Unit - IV

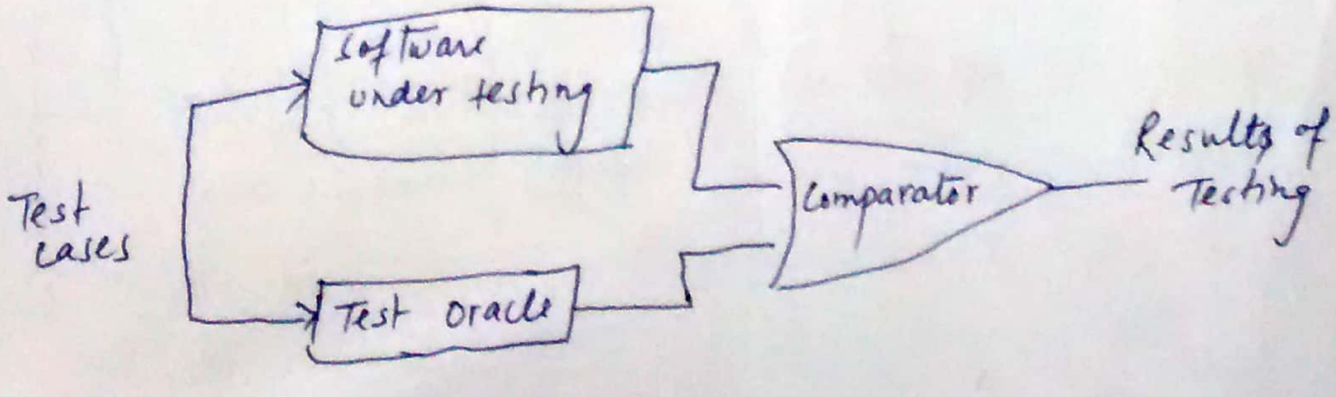
Testing: Testing is the process of executing a program with the intent of finding errors.

Error, Fault and Failure :

Errors refers to the discrepancy between a computed, observed or measured value and the true specified or theoretically correct value. That is error refers to the difference between the actual output of the software and the correct output.

Fault is a condition that causes a system to fail in performing its required function.

Failure is the inability of a system or component to perform a required function according to its specification. A failure is produced only when there is a fault in the system. However, presence of fault does not guarantee a failure.

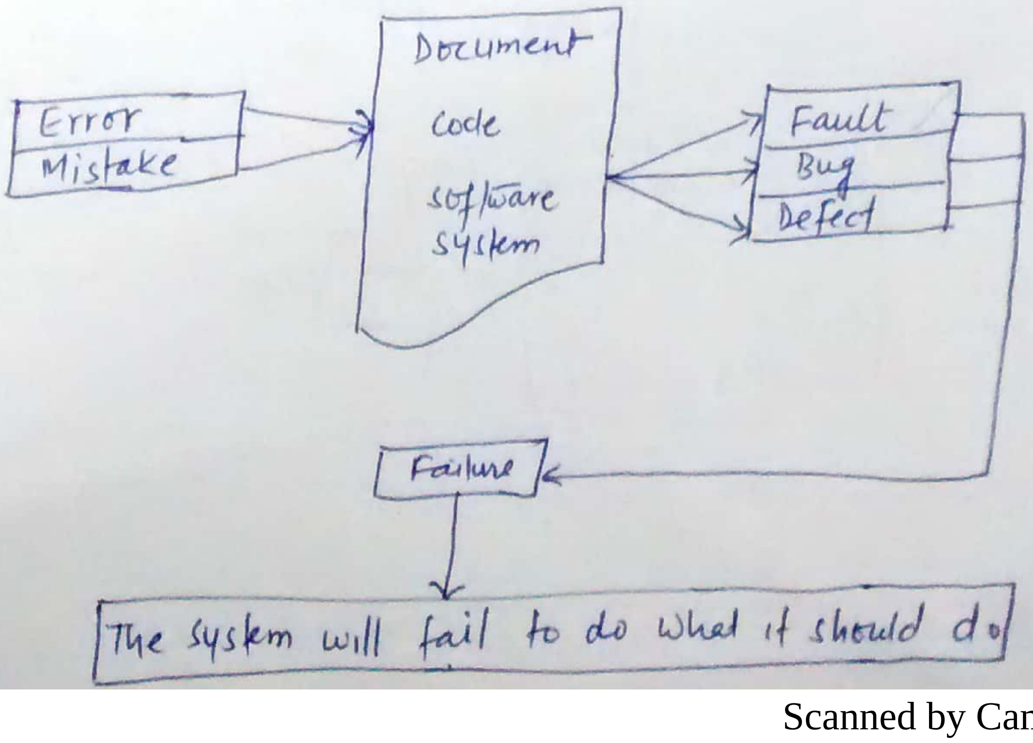


Test Oracles – To test any program, we need to have a description of its expected behavior and a method of determining whether the observed behaviour conforms to the expected behaviour.
For this we need a test oracle.
Test oracle is a mechanism, different from the program itself, that can be used to check the correctness of the output of the program for the test cases.

Test Case and Test Suite

Test case describes an input description and an expected output description.
Test suit is a set of test cases.

$$\boxed{\text{Testing} = \text{Verification} + \text{Validation}}$$



Levels of Testing:

$$\begin{tikzcd}
\text{Client needs} \arrow[d] \arrow[r, leftrightarrow] & \text{Acceptance Testing} \\
\text{Requirements} \arrow[d] \arrow[r, leftrightarrow] & \text{System Testing} \arrow[u] \\
\text{Design} \arrow[d] \arrow[r, leftrightarrow] & \text{Integration Testing} \arrow[u] \\
\text{code} \arrow[r, leftrightarrow] & \text{Unit Testing} \arrow[u]
\end{tikzcd}$$

- Unit Testing
- Integration Testing
- System Testing
- Acceptance testing

① Unit Testing - The first level of Testing is called Unit Testing. In this, different modules are tested against the specification produced during design for the module.

Unit Testing is essentially for verification of the code produced during coding phase, and hence the goal is to test the internal logic of the modules. It is typically done by the programmer of the module.

(2) Integration Testing: In this, many unit tested modu are combined into subsystem, which are then tested. The goal here is to see if the modules can be integrated properly. Hence the emphasis is on testing interfaces between modules. This testing activity can be considered testing the design.

(3) System Testing: Here the entire software system is tested. The reference document for this process is the requirement document and the goal is to see if the software meets its requirements.

(4) Acceptance Testing: Testing here focuses on the external behaviour of the system; the internal logic of the program is not emphasized. Mostly functional testing is performed at this level.

Integration Testing strategy

- Big-bang integration testing
- Top-down integration testing
- Bottom-up integration testing
- sandwich integration testing

Big-bang Integration testing: It is the simplest integration testing approach, where all the modules making up a system are integrated in a single step. ie. all the modules of the system are simply put together and tested. This technique is practicable only for very small systems.

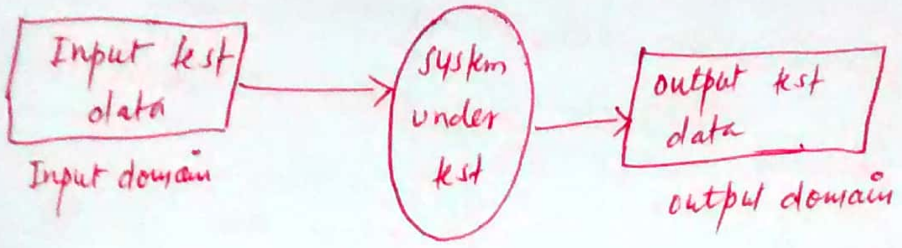
Top-down integration testing: Top-down integration testing starts with the main routine and one or two subroutine in the system. After the top-level skeleton has been tested, the immediate subroutines of the skeleton are combined with it and tested. This approach requires the use of program stubs to simulate the effect of lower level routine that are called by the routines under test. A pure top-down integration approach does not require driver routines.

Bottom-up integration testing -: In bottom-up testing, each subsystem is tested separately and then the full system is tested. The primary purpose of testing each subsystem is to test the interfaces among various modules making up the subsystem. Both control and data interfaces are tested. In pure bottom-up testing no stubs are required, only the test drivers are required.

Sandwich integration testing: A sandwich (also called mixed) integration testing follows a combination of top-down and bottom up testing approaches. In the mixed testing approach, testing can start as and when modules become available, whereas in top-down, testing can start only after the top-level modules have been coded and unit tested and bottom-up can starts only after the bottom level modules are ready. Therefore sandwich integration testing approach is one of the most commonly used integration testing approach.

Functional Testing (Black Box Testing)

Functional testing refers to testing, which involves only observations of the output for certain input values. Functional testing also referred as black box testing in which contents of the black box are not known.



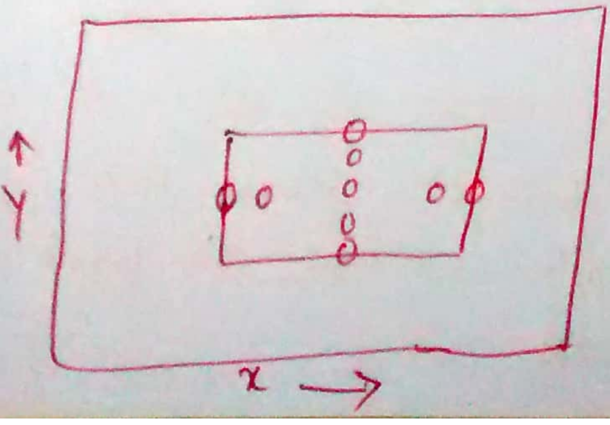
Black Box Testing

Types:

- Boundary Value Analysis
  - Boundary Value checking
  - Robust
  - Worst
- Equivalence Class partitioning
- Decision Table

Boundary value checking (BVC):

total number of test cases = 4n+1 ; n is no of variables

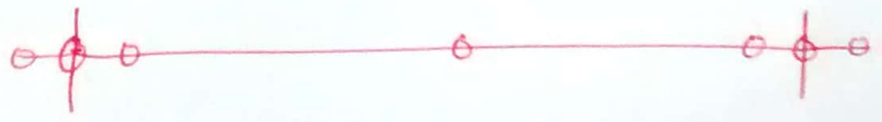


Input value may be on the boundary, just below the boundary (upper side) or just above the boundary (lower side). suppose we have an input variable x with a range from 1-100, The boundary values are 1, 2, 99 and 100.

Note: One variable varies while remaining n-1 variable are fixed at their mid value.

Robust: Total test cases are 6n+1.

In this two additional values one below the boundary (lower side) and one above the boundary (upper side) are also considered apart from five values in BVC method.



for $1 \le x \le 100$

$x = \{0, 1, 2, 55, 99, 100, 101\}$

Worst: total test cases are 5^n.

Input values are same as in BVC method but difference between them is that 1 variable is varies and (n-1) are fixed on all possible values.

Equivalence Class Partitioning:

In this method, input domain of a program is partitioned into a finite number of equivalence classes such that one can reasonably assume, but not be absolutely sure, that the test of a representative value of each class is equivalent to a test of any other value.

Invalid	valid	Invalid
class	class	class

suppose a variable x has a range [1, 100]

i) any number between 1 and 100 is valid input
ii) any number less than 1 is invalid
iii) any number greater than 100 is invalid
iv) If it is not a number, it should not be accepted.

Structural Testing (White Box testing)

A complementary approach to functional testing is called structural/white Box testing. It permits to examine the internal structure of the program.

- statement coverage
- branch coverage
- path coverage

path coverage method includes both statement and branch coverage. It Involves:

i) Generating a set of paths that will cover every brand in the program

ii) finding a set of test cases that will execute every path in this set of program paths.

In this testing, control flow graph (CFG) of the given source code is generated and its cyclomatic complexity is measured. The total independent paths are equals to the cyclomatic complexity of source code.

$\boxed{\text{Independent paths} = \text{Cyclomatic Complexity}}$

Objectives of. Software Testing

Ensure the quality of product

Defect prevention and detection

verify and validate user requirement

Focus on accurate and reliable result

Gain confidence of work

Evaluate the capabilities of a system and its performance

Decision Table :

Decision Table Testing is a black box Testing technique to determine the test scenario for complex business logic.

It is a good way to deal with different combination inputs with their associated output.

It is also called cause-effect table.

Decision table testing is a testing technique used to test system behavior for different input combinations

conditions	Printer does not Print	T	T	T	T	F	F	F	F
	Red light is flashing	T	T	F	F	T	T	F	F
	Printer is unrecognised	T	F	T	F	T	T	T	F
Actions	Check the power cable			X					
	check the printer cable	X		X					
	Ensure printer software is installed	X		X	X	X	X		
	check/replace ink	X	X		X	X	X		
	Check for paper jam		X		X				

Create decision Table for the following program in an office e-mail system:

→ send e-mail when receipient address present, subject present before 5:30

→ if after 5:30, then put in pending folder

→ if address or subject is missing, give warning message.

Condition	Address present	T	T		T
	subject present	T		T	T
	Before 5:30	T	T	T	
Action	Send e-mail	X			
	Error message		X	X	
	Make pending				X

Regression Testing : Regression Testing is a type of software testing that intends to ensure that changes (enhancements or defect fixes) to the software have not adversely affected it.

The purpose of regression testing is to ensure that changes like enhancements, patches and configuration changes have not introduced new faults.
One of the main reason of regression testing is to determine whether a change in one part of the system has any negative affect on other parts of the software.

When to perform Regression Testing :

① Any defect is fixed.
② Any new feature or new functionality is added.
③ Any enhancement is done to a previous functionality.

Mutation Testing - In this few arbitrary changes are made to the program, Each time the program is changed, it is called mutated program and the change effected is called mutant. On a given test case if mutants are identified they are called dead mutant. If mutant are still alive, test data are enhanced to kill the mutants.

Performance Testing :

performance testing is performed to evaluate the performance of components of a particular system under a particular work load. During this testing, system components are monitored to verify the stability of the system under test.

Performance testing techniques –

Load testing – Testing the behavior of the system under a specific load or to get the breakeven point where system starts downgrading its performance.

stress testing – It is performed to find the upper limit capacity of the system and also to determine how the system performs if the current load goes well above the expected maximum.

security testing

Usability testing

Portability testing

Static Testing

static testing technique provides a powerful way to improve the quality and productivity of software development by assisting engineers to recognize and fix their own defects early in the development process. In this software is tested without executing the code.

static testing may be conducted manually or through the use of various software testing tools. It starts early in the development life cycle and so it is done during the verification process.

Types of defects that are easier to find during static testing are:

- deviations from standards
- missing requirements
- design defects
- non-maintainable code
- inconsistent interface specifications

Advantage of static testing

→ Since static testing can start early in the life cycle, early feedback on quality issues can be established.

→ By detecting defects at an early stage, rework costs are most often relatively low.

→ static tests contributes to an increased awareness of quality issues.

~~→ Code Walkthrough~~
~~→ Code Inspection~~
~~→ Clean Room Testing~~

Types: ① Review – Typically used to find and eliminate errors or ambiguities in documents such as requirements, design, test cases.

② static Analysis – The code written by developers are analyzed (usually by tools) for structural ~~differents~~ defects that may lead to defects.

Types of Review

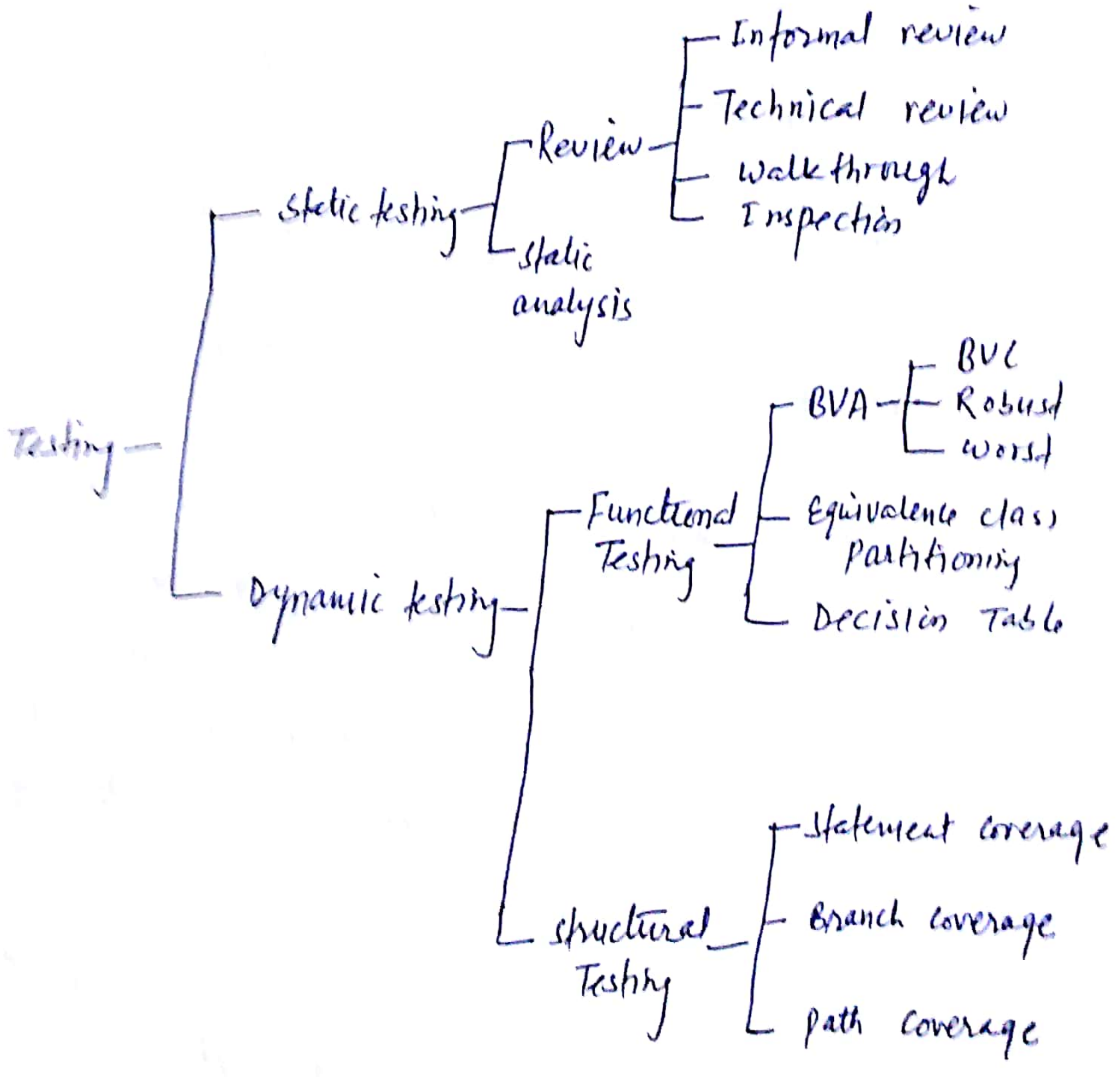
- Informal Review
- Technical Review
- Walk through
- Inspection

Informal Review — This is one of the type of review which does not follow any process to find errors in the document. Under this technique one just review the document and give informal comments on it.

Technical Review — A team consisting of your peers, review the technical specification of the software product and checks whether it is suitable for the project. This review concentrates mainly on the technical document relates to the software such as requirement specification document, design document, test strategy or test cases.

Walk through — The author of the work product explaining the product to his team. participants can ask question if any.

Inspection - The main purpose is to find defects and meeting is led by trained moderator. This review is a formal type of review where it follows strict process to find the defects. Reviewers have checklist to review the ~~p~~ work products. They record the defects and inform the participants to recitify those errors.



coding standards and Guidelines

Coding standards : ~~Adhere~~ Ad here to well defined & standard style of coding.

1. Rules for limiting the use of global data.
2. Contents of the headers preceding codes for different modules.
3. naming conventions for global variables, local variables and constant identifiers
4. Error return conventions and exception handling mechanism.

Coding Guidelines :

1. Do not use a coding style that is too clever or too difficult to understand.
2. Avoid obscure side effects.
3. Do not use a identifier for multiple purpose
4. The code should be well documented
5. Do not use goto statements

Reasons to follow coding standards

→ A good coding standard gives a uniform appearance to the codes written by different engineers.

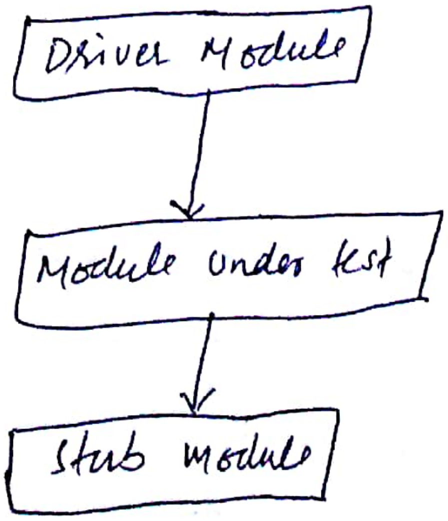
→ It provides sound understanding of the code.

→ It encourages good programming practices

Driver and Stub Modules

Modules required to provide the necessary environment (which either call or are called by the module under test) are usually not available until they, too, have been unit tested.
Stub and driver modules are designed to provide the complete environment for a module.

A stub procedure is a dummy procedure that has the same input-output parameters as the given procedure but has a highly simplified behaviour.
A driver module would contain the non local data structures accessed by the modules under test, and would also have the code to call the different functions of the module with appropriate parameter values.



Unit testing with the help of driver & stub modules