maintainability and testability. Cyclomatic complexity is defined as

$$CC = E - N + P$$

Where, E = No. of edges of the graph
N = No. of nodes of the graph
P = No. of connected components

In case of connected graph;

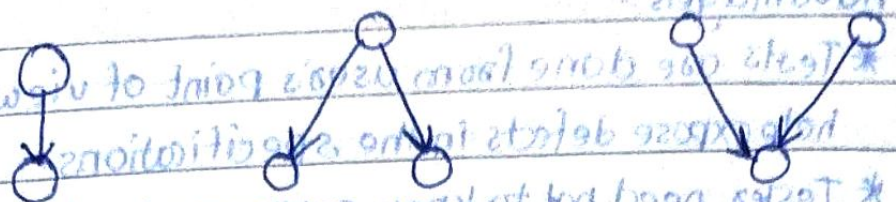
$$CC = E - N + 2$$

Simplified formula, $CC = D + 1$

D = No. of decision points in the graph

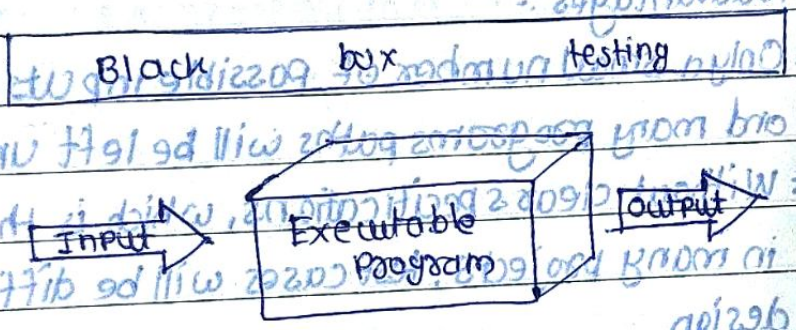
Control Flow Graph (CFG):

It is a graphical representation of program module. Control flow graph is a directed graph in which node represents programs' regions and edges represents flow from one program region to other.



2.2 Black box testing :-

It is also known as Behavioural testing. It is a software testing method in which internal structure of the item being tested is not known to the tester.



This method of attempts to find errors in the following categories:

* Incorrect or missing function
* Interface errors
* Errors in data structures or external database access
* Behaviour or performance errors
* Initialization and termination of errors.

Advantages :-

* Tests are done from user's point of view and will help expose defects in the specifications.
* Tester need not to know programming language or how the software has been implemented.
* Tests can be conducted by a body independent from the developers, allowing for an objective perspective and the avoidance of developer-bias.
* Test cases can be designed as soon as the specifications are completed.

Disadvantages :-

* Only a small number of possible inputs can be tested and many programs paths will be left untested.
* Without clear specifications, which is the situation in many projects, test cases will be difficult to design.
* Tests can be redudant if the software designer/ developer has already run a test case.

Techniques for black box testing :-

1. Requirement Based Testing
2. Positive and Negative testing
3. Boundary Value Analysis
4. Decision Tables
5. Equivalence Partitioning
6. User Documentation Testing
7. Graph Based Testing

① Requirement based Testing :-
It is a testing approach in which test cases, conditions and data are derieved from requirements. It includes functional test and also non-functional attributes such as performance, reliabity and usability.

Stages in Requirement based testing :-
* Defining Test completion criteria
* Design Test cases
* Execute tests
* Verify test results
* Verify test coverage
* Track and manage defects.

Requirement based testing process :-
* Testing must be carried out in a timely manner
* Testing process should add value to software life cycle, hence it needs to be effective.
* Testing the system exhaustively is impossible hence the testing process needs to be efficient as well.
* Testing must provide the overall status of project hence it should be manageable.

② Positive & Negative Testing :-
Positive Testing is testing where the system validated against the valid input of data. The tester always check for only valid set of values and check if the application behaves as expected with its expected inputs.

The main intention of this testing is to check whether software application not showing error when not suppased to and showing error when supposed to

Ex:- Age:- 19
① Enter only numbers

Negative Testing :- Negative Testing is the testing process where the data system validated against the invalid input data. A negative test checks if a application behaves as expected with its negative inputs.

The main intention of this testing is to check whether software application not showing error when suppased to and showing error when not suppased to.

Ex:- Age:- abcdeE8
① Enter only numbers

Scenario :-
If the requirement is saying password text field should accept 6-20 characters and only alphanumerical characters.

* Postive Test Scenarios :-
① Password Textbox should accept 6 characters
② Password Textbox should be upto 20 characters
③ Password Textbox should accept any value in between 6-20 chars in length
④ Password Textfield should accept all numeric and alphanumeric values

*Negative Test Scenarios :-

① Password Textbox should not accept less than 6 characters

② Password Textbox should not exceed more than 20 characters.

③ Password Textbox should not accept special characters

(3) Boundary Value Analysis

It is also called boundary condition data testing and is based on testing at the boundaries between partition

Ex:- Suppose you have a software which accepts values between 1 to 1000, so the valid partition will be (1-1000). equivalence partitions will be like :-

Invalid partion	valid partion	Invalid partion
0	1-1000	1001 & above,

Boundary Value Analysis is a black box test design where test cases are designed using boundary values, BVA is used in range checking.

Ex:- A store in city offers different discounts depending on the purchases made by the individual. If the purchase is in range of $1 to $50 → no discounts
purchase over $50 & upto $200 → 5% discount
purchase over $200 & upt $500 → 10% discount
purchase over $500 → 15% discount

Invalid partition	Valid Partition (No discounts)	Valid partition 5%	Valid Partition 10%	Valid Partition 15%
$0-01	$1 - $50	$51 - $200	$201 - $500	$501 - Above

Boundary Values → 0.00 ... 501

Sub boundary conditions:

It is also called internal boundary analysis. It takes internal sub-boundary of the program input data.

While testing this, tester should take the last possible valid input and with an invalid input just outside the internal boundary value.

Ex:- powers of two ⇒ Powers of Two

(4) Decision Tables:

A decision table is a good way to deal with combination of things. They can be used in test design whether or not they are used in specifications as they help tester to explore the effects of combinations of different inputs and other software states that must correctly implement business rules.

How to use decision tables:

* Identify a suitable function or subsystem which reacts according to the combination of input or events.
* The system should not contain too many inputs or else it may become unmanageable.
* Once identified all the aspects, then combine and put them into a table listing.

Page No. 27
Date

Example of credit card:

Conditions	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7	Rule 8
New customer (15%)	T	T	F	F	T	T	F	F
Loyalty Card (10%)	T	T	F	F	T	T	F	F
Coupon (20%)	T	F	T	F	T	F	T	F
Action/Outcomes								
Discount%	X	X	20	15	30	10	20	0

The conditions and actions are listed in left hand column All other columns in decision table each represent a seperate rule. If the rules and combinations are more, then we are likely to sample them by selecting a rich subset for testing.

⑤ Equivalence Partitioning

It is a software testing technique that divides the input test of data of the application under test into each partition at least once of equivalent data from which test cases are derieved.

The below example best describes the equivalence partitioning :-

* Assume that the application accepts an integer in the range 100 to 999
* Valid equivalence class partitioning :- 100 to 999 inclusive
* Non valid equivalence class partions :-
  less than 100, more than 999, decimal nos, alphabets/non-numeric characters

⑥ User Documentation Testing :-

User Documentation covers all the manuals, user guides, installation guides, setup guides, readme files, software release notes and online help that are released along with software in order to help end user understand it. It should have two objectives :-

* To check if what is available in document is available in the software
* To check if what is there in the product is explained correctly in the document

Page No. 29
Date

① Graph based Testing :-

It is also known as state based testing. Graphs of state provide framework for model based testing. After arriving at executable state graph model, we execute or simulate the state graph model with event sequences as test cases.

Graph based testing is useful to represent a transaction or work flows. Consider an application of leave by employee.

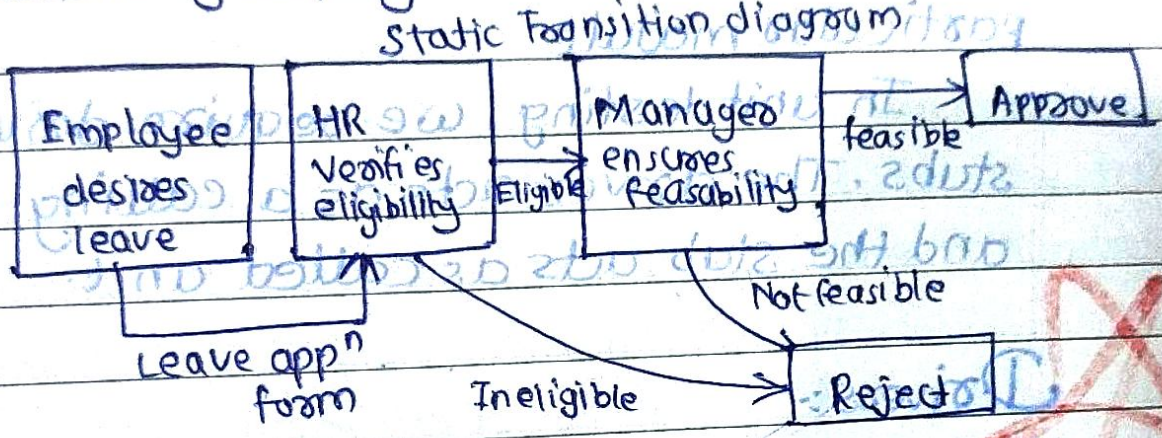


Fig :- Graph testing for employee application system.