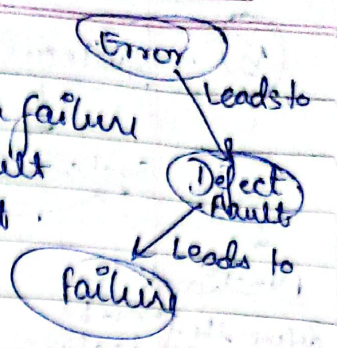
Software Testing

(Unit 1)

Date
Page

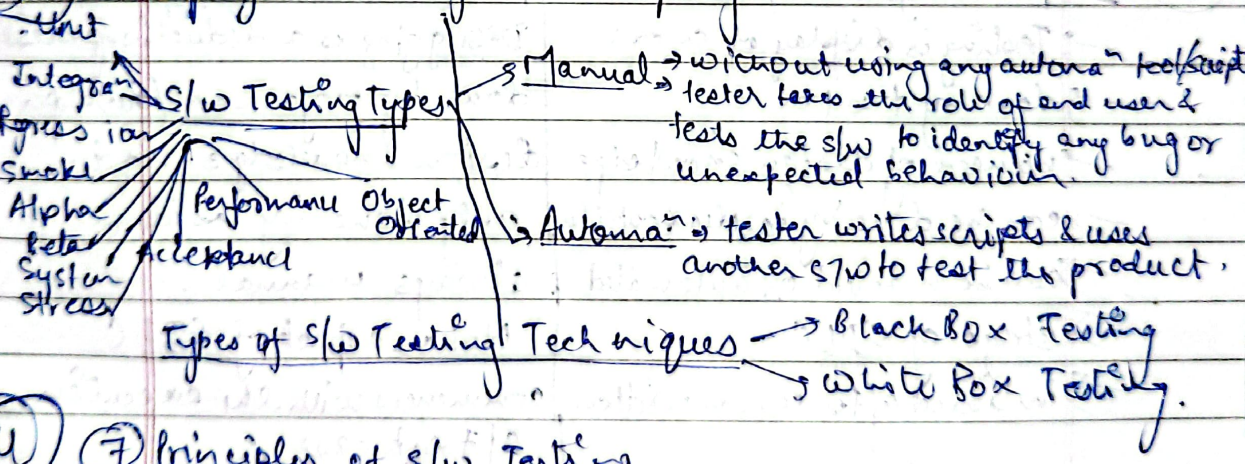
(1) Defect/Bug/Mistake

Fault :- Discrepancy in code that causes a failure
Error :- Human mistake that caused fault
failure :- External behaviour is not correct.



(2) Software Testing :- process of verifying & validating whether a s/w is ~~bug~~ :-
→ bug free
→ meets the technical requirements as guided by its design & dev.
→ meets the user requirements effectively & efficiently by handling all the exceptional & boundary cases.

Testing Objectives :-
→ It aims not only at finding faults in the existing s/w, but also at finding measures to improve the s/w in terms of efficiency, accuracy & usability.
→ It mainly aims at measuring the specification, functionality & performance of a s/w prog.

(5)


(4) (7) Principles of s/w Testing

① Testing shows the presence of defects → goal is to make s/w fail
② Exhaustive testing is not possible → s/w can never test at every test case
③ Early testing → defect detected in early phase of SDLC is very less expensive
④ Defect Clustering → a small no. of modules contain most defects
⑤ Pesticide Paradox → repeating same test cases won't find new bugs.
⑥ Testing is context dependent → Dif types of s/w require diff. types of testing
⑦ Absence of errors fallacy → 99% bug free s/w doesn't mean it's useful to user, it should follow user req.

Sangram®

Date
Page

(5) S/w Requirements → Business Req :- general overview of project, its primary use, why it is needed, its scope & vision, intended audience etc.

→ System Req

→ User Req :- these requirements are gathered using use cases, user scenarios & user stories

→ Functional — define the f^ns to be performed & features to be possessed by the s/w

→ Non-Functional — used to evaluate & assess s/w behaviour under unexpected condⁿs & environments.

(6) Behaviour Testing → testing of external behaviour of prog; also w/a black box testing. It is usually a functional testing.

Correctness → can be defined as the adherence to the specifications that determine how users can interact with s/w & how s/w should behave when it is used correctly.

(7)

Testing	Debugging
process to find bugs & errors	process to correct bugs found during testing
Testing is display of errors	Debugging is a deductive process.
done by tester	done by programmer/developer.
no need of design knowledge	design knowledge reqd.
can be done by insiders & outsiders	only done by insiders
can be manual or automated	always manual.
stage of SDLC	Not an aspect of SDLC
initiated after code is written	commences with the execution of test case

(8) S/w Measurement :- deals with measuring the size, #qty, amount or dimension of a particular attribute of a product or process. eg. No. of errors.

S/w Metrics :- provides functions by which we can measure the attr of s/w system, product or process eg. No. of errors per hour.

Date
Page

Need of s/w measurement & metrics

→ to ensure that the s/w is bug free before release.
→ to anticipate future qualities of process/product,
→ to enhance the quality of a product/process.

Metrics → **Product M** ⇒ describes chars. of product ⇒ measures size, complexity, design features, performance, reliability, functionality, Quality level etc.

Project M. ⇒ describes proj. chars. & execn ⇒ measures no. of s/w developers, cost, schedule, productivity, quality, assess status of ongoing project.

Process M ⇒ used to improve the development process & maintenance activities of s/w product, no. of defects, tools & tech, quality & efficiency.

(9)

Verification	Validation
→ Verify the requirements which we have & verify whether we r developing the product accordingly or not.	→ to validate the product which we have developed is right or not.
→ Are r building a product right?	→ Are r building a right product
→ Low Level Activity	→ High Level Activity
→ Low Cost	→ High cost
→ no code execn	→ involves code execn

(10) **S/w Quality** :→ quality s/w. is reasonably bug free, delivered on time & within budget, meets requirements & is maintainable.

⇒ both quality of design & quality assurance needs to be ensured across all phases of SDLC.
⇒ Quality is everything until pdt is to operate (0-hrs)
→ defects are mistakes that escaped final test.

S/w Reliability :- the probability of failure free s/w operation for a specified period of time in a specified environment.

⇒ based on fault, error & failure.

Sangam® ⇒ Reliability is everything happening after 0-hrs.

⇒ defects that appear over time

Date
Page

(11) **Software Defect Tracking** : process of tracking the logged defects in a product (from beginning to closure (by inspection, testing or recording feedback), & making new versions of the product that fixes the defect.

→ The no. of defects get multiplied over a period of time, & to effectively manage them, defect tracking system is used to make the job easier.

→ It is a very difficult task to manage, evaluate & prioritize the defects.

Eg. Backlog, Bugyard, Bugzilla, ZohoBug Tracker etc.

Unit 2

(1) **Static Testing** :- type of S/W Testing method which is performed to check defects without actually executing the code of S/w appn.

→ In Dynamic Testing, the code is executed to detect defects.

→ performed in early stage of development.

→ errors that can't be found using Dynamic, can easily be found using Static Testing.

2 types

- **Review** → Informal → by audience
  - Walkthrough → by experienced/expert
  - Peer review → by one-another in a team.
  - Inspecn → by higher authorities.
- **Static Analysis** :- evaluation of code quality written by developers.
  - Data Flow :- related to streaming process
  - Control Flow :- how the statements are executed
  - Cyclomatic Complexity :- measurement of complexity of prog.

Static Analysis Tools :- CodeScene, Parasoft, codeSonar, CodeCompare, Sourcemeter, Watchtower, WatchTower, Rosechecker, QC Lint, cloc, etc.

Date
Page

(2) Structural Testing :- An approach where the tests are derived from the knowledge of the s/w's str. or internal implementation.
→ Also kn as glass box, white box, clear box, open box, logic driven, path driven testing.
→ Techniques → Statement Coverage → exercising all programming statement with minimal tests.
↓ Path Coverage → testing all possible paths which means that each statement & branch are tested.
↓ Branch coverage → ensure that all branches are tested atleast once.

Adv	Disadv
→ reason carefully about implementation	→ expensive (time & money)
→ reveals errors in hidden code	→ in depth knowledge of prog. lang. is necessary
→ spots dead code.	

(3) Unit Testing :- individual components (func^n or procedure) of a s/w are tested by developer himself.
Performed during development of appn.
→ Types → Manual
→ Automated
→ Techniques → Black box, white box, gray box testing.

(4) Functional Testing :- to verify the functionality of the s/w appn whether f^n is working acc^g to the requirement specification.
Types :- Regression, Unit, Smoke, Integration, Acceptance, Database, Retesting etc.

(5) Code Coverage Testing :- Degree of which (how much) the source code of the s/w has been tested.
→ white box testing.

Sangam®

Code Coverage Criteria :-

(i) Statement Coverage :- $\frac{\text{No. of statements executed}}{\text{Total no. of statements}} \times 100$

(ii) Decision Coverage :- $\frac{\text{No. of decision outcomes exercised}}{\text{Total no. of decision outcomes in source code.}} \times 100$

(iii) Function Coverage :- $\frac{\text{No. of f}^{n}\text{s called}}{\text{Total no. of f}^{n}\text{s}} \times 100$

(iv) Condⁿ Coverage :- $\frac{\text{No. of executed operands}}{\text{Total no. of operands}} \times 100$

(6) Requirement based Testing : Test cases, condⁿs & data are derived from requirements.

Stages :- Defining Test Completion Criteria
↳ Design Test Cases
↳ Execute Tests
↳ Verify Test Results
↳ Verify Test Coverage
↳ Track & Manage Defects.

all final & nonfinal tests should be passed.

(7) Boundary Value Analysis :- black box testing technique
→ closely associated with equivalence class partitioning.
→ we analyze the behaviour of the appⁿ with test data residing at the boundary values of the equivalence classes.
Eg. An appⁿ that accepts a numeric no. as i/p b/w 10 to 100.

Equivalence Class	Test Data using BVA
Nos. b/w 10 to 100	10, 100
Nos. less than 10	9
Nos. greater than 10	101

Class

(8) Equivalence Partitioning :- ECP → black-box testing, divides i/p domain into classes of data, with the help of these classes, test cases can be derived.

~~Eg. Same as Eg.~~

Percentage [ ] *Accepts % b/w 50 & 90.

Equivalence partitioning

Invalid	Valid	Invalid
<=50	50-90	>90

(9) Model Based Testing :- Test cases are derived from a model that describes the final aspects of the system under test.

⇒ It makes use of a model to generate tests that include both offline & online testing.

Adv:
- Higher level of automation achieved
- Exhaustive testing possible
- Changes to model easily listed

Disadv
- Requires formal specification of model.
- Test cases tightly coupled to model

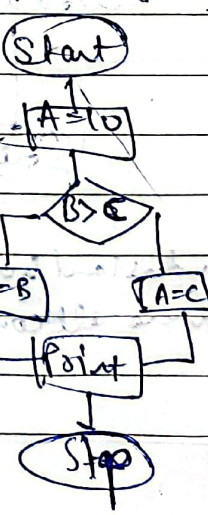
Model Checking :- checking whether a final-state model of a system meets a given specification (also k/a correctness) using model based testing

(10) Code Complexity Testing :- s/w metrics used to indicate complexity of prog. computed using control flow graph.

$M = E - N + 2P$

E = no. of edges
N = no. of nodes
P = connected component

$M = 7 - 7 + 2$
$= 2$



If no control statement, then it is 1,
If it contain 'if' condi, then it is 2
&

Adv → easy to apply quality metrics

Disadv → hard to understand
→ measure prog. control, not data complexity.

Date
Page

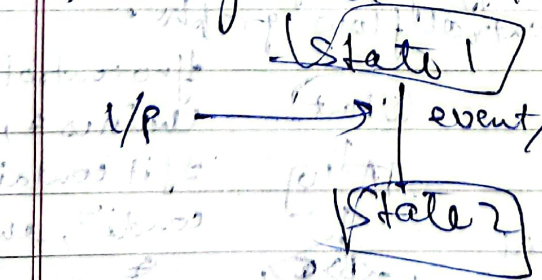
(11)

Black Box Testing	White Box Testing
→ internal str./code of is hidden from tester.	→ tester has knowledge about internal s/w/code of the s/w.
→ Implementation of code is not needed.	→ Code implementation is necessary
→ done by s/w testers	→ done by s/w developers.
→ outer/external s/w testing	→ inner/internal s/w testing.
→ functional test of s/w	→ Structural test of s/w
→ No programming knowledge reqd.	→ reqd.
→ behaviour testing	→ logical testing.
→ applicable to higher level of testing	→ lower level of testing
→ Also called closed testing	→ clear box testing.
→ least time consuming	→ Most time consuming.
→ Eg. Search strong goggle by using keywords	→ By i/p to check & verify loops etc.
→ Types: Functional, Non-functional, Regression	→ Path, Loop, Condition Testing
→ less exhaustive	→ more exhaustive.

(12)

State trans° in testing :- performed to check the change in state of app° under varying i/p. The condi° of i/p passed is changed & change in state is observed.

⇒ kind of black box testing.



Adv. ⇒ understand behaviour
⇒ covers all condi°

Disadv ⇒ can't be performed anywhere
⇒ not reliable.

Sangam®