Testing

Mehwish Mumtaz

Software Faults and Failures

Why Does Software Fail?

- Wrong requirement: not what the customer wants
- Missing requirement
- Requirement impossible to implement
- Faulty design
- Faulty code
- Improperly implemented design

Objective of Testing

- Objective of testing: discover faults
- A test is successful only when a fault is discovered
  - Fault identification is the process of determining what fault caused the failure
  - Fault correction is the process of making changes to the system so that the faults are removed

Elements of a Test Case

- Purpose
- Input
- Expected Output
- Actual Output
- Sample Format:

Test Case ID	What To Test?	How to Test?	Input Data	Expected Result	Actual Result	Pass/Fail
.	.	.	.	.	.	.

Types of Faults

- Algorithmic fault: e.g:**functionality of a search engine**
- Computation and precision fault
  - a formula's implementation is wrong: example round off error
- Documentation fault
  - Documentation doesn't match what program does
- Capacity or boundary faults
  - System's performance not acceptable when certain limits are reached.
- Performance faults
  - System does not perform at the speed prescribed

Different Level of Failure Severity

- Catastrophic: causes death or system loss e.g radiation machine
- Critical: causes severe injury or major system damage:example reactors,airplane
- Marginal: causes minor injury or minor system damage
- Minor: causes no injury or system damage

Testing Issues

Who Performs the Test?

- Independent test team
  - avoid conflict
    - personal responsibility vs need to discover faults
  - allow testing and coding concurrently

Testing Types

The Black Box Test is a test that only considers the external behavior of the system; the internal workings of the software is not taken into account.

White Box Test is a method used to test a software taking into consideration its internal functioning. It is carried out by testers.

Test Cases for Triangle Problem

Test cases based on equivalence classes within valid range

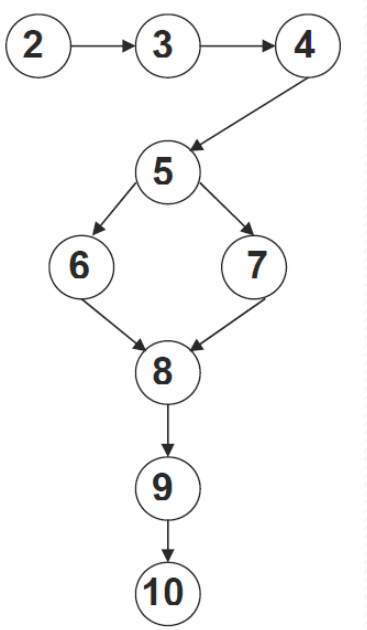
TC Id	What to test	A	B	C	EO
1	EC 1	15	15	15	Equilateral
2	EC 2	10	20	10	Isosceles
3	EC 3	5	6	7	Scalene
4	EC 4	1	2	10	Not a triangle

Test cases based on input range

TC Id	What to test	A	B	C	EO
1	EC1 of A	-10	15	15	Value of A not in valid range
2	EC1 of B	15	-10	15	Value of B not in valid range
3	EC1 of C	15	15	-10	Value of C not in valid range
4	EC3 of A	-205	15	15	Value of A not in valid range
5	EC3 of B	15	-205	15	Value of B not in valid range
6	EC3 of C	15	15	-205	Value of C not in valid range
7	EC2 of A, B, C	15	15	15	Equilateral

Whitebox Testing
Control Flow Graph

1. *Program 'Simple Subtraction'*
2. *Input (x, y)*
3. *Output (x)*
4. *Output (y)*
5. *If x > y then DO*
6. *x – y = z*
7. *Else y – x = z*
8. *EndIf*
9. *Output (z)*
10. *Output "End Program"*



Edges and Nodes Method
Cyclomatic Complexity = E – N + 2 = 9 – 9 + 2 = 2

Branch Method(Ifs,for,while)
C=B+1=1+1=2

Whitebox Testing
Control Flow Graph

Acceptable ranges. . .

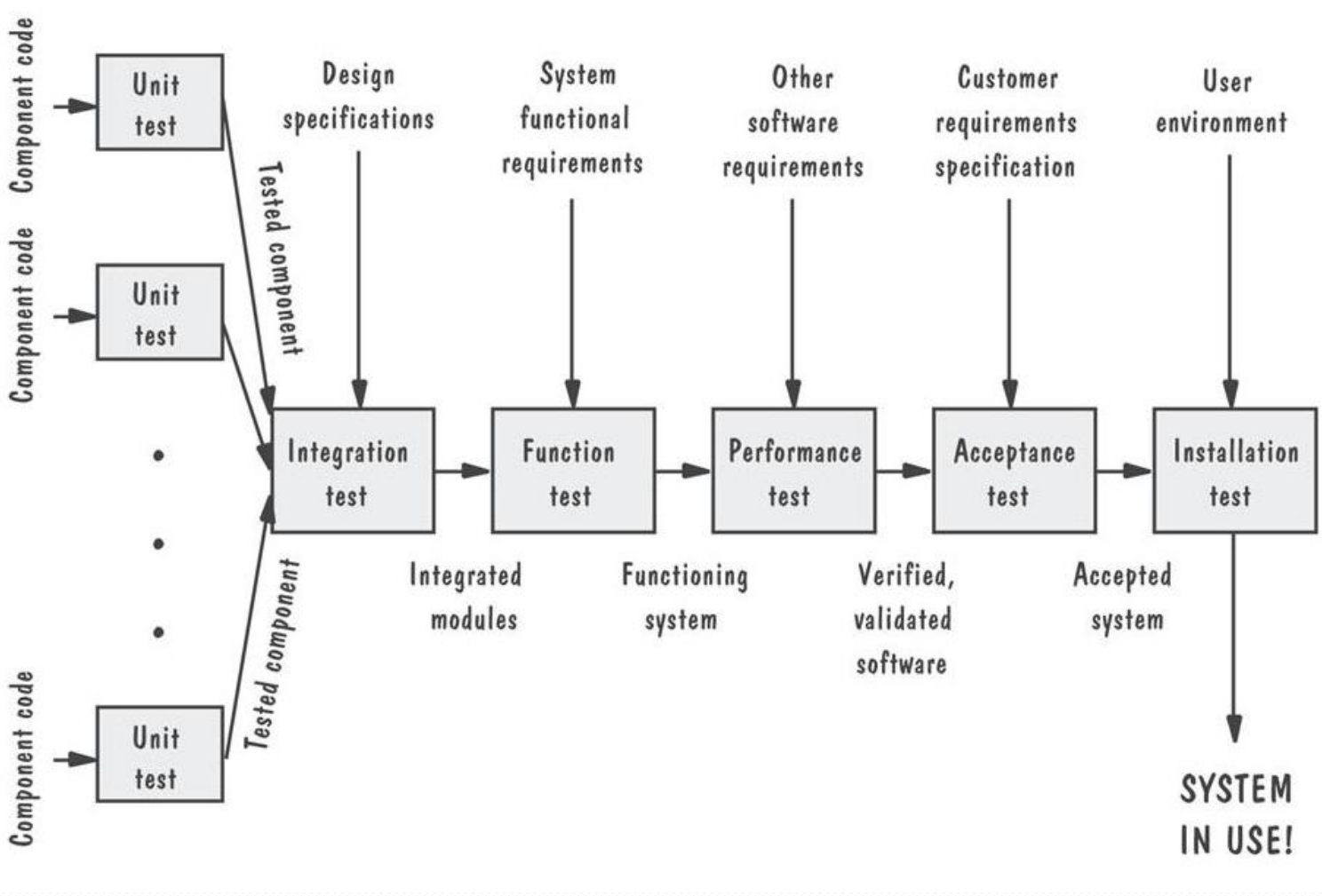
- 1 – 5 = **Easy to maintain**
- 6 – 10 = **Difficult**
- 11-15 = **Very Difficult**
- 20+ = **Approaching Impossible**

Levels of Testing

- Module testing, component testing, or unit testing
- Integration testing
- System Testing
  - Function testing
  - Performance testing
- Acceptance testing
- Installation testing

Levels of Testing

Testing Organization Illustrated



Unit Testing

Code Review

- Code walkthrough(Informal)
- Code inspection(Formal)

Unit Testing

- Testing the unit for correct functionality
- Testing the unit for correct execution

Unit Testing

Steps in Testing

- Determining test objectives
- Selecting test cases
- Executing test cases

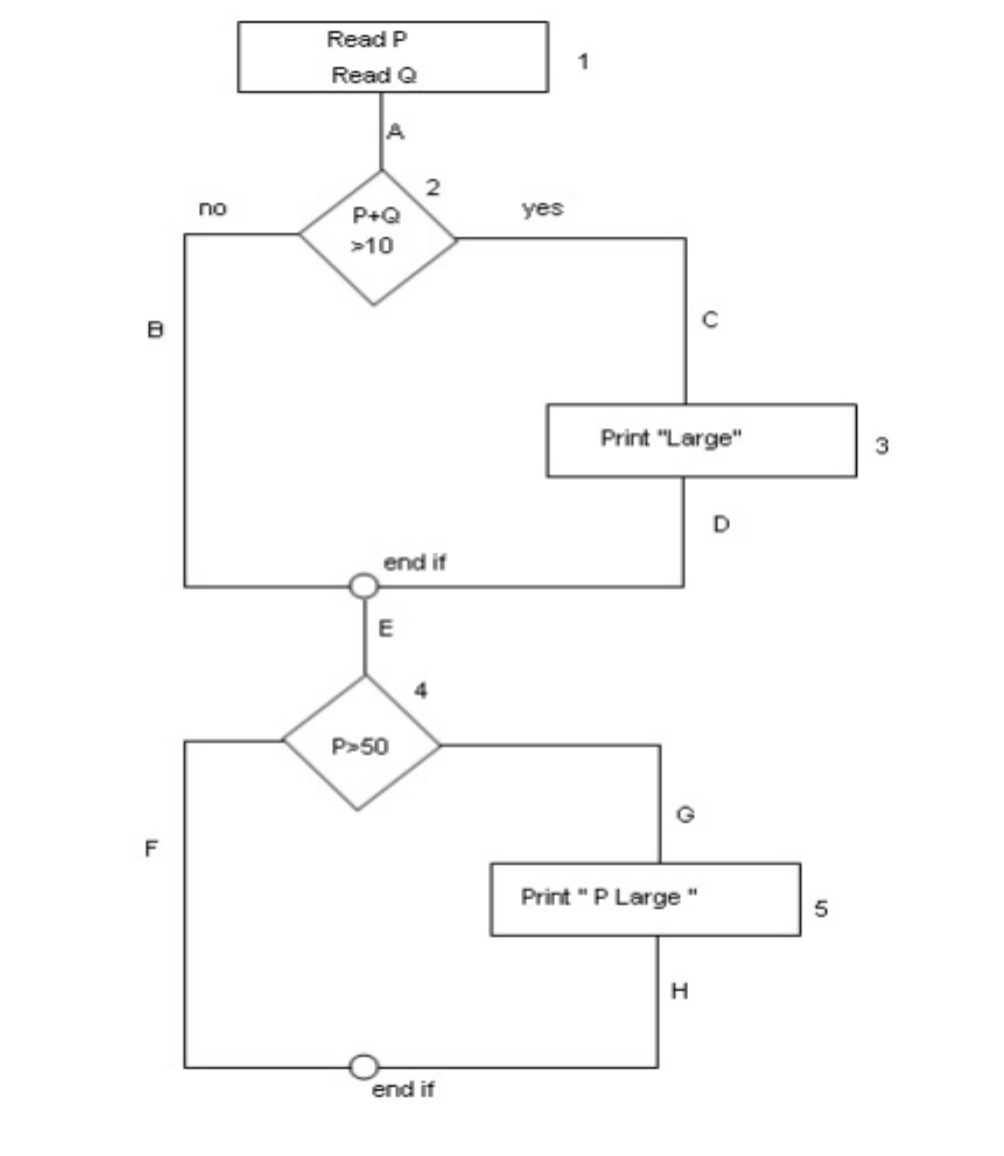
Unit Testing

- Statement testing
  - In this the test case is executed in such a way that every statement of the code is executed at least once
- Branch testing

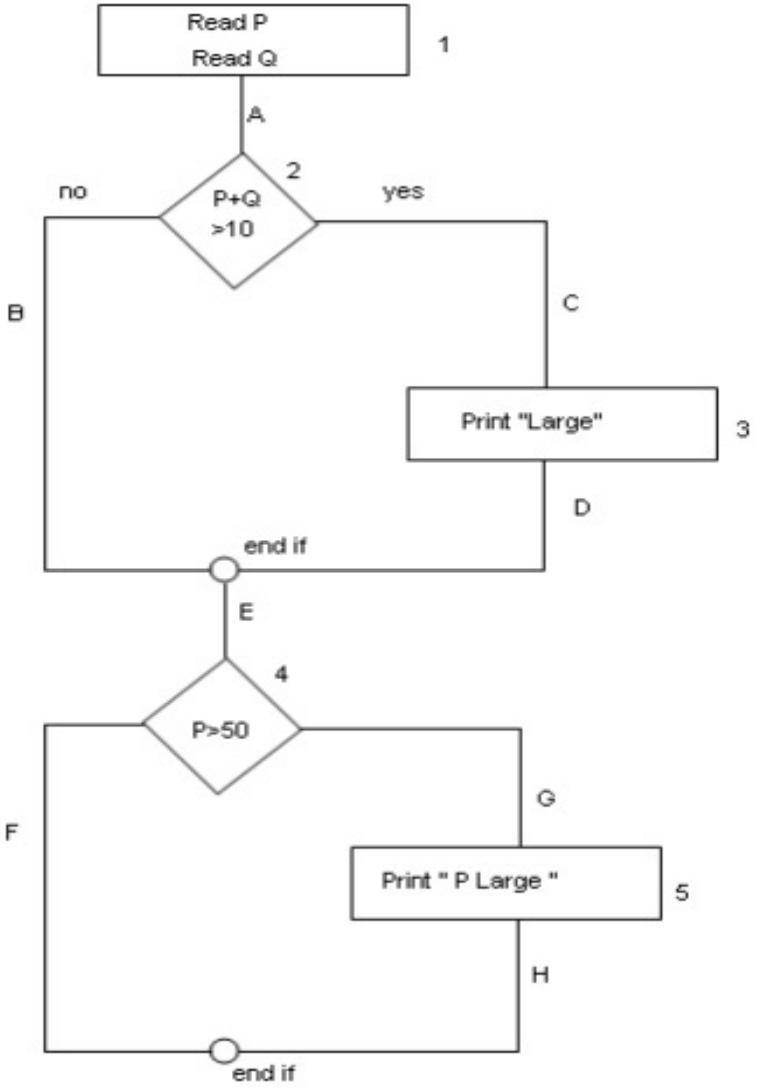
  Test coverage criteria requires enough test cases such that each condition in a decision takes on all possible outcomes at least once, and each point of entry to a program or subroutine is invoked at least once. That is, every branch (decision) taken each way, true and false. It helps in validating all the branches in the code making sure that no branch leads to abnormal behavior of the application.
- Path testing
  - In this the test case is executed in such a way that every path is executed at least once.

Example:
Read P
Read Q
IF P+Q > 100 THEN
Print “Large”
ENDIF
If P > 50 THEN
Print “P Large”
ENDIF

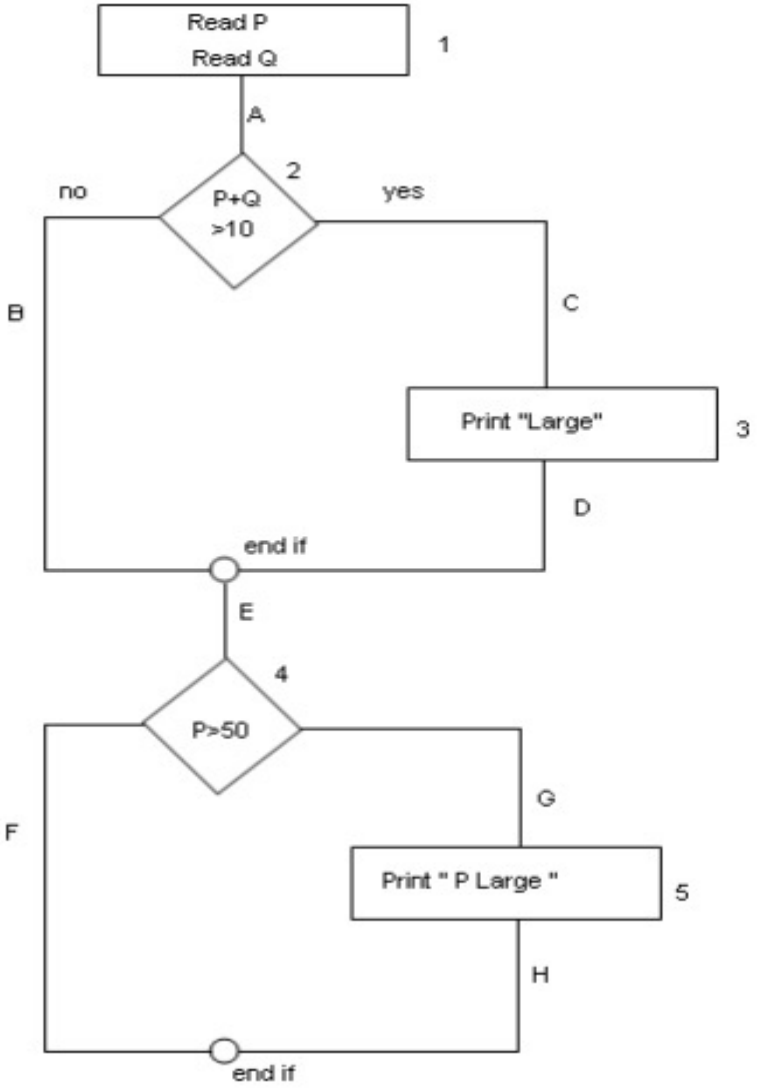
Consider the flow chart



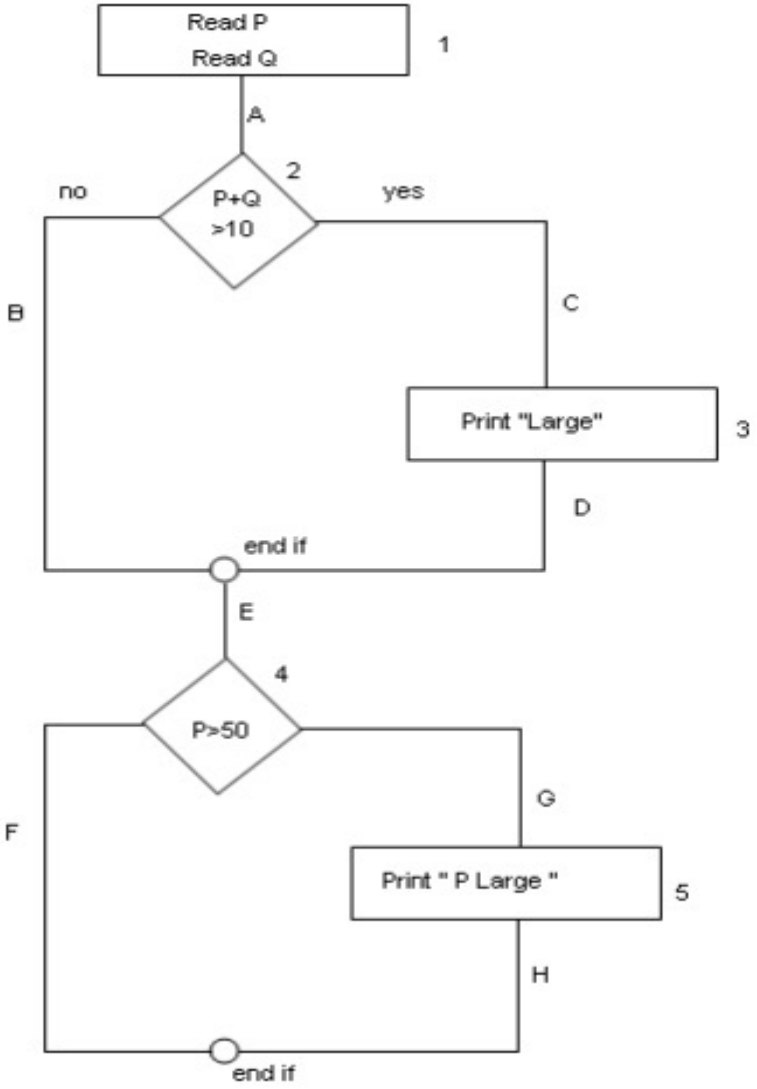
Statement Coverage (SC):
To calculate Statement Coverage, find out the shortest number of paths following
which all the nodes will be covered. Here by traversing through path 1A-2C-3D-E-4G-5H all
the nodes are covered. So by traveling through only one path all the nodes 12345 are covered,
so the Statement coverage in this case is 1.



To calculate Branch Coverage, find out the minimum number of paths which will ensure covering of all the edges. In this case there is no single path which will ensure coverage of all the edges at one go. By following paths 1A-2C-3D-E-4G-5H, maximum numbers of edges (A, C, D, E, G and H) are covered but edges B and F are left. To covers these edges we can follow 1A-2B-E-4F. By the combining the above two paths we can ensure of traveling
through all the paths. Hence Branch Coverage is 2. The aim is to cover all possible true/false decisions.



Path Coverage (PC):
Path Coverage ensures covering of all the paths from start to end.
All possible paths are-
1A-2B-E-4F
1A-2B-E-4G-5H
1A-2C-3D-E-4G-5H
1A-2C-3D-E-4F
So path coverage is 4.
Thus for the above example SC=1, BC=2 and PC=4.

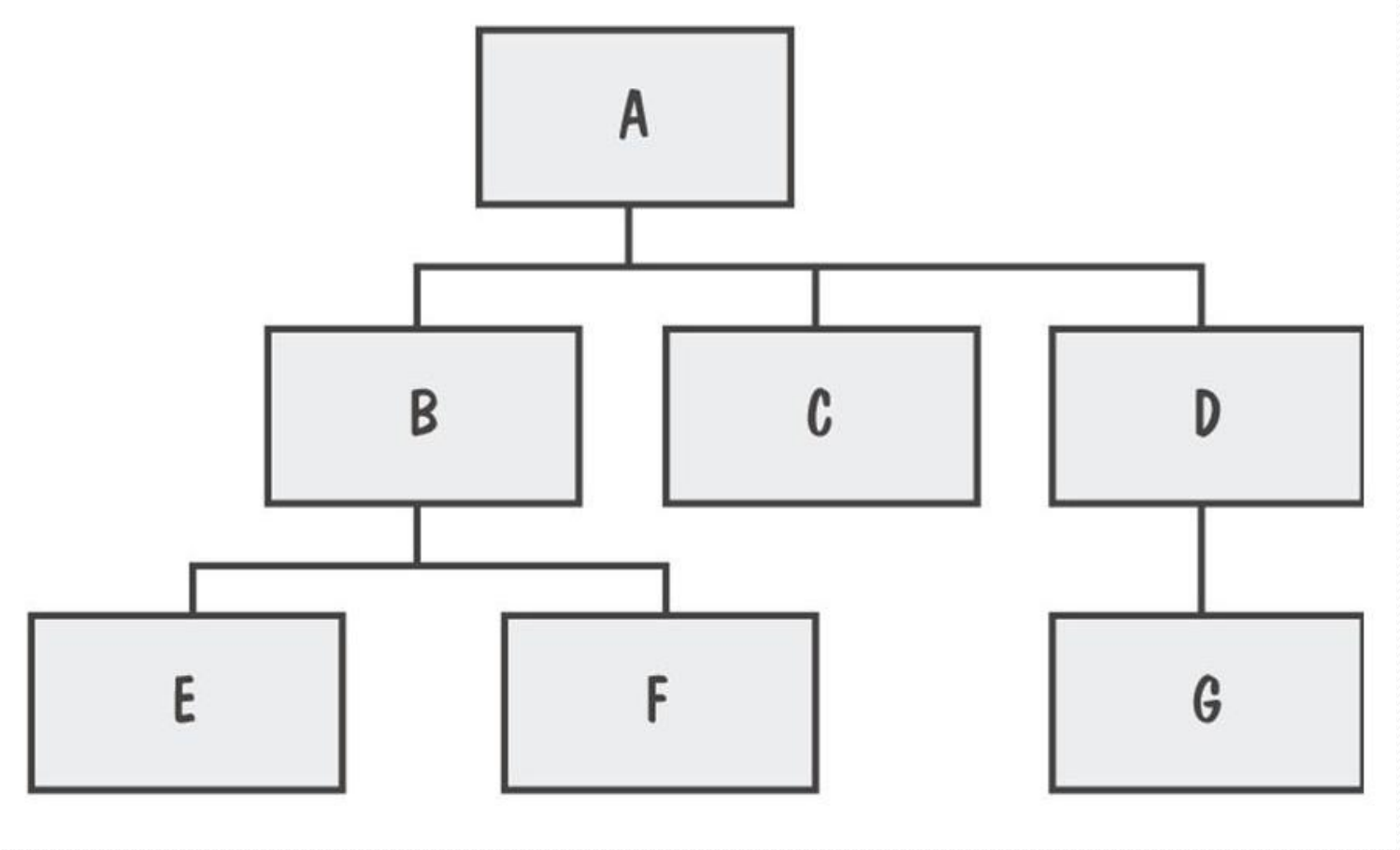


Integration Testing

- Big-bang
- Bottom-up
- Top-down
- Sandwich testing

Integration Testing

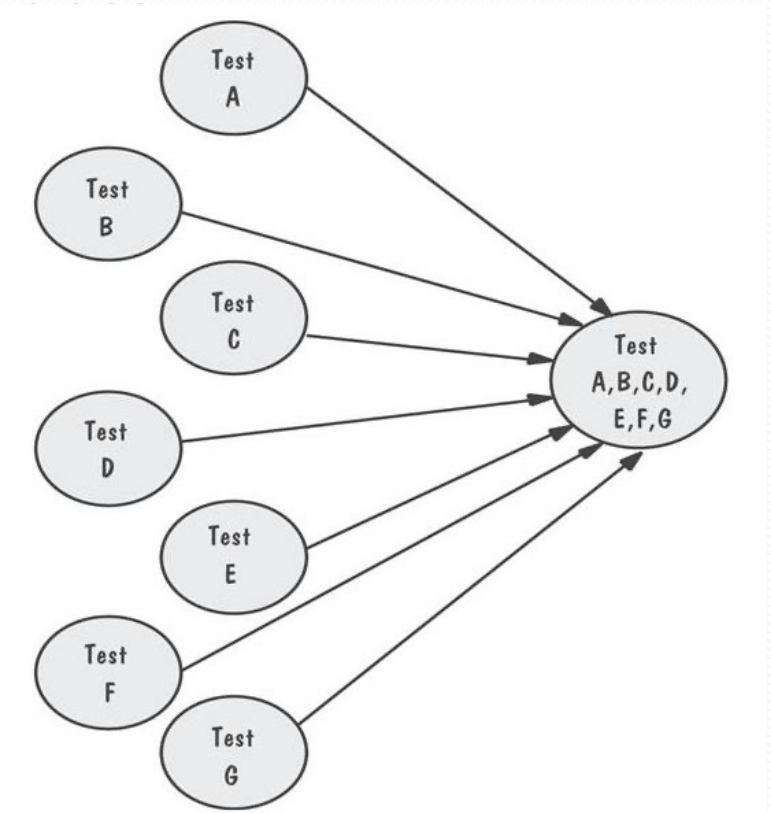
- System viewed as a hierarchy of components



Integration Testing

Bing-Bang Integration Example

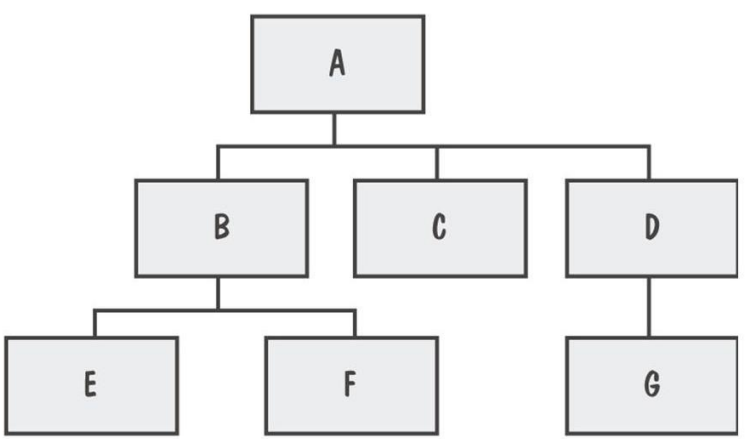
- All components integrated at
- Locating faults?

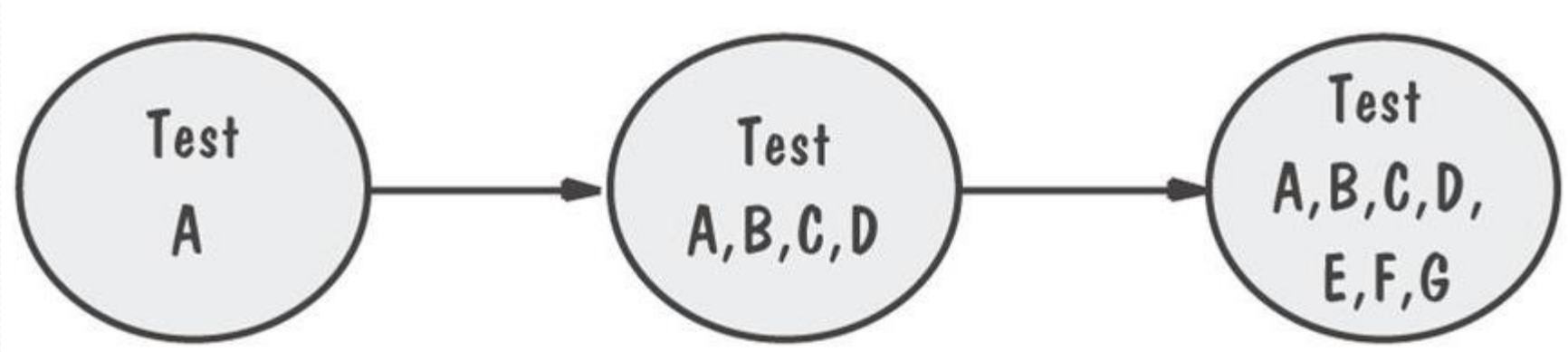


Integration Testing

Top-Down Integration Example

- Integrate from Top to bottom
- Locating faults?

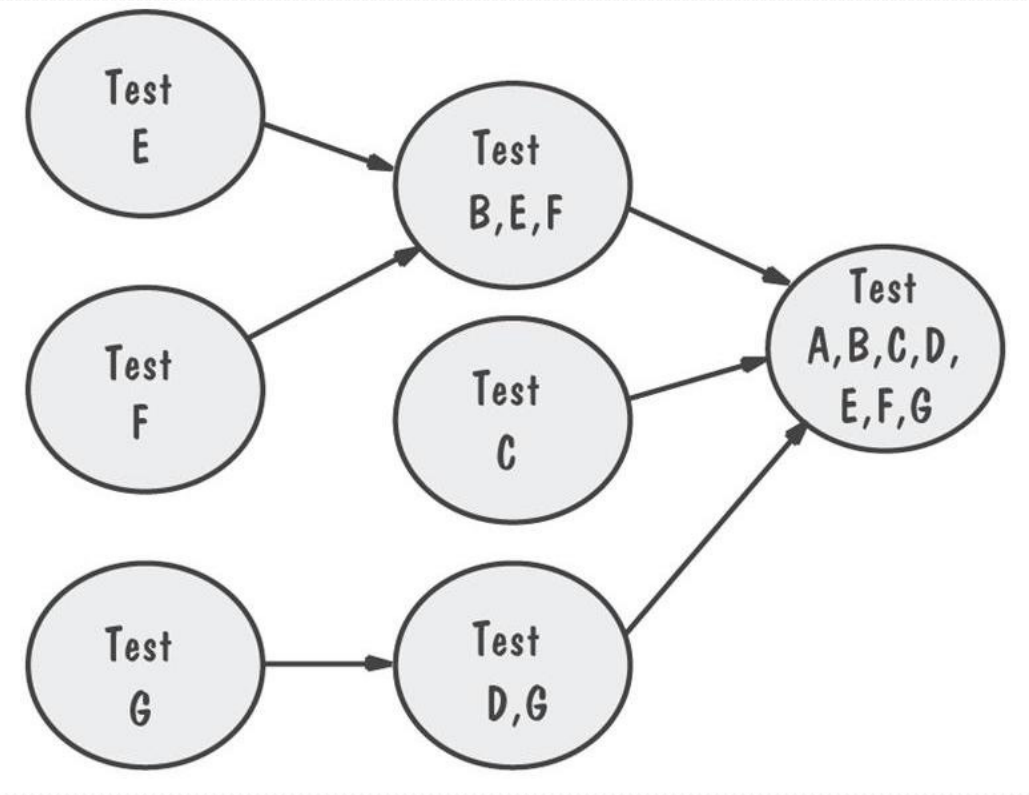




Integration Testing

Bottom-Up Integration Example

- Locating faults?



Integration Testing

Sandwich Integration

- Integrate both

System Testing

- System is tested as a whole
- Different types of testing considered during System Testing:

Assignment

Functional Testing (GUI)
- Performance Testing
- Usability
- Load
- Volume
- Stress
- Security
- Scalability
- Sanity
- Smoke
- Regression
- Compatibility
- Installation
- Ad hoc

Steps to do System Testing

The following steps are important to perform System Testing:

- Step 1: Create a System Test Plan
- Step 2: Create Test Cases
- Step 3: Carefully Build Data used as Input for System Testing
- Step 3: If applicable create scripts to
- - Build environment and
- - to automate Execution of test cases
- Step 4: Execute the test cases
- Step 5: Fix the bugs if any and re test the code
- Step 6: Repeat the test cycle as necessary

Acceptance Tests

Purpose and Roles

- Enable the customers and users to determine if the built system meets their needs and expectations
- Written, conducted and evaluated by the customers

Acceptance Tests

Types of Acceptance Tests

- Pilot test: install on experimental basis
- Alpha test: in-house test
- Beta test: customer pilot

Result of Acceptance Tests

- List of requirements that
  - are not satisfied
  - must be deleted
  - must be revised
  - must be added

Installation Testing

- Before the testing
  - Configure the system
  - Attach proper number and kind of devices
  - Establish communication with other system
- The testing
  - to verify that the system has been installed properly and works

Software Testing Tools

- Automated Testing Tools
  - Selenium, QTP, SilkTest, WinRunner, LoadRunner, Jmeter
- Testing Management Tools
  - TestManager, TestDirector
- Bug Tracking/Configuration Management Tools
  - Bugzilla, Jitterbug, SilkRadar

When to Stop Testing

- No time left
- No money left
- Statistical Criteria
  - Number of defects found per week becomes lower than a set threshold

Test Planning

- Establish test objectives
- Design and Write test cases
- Test test cases
- Execute tests
- Evaluate test results

Test Planning

Purpose of the Plan

- Test plan explains
  - who does the testing
  - why the tests are performed
  - how tests are conducted
  - when the tests are scheduled

Test Planning

Contents of the Plan

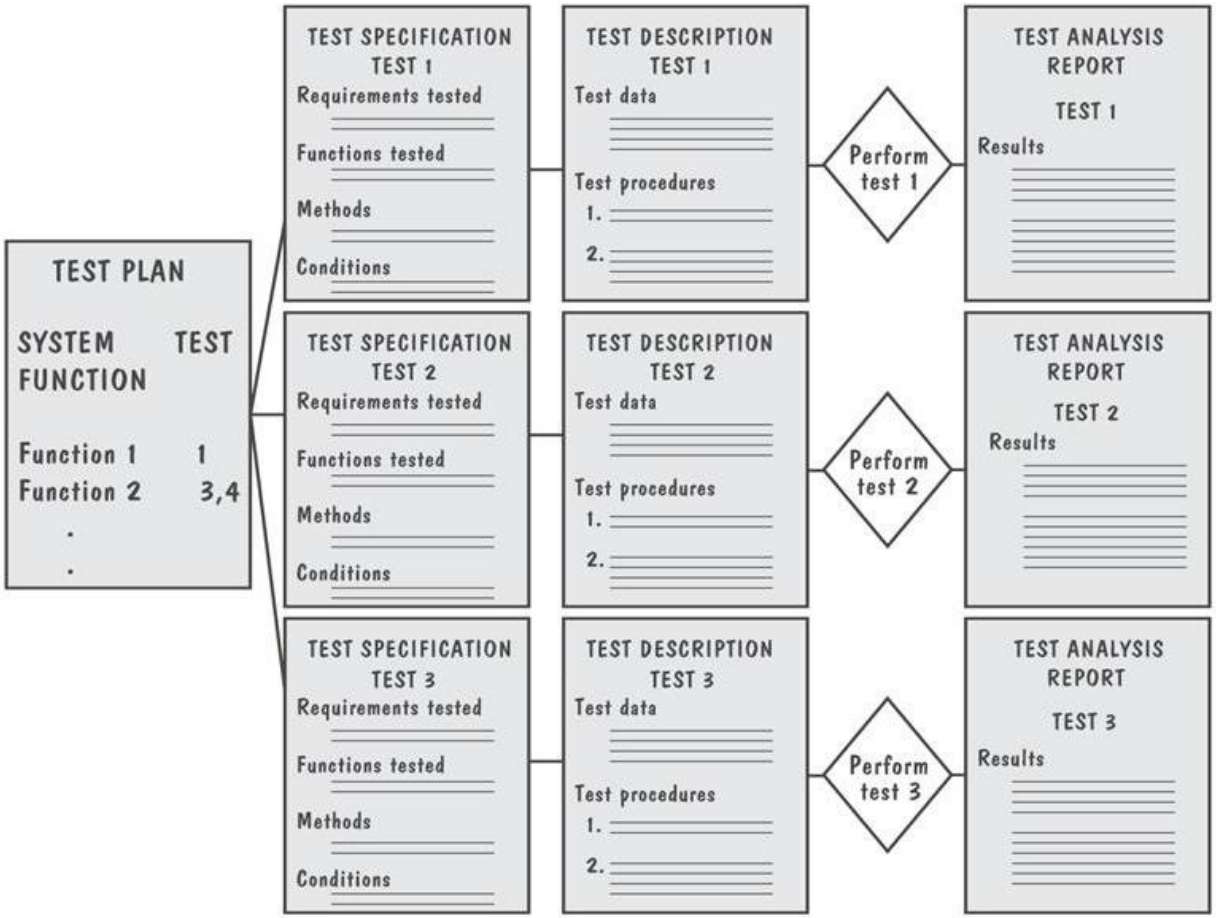
- What the test objectives are
- How the test will be run
- What criteria will be used to determine when the testing is complete

Test Documentation

- Test plan: describes system and plan for exercising all functions and characteristics
- Test specification and evaluation: details each test and defines criteria for evaluating each feature
- Test description: test data and procedures for each test
- Test analysis report: results of each test

Test Documentation

Documents Produced During Testing



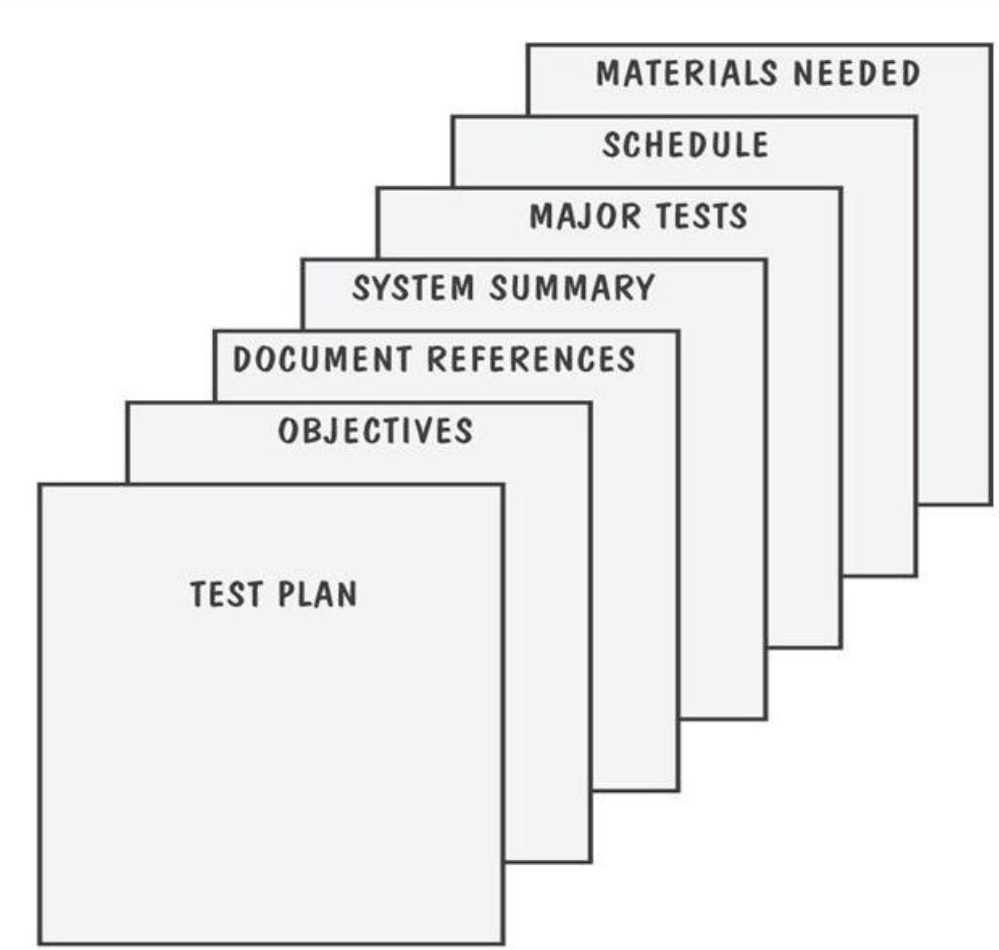
Test Documentation

Test Plan

- The plan begins by stating its objectives, which should
  - guide the management of testing
  - guide the technical effort required during testing
  - establish test planning and scheduling
  - explain the nature and extent of each test
  - explain how the test will completely evaluate system function and performance
  - document test input, specific test procedures, and expected outcomes

Test Documentation

Parts of a Test Plan



Test Documentation

Test Analysis Report

- Documents the result of test
- Provides information needed to duplicate the failure and to locate and fix the source of the problem
- Provides information necessary to determine if the project is complete
- Establish confidence in the system's performance

Test Documentation

Problem Report Forms

- Location: Where did the problem occur?
- Timing: When did it occur?
- Symptom: What was observed?
- End result: What were the consequences?
- Mechanism: How did it occur?
- Cause: Why did it occur?
- Severity: How much was the user or business affected?
- Cost: How much did it cost?

References

- UCF Slides
- Software Testing, A Craftsman's Approach by Jorgensen
- Software Testing Tools by Prasad
- Software Engineering, Business Continuity, and Education. Kim, Adeli et al (Eds.)
- https://www.guru99.com/cyclomatic-complexity.html