

# Software Testing (Sem V-22518)



Presentation Made By  
Shafaque Julaha(Sr.lecturer. Computer)  
M. H Saboo Siddik Polytechnic

# Syllabus Structure of Software testing

Teaching Scheme			Credit (L+T+P)	Examination Scheme														
L	T	P		Theory								Practical						
				Paper Hrs.	ESE		PA		Total		ESE		PA		Total			
					Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min		
3	-	2	5	3	70	28	30*	00	100	40	25@	10	25	10	50	20		

# Software testing

Unit No.	Unit Title	Teaching Hours	Distribution of Theory Marks			
			R Level	U Level	A Level	Total Marks
I	Basics of Software Testing and Testing Method	10	04	04	06	14
II	Types and Levels of Testing	12	04	06	08	18
III	Test Management	10	04	04	06	14
IV	Defect Management	08	04	02	06	12
V	Testing Tools and Measurements	08	02	04	06	12
Total		48	18	20	32	70

# Unit I

Basic of Software Testing  
And  
Testing Method

# s/w quality

- The degree to which a system meets



- Consistently meeting user requirements in terms of
  - s/w cost
  - on time deployment
- Example :

# S/W Testing

- Execution of work product or s/w with intent to find defect.
- Defect : failure to address end user requirements. **Example** : Calc s/w give wrong output of Addition.
- Is a process of validating and verifying.
- s/w testing expose hidden defects .

# Objectives of testing

- Finding errors
- Quality improvement
  - Finding errors in a minimum amount of time and effort.
- Satisfying Customer requirements
  - Ensure s/w meets end user requirements.

# s/w testing Case studies

- Case studies to know why testing is required.
  1. Disney's lion king
  2. Y2K bug



# Goal of s/w testing

- Find bug
- Find bug as early as possible
- Make sure that bug is get fixed.
- **Classification:**
  1. immediate goals
  2. Long term goals
  3. Post implementation goal

# Goal of s/w testing

- 1. immediate goals

- a) Bug discovery.
- b) Bug prevention: consequent action of bug discovery.

- 2. Long term goals :affect the s/w quality

- a) Customer Satisfaction
- b) Reliability
- c) Risk management





What is Risk  
mgmt?

# Goal of s/w testing

- 3. Post Implementation Goals:
  - a) Reduce maintenance cost
  - b) Improved s/w testing process.

# Role of tester

- **Test lead / manager**
  - Lead testing team
  - Implementing effective testing process
- **Test designer**
  - Design test cases. 
  - Execute test cases.
  - Record the result
  - Document the identified defect
  - Uses testing techniques

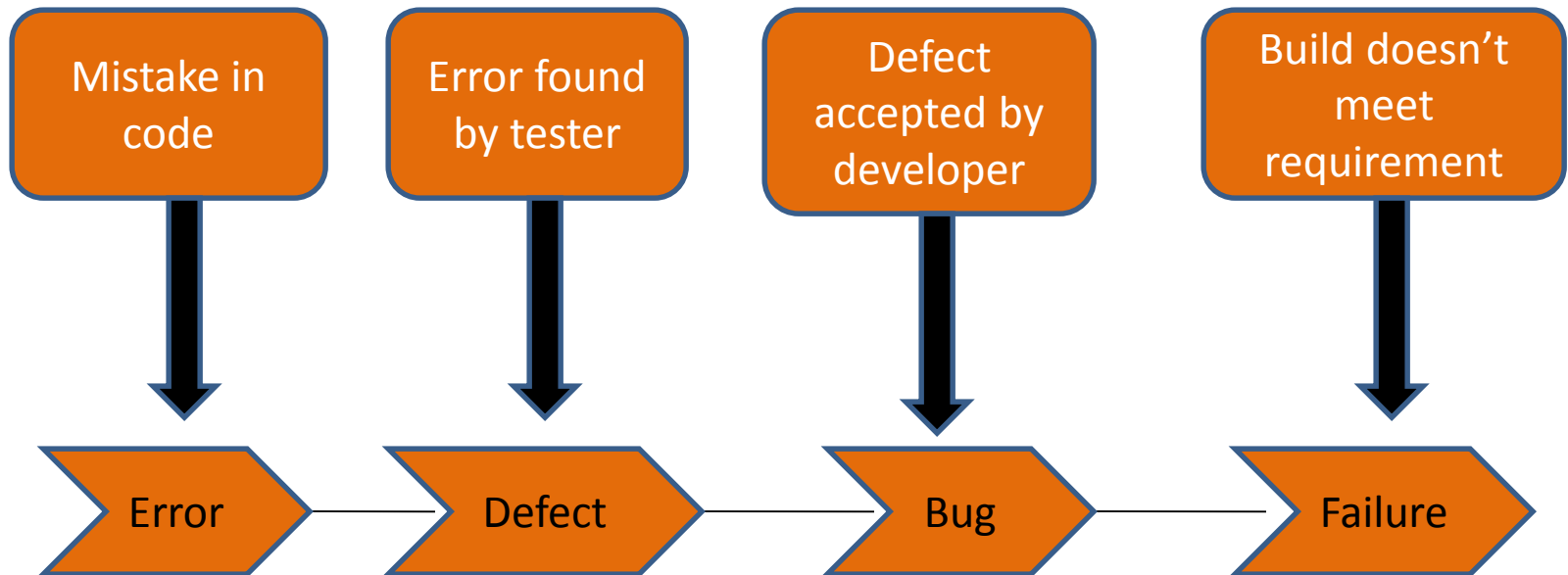


What is  
test  
case?

# Role of tester

- **Test architect :**
  - Support testing process , leverages the available testing infrastructure
  - Knowledge of short term goal , long term goal and resource availability.
- **Test automation engr:**
  - Create automated test case scripts
  - Work in corporation of test designer
  - Automated test scripts created based on test cases designed by designer.
- **Test methodologist:**
  - Provide test organization with resources

# Testing terminology



# Testing terminology

**1 . Error :** mistakes made by programmer.

- Reasons....
  - Confusion in understanding the functionality of s/w
  - Miscalculation of values
  - Missing operators

**2. Defects :** several troubles with s/w product  
error in coding or logic

- Reasons.....
  - Interface defects
  - Documents defects
  - Initialization defects
  - Redundant code

**3. Failure :** s/w unable to perform function according to the specification.

- Reasons ....
  - Problem in h/w
  - Human errors
  - Someone deliberately trying to cause failure

- **4. Bug :** logical mistake
- Bug occur when one or more of the following rules is true.
- **Rule #1:** s/w doesn't do something that product specification say it should do
- **Example:** specification of Calc s/w are addition , subtraction , multiplication.
- And if u get wrong output of addition , that is **Bug** bcoz of Rule #1



- Rule #2:s/w does ....but product spec says it should not do.
- **Example** : Specification state that Calc never lock or freeze . But if Calc stop responding than that's a **Bug** bcoz of Rule#2
- Rule #3: s/w does....but product spec doesn't mention.
- **Example** : if Calc also perform square root of a number than that's a **Bug** bcoz of Rule #3

- Rule #4: s/w doesn't do ....but product spec doesn't mention but it should do.
- Example : if battery get weak than correct calculation didn't happen with weak batteries.  
so there is a need of displaying alert msg .  
it was not specified but it should be.
- Rule #5: s/w is difficult to understand .
- Example: Calc buttons were too small or placement of buttons made it hard to use.  
all of these are **Bug** bcoz of Rule #5.

# Skills of tester

- They r explorer: like to get a new piece of s/w
- They r trouble shooter: can find defects
- They r relentless: keep trying
- They r creative
- They r perfectionists
- They exercise good judgment
- They r persuasive: good in making point of view clear

# Test Cases

- Test cases is a well documented procedure designed to test functionality of s/w
- Test cases are the specific inputs that you will try and the procedure that you will follow to test s/w.
- It consist of :
  - Input values
  - Execution precondition
  - Expected results
  - Actual results

# Test case template

Project Name:

## Test Case Template

Test Case ID: Fun\_10

Test Designed by: <Name>

Test Priority (Low/Medium/High): Med

Test Designed date: <Date>

Module Name: Google login screen

Test Executed by: <Name>

Test Title: Verdy login with valid username and password

Test Execution date: <Date>

Description: Test the Google login page

Pre-conditions: User has valid username and password

Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Navigate to login page	User: <u>sample@gmail.com</u>	User should be able to login	User is navigated to	Pass	
2	Provide valid username	Password: <u>1234</u>		dashboard with successful		
3	Provide valid password			login		
4	Click on Login button					

Post-conditions:

User is validated with database and successfully login to account. The account session details are logged in database.

- **Test case ID:** unique for each test case
- **Test priority:** based on priority test cases get execute
- **Module name :** name of main module or sub module
- **Test designed by :** name of tester
- **Test executed by :** name of tester who execute test case
- **Test title:** test case title
- **Pre condition :** list all pre requisite in order to successfully execute test case. Example:
- **Test steps:** write all execution steps in sequence for test case.
- **Test data / test input :** state what input's are given to test case.
- **Expected result:** describe expected output
- **Actual result :** mention actual result after test case execution
- **Status (pass/ fail):** if actual result is not same as expected result than mark status of test case as Fail otherwise set pass status

# Test case approach

- **Test to pass approach:**
  - Don't push capabilities
  - Straight forward test cases
  - Always run test to pass approach first
- **Test to Fail approach:**
  - Push capabilities
  - Break the s/w

# When to start testing

- **Entry criteria** : minimum set of conditions that should be met in order to start testing.
- All documents , design , reqmts available
- s/w tools available
- All personnel must be trained
- All possible test cases are prepared.



- **Exit criteria:** minimum set of conditions in order to close a particular project phase.
- Completion of all test case execution
- Completion of code coverage
- No higher priority or severe bugs left
- Testing Dead line occur
- Management decision
- Over budget ...

# QA / QC / testing

- QA :set of activities for ensuring quality in the process by which product are developed .
- QC : set of activities to ensure Quality in a developed work
- Product.
- Testing : intention of finding errors



# Quality assurance

- Ensure quality in the process
- Focus on process used to make product
- It is proactive quality process
- Goal : improve development process
- Establish a good quality mgmt system
- Verification is an example of QA
- QA is managerial tool

# Quality Control

- Ensure quality in product.
- Focus on finding defects in actual product
- Reactive quality process
- Goal : Identify defects after product is developed but before released.
- Finding and eliminating sources of quality problems through tools
- Validation is an example of QC
- QC is corrective tool

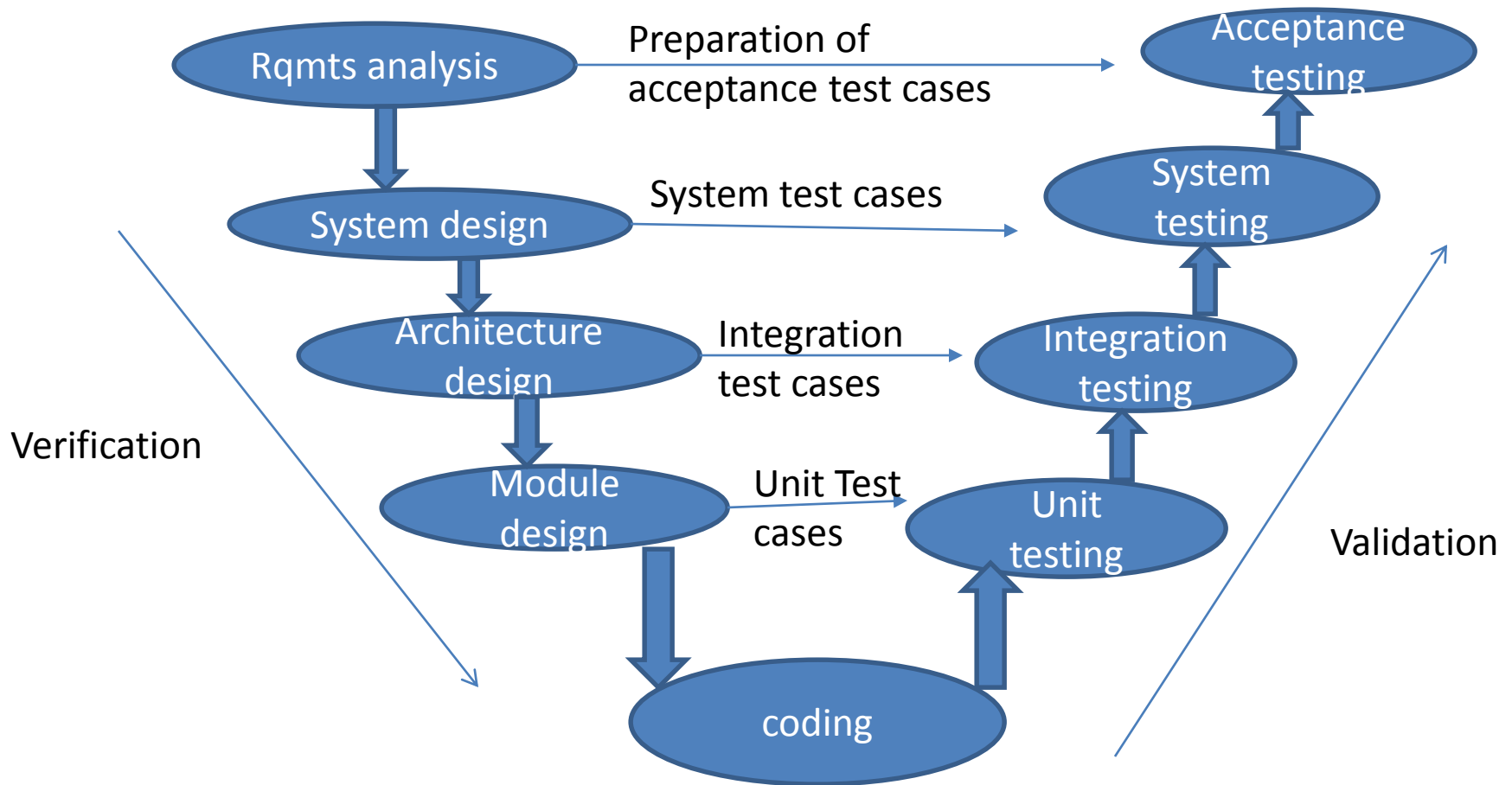
## Verification

- Are we building the product right ?
- It doesn't involve executing the code
- Uses methods like : reviews walkthrough , inspection
- Check whether the s/w conforms to specification
- It is low level exercise.
- Done by QA team

## Validation

- Are we building the right product?
- It involve executing the code
- Uses methods like : black box , white box testing
- Check whether s/w meets customer rqmts.
- It Is high level exercise
- Done by QC team

# V Model



# V Model

- Also known as verification validation model.
- Is an extension of the waterfall model and is based on association of testing phase for each corresponding development stage.
- In this process “ Do Procedure “ would be followed by the developer team and the “check procedure” done by testing team.
- Both activities are working parallel.

- **Advantages:**

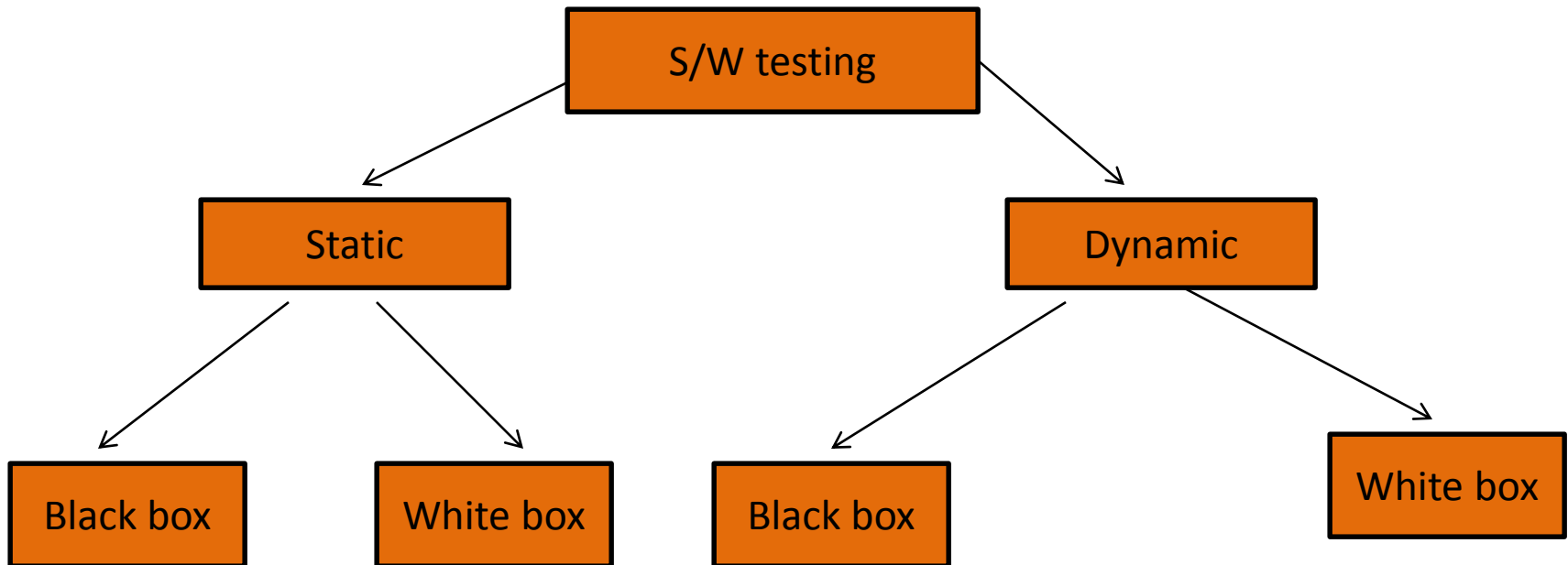
- Testing activities like planning , test designing happens well before coding.
- Time saving and quick
- Work well for smaller projects where requirements are well understand

- **Disadvantages:**

- The model is not flexible to changes
- V- model is very rigid
- Not good for complex projects



# Types of Testing



# Static

- Testing done without executing program
- Does verification
- Prevention of defects
- Checklist and process to be followed
- Cost of finding and fixing defects is low

# Dynamic

- Testing done by executing the program
- Does validation
- Finding and fixing defect
- Involves test cases for execution
- Cost of finding and fixing defects is high

# White box testing

- Detailed investigation of internal logic and structure of the code.
- Also known as glass box / structural testing.
- View code.

- **Advantages Of White box :**
- Easy and simple to find legal i/p data,
- Help in optimizing code.
- Maximum code coverage is done.
- **Disadvantages of White box :**
- Knowledge of code
- Impossible to look into every bit of code

# 1.1 Static White Box testing(SWB)

- Is a type of testing in which the program source code is tested without running it.
- Tester only review and examine s/w design , architecture or code for bug without executing it.
- Eg : review ,inspection , walkthrough
- **Advantages:**
  - cost effective . By reviewing , tester can identify bug.
  - gives ideas to black box tester for test cases.
- **Disadvantages:**
  - Time consuming
  - doesn't find run time environment bugs.

# Reviews in white box testing


- Examining the documents such as rqmts , design , test cases.
- Informal review
- Formal review: simple meetings
- 4 key elements of FR :
- **Identify problem.**
- **Follow rules:** like : amount of code , time , different role.
- **Prepare** : each participant is supposed to be prepare
- **Write a report** : summarizing the result of the review

# Peer Reviews

- Small group of programmers review the code together and look for problem .
- Group consist of :
  - -programmer who wrote the code
    - One or two other programmers or testers acting as reviewer.
- All the 4 key elements must be follow to make review highly effective.
- **Advantages:**
  - fewer scheduling is require.
  - Programmer know code very well.
- **Disadvantages:**
  - A programmer is not the best person to detect problems.


## 1.1.b

## Walkthrough

- Programmer present the code to a small group.
- Large number of participant as compared to peer review , so important to follow rules
- Reviewers should receive copies of s/w in advance. 
- Having at least one senior programmer as reviewer
- Presenter read through the code , reviewer listen and raise question.
- Presenter write a report.

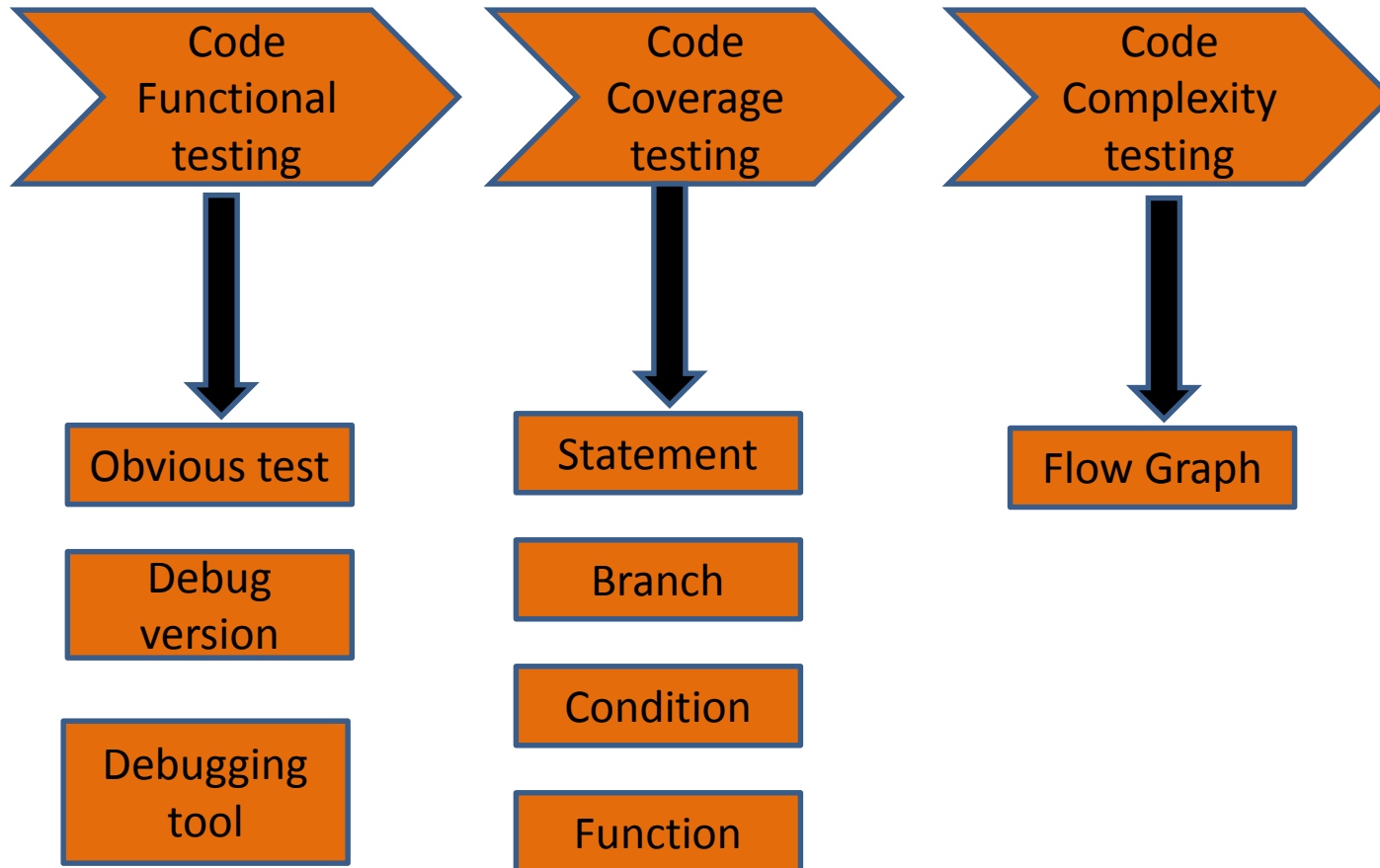


# 1.1. c Inspection

- Most formal type of reviews
- Highly structured and require training for each participant
- Person who present the code , isn't the original programmer.
- Other participant is called inspectors.
- Reviewing the code from different perspective such as a user , a tester or a product support person.
- Some inspector called moderator , recorder : assure that the rules are followed
- One inspector review the code backward.
- After inspection , inspector meet again to discuss the defect they found and prepare report.
- Programmer make changes
- Depending of magnitude of changes , re inspection may be needed
- It require training 

# Dynamic White Box testing(DWB)

## Structural testing



## 1.2 Structural / dynamic white box testing

- Test the s/w by running the code and can also examine the code . Like Testing the s/w with X-ray glasses.
- Tester are required to have knowledge of internal implementation
- Determine what to test , what not to test.
- Also called structural testing.

# 1. 2. a code functional testing

- Quick check.
- Before submitting code to code coverage
- **Methods :**
- Obvious test : knowing i/p and expected o/p
- Debug version : make sure , program is passing through right loops , iterations the right number of times.
- Debugging tools : adding breakpoint in the module to view certain state of variable.

## 1.2.b Code coverage testing

- Is a measure used to describe the degree to which the source code of a program is tested
- Highlight aspects of the code which may not be adequately tested and which require additional testing.
- **Compiler debugger** : it is sufficient for small programs.
- **Code coverage analyzer:**

- **Code coverage analyzer** : hook into s/w . Run transparently in the background while testing.
- Each time a function , line of code , loop is executed , code coverage analyzer record the information
- Prepare statistics which shows :
- What part of the s/w your test case don't cover.
- Which test cases are redundant.
- What new test cases need to be created for better coverage.

# Code coverage methods

- Statement Coverage
- Branch Coverage
- Conditional Coverage
- Function Coverage

# Statement coverage

- It make sure that every statement in the program should execute at least once.
- $\frac{\text{no.of statement exercised}}{\text{total no.of statements}} * 100$

It covers only true conditions to execute all statements

Eg : 1: print "hello world"  
2: print " date is :" ; Date\$  
3: print " time is :" ; Time\$  
4 : End



- Advantage :
- It verify what the written code is expected to do
- Disadvantage:
- It cant test false condition
- It doesn't report that whether the loop reaches its termination condition
- It doesn't understand logical operators

# Branch coverage

- It cover all the paths in the s/w
- Ensure that whether a program can jump to all possible destinations
- Also known as decision coverage bcoz it check for both True and False .
- Report whether boolean expressions tested in control structure evaluated to both true & false.
- $\frac{\text{Number of decisions outcomes tested}}{\text{Total number of decision outcomes}} * 100$

- Eg : 1: print “hello world”
  - 2: if Date\$=“01-01-2016 then
  - 3: print “ happy new year”
  - 4:end if
  - 5:print “ date is: “ ; Date\$
  - 6: print “ time is “ ;Time\$
  - 7:End

Date \$	Line #
01-01-2016	1 ,2 ,3 ,4 ,5 ,6 ,7
01-02-2016	1 , 2 , 5 , 6 , 7

Disadvantage : ignores branches within boolean expression.

# Condition Coverage

- Also known as predicate coverage
- Each one of the boolean expression have been evaluated for both TRUE FALSE.
- Takes the extra conditions on the branch statement.
- $\frac{\text{Total decisions exercised}}{\text{Total no.of decisions}} * 100$

- Eg : 1: print “hello world”
  - 2: if Date\$=“01-01-2016 AND Time\$=“00:00:00 “ then
  - 3: print “ happy new year”
  - 4:End if
  - 5:print “ date is: “ ; Date\$
  - 6: print “ time is “ ;Time\$
  - 7:End

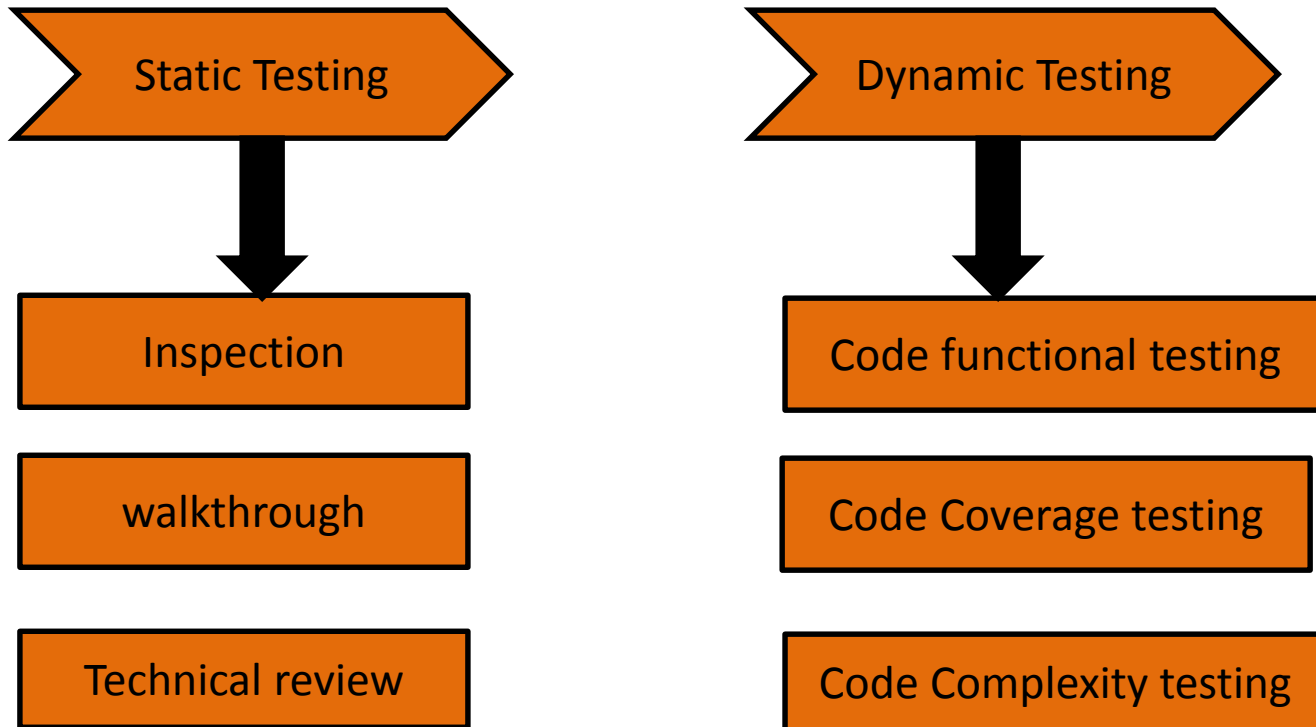
Date\$	Time\$	Line #
00-00-2016	00:00:00	1 , 2, 5 , 6, 7
01-01-2016	11:11:11	1 , 2 , 5, 6, 7
00-00-2016	11:11:11	1 , 2 , 5, 6, 7
01-01-2016	00:00:00	1 ,2 ,3 ,4, 5, ,6 ,7

- When consider branch coverage , first 3 conditions would be redundant and could be equivalence partitioned into single test case.
- With condition coverage all 4 cases are important.
- Condition coverage is sufficient for code coverage.

# Function coverage

- Identify how many program functions are covered by test cases
- Measure how many times a given function is called

# 1 . White box testing

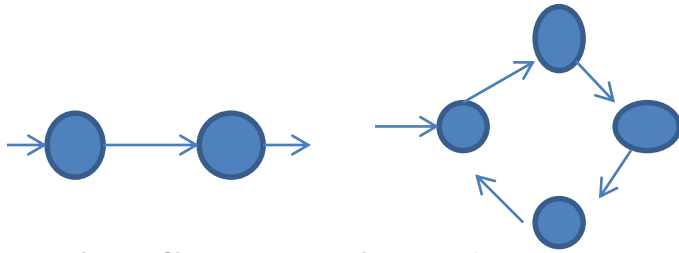




## 1.2.c : Code complexity testing

- Cyclomatic complexity is a source code complexity measurement , calculated by developing a control flow graph of the code.
- It measure the amount of decision logic in s/w

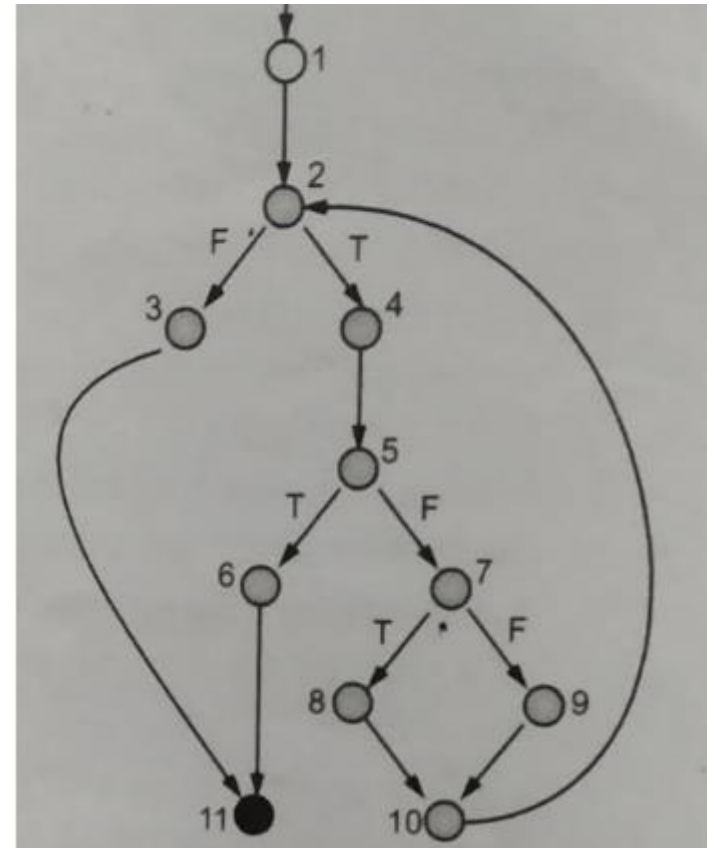
# Flow graph notation



- circle : flow graph node represent statement.
- Arrows: edges , represent flow of control
- An edge must terminate at a node , even if the node doesn't represent any statements.
- Areas bounded by edges and nodes are called regions.
- Complexity is computed in one of three way:
- 1. no. of regions of flow graph
- 2.  $V(G) = E - N + 2$ 
  - E- number of edges
  - N - number of nodes
- 3.  $V(G) = P + 1$ 
  - p - number of decision points

# Example of flow graph

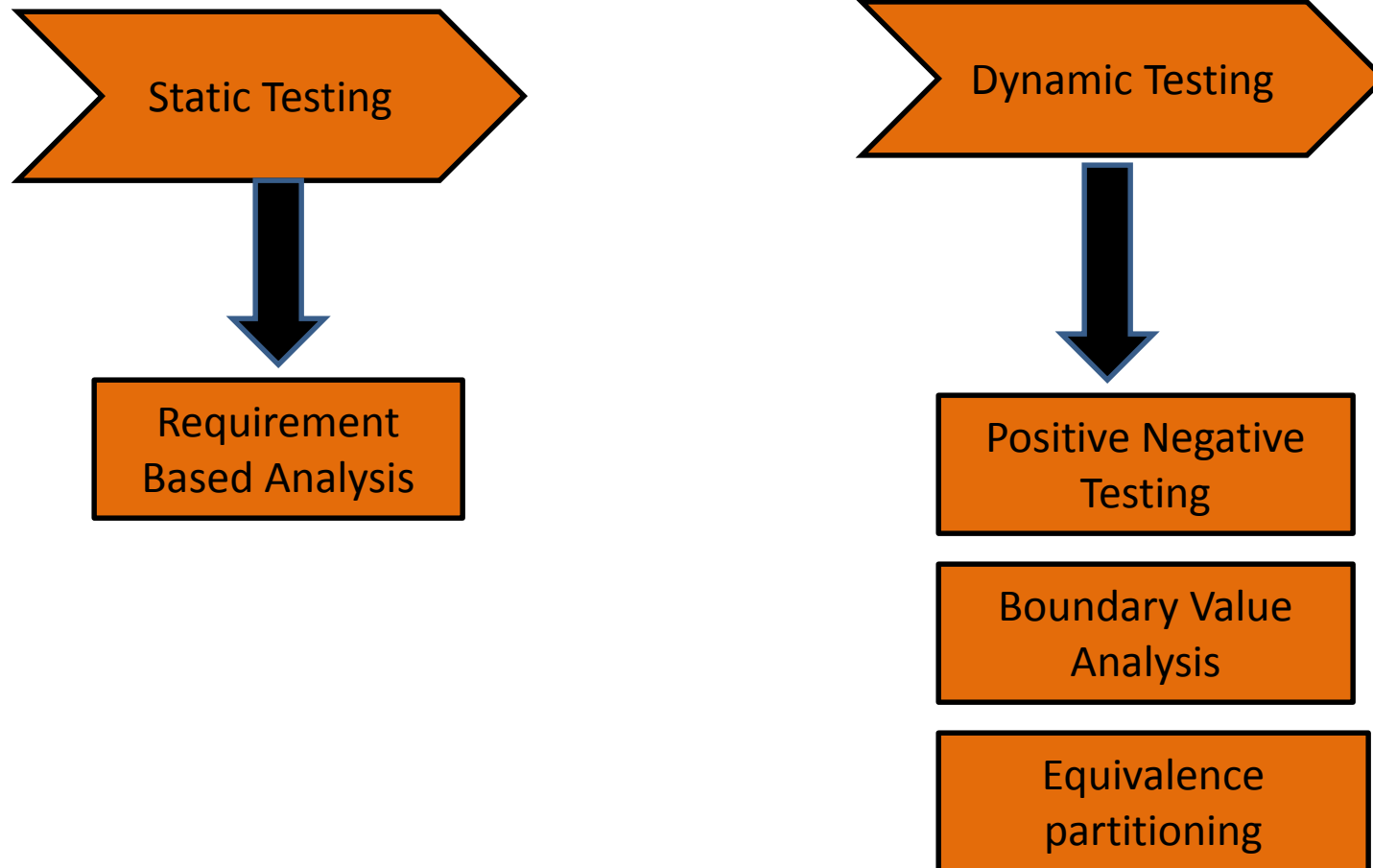
- $V(G)=E-N+2$
- Where  $E=14$  and  $N=12$
- So  $V(G)=14-12+2=4$
- Another way to compute
- $V(G)=P+1$
- $P=3$
- So  $V(G)=3+1=4$



# Black box testing

- Behavioral testing
- Internal implementation of s/w being tested is not known to the tester
- Derives sets of i/p conditions that will fully exercise all functional requirements for a program
- Eg : tester , without knowledge of the internal structure of a website, tests the web pages by using a browser , providing i/p's and verifying o/p's

# Black Box testing



# Black box testing

- **Advantage:**
  - Tester can be nontechnical
  - No need to have detailed functional knowledge of system.
- **Disadvantage:**
  - Challenging to design test cases without having clear functional specification.
  - Difficult to identify tricky i/p's if test cases not developed based on specification.

# Static Black Box Testing(SBB)

- **Static means** : no execution.
- **Black box means** : code is not visible
- **SBB** : tester test the software by reviewing the things (no execution) without looking inside the code.
- Example :
- Tester can only examine the requirements.

# Requirements based analysis

- Detailed review of the requirements specification. Pretend to be a customer.
- Test cases , data , conditions are derived from requirements.
- It includes functional tests : standards , guidelines , review similar s/w .
- non functional attributes such as statements are feasible , consistent , relevant , code free .
- Terminology check list :
- -- should contain : if ... then missing
- Avoid the terms : Always , sometime , often , good , fast , efficient



# Dynamic Black Box Testing(DBB)

- **Dynamic means** : execution of code
- **Black Box means** : code is not visible
- **DBB** : tester can test the s/w by executing the code but have no access of code.
- Tester simply execute the created test cases and check the result with expected result.
- Example : tester test Website by using browser without knowledge of internal structure of website

# Positive Negative testing

- **Positive** : System validated against valid i/p data.
- It check whether an application behaves as expected with positive i/p.
- Intention is to check whether s/w application not showing error when not supposed to and showing error when supposed to.

Enter only numbers

999999

- **Negative** : system validated against the invalid i/p data.
- It check whether an application behaves as expected with negative i/p.
- Intention is to check whether s/w application not showing error when supposed to and showing error when not supposed to.
- Goal is to check stability of the s/w against incorrect data set.

Enter numbers

abcd

## **Positive testing**

- Testing System by giving valid data
- Check for only valid set of values
- Verify the known set of test conditions
- Aim: project works as per specifications

## **Negative testing**

- Testing System by giving invalid data
- Check for only invalid set of values
- Done to break the project with unknown set of test conditions
- Aim: break the application by providing invalid set of data.

# Boundary value analysis(BVA)

- If s/w can operate on the edge of its capabilities , it will almost operate well under normal condition.
- BVA : testing at the boundaries
- Test cases are designed by using boundary values
- **Eg** :if password field should accept minimum 8 and maximum 12 character then
- check password field on its boundaries (**valid partition**) ie check field by applying i/p of exactly 8 characters and by applying i/p of exactly 12 characters
- Check password field outside **boundaries(Invalid partition)** ie check field by applying i/p of 7 character or 13 character.

- Types of boundary condition :
- Numeric -- character -- quantity -- speed
- Characteristics of those types:
- first/last -- min/max -- start/finish
- Eg : field allow 1 -255 characters
- Eg : program read / write to a CD
- Eg : program allow to print multiple pages onto single page

# Equivalence partitioning

- Is a process of methodically reducing the huge set test case into smaller manageable set that still adequately test s/w.
- If s/w behaves in an identical way for a set of value , then the set is termed as equivalence partition.
- Objective : i/p data is analyzed and divided into equivalence classes which produces different o/p.
- Example : two test cases which gives same output then do equivalence partition , and select one test case and ignore other .
- Systematic means for selecting test cases that matter and ignoring the one that don't
- Can be subjective :
- 2 testers have two different set of partition

- Two steps :
  - -- identifying equivalence partition
  - -- picking one value from each partition for the complete coverage.
- Advantages :
  - Reduces total number of test cases.
- Disadvantage:
  - Risk of eliminating test cases that could reveal bug



- Example : to copy in calculator: five ways:
  - 1. Click copy menu item
  - 2. type c
  - 3. type C when menu displayed
  - 4. press ctrl+c
  - 5. press ctrl+shift+c
- 
- Partition these 5 i/p paths into 3
  - 1. Click copy menu item
  - 2. type c
  - 3. press ctrl+c