and methods.

Page No. 51
Date

Software testing is to :

① exercise the internal logic and interfaces of every software component and

② exercise the input and output domains of the program to uncover errors in program function, behaviour and performance.

Testability can be defined as "How easily a computer program can be tested?"

Characteristics of Testable software :-

① Operatability :- This is an ability to get easily operate.
"The better it works, the more efficiently it can test."

② observability :- "What you see is what you test."
* Inputs provided as part of testing produce distinct outputs.
* System states and variables are visible during execution.
* Incorrect output is easily identified
* Internal errors are automatically detected & reported.
* Source code is accessible.

③ Controlability :-

"The better we can control the software, the more testing can be automated and optimized."

* All possible outputs can be generated through some combination of input, and I/O formats are consistent and structured.
* All code is executable through some combination of input.
* Software and hardware states and variables can be controlled directly by the test engineer.
* Tests can be conveniently specified, automated and reproduced.

④ Decomposability :-

"By controlling the scope of testing, we can more quickly isolate problems and perform smarter testing."

* The software system is built from independent modules that can be tested independently.

⑤ Simplicity :-

"The less is there to test, the more quickly we can test it."

* The program should exhibit functional simplicity, structural simplicity and code simplicity.

⑥ Stability :-

"The fewer the changes, the fewer the disruptions to testing".

* Changes to the software are infrequent, controlled when they do occur, and do not invalidate existing tests.
* The software recovers well from failures.

⑦ understandability :-

"The more information we have, the smarter we will test".

* The architectural design and the dependencies between internal, external, and shared components are well understood.
* Technical documentation is instantly accessible, well organised, specific and detailed, and accurate. Changes to the design are communicated to testers.

Good test / test characteristics / Attributes of good test :-

① A good test has a high probability of finding an error.
② A good test is not redundant.
③ A good test should be "best of breed".
④ A good test should be neither too simple nor too complex.

Sucessful Test :-

Glen Mayer's objectives of the testing which makes a test sucessful test :-

① Testing is a process of executing a program with the intent of finding an error.

② A good test case is one that has a high probability of finding an as yet undiscovered error.

③ A successful test is one that uncovers an as yet undiscovered error.

Advantages of Testing :-

① Testing uncovers errors, which helps producing outputs according to user requirements.

② Testing ensures that the software conforms to business as well as user's need.

③ Testing ensures that the software is developed according to user requirements.

④ Testing improves the quality of software by removing maximum possible errors from it.

⑤ Testing removes errors that lead to software failure.

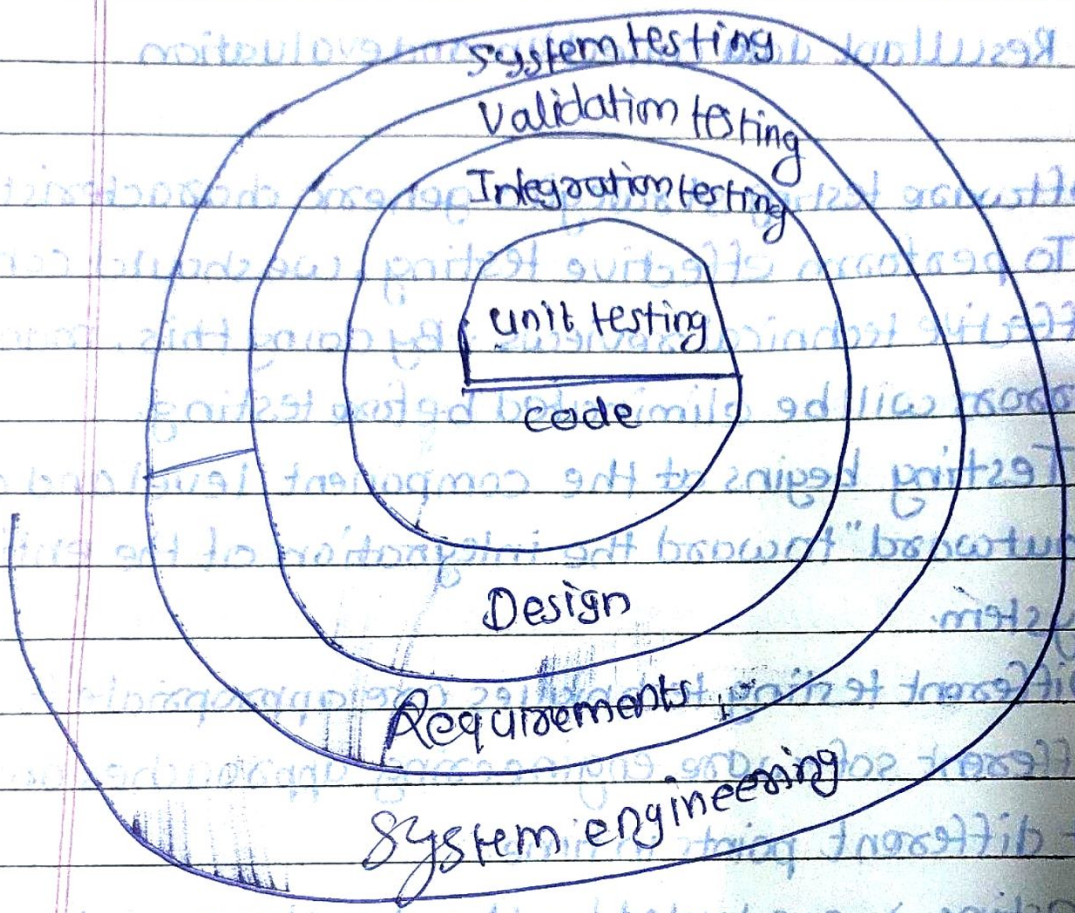
⑥ Testing gives confidence that the software will work as intended.

Software Testing Strategies :-

* A strategy for software testing provides a road map/ template that describe:
  1. Steps to be conducted as part of testing.
  2. When these steps are planned and then completed
  3. How much effort, time and resources will be required.

* Testing Strategies must Incorporate :-
  1. Test planning
  2. Test case design
  3. Test execution
  4. Resultant data collection and evaluation

* Software testing strategies generic characteristics :-
  1. To perform effective testing, we should conduct effective technical reviews. By doing this, many errors will be eliminated before testing.
  2. Testing begins at the component level and works "outward" toward the integration of the entire system.
  3. Different testing techniques are appropriate for different software engineering approaches and at different points in time.
  4. Testing is conducted by the developer of the software and an independent test group.
  5. Testing and debugging are different activities, but debugging must be accomodated in any testing strategy.



The software process may be viewed as the spiral shown in the following figure.

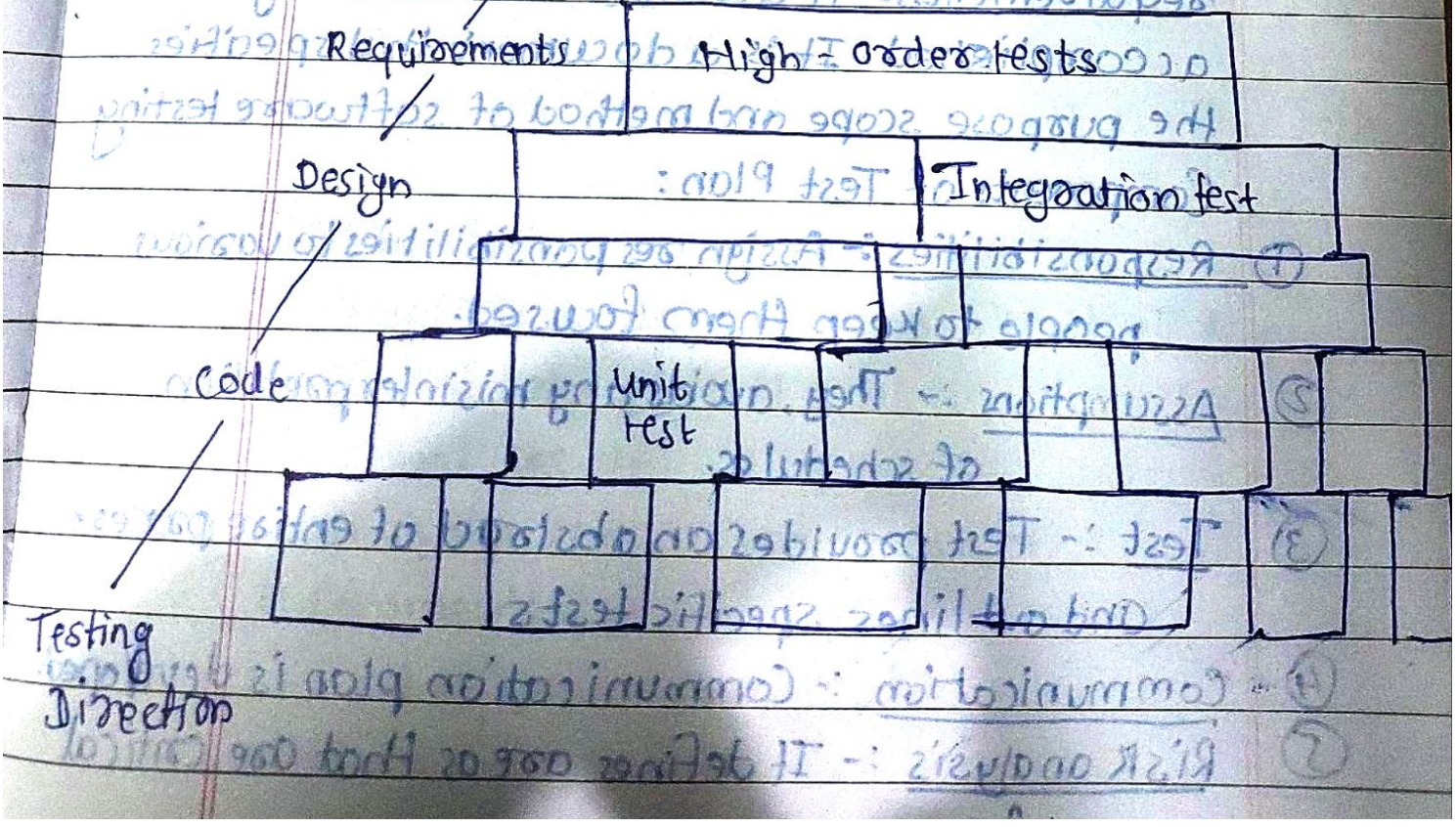
Initially system engineering defines the role of software and software requirements analysis, where the information domain, function, behaviour, performance, constraints and validation criteria are developed.

Moving inward we come to design and finally to coding. To develop computer software, we spiral inward (counter-clockwise). Testing progresses by moving outward to integration testing, where the focus is on design and construction of software architecture.

To test a computer software, we spiral out in clockwise direction.

Software Testing Steps:-

Procedural point of view there are four steps as shown in the following figure:



Initially tests focus on each component individually, ensuring that it functions properly as a unit. Hence, the name unit testing.

Next, components must be assembled or integrated to form the complete software package. Integration testing addresses the issues associated with dual problems of verification and program construction.

After the software has been integrated, a set of higher order tests is conducted. Validation criteria must be evaluated here.

Test plan :-

Test plan is an approach to test a system. A test plan documents the strategy that will be used to verify and ensure that a product or system meets its design specifications and other requirements. It describes how the testing will be accomplished. It is a document that specifies the purpose scope and method of software testing.

Components of Test Plan:

1. Responsibilities :- Assign responsibilities to various people to keep them focused.
2. Assumptions :- They, avoids any misinterpretation of schedules.
3. Test :- Test provides an abstract of entire process and outlines specific tests.
4. Communication :- Communication plan is developed
5. Risk analysis :- It defines areas that are critical for sucess.

⑥ Defect Reporting :- It specifies the way in which a defect should be documented so that it may reoccur and be retested and fixed.

⑦ Environment :- It describes the data, interfaces, work area, and the technical environment used in testing.

Test Case :

A test case is a document, which has a set of test data, pre conditions, expected results and post conditions. It acts as a starting point for the test execution, and after applying a set of input values, the application has a definitive outcome and leaves the system at some endpoint or also known as execution post condition.

Typical Test case parameters :

- Test case ID
- Test scenario
- Test case Description
- Test steps
- Prerequisite
- Test Data
- Expected result
- Test Parameters
- Actual Result
- Environment Information
- Comments

Example:
Let us say that we need to check an input field that can accept maximum of 10 characters. While developing the test cases for the above scenario, the test cases are documented the following way. In the below example, the first case is a pass scenario while second one is fail.

Scenario	Test step	Expected result	Actual outcome
① Verify that input field that can accept max of 10 characters	Login app^n and key in 10 char.	App^n should be able to accept all 10 char.	App^n accepts all 10 chars.
② Verify that input field that can accept max of 11 characters	Login app^n and key in 11 char.	App^n should NOT accept all 11 chars.	App^n accepts all 10 chars.

If the expected result doesn't match with the actual result, then we log a defect.

Test Data:

* Test data are the data which have been specifically identified for use in tests.
* Test data can be used for inputting data to the tests to produce the expected output.

Software Testing Strategies for Conventional Softwares :-

1] Unit Testing :-

Unit Testing focuses on verification of smallest unit of software. By using the component level design description, important control paths are tested to uncover errors within the boundary of the module.

The relative complexity of tests and errors is limited due to limited scope of unit testing. The unit test focuses on the internal processing logic and data structures within the boundaries of the component.

Considerations :-

① Module interface is tested to ensure that ~~the module~~ information properly flows into and out of the module.

② Local data structures are examined to ensure that data stored temporarily maintains its integrity.

③ Independent paths are checked.

④ Boundary conditions are tested to ensure that the module operates properly at boundaries.

⑤ All error handling paths are tested.

Unit test procedure:

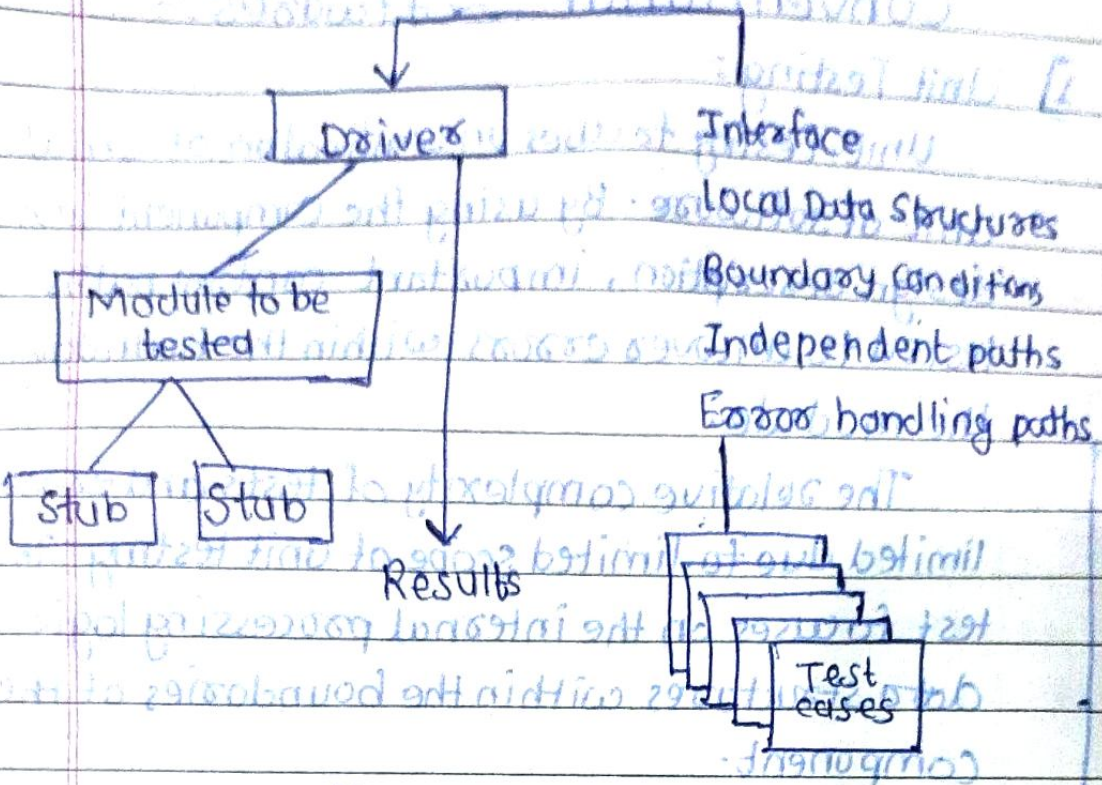


Fig:- Unit test environment

Unit testing is normally adjacent to coding step. The design of unit test can occur before the coding begins or after source code has been generated. Because a component is not a standalone program, driver and/or stub software must often be developed for each unit test.

A driver is nothing but main program that accepts test case data, passes such data to the component to be tested and prints relevant result.

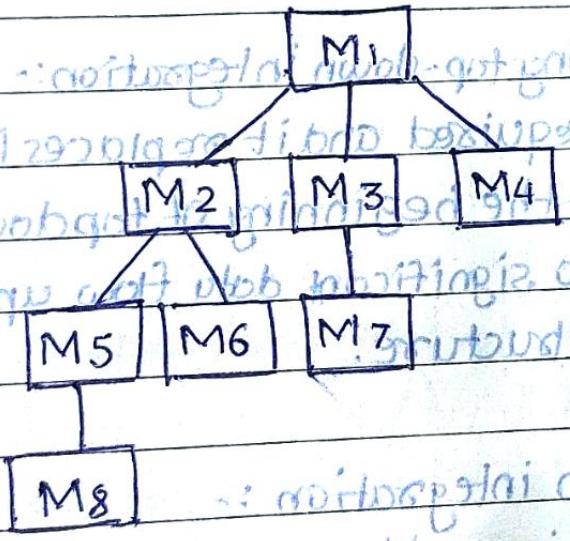
Stubs replace modules that are subordinate to the component to be tested. A stub or "dummy subprogram" do minimal data manipulation, prints verification of entry and returns the control to the module undergoing testing.

2] Integration testing :-

Integration testing is a systematic technique for constructing the software architecture while at the same time conducting errors associated

Top-Down Integration Testing :-

Top-Down Integration testing is an incremental approach to construction of software architecture. Modules are integrated by moving downward through the control hierarchy, beginning with the main control module. It follows depth first or breadth first approach.



Depth first integration integrates all components on a major control path of the program structure. Breadth first integration incorporates all components directly subordinate at each level, moving across the structure horizontally.

Integration testing process is performed in five steps:-

① The main control module is used as a test driver and stubs and substituted for all components directly subordinate to main control module.

② Depending on the integration approach selected, subordinate stubs are replaced one at a time with actual components.

③ Tests are conducted as each component is integrated

④ On completion of each set of tests, another stub is replaced with the real component.

⑤ Regression testing may be conducted to ensure that new errors have not been introduced.

Problems during top-down integration:-

① Stubs are required and it replaces low level modules at the beginning of topdown testing; therefore no significant data flow upward in the program structure.

Bottom-up integration :-

It begins construction and testing with atomic modules. As components are integrated from the bottom up, the functionality provided by components subordinate to a given level is always available and the need for stubs is eliminated.

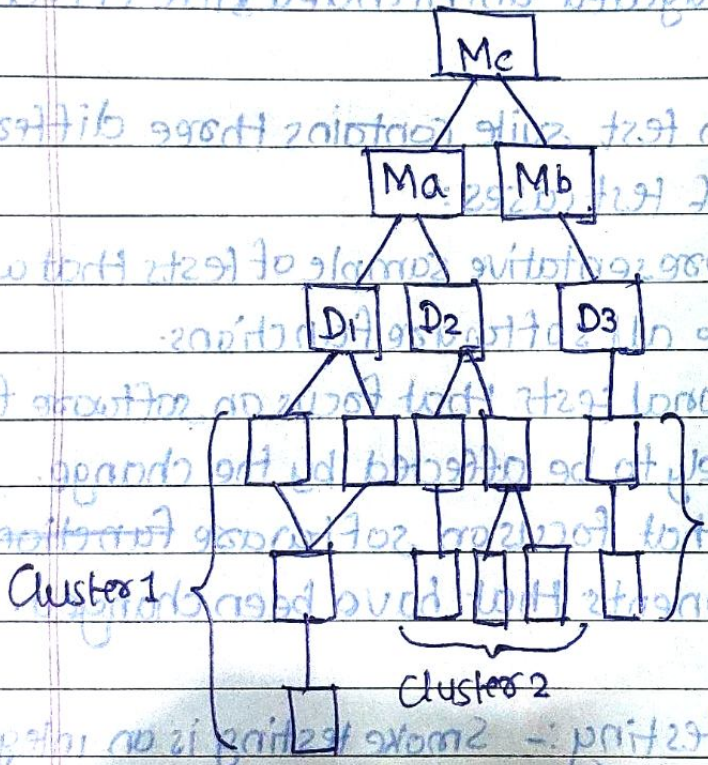
A bottom up integration is implemented with following tests:

① Low level components are combined into clusters that perform specific software sub function.

② A driver is written to coordinate test case input and output.

③ The cluster is tested

④ Drivers are removed and clusters are combined moving upward.



As integration moves upward, the need for seperate test drivers lessens. If the top two levels of program structure are integrated top down, the number of drivers can be reduced and integration of clusters is greatly simplified.

Date

Regression Testing :- Each time a new module is added as a part of integration testing, the software changes. New data flow paths are established, new I/O may occur, and new control logic is invoked.

These changes may cause problems with functions that previously worked correctly. Regression testing is the re-execution of some sub-set of tests that have already been conducted to ensure that changes have not propagated unintended side effects.

Regression test suite contains three different classes of test cases :-

1. A representative sample of tests that will exercise all software functions.
2. Additional tests that focus on software functions that likely to be affected by the change.
3. Tests that focus on software components that have been changed.

Smoke testing :- Smoke testing is an integration testing approach that is commonly used when software is developed. It is designed as a pacing mechanism for time-critical projects, allowing the software to access the projection a frequent basis.

Smoke testing approach contains following activities:

① Software components are integrated into a build. A build includes all data files, libraries, reusable modules, and engineered components that are required to implement one or more product functions.

② Series of tests is designed to expose errors. The intent should be to uncover main errors which may cause delay or faliure in schedule.

③ The build is integrated with other builds, and the entire product is smoke tested daily. The integration approach may be top down or bottom up.

Smoke testing benefits / advantages :-

① Integration risk is minimized

② The quality of end product is improved

③ Error diagnosis and correction are simplified

④ Progress is easier to access.

Verification and Validation :-

Verification refers to the set of tasks that ensure that software correctly implements a specific function.

Validation refers to a different set of tasks that has been built is traceable to customer requirements.

	Verification	Validation
①	It refers to the set of tasks that ensure the software correctly implements a specific function.	It refers to a different set of tasks that the software has been built is traceable to customer requirements
②	Are we building the product right?	Are we building the right product?
③	Ensures that software system meets all the functionality	Ensures that the functionality meet the intended behaviour.
④	Done by developer	Done by tester
⑤	Verification evaluates plans, documents, code, specification, etc.	Validation evaluates product itself.

Alpha and Beta Testing :-

Alpha testing :- The alpha test is conducted at the developer's site by representative group of end users. The software is used in a natural setting with the developer "looking over the shoulder" of the users and recording errors and usage problems.

Beta testing :- The beta test is conducted at one or more end user sites. Unlike alpha testing, the developer generally is not present. Therefore beta test is a "live" application of the software in an environment that cannot be controlled by the developer.

Alpha testing	Beta testing
① It is performed by customer at the developer's site	It is performed by customer at the customer's site
② Alpha testing is not open to market and public	Beta testing is always open to the market and public
③ It is conducted for software application and project	It is usually conducted for software product
④ It is performed in the environment of developer	It is performed in uncontrolled environment
⑤ Developer is present	Developer is not present
⑥ It is not live testing	It is a kind of live testing
⑦ It comes under both black and white box testing	It comes under black box testing.
⑧ It takes place for shorter period of time	It takes place for very long period of time

System testing:- System testing is a series of different tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different purpose, all work to verify that system elements have been properly integrated and perform proper functions.

① Recovery Testing :- Recovery testing is a system test that forces the software to fail in variety of ways and verifies that recovery is properly performed.
If recovery is automatic (performed by the system itself), re-initialization, data recovery and restart are evaluated for correctness. If recovery requires human intervention, the Mean Time to Repair (MTTR) is evaluated to determine whether it is within acceptable limits.

② Security testing :- Security testing verifies the protection mechanisms of the system. System security must be tested for frontal and rear attacks. During security testing the tester plays the role of a person who want to penetrate the system.
If given enough time and resources, good security testing will ultimately penetrate the system.

③ Stress testing :- Stress tests are designed to test programs in abnormal situations. The tester who performs stress testing asks: "How high we can crank this up before it fails?" The tester attempts to break the program in system testing.

④ Performance testing :- Performance testing is designed to test the run-time performance of software within the context of an integrated system. It occurs throughout all steps in the testing process. Performance tests are often coupled with stress testing and usually require both hardware and software instrumentation.

⑤ Deployment testing :- Deployment testing is called configuration testing, it exercises the software in each environment in which it is to operate. Deployment testing examines all installation procedures and specialized installation software that will be used by customer, and all documentation including user manuals.

⊞ White box testing :- If we know the internal workings of a product, then tests can be conducted to ensure that internal operations are performed according to specifications and all internal components were properly checked. It requires an internal view and so it is called as white box testing.

In white box testing :-

① Procedural details are checked

② Logical paths through the software and collaborations between components are tested by exercising specific sets of conditions and/or loops.

③ Important data structures can be tested for validity

④ It tests the program logic exhaustively. Unfortunately, exhaustive testing presents certain logistical problems. A limited number of important logical paths are checked.

White box testing techniques:

① Basic path testing :- In this, test cases are derieved to exercise the basis set are guaranteed to execute every statement in the program at least one time during testing.

② control structure testing :- It tests the various control structure for its correctness.

Advantages :-

① It can cover the larger part of the program code while testing.

② It can start at the earlier stage. No need to wait for GUI.

③ Testing is more through, with the possibility of covering most paths.

④ It helps in optimizing the code

⑤ Exra lines code can be removed which can bring hidden defects.

Date

Disadvantages :-

① As skilled tester is needed to perform white box testing, the cost is high.

② Test script maintainance can be a burden if the implementation changes frequently.

③ High skilled resources are needed.

④ Sometimes, it is impossible to look into every part to find out the hidden error that may create problems.

Black Box Testing :-

Black box testing is also called as behavioural testing; it focuses on the functional requirements of the software. That is, black box testing technique enable us to derieve sets of input conditions that will fully exercise all functional requirements of program.

Black box testing finds errors in following categories:-

① Incorrect or missing functions

② Interface errors

③ Errors in data structures or external databases

④ Behaviour or performance errors

⑤ Initialization and termination errors.

Black box testing methods:

① Graph based
② Equivalence partitioning
③ Boundary value analysis
④ Orthogonal array testing

Advantages:

① Tests are done from user's point of view and will help in exposing discrepancies in the specifications.
② Code access not required.
③ Tester need not know programming languages or how the software has been implemented.
④ Tests can be conducted by third party
⑤ Test cases can be designed as soon as the specifications are complete.
⑥ Well suited and efficient for large code segment

Disadvantages:-

① In this testing only a small number of possible inputs can be tested and many program paths will be left untested.
② Limited coverage
③ Inefficient testing due to limited knowledge
④ Tests can be redudant.
⑤ Difficult to design test cases because of unclear specifications.
⑥ Blind coverage

White box testing	Black box testing
① It tests the internal structure of software	It tests functionality of software
② Knowledge of internal structure of program is required	No knowledge of internal structure of program is required
③ Does not ensure the complete implementation of all the specification motioned in user require-ments	Concerned with testing the specification and does not ensure that all the components of software that are implemented are tested.
④ Checks the basic path & control structure of the program	Checks validity, behaviour and performance of testing.
⑤ Test cases are generated on the basis of internal code or structure	Test cases are not generated on the basis of the internal structure or code
⑥ Done by testers and developers	Done by end users, testers and developers

Debugging - Concept & Need :

Debugging occurs as a consequence of successful testing. That is, when a test case finds an error, debugging is the process that helps in removal of error. External view or error and its internal cause may be entirely different. The debugging process attempts to match symptom with cause, thereby leading to error correction.

The debugging process will have one of two outcomes:
① The cause will be found and corrected
② The cause will not be found

Characteristics of bugs:-
① The symptom and the cause may be geographically remote i.e the symptom may appear in one part of a program, while the cause may actually located at the other part.
② The symptom may disappear when another error is corrected.
③ The symptom may actually be caused by non-errors
④ The symptom may be caused by human error that is not easily traced.
⑤ The symptom may be result of timing problems rather than processing problems.

Debugging Strategies / Tactics:

There are 3 debugging strategies:-
① Brute force
② Back-tracking
③ Cause elimination

① Brute force:

It is most common but least efficient method for isolating the cause of a software error. It is used when all other methods fail.

Using a "let the computer find the error" philosophy memory dumps are taken, run-time traces are invoked, and the program is loaded with output statements. We hope that somewhere in the mass of information we will find a clue that can lead to the cause of an error.

Mass of information produced may lead to sucess; it generally leads to wasted effort and time.

② Backtracking :-

It is common debugging approach that can be used sucessfully in small programs. Beginning at the site where a symptom has been uncovered, the source code is traced backward manually until the cause is found.

Unfortunately, as the number of source lines increases, the number of backward paths may become too large.

③ Cause elimination :-

Data related to error occurance are organised to isolate causes. A cause hypothesis is made and data are used to prove or disprove the hypothesis. List of all possible causes is developed and tests are conducted to eliminate each

Induction strategy:-

It is a disciplined through process, where errors can be debugged by moving outwards from particulars to the whole

Deduction:-

In this, first step is to identify all possible causes and then using the data, each cause is analysed and eliminated if it is found invalid