> Date ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri ☐ Sat ☐ Sun

SOFTWARE TESTING:

→ **What is software testing?**

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

→ **Why should we test?** Testing is expensive, but releasing S/w without testing may lead to cost much higher than that of testing. The earlier the errors are discovered & removed, lower is the cost of their removal.

→ **Who should do testing?**

It is difficult for developers to point out errors from their own creations. So there should be a separate group of people for testing the software.

→ **What should we test?**

Program's response to every possible input.
Complete testing is neither possible, nor feasible.

Error/Mistake: Syntax error / misunderstanding of specifications.
Bugs: When developer make mistakes while coding.
Fault: representation of error, where representation is mode of expression such as DFD, ER diagrams, narrative text etc.
Failure: failure occurs when a fault executes.

Test case / Test: It describes an input description & an expected output description.

Date ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri ☐ Sat ☐ Sun

Good test case : Test cases with good capability

Test suite : Set of good test cases.

Testing = Verification + Validation

Verification : evaluating a system/component to determine whether the products of a given development phase satisfy the conditions imposed at the start of the phase.

Validation : evaluating a system/component at the end of development process to determine whether it satisfies the specified requirements.

Acceptance testing :

- It is done when a s/w is developed for a specific customer.
- The tests are conducted by end user/customer.
- It test if s/w aligns with user needs & bussiness req. & assess whether it is acceptable for delivery or not.

Alpha testing	Beta testing
- Conducted by some potential customers at developer's site.	- conducted by customers/end users at their site.
- Tests are conducted in a controlled environment.	- Tests are conducted in real env. that cannot be controlled by developer.
- started when formal testing procedure is near completion.	- A pre-release version is made available for testing to a chosen set of external users.

> Date ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri ☐ Sat ☐ Sun

Black box testing

- Functional testing
- It can be initiated based on req. specification document.
- Knowledge of programming not required.
- less time consuming.
- behaviour testing of s/w.
- done by software testers.

~~Types do~~

Types: Boundary value analysis
Robustness testing
Worst-case analysis

White-box testing

- Structural testing
- It is started after a detail design document.
- Required
- more time consuming.
- logic testing of software.
- done by s/w developers.

Types: Path testing (CFG)
Mutation testing

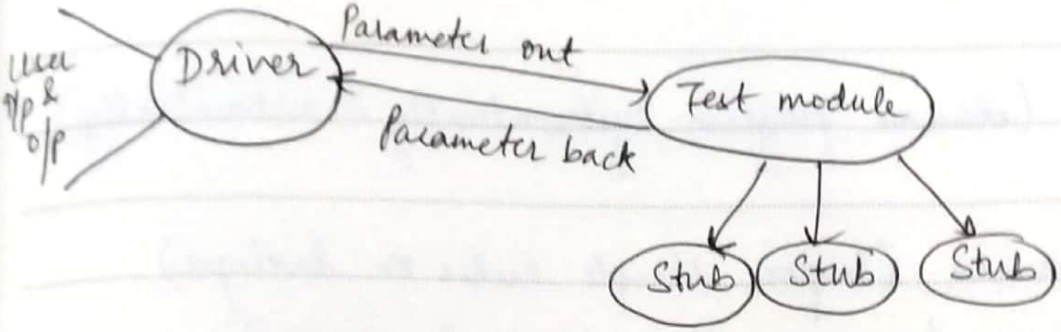
Levels of Testing:

① Unit testing: Involves analysing each unit or individual component of s/w. Reasons – locating error is easier in a single module.
– we can attempt to test it in ~~detail~~ exhaustive fashion

Scaffolding: For a testing a module in isolation, we construct an appropriate driver ~~unit~~ routine & simple stubs ~~to be~~ called by it, & insert output statements in it.

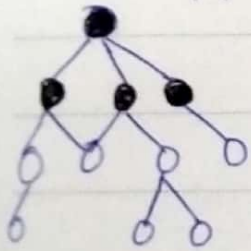
Stubs serve to replace modules that are ~~subordinate~~ subordinate to the module to be tested. It is a dummy that may do minimal data manipulation, print verification & returns. This overhead code is called scaffolding.

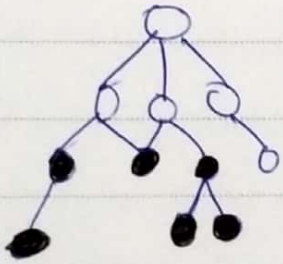
> Date ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri ☐ Sat ☐ Sun

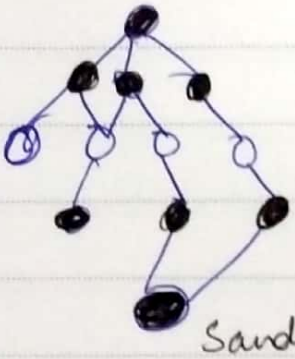


Scaffolding

② Integration testing: performed to determine that the interface b/w modules is correct (whether parameters match on both sides, permissible ranges, meaning & utilization).
It test data flow from one module to other module
Three approaches:


Top down Integration


Bottom up Integration


Sandwich Integration

③ System testing: performed to demonstrate performance - to evaluate a product in terms of our expectations.
Application as a whole system is tested.
Attributes that represent operational correctness of the product are tested - eg. usability, security, compatibility, dependability etc.

> Date ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri ☐ Sat ☐ Sun

Types of Testing tools :

1. Static analyzers (examine program systematically & automatically)
2. Code inspectors
3. Standard enforcers (impose simple rules on developer)
4. Coverage analyzers (measure the extent of coverage)
5. Output comparators (determine whether output is appropriate or not)
6. Test file generators

⇒

→ Black box testing

1) Boundary value analysis
   n input variables ⇒ (4n+1) test cases
   (min, min+1, max-1, max) + (mid)^all

2) Robustness testing
   n input variables ⇒ (6n+1) test cases
   (min-1, min, min+1, max-1, max, max+1) + (mid)^all

'Single fault' assumption

3) Worst case analysis
It reject 'single fault' assumption theory.
   n input variables → 5^n test cases.
   (min, min+1, mid, max-1, max)

Date ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri ☐ Sat ☐ Sun

→ White box testing

1) Path testing : It involves generating a set of paths that will cover every branch in the program & finding a set of test cases that will execute every path in the set of program paths.

CFG : McCabe's cyclomatic complexity :

$V(G) = e - n + 2P$, where e = no. of edges, n = no. of nodes
P = no. of exit points

$= \pi + 1$, where π = predicate nodes / conditional nodes

= no. of regions in CFG.

2) Mutation Testing : It is a fault simulation technique, similar to fault seeding. Multiple copies of a program are made, & each copy is altered ; the altered copy is called a mutant. Mutants are executed with test data to determine whether test data is capable of detecting the change b/w original & mutated program.

Killed mutant : A mutant that is detected by a test case.

Live mutant : ——— not detected ———

Equivalent mutant : They have diff syntax but same meaning as original source code.

Score of a test suite = $\frac{\text{killed mutant}}{\text{total - equivalent}} \times 100\%$

> Date ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri ☐ Sat ☐ Sun

Type of mutation testing:

i) Single order mutation testing: one line change in code

ii) Second order / higher order mutation testing

STLC (Software Testing Life Cycle)

It is a systematic approach to testing a S/W to ensure that it meets the requirements & is free of defects.

Phases:

1) Requirement Analysis: Review SRS, interview skateholders, identify ambiguities, inconsistencies and any potential risk or issues that may impact the testing process.

2) Test planning: Identify testing objectives, develop strategy, identify test env, resources & test cases, assign role & responsibili and review & approve test plan.

3) Test case development: ~~Identify~~ Create test scenario, write clear, concise, understandable test cases, review & validate test cases.

4) Test Environment setup: decides the conditions on which S/W is tested, either developer or customer creates testing environment.

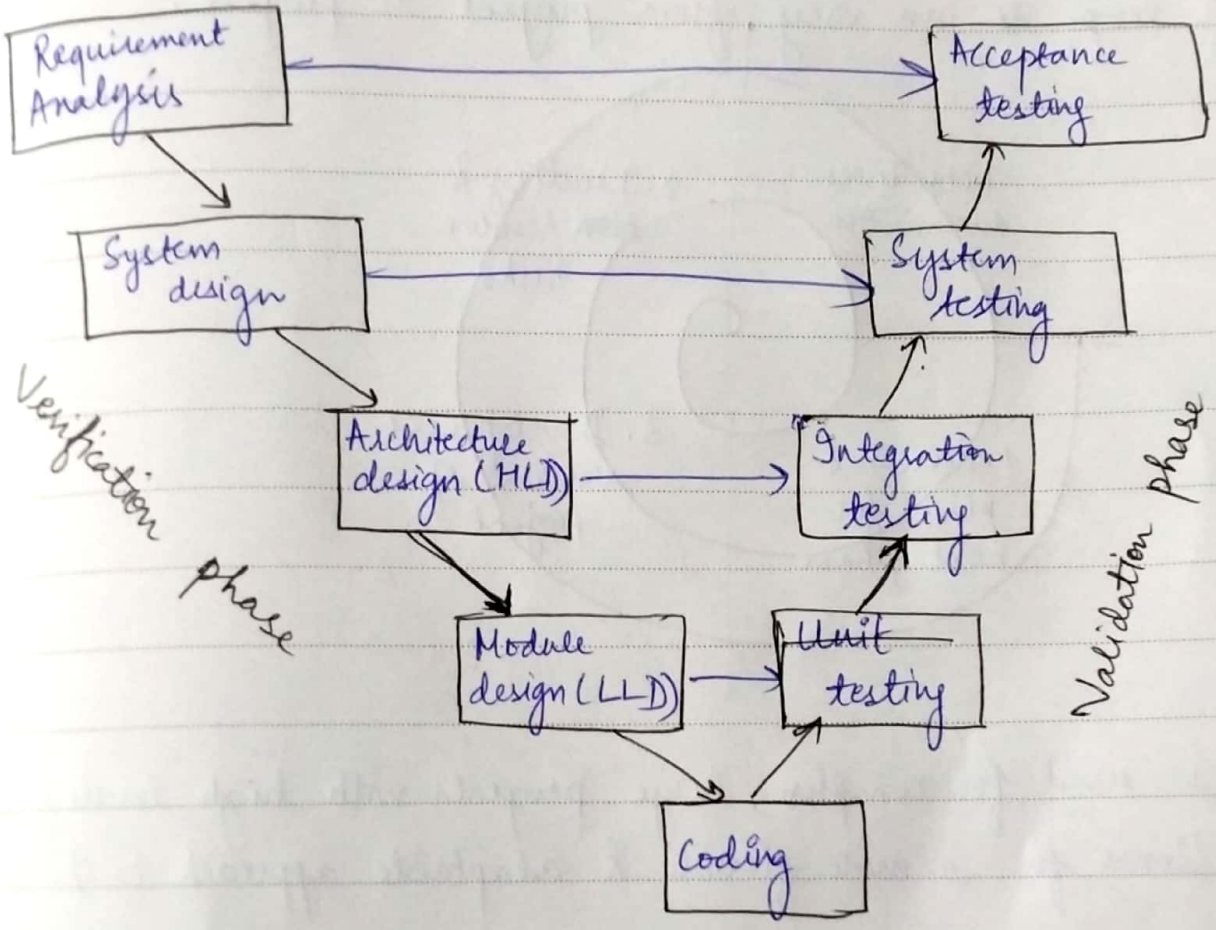
5) Test Execution: Testing team starts executing the prepared test cases - Result analysis, defect logging also takes place.

6) Test Closure: All testing related activities are completed & documented. Test summary report, test env. clean up, feedback & improvements.

Date ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri ☐ Sat ☐ Sun

V Model (Verification & Validation Model)

It is based on association of a testing phase for each corresponding development phase, i.e for each development activity, there is a testing activity corresponding to it.

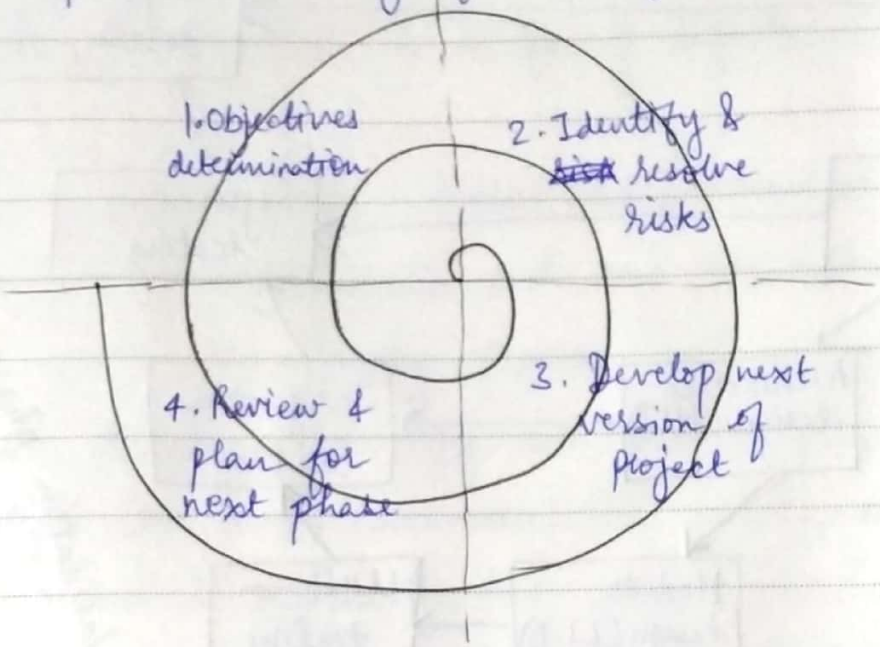


Importance: early defect detection
improved quality assurance
improved cooperation & traceability
determining the phases of development & testing

> Date ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri ☐ Sat ☐ Sun

SPIRAL MODEL

- It is a combination of waterfall model & iterative model.
- It provides support for risk handling.
- It looks like a spiral in diagramatic representation. Each loop of spiral is called a phase of s/w development process. The exact no. of loops can vary from project to project.



- It is used for complex, large projects with high levels of risk as it allows for a more flexible & adaptable approach to s/w development

Phases: 1) Objectives determination: Gather req. from customers, identify, elaborate & analyse objectives at start of every phase.

2) Identify & resolve risks: All sol" are evaluated to select the best possible sol". Then risk associated with that sol" are identified & resolved using best possible strategy. Prototype for best sol" is built.

3) Develop the next version of product: The identified features are developed & verified through testing. At the end of this phase, next version of s/w is available

Date ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri ☐ Sat ☐ Sun

4. Review & plan for next phase: Customers evaluate the so-far developed version of s/w & planning for next phase is started.

RISK MANAGEMENT

- A risk is a probable problem. It may or may not happen, but if it happens, undesirable losses will occur.
- Risk management is a systematic process of recognising, evaluating & handling risk. It is important because it helps org. to prepare for unexpected circumstances.
- Risk management process consists of a sequence of steps:

i) Risk identification: process of recognizing and evaluating potential threats that could negatively impact org, its operation or its workforce.

ii) Risk analysis / assessment: evaluating & understanding the potential impact of identified risks on an org. It prioritize each risk based on their consequences to manage them efficiently.

iii) Risk planning: develop strategies & actions to manage & mitigate identified risks effectively - involves planning & mitigati of risk.

iv) Risk monitoring: involves continuously tracking & overseeing identified risks to assess their status, changes & effectiveness of mitigation strategies, so as to adapt to new development & challenge