1.Basic of Software Testing and Testing Methods

- Introduction
- Terminologies
- Entry &Exit criteria
- Verification and Validation
- Quality Assurance
- Methods of Testing
- The Box Approach
- Black Box Testing

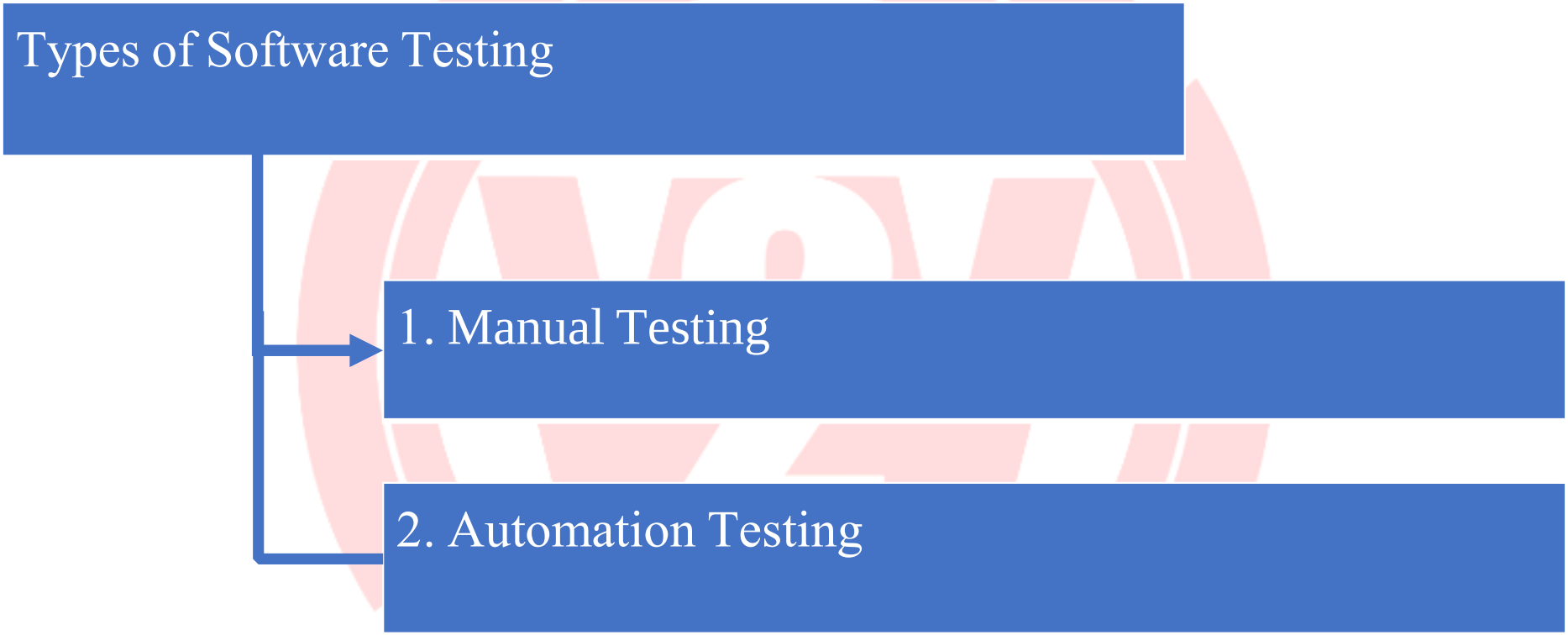
Software Testing

- Verify weather actual results are same as expected results?
- Finding defects/errors
- To find weather all user requirements are fulfilled or not?
- Testing gives assurance about the quality of the product
- Testing uses-Test Cases and Test Data
- Test Case- collection of actions applied on software to check specific feature or functionality
- Test Suit- collection of test cases
- Test data is used for positive testing and negative testing(expected result and unexpected input)

Advantages of Software Testing

- Testing reduces the possibility of failure by removing errors
- Helps to deliver qualitative software to customer
- Ensures correctness and Completeness of software
- Allows to verify and validate that s/w meets customers requirements
- Enables s/w to produce expected output
- Prevents s/w from producing unexpected outcomes

Types of Software Testing



Manual Testing

- Tester treats himself as a user and tests all modules according to requirements provided by users
- Tester manually runs all test cases without any automation tools
- It's a very basic type
- Manual testing is performed before going for automation testing
- Knowledge of testing tools is not required
- More efforts required
- Used to check automation feasibility
- 100% automation is not possible so we need to perform manual testing

Goals of Manual Testing

- To give assurance that software is defect free and fulfills all requirements of end users
- Test suits designed in testing phase test all functionalities of s/w
- Manual testing assures that defects are fixed by developed and tester performs testing after fixing defects
- Manual testing verifies quality of the product and deploys bug free software
- Types Manual Testing :-
- Unit Testing
- Integration Testing
- System Testing

- Acceptance Testing

Automation Testing

- Automation tools are used to run test cases to find out bugs from s/w products
- Automation s/w can enter test data into testing s/w and matches expected and actual results to create test reports
- Requires more money and resources such as employees and testing tools etc.
- After every occurrence of error we need to make changes into the code and run the same test suite again and again

- Test suite can be recorded using test automation tools and replay it when needed
- Human interaction is not required during testing
- Automation testing decreases the test cases executed manually
- It does not replace manual testing completely
- For automation testing we need generation of –test cases , test data and test scripts
- Automation testing is performed if requirements of the projects are stable (not frequently changing)
- Types of automation tools:
- Selenium
- Mentis
- Quality Test Professional(QTP)

- Bugzilla
- HP ALM(Application Lifecycle Managemant)

➤ (MSBTE : S-17)

Q. 1.1.5 Differentiate between manual and automation testing. (Ref. Sec. 1.1.3) **S-17, 4 Marks**

Parameter	Manual Testing	Automation Testing
Human interaction	To perform manual testing human interaction is required for test execution.	To perform Automation Testing, human interaction is not required. In automation testing we use tools to run the test cases.
Requirements	To perform manual testing, we need skilled employees, more time and high costs.	To perform automation testing, we require automation tools. Automation testing saves time, cost and manpower. Once test suite is recorded in automation tool then it can easily run test suit.
Application to Test	Any application can be first tested manually. Informal testing like ad-hoc and monkey testing are performed manually.	Automated testing is used to test an application whose requirements are stable at some extent and automation testing mainly used to perform Regression Testing.
Execution of Same test cases	We cannot use manual testing while executing same test suit. Repetitive execution of test cases become boring and error prone.	The most boring part which is of running same test cases repetitively can be performed with the help of automation testing.

Principles of Software Testing

1) Complete testing of a software is not possible-best possible amount of testing is done with the help of risk assessment
2) Defect clustering- risky modules are identified and tested
3) Pesticide Paradox –repeated use of test cases cannot find new defects so test cases are reviewed and revised-new test cases are added to find new defects
4) Testing shows presence of defects-but it cannot prove that system is 100% error free
5) Absence of error-does not mean system meets exact requirements

6) Early testing-test must start as early as possible-its less costly to correct bugs at early stages
7) Testing is context dependent-it differs from project to project

Objectives of Software Testing

- Finding defects generated during coding
- Report defects to developers for correction and retest for assurance
- Testing is done to guarantee that s/w fulfills all requirements from end user
- To give assurance that quality software is delivered
- To make sure that requirements present in BRS (Business

Requirement Specification)and SRS (System Requirement Specification) are satisfied

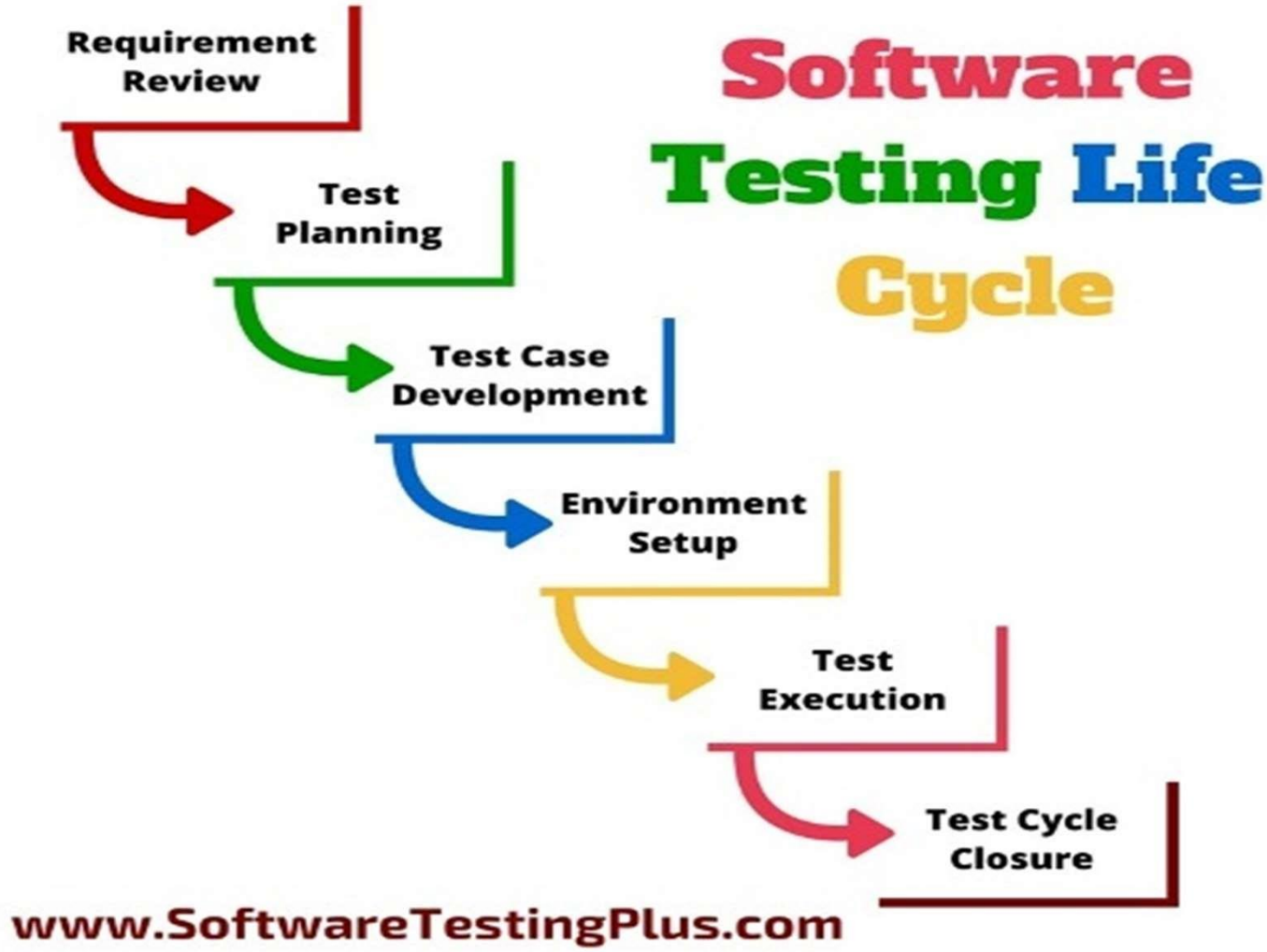
- Get confidence of customers by providing user friendly s/w that fulfills are requirements and expectations of customers

Software Testing Process

- Testing process is sequence of activities which help to certify the software system
- Testing process activities are:
  1. Requirement Analysis
  2. Test Planning
  3. Test Case Development
  4. Environment Set Up

5. Test Execution
6. Test Cycle Closure

- Each activity has defined entry and exit criteria
- Entry criteria : prerequisite conditions must be satisfied before testing begins
- Exit Criteria : conditions that must be satisfied before testing is concluded



- Requirement Analysis:
- All requirements of the system are gathered and studied to check weather they are testable or not.
- Depending upon the requirements ,type of testing (focus and priority) is decided
- At the end of this phase Requirement Traceability Matrix (RTM) is prepared

- Test Planning: (Test Strategy Phase)
- It is responsibility of test manager
- Manager finds out required efforts and estimated cost for the system under development
- At the end of this phase test plan is prepared

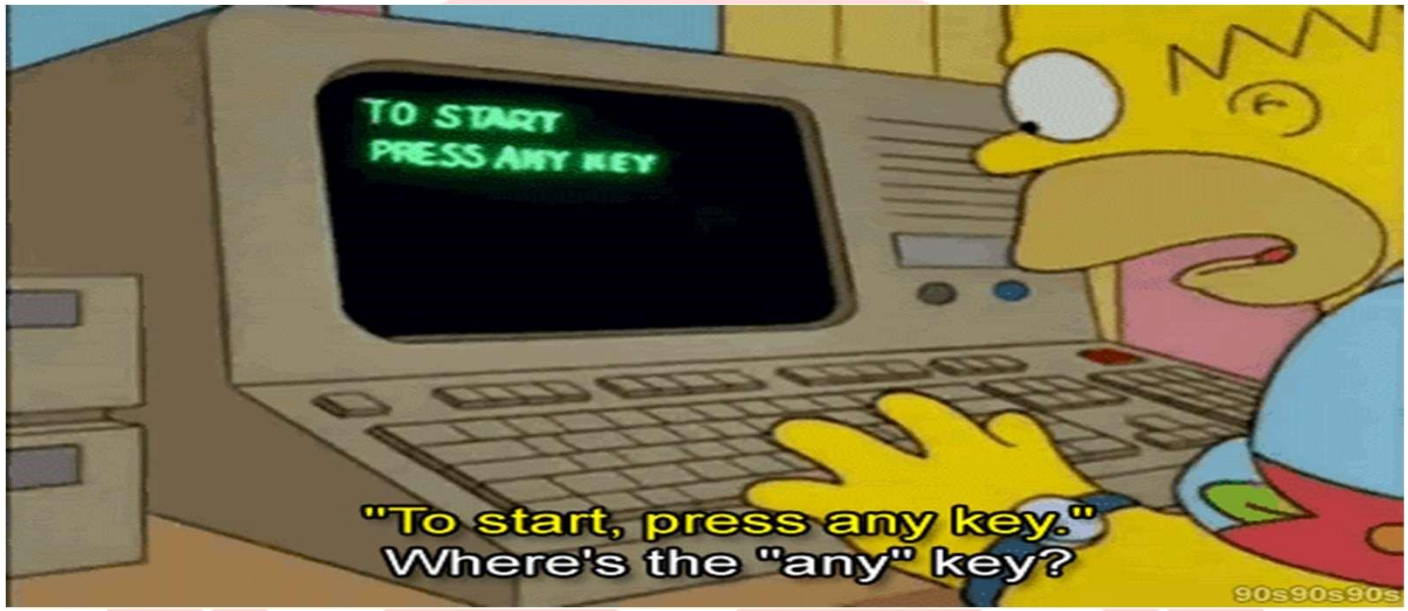
- Test Case Development:
- Test data is determined, reviewed and networked if necessary
- Includes – generation, verification and rework(as per requirements) of test cases and test scripts
- Outcomes : test cases and test scripts

- Test Environment Set Up
- Environment is set up and smoke test is done to check stability of the system(hardware doesn't catch fire)
- Outcomes: test environment and result of smoke test

- Test Execution:
- Testing is carried out by the team according to test plan and test cases
- Target is to find bugs
- Bugs are reported to development team for correction • After correction retest is performed for confirmation
- Test Cycle Closure:
- End of the testing process
- Completion criteria is determined depending upon time, test coverage, quality of system, cost of software, business objectives etc.
- Using these attributes test matrix is prepared
- Analysing available defect logs , test closure report is prepared

Terminologies :Failure ,Error ,Fault ,Defect ,Bug

- Failure :
- When a s/w is not performing required function and results are far from expected results then it is termed as s/w failure
- Failure leads a s/w to deliver services that are not in compliance with the specifications



- Reasons of Failure:
- Human Error: mistakes while interacting with s/w(wrong inputs)
- Environmental Conditions: impact hardware
- Users: users try harmful functions intending breaking the system
- System Usage: errors in the way system is used

- Ways to prevent failure:
- Identify and analyze errors and issues
- Adapt effective preventive techniques
- Ensure re-testing
- Revise and revisit specifications and requirements
- Error /Mistake:
- Mistakes made by s/w developer of programmer while preparing the code or design of s/w
- Errors are mainly a deviation that further changes functionality of s/w

- Reasons of errors and mistakes:
- Misconceptions or misunderstanding on the part of s/w
- Wrong login , loop, syntax
- Mistakes in program
- Confusion in understanding requirements

- Miscalculation of some values
- Misinterpretations and issues in coding
- Developer is unable to run/compile program
- Mistake in design or requirement
- Deviation/discrepancy between actual and expected results
- Ways to prevent errors:
- Improve s/w quality with code and system review
- Identify issues and prepare appropriate plans
- Verify fixes and validate their quality and accuracy
- Indulge in thorough testing
- Adopt effective s/w development methodologies

- Fault :
- Fault is introduced in s/w because of an error
- Fault is another discrepancy found by team of testers
- It is merely a manifestation of an erroring s/w
- Fault forces a system to act in an unanticipated manner

- Reasons for fault:
- Issue in code that causes failure
- Incorrect step, process, data definition
- Irregularity in s/w ,which makes s/w behave in incorrect way
- Ways to prevent faults:
- Implementing thorough code analysis
- Peer review
- Measure functional requirements
- Validate accuracy of software design and programming
- Defect :

- When actual result of the software deviates from the expected result then it is called as defect.
- Deviation or irregularity from the specifications mentioned in the ‘product functional specification document’ by client or any stakeholders
- Defects impact functionalities

Reasons of defects:

Errors impact software and causes malfunction

- Deviation in customer requirement
- Mistakes in external behaviour and internal structure of the system
- Wrong inputs
- Ways to prevent defects
- Peer review
- Adopt efficient programming techniques
- Regular code reviews

Bug :

Bugs are most integral part of software

- Impact performance as well as functionality
- Produces unexpected outputs
- Affect behavior of software

Reasons of bugs:

Errors or flaws in development environment of software

- Issue in software or hardware lead to malfunction
- Discrepancies in operating system used
- Sometimes incorrect code produced by compiler
- Ways to prevent bugs
- Adopting innovative and effective development methodologies
- Offering programming language support
- Analysing the code thoroughly
- Test-driven development

- A person makes an Error
- That creates a fault in software
- That can cause a failure in operation

- Error: An error is a human action that produces the incorrect result that results in a fault.
- Bug: The presence of error at the time of execution of the software.
- Fault: State of software caused by an error.
- Failure: Deviation of the software from its expected result. It is an event.
- Defect: A defect is an error or a bug, in the application which is created

Test Case

- It’s a group of positive and negative scenarios executed to test specific functionality of software
- Test case is a document which includes
  - – set of data
  - - preconditions
  - - expected results
  - - actual results for specific case
- Test case is starting point of test execution
- Actual result is compared with expected result

- If not same then bug/error is detected
- Sample test cases: for Log In page
- Test Case 1: check result for valid user name and valid password
- Test Case 2:check result for invalid user name and invalid password
- Test Case 3:check result for empty user name and empty password
- Test Case 4:check result for empty user name and fill password

- Test Case 5:check result for invalid user name and valid password
- Test Case 6:check result for valid user name and invalid password
- Test case Parameters

1. Test Case ID
2. Test Objectives
3. Preconditions
4. URL(optional)
5. Test Steps
6. Test Data
7. Expected Result

8. Actual Result
9. Pass or Fail

- Characteristics of Good Test Cases
- Test cases required being simple and transparent
- Create test case with end users mind
- Avoid test case repetition
- Do not consider functionality and features –use specification and documents
- Test results are repeatable and self-standing
- Peer review

Entry and Exit Criteria

- When to start and stop testing of software?
- Software testing process is not bound by any criteria.
- This may lead to negative consequences such as • Absence of goals and objectives .
- Unable to meet the deadline .
- Over budget.
- Inefficient testing due to inadequate requirements and understanding of products .
- QA team needs to understand the right time to start and terminate the testing .

- Testing process should be defined and planned along with entry and exit criteria.
- Entry Criteria:
- Condition or set of condition which should exist or meet in order to start a process.
- To begin an activity of software testing some condition and factors are defined and specified during planning phase.
- Important entry criteria:
- Availability of test environment supporting hardware, software, network configuration ,settings and tools.
- Availability of proper and adequate test data.

- Presence of proper testable data.
- Testers are trained and necessary resources are available.
- Requirements should be clearly defined and approved.
- Test design and documentation plan has to be ready.
- Exit Criteria:
- Exit criteria decides completion or termination of testing task.
- It is a set of conditions which decides completion of an activity or meeting of targets or goals.
- Exit criteria is defined and outlined during test planning phase.
- Exit criteria is more difficult to decide than entry criteria.
- Important Exit Criteria:

- All test cases have been executed.
- Sufficient coverage of requirements and functionalities has been achieved.
- All high priority bugs are fixed.
- High risk identified area is taken up and tested.
- Budget gets depleted(reduced).
- Deadlines reached.

•V Model (Verification and Validation)

- Verification: a process of estimating the intermediary work products of a software development life cycle to verify that we are in the correct track of creating the end user product.

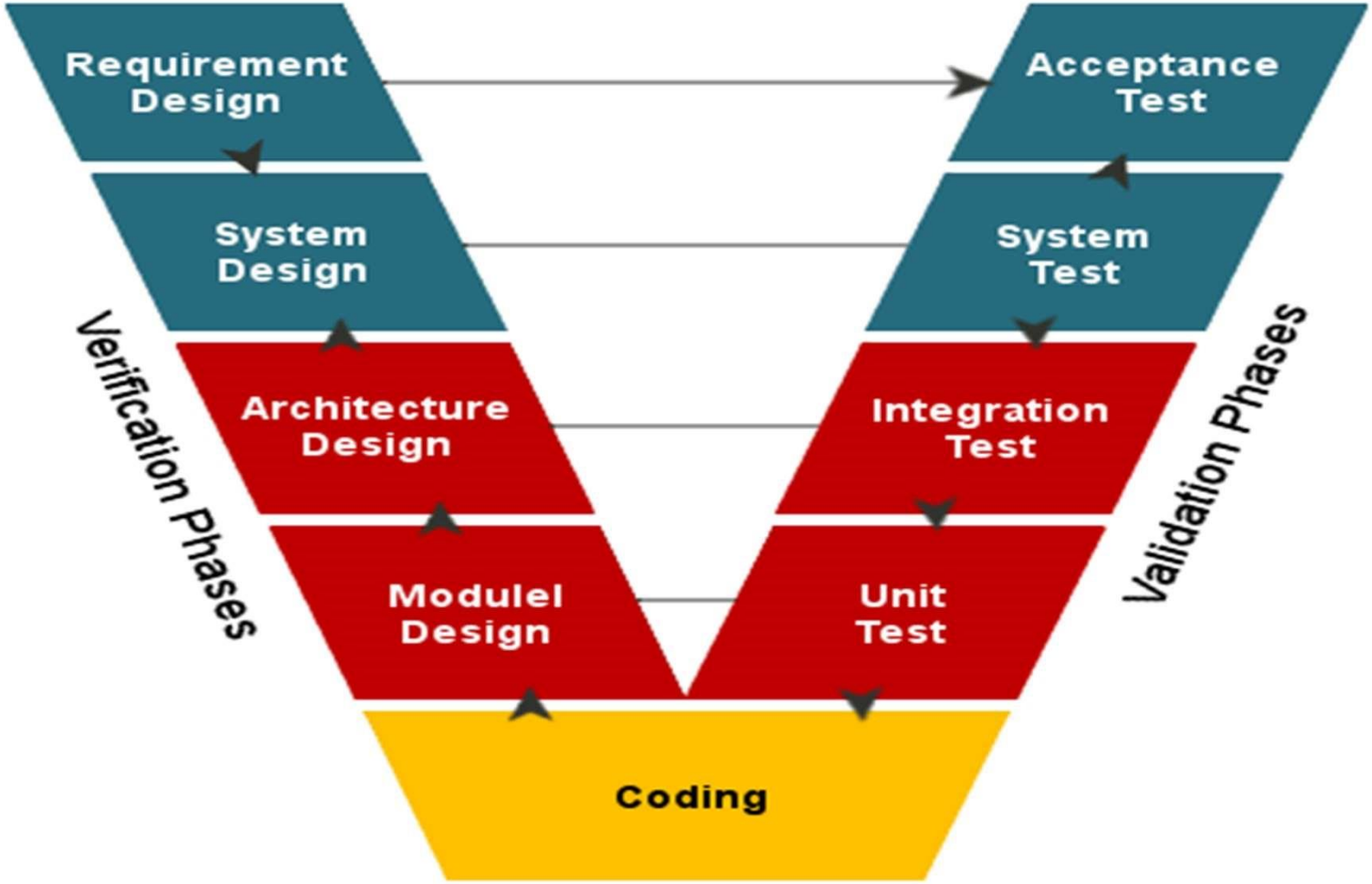
- Intermediary products:- generated during development phase design document, database tables ,ER diagrams, test cases , traceability matrix etc.
- Verification helps to find out high quality software.
- Focus is to make well-engineered and error free software.

- Validation: process of evaluating final product to ensure that the software meets all specified requirements.
- Validation is performed after verification phase.

S-16. S-18. 4 Marks

Q.1.5.1 Differentiate between verification and validation. (Ref. Sec. 1.5.1)

Parameter	Verification	Validation
Focus	Concentrates on right way to build a system.	Concentrates on output of build system.
Definition	Verification can be defined as it is a process of estimating the intermediary work products of a software development lifecycle to verify that we are in the correct track of creating the end user product.	Validation can be defined as it is the process of evaluating the final product to ensure that the software meets all the specified requirements.
Aim	The aim of Verification is to ensure that the product being develop is as per the requirements and design specifications.	The aim of Validation is to ensure that the product really meets up the user's requirements, and verify whether the specifications be correct in the first place.
Inclusion	Verification contains Reviews, Meetings and Inspections.	Validation contains testing such as black box testing, white box testing, gray box testing etc.
Performer	Verification is conducted by quality assurance team.	Validation is conducted by testing team.
Execution of code	Execution of code is not a part of Verification.	Execution of code is a part of Validation.
Explanation	Verification process gives explanation about whether the outputs are as per the inputs or not.	Validation process gives explanation about acceptance of software by the user or not.
Processes	Plans, Requirement Specifications, Design Specifications, Code, and Test Cases etc. are evaluated in verification.	Testing of actual software is done in validation.



- V- Model:
- It is an extension of waterfall model.
- The process steps are bent upwards after coding phase to for V shape.
- V-model demonstrates relationship between each phase of development life cycle and its associate phase of testing.

- Model is divided into 2 phases: • verification and
- validation.
- Verification:

i) Requirement analysis:

- First stage.
- Requirements of the system are collected by analyzing needs of users.
- Users are interviewed and requirements document is generated.
- Requirement document describes requirements like – functional ,interface ,performance, data ,security etc.
- Users do review the document.

- User acceptance tests are designed in this phase
- Ways of gathering requirements: interviewing, questionnaires, document analysis, observation, throw away prototype, use case, static and dynamic views with users.
- ii) System Design:

- Here system engineers analyze and understand business of proposed system by studying user requirement document.
- They figure out techniques for implementation.
- If any requirement is not feasible->user is informed and solution is found->document is edited .
- Software specification is generated.

- Which contains-general system organization, menu structure, data structures etc. and business scenarios, sample windows, reports for understanding.

- Entry diagram and data dictionary is also prepared here.
- The document for system testing is prepared here.

- iii) Architecture Design:

- This phase is of design of computer architecture and software architecture.
- This is high level design.
- It consists of -List of modules, brief functionality of each module, their interface relationship, dependencies, database tables, architecture diagrams, technology details etc.

- Integration testing design is carried out in this phase.

- iv) Modular Design:

- Also referred as low level design.
- System is divided into smaller units or modules.
- Each module is explained so that programmer can start coding directly.
- It contains: detailed functional logic of module in pseudo code, database tables(with elements ,their types and sizes),all interface details with references, all dependency issues, error message listing, complete input and output for a module,
- Unit test design is developed in this stage.

2) Validation:

- i) Unit Testing:
- UTP-unit testing plans are developed during module design phase.
- UTP's are executed to eliminate bugs at code level/unit level.
- Smallest entity existing independently is called as Unit(a program, module).
- Unit testing verifies that the smallest entity can function correctly when isolated from rest of the codes/units.
- ii)Integration Testing:

- Integration Test Plans are developed during the Architectural Design Phase.

- These tests verify units created and tested coexist and communicate properly.

- Test results are shared with customers team.
- iii) System Testing:

- System test plans are developed in this phase.
- These plans are composed by clients business team.

- Whole application is tested for its functionality, interdependency and communication.
- Subsets of system testing- load and performance testing, stress testing, regression testing, etc.
- iv)User Acceptance Testing:
- UAT- plans are developed during requirement analysis phase.

- Test plans are composed by business users.

- Tests are performed in user environment that resembles production environment using realistic data.

- UAT checks weather system is ready for use in real time.
- Advantages of V-model:

- Simple and easy to use.
- Planning and test design occurs prior to coding, it avoids wastage of time.
- Better success than waterfall model.
- Diagnosis of defects is done in early stages
- Avoids downward flow of defects.

- This is good for small projects where requirements are well understood.

•Disadvantages of V-model:

- This model is very rigid and less flexible
- Development of s/w is done in implementation phase
- Hence no early prototypes are produced
- In case of midway design changes then we need to update test document as well

Quality Assurance:

- SQA-Software Quality Assurance.

- Group of activities to guarantee quality of software.
- SQA is ongoing activity in SDLC.
- Most of the s/w development industries adapt SQA.
- SQA incorporates software testing methodologies.

- SQA has following tasks:

oTo identify weaknesses in the process oCorrect those weaknesses to constantly improve the process

- SQA includes following activities:
  - oProcess definition and accomplishment
  - oAuditing oGuidance
    - Processes could be :
    - Software Development method.
    - Project Management.
    - Configuration Management.
    - Requirements Development/Management.
    - Estimation.
    - Software Design.
    - Testing etc.

- Components of SQA system ….(6 Classes):

1. Pre-project component:
   - Resources, schedule and budget are taken into account to define project commitments.
   - Improvement and quality plans are determined.

2. Components of project life cycle activities assessment:
   - Project life cycle is composed of 2 stages:
     A. Development life cycle stage
        - Components identify design and programming flaws.

- Components are divided in sub classes – reviews, expert opinion, software testing.

B. Operation maintenance stage

- Includes specialized maintenance components as well as development life cycle components.
- Applied mainly for functionality to improve maintenance tasks.

3. Components of infrastructure defect avoidance and improvement:

- Aim is to delete or minimize rate of errors. 4. Components of software quality management:

- Deals with number of goals- development, maintenance, managerial support to avoid budget failure.

5. Components of standardization, certification and SQA system Evaluation:
   - Adapt international values.
   - 2 main groups of standards-quality management , project process standard.
6. Organizing for SQA- the human component:
   - Managers, testing professionals, SQA committee members, etc.

- Aim: - to make first move and maintain implementation of SQA components, detect deviation, advice improvements.
- Pre project SQA
- Components helping groundwork before starting a project.

1. Deal Review:
   - Deal/agreement with client.
   - Development company is loyal with approved functional specification, budget, schedule.

   - Contract review – detailed examination of project tender draft/contract draft.

     - Contract activities:

- Clarification of client's requirements.
- Review of schedule and requirement estimate.
- Assessment of professional resources.
- Evaluation of customer's activity.
- Evaluation of development risk.

2. Development and Quality Plans

- After signing contract development plan is prepared.
- Plan preparation might take time.
- Plans are revised to reflect changes.

i)The main problems treated in the project development plan are:

a. Schedules.
b. Manpower and h/w resources
c. Risk evaluation
d. Organizational problems related to team members or partnership.
e. Project methodologies and development tools.
f. Software reuse plans.

ii) The main issues treated in the project's quality plan:

- Quality aims.
- Criteria for initial and final project stage.

- List of reviews, tests, schedules and validation activities.

iii) Software metrics (standards) can be classified into 3 categories:

a. Product metrics-volume, complexity, design features, performance, quality level.

b. Process metrics-development and maintenance activity.

c. Project metrics-project characteristics and implementation.

iv) Software quality metrics can be further divided into 3 categories:

a. Mean time to failure- time difference between failures (used for safety critical systems such as airline control system, weapons etc.

b. Defect density- errors related to software size (code quality per unit).

c. Customer problems-defect and non-defect oriented problems.

v)Client satisfaction- measures through 5 points-based on them major or minor changes are done:

1. Very satisfied.
2. Satisfied.
3. Neutral.
4. Dissatisfied.

5. Very satisfied.

- In-process Quality Metrics

- Tracking fault during arrival of machine.
- This metrics includes
- Defect density during machine testing.
- Defect arrival method during machine testing.
- Phase based defect removal pattern.
- Defect elimination efficiency.
- SQA Activities:

1) Process definition and implementation.
2) Auditing.
3) Training.

• SQA Processes:

1) S/W Development Methodology.
2) Project Management.
3) Configuration Management.
4) Requirement Development.
5) Estimation.
6) Software Design.
7) Testing

- Once the process is defined and implemented, QA has following responsibilities:
- Identifying weaknesses in the process.
- Correcting weaknesses.
- Quality standards used are- CMMI, Six Sigma, ISO 9000.
- Benefits of SQA:

1. Monitoring and improving the process

2. Make sure that standard and procedures are followed

3. Preventing quality problems

• Phases of SQA

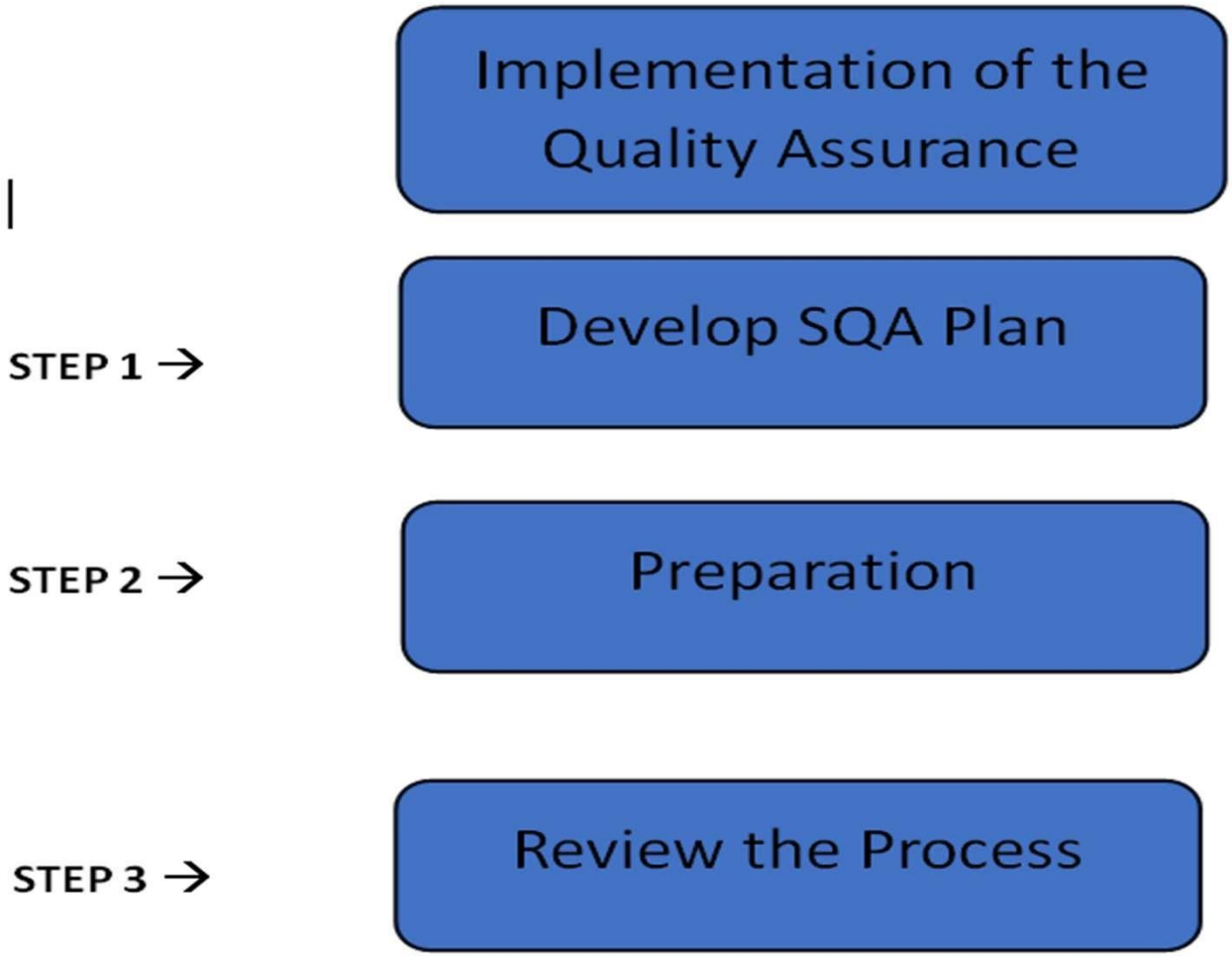
A) Test Management Reviews and Audit:

oFocuses on software methods.

oActivities planned to guarantee that project manager follows regular process which is already defined in system.

oAudit: assessment of the work to check weather the regular process was carried out or not.

B. Implementation of the Quality Assurance



Step 1 – develop SQA plan (3 Slides)

- As testing needs test plans SQA also needs SQA plans.
- They plan process and measure to make sure that products are manufactured.
- SQA plans are prepared by project manager during project planning.

- SQA audit is scheduled periodically.

-Responsibilities of test manager:

- Identify the role and responsibilities of SQA team -

Every team member is responsible for quality of their work.

-SQA team is a group of persons who play major role in the project.

-No business can run without QA.

-Test manager makes clear the responsibility of each SQA team member.

-Quality of the project is reviewed and evaluated. -

Management board and project team coordinate to

access requirements and engage in review and status meeting.

-Design is tracked and metrics are collected to monitor project quality.

- List of the work products that SQA auditors will review and audit:
- Test manager will list out the work products.
- He defines which facility or equipment SQA auditor can access to perform SQA tasks such as evaluation and audit.
- Create the schedule to perform SQA tasks:
- Test manager creates schedule of SQA.

- This schedule is driven by project development schedule.
- SQA tasks are performed in relationship with s/w development activities.

Step 2 – define standards/methodology

- Define the policies and procedures to prevent defects from management process.
- Document the policies and procedures.
- Inform and train the staff to use these processes.
- Step 3 – Review the Process
- Review project activities to verify compliance with the defined management process.
- 5 SQA reviews:
- Review Project Planning.

- Review Software Requirement Analysis.
- Review Test Design.
- Review Before Release.
- Review Project Closing .

Quality Evaluation Standards

- To evaluate software products AQC lab was established.
- They evaluate quality of s/w against ISO/IEC 25000 standard.
- Many more s/w quality certifications exist.

Quality Control (2 slides)

- SQC(Software Quality Control) is a set of activities to guarantee that the quality in software products is maintained.
- These activities determine errors in products.
- s/w quality attributes like maintainability, usability, reliability cannot be accurately specified and measured in s/w quality management.
- it early stages it is difficult.

- it may happen s/w confirms its requirement but users don't get quality as per expectation.

3 categories of s/w quality management:

- Quality Assurance
- Development of organizational procedures and standards that leads to high quality software.
- Quality Planning
- Selection of appropriate procedures and standards from project framework and user for a specific s/w project.
- Quality Control
- It defines the process ensuring that s/w development follows the quality procedures and standards.

- Quality management provides an independent test on s/w and s/w development.
- It ensures deliverables follow organizational standards and goals.

Q.1.7.2 Differentiate between Quality Assurance ...

Parameter	Quality Assurance	Quality Control
Definition	QA is a set of activities for ensuring quality in the processes by which products are developed.	QC is a set of activities for ensuring quality in products. The activities focus on identifying defects in the actual products produced.
Focus on	QA aims to prevent defects with a focus on the process used to make the product. It is a proactive quality process.	QC aims to identify (and correct) defects in the finished product. Quality control, therefore, is a reactive process.
Goal	The goal of QA is to improve development and test processes so that defects do not arise when the product is being developed.	The goal of QC is to identify defects after a product is developed and before it's released.
How	Establish a good quality management system and the assessment of its adequacy. Periodic conformance audits of the operations of the system.	Finding and eliminating sources of quality-problems through tools & equipment so that customer's requirements are continually meet.
What	Prevention of quality problems through planned and systematic activities including documentation.	The activities or techniques used to achieve and maintain the product quality, process and service.
Responsibility	Everyone on the team involved in developing the product is responsible for quality assurance.	Quality control is usually the responsibility of a specific team that tests the product for defects.
Example	Verification is an example of QA.	Validation/Software Testing is an example of QC.
Statistical Techniques	Statistical Tools & Techniques can be applied in both QA and QC. When they are applied to processes (process inputs and operational parameters), they are called Statistical Process Control (SPC); and it becomes the part of QA.	When statistical tools & techniques are applied to finished products (process outputs), they are called as Statistical Quality Control (SQC) and comes under QC.

Methods of Software Testing:

- Fundamental methods:
  - Black Box Testing.
  - White Box Testing.
- Other methods
  - Static testing.
  - Dynamic Testing.

- GUI Testing.
- Static Testing (5 slides)
- Tester examines s/w code and documentation
- In this type code is not executed.
- Testing is done manually or by using set of tools.
- Here requirement document and design document is checked and review comments are given on the work document.
- Static testing finds out: errors, code flaws, potentially malicious code.

- Static testing starts earlier in development of SDLC .
- Also called as verification testing.

Static testing is done on:

- Requirement specification
- Design document
- Source code
- Test plans
- Test scripts
- Test cases
- Web page contents

- The Static Test Techniques:
- Inspection
- Finding defects.
- Code walkthroughs are conducted by moderators.
- Checklist is prepared to review.
- Walkthrough
- Meeting is lead by author to explain the product.
- Participants can ask questions and make notes.
- Technical reviews
- Technical round of review to check coed for technical satisfaction.
- Test plans, test strategies and test scripts are reviewed here.
- Informal reviews
- Document is reviewed informally. • Informal comments are provided.

Advantages of Static Testing

- Helps in identifying flaws in code.
- Conducted by trained developers with good knowledge.
- Fast and easy to find and fix errors.
- With automated tools becomes more faster.
- Errors are found in early stages which reduces cost of fixing them.

Disadvantages of Static Testing

- Needs more time when its done manually.
- Automated tools work with limited programming languages.
- Automated tools may provide false positive or false negative results.
- Automated tools only scan the result.

- Cant pinpoint the weak points.

Dynamic Testing (4 slides)

- It’s done when code is in operation.
- Performed in runtime environment.
- Input and output are checked and compared
- We can observe functional behavior, monitor system memory, CPU response time, performance of the system.
- Dynamic Testing is known as Validation Testing.

- Two Types:
- Functional Testing
- Non-Functional Testing

Types of dynamic testing techniques:

- Unit Testing
- Testing of individual modules by developers.
- Source code is tested.
- Integration Testing
- Testing of interface between different modules.
- Later they are joined.

- System Testing
- Testing performed on the system as whole.
- Acceptance Testing
- Testing at users end from user point of view.

Advantages of Dynamic Testing:

- Helps in identifying weak areas in run time environment.
- Allows validation of static code.
- Can be applied to any application.

Disadvantages of Dynamic Testing:

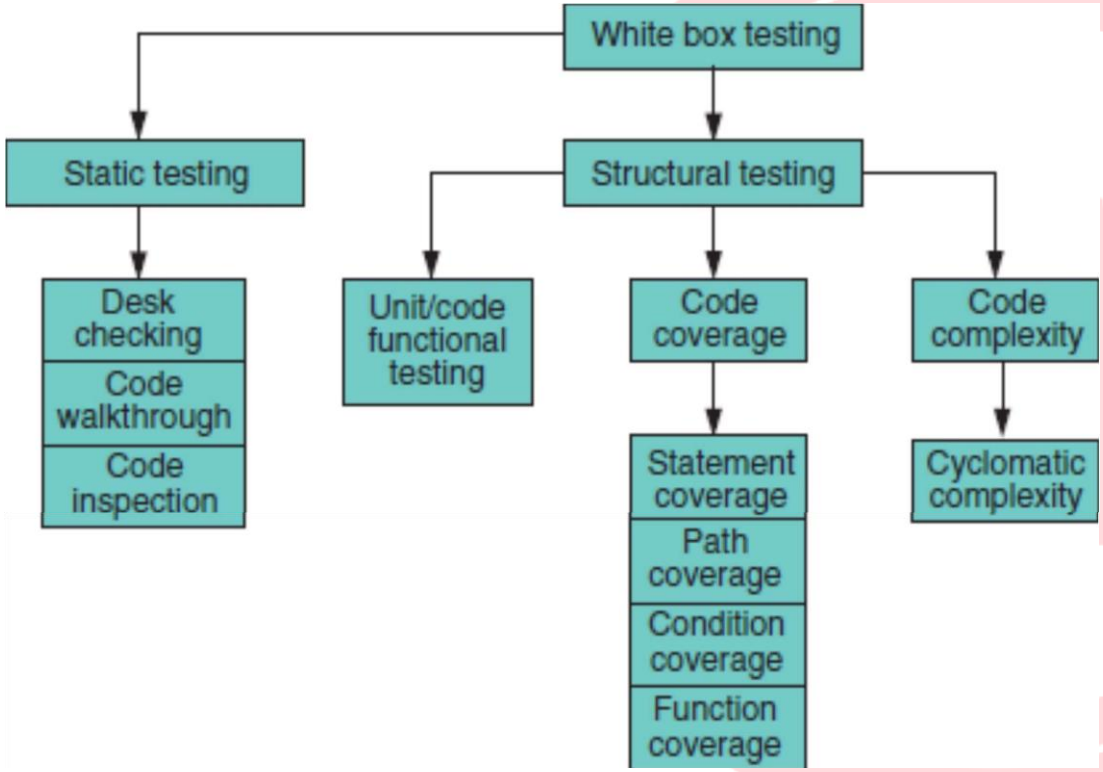
- Automated tools may give false security that everything is checked.
- False positive or false negative.
- Not easy to find trained professional for dynamic testing.
- Difficult to check vulnerability in the code
- Takes longer to fix problems.
- Costly to fix errors.

Parameter	Static Testing	Dynamic Testing
Execution	Static testing is white box testing which is done at early stage if development life cycle. It is more cost effective than dynamic testing	Dynamic testing on the other hand is done at the later stage of development lifecycle.
Statement coverage	Static testing has more statement coverage than dynamic testing in shorter time.	Dynamic testing has less statement stage because it is covers limited area of code.
When to do	It is done before code deployment.	It is done after code deployment.
Performing stage	It is performed in Verification Stage.	It is done in Validation Stage.
Execution of code	This type of testing is done without the execution of code.	This type of execution is done with the execution of code.
Gives	Static testing gives assessment of code as well as documentation.	Dynamic testing gives bottlenecks of the software system.
Process	In static testing techniques a checklist is prepared for testing process	In dynamic testing technique the test cases are executed.
Includes	Static testing methods include Walkthroughs, code review.	Dynamic testing involves functional and non-functional testing

- Box Approach

1)White-Box Testing

2)Black-Box Testing



- 1.White-Box Testing (5 Slides)

- Internal coding and infrastructure of software product is tested.
- Based on analyzing the internal implementation of system.
- Concentrates on testing flow of inputs and outputs through s/w. -

Also called as

- Clear Box Testing
- Open Box Testing
- Structural Testing
- Transparent Box Testing

- Code-Based Testing
- Glass Box Testing

- Methods Used:
  - Statement Coverage: - testing all programming statements using minimum number of tests.
  - Branch Coverage: - all branch conditions are tested at least once.
  - Path Coverage: - each statement and branch is tested at least once.
  - Other coverages condition coverage, multiple condition coverage, path coverage, function coverage etc.
- 2 steps in White Box Testing:

1. Understand the Source Code:

- Code is understood by the testers.
- Tester must have high knowledge of programming languages.
- Tester should be able to understand security problem(hacker attack)

2. Create test cases and execute:

- Check flow of the control and structure.
- Small test cases generated by testers for every process or group of processes.
- Tester should have high knowledge of source code.

- White box testing can be performed by developers as they have high knowledge of programming languages .
  - Advantages of White Box Testing:
  - It tests all paths in the system so it is a thorough testing.
  - White box testing can be applied at earlier stages – no need to wait for GUI.
  - As tester has knowledge of programming, he can find out type of input suitable for testing.
  - White Box Testing provides code optimization by finding hidden bugs.

- Automation can be done easily.
- Each statement is tested at least once.
- Can be started early – when the first module is created.

- Disadvantages of White Box Testing Complex and expensive process.

- For larger application it becomes impossible to perform exhaustive testing.
- It works on existing software so missing functionality may not be discovered.
- All inputs are to be considered.

- If developer performs testing there are less chances to find development errors.
- Expert persons are needed.

Inspections:

- It is a formal type of code review.
- It is a static testing used to avoid defects in next stage.
- Documents are read before meeting to give assurance of consistency.

- They are conducted by trained moderators who have not developed that code.
- It is very effective test method.
- Time consuming but detects up to 80% defects.
- Roles of inspections:
  - Author
    - Person who created the work product that is going to be inspected
  - Moderator
    - Leader of inspection.
    - Plans inspection and manages it.
  - Reader

- Reads the document.
- Single document at a time.
- Remaining inspectors find out defects.

• Recorder

- Person responsible for recording defects.

• Inspector

- Observes the work product to find possible defects.

Walkthroughs:

- It’s a review meeting.
- Not similar to inspection.

- Its not a formal process.
- Started by author of code.
- He reads document or code and writes down defects or suggestions.
- No need of moderators.
- Open ended discussion- list of observations is not created.
- Objectives of walkthrough:

- Understand and learn the development of s/w product.
- Detect defects.

- Explain information present in document.
- Verify and discuss validity of proposed system.
- Report suggestions.

Parameters	Inspection	Walkthrough
Formal or informal testing	It is formal type of testing.	It is informal type of testing.
Started by	It is started by the project team.	It is started by the author of code or document.
Important factor	Inspection includes meeting which is planned with all the members included in project.	Walkthrough includes unplanned meeting.
Process	In inspection, reader reads code of software product. Every member examines it and find out defects.	In walkthrough, defects are collected by author who reads the product code and his team members detect the defects and give suggestions.
Suggestions recorded by	In inspection, recorder records the defects.	In walkthrough author makes a note of defects and suggestions which is given by the team members.
Moderator Requirement	Moderator makes sure that the discussion goes in fruitful direction.	It is informal so moderator is not involved in this process of testing.

Technical Reviews:

- Process performed to give assurance of s/w quality.
- Aims:
  - To Find errors in function, logic or coding.
  - To check s/w fulfills requirements.
  - Assurance of design as per customers' requirements.
  - To create s/w in uniform order. To create manageable project.
- Technical review can be provided during training to junior developers to study different ways of s/w analysis, design, development.
- Technical reviews are class of reviews containing: walkthroughs, inspections, etc.
- Technical reviews are performed as a meeting.

- If planned, controlled and attended in correct way it becomes successful.

Functional Testing:

- System is tested against functional requirements/specifications.
- Testing done based on inputs and outputs.
- Result oriented testing.
- Simulates actual system usages.
- Here black box testing is used where internal logic is not known to tester.
- Used during system testing and acceptance testing.
- Steps involved:
  - Identify functions that s/w is expected to perform.
  - Create input based on function's specification.

- Determine out put based on function's specification.
- Execute the test case.
- Compare actual and expected results.

Code Coverage Testing (2 slides)

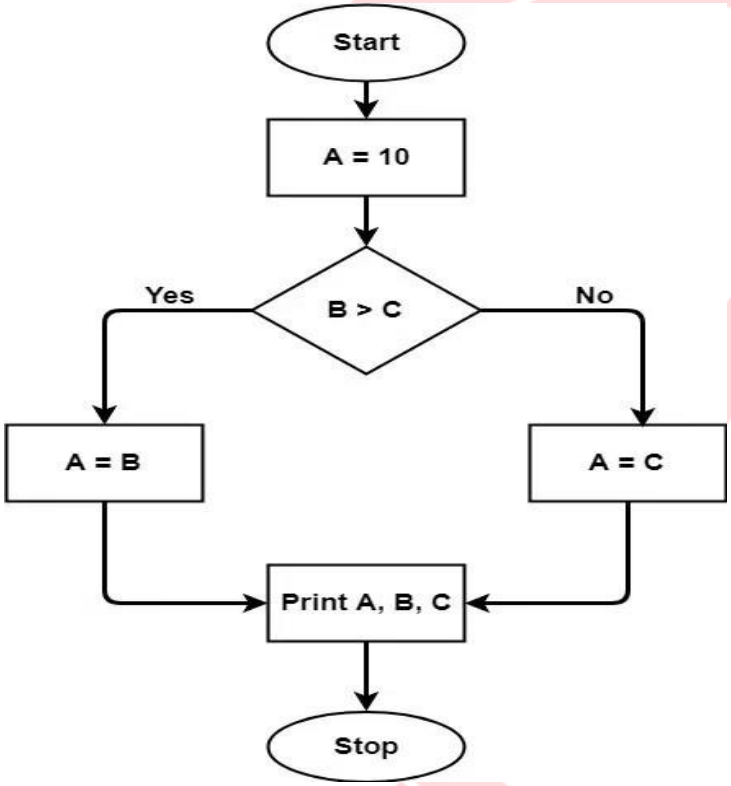
- Test Coverage: degree to which source code of a program is executed when particular test suit runs.
- Program with high test coverage measures as percentage.
- It has more source code executed during testing.
- It has lower chance of containing undetected s/w bugs.
- To measure the percentage of the code exercised by test suit different coverage criteria are used.
- Coverage Criteria: rules or requirements which a test suite needs to satisfy.

- Basic Coverage Criteria:
- Function Coverage: has each function in the program been called?
- Statement Coverage: has each statement in the program been executed?
- Edge Coverage: has every edge in the control flow graph been executed?
- Branch Coverage: has each branch in the control flow been executed?
- Condition Coverage(predicate coverage): has each Boolean subexpression evaluated both to true and false?

- Code Complexity Testing (3 slides)

- Cyclomatic Complexity: used to determine complexity of piece of a code or functionality.
- 3 questions technique developed by MaCabe:
- Is the feature/program testable?
- Is the feature/program is understood by everyone?
- Is the feature/program reliable enough?
- If value of cyclomatic complexity is bigger then piece of functionality is more complex.
- Hence it needs in-depth testing.
- How To Calculate CC:
- CC revolves around 2 concepts

- Nodes : statements in the program



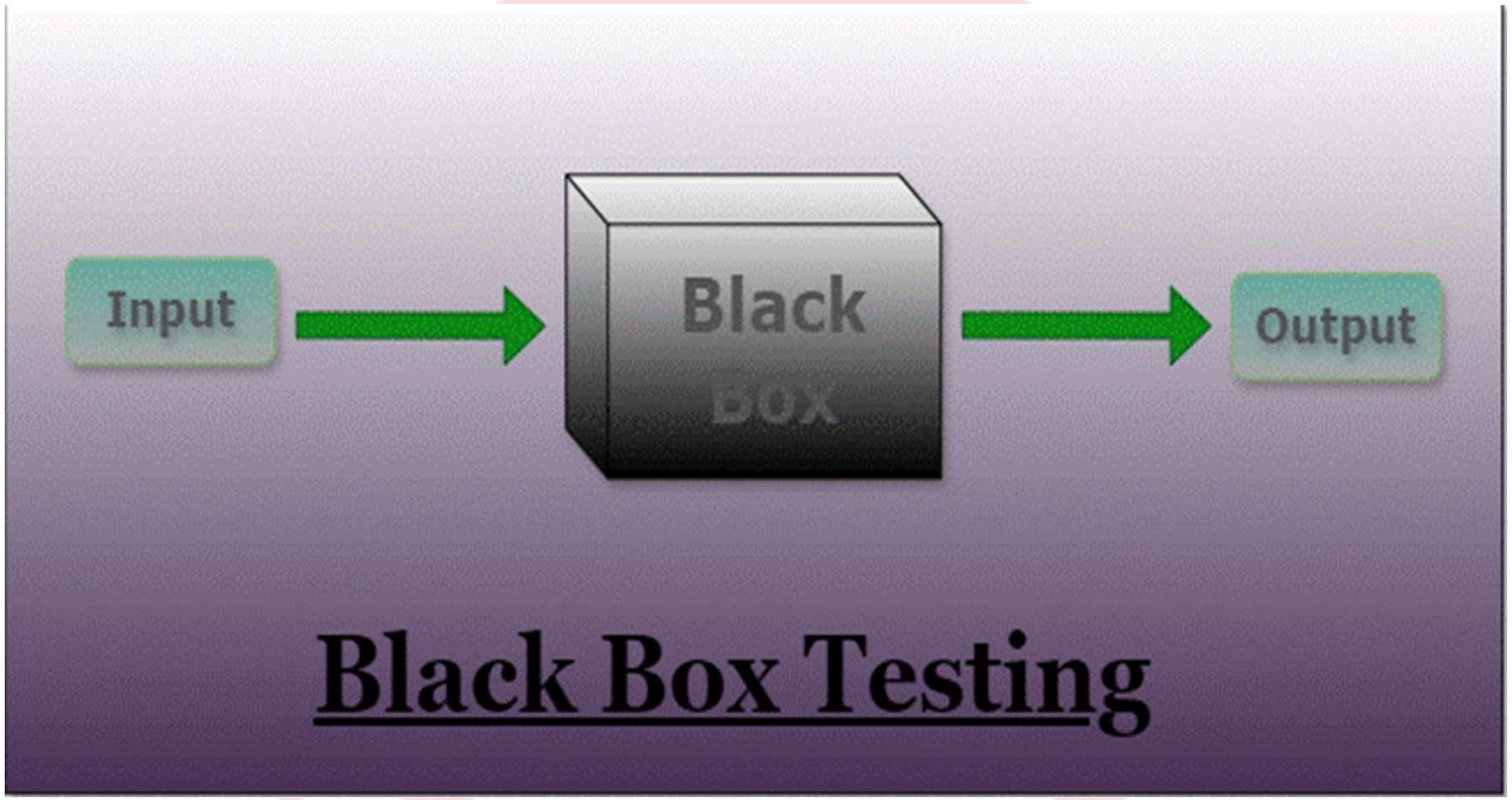
- Edges : control paths from one statement to another
- CC=E~N+2
- E – number of edges
- N – no of nodes

The graph shows seven shapes (nodes), and seven lines (edges), hence cyclomatic complexity is 7-7+2 = 2.

Testers can use this number to determine level of their testing. There are 2 levels of testing 1. Length testing

2. Breadth testing

- Black Box Testing



- Black-box testing is a type of software testing in which the tester is not concerned with the software's internal knowledge or

implementation details but rather focuses on validating the functionality based on the provided specifications or requirements.

- Test cases are built based on what application is supposed to do.
- Also known as behavioural testing or specification testing.
- Focus is on input and output.
- No compulsion of knowledge of programming language.
- Can be applied in each stage of s/w testing.
- Steps used in Black Box Testing:
- Requirements/specifications are analyzed.
- Valid test cases are selected.

- Invalid data foe negative test cases is selected.
- Tester states expected output.
- Test cases are generated and run on selected test data.
- Comparison between actual and expected result is done.
- If not same – defect is detected.
- Defect reported to developer who will fix the defect.
- Application is retested.
- Methods used for developing test cases in Black Box Testing:
- Boundary Value Analysis (BVA):
- It tests the boundaries of the input values
- It is Common technique in BBT
- Input values are selected as :-

- Minimum
- Just above minimum
- Normal
- Maximum
- Just below maximum

- Equivalent Class Partitioning:
- Divides inputs into classes.
- Tests application thoroughly.
- Avoids redundancy of inputs.
- Can be applied at all levels of testing.

- Always performed before BVA.

- Cause-Effect Graphing:
- Graphical representation of output and factors affecting output.
- Helps to identify causes of defects.
- Tests only external behavior.
- Error-Guessing:
- Based on testers experience and skill.
- Finding out defects which may not be determined by formal techniques.
- Done after formal testing.
- Completely depends upon knowledge and skill of tester.

•Types OF Black Box Testing:

- Functional testing
- Checks every function in the s/w.
- Does not focus on the code.
- Functionalities are verified with the help of test data.
- If there is difference between actual and expected results BUG is detected.
- Functional testing is done using Requirement Specification Document.
- Non-functional testing
- Checks non-functional requirements such as:
  performance, scalability, usability, security etc.
- Regression testing

- It checks changes made in code because of errors does not affect existing working functionality.
- Already executed test cases are used to assure old functionalities work well after changes.
- This testing guarantees that there is no interruption in working of existing functionalities.

•Advantages of Black Box Testing:

- Efficient for large systems.
- Identifies contradiction in functional specification.
- Detailed knowledge of system is not required.
- Testers and developers work independently.

- Disadvantages of Black Box Testing:
- Difficult to find out all possible inputs for test cases in limited time.
- Not useful for complex code.
- Test cases can not be designed without knowledge of functional specifications.
- It is not possible to detect the root cause of a defect.
- The intermediate stages of the software go untested.

- Requirement Based Testing
- This technique is all about validating that requirements are:
- Complete
- Consistent
- Unambiguous
- Logically connected
- This technique revolves around requirements.
- Aim is defect prevention than defect detection.

- Activities in Requirement Based Testing:
  - Define test completion criteria: test coverage is

100%

- Design test cases: as per requirement specification
- Build test cases: join logical parts together to form test cases
- Execute test cases: execute test cases to evaluate results
- Verify test results: check if any deviation
- Verify test coverage: check for functional test coverage
- Manage test library: keeping record of passed or failed test cases

•s/w projects fail because:

- incomplete requirements or specifications.

- frequent changes in requirement or specifications.
- lack of user input to requirements.

•testing solves these issues as follows:

- Requirement based testing is started in early stages where error correction is easier.
- As it starts at requirement phase- bugs have roots here.
- Focus is on improvement of quality of requirements.
- Boundary Value Analysis
- Its a black box testing method
- Checks bugs at boundary of input

- Checks both valid and invalid boundary partitions
- BVA is negative or stress testing
- Example: Consider a system that accepts ages from 18 to 56.

- **Boundary Value Analysis(Age accepts 18 to 56)**		
- Invalid - (min-1)	- Valid - (min, min + 1, nominal, max – 1, max)	- Invalid - (max + 1)
- 17	- 18, 19, 37, 55, 56	- 57

- Valid Test cases: Valid test cases for the above can be any value entered greater than 17 and less than 57.

- Enter the value- 18. • Enter the value- 19. • Enter the value- 37. • Enter the value- 55.
- Enter the value- 56.
- Invalid Test cases: When any value less than 18 and greater than 56 is entered.
- Enter the value- 17.
- Enter the value- 57.
- Equivalent Partitioning
- This is a black box method.
- Can be used at every level of testing.

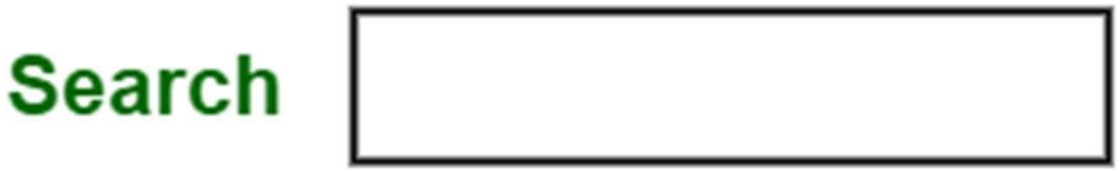
- Separate test conditions for different classes.
- Can be used when range of input is available.

- Let us consider an example of an online shopping site. In this site, each of products has a specific product ID and product name. We can search for product either by using name of product or by product ID. Here, we consider search field that accepts only valid product ID or product name.
- Let us consider a set of products with product IDs and users wants to search for Mobiles. Below is a table of some products with their product Id.

- If the product ID entered by user is invalid application will redirect customer or user to page. If product ID entered by user is valid for mobile, then equivalence partitioning will show a valid product ID.

Product	Product ID
Mobiles	45
Laptops	54
Pen Drives	67
Keyboard	76
Headphones	34

then error i.e. 45 method

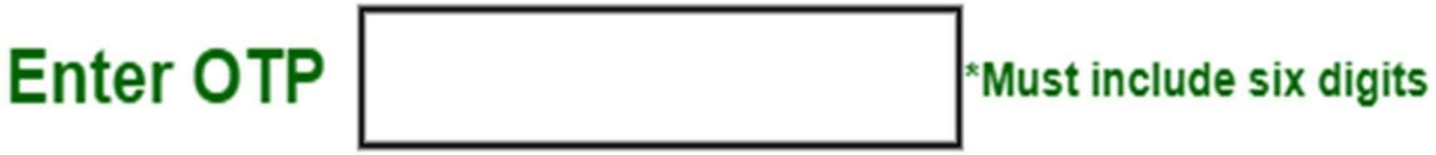


Equivalence Partioning		
Invalid	Invalid	Valid
77	84	45

- Example-3 :

Let us consider an example of software application. There is function of software application that accepts only particular number of digits, not even greater or less than that particular number.

- Consider an OTP number that contains only 6 digit number, greater and even less than six digits will not be accepted, and the application will redirect customer or user to error page. If password entered by user is less or more than six characters, that equivalence partitioning method will show an invalid OTP. If password entered is exactly six characters, then equivalence partitioning method will show valid OTP.



Equivalence Partioning			
Invalid	Invalid	Valid	Valid
Digits>=7	Digits<=5	Digits=6	Digits=6
67545678	9754	654757	213309

- Example 1: Equivalence and Boundary Value
- Let's consider the behaviour of Order Pizza Text Box Below.
- Pizza values 1 to 10 is considered valid. A success message is shown.
- While value 11 to 99 are considered invalid for order and an error message will appear, "Only 10 Pizza can be ordered".
- Order Pizza:
- Here is the test condition.
- Any Number greater than 10 entered in the Order Pizza field(let say 11) is considered invalid.
- Any Number less than 1 that is 0 or below, then it is considered invalid.

- Numbers 1 to 10 are considered valid
- Any 3 Digit Number say -100 is invalid.
- We cannot test all the possible values because if done, the number of test cases will be more than 100.
- To address this problem, we use equivalence partitioning hypothesis where we divide the possible values of tickets into groups or sets as shown below where the system behavior can be considered the same.

Parameter	Black Box Testing	White Box Testing
Applicable levels	It is mainly applicable at higher levels of testing : Acceptance and system testing.	It is mainly applicable at lower levels of testing i.e. unit and integration testing.
Base for testing	Requirement specification is the basis for Black Box Testing.	Detail design is the basis for White Box Testing.
Performed by	It is carried out by software testers.	It is carried out by software developers.
Aim	It aims at what functionality is system performing.	It aims at how system is performing its functionality.
Requirements	Programming knowledge and implementation knowledge is not necessary to carry out BBT.	Programming knowledge and implementation knowledge is necessary to carry out WBT.
Cost	It is less expensive than WBT.	It is more expressive than BBT
Focus on	In black box testing, we do not test the code and focuses on the functionality of software product.	In white box testing, we test the code and focuses on the security of software product and flow of control.
Facility provided	Black box testing provides facility of verifying communication among different modules present in software product.	White box testing does not provide facility of verifying communication among different modules present in software product.