Software Testing

EDITION : 2019

Sub Code : 22518

MSBTE I SCHEME PATTERN
T. Y. DIPLOMA SEM V
COMPUTER ENGG. GROUP
(CO/CM/CW)

INCLUDES-I SCHEME PATTERN
SAMPLE PAPERS

- CHAPTERWISE SOLVED MSBTE QUESTIONS SUMMER 2015 to SUMMER 2019

A. A. Puntambekar

As per Revised Syllabus of

MSBTE - I SCHEME

Software Testing

T. Y. Diploma (Semester - V)
Computer Engineering Group (CO/CM/CW)

Mrs. Anuradha A. Puntambekar

M.E. (Computer)
Formerly Assistant Professor in
P.E.S. Modern College of Engineering,
Pune

Vaishali Rane

B.E., M.E. (CMPN)
HOD (Computer Engineering),
Thakur Polytechnic
Thakur Complex, Kandivali E, Mumbai-401101

Nilesh J. Vispute

M.Tech. (Computer Science)
Senior Lecturer,
Pravin Rohidas Patil Polytechnic
Mira Bhayandar, Mumbai 401105

TECHNICAL PUBLICATIONS
An Up-Thrust for Knowledge

Website : www.technicalpublications.org
https://www.facebook.com/technicalpublications

Software Testing

T. Y. Diploma (Semester - V)
Computer Engineering Group (CO/CM/CW)

First Edition : June 2019

Published by :

TECHNICAL PUBLICATIONS™ Amit Residency, Office No.1, 412, Shaniwar Peth, Pune - 411030, M.S. INDIA
Ph.: +91-020-24495496/97, Telefax : +91-020-24495497
Email : sales@technicalpublications.org Website : www.technicalpublications.org

Printer :

Yogiraj Printers & Binders
Sr.No. 10\1A,
Ghule Industrial Estate, Nanded Village Road,
Tal-Haveli, Dist-Pune - 411041.

Price : ₹ 85/-

ISBN 978-93-89180-04-6

MSBTE I

PREFACE

The importance of **Software Testing** is well known in various engineering fields. Overwhelming respondse to our books on various subjects inspired us to write this book. The book is structured to cover the key aspects of the subject **Software Testing.**

The book uses plain, lucid language to explain fundamentals of this subject. The book provides logical method of explaining various complicated concepts and stepwise methods to explain the important topics. Each chapter is well supported with necessary illustrations, practical examples and solved problems. All chapters in this book are arranged in a proper sequence that permits each topic to build upon earlier studies. All care has been taken to make students comfortable in understanding the basic concepts of this subject.

Representative questions have been added at the end of each section to help the students in picking important points from that section.

The book not only covers the entire scope of the subject but explains the philosophy of the subject. This makes the understanding of this subject more clear and makes it more interesting. The book will be very useful not only to the students but also to the subject teachers. The students have to omit nothing and possibly have to cover nothing more.

We wish to express our profound thanks to all those who helped in making this book a reality. Much needed moral support and encouragement is provided on numerous occasions by our whole family. We wish to thank the **Publisher** and the entire team of **Technical Publications** who have taken immense pain to get this book in time with quality printing.

Any suggestion for the improvement of the book will be acknowledged and well appreciated.

Authors
A. A. Puntambekar
Vaishali Rane
Nilesh J. Vispute

Dedicated to God.

SYLLABUS

Software Testing (22518)

Teaching Scheme			Credit (L + T + P)	Examination Scheme											
				Theory						Practical					
L	T	P		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

Unit	Unit Outcomes (UOs) (in cognitive domain)	Topics and Sub - topics
Unit - I **Basics of Software Testing and Testing Methods**	1a. Identify errors and bugs in the given program. 1b. Prepare test case for the given application. 1c. Describe the Entry and Exit Criteria for the given test application. 1d. Validate the given application using V model in relation with quality assurance. 1e. Describe features of the given testing method.	1.1 Software Testing, Objectives of Testing. 1.2 Failure, Error, Fault, Defect, Bug Terminology. 1.3 Test Case, When to Start and Stop Testing of Software (Entry and Exit Criteria). 1.4 Verification and Validation (V Model), Quality Assurance, Quality Control. 1.5 Methods of Testing: Static and dynamic Testing 1.6 The box approach : White Box Testing : Inspections, Walkthroughs, Technical Reviews, Functional Testing, Code Coverage Testing, Code Complexity Testing. 1.7 Black Box Testing: Requirement Based Testing, Boundary Value Analysis, Equivalence Partitioning,
Unit - II **Types and Levels of Testing**	2a Apply specified testing level for the given web based application. 2b Apply Acceptance testing for given web based application 2c Apply the given performance testing for the specified application. 2d Generate test cases for the given application using regression and GUI testing.	2.1 Levels of testing Unit Testing : Driver, Stub 2.2 Integration Testing : Top-Down Integration, Bottom-Up Integration Bi-Directional Integration. 2.3 Testing on Web Application : Peformance Testing : Load Testing, Stress Testing, Security Testing, Client-Server Testing 2.4 Acceptance Testing : Alpha Testing and Beta Testing, Special Tests : Regression Testing, GUI testing.

Unit - III **Test Management**	3a. Prepare test plan for the given application. 3b Identify the resource requirement of the given application. 3c. Prepare test cases for the given application. 3d. Prepare test report of executed test cases for given application.	3.1 Test Plannning : Preparing a Test Plan, Deciding Test Approach. Setting Up Criteria for testing, identifying Responsibilities, Staffing, Resource Requirements, Test Deliverables, Testing Tasks. 3.2 Test Managment : Test Infrastructure Management, Test People Management. 3.3 Test Process : Base Lining a Test Plan, Test Case specification. 3.4 Test Reporting : Executing Test Cases, Preparing Test Summary Report.
Unit - IV **Defect Management**	4a. Classify defects on the basis estimated impact. 4b. Prepare defect template on the given application. 4c. Apply defect management process on the given application. 4d. Write procedure to find defect using the given technique.	4.1 Defect Classification, Defect Management Process. 4.2 Defect Life Cycle, Defect Template 4.3 Estimate Expected Impact of a Defect, Techniques for finding Defects, Reporting a Defect
Unit - V **Testing Tools and Measurements**	5a. Improve testing efficiency using automated tool for given application. 5b. Identify different testing tools to test the given application. 5c. Describe Metrics and Measurement for the given application 5d. Explain Objct oriented metrics used in the given testing application	5.1 Manual Testing and Need for Automated Testing Tools 5.2 Advantages and Disadvantages of Using Tools 5.3 Selecting a Testing Tool. 5.4 When to Use Automated Test Tools, Testing Using Automated Tools. 5.5 Metrics and Measurement : Types of Metrics, Product Metrics and Process Metrics, Object oriented metrics in testing.

TABLE OF CONTENTS

Unit - I

Unit - II

Unit - III

Unit - IV

Unit - V

UNIT - I

1 Basics of Software Testing and Testing Methods

1.1 Software Testing

1.1.1 Definition

- Testing is defined as execution of work product with an intent to find the defect.
- Testing is a process of uncovering as many errors as possible.
- Testing is used to confirm that a program performs its intended functions correctly.

1.1.2 Objectives of Software Testing

Following are objectives of software testing –

1) Testing should find the defects before customer finds them out.
2) Testing should be applied all through the software life cycle.
3) The testing must be conducted with some goal or reason.
4) Testing must prevent the defects.
5) Tests the tests firsts.
6) During testing, find the scenario where the product does not perform as per the requirements and perform testing there.
7) During testing, find the scenario where product does things that it is not supposed to do.
8) Testing must be a fine balance of defect prevention and defect detection.

1.1.3 Principles of Testing

The fundamental principles of testing are -

1. The goal of testing is to **find the defects** before customer finds them out.
2. Always understand the **reason behind** the testing.
3. Testing is to be **carried out throughout the software life cycle** and not simply at the end of the software development process.
4. First, **test the test cases** adopted for testing.
5. Testing encompasses **defect prevention.**
6. Testing should help for both **defect prevention** and **defect detection.**
7. Testing is a process that need to be **carried out constantly.**
8. Exhaustive testing is not possible. During the testing of the program, only the **presence** of defects can be shown and **not their absence.**
9. The testing activity must be done with **intelligent and systematic automation tool.**
10. Testing requires **talented, committed staff** who possess the ability to work in a team.

1.1.4 Skills of Software Tester

Following skills are required by software tester –

1) Analytical Skills :

- Analytical skills are those skills that help to break up a complex software system into smaller units to gain better **understanding of the system.**
- This kind of skill also helps in creating appropriate **test cases.**

2) Communication Skills :

- A software tester must have verbal as well as written communication skills.
- He/she should be a good listener.
- He should be able to convince the need for testing the module to developer as well as customer.

3) Presentation Skills :

- A good tester must also possess good presentation skills to provide the exact status of the test project and application under test.
- The tester is supposed to present the test results to developer team, customer and management team and convince them for further improvements.

4) Technical Skill :

- A good tester must have the knowledge of database, programming, and commands.
- He/She should have knowledge and hands on experience of test management tools, and automation tools.
- He/She should be enthusiastic to learn new techniques and skills required for testing.

5) Management and Organization Skills :

- Testing at times could be a demanding job especially during the release of code.
- A software tester must efficiently manage workload, have high productivity, exhibit optimal time management, and organization skills.

Board Questions

1. *List all objectives of testing.* **MSBTE : Winter-15, Marks 4**
2. *What is software testing ? State objectives of software testing.* **MSBTE : Summer-16, 18, Winter-17, 18, Marks 4**
3. *State any four testing principles.* **MSBTE : Summer-16, Marks 4**
4. *Define software testing. List all skills of software tester.* **MSBTE : Summer-17, Marks 4**
5. *List and describe any four skills of software tester.* **MSBTE : Winter-16, 18, Marks 4**
6. *Define software testing and role of testing.* **MSBTE : Summer-19, Marks 4**

1.2 Failure, Error, Fault, Defect, Bug Terminology

Following are some important terminologies used in relation with software testing -

Mistake : It is an issue or a problem identified during peer reviewing. It is of low cost and can be fixed quickly.

Error : It is an issue identified internally or during unit testing. Normally error occurs when human actions produce undesirable results

Defect : It is an issue identified by customer.

Bug : Bug is an initiation of error or problem because of which the fault may occur in the system.

Fault : It is a condition that causes the software to fail to perform its required function.

Failure : It is the inability of a system or component to perform required function according to its specification.

Board Question

1. *Explain the terms mistake, error, defect, bug, fault and failure in relation with software testing.* **MSBTE : Summer-16, Marks 6, Winter-17, Marks 4**

1.3 Test Case

1.3.1 What is Test Case ?

A Test case is a set of conditions and expected results under which a tester will determine whether a system under test satisfies requirements or works correctly.

Test cases are normally designed for particular test scenario in order to verify compliance against a specific requirement.

Various parameters using which the test case is prepared are –

1) **Test case ID :** It is an ID for the test case
2) **Test case scenario :** The description of the scenario for which the test case is to be prepared.
3) **Test case description :** The purpose of the test case is described under this section
4) **Prerequisites :** Any precondition that must be fulfilled before execution of the test case must be mentioned here.
5) **Test procedure :** The step by step procedure that demonstrates the testing procedure.
6) **Test data :** The data to be used while conducting the test.
7) **Expected result :** The expected result of the test

8) **Actual result :** Actual result of the test which is to be filled after executing the test

9) **Status :** The status can be PASS or FAIL or being a test passed or failed.

1.3.2 Features of Test Cases

1. Test cases must be simple and transparent.
2. Do not assume functionalities and some extra features for test case design.
3. Avoid repetition of test cases.
4. Test cases must be identifiable.
5. Create test case by keeping end user in mind.

1.3.3 When to Start and Stop Testing of Software ?

Entry criteria

- Software testing should start early in the Software Development Life Cycle. This helps to capture and eliminate defects in the early stages of SDLC i.e requirement gathering and design phases.
- An early start to testing helps to reduce the number of defects and ultimately the rework cost in the end.
- **Definition :** Entry criteria for testing can be defined as "Specific conditions or on-going activities that must be present before a process can begin." The Software Testing Life Cycle (STLC) specifies the entry criteria required during each testing phase.
- It also defines the time interval or the expected amount of lead-time to make the entry criteria item available to the process.
- Following are the inputs necessary for the entry criteria –
  - The requirements documents
  - Complete understanding of the application flow
  - Test plan

Exit criteria

- **Definition :** It can be defined as "The specific conditions or on-going activities that should be fulfilled before completing the software testing life cycle."
- The exit criteria can identify the intermediate deliverables.
- The following exit criteria should be considered for completion of a testing phase:
  - Ensuring all critical Test Cases are passed
  - Achieving complete Functional Coverage
  - Identifying and fixing all the high-priority defects
- The output achieved through exit criteria are –
  - Test summary report
  - Test logs

Board Questions

1. *What is a 'test case' ? State its specification parameter.* **MSBTE : Summer-16, Marks 4**
2. *What is entry and exit criteria of software testing ?* **MSBTE : Summer-15, 16, Marks 4**
3. *Explain when to start and stop testing.* **MSBTE : Summer-19, Marks 4**

1.4 Verification and Validation, Quality Assurance, Quality Control

1.4.1 Verification and Validation (V Model)

- The **software verification and validation** is a checking and analysis process of developing software.
- In V and V requirements review, design review, code inspection and testing are the various activities that are conducted.
- Boehm has described the verification and validation as :
  - Verification means asking *"Are we building the right product ?"*
  - Validation means asking *"Are we building the product right ?"*

1.4.1.1 Verification

- Verification is a technique of evaluating whether software product fulfills the requirements or conditions imposed on them by standards and processes.
- It is a **static technique** as there is no execution of code or product. During verification, the work

product is carefully read and analyzed for detecting defects with respect to standard processes.

- **Verification criteria :** The verification is to ensure whether the program running on a particular platform satisfies the requirements and development processes.

Advantages

1. Each work product is analyzed for finding out the defects. Hence it reduces the cost of finding and fixing the defects from the system as a whole.
2. Verification confirms that during the development of work product the software development processes are correctly followed.
3. People can be trained for verification process as there is no execution of code or product.
4. The defects can be located fasters as the individual work product is getting analyzed.

Disadvantages

1. The verification process simply confirms that the development process is completely followed or not, it does not show whether the developed software product is correct or not.
2. The defects during the execution of work product cannot be detected by verification.

1.4.1.2 Validation

- Validation is a technique, to evaluate whether the final built software product fulfils the customer requirements.
- It is also called as dynamic testing as the application is executed during validation in order to find out the defects.
- Normally the validation must be done by independent users and functional experts.

Advantages

1. Validation is the only to get ensured about the functioning of the software product. Validation help to test the system during the execution.
2. The defects that can not be identified during verification can be identified during validation.

Disadvantages

1. The validation is the most time consuming process because its aim is to execute the application/software or code and hence as a result more test cases are needed to execute.
2. The cost of the product may get increased due to validation as validation process is during the execution of the work product. The bugs fixed in later stage of software development can result in increase of cost.

Difference between Verification and Validation

Sr. No.	Verification	Validation
1.	Verification refers to the set of activities that ensure software **correctly** implements the **specific function.**	Validation refers to the set of activities that ensure that the software that has been built is traceable to **customer requirements.**
2.	After a valid and complete specification the verification **starts.**	Validation begins as soon as **project starts.**
3.	Verification is for **prevention of errors.**	Validation is for **detection of errors.**
4.	Verification is conducted using **reviews, walkthroughs, inspections and audits.**	Validation is conducted using system **testing, user interface testing and stress testing.**
5.	Verification is also termed as white box testing or static testing as work product goes through reviews.	Validation can be termed as black box testing or dynamic testing as work product is executed.
6.	Verification finds about **50 to 60 %** of the **defects.**	Validation finds about **20 to 30 %** of the **defects.**
7.	Verification is based on the opinion of reviewer and may change from person to person.	Validation is based on the fact and is often stable.

8.	The verification verifies the problem statements, decisions taken during the development and execution paths.	The validation validates the requirements, functionalities and features of the product.
9.	Verification is **about process, standard and guideline.**	Validation is **about the product.**

1.4.2 V Model

- V model means the Verification and Validation model.
- This model is **V-shaped** hence is the name. It is **just like waterfall model** in which each phase must be completed before the next phase begins.
- This model contains various stages of software development along with various types of testing that can be conducted at each stage of development. The test plans serve as a link between the development stages and various tests.

When will be V model applicable ?

Following are the situations in which the V model is applicable

1. The requirements are well defined and are not ambiguous.
2. The acceptance criteria is clear.
3. Project is short to medium size.
4. Technology and tools are not changing.

Model

- As shown by this model at each stage specific testing can be carried out. Refer Fig. 1.4.1.

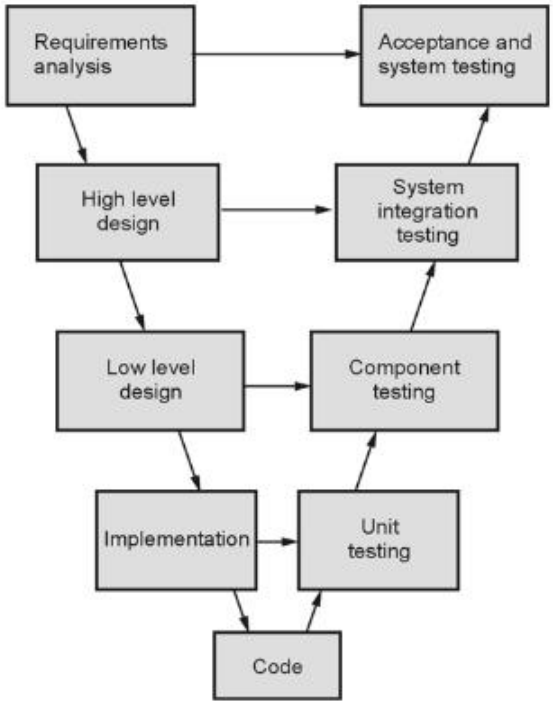

Fig. 1.4.1 The V-model

Requirements analysis

As per the waterfall model of software development process model the V model begins with requirements gathering and analysis. The Software Requirements Specification(SRS) is created during this phase. The **acceptance and system testing** is conducted in which the focus is on meeting of functionalities specified in the requirements gathering.

High level design

During this phase the system architecture is designed. It provides the overview of platform used, system, product, and service or process. An **integration test** plan is prepared and testing is conducted to test the pieces of the software systems to ensure about the ability to work together.

Low level design

During this phase actual software components are designed. The relationship with other components are defined at this level. Hence **component testing** is conducted for this phase.

Implementation

All coding takes place at this phase. The unit **testing** is carries out to test every source code module. The creation and review of various test cases are carried out in this phase.

Advantages

1. The V model is simple and easy to manage as each phase has well defined objectives and goals.
2. Development and progress of this model is very systematic.
3. It works well for small to medium sized projects.
4. Testing starts from beginning of the development stage and errors can be identified and corrected from the beginning of the software development.

Disadvantages

1. This model is not suitable for large and complex projects.
2. For the projects in which the requirements are not consistent, this model is not suitable.
3. The working software can not be produced during the intermediate stage. It is available only at the end of the development cycle.
4. There is no provision for risk analysis. Hence there is always an uncertainty about the risks.

Board Questions

1. *Describe V-model with labelled diagram. State its any two advantages and disadvantages. Also write where it is applicable.*
   MSBTE : Winter 15, 17, 18, Marks 6
2. *Differentiate between verification and validation.*
   MSBTE : Summer-16, Marks 4, Summer 18, Marks 6
3. *Explain V model with diagram.*
   MSBTE : Winter-16, Marks 6
4. *Describe V-model with labelled diagram.*
   MSBTE : Summer-17, Marks 6
5. *Explain verification and validation with neat diagram.*
   MSBTE : Summer-19, Marks 8

1.4.3 Quality Assurance

- Software quality can be defined as "the conformance to explicitly stated functional and performance requirements, explicitly documented development standards and implicit characteristics that are expected of all professionally developed software".
- **Definition of quality assurance :** It is planned and systematic pattern of activities necessary to provide a high degree of confidence in the quality of a product. It provides quality assessment of the quality control activities and determines the validity of the data or procedures for determining quality.
- The quality assurance consists of set of reporting and auditing functions.
- These functions are useful for assessing and controlling the effectiveness and completeness of quality control activities.
- The goal of quality assurance is to ensure the management of data which is important for product quality.

1.4.4 Quality Control

- **Definition :** Quality control is a process in which activities are conducted in order to maintain the quality of product. These activities are series of inspections, reviews and tests used throughout the software process. These activities ensure whether each work product is satisfying the requirements imposed on it.
- While applying the quality control there should be a feedback loop to the process which generates the work product. With the help of such feedback we can tune the process if it does not satisfy the requirements. The feedback loop helps in minimizing the defects in the software product.
- The quality control activities can be fully automated or it can be completely manual or it can be a combination of automated tools and manual procedures.

1.4.5 Difference between Quality Assurance and Quality Control

Sr. No.	Quality control	Quality assurance
1.	This is an activity with the primary goal as to prevent the defects.	This is an activity with the primary goal as to identify and correct the defects.
2.	This process is intended to provide the assurance that the quality request will be achieved.	This process is intended to focus on quality being requested.
3.	This method is to manage the quality verification.	This method is for quality validation.
4.	It does not involve executing of the program.	During this method the program is executed.
5.	It is a preventive technique.	It is a corrective technique.
6.	It is a proactive measure	It is a reactive measure.
7.	It involves the full software development life cycle.	It involves testing phase of software development life cycle.
8.	It's main goal is to prevent defects in the system. It is less time consuming activity.	It's main goal is to correct the defects in the system. Hence it is more time consuming activity.

Board Questions

1. *Give difference between quality assurance and quality control. (Any Four)*
   MSBTE : Summer-15, 18, Winter-16, 18, Marks 4
2. *Describe quality assurance and quality control.*
   MSBTE : Summer-19, Marks 4

1.5 Methods of Testing

1.5.1 Static Testing

- **Definition :** Static testing is a testing technique in which software is tested without executing the code. As the code, requirement documents and design documents are tested manually in order to find errors, it is called static.
- This kind of testing is also called as verification testing.

Static testing techniques :

- **Informal reviews :**
  - In this technique, the team of reviewers just checks the documents and give comments.
  - The purpose is to maintain the quality from the initial stage. It is non-documented in nature
- **Formal reviews :**
  - It is well structured and documented and follow six main steps: Planning, kick off, preparation, review meeting, rework follow-up
  - **Technical reviews :** The team of technical experts will review the software for technical specifications. The purpose is to pin out the difference between the required specification and product designed and then correct the flaws. It focuses on technical documents such as test strategy, test plan, and requirement specification documents.
- **Walk-through :** The author explains about the software to the team and teammates can raise questions if they have any. It is headed by the author and review comments are noted down.
- **Inspection process :** The meeting is headed by a trained moderator. A formal review is done, a record is maintained for all the errors and the authors are informed to make rectification on the given feedbacks.
- **Static code review :** Code is reviewed without execution, it is checked for syntax, coding standard, and code optimization. It is also referred as white box testing.

Advantages :

1) It is fast and easy technique used to fix errors
2) It helps in identifying flaws in code
3) With the help of automated tools it becomes very easy and convenient to scan and review the software.

4) With static testing it is possible to find errors at early stage of development life cycle.

Disadvantages :

1) It takes lot of time to conduct the testing procedure if done manually.
2) Automated tools work for restricted set of programming languages.
3) The automated tools simply scan the code and can not test the code deeply.

1.5.2 Dynamic Testing

Definition : Dynamic testing is a process by which code is executed to check how software will perform in a runtime environment. As this type of testing is conducted during code execution it is called dynamic.

It is also called as validation testing.

Dynamic testing techniques

Unit testing : As the name suggests individual units or modules are tested. The source code is tested by the developers.

Integration testing : Individual modules are clubbed and tested by the developers. It is performed in order to ensure that modules are working in a right manner and will continue to perform flawlessly even after integration

System testing : It is performed on a complete system to ensure that the application is designed according to the requirement specification document.

Advantages

1) It identifies weak areas in a runtime.
2) It helps in performing detail analysis of code.
3) It can be applied with any application.

Disadvantages

1) It is not easy to find trained software tester for performing dynamic testing
2) It becomes costly to fix errors in dynamic testing.

Difference between static and dynamic testing

Sr. No.	Static testing	Dynamic testing
1.	Static testing is a testing process which is done at early stage of development life cycle.	Dynamic testing is a testing process done later stage of development life cycle.
2.	It involves walkthrough, code review.	It involves functional and non functional testing.
3.	This testing is a verification process.	This testing is a validation process.
4.	This type of testing is done before execution of code.	This type of testing is done during code execution.
5.	It is about prevention.	It is about cure.
6.	It is cost effective.	It is costly.

Board Questions

1. *What is static testing ? State advantages and disadvantages of static testing two each.* **MSBTE : Winter-15, Marks 4**
2. *Explain the static testing and dynamic testing.* **MSBTE : Summer-16, Marks 4**
3. *Describe inspection under static testing.* **MSBTE : Summer-15, 17, Marks 4**
4. *Describe structural walk through under static testing.* **MSBTE : Winter-16, Marks 4**
5. *Describe technical review under static testing.* **MSBTE : Summer-18, Marks 4**

1.6 The Box Approach

1.6.1 White Box Testing

- In white box testing the **procedural details are closely examined.**
- In this testing the **internals of software are tested** to make sure that they operate according to specifications and designs.
- Thus **major focus** of white box testing is on internal structures, **logic paths, control flows, data flows, internal data structures, conditions, loops,** etc.

- The white box testing is also called as clear box, glass box or open box testing.

Advantages :

1. Each procedure can be tested thoroughly. The internal structures, data flows, logical paths, conditions and loops can be tested in detail.
2. It helps in optimizing the code.
3. White box testing can be easily automated.
4. Due to knowledge of internal coding structure it is easy to find out which type of input data can help in testing the application efficiently.

Disadvantages :

1. The knowledge of internal structure and coding is desired for the tested. Thus the skilled tester is required for white box testing.
2. This type of testing is costly.
3. Sometimes it is difficult to test each and every path of the software and hence many paths may go untested.
4. Maintaining the white box testing is very difficult because it may use specialized tools like code analyzer and debugging tools are required.
5. The missing functionality can not be identified.

Classification of white box testing

Following Fig. 1.6.1 represents the classification of white box testing –

Static testing :

- Static testing is carried out only on source code. It does not require any executable code.
- Following are the tasks performed during static testing –

1) To check whether code works according to functional requirements or not.
2) To check whether the code is written according to the design of system.
3) To check if any functionality of the code is missed out or not,
4) To check whether the code handles errors properly or not.

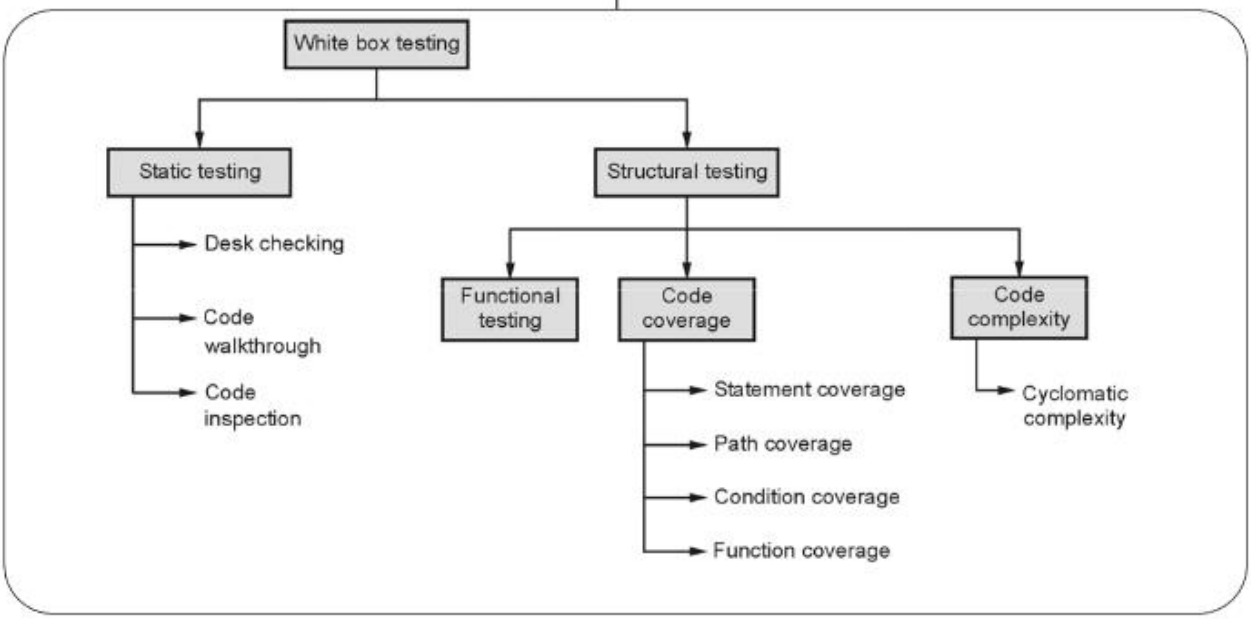

Fig. 1.6.1

- Various methods of static testing are –

1) Desk checking
2) Code walkthrough
3) Code review
4) Code inspection

1.6.1.1 Inspections

The **main goal** of inspection is to find out the defects.

Inspection is a kind of review by the group of peers by following clearly defined processes. The inspection is very formal process of verification.

Following are some guidelines given for the inspection process -

1. The inspection must be conducted by the technical people for technical people.
2. It is a structured process in which every participant have definite role.
3. The focus of code inspection is to identify the problems and not to solve them.
4. The review data is recorded and monitored for further improvement.

The inspection is basically carried out by the team of reviewers. The **author** of the team is **moderator.**

The moderator has overall responsibility to ensure that the review is done in proper manner and all the steps of review process are followed.

The inspection process can be carried out in different phases such as planning, preparation and overview, group review meeting, rework and followup.

Advantages :

1. Single inspection can discover multiple errors; on the other hand one error may hide another error. From output of the system we cannot predict whether the wrong output is due to existing error or because of some new error.
2. Incomplete version of the system can be inspected but it is difficult to test such version.
3. Using inspections we can search for programming defects, poor programming style or use of inappropriate algorithm.
4. More than 90 % of errors can be detected by conducting formal inspections. If the defects are found during inspections, programmers can avoid the some mistakes in the later phases of software development.

Disadvantages :

1. Software engineers are reluctant to accept that the inspections can be more effective for defect detection and testing.
2. Project managers feel that inspections may require additional cost if they are conducted during the software design and development.
3. Since inspections take time to arrange and it slows down the development process, conducting inspection is avoided.

Roles and responsibilities

The program inspection is a formal process. It is conducted with the help of four people and roles of them are : Author, reader, tester and moderator.

Author or owner is a person who has created the program or design. This person is responsible for fixing the defects during the inspection process.

Moderator is the leader of the inspection process who plans and co-ordinates the inspection. He reports the process results to the chief moderator.

Reader is the person who reads the code aloud to the inspection team.

Tester or inspector is the person who inspects the code from testing perspective. He finds the errors, omissions and inconsistencies in the software.

Scribe records the results of inspection meeting.

Chief moderator is responsible for inspection process improvements, preparing or updating checklists.

Inspection process

The inspection process can be illustrated by following Fig. 1.6.2.

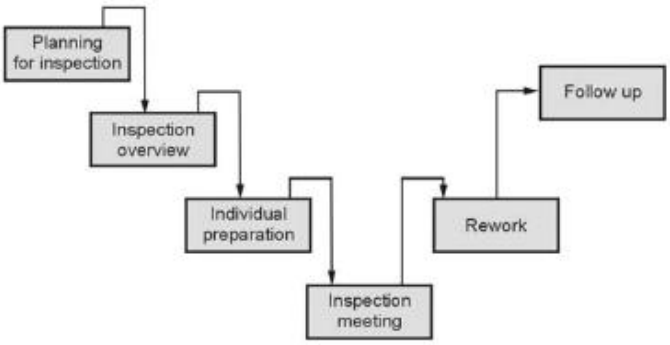

Fig. 1.6.2 Inspection process

Planning : The inspection process is planned by the moderator. Various tasks that can be conducted during planning are :

- Selecting inspection team
- Organising meeting room
- Ensuring that the required documents and resources for the inspection are available.

Overview : The program or the code to be inspected is presented to the inspection team during the overview. The author of the program/code explains its purpose.

Individual preparation : Each inspection team member then studies the code and tries to find the possible defects.

Inspection meeting : During this meeting, the focus should be on pinpointing the defects from each part of the program, non-compliance to the standards and poor-quality programming. But the inspection team must not suggest how to correct the defects.

Rework : After the inspection meeting, author should understand the mistakes and should correct the identified defects. He should try to remove as many anomalies as possible.

Follow-up : During this stage the reworked program is presented to the moderator. The moderator then has to decide whether the re-inspection is needed or not. If all the identified defects are removed successfully and the inspection team gets satisfied with the work product, moderator approves the program for the release.

Inspection checks

During the inspection a checklist is prepared for common programming errors. It can be prepared by the person(s) having adequate experience in application domain. This checklist is very useful to the programmer. The checklist varies for different programming languages because different programming languages have different programming constructs. For example :

Checklist for C Program

- Is the loop executed for correct number of times ?
- Do the parameter and argument types in the function call and definitions match ?
- Are the global variables definitions consistent throughout the program ?

1.6.1.2 Walkthrough

- Walkthrough is more formal than peer review. Many times it is called as semi-formal review.
- During walkthrough author lead the review process and other team members ask the questions and identify the all possible errors.
- The main purpose of walkthrough is to enable to understand the contents of the documents under review and to find the defects.
- Walkthrough helps the teams to develop the communication during review.

Advantages :

1. It is useful for the communication about the issues under review.
2. Team members can understand the things that are expected during the review.
3. The problems are recorded and suggestions can be obtained from all the team members. Hence all the team members can participate in improvement of the work product.

Disadvantages :

1. It is difficult to handle the team of large size during walkthrough.
2. It can be time consuming activity.

Difference between walkthrough and inspection

Sr. No.	Walkthrough	Inspection
1.	It is an informal method of verification.	It is more formal method of verification.
2.	It is initiated by author.	It is initiated by project team.
3.	This method of verification is not planned and systematic.	This method involves planned meetings, and fixed roles are assigned to the team members for verification.
4.	Author simply notes the defects and suggestions offered by the team members.	Recorder records the defects. Moderators directs in such a ways that the discussion proceeds on productive line.

1.6.1.3 Technical Reviews

- This is an informal way of verification and validation. There are two types of review - self review and peer review.
- The self review is normally done by the tester himself who has written the test cases. He can verify whether all the requirements are covered or not by looking into software requirements specification. Whereas peer review is done by another tester who hasn't written those test cases but is familiar with the system under test. This type of review is also known as maker and checker review.
- There are two kinds of peer reviews - 1. Online peer review and offline peer review.
- The online peer review in which the author and the reviewer sit together and review the work product jointly. In this meeting any explanation can be immediately provided by the author to the reviewer.
- The offline peer review is a kind of review in which the author informs the reviewer that the work product is ready for review. The reviewer reviews the work product as per his or her convenience and sends the review report to the author. There is no direct interaction of author and reviewer.

Advantages :

1. Review is a very useful tool in defect finding.
2. Review can be conducted at convenient timings of both the author and reviewer.

Disadvantages :

1. Sometimes people involved in self review may not conduct the review in reality due to time constraints.
2. In peer review, if the other person doing the review is not expert or possessing the lack of knowledge about the system under review then his/her suggestions may not be valid.

1.6.1.4 Unit/Code Functional Testing

- This is structural testing method in which some **quick checks** are made before the **code coverage testing** or **code complexity testing.**
- Following are three approaches used during unit/code functional testing –
  - Initially the developer performs some **typical tests** or **obvious tests** with **known input sets** and corresponding **expected outputs**. These tests are repeated for multiple inputs and any obvious mistakes can be removed from the code. This increases the level of confidence. These quick tests are generally performed before the **formal reviews** of static testing.
  - For complex logic testing, developer prepares the **debug version** of the code. The **debug version** is a kind of version of your source code in which the **print** statements are inserted at intermediate places in the code. This can be done to test if the program is executing through right loops and iterations for right number of times. After fixing the errors, these print statements can be removed.

- In this approach the code is run under a **standard debugger** or **Integrated development Environment(IDE)**. These **standard tools debug** each and every instruction of your program. The developer can watch the values of variables or other functional parameters at each step in the iteration.

1.6.1.5 Code Coverage Testing

- This is a testing process in which the **test cases** are designed and executed for different parts of code. Thus some amount of **code is covered by testing**. The amount of code covered by testing is found out by a technique called **instrumentation of code**. The instrumentation of code can be performed used some standard tools.
- Code coverage testing can be performed using following techniques –
  - Statement coverage
  - Path coverage
  - Condition coverage
  - Function coverage

Let us discuss these techniques –

1) Statement coverage :

- There are various types of programming statements such as **sequential control flow, if-else** statements, **switch-case** statements, **loops such as for, while,do-while**
- **Sequential control flow statements :** For these kind of statements, the test can be designed to run through from **top** to **bottom.** A test case starts from top and runs covering full section of the code upto bottom.
- **If-else statement :** The test cases are written to cover all parts of code. That means the set of test cases are executed for if **part** and another set of test cases are executed for else **part.**
- **Switch-case statement :** There should be multiple test cases executed covering each case statement in the switch-case control structure.
- **Loop construct(for, while, do while) :** This is the most complex part for statement coverage. Many defects occur in the program if the loops do not execute correctly. Hence set of test cases must be executed for -

1) Exercising the loop at least once and maximum number of times to ensure all the normal flow of execution of loop.
2) Skipping the loop completely, so that termination condition of the loop can be tested before starting the loop.
3) Testing the boundary of the loop. For example if the statement is

```
while(i<=n)
{
    ...
}
```

Then the test case must be written when **i** has exact value of **n** or **n–1**

Calculation of statement coverage

Statement coverage is an indication of the percentage of statements actually executed in a set of tests. The statement coverage can be calculated using the formula –

$$\text{Statement coverage} = \frac{\text{Total statements exercised}}{\text{Total number of executable statements}} \times 100$$

2) Path coverage :

- This is a kind of testing in which the program is divided into number of distinct paths and test cases can be written for each path of the program.
- For example – Consider following code for simple subtraction

1. Input x and y
2. if x > y then
3. z = x – y
4. else z = y – x
5. endif
6. output z

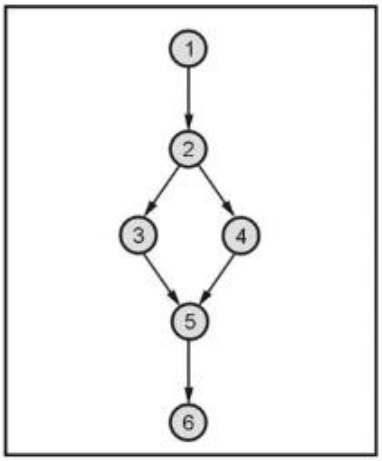

Fig. 1.6.3

Refer Fig. 1.6.3 for flow graph.

Here we need to test two paths

Path 1: 1,2,3,5,6

Path 2: 1,2,4,5,6

- Thus regardless of the number of statements in each of these paths, if we can execute these paths, then we would have covered most of the typical scenarios.
- Hence path coverage can be calculated using formula –

$$\text{Path coverage} = \frac{\text{Total paths exercised}}{\text{Total number of paths in program}} \times 100$$

3) Condition coverage

- Condition coverage exercises all kinds of situations that the path coverage may not perform
- For example – Consider following code for simple subtraction

1. Input x and y
2. if x > y then
3. z = x – y
4. else z = y – x
5. endif
6. output z

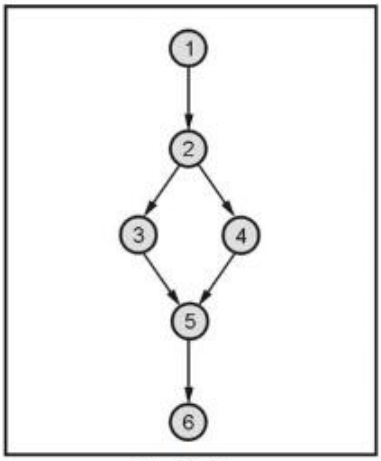

Fig. 1.6.4

Here by path coverage criteria two paths can be tested (1-2-3-5-6 and 1-2-4-5-6) for the conditions x > y and x < y. But what if the value of x = y, as there is no such path for this condition. Hence it is preferred to perform testing based on various conditions.

- Hence condition coverage can be calculated using formula –

$$\textbf{Condition coverage} = \frac{\textbf{Total decisions exercised}}{\textbf{Total number of decisions in program}} \times 100$$

4) Function coverage :

- In this technique the tests cases are executed for various functions present in the program.
- While providing function coverage, test cases can be written so as to exercise each of the different functions in the code.

Advantages of functional coverage :

- Functions are easier to identify in the program and hence test case be written easily for them.
- The complete code coverage for the function is possible during testing.
- Functions are logical mapping of requirements hence direct testing of direct requirements of is possible.
- It is easy to prioritize functions for testing.
- It provides natural transition to black box testing.

- Hence function coverage can be calculated using formula –

$$\text{Function coverage} = \frac{\text{Total functions exercised}}{\text{Total number of functions in program}} \times 100$$

1.6.1.6 Code Complexity Testing

- The code complexity is measured by cyclomatic complexity measure.
- The cyclomatic complexity can be defined as a measure used to determine the complexity of a piece of code or functionality.
- This metric measures **independent paths** through program source code. Independent path is defined as a path that has at least one edge which has not been traversed before in any other paths.
- The cyclomatic complexity can be calculated for function, module, class, methods present in the program.
- For computing the cyclomatic complexity the flow graph need to be designed.
- Flow Graph notation for a program is defines. several nodes connected through the edges. Below are Flow diagrams for statements like if-else, While, until and normal sequence of flow.

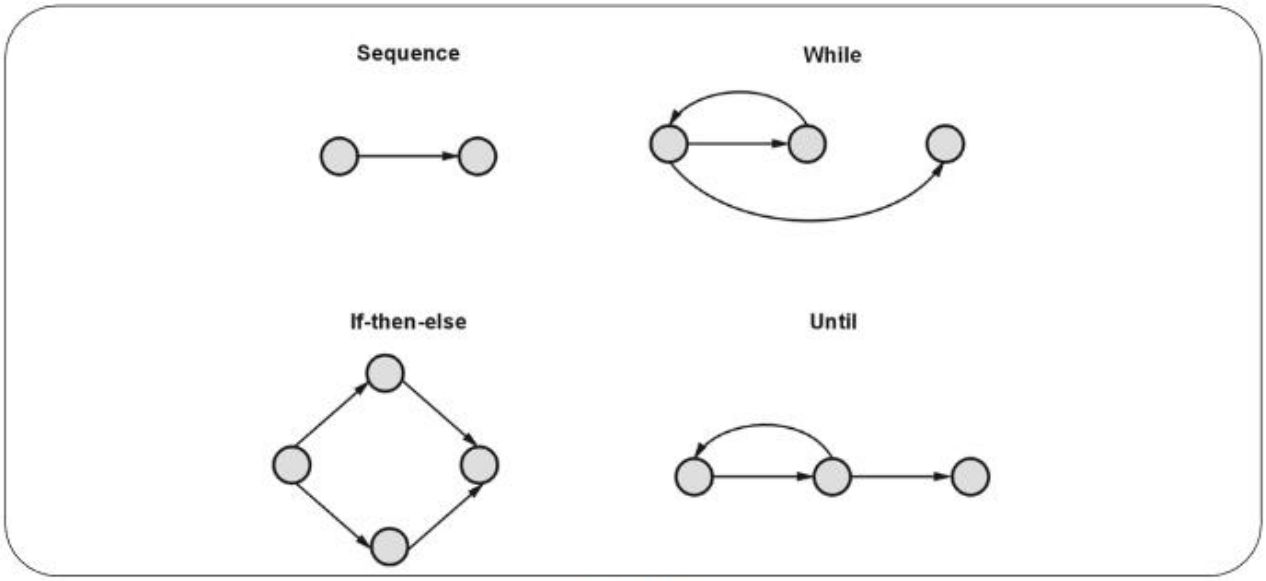

Fig. 1.6.5

- Cyclomatic complexity can be computed in three ways –
  - Cyclomatic complexity = E + V – 2
  - Where E is number of edges, V is number of nodes.
  - Cyclomatic complexity = P + 1

    Where P is number of predicate nodes. The predicate node is a node that contains some condition.
- This method is also called as basis path testing method as number of independent paths are obtained in this method.
- How to perform **Code Complexity testing ?**

Following steps are followed for code complexity testing

Step 1 - Construction of graph with nodes and edges from the code

Step 2 - Identification of independent paths

Step 3 - Cyclomatic complexity calculation

Step 4 - Design of test cases. The number of test cases is equal to cycomatic complexity measure.

Once the basic set is formed, **test cases** should be written to execute all the paths.

Ex. 1.6.1 : ***Draw the flow graph for finding maximum of three numbers and derive the testcases using cyclomatic complexity.***

Sol. : The flow graph for given program is - (From Fig. 1.6.6)

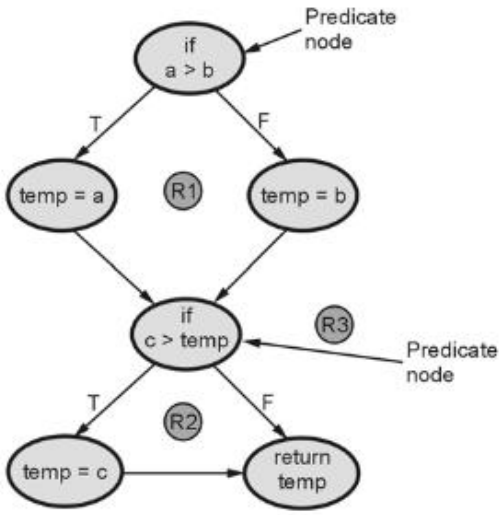

Fig. 1.6.6 Flow graph for finding maximum of three numbers

Cyclomatic complexity = E – N + 2 = 7 – 6 + 2 = 3

Cyclomatic complexity = P + 1 = 2 + 1 = 3

Cyclomatic complexity = Regions encountered = 3

Hence cyclomatic complexity of given program is 3.

Board Questions

1. *Explain what the static white box and black box testing is required.* **MSBTE : Summer-16, Marks 4**
2. *Describe inspection process performed under white box testing.* **MSBTE : Winter-15, Marks 4**
3. *Give any two advantages and any two limitations of function/test matrix.* **MSBTE : Summer-18, Marks 4**
4. *State purpose of code coverage. How it is used in analysing coding of software ?* **MSBTE : Winter-15, Marks 4**
5. *What is white box testing ? Classify static white box testing. State any one situation where white box testing used.* **MSBTE : Winter-15, Marks 4**
6. *Draw classification of white box testing. Explain any one type of white box testing in detail.* **MSBTE : Summer-15, Marks 4**
7. *With the suitable example, explain how 'Basis Path Testing' is used to derive the code complexity for the testing.* **MSBTE : Summer-16, Marks 8**
8. *What is code coverage testing ? Explain the following types of coverage :*
   i) Path coverage ii) Condition coverage. **MSBTE : Winter-16, Marks 8**
9. *What is the use of code complexity testing ? Also compute code complexity with the help of suitable example.* **MSBTE : Summer-17, Marks 6**
10. *What is white box testing ? Explain any one technique of static white box testing.* **MSBTE : Winter-17, Marks 4**
11. *With the suitable example, explain how 'Basis Path Testing' is used to derive the code complexity for the testing.* **MSBTE : Winter-17, Marks 8**
12. *Explain any one type of structural white box testing in detail.* **MSBTE : Summer-18, Marks 4**
13. *Give any four differences between walkthrough and inspection.* **MSBTE : Winter-18, Marks 4**
14. *Draw classification of white box testing. Explain the purpose of code coverage testing* **MSBTE : Winter-18, Marks 4**
15. *Explain inspection and walkthrough.* **MSBTE : Summer-19, Marks 4**
16. *Explain code functional testing and code coverage testing with example.* **MSBTE : Summer-19, Marks 8**

1.7 Black Box Testing

- The black box testing is used to demonstrate that the software functions are operational.
- As the name suggests in black box testing it is tested whether the input is accepted properly and output is correctly produced.

Advantages :

1. The black box testing focuses on fundamental aspect of system without being concerned for internal logical structure of the software.
2. The advantage of conducting black box testing is to uncover following types of errors.
   i. Incorrect or missing functions
   ii. Interface errors
   iii. Errors in external data structures
   iv. Performance errors
   v. Initialization or termination errors

Disadvantages :

1. All the independent paths within a module cannot be tested.
2. Logical decisions along with their true and false sides cannot be tested.
3. All the loops and the boundaries of these loops cannot be exercised with black box testing.
4. Internal data structure cannot be validated.

Types of black box testing

Some commonly used black box testing techniques are –

1. Requirement based testing
2. Boundary value analysis
3. Equivalence partitioning

1.7.1 Requirement Based Testing

- This is a testing approach in which test cases are designed using complete and consistent set of requirements.
- This is testing technique which revolves around the requirements.

- The strategy of requirement based testing is to integrate testing throughout the life cycle of the software development process, to assure quality of the requirement specification.

How to perform requirements based testing ?

Step 1 : Define test Completion criteria :

Testing should be defined in quantifiable terms. The goal is considered to be achieved only when test coverage is 100 %.

Step 2 : Design test cases :

Test cases must be in accordance with requirements specification.

Step 3 : Build test cases :

Join the logical parts together to form/build test cases .

Step 4 : Execute test cases :

Execute the test cases to evaluate the results.

Step 5 : Verify test results :

Check whether actual results deviate from the expected ones.

Step 6: Verify test coverage :

Check for functional test coverage.

Step 7: Manage test library :

Test manager is responsible for monitoring the test case executions, that is, the tests passed or failed, or to ascertain whether all tests have been successfully performed.

Concept of requirement traceability matrix

- Requirements are tracked by Requirements Traceability Matrix (RTM).
- RTM traces all the requirements from design, development, and testing.
- The traceability matrix is typically a worksheet that contains the requirements with its all possible test scenarios and cases and their current state, i.e. if they have been passed or failed. This would help the testing team to understand the level of testing activities done for the specific product.

Parameters to be included in requirement Traceability matrix

(1) Requirement ID

(2) Requirement type and description

(3) Test cases

(4) Test case status

Example

Test case ID	Test case description
TC#001	Login with invalid username but valid password
TC#002	Login with valid username but invalid password
TC#003	Login with valid user name and valid password

Requirement ID	Requirement Description	Test case ID	Status
101	Login the website	TC#001	Pass
		TC#002	Pass
		TC#003	Pass

Advantages of requirements traceability matrix

1. It confirms 100 % test coverage.
2. It highlights any requirements missing or document inconsistencies.
3. It shows the overall defects or execution status with a focus on business requirements.
4. It helps in analyzing or estimating the impact on the QA team's work with respect to revisiting or re-working on the test cases.

1.7.2 Boundary Value Analysis

- Boundary value analysis is done to check boundary conditions.
- A boundary value analysis is a testing technique in which the elements at the edge of the domain are selected and tested.
- Using boundary value analysis, instead of focusing on input conditions only, the test cases from output domain are also derived.

- Boundary value analysis is a test case design technique that complements equivalence partitioning technique.
- **Guidelines** for boundary value analysis technique are -
  1. If the input condition specified the range bounded by values x and y, then test cases should be designed with values x and y. Also test cases should be with the values above and below x and y.
  2. If input condition specifies the number of values then the test cases should be designed with minimum and maximum values as well as with the values that are just above and below the maximum and minimum should be tested.
  3. If the output condition specified the range bounded by values x and y, then test cases should be designed with values x and y. Also test cases should be with the values above and below x and y.
  4. If output condition specifies the number of values then the test cases should be designed with minimum and maximum values as well as with the values that are just above and below the maximum and minimum should be tested.
  5. If the internal program data structures specify such boundaries then the test cases must be designed such that the values at the boundaries of data structure can be tested.

For example :

Integer D with input condition [– 2, 10],

Test values : – 2, 10, 11, – 1, 0

If input condition specifies a number values, test cases should developed to exercise the minimum and maximum numbers. Values just above and below this min and max should be tested.

Enumerate data E with input condition : {2, 7, 100, 102}

Test values : 2, 102, – 1, 200, 7

1.7.3 Equivalence Partitioning

- It is a black box technique that divides the input domain into classes of data. From this data test cases can be derived.
- An ideal test case uncovers a class of errors that might require many arbitrary test cases to be executed before a general error is observed.
- In equivalence partitioning the equivalence classes are evaluated for given input condition. Equivalence class represents a set of valid or invalid states for input conditions.
- Equivalence class guidelines can be as given below :

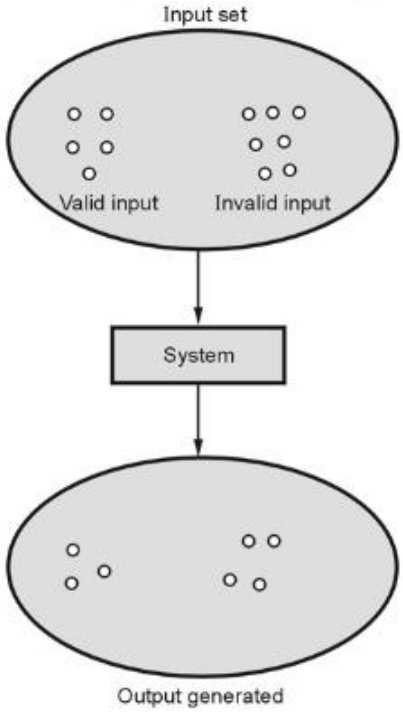

Fig. 1.7.1

- If input condition specifies a range, one valid and two invalid equivalence classes are defined.
- If an input condition requires a specific value, one valid and two invalid equivalence classes are defined.
- If an input condition specifies a member of a set, one valid and one invalid equivalence class is defined.
- If an input condition is Boolean, one valid and one invalid equivalence class is defined.

For example :

Area code : Input condition, Boolean - The area code may or may not be present.

Input condition, range - Value defined between 200 and 700.

Password : Input condition, Boolean - A password may or may not be present.

Input condition, value - Seven character string.

Command : Input condition, set - Containing commands noted before.

1.7.4 Difference between Black Box and White Box Testing

Sr. No.	Black Box Testing	White Box Testing
1.	Black box testing is the software testing method which is used to test the software **without knowing the internal structure** of code or program.	White box testing is the software testing method in which **internal structure** is being **known** to tester who is going to test the software.
2.	This type of testing is **carried out by testers.**	Generally, this type of testing is carried out **by software developers.**
3.	**Implementation knowledge is not required** to carry out Black Box Testing.	**Implementation knowledge is required** to carry out White Box Testing.
4.	Programming knowledge is not required to carry out Black Box Testing.	Programming knowledge is required to carry out White Box Testing
5.	Testing is applicable on **higher levels of testing** like System testing, Acceptance testing.	Testing is applicable on **lower level of testing** like Unit Testing, Integration testing
6.	Black box testing means functional test or **external testing.**	White box testing means structural test or **interior testing.**
7.	Black Box testing can be started based on **Requirement Specifications documents.**	White Box testing can be started based on **Detail Design documents.**
8.	The functional testing, Behavior testing, **Close box testing** is carried out under Black Box testing.	The Structural testing, Logic testing, Path testing, Loop testing, Code coverage testing, Open box testing is carried out under White Box testing.

Board Questions

1. *Distinguish between white box testing and black box testing. (any six)* **MSBTE : Summer-18, Winter-18, Marks 6**
2. *With the help of example explain boundary value analysis.* **MSBTE : Summer-15, 17, Winter-17, Marks 8**
3. *Explain the boundary value analysis technique used in black box testing with example.* **MSBTE : Summer-16, Marks 4**
4. *Why boundary value analysis is required ? Give example.* **MSBTE : Winter-16, Marks 4**
5. *What is boundary value analysis ? List any three guidelines for boundary value analysis.* **MSBTE : Summer-18, Marks 4**
6. *Explain concept of application of equivalence partitioning. How it is useful in result analysis of diploma ?* **MSBTE : Winter-15, Marks 6**
7. *Illustrate process of equivalence partitioning with example.* **MSBTE : Summer-15, 17, Marks 4, Winter-16, Marks 6**
8. *Explain equivalence partitioning with respect to equivalence classes.* **MSBTE : Summer-18, Marks 4**
9. *Explain requirement based testing with its stages and requirement testing process.* **MSBTE : Winter-18, Marks 4**
10. *Explain the impact of equivalence partitioning in coding and testing.* **MSBTE : Summer-19, Marks 4**

❑❑❑

UNIT - II

2 Types and Levels of Testing

2.1 Levels of Testing

We begin by 'testing-in-the-small' and move toward 'testing-in-the-large'.

Various testing strategies for conventional software are

1. Unit testing
2. Integration testing
3. Validation testing
4. System testing

1. **Unit testing** - In this type of testing techniques are applied to detect the errors from each software component individually.

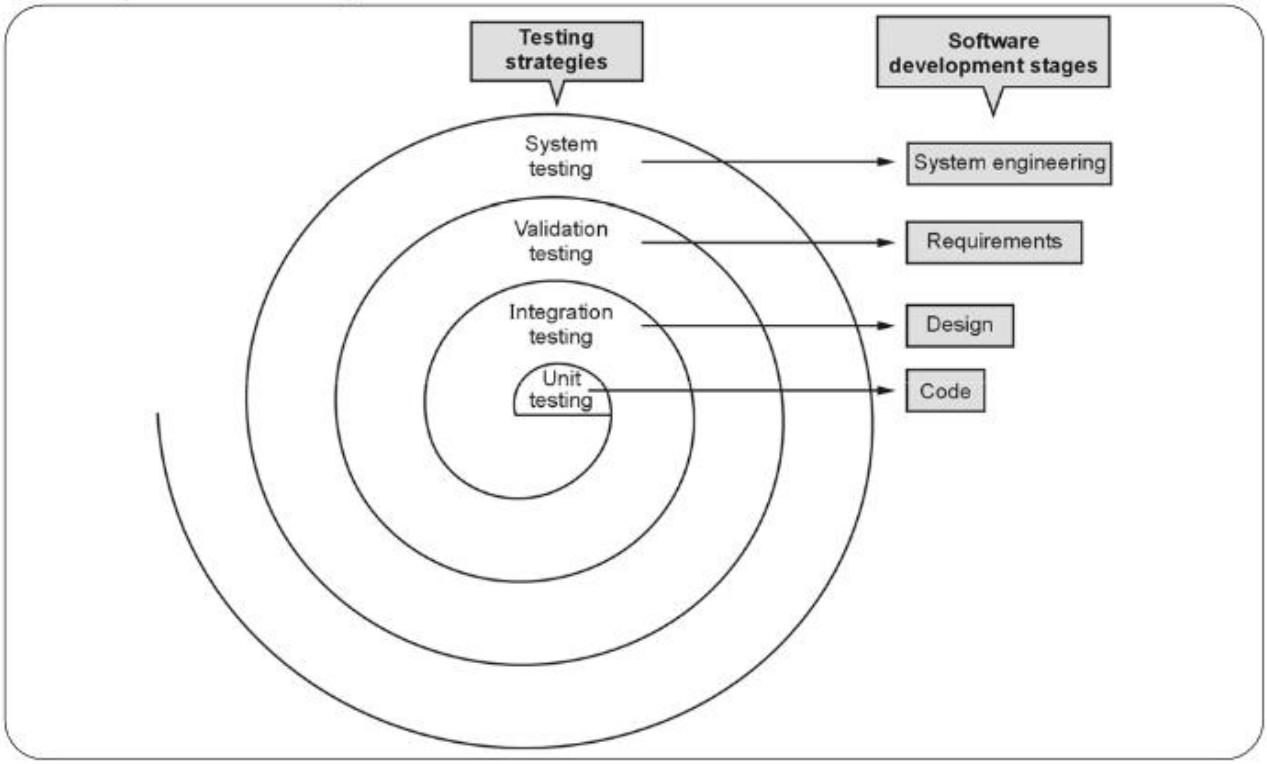

Fig. 2.1.1 Testing strategy

2. **Integration testing** - It focuses on issues associated with verification and program construction as components begin interacting with one another.

3. **Validation testing** - It provides assurance that the software validation criteria (established during requirements analysis) meets all functional, behavioural and performance requirements.
4. **System testing** - In system testing all system elements forming the system is tested as a whole.

2.2 Unit Testing

- In unit testing the individual components are tested independently to ensure their quality.
- The focus is to uncover the errors in design and implementation.
- The various tests that are conducted during the unit test are described as below.
  1. Module interfaces are tested for proper information flow in and out of the program.
  2. Local data are examined to ensure that integrity is maintained.
  3. Boundary conditions are tested to ensure that the module operates properly at boundaries established to limit or restrict processing.
  4. All the basis (independent) paths are tested for ensuring that all statements in the module have been executed only once.
  5. All error handling paths should be tested.

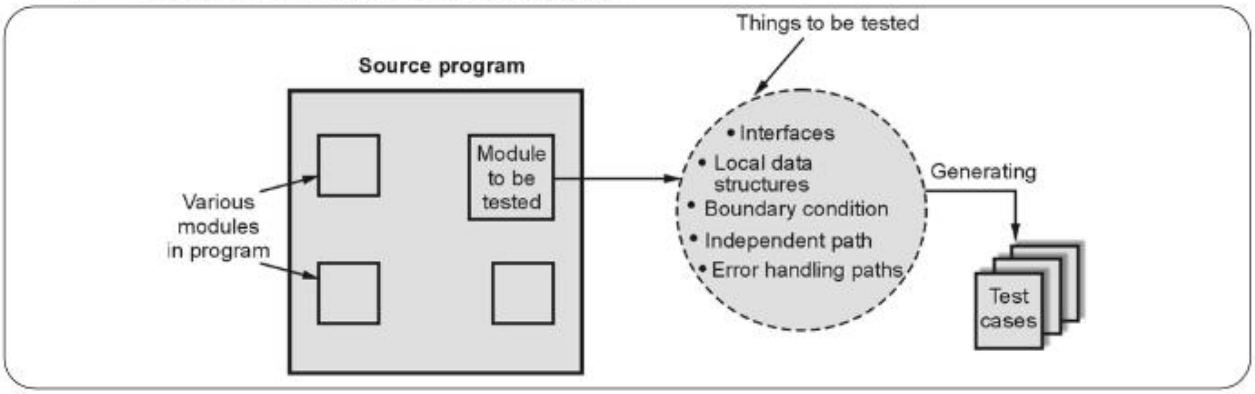

Fig. 2.2.1 Unit testing

6. Drivers and stub software need to be developed to test incomplete software. The "driver" is a program that accepts the test data and prints the relevant results. And the "stub" is a subprogram that uses the module interfaces and performs the minimal data manipulation if required. This is illustrated by following Fig. 2.2.2.

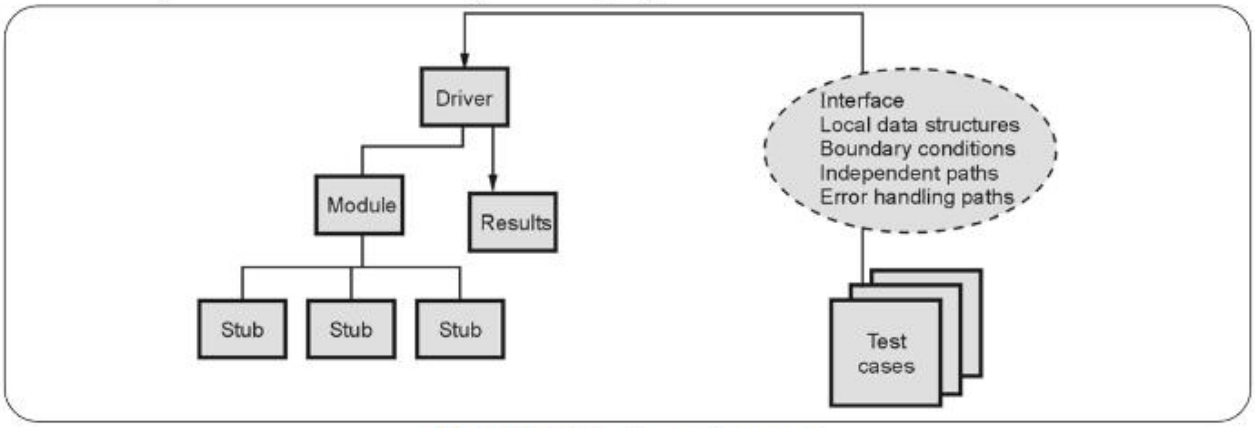

Fig. 2.2.2 Unit testing environment

7. The unit testing is simplified when a component with high cohesion (with one function) is designed. In such a design the number of test cases are less and one can easily predict or uncover errors.

2.2.1 Errors Identified during Unit Testing

Various data structure errors that can be identified during the unit testing are -

1. Incorrect arithmetic precedence.
2. Mixed mode operations.
3. Precision inaccuracy.
4. Comparison of different data types.

2.2.2 Unit Testing and Debugging

Many developers think that unit testing and debugging are one and the same thing. But there is difference between the two. The comparison between them is as follows -

Unit testing	Debugging
Unit testing is a process in which the bug is identified from the software unit.	Debugging is a process in which the bug or error is corrected by the programmer.
Unit testing starts with the execution results from the test cases.	Debugging starts after the testing process.
Test cases are defined for unit testing.	The test cases are not defined during debugging.
For the valid and invalid set of inputs the testing is done.	During debugging the focus is whether the program works for the valid set of inputs.

2.2.3 Driver and Stub

- Stubs and drivers are two such elements used in software testing process, which act as a temporary replacement for a module.
- The driver and stubs are generally dummy codes or fake codes that help to simulate the behavior of existing code.
- The stubs and drives are specifically developed to meet the necessary requirements of the unavailable modules and are useful in getting expected results.
- Generally driver and stubs are used in unit testing or during integration testing.

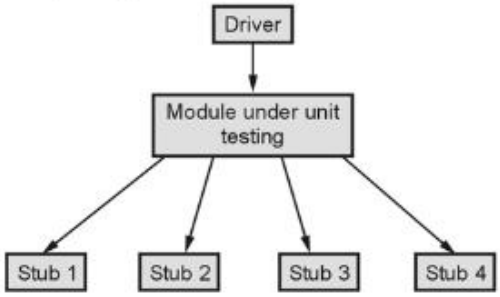

Fig. 2.2.3 Driver and stub

- **Stub**
  - These are dummy code.
  - These are used in testing when lower level of code are not developed.
  - To test the upper level of code the stubs are used.
  - It just accepts the value from calling module and returns null value.
- **Driver**
  - These are dummy code.
  - These are used in testing when upper level of code is not developed.
  - To test the lower level code the drivers are used.
  - Basically we call the driver as main function which calls other modules to form complete applications.

Board Questions

1. *Describe how drivers and stubs can be used in unit testing with neat diagrams.*
   MSBTE : Winter-16, Marks 4
2. *Explain unit testing. State its additional requirements.*
   MSBTE : Summer-16, Marks 4
3. *With the help of neat diagram, describe unit testing.*
   MSBTE : Summer-18, Winter-18, Marks 4
4. *Explain the need of stubs and drivers with diagram and its importance in software testing.*
   MSBTE : Summer-19, Marks 4

2.3 Integration Testing

- A group of dependent components are tested together to ensure their quality of their integration unit.
- The objective is to take unit tested components and build a program structure that has been dictated by software design.
- The focus of integration testing is to uncover errors in :
  - Design and construction of software architecture.
  - Integrated functions or operations at subsystem level.
  - Interfaces and interactions between them.
  - Resource integration and/or environment integration.
- The integration testing can be carried out using two approaches.
  1. The non-incremental integration
  2. Incremental integration
- The non-incremental integration is given by the **"big bang"** approach. All **components are combined in advance**. The **entire program is tested** as a whole. And chaos usually results. A set of errors is tested as a whole. Correction is difficult because isolation of causes is complicated by the size of the entire program. Once these errors are corrected new ones appear. This process continues infinitely.

Advantage of big-bang : This approach is simple.

Disadvantages :

1. It is hard to debug.
2. It is not easy to isolate errors while testing.
3. In this approach it is not easy to validate test results.
4. After performing testing, it is impossible to form an integrated system.

- An incremental construction strategy includes

1. Top down integration
2. Bottom up integration
3. Bi-directional integration

2.3.1 Top Down Integration

- Top down testing is an incremental approach in which modules are integrated by moving down through the control structure.
- Modules subordinate to the main control module are incorporated into the system in either a depth first or breadth first manner.
- Integration **process** can be performed using **following steps**.

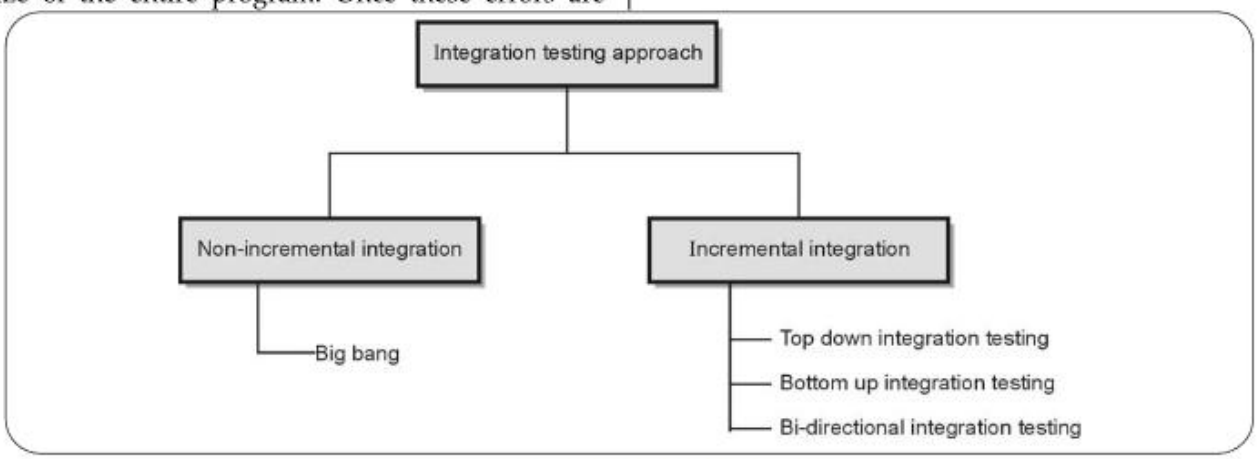

Fig. 2.3.1 Integration testing approach

1. The main control module is used as a test driver and the stubs are substituted for all modules directly subordinate to the main control module.
2. Subordinate stubs are replaced one at a time with actual modules using either depth first or breadth first method.
3. Tests are conducted as each module is integrated.
4. On completion of each set of tests, another stub is replaced with the real module.
5. Regression testing is conducted to prevent the introduction of new errors.

For example :

In top down integration if the depth first approach is adopted then we will start integration from module M1 then we will integrate M2 then M3, M4, M5, M6 and then M7.

If breadth first approach is adopted then we will integrate module M1 first then M2, M6. Then we will integrate module M3, M4, M5 and finally M7.

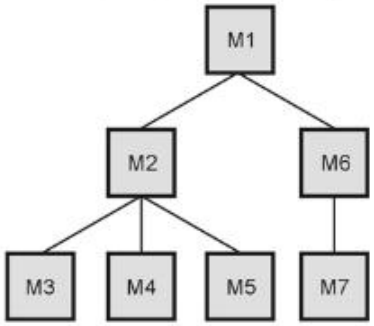

Fig. 2.3.2 Program structure

Advantages of Top Down Integration Testing

1. The major problems can be detected at the early stage of the module as the top most layer is created first.
2. We can create partially working framework (prototype) to the customers as well as to the top management and the flaws can be detected immediately by taking input from the customer.
3. As the framework is created at the topmost layer the feasibility of the program can be detected at the early stage.

Disadvantages of Top Down Integration Testing

1. The top level modules can not be tested perfectly as every time stubs are replaced with real modules.
2. It is difficult to test the units and modules alone before integration. Hence the problems might remain in the individual unit
3. Stubs need to be written and tested before they can be used for integration testing. These tested stubs are then finally getting replaced by the actual module. Hence creating and testing large number of stubs and making them defect free is actually time consuming.

2.3.2 Bottom Up Integration

In bottom up integration the modules at the lowest levels are integrated at first, then integration is done by moving upward through the control structure.

The bottom up integration **process** can be carried out using **following steps**.

1. Low-level modules are combined into clusters that perform a specific software subfunction.
2. A driver program is written to co-ordinate test case input and output.
3. The whole cluster is tested.
4. Drivers are removed and clusters are combined moving upward in the program structure.

For example :

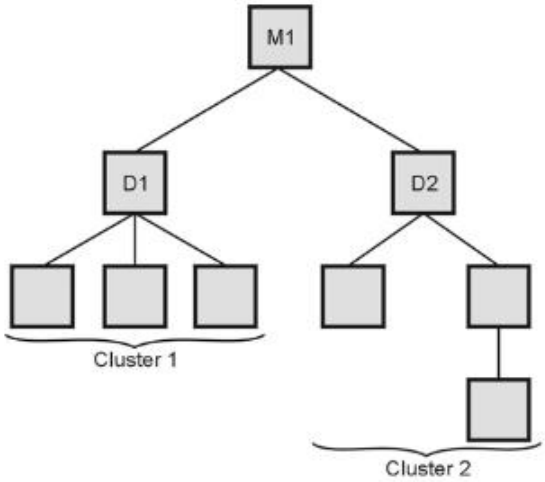

Fig. 2.3.3 Bottom up integration testing

First components are collected together to form cluster 1 and cluster 2. Then each cluster is tested using a driver program. The clusters subordinate the driver module. After testing the driver is removed and clusters are directly interfaced to the modules.

Advantages of Bottom up Integration Testing

1. Starting at the bottom of hierarchy means critical modules are created and tested first. If these modules are working perfectly then only they are integrated and move upward in the hierarchy.
2. As stubs are created first, test conditions can be easily identified or created.
3. The test results are easy to observer and therefore the robust system can be created.

Disadvantages of Bottom up Integration Testing

1. The program as an entity does not exist until the last module is added.
2. Stubs and drivers need to be written before using them into integration testing. It can be waste of time as they are not going to be the final working module of the program.
3. As the testing begins from the bottom, there is no partially working module that can be represented to the customer.
4. Any form of errors in the interface can be caught at the later stage of the testing as we move upwards towards the top most layer

Difference between Top Down and Bottom UP Testing

Sr. No.	Top-down integration	Bottom-up integration
1	The major controls or decisions are verified at early stage itself.	After integrating all the components at the bottom level, major controls or decisions can be verified.
2	Testing can be performed from the early stage.	Individual components can be tested at higher level, but after integration of low level components into cluster testing is required.
3	In top-down testing, testing stubs are created.	In bottom-up testing, test drivers are required.
4	Working system can be available at early stage.	Working system is available only after integrating all the components.

2.3.3 Bi-Directional Integration

- Bi-directional integration is a combination of the top down and bottom up integration testing approaches used together to derive the integration process.
- It is also called as **sandwich integration**.
- **Process** of bi-directional testing is as follows -
  - In this type of testing, the testing begins from the middle layer.
  - When the bottom up approach is used then testing is done from middle layer to the top layer.
  - When the top down approach is used then testing is done from middle layer to bottom layer.
- For example - Consider the program structure as given in Fig. 2.3.4.

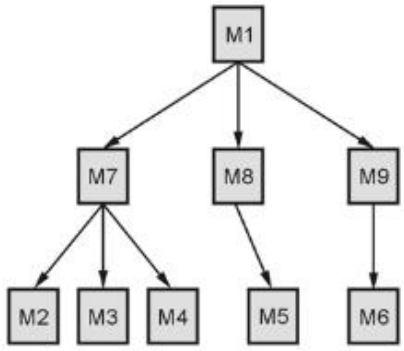

Fig. 2.3.4 Program structure

The modules M1, M2, M3, M4, M5 and M6 will be tested separately. The bidirectional integration is performed initially with the help of stubs and drivers.

Using stub the downward modules are connected and using drivers the upward modules are connected. After testing the functionality of these integrated components, the drivers and stubs are discarded. When the modules M7, M8 and M9 becomes available, the actual components are tested and integrated. The integration can be carried out in following steps.

Bottom up integration approach :
(M7-M2-M3-M4), (M8-M5), (M9-M6)

Top down integration approach :
(M1-M7-M2-M3-M4), (M1-M8-M5), (M1-M9-M6)

Advantages of Bi-directional Integration

1. Top down and bottom up layers can be handled in parallel.
2. This type of integration requires less number of stubs and drivers.
3. As this integration has better coverage control, this integration technique can be used for handling the large projects.
4. Construction of tests cases is easy.
5. Integration can be done as soon as the component is implemented.

Disadvantages of Bi-directional Integration

1. This approach still needs to throw away the stubs and drivers as the actual component is available.
2. The different skill sets of tester are required as it is a combined approach.
3. This type of integration is not suitable for smaller projects.
4. The cost involved in this testing is more than top down or bottom up approach.

Board Questions

1. *State and explain top-down approach of integration testing with diagram.* **MSBTE : Winter-15, Marks 4**
2. *Describe top-down integration testing with labelled diagram.* **MSBTE : Summer-17, Marks 4**
3. *What is integration testing ? List types of integration testing and explain any four types in brief.* **MSBTE : Winter-17, Marks 8**
4. *State process of bi-directional integration testing with two advantages and two disadvantages.* **MSBTE : Summer-15, Marks 4**
5. *Explain the integration testing and its two types. In each type, explain with example the steps of integration.* **MSBTE : Summer-16, Marks 8**
6. *Describe bi-directional/sandwitch integration testing with neat diagram.* **MSBTE : Winter-16, 18, Marks 4**
7. *Illustrate process of bi-directional integration testing. State its two advantages and disadvantages.* **MSBTE : Summer-17, Marks 4**
8. *Explain following concepts related to integration testing with neat and labelled diagram :*
   (i) Top-down testing. (ii) Bottom-up testing. **MSBTE : Summer-18, Marks 8**
9. *Explain bi-directional integration.* **MSBTE : Summer-19, Marks 4**

2.4 Testing on Web Application

- Testing strategy suggests to use following basic principles of software testing -
  1. The content model must be reviewed in order to uncover the errors.
  2. The interfaces model is reviewed to ensure all the use cases.
  3. The design model is reviewed to uncover navigation errors.
  4. User interface must be tested to remove the navigation errors.
  5. For selected function components unit testing is done.
  6. Navigation must be tested for the complete web architecture.
  7. The web application is tested in different environmental configuration.
  8. Security tests must be conducted to exploit vulnerabilities of the web application.
  9. Performance of the web application must be tested using performance testing.
  10. Finally controlled and monitored group of users must test the web application.

- The web application evolves continuously and hence it is an on-going activity.
- There are four approaches of web application testing and those are -

1. Performance testing
2. Load testing
3. Stress testing
4. Security testing

2.4.1 Performance Testing

Performance is the basic requirement of any product and it is a major concern for testing as well. Performance testing is done to check whether system meets its performance requirements under the normal load.

Definition of performance testing : Performance testing is a kind of testing performed to evaluate the **response time, throughput, and utilization of the system** to execute the required functionality in comparison with the competitive products.

Why to perform the performance testing ?

Following are few reasons for performance testing -

1. Performance testing is carried out to check if the product is **available or running** on different load conditions.
2. It is done to check that the **product responds fast enough** on different load conditions.
3. It is carried out to ensure that the required number of **transactions can be processed** by the product at any given interval.
4. The performance testing can be carried out to decide the kind of resources required by the system for different load conditions.
5. It is done to compare the product with other versions or **other competitive products** in the market.

2.4.2 Process for Performance Testing

Efforts required for the performance testing are more. Also there is a need for multiple resources during the performance testing. Hence performance testing is **costly**. It is also repetitive testing as the test cases in performance testing need to be evaluated repeatedly for different resources at varying time interval.

Fig. 2.4.1 represents the process of performance testing. (See Fig. 2.4.1 on next page)

1. **Requirements gathering and analysis :** The requirements for the performance testing are changing frequently for the performance. Majority of performance issues require rework or changes in the architecture or design. Hence it is important to collect requirements. It is equally important to test all those requirements against various performance critera.
2. **Create performance test plan :** The test plan documents the issues that need more focus during the testing or during the design of test cases. The test plan includes -
   a. **Resource requirements** : All the required resources must be listed.
   b. **Required test set up or environment** : Performance testing requires large number of test resources with special configuration which is to be documented in the test plan.
   c. **Responsibilities :** Changes in the architecture or design, or the requirements changes need to be documented promptly and must be conveyed to all the team members.
   d. **List of traces or audits :** Schedule for regular traces and audits to evaluate the work must be documented in the test plan.
   e. **Entry and exit criteria** : Execution of performance test case occurs only after meeting the performance criteria. This set of criteria must be planned well in advance. Similarly the exit criteria is needed to conclude the result of test case execution. The set of exit criteria must be present in the test plan.
3. **Design and automate test Case :** The test case must be designed based on the performance requirements. The performance testing is always carried out with the help of some suitable automation tool.

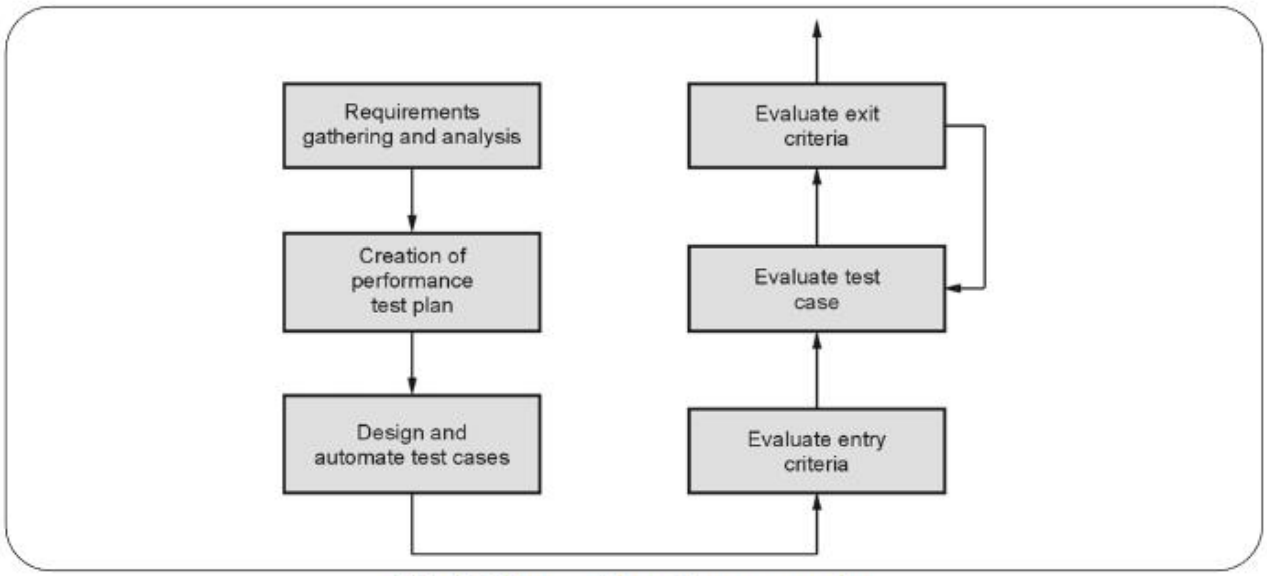

Fig. 2.4.1 Process for performence testing

4. **Evaluate entry criteria :** During performance testing when **certain performance criteria get satisfied** then the performance test begins its execution. This is called entry criteria. The entry criteria must be evaluated at regular intervals during the performance testing. There is separate set of criteria for each of performance test case.
5. **Evaluate test case :** Execute and evaluate each test case.
6. **Evaluate exit criteria :** After execution of the test case product is evaluated to see whether it has met all the required performance criteria after test case execution. If the criteria are not met then the test case is again executed. This process is repeated until the performance criteria are met. The results are collected after evaluation of exit criteria.

Thus performance testing is conducted. This is one of the crucial testing process.

2.4.3 Load Testing

- Load testing is a testing process used to check how systems behave under normal or peak load conditions.
- Load testing gives the confidence in system about its reliability and performance.
- It also helps in identifying the bottlenecks in the systems under heavy workload conditions.
- Load testing helps in identifying -
  - Response time for each transaction
  - Performance of system components under load.
  - Network delay between client and server.
  - Server configuration issues such as web server, application server, database server

Advantages of load testing :

1. Performance bottlenecks identification before production
2. Improves the scalability of the system
3. Minimize risk related to system down time
4. Reduced costs of failure
5. Increase customer satisfaction

Disadvantages of load testing :

1. Need programming knowledge to use load testing tools.
2. Tools can be expensive as pricing depends on the number of virtual users supported.

2.4.4 Stress Testing

- Determines breakpoint of a system to establish maximum service level.

- In stress testing the system is executed in a manner that demands resources in abnormal quantity, frequency or volume.
- A variation of stress testing is a technique called sensitivity testing.
- The sensitive testing is a testing in which it is tried to uncover data from a large class of valid data that may cause instability or improper processing.

Difference between load testing and stress testing

Sr. No.	Load testing	Stress testing
1.	This is a testing method that helps to determine reliability of application.	This is a testing method that helps to observe stability of application.
2.	It finds out the performance and behavior of application under various load	It finds the breaking point of the server.
3.	The application is tested with maximum number of users.	The application is tested with more than maximum number of users.

Advantages

1) It finds out if data can be corrupted by overstressing the system.
2) It helps to determine what kind of failures are most valuable to plan for.
3) Offers assistance in determining the stability of the software system beyond its normal operational capacity.
4) Makes the software crash-proof in scenarios where the computational resources are scarce.
5) It helps to find deadlocks in software product.

2.4.5 Security Testing

Definition of security testing : Security testing is a testing technique to determine if an information system protects data and maintains functionality as intended.

- Security testing is done to protect the application against unfortunate incidences. The unfortunate incidences could be loss of privacy, loss of data, modification in system data and so on.
- The unfortunate incidences could be internal or external.
- During security testing system is basically finding out all possible loopholes and weakness which might result in vulnerability to outside attacks or unauthorized entry in the system.

Objectives of Security Testing

- The goal of security testing is to identify the threats in the system and measure its potential vulnerabilities.
- Security testing helps in improving the current system and also help in ensuring that the system will work for longer time.
- Security testing helps in finding out loopholes that can cause loss of important information.

Security Testing Technique

The main security testing techniques are

1. **Vulnerability scanning :** The vulnerabilities are scanned using automated testing.
2. **Security scanning :** In this scanning, the network and system weaknesses are identified and solutions are obtained. This scanning is done using both automated and manual testing.
3. **Penetration testing :** In this testing, tester tries to peep into the system via the loophole that application has kept unknowingly. It is one kind of simulation of attack from malicious hacker.
4. **Ethical hacking :** In this type of testing, the number of penetration tests are conducted over the wide network on the system under test with the intention to expose security flaws in the system.

5. **Risk assessment :** In this testing, the analysis of security risks is done with the help of interviews, discussion and analysis.
6. **Security auditing :** It is a technique in which internal inspection of operating system and application is done by line by line inspection of code.
7. **Password cracking :** There are some special programs that can be used to identify the weak passwords.

Common Areas of Security Testing

There are common area for which the security testing is must. These are as discussed below -

1. **For E-commerce systems** high protection for the user is required in order to protect their privacy and information involving financial transactions.
2. **Communication systems** require the protection from loss of data during the transmission from one system to another.
3. **The database systems**, or business logic systems are prone to threats as penetrators can attack and modify the data present in such systems. Hence system testing must be carried out for such systems.

Board Questions

1. *How performance testing is performed ? List steps involved in it ?*
   MSBTE : Winter-15, Marks 4
2. *Explain the performance testing and its criteria.*
   MSBTE : Summer-16, winter-18, Marks 4
3. *Describe any four testing approaches of web application.*
   MSBTE : Winter-16, Marks 4
4. *What is load testing and stress testing ? Describe with respect to system testing.*
   MSBTE : Winter-16, Marks 4
5. *Describe how to perform security and performance testing.*
   MSBTE : Summer-15, Marks 4
6. *Describe following testing with example. (1) Security testing (2) Performance testing*
   MSBTE : Summer-15, Marks 4
7. *How to perform security testing ? State elements of security testing.*
   MSBTE : Summer-17, Marks 4

2.4.6 Client Server Testing

- This is a type of testing which is applied for 2 tier applications. The two tier includes front end and back end.

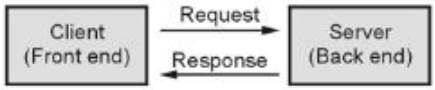

Fig. 2.4.2 Two tier architecture

- In client server applications - the front ends are developed in VB, VC++, Java and so on. The backend are developed as the database systems in MS ACCESS, MySQL, Oracle and so on.
- Testing of client server architecture is **very difficult because of following reasons** -
  1. Distributed nature of client server
  2. Performance issues of different hardware platforms
  3. Complexities of communication network
  4. Need for servicing multiple clients
  5. Database access by multiple clients and their transaction processing.

Features of Client Server Application

- This application runs in two or more machines
- Application is menu driven
- There are limited number of users for this application
- It has less number of network issues when compared to web app.
- The client and server work in a connected mode.

Types of Tests Conducted for Client Server Architecture

Various approached of testing the client server architecture are as follows -

(1) Application function test : The standalone application function is tested in this type.

(2) Server test : Co-ordination and data management activities are tested. The server performance is also tested

(3) Database test and transaction tests : The integrity of data store and performance of transaction processing are tested.

(4) Network communication test : The communication among various nodes, performance and network security are tested.

Board Questions

1. *With the help of diagram describe client-server testing.* **MSBTE : Summer-15, Winter-17, 18, Marks 4**
2. *Explain the client-server application testing.* **MSBTE : Summer-16, 18, Marks 4**
3. *List any four features of client-server application. Explain any four testing approaches of client - server testing.* **MSBTE : Winter-16, Marks 8**
4. *Describe use of load testing and stress testing to test online result display facility of MSBTE website.* **MSBTE : Winter-17, Marks 6**

2.5 Acceptance Testing

- Acceptance testing is a testing phase which is conducted normally after system testing by the customer or representative of customer.
- The **goal of acceptance testing** is to establish the confidence in the system.
- The acceptance testing **defines criteria for acceptance** or rejection of the software.
- The acceptance test cases are black box type test cases. They make use of one or more **acceptance criteria** for accepting the product.

Importance of Acceptance Criteria

1. Acceptance criteria are the **conditions** that a software product must satisfy to be accepted by a user, customer, or in the case of system level functionality.
2. The acceptance criteria is very useful in acceptance testing because it represents the customers' expectations from the system.
3. It also gives the critical success factor to determine whether purchasing such products will be useful or not.
4. Acceptance criteria determines the confidence level in the final deliverable of the product.
5. It defines clearly whether the system meets predefined attributes or not.

Types of Acceptance Criteria

1. **Product acceptance :** Each requirement is normally associated with the acceptance criteria. Whenever there is change in requirements the acceptance criteria must be modified accordingly.
2. **Procedure acceptance :** Acceptance criteria can be defined based on procedures followed for delivery of the product. For example - 1. User manual, troubleshooting guides should be the part of the release. 2. The training should be provided to the users of the system before the onsite deployment.
3. **Service level agreements :** The service level agreement can also become the part of acceptance criteria. It is sort of contract signed between customer and owner of the product. For example - If the defects occur within first 2 months of deployment then those need to be fixed free of cost. This is a service level agreement.

2.5.1 Alpha Testing

Alpha testing is a kind of acceptance testing in which version of complete software system is tested by the customer under the supervision of developer. This type of testing is performed at developer's site. The software is used in natural settings in presence of developer. This test is actually conducted in controlled environment.

Advantages

1. The users of the system get systematic training to use the new system.
2. The acceptance of the system can be immediately known to the developers.

3. The primary problems in the identified and fixed in immediately before delivery of the product to the customer.
4. Any mis-understanding about the scope of the project or in requirements given by the customer can be cleared during this type of testing.

Disadvantages

1. In depth functionality cannot be tested as software is still under development stage. Sometimes developers and testers are dissatisfied with the results of alpha testing.
2. Alpha testing is conducted in lab environment or testing environment which may not be similar to real-life environment.
3. Data used during this testing may not be similar to the business data or actual data. Hence false image of working module can be created and actual problems remain hidden.

2.5.2 Beta Testing

The beta testing is a kind of testing in which the version of complete software is tested by the customer **without the developer being present**. This testing is done at Customer's site. As there is no presence of developer during testing, it is not controlled by developer. The end user records the problems and report them to developer. The developer then makes appropriate modifications.

Advantages :

1. Beta testing improves the product quality via customer feedback.
2. Any training requirement for use of the product can be identified during beta testing.
3. The testing is done in actual work environment without the interference of developer. Hence addition in requirements, changes in the requirements can be identified.
4. The security loopholes can be identified by the customer itself.
5. Beta testing allows the organization to test post-launched infrastructure.
6. Due to customer feedback product failure risk can be reduced.
7. Usability of the application can be identified and verified by the customer.

Disadvantages :

1. Finding right beta user and maintaining his participation can be real challenge.
2. Tests can not be management because beta testing is conducted in actual working environment and not in the organization.

Difference between Alpha and Beta Testing

Sr. No.	Alpha testing	Beta testing
1	Performed at developer's site	Performed at end user's site
2	Performed in controlled environment as developer is present	Performed in uncontrolled environment as developer is not present
3	Less probability of finding of error as it is driven by developers.	High probability of finding of error as end user can use it the way he wants.
4	It is not considered as live application	It is considered as live application
5	It is done during implementation phase of software	It is done as pre-release of software
6	Less time consuming as developer can make necessary changes in given time.	More time consuming. As user has to report bugs if any via appropriate channel.

Board Questions

1. *List any four advantages of acceptance test before launching any software.*
   MSBTE : Winter-15,18, Marks 4
2. *Explain the alpha testing. State its limitations.*
   MSBTE : Summer-16, Marks 4

3. *Compare alpha testing and beta testing. (Any four differences).*
MSBTE : Winter-16, 17, 18, Summer-18, Marks 4
4. *Describe alpha testing with its entry and exit criteria.*
MSBTE : Summer-17, Marks 4
5. *Differentiate between alpha testing and beta testing.*
MSBTE : Summer-19, Marks 4

2.6 Special Tests

2.6.1 Regression Testing

- Regression testing is used to check for defects propagated to other modules by changes made to existing program. Thus regression testing is used to reduce the side effects of the changes.
- There are three different classes of test cases involved in regression testing -
- Representative sample of existing test cases is used to exercise all software functions.
- Additional test cases focusing software functions likely to be affected by the change.
- Tests cases that focus on the changed software components.
- After product had been deployed, regression testing would be necessary because after a change has been made to the product an error that can be discovered and it should be corrected. Similarly for deployed product addition of new feature may be requested and implemented. For that reason regression testing is essential.

Board Question

1. *Explain the regression testing. State when the regression testing shall be done.*
MSBTE : Summer-16, Winter-17, Marks 4

2.6.2 GUI Testing

Testing the graphical user interface is very challenging because GUI consists of many reusable components. This actually increases the complexity of GUI and leads to more difficulty in design and execution of test cases.

Approaches of GUI Testing

Following are some ways to test the GUI -

1. **Use of standard tests :** Many modern GUIs have same look and feel hence a series of standard tests can be derived.
2. **Use of finite state modeling graph :** The finite state modeling graphs can be created for testing specific data and program objects of GUI.
3. **Use of automated tools :** Due to large number of permutations in GUI applications the automated tools are used for testing.

Checklist for GUI Testing

1. Check all the GUI elements for size, position, width, length and acceptance of characters or numbers.
2. Check you can execute the intended functionality of the application using the GUI
3. Check error messages are displayed correctly
4. Check for clear representation of different sections on screen
5. Check font used in application is readable
6. Check the alignment of the text is proper
7. Check the color of the font and warning messages is aesthetically pleasing
8. Check that the images have good clarity
9. Check that the images are properly aligned
10. Check the positioning of GUI elements for different screen resolution.

Advantages of GUI Testing

1. Good GUI help the user to give the better experience of handling the application.
2. The user can readily accept the application if the GUI is well tested and working in appropriate manner.
3. The availability of help feature allows even the novice user to handle the application in the required manner.
4. Consistency in the page layout and arrangement improves the usability of the application.

5. User can understand the behavior of the system with the help of good GUI.

Disadvantages of GUI Testing

1. Special users such as blind, or underage users can not handle the application using GUI.
2. If the spelling or arrangement mistakes occur in GUI then it affects the overall aesthetics of the application.
3. Tester give least importance to the GUI testing and more attention is paid to functionality of the system component. If the GUI is not well tested then it may lead to confusion in handling the application.
4. If the GUI testing is not done properly, then consistency in layout and arrangement of documents is lost.

Board Questions

1. *Describe Graphical User Interface (GUI) testing and its important traits.*
   MSBTE : Winter-15, Marks 4
2. *Explain GUI testing with suitable example.*
   MSBTE : Summer-15, Marks 6, Winter-17, Marks 4
3. *Describe any two special tests in testing process.*
   MSBTE : Summer-17, Marks 4
4. *Explain GUI testing with example.*
   MSBTE : Summer-19, Marks 4

❑❑❑

Notes

UNIT - III

3 Test Management

3.1 Test Planning

Testing is an important phase which can be planned, executed, tracked and periodically reported on.

Definition of test plan : Test plan is a detailed document that outlines the test strategy, testing objectives, resources such as manpower, software, hardware required for testing, test schedule, test estimation and test deliverables.

The test plan serves as a blueprint to conduct software testing activities.

3.1.1 Preparing a Test Plan

Test plan is an important document for execution, tracking and reporting the entire testing.

Following things are required to prepare the test plan.

Step 1 : Write the **Scope of testing** including the clear indication of what will be tested and what will not be tested.

Step 2 : Testing is performed by **breaking it down into small and manageable tasks** and identifying strategies to be used for carrying out the tasks.

Step 3 : Find out the **resources needed** for testing.

Step 4 : Design **timeline** by which testing activities will be performed.

Step 5 : List of **Risks faced** in all of the above steps.

Advantages of Test Plan

1. Serves as a guide to testing throughout the development.
2. Test plan defines test points during the testing phase.
3. Serves as a valuable record of what testing was done.
4. Test plan ensures all functional and design requirements as specified in the documentation.

Standard Template for Test Plan

1. Introduction
   1.1 Scope
2. References
3. Test methodology and Strategy/Approach
4. Test criteria
   4.1 Entry criteria
   4.2 Exit criteria
   4.3 Suspension criteria
   4.4 Resumption criteria
5. Assumptions, dependencies and risks
   5.1 Assumption
   5.2 Dependencies
   5.3 Risk and risk management plans
6. Estimations
   6.1 Size estimate
   6.2 Effort estimate
   6.3 Schedule estimate
7. Test deliverables and milestones
8. Responsibilities
9. Resource requirement
   9.1 Hardware resources
   9.2 Software resources
   9.3 People resources
   9.4 Other resources
10. Training requirements
11. Defect logging and tracking process
12. Metrics plan
13. Product release criteria

Board Questions

1. *What is test plan ? List test planning activities.* **MSBTE : Summer-15, 18, Marks 4**
2. *How to prepare a test plan ? Explain in detail.* **MSBTE : Summer-15, 17, Marks 4**
3. *State the contents of standard template of a test plan.* **MSBTE : Summer-16, Marks 4**
4. *What is test plan ? Write its two advantage.* **MSBTE : Summer-17, Marks 4**
5. *Give any four advantages of test planning.* **MSBTE : Summer-18, Marks 4**
6. *Explain how a test plan is prepared.* **MSBTE : Summer-19, Marks 4**

3.1.2 Deciding Test Approach

- Test approach is a part of test plan. It identifies right type of testing required for the system.
- Test approach is a kind of approach in which we have to decide what needs to be tested in order to estimate size, effort and schedule. This includes identifying following factors -
  - What type of testing is to be used for testing functionality ?
  - What scenarios are required for testing the features ?
  - What kind of integration testing is required ?
  - What localization validations would be needed ?
  - What kind of non functional tests(tests related to reliability, performance, security and so on) are required ?

Board Question

1. *Describe the factors considered to decide test strategy or test approach.* **MSBTE : Summer-18, Marks 4**

3.1.3 Setting up Criteria for Testing

- There are various criteria used for testing such as entry criteria, exit criteria, suspension and resumption criteria.
- **Entry criteria** for test specify threshold criteria for each phase or type of test. There must be entry criteria for entire testing activity to start.
- The **exit criteria** specify when test cycle can be completed.
- Testing can be suspended at various points of time and then can be resumed.
- Some of the **suspension criteria** include -
  - Encountering more than a certain number defects, causing frequent stoppage of testing activity.
  - Hitting show stoppers that prevent further progress of testing.
  - Developers releasing a new version.
- When such conditions are addressed, the tests can resume.

Board Question

1. *Why is it essential to setup criteria for testing ? List any three criteria in different situations.* **MSBTE : Winter-15, Marks 4**

3.1.4 Identifying Responsibilities, Staffing and Training Needs

- Testing process requires different people to play different roles. The most common roles are - test engineers, test leads and test managers.
- Each role must accomplish certain responsibilities.
- **How to identify responsibilities ?**

  Following guideline need to be considered for identifying the responsibilities -

  1) Each person should know clearly what he or she should has to do.
  2) List out the responsibilities of each function in the testing process.
  3) Everyone should know how his work is important in performing testing activity.
  4) Complement and co-operate each other.
  5) No task should be left unassigned.
- **Staffing**
  - Staffing is done based on estimation of effort and time available for the project for completion.
  - The tasks in testing are prioritized on the basis of effort, time and importance.
  - The people are assigned to tasks based on the requirements of job, skills and experience of people.

- If people are not having the skills for specific requirements of the job then appropriate training programs are needed.

Board Question

1. *Describe how to identify responsibilities in testing.*
MSBTE : Summer-15, Marks 4

3.1.5 Identifying Resource Requirements

- While planning for testing the project, the project manager should get the estimates of hardware and software requirements.
- Following factors shall be considered while selecting resource requirements -
  - RAM, processor, disk required for the system under test.
  - Test automation tool - if required.
  - Supporting tools such as compiler, test configuration management tool and so on.
  - Appropriate number of licenses of all the software.
  - Operating systems required for the execution.
  - Some machine intensive tests such as load tests and performance tests.

Additional Resource Requirements …

- Office space
- Some supporting staff such as HR.

Board Questions

1. *What factors shall be considered while selecting resource requirements ?*
MSBTE : Winter-16, Marks 4
2. *How to identify resource requirement of test plan ?*
MSBTE : Summer-17, Marks 4
3. *Explain the need of staff training and resource requirements in test planning in software testing.*
MSBTE : Summer-19, Marks 8

3.1.6 Identifying Test Deliverables

- **Definition of test deliverables :** Test deliverables are the artifacts which means "things" that are produced by people involved in the process and are given to the stakeholders.
- **Milestones** are the indicates about the completion of key project tasks.
- Some common test deliverables are -
  1) Test plan itself
  2) Test case design specifications
  3) Test cases
  4) Test logs produced by running the tests
  5) Test summary report.

Board Questions

1. *Explain test deliverables in details.*
MSBTE : Summer-15, Winter-17, Marks 4
2. *What is test deliverables and milestones ? Explain any four test deliverables.*
MSBTE : Summer-16, Winter-18, Marks 6
3. *Explain the need of test deliverables for test planning.*
MSBTE : Summer-19, Marks 4

3.1.7 Testing Tasks

- Various tasks of tester are -
  - Identify bugs and errors, describe and send for revision.
  - Check the implemented improvements.
  - Test the product until the desired result is achieved.
  - Test the software solution on all necessary devices and screen resolutions.
  - Test the product compliance with the requirements.
- **Estimation** is one of the important testing task. There are three phases of estimation -
  - Size estimation
  - Effort estimation
  - Schedule estimation

1) Size estimation

- It can be defined as actual amount of testing that need to be done.
- Size estimated is expressed in terms of
  - Number of test cases.
  - Number of test scenarios.
  - Number of configurations to be tested.

2) Effort estimations

- Effort estimation is derived from size estimation. For example - if some test case is reused in the project from existing standard library of test cases then effort involved in developing the test cases is zero.
- Effort estimate can be given in person days, person months or person years.

3) Schedule estimation

- Effort estimation is translated into schedule estimate.
- Schedule estimation can be defined as the translation of efforts into particular time frame.
- Following are the steps required for performing schedule estimation -
  1. Identify external and internal dependencies among the activities.
  2. Sequence the activities based on expected duration.
  3. Identify time required by each activity.
  4. Monitor the progress in terms of time and efforts.
  5. Rebalance the schedule as well as resources if required.

3.2 Test Management

- Test planning is a process of identifying the test activities and resource requirements for implementation of testing strategy.
- Test management is a process of managing the test process by using series of activities such as planning, execution, monitoring and controlling.

3.2.1 Test Infrastructure Management

- The robust test infrastructure or skeleton must be planned in advance.
- There are three elements of test infrastructure

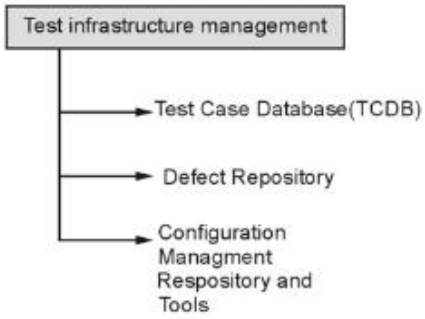

Fig. 3.2.1

1) Test Case Database(TCDB) :

Test case database is a collection of the information about test cases in an organization. The contents of test case database are as given below -

Entity	Purpose	Attributes
Test case	Records all the information about the tests.	(1) Test case ID (2) Name of test case (3) Owner of test case (4) Status
Test case-product cross reference.	Provides mapping between test case and corresponding feature of the product.	(1) Test case ID (2) Module ID
Test case run history.	Provides the history of when test was running and what was the result.	(1) Test case ID (2) Run date (3) Time (4) Status of Run (Pass or Fail).
Test case - Defect cross reference.	Details of test cases introduced to test certain defects detected in product.	(1) Test case ID (2) Defect reference.

2) Defect repository

Defect repository captures all the relevant details of defects reported for a product.

The information stored in defect repository is given as follows -

Entity	Purpose	Attributes
Details of defect	Record all static information about test.	(1) Defect ID (2) Defect priority (3) Defect description (4) Affected product (5) Environment information. (6) Customer who encountered with the defect. (7) Date and time of defect occurrence.
Details of defect test	Details of test cases for a given defect. Cross reference with the TCDB.	(1) Test case ID (2) Defect ID
Fixing details	Details of the fixes for given defect.	(1) Defect ID (2) Fix details
Communication	Details of the communication that occurs among various stakeholder during defect management.	(1) Test case ID (2) Defect Reference. (3) Details of communication.

3) Configuration management and tools :

- Software Configuration Management(SCM) repository is also known as Configuration Management(CM) repository.
- It keeps track of change control and version control of all the files present in the software product.
- **Change control** ensures that -
  - Changes made during testing of a file must be in controlled manner.
  - Changes made by one tester must not be lost or overwritten by another tester.
  - For each change made in the file appropriate version of the file must be created.
  - Everyone must get an access of most updated version of the file.
- **Version control** ensures that the test scripts associated with a given release of a product are baselined along with the product files.

Working of Test Infrastructure Components

The TCDB, SCM and DR work together by cooperating each other as shown in Fig. 3.2.2.

For example Defect Repository(DR) links defects, fixes and tests. These files are present in SCM. The metadata about the modified files is present in TCDB and find the corresponding test case files and source files from SCM.

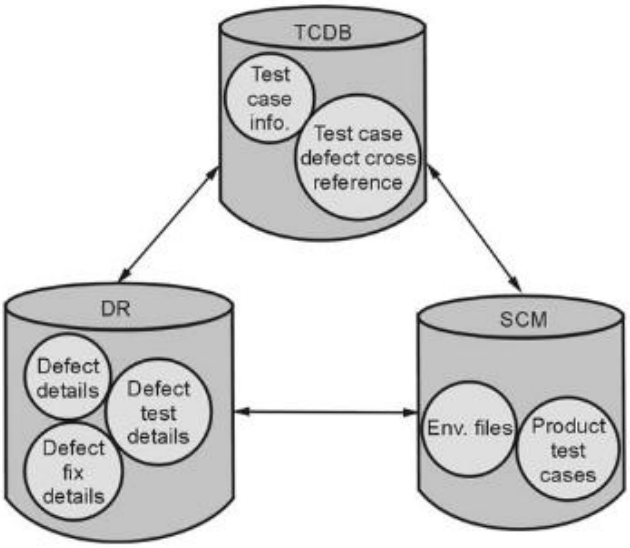

Fig. 3.2.2 Components of test infrastructure

Board Questions

1. *What is test planning and test management ?* **MSBTE : Summer-16, Winter-17, Marks 4**
2. *Explain the 'Test Infrastructure' components with diagram.* **MSBTE : Summer-16, Winter-17, Marks 4**
3. *Write the entity, purpose and attributes of following elements of test infrastructure management :*
   (i) A Test Case Database (TCDB)
   (ii) A defect repository. **MSBTE : Winter-16, Marks 8**
4. *Explain test infrastructure management with its components.* **MSBTE : Summer-19, Marks 4**

3.2.2 Test People Management

- People management is an important aspect of any project.
- Good people help in
  (1) Achieving the project target.
  (2) Communicating more effectively.
  (3) Co-operate effectively.
- For people management following are general steps to be taken up -
  1. Have effective communication with the people.
  2. Build the healthy relationships.
  3. Influence people.
  4. Motivate people.
  5. Handle ethical issues properly.
- The team-building exercises should be ongoing and sustained. The effects of these exercises tend to wear out under the pressure of deadlines of delivery and quality.

Board Question

1. *Explain people management in test planning.*

MSBTE : Summer-19, Marks 4

3.3 Test Process

3.3.1 Base Lining a Test Plan

- Test plan is an important document that acts as a anchor point for entire testing project.
- Using some checklist the test plan is prepared. Each testing project puts together a test plan based on template. If some changes are required in the test plan then those changes are made only after careful deliberations.
- The test plan is then reviewed by expert people in the organization.
- It then is approved by a competent authority, who is independent of the project manager directly responsible for testing.
- After this the test plan is baselined into configuration management repository. From this test plan becomes the basis for testing process of the project.
- If any significant change is made in the testing then that should be reflected in the test plan and again this changed test plan must be in configuration management repository.

3.3.2 Test Case Specification

Using **test plan** as basis, the testing team designs **test case specification** which then becomes the basis for preparing individual test cases.

Test case be defined as series of steps executed on product using predefined set of input data, expecting to produce predefined set of outputs in a given environment.

Following things need to be identified -

(1) The purpose of test : The intention for which the test is performed is specified.

(2) Items to be tested.

(3) Software and hardware environment setup.

(4) Input data to be used for test case.

(5) Steps to be followed to execute test.

(6) Expected results that are considered to be correct result.

(7) Actual result produced after performing tests.

(8) Relationship of current test with other tests.

3.3.3 Executing Test Cases

- The prepared test cases has to be executed at appropriate times during the project. For example - system testing must be executed during system test.
- During the test case execution, the Defect Repository(DR) must be updated with -
  - Defects that are fixed currently.
  - New defects that are found.
- Test team communicates with the development team with the help of defect repository.
- Tests has to be suspended during its run and it should be resumed only after satisfying the **resumption criteria.**
- Tests should be run when **entry criteria** for the test gets satisfied and should be accomplished only when the **exit criteria** satisfied.
- On successful execution of test cases-the traceability matrix must be updated.

Ex. 3.3.1 : ***Prepare six test cases for admission form for college admission.*** **MSBTE : Summer-15, Marks 4**

Sol. :

Test case ID	Description	Test data	Expected result	Actual result	Status
TC01	Enter data for name.	Enter xyz as name.	It will accept name and prompt for next subsequent field.	It accepts name and prompt for next subsequent field.	Pass
TC02	Enter data for address field.	Fno = 101, Building name "ABC",Model colony, S.B. Road, Pune.	It will accept address and prompt for next subsequent field.	It accepts address and prompt for next subsequent field.	Pass
TC03	Enter data for invalid phone number.	Any data more than 10 digits or other than a numeric value is entered.	It will display 'Invalid Phone number, please enter again'.	It displays 'Invalid Phone number, please enter again'.	Pass
TC04	Enter data with valid phone number.	Valid phone number of 10 digits.	It will accept phone number and prompt for next subsequent field.	It accepts phone number and prompts for next subsequent field.	Pass
TC05	Enter valid marks.	Marks = 94.08 %	It will prompt to click submit button and on clicking submit button admission procedure page will be displayed.	It prompts to click submit button and then displays the college admission procedure page.	Pass
TC06	Marks field is left blank.	No data in marks field.	It will display 'Please enter marks'.	It displays 'Please enter marks'.	Pass

Ex. 3.3.2 : ***Write important six test cases for the 'Login Form' of the facebook website.***

MSBTE : Summer-15, 16, Marks 4

Sol. :

Test case ID	Description	Test data	Expected result	Actual result	Status
TC01	Invalid username is entered.	Enter xyz as username.	It will prompt 'couldn't find your account' message.	It prompts couldn't find your account.	Pass
TC02	Valid username and invalid password is entered.	Enter 1111 as password.	It will display 'Wrong password' message.	It displays 'Wrong password' message.	Pass
TC03	Username field is left blank.	No data in user name field.	It will display 'enter username'.	It displays 'enter username'.	Pass

TC04	Valid username and no password is entered.	No data in password field.	It will display 'enter password'.	It displays 'enter password'.	Pass
TC05	Both user name and password field is left blank.	No data in user name and password field.	It will display 'enter username/password'.	It displays 'enter username/password'.	Pass
TC06	Valid user name and password is entered.	User name = abc and password = abc1234.	It will display your account's facebook page.	It displays your account's facebook page.	Pass

Ex. 3.3.3 : ***The student passing the diploma shall be awarded class of passing as - distinction, first class, second class, pass class. The class will be applicable only if the student has his/her result 'pass' in all of subjects. Write the test cases (minimum-4) for the class awarding algorithm (module).*** **MSBTE : Summer-16, Marks 4**

Sol. :

Test case ID	Description	Test data	Expected result	Actual result	Status
TC01	Enter marks in all the subjects in the range of 75-100.	Any value for all subjects within 75 to 100.	It will display 'Distinction'.	It displays 'Pass with Distinction '.	Pass
TC02	Enter marks in all the subjects in the range of 60 -74 %.	Any value for all subjects within 60 to 74.	It will display 'First class'.	It displays 'Pass with First Class '.	Pass
TC03	Enter marks in all the subjects in the range of 50-59.	Any value for all subjects within 5 to 59.	It will display 'Second class'.	It displays 'Pass with Second class'.	Pass
TC04	Enter marks in all the subjects in the range of 40-49.	Any value for all subjects within 40 to 49.	It will display 'Pass Class'.	It displays 'Pass with Pass Class '.	Pass
TC05	Enter the marks in one or more subjects that are less than 40.	Any value less than 40 in one or more than one subject.	It will display 'Fail'.	It displays 'Fail'.	Pass

Ex. 3.3.4 : ***Write four test cases to test sign-in form of gmail account.*** **MSBTE : Winter-16, Marks 4**

Sol. :

Test case ID	Description	Test data	Expected result	Actual result	Status
TC01	Invalid username is entered.	Enter xyz as username.	It will prompt 'couldn't find your account' message.	It prompts couldn't find your account.	Pass
TC02	Valid username and invalid password is entered.	Enter 1111 as password.	It will display 'Wrong password' message.	It displays 'Wrong password' message.	Pass
TC03	Username field is left blank.	No data in user name field.	It will display 'enter username'.	It displays 'enter username'.	Pass

TC04	Valid username and no password is entered.	No data in password field.	It will display 'enter password'.	It displays 'enter password'.	Pass
TC06	Valid user name and password is entered.	user name =abc@gmail.com and password=abc1234.	It will display your account' page.	It displays your account's page.	Pass

Ex. 3.3.5 : ***Prepare six test cases for home page of marketing site.*** **MSBTE : Summer-17, Marks 6**

Sol. :

Test case ID	Description	Test data	Expected result	Actual result	Status
TC01	Invalid user name and/or invalid password is entered.	user name ="xyz" and password=abc1234.	It will display "invalid username and/or password".	It will display "invalid username and/or password".	Pass
TC02	Valid user name and password is entered.	user name ="abc" and password=abc1234.	It will display home page for marketing site.	It displays home page for the marketing site.	Pass
TC03	Search for the desired product.	Click on the product category and then choose the desired product. For example click on "TV& Appliances" and then choose "Split Acs".	It will display the list of products along with their prices and other details.	It displays the list of products along with their prices and other details.	Pass
TC04	Select the desired product for reviews.	Click on "Voltas 1.2 Ton 5 Star Split Inverter AC" and select the ratings and reviews.	It will display both positive and negative reviews.	It displays positive and negative reviews.	Pass
TC05	Select the desired product for purchase.	Click on "Voltas 1.2 Ton 5 Star Split Inverter AC" and "Buy Now" or "Add to Cart" button.	It will navigate to order summary and then to payment option pages.	It navigates to order summary and then to payment option pages.	Pass
TC06	Make payment	Click on payment mode as "Card" and then enter card number and CVV code.	It will then authenticate the payments and will proceed only if payment is authenticated.	It authenticates the payment, if it is invalid then rejects the order otherwise proceeds for successful transaction page.	Pass

Ex. 3.3.6 : ***Prepare and write six test cases for simple calculator application.*** **MSBTE : Summer-18, Marks 6**

Sol. :

Test case ID	Description	Test data	Expected result	Actual result	Status
TC01	To perform addition	Press buttons for number 123, then press '+' then press buttons for number 456 and then press '=' button.	It will display the result of 123+456 as 579.	It displays 579 on screen.	Pass
TC02	To perform subtraction	Press buttons for number 123, then press '-' then press buttons for number 111 and then press '=' button.	It will display the result of 123-111 as 12.	It displays 12 on screen.	Pass
TC03	To perform division.	Press buttons for number 55, then press '/' then press buttons for number 11 and then press '=' button.	It will display the result of 55/11 as 5.	It displays 5 on screen.	Pass
TC04	To perform multiplication.	Press buttons for number 8, then press '*' then press buttons for number 7 and then press '=' button.	It will display the result of 8*7 as 56.	It displays 56 on screen.	Pass
TC05	To clear all the operations.	Press C button.	It will display 0.	It displays 0.	Pass
TC06	To delete some number.	Press backspace key and delete the number by deleting each digit from right to left.	The entered number will be deleted.	It deletes the number.	Pass

Ex. 3.3.7 : ***Prepare and write four test cases for library management system of college.***

MSBTE : Summer-18, Marks 4

Sol. :

Test case ID	Description	Test data /Test steps	Expected result	Actual result	Status
TC01	Valid user name and password is entered.	user name ="abc" and password=abc1234.	It will display home page for library management site.	It displays home page for library management site.	Pass
TC02	Search for the desired book.	Click on the book category and then choose the desired book. For example click on "Fiction" and then choose "The Secret".	It will display the availability of the book.	It displays "book available" if it is present in the library otherwise it will display "Book is not available" message.	Pass
TC03	Issue the books.	Click on "Book issue" and enter the details of the book and return date for the book.	It will display "Book issued" message.	It will displays "Book issued" message.	Pass
TC04	Return the books.	Click on "Book Return" and enter the details of the book and check if any fine is charged for late return.	It will display "Book returned" message.	It will displays "Book returned" message.	Pass

TC05	Membership renewal.	Click on "Renewal of Membership" button and enter the extended period and payments made.	It will display "Membership renewal made".	It displays "Membership renewal made".	Pass
TC06	Delete membership.	Click on "Close Membership" button and get the payments pending.	It will display "Membership is closed".	It displays "Membership is closed".	Pass

Board Questions

1. *How test case specifications useful in designing test cases ?* **MSBTE : Winter-15, Marks 8**
2. *What are the things that test case specification shall identify ?* **MSBTE : Winter-16, Marks 4**
3. *Describe test case specification of test process.* **MSBTE : Summer-17, Marks 4**
4. *What is test case ? Which parameters are to be considered while documenting test cases ?* **MSBTE : Winter-15, 17, 18, Marks 4**

3.4 Test Reporting

During testing constant communication between test team and development team takes place with the help of a document called test report.

Various types of test reports are -

(1) Test incident report (2) Test cycle report (3) Test summary report

3.4.1 Test Incident Report

- Test incident report is nothing but an entry made in Defect Repository(DR).
- Each defect has unique ID and it is used to identify the incidents.
- A test incident report is a communication that happens through the testing cycle when the defects are encountered.
- The high impact test incidents or defects are highlighted in the test summary report.

3.4.2 Test Cycle Report

Testing takes place in units of test cycles.

A test cycle report at the end of each cycle gives following things -

1) Description of activities carried out during test cycle.
2) Defects that are detected during the test cycle along with their severity and impact.
3) Progress in fixing the defect from one cycle to next cycle.
4) Outstanding defects that are yet to be fixed.
5) Any variation in schedule.

3.4.3 Preparing Test Summary Report

A report that summarizes the results of test cycles is called test summary report.

There are two types of test summary reports -

1) Phase wise test summary report which is produced at the end of every cycle.
2) Final test summary report : It contains following things -
   A) **Summary of the activities** carried out during the test cycle

B) **Variation in actual activities and planned** activities. This includes -

- Tests that are planned to run but could not be run.
- Modified tests.
- Additional tests that are required.
- Difference in effort and time between planned and actual.
- Any other deviation from plan.

C) Summary of results should include

- Tests that are failed with root cause.
- Severity of impact of uncovered defects.

D) Comprehensive assessment and recommendation for release.

Board Questions

1. *Describe test reporting in detail and how to prepare a summary report.* **MSBTE : Summer-15, Marks 8**
2. *State the contents of 'Test Summary Report' used in test reporting.* **MSBTE : Summer-16, Winter-18, Marks 4**
3. *Describe two types of test reports.* **MSBTE : Winter-16, Marks 4**
4. *What are types of test report ? Write contents of test summary report.* **MSBTE : Summer-17, Marks 4**
5. *Describe test reporting in detail. How to prepare a summary report ?* **MSBTE : Summer-18, Marks 8**
6. *Explain how summary report is prepared in test planning.* **MSBTE : Summer-19, Marks 4**

❑❑❑

UNIT - IV

4 Defect Management

4.1 Defect Classification and Management

Defect is something by which the customer requirements do not get satisfied. A defect is basically the difference between the expected result and the actual result.

Defect is a specific concern about the quality of an application under test.

Defects are expensive because finding and correcting defects is supposed to be a complex activity in software development. Defect is supposed to be realization of risks.

Causes of defects

Following are some important causes of defects -

1. The **requirements** defined by the customer are not clear. Sometimes development team makes some assumptions.
2. The **software designs** are **incomplete** and does not accommodate requirements of the customer.
3. **People** working on product design, development and testing are not **skilled.**
4. The **process** that are intended for product design, development and testing are not **capable of producing the desired results.**

Effects of defects

Due to the defects present in the system, customer has total dissatisfaction about the system. Moreover there are some serious effects of defects as given below -

1. Performance of the system will not be at the acceptable level.
2. Security of the system can be problematic and there are chances of external attacks on the system.
3. Required functionality might be absent from the system which may result in rejection of the system by the customer.

4.1.1 Defect Classification

Various types of defects are

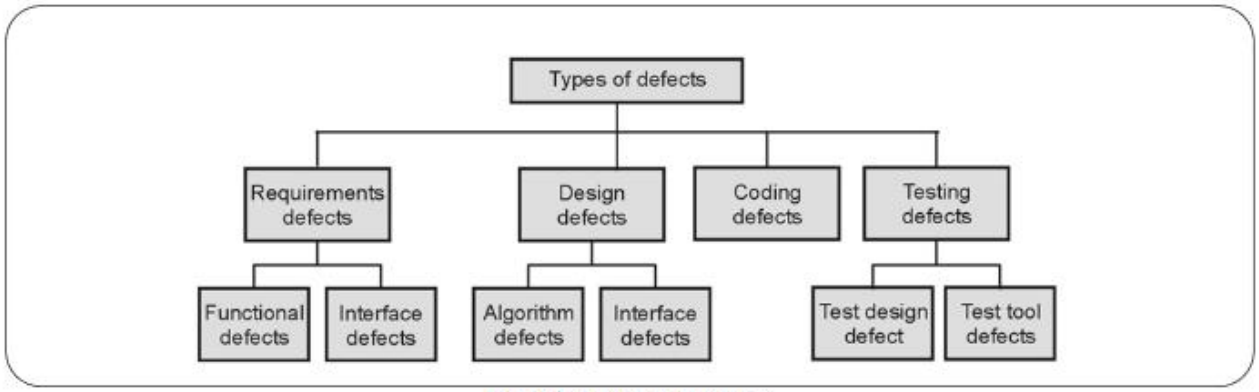

Fig. 4.1.1 Types of defects

Let us discuss them in detail

1. **Requirement defects :** When the developers can not understand the customer requirements properly, the requirements defect occur.

The requirement defects can be further classified as -

i) **Functional defect :** If the customer expected functionality is not present in the system then it is called functional defect.

ii) **Interface defect :** If the defect remain the module when one module is interfaced with the other module then it is called interface defects.

2. **Design defects :** If the software design is not correct or created without understanding the requirements thoroughly, the design defects occur. The design defects occur normally due to the algorithm chosen for the design and the design interfaces.

   Algorithm defects : If the design of algorithm is unable to translate the requirements correctly to the coding then algorithmic defects occur.

   Interface defects : Due to lack of communication the interface defect occurs. When parameter from one module does not get passed to other module correctly then interface defect occurs. Similarly due to look and feel of the application and due to navigation problem the user interface defects get introduced in the system.

3. **Coding defects :** If the coding standards and design standards are not followed properly the coding defects occur. For example

The defects by which the variables do not get initialized properly is a coding defect.

4. **Testing defects :** If the testing is not conducted properly then testing defects occur.

Various types of testing defects are -

1. **Test design defects :** If the test plan, test cases, test scenarios and test data are not properly defined then test design defects occur.
2. **Test tool defect :** If there is defect in the test tool then it is difficult to identify and resolve that defect. One has to conduct manual testing instead of using automated testing tools.

Board Questions

1. *Which are different causes of software defects ?* **MSBTE : Winter-15, Marks 4**
2. *Explain defect classification.* **MSBTE : Winter-15, Marks 4, Winter-18, Marks 6**
3. *What is defect management ? Give defect classification in detail.* **MSBTE : Summer-15, Marks 4**
4. *Give the defect classification and its meaning.* **MSBTE : Summer-16, Winter-17, Marks 4**
5. *Describe the requirement defects and coding defects in details.* **MSBTE : Winter-16, Marks 6**
6. *List all defect classification. Also describe any one defect in brief.* **MSBTE : Summer-17, Marks 4**
7. *Explain requirement defects and design defects.* **MSBTE : Summer-18, Marks 4**

4.1.2 Defect Management

Defect management process can be carried out in various phases as - defect prevention, baseline delivery, defect discovery, defect resolution and process improvement.

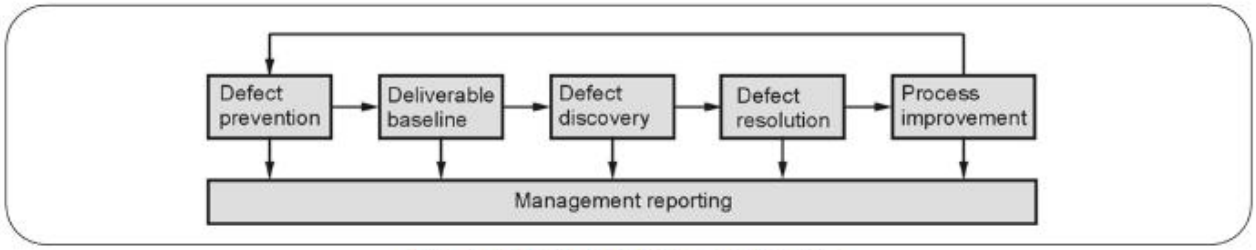

Fig. 4.1.2 Defect management process

1. **Defect prevention :** Defect prevention is the highest priority activity in the defect management process. This stage suggests to prevent the introduction of defects in the software product. Following are the steps taken for defect prevention -
   i) Identify the cause of the defect and reduce the occurrence of defect.
   ii) There are some commonly occurring defects - such as coding defects or interface defects. Focus on these common causes of defects.
   iii) Identify critical risks, assess it and minimize the expected impact of the risks.
2. **Baseline delivery :** The baseline means the work product which is in **deliverable stage.** The deliverable is baselined when it reaches a predefined milestone in its development. If newly created product satisfies the acceptance criteria, then it is baselined. And only baselined product moves to the next stage.
3. **Defect discovery :** Defect discovery means defects are **brought to attention** to the developers. The defect must be **identified** before it gets a serious problem. As soon as defect gets identified, it must be reported to the authority team in order to resolve it.
4. **Defect resolution :** Once the developers have acknowledged a valid defect, the resolution process begins. The **defect resolution** must be done in following steps -
   1. **Determine the importance** of the defect.
   2. **Schedule and fix** the defect as per the order of its importance.
   3. **Notify** it to all the concern parties when defect gets resolved.
5. **Process improvement :** This is the most neglected step in defect management process. This step suggests that participants should **go back to the process** that originated the defect to **understand what caused the defect.** Then they should go back to the validation process that should have caught the defect earlier in the process. Thus treat the defects as an opportunity to improve the software development activities.
6. **Management reporting :** It is important that the defect information must be analyzed and **communicated** to both the **project management and senior management.** The **purpose** of collecting such information is -
   1. To know the status of individual defect.
   2. To provide insight into the processes that need the improvement.
   3. To provide the strategic information to senior management to make important decisions.

Board Questions

1. *Illustrate defect prevention process of defect fixing process with diagram.* **MSBTE : Summer-15, Marks 4**
2. *Draw labelled diagram of defect management process. List any two characteristics of defect management process.* **MSBTE : Winter-16, 18, Marks 4**
3. *Explain defect management process with proper diagram.* **MSBTE : Winter-15, Marks 6**
4. *Describe steps in defect management process.* **MSBTE : Summer-15, Marks 6**
5. *Explain defect management process.* **MSBTE : Summer-16, Winter-17, Marks 4**

6. *Describe defect management process with neat and labelled diagram.* **MSBTE : Summer-18, Marks 6**
7. *Explain defect management process with suitable diagram.* **MSBTE : Summer-19, Marks 8**

4.2 Defect Life Cycle and Template

4.2.1 Defect Life Cycle

Definition : Defect life cycle is a cycle of defect in which defect goes through different stages during its lifetime.

The defect life cycle starts as a soon as new defect is found by a tester and comes to an end when tester closes that defect assuring that it won't get reproduced again.

Defect States

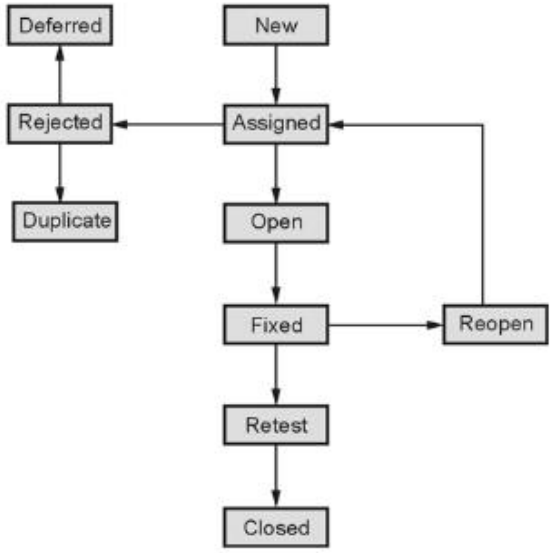

Fig. 4.2.1 Defect life cycle

1) **New :** This is the first state of a defect in the defect life cycle. When any new defect is found, it falls in a 'New' state and validations and testing are performed on this defect in the later stages of the defect life cycle.
2) **Assigned :** In this stage, a newly created defect is assigned to the development team for working on the defect. This is assigned by the project lead or the manager of the testing team to a developer.
3) **Open :** Here, the developer starts the process of analyzing the defect and works on fixing it, if required. If the developer feels that the defect is not appropriate then it may get transferred to any of the below four states namely Duplicate, Deferred, Rejected or Not a Bug-based upon the specific
4) **Fixed :** When the developer finishes the task of fixing a defect by making the required changes then he can mark the status of the defect as 'Fixed'.
5) **Retest :** At this point, the tester starts the task of working on the retesting of the defect to verify if the defect is fixed accurately by the developer as per the requirements or not.
6) **Reopen :** If any issue still persists in the defect then it will be assigned to the developer again for testing and the status of the defect gets changed to 'Reopen'.
7) **Rejected :** If the defect is not considered as a genuine defect by the developer then it is marked as 'Rejected' by the developer.
8) **Duplicate :** If the developer finds the defect as same as any other defect or if the concept of the defect matches with any other defect then the status of the defect is changed to 'Duplicate' by the developer.
9) **Deferred :** If the developer feels that the defect is not of very important priority and it can get fixed in the next releases or so in such a case, he can change the status of the defect as 'Deferred'.
10) **Closed :** When the defect does not exist any longer then the tester changes the status of the defect to 'Closed'.

Board Questions

1. *Explain defect life cycle to identity status of defect with proper labelled diagram.* **MSTE : Winter-15, Winter-17, Marks 4**
2. *Draw the diagram of Defect / Bug life cycle and explain its process.* **MSBTE : Summer-15, Marks 4**
3. *Explain the defect tracking with defect life cycle diagram and the different defect states.* **MSBTE : Summer-16, Marks 8**
4. *Describe defect life cycle with neat diagram.* **MSBTE : Summer-17, 18, 19, Winter-18, Marks 4**

4.2.2 Defect Template

Defect template is a document used by every organization for capturing defect data.

Defect Attribute : Every defect has distinctive attributes in comparison to a test case. Following are the defect attributes used in defect template –

(1)	**Defect ID :**	Unique identifier given to the defect
(2)	**Defect Name :**	Name of the defect. It should be intuitive about the kind of defect.
(3)	**Project Name :**	Name of the project in which the defect is found
(4)	**Module Name :**	Name of the module or submodule in which the defect is located
(5)	**Phase Introduced :**	There are various project life cycle phases such as requirements gathering, design, coding, testing and maintenance. The defect might occur in any of these phases. The defect template gives information about when defect has been introduced. If the tester is not aware of the defect introduction phase, the person doing an analysis may add this defect introduction in defect template. This also helps in identifying defect leakage from one phase to another.
(6)	**Phase Found :**	Phases of the project when the defect found is added here.
(7)	**Defect Type :**	Various defect types are - Functionality defects, Security defects, user interface defects. The information about defect types is written here.
(8)	**Defect Severity :**	Defect severity might be from critical, high, medium, or low. For instance – S1 indicates Defect is critical because some functionality might be missing, S2 indicates that the functionality is not present but the work around is possible. S3 indicates that defect is of minor nature.
(9)	**Priority :**	Priority defines how to take defects for fixing. The priority is defined as high, medium and low. The highest priority defects affect the people or users.
(10)	**Summary :**	The defect is described in shortest possible way.
(11)	**Status :**	Typically status of defect is 'open', 'assigned', 'closed','hold' and so on.
(12)	**Reported by :**	Name of tester who found the defect and date of defect on which it is found.
(13)	**Assigned to :**	The name of the person who is responsible for handling the defect.

Board Questions

1. *Explain the defect template with its attributes.* **MSBTE : Winter-16, Marks 4**
2. *Enlist any six attributes of defect. Describe them with suitable example.* **MSBTE : Summer-17, Marks 6**
3. *Explain defect template.* **MSBTE : Summer-18, Marks 4**
4. *Which parameters are considered while writing good defect report ? Also write contents of defect template.* **MSBTE : Winter-15, 17, Marks 4**
5. *Describe defect template with its attributes.* **MSBTE : Summer-19, Marks 4**

4.3 Impact, Technique and Reporting

- Actual impact of the defect can be realized when the risk becomes a reality. But it is possible to estimate the probable impact with some level of certainty.
- Some organize classify the risk as high, medium and low based on some model.

4.3.1 Estimate Expected Impact of a Defect

Following are some ways to handle the risks –

1) **Accept the risk as it is :** If there occurs some natural disaster then there would not be any solution for this. The actions or solutions to handle

such risk are very costly. In this case, customer or user is given information about probable failure and effect of such failure.

2) **Bypassing the risk :** The risk can be by passed when user cannot accept the risks, or no action can be taken to reduce probability of risk.

How to minimise risk impact or probability ?

Minimization of risk impact can be done in following manner

1) Eliminate risk :

- Risk probability can be reduced by reducing the cause of the risk.
- Although it is not possible to eliminate risk completely, the probability of the risk can be reduced to certain level.
- There are some **preventive controls** that help in reducing the probability of risks. These preventive controls are usually decided by organizational management.
- If probability of risks are very high, then there is no use of any preventive control but if the probability of risk is low then preventive controls become useful.

2) Mitigation of risk :

- Actions taken by organization to minimize possible damage due to realization of risks are called mitigation actions.
- The mitigation actions are planned by organization.
- Corrective controls help in mitigation of risks.

3) Detection ability Improvement :

- If people are prepared for the risk then they can be handled effectively.
- **Detective controls** can be used to visualize the risks.

4) Contingency planning :

- Contingency planning means the **actions** initiated by organization when preventive or corrective actions fail and risk actually occurs.
- These are the ways which are already planned actions by keeping in mind that the preventive and corrective actions might fail.

4.3.2 Techniques for Finding Defects

There are three techniques for finding defects -

1. **Static technique :** This is a technique in which the testing is done without executing the program. The code review is an example of static technique.
2. **Dynamic technique :** This is testing technique in which the system components are executed in order to identify the defects. **Execution of test cases** is an example of dynamic testing technique.
3. **Operational technique :** The system containing defects are delivered to the users and then defects are identified by the users, customers or auditors. Thus defects are actually **detected during the operation of the system.**

4.3.3 Reporting a Defect

- Finding and reporting defects is an important step in software development life cycle.
- It is necessary to find root cause of defect in all the software development life cycle and prepare the document about it.
- Following are some to be noted in reporting defects –
  - **Give complete Record of inconsistency :**
    - Complete description of defect help the tester and user to take preventive, and corrective actions about the defect.
    - Complete record description also helps in process improvement.
  - **Defect report forms a base of quality measurement :**
    - Number of defects serve as a measure of software quality. It must define severity, priority and category of defects.
    - More defects mean quality of software is poor.

- Thus defect report is helpful in deciding the quality of software.

Board Questions

1. *Describe techniques for finding defects.*
   MSBTE : Winter-16, 18, Summer-17, 18, Marks 4
2. *Give any two root causes of defects. Also give any two effects of defects.* **MSBTE : Summer-18, Marks 4**
3. *What are the points considered while estimating impact of a defect ? Also explain techniques to find defect in short.*
   MSBTE : Winter-15, Marks 4, Winter-17, Marks 6
4. *State how to minimize risk impact while estimating defect.* **MSBTE : Summer-15, Marks 4**
5. *What do you mean by 'Defect Impact' ? Explain how to estimate the defect impact.*
   MSBTE : Summer-16, Marks 6
6. *What are the different points to be noted in reporting defects ?* **MSBTE : Winter-16, Marks 4**

❑❑❑

Notes

UNIT - V

5 Testing Tools and Measurements

5.1 Manual Testing and Need for Automated Testing Tools

Introduction

- Now a days, rigorous application testing is a critical part of all software development life cycles. The need of the same is greatly increasing as the organizations are developing mission – critical systems to support their business activities in day to day life.
- Also it becomes necessary to ensure that these systems are reliable, built according to specification and have the ability to support business processes. May internal and external factors are forcing the organizations to ensure a high level of software quality and reliability.

5.1.1 Manual Testing

MSBTE : Winter-15, 16, 17, 18, Summer-19

- To ensure the completeness of software , the testers and the developers often follow a written test plan. This leads them through a set of important test cases.
- These test cases are physically exerted upon the software to test many it for many factors. This is what is the manual testing is!!!
- Manual testing is the process of manually testing software for defects. It requires a tester to play the role of an end user whereby they use most of the application's features to ensure correct behaviour.
- To guarantee completeness of testing, the tester often follows a written test plan that leads them through a set of important test cases.

Definition : Manual testing is the process of using the functions and features of an application as an end-user would in order to verify the software is working as required. With manual testing, a tester manually conducts tests on the software by following a set of pre-defined test cases.

- In manual testing, testers manually execute test cases without using any automation tools.
- It requires a tester to play the role of an end user.
- Any new application must be manually tested before its testing can be automated.
- Manual testing requires more effort, but is necessary to check automation feasibility.
- Manual testing does not require knowledge of any testing tool.
- One of the software testing fundamental is '100% automation is not possible'. This makes manual testing imperative.

5.1.2 Advantages of Manual Testing

We have seen what is manual testing with its definition. Now , let's see what are the advantages of using this technique to test the software.

Some of the advantages are as listed below :

1. It is preferable for products with short life cycles.
2. It is preferable for products that have GUIs that constantly change.
3. It requires less time and expense to begin productive manual testing.
4. Automation cannot replace human intuition, inference, and inductive reasoning.

5. Automation cannot change course in the middle of a test run to examine something that had not been previously considered.
6. Automation tests are more easily fooled than human testers.
7. It is preferable for products with short life cycles.
8. It is preferable for products that have GUIs that constantly change.
9. It requires less time and expense to begin productive manual testing.
10. Automation cannot replace human intuition, inference, and inductive reasoning.
11. Automation cannot change course in the middle of a test run to examine something that had not been previously considered.
12. Automation tests are more easily fooled than human testers.
13. Batch Testing is not possible, for each and every test execution **Human user interaction is mandatory.**

5.1.3 Disadvantages of Manual Testing / Limitation of Manual Testing MSBTE : Winter-15, 16, 17, 18

- In spite of having so many advantages of testing the software manually, this technique is now being obsolete in the market due to certain reasons. People need the work to be finished faster. More work has to completed in less time with greater efficiency.
- The limitations/ disadvantages of using the manual testing techniques are as follows :
  1. Manual Test Case **scope is very limited.**
  2. **Comparing large** amount of data is impractical.
  3. Checking **relevance of search** of operation is **difficult.**
  4. **Processing change requests** during software maintenance **takes more time.**
  5. Manual testing is slow and costly.
  6. It is very labour intensive, it takes a long time to complete tests.
  7. Manual tests don't scale well. As the complexity of the software increases the complexity of the testing problem grows exponentially. This leads to an increase in total time devoted to testing as well as total cost of testing.
  8. Manual testing is not consistent or repeatable. Variations in how the tests are performed as inevitable, for various reasons. One tester may approach and perform a certain test differently from another, resulting in different results on the same test, because the tests are not being performed identically.
  9. Lack of training is the common problem, although not unique to manual software testing.
  10. GUI objects size difference and color combinations are not easy to find in manual testing.
  11. Not suitable for large scale projects and time bound projects.
  12. Batch testing is not possible, for each and every test execution Human user interaction is mandatory.
  13. Comparing large amount of data is impractical.
  14. Processing change requests during software maintenance takes more time.

Board Questions

1. *State limitations of manual testing. Write any four.* **MSBTE : Winter-15 , Marks 4**
2. *Describe any four limitations of manual testing.* **MSBTE : Winter-16 , Marks 4**
3. *Write down any four limitations of manual testing.* **MSBTE : Winter-17 , Marks 4**
4. *State any eight limitations of manual testing.* **MSBTE : Winter-18, Marks 4**
5. *State various advantages and disadvantages of using manual testing tools.* **MSBTE : Summer-19, Marks 4**

5.1.4 Comparison between Manual Testing and Automation Testing

MSBTE : Summer-17, 18

Sr.No.	Automation testing	Manual testing
1.	Perform the same operation each time	Test execution is not accurate all the time. Hence not reliable.
2.	Useful to execute the set of test cases frequently.	Useful when the test case only needs to run once or twice.
3.	Fewer testers required to execute the test cases.	Large number of tester required.
4.	Platform independent.	It is platform dependent.
5.	Testers can fetch complicated information from code.	Does not involve in programming task to fetch hidden information.
6.	Faster	Slow.
7.	Not helpful in UI	Helpful in UI.
8.	High cost	Less than automation.

Board Questions

1. *Differentiate between manual testing and automation testing.* **MSBTE : Summer-17, Marks 4**
2. *Give any four difference between manual and automated testing.* **MSBTE : Summer-18, Marks 4**

5.1.5 Need of Automated Testing Tools

MSBTE : Winter-15, Summer-16, 19

i. An automated testing tool is able to playback pre-recorded and predefined actions, compare the results to the expected behavior and report the success or failure of these manual tests to a test engineer.

ii. Once automated tests are created they can easily be repeated and they can be extended to perform tasks impossible with manual testing.

iii. Because of this, savvy managers have found that automated software testing is an essential component of successful development projects.

Needs of automated testing tools can be listed as follows :

1. **Speed :** Think about how long it would take you to manually try a few thousand test cases for the windows calculator. You might average a test case every five seconds or so. Automation might be able to run 10, 100 even 1000 times that fast.
2. **Efficiency :** While you are busy running test cases, you can't be doing anything else. If you have a test tool that reduces the time it takes for you to run your tests, you have more time for test planning and thinking up new tests.
3. **Accuracy and Precision :** After trying a few hundred cases, your attention may reduce and you will start to make mistakes. A test tool will perform the same test and check the result perfectly, each and every time.
4. **Resource Reduction :** Sometimes it can be physically impossible to perform a certain test case. The number of people or the amount of equipment required to create the test condition could be prohibitive. A test tool can used to simulate the real world and greatly reduce the physical resources necessary to perform the testing.
5. **Simulation and Emulation :** Test tools are used to replace hardware or software that would normally interface to your product. This "face" device or application can then be used to drive or respond to your software in ways that you choose-and ways that might otherwise be difficult to achieve.
6. **Relentlessness :** Test tool and automation never tire or give up. It will continuously test the software.

Board Questions

1. *Which different benefits help to recommend automated testing ? Write advantages of switching to automated testing.* **MSBTE : Winter-15, Marks 4**
2. *What is software test automation ? State types of test automation tools.* **MSBTE : Summer-16, Marks 4**
3. *Which types of test are the first candidates for test automation ? Why ?* **MSBTE : Summer-16, Marks 4**
4. *Explain the need of automation testing.* **MSBTE : Summer-19, Marks 4**

5.1.6 Why Automated Testing ???

- Difficult to test for multi lingual sites manually
- Does not require human intervention
- Increases speed of test execution
- Increase test coverage
- Manual testing can become boring and hence error prone.

5.2 Advantages and Disadvantages of using Tools

5.2.1 Advantages of using Tools

MSBTE : Summer-16, 17, 18, Winter-16, 17

There are many benefits of using the testing tools for supporting the software testing process. They are :

1. **It Saves Time :** Most of the testers find problems with the time required to write long test scripts to perform testing especially when it is regression testing. This takes a lot of time of the testers and the delivery of the bug-free application is delayed.

   A delayed product delivery is not good for any business.

2. **Checks Quality :** Once manual testing procedures are finished, if one applies automated testing processes, it helps to cross-check the test results. Thus, improves the quality of the manual test scripts.

3. **Early Bug Detection :** While developing a software, bugs are easily found when software testing is performed via automated software testing tools. This can save a lot of time and efforts on the SDLC.

4. **Performing Tests 24/7 :** The tests are exerted without stoppage day and night. The machines and tools never get tired. Whatever may be the quantity if testing, can be performed without any kind of human intervention and continuously. The test results are also produced automatically after the tests are performed.

5. **Reusability :** This goes perfectly very easy to understand with automated testing. When the have test scripts are already prepared using test automation tools, they are saved for the future requirements. So, they can be utilized as many times as the testers want especially for automating regression testing.

6. **Distributed Test Execution :** Automated testing comes with distributed testing feature. One can easily execute the test scripts on more than one computer on a shared network or server simultaneously. So, more than one tool is not needed, but only one automated testing tool will be needed.

7. **Easy and Robust Reporting :** Automation testing tools have this amazing benefit of tracking each and every test script. Each and every test script executed can be seen in visual logs. These reports can clearly show the no. of test scripts already executed, scheduled, their reported bugs and how they had been fixed.

8. **Testing Capabilities :** When it comes to capabilities, automated testing tools can test the web applications on the various browsers available in the market via browser testing automation.

   Also, when it comes to mobile application testing, one can test them on various devices. This is next to impossible to achieve with manual testing.

9. **Improves Test Coverage :** Many a time, there are test cases with more than thousand lines of code and to write it and test it would be very difficult with manual testing. This can be easily done with automated testing. Also, these tools will make sure about the in and out of the application like the databases, UI, web services, etc. works according to the requirements. Thus, improving the overall test coverage.

10. **Manpower Utilization :** This is relative to the above benefits. Implementing test automation in testing processes will require less manual efforts. Thus, it will decrease the number of people on just one particular project and can utilize them in the various testing projects.

11. **Improves Team Motivation :** This is a big challenge in the testing industry. When the manpower is distributed among various testing projects, the team can learn about new challenges, new bugs, new skill sets, etc.

12. **Testing Flexibility :** The automated testing tools are developed by the teams who have been into manual testing for years. So, these tools are going to be flexible to match the future testing specifications. These testing tools can be utilized for longer periods of time - say for years.
13. **Improves Accuracy :** Here, no tester is being blamed. A tester with more than 10 years of experience can also make mistakes when they have to prepare the same old boring manual test scripts again and again. When this done with automated testing, not only the results are accurate, but also saves time.
14. **Overcomes Failures of Manual Testing :** There are times when accurate test results are next to impossible with manual testing, like in the stress testing, test automation has proved to be a benefit here.
15. **Improves ROI :** The most important benefit is the return on investment. Obviously, when planning to invest in automated testing tools, first thing needed is to figure out how it will be benefited with those tools in terms of ROI.

    The cost of manual testing includes the time, cost of manual hours and the efforts of the testers, QA managers, etc. And, if the automated testing tools are available, testing will be faster, easily, efficiently, accurately and would be delivering bug-free application within the delivery time period.
16. **Repeatable :** Execution of the same set of automation test cases is possible. That's too without any human intervention and faster.
17. **Reusable :** Re-use of the same set of automation test cases on a different version of an application is possible. Also the same version of an application on a different browser can be tested with the same test cases.
18. **Reliable :** Automation is nothing but the program. If it is written as per expectation, then there are no chance automation program will do anything which is not written. If any program is in working condition, it will work with the same efficiency for the long time.
19. **Quality Software :** Automation can allow us to spend more time to spend on adding the new features and improving the quality of the software. Instead of wasting the time on testing manually, one can think of increasing the grade of the software.
20. **Cost Factor :** Automation in testing can replace the manual repeated tasks. Ultimately it will reduce the costing of the project.
21. **Correct Estimation :** After some iteration of automation testing, person would be aware of the exact time required for the execution. So the correct estimate with more accuracy can be given .

Board Questions

1. *Elaborate the advantages (any four) of using the test automation tools.* **MSBTE : Summer-16 Marks 4**
2. *Explain three types of product metrics.* **MSBTE : Summer-17, Marks 4**
3. *Give any two advantages and any two disadvantages of using testing tools* **MSBTE : Summer-18, Marks 4**

5.2.2 Disadvantages of using Testing Tools

Though the automation testing has many advantages, it has its own disadvantages too. Some of the disadvantages are :

1. Proficiency is required to write the automation test scripts.
2. Debugging the test script is major issue. If any error is present in the test script, sometimes it may lead to deadly consequences.
3. Test maintenance is costly in case of playback methods. Even though a minor change occurs in the GUI, the test script has to be rerecorded or replaced by a new test script.
4. Maintenance of test data files is difficult, if the test script tests more screens.
5. Test automation requires lot of efforts at initial stage.

In software testing two important tasks, one is test design and another is test execution, For test design user (Tester) interaction is mandatory, Testers only create test scripts using Test Tool features and Programming features, It takes more time than manual test case design.

6. 100% Test automation is impractical.

   Generally it is tried to automate maximum test cases, not all Test cases. For some test human user observation is required and is must. Due to some environment limitations we can't automate all testable requirements.

7. All types of Testing not possible (Ex: Usability).

   Functionality tests, performance Tests can be automated, but it is not possible to automate tests that verify user friendliness of the system(AUT).

8. Debugging issues : Programming syntax/logic is used to write tests, some times locating errors in test script is difficult. It is difficult to write such test cases.

9. Tools may have their own defects.

   Test tool is also a software, it may have its own defects in it, so that we may not achieve desired benefits.

10. Programming Knowledge is required.

    Every test tool uses any one of the programming languages (Example UFT supports VBScript, Selenium supports Java, Perl, PHP, C#, PHP and Ruby) to write test scripts. So in order to create and edit test scripts programming knowledge is mandatory.

    In manual testing, no programming knowledge is required.

11. Test tools have environment limitations.

    Test tools have some compatibility issues with operating systems and browsers etc.

    Example :

    QTP supports windows operating environment only, doesn't support other operating environments like UNIX, Macintosh etc…

    Selenium supports web application test automation only, doesn't support desktop / windows based applications.

    For manual testing no environment limitations, we can test computer software or mobile software on any operating system and any browser.)

12. Not suitable for dynamically changing UI designs.

    Most of the test tools support test automation based on front-end objects, if User Interface design changes dynamically then it is difficult to automate.

13. Wrong expectations : Team keeps a lot of expectation from the automation tool. We need to understand it is just a program which will do what it is asked to do.

14. Wrong Costing : In initial days of framework development and converting manual test case in automation test cases takes time. And as this is the first time testing team implementing the automation, this can lead the inaccurate estimation.

15. Results false sense of quality : Automation program will do only things which have been developed. As automation does not have its own brain to sense the defect and test accordingly.

16. Not a replacement of actual testing : It is not the replacement of actual testing, Manual testing can be more interactive and innovative while testing. Innovation from the automation while execution cannot be expected.

17. Maintenance time : If an application changes, it is needed to spend separate time to update the automation scripts.

18. Less number of bugs identified : Automation will execute same test cases again and again, so over the period of time Automation will become less effective.

    Some of the above disadvantages often cause damage to the benefit gained from the automated scripts. Though the automation testing has pros and corns, it is adapted widely all over the world.

Board questions

1. *What are the advantages and disadvantages of using automated tools for testing? List any two each.*
   MSBTE : Winter-16, Marks 4
2. *What is automated testing ? Write down advantages of using automated testing tools in software testing.*
   MSBTE : Winter-17, Marks 4

5.2.3 Features of Test Tool : Guidelines for Static and Dynamic Testing Tools

MSBTE : Summer-15, 16, 19

Test tools : Software testing tools : Tools from a software testing context can be defined as a product that supports one or more test activities right from planning, requirements, creating a build, test execution, defect logging and test analysis.

Classification of tools : Tools can be classified based on several parameters. They include :

- The purpose of the tool
- The activities that are supported within the tool
- The type/level of testing it supports
- The kind of licensing (open source, freeware, commercial)
- The technology used.

This section deals with the major classification of testing tools and how they are used.

1. Static Testing Tool

i. Static analysis tools are generally used by developers as part of the development and component testing process. The key aspect is that the code (or other artefact) is not executed or run but the tool itself is executed, and the source code we are interested in is the input data to the tool.

ii. These tools are mostly used by developers.

iii. Static analysis tools are an extension of compiler technology - in fact some compilers do offer static analysis features. It is worth checking what is available from existing compilers or development environments before looking at purchasing a more sophisticated static analysis tool.

iv. Other than software code, static analysis can also be carried out on things like, static analysis of requirements or static analysis of websites (for example, to assess for proper use of accessibility tags or the following of HTML standards).

v. Static analysis tools for code can help the developers to understand the structure of the code, and can also be used to enforce coding standards.

Features or characteristics of static analysis tools are :

- To calculate metrics such as cyclomatic complexity or nesting levels (which can help to identify where more testing may be needed due to increased risk).
- To enforce coding standards.
- To analyze structures and dependencies.
- Help in code understanding.
- To identify anomalies or defects in the code.

2. Dynamic Testing Tool

MSBTE : Summer-15, Winter-18

i. **Dynamic analysis tools** are 'dynamic' because they require the code to be in a running state. They are 'analysis' rather than 'testing' tools because they analyze what is happening 'behind the scenes' that is in the code while the software is running (whether being executed with test cases or being used in operation).

ii. Let us take an example of a car to understand it in a better way. If you go to a showroom of a car to buy it, you might sit in the car to see if is comfortable and see what sound the doors make - this would be static analysis because the car is not being driven. If you take a test drive, then you would check that how the car performs when it is in the running mode e.g. the car turns right when you turn the steering wheel clockwise or when you press the break then how the car will respond and can also check the oil pressure or the brake fluid, this would be dynamic analysis, it can only be done while the engine is running.

Features or characteristics of dynamic analysis tools are as follows :

i. To detect memory leaks;

ii. To identify pointer arithmetic errors such as null pointers;

iii. To identify time dependencies.

- Eventually when your computer's response time gets slower and slower, but it get improved after re-booting, this may be because of the 'memory leak', where the programs do not correctly release blocks of memory back to the operating system. Sooner or later the system will run out of memory completely and stop. Hence, rebooting restores all of the memory that was lost, so the performance of the system is now restored to its normal state.
- These tools would typically be used by developers in component testing and component integration testing, e.g. when testing middleware, when testing security or when looking for robustness defects.
- Another form of dynamic analysis for websites is to check whether each link does actually link to something else (this type of tool may be called a 'web spider'). The tool does not know if you have linked to the correct page, but at least it can find dead links, which may be helpful.

Testing tools can be classified into following two categories :

1. Static test tools 2. Dynamic test tools

1. Static test tools

Static testing tools are used during static analysis of a system.

They do not involve actual input and output. Static testing tools : are used throughout a software development life cycle, e.g. tools used for verification purposes.

- There are many varieties of static testing tools used by different people as per the type of system being developed.
- These tools do not involve actual input and output. Rather, they take a symbolic approach to testing, i.e. they do not test the actual execution of the software. e.g.
  - **a. Flow analyzers :** Ensure the consistency in data flow from input and output.
  - **b. Coverage analyzers :** It ensures that all logic paths are tested.
  - **c. Interface analyzer :** It examines the effects of passing variables and data between modules.
  - **d. Path tests :** They find unused code and code with contradictions.
- Code complexity measurement tools can be used to measure the complexity of a given code. Similarly, data-profiling tools can be used to optimize a database. Code-profiling tools can be used to optimize code. Test-generators are used for generating a test plan form code. Syntax-checking tools are used to verify correctness of code.

2. Dynamic test tools

Dynamic testing tools are used at different levels of testing starting from unit testing and which may go up to system testing and performance testing.

- These tools are generally used by tester.
- These tools test the software system with live data. e.g.
  - **a. Test driver :** It inputs data into a module – under-test (MUT)
  - **b. Test beds :** It simultaneously displays source code along with the program under execution.
  - **c. Emulators :** The response facilities are used to emulate parts of the system not yet developed.
  - **d. Mutation analysers :** The errors are deliberately fed into the code in order to test fault tolerance of the system.
- There are many different tools used for dynamic testing.
- Some of the areas covered by testing tools are :
  1. Regression testing using automated tools.
  2. Defect tracking and communication systems used by tracking and communication.

  Performance, Load, stress-testing tools.

Board Questions

1. *Explain needs of automation testing.* **MSBTE : Summer-19, Marks 4**
2. *Explain static and dynamic testing tools in details.* **MSBTE : Summer-15, Marks 4**
3. *What is software test automation ? State types of test automation tools.* **MSBTE : Summer-16, Marks 4**
4. *What do you mean by software metrics ? List any three types of metrics.* **MSBTE : Winter-18, Marks 4**
5. *What are static and dynamic testing tools ?* **MSBTE: Summer-19, Marks 4**

5.3 Selecting a Testing Tool

MSBTE : Summer-15, 19, Winter-15, 18

- Automation testing success largely depends on the selection of right testing tools. It takes a lot of time to evaluate relevant automation tools available in the market. But this is a must one-time exercise that will benefit your project in long run.
- There were few situations where I got the chance to review and select automation tool for my projects. The task was difficult as we had to manage our testing needs and cost restrictions but it was a worth experience.

Here are the criteria you need to consider before selecting any testing tool :

Test Automation Tool Evaluation Criteria

1) Do you have the necessary skilled resource to allocate for automation tasks ?
2) What is your budget ?
3) Does the tool satisfy your testing needs ? Is it suitable for the project environment and technology you are using ? Does it support all tools and objects used in the code ? Sometime you may get stuck for small tests due to inabilities of the tool to identify the objects used in the application.

   Above three factors are considered as most important for selecting any tool. The factors include :
4) Does the tool provide you the free trial version so that you can evaluate it before making a decision ? Also, does the tool have all features available in the trial version ?
5) Is the current tool version stable ? Is the vendor company established with good customer support as well as online help resources and user manual ?
6) How is the tool learning curve ? Is the learning time acceptable for your goals ?
7) Do you want automation tool for only your project needs or you are looking for a common tool for all projects in your company ? It would be a good choice if you select a tool that supports most of the coding languages on your projects.
8) Which testing types does it support ? A tool which supports maximum testing types (Unit, functional, regression etc.) is always a better choice. Warning – Don't go for a tool just because it is supporting all testing types. It's also important that the tool should be powerful enough to automate your complex requirements.
9) Does the tool support easy interface to create and maintain test scripts ? Record and playback tool with abilities to edit recorded scripts could be a good solution.
10) Does it provide simple interface yet powerful features to accomplish complex tasks ?
11) How easy is it to provide input test data for complex or load tests ? A tool supporting test data input from various data files such as Excel, XML, text file etc. would be a big relief for the automation the testers.
12) Does it provide the powerful reporting with graphical interface ? Clear and concise reports will always help you to conclude the test results quickly.
13) Does it integrate well with your other testing tools like project planning and test management tools ?

    You may also want to consider other criteria like :
14) Tool vendor refund policy
15) Existing customer reviews for the tool
16) Is the vendor providing initial training ?

i. While introducing the tool in the organization it must match a need within the organization, and solve that need in a way that is both effective and efficient.

ii. The tool should help in building the strengths of the organization and should also address its weaknesses.

- The organization needs to be ready for the changes that will come along with the new tool.

i. If the current testing practices are not good enough and the organization is not mature, then it is always recommended to improve testing practices first rather than to try to find tools to support poor practices. Certainly, we can sometimes improve our own processes in parallel with introducing a tool to support those practices and we can always pick up some good ideas for improvement from the ways that the tools work.

ii. Do not depend on the tool for everything, but it should provide support to your organization as expected.

The following factors are important during tool selection :

i. Assessment of the organization's maturity (e.g. readiness for change);

ii. Identification of the areas within the organization where tool support will help to improve testing processes;

iii. Evaluation of tools against clear requirements and objective criteria;

iv. Proof-of-concept to see whether the product works as desired and meets the requirements and objectives defined for it;

v. Evaluation of the vendor (training, support and other commercial aspects) or open-source network of support;

vi. Identifying and planning internal implementation (including coaching and mentoring for those new to the use of the tool).

Board Questions

1. *Enlist factors considered for selecting a testing tool for test automation.* **MSBTE : Winter-15, Marks 4**
2. *Which are features for selecting static test tools ? Also list any two available test tools (static).* **MSBTE : Winter-15, Marks 4**
3. *Enlist factors considered for selecting a testing tool for test automation.* **MSBTE : Winter-18, Marks 4**
4. *Explain criteria to select testing tool.* **MSBTE : Summer-19, Marks 4**

5.4 When to use Automated Test Tools, Testing using Automated Tool

5.4.1 When Does Test Automation Make Sense ?

Automated tests are a very useful and impressive tool used to make testing more efficient. However, automated tests are not suited to all projects – this may be due to lack of time available, or due to technical limitations.

Automated tests take time to create. Depending on the testers, it can take 3-10 times the amount of time to create an automated test as it takes to run the same test manually. Therefore, automated tests will only start to be valuable when they are run more than 3-10 times.

Automated tests are suitable for the following purposes :

- Regression testing for a stable system that will be run on a regular basis.
- Fast data creation in test systems where the database must be wiped on a regular basis.

i. When there are many repetitive tests.

ii. When there are frequent regression testing iterations.

iii. When you need to simulate large number of users who are using the application resources.

iv. When AUT is having comparatively stable UI.

v. When you have large set of BVT cases.

vi. When you can't rely solely on manual test execution for critical functionality.

- Automated tests are NOT suitable for the following purposes :
  - Testing new functionality – this should be done manually before automated tests are created.
  - Regression testing systems that are expected to have significant user interface changes. Large changes to the user interface require a lot of maintenance for automated tests.

When automating tests, it is wise to only automate as many tests as your team can easily maintain. If some tests are becoming difficult to maintain, it may be worth considering retiring those tests.

Above all, remember that automated tests will never detect as many bugs as a human tester executing the same steps. This is because the human tester's eyes can pick up many things, whereas the test will only notice what it is programmed to notice.

5.4.2 Testing using Automated Tools (Test Automation)

What is Test Automation ?

- Software test automation makes use of specialized tools to control the execution of tests and compares the actual results against the expected result. Usually, regression tests, which are repetitive actions, are automated.
- Testing tools not only helps us to perform regression tests but also helps us to automate data set up generation, product installation, GUI interaction, defect logging, etc. Automation tools are used for both functional and non-functional testing.

Criteria for Tool Selection :

For automating any application, the following parameters should be considered :

- Data driven capabilities
- Debugging and logging capabilities
- Platform independence
- Extensibility and customizability
- E-mail notifications
- Version control friendly
- Support unattended test runs.

Types of Frameworks :

Typically, there are four test automation frameworks that are adopted popular while automating the applications :

- Data Driven Automation Framework
- Keyword Driven Automation Framework
- Modular Automation Framework
- Hybrid Automation Framework.

Tools that are used for functional automation :

Product	Vendor
Quick Test Professional	HP
Rational Robot	IBM
Coded UI	Microsoft
Selenium	Open Source
Auto IT	Open Source

Popular tools that are used for non functional automation :

Product	Vendor
Load Runner	HP
Jmeter	Apache
Burp Suite	PortSwigger
Acunetix	Acunetix

5.5 Metrics and Measurement : Types of Metrics, Product Metrics and Process Metrics, Object Oriented Metrics in Testing

MSBTE : Summer-15, 17, Winter-17

- In recent years software testing technologies have emerged as a dominant software engineering practice which helps in effective cost control, quality improvements, time and risk reduction etc.
- The growth of testing practices has required software testers to find new ways for estimating their projects. A key research area in this field has been 'measurement of and metrics for' the software testing.
- Measurement since plays a critical role in effective and efficient software development, making measurements of the software development and test process is very complex as shown in the Fig. 5.5.1.

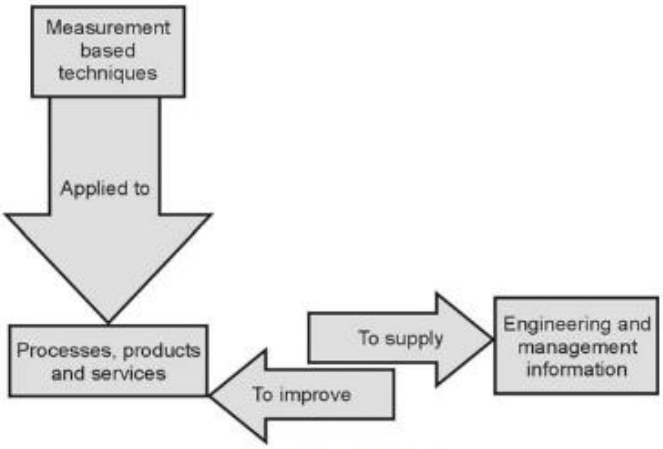

Fig. 5.5.1 : Metrics

5.5.1 Metrics

Software Metrics : Metric is a standard unit of measurement that quantifies results. Metric used for evaluating the software processes, products and services is termed as software metrics.

Definition of Software Metrics : Software metrics is a measurement based technique which is applied to processes, products and services to supply engineering and management information and working on the information supplied to improve processes, products and services, if required.

Importance of Metrics

- Metrics is used to improve the quality and productivity of products and services thus achieving customer satisfaction.
- Easy for management to digest one number and drill down, if required.
- Different Metric(s) trend act as monitor when the process is going out-of-control.
- Metrics provides improvement for current process.
- In software testing there are three main areas which needs to be considered while thinking about metrics and measurement.

i. Defining the metrics

- Small and quality set of metrics should be chosen, large set of metrics should be avoided as it is very confusing to understand large set of metrics.
- Metrics should also be uniform and everybody in team should agree with it.

ii. Tracking test metrics

- After defining the metrics the next step is to track the metrics.
- Since tracking is a constant activity so it's always nice to automate the tracking part.
- Automation reduces time required to track the metrics, analyze them and measure them.

iii. Reporting

- Reporting of the metrics is the most important step, you should report test metrics to stakeholders so that they have clear picture of project progress.
- "A metric is a quantitative measure of the degree to which a system, system component, or process possesses a given attribute. Metrics can be defined as "STANDARDS OF MEASUREMENT". Software metrics are used to measure the quality of the project. Simply, metric is a unit used for describing an attribute. Metric is a scale for measurement. "
- "How many issues are found in thousand lines of code ?", here **No. of issues is one measurement and No. of lines of code is another measurement. Metric is defined from these two measurements.**

Metrics Lifecycle

The process involved in setting up the metrics :

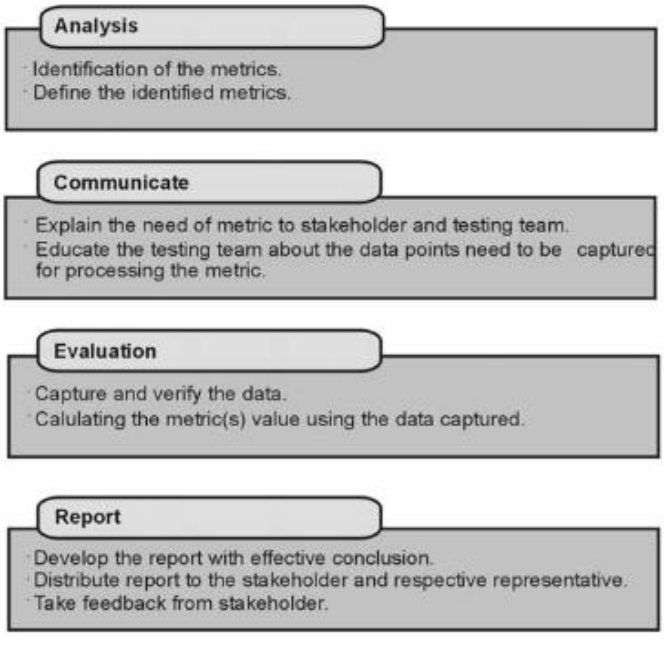

Fig. 5.5.2 : Metrics lifecycle

5.5.2 What is Software Test Measurement ?

i. Measurement is the quantitative indication of extent, amount, dimension, capacity, or size of some attribute of a product or process.

ii. Test measurement example : Total number of defects.

iii. Please refer below Fig 5.5.3 for clear understanding of the difference between measurement and metrics.

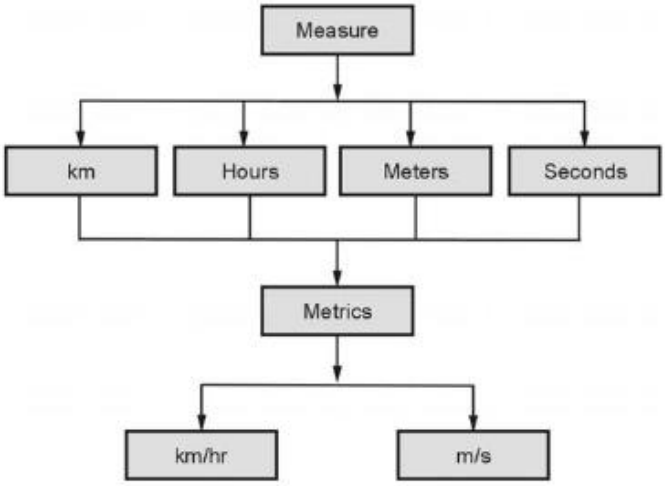

Fig. 5.5.3 Example of measure and metric

Why Test Metrics ?

Generation of software test metrics is the most important responsibility of the software test lead/manager.

Test Metrics are used to,

i. Take the decision for next phase of activities such as, estimate the cost and schedule of future projects.

ii. Understand the kind of improvement required to success the project.

iii. Take decision on process or technology to be modified etc.

Importance of Software Testing Metrics :

i. As explained above, test metrics are the most important to measure the quality of the software.

ii. Now, how can we measure the quality of the software by using Metrics ?

iii. Suppose, if a project does not have any metrics, then how the quality of the work done by a Test analyst will be measured ?

For Example : A Test Analyst has to,

i. Design the test cases for five requirements

ii. Execute the designed test cases

iii. Log the defects and need to fail the related test cases

iv. After the defect is resolved, need to re-test the defect and re-execute the corresponding failed test case.

- In above scenario, if metrics are not followed, then the work completed by the test analyst will be subjective i.e. the test report will not have the proper information to know the status of his work/project.
- If metrics are involved in the project, then the exact status of his/her work with proper numbers/data can be published.

For example, in the Test report, we can publish :

1. How many test cases have been designed per requirement ?
2. How many test cases are yet to design ?
3. How many test cases are executed ?
4. How many test cases are passed/failed/blocked ?
5. How many test cases are not yet executed ?
6. How many defects are identified and what is the severity of those defects ?
7. How many test cases are failed due to one particular defect ? etc.

- Based on the project needs we can have more metrics than above mentioned list, to know the status of the project in detail.

Based on the above metrics, test lead/manager will get the understanding of the below mentioned key points :

a) Percentage of work completed

b) Percentage of work yet to be completed

c) Time to complete the remaining work

d) Whether the project is going as per the schedule or lagging ? etc.

- Based on the metrics, if the project is not going to complete as per the schedule, then the manager will raise the alarm to the client and other stake holders by providing the reasons for lagging to avoid the last minute surprises.

5.5.3 Types of Metrics

MSBTE : Summer-16, 18, Winter-18

5.5.3.1 Types of Manual Test Metrics

Testing Metrics are mainly divided into two categories.

1. **Base Metrics :** Base Metrics are the Metrics which are derived from the data gathered by the test analyst during the test case development and execution.

   This data will be tracked throughout the test life cycle. i.e. collecting the data like, total no. of test cases developed for a project (or) no. of test cases need to be executed (or) no. of test cases passed/failed/blocked etc.

2. **Calculated Metrics :** Calculated Metrics are derived from the data gathered in base metrics. These metrics are generally tracked by the test lead/manager for test reporting purpose.

5.5.3.2 Examples of Software Testing Metrics

Let's take an example to calculate various test metrics used in software test reports :

Below is the table format for the data retrieved from the test analyst who is actually involved in testing :

Sr.No.	Testing metric	Data retrieved during test case development and execution
1.	No. of requirements	5
2.	Avg. no. of testcases written per Requirement	20
3.	Total no. of testcases written for all requirements	100
4.	Total no. of testcases executed	65
5.	No. of test cases passed	30
6.	No. of test cases failed	26
7.	No. of test cases blocked	9
8.	No. of test cases unexecuted	35
9.	Total No. of defects identified	30
10.	Critical defects count	6
11.	High defects count	10
12.	Medium defects count	6
13.	Low defects count	8

Importance of metrics and measurement in SDLC

i. During all the software development life cycle it is very important to apply metrics and measurement because metrics and measurement set expectations. If there are well established metrics and measurements in project then the test analyst can easily track and report quality results to the management.

ii. If the metrics and measurements are not established properly then the assessment of software quality is purely subjective which arises disputes at the end of development life cycle. You can consider some the following areas where you can apply metrics and measurement. This list is not exhaustive, you can have metrics for lot more things :

   1) Schedule of project
   2) Coverage
   3) Planned and actual cost
   4) Workload and resource usage
   5) Product risk and project risk
   6) Defects

iii. While doing test planning we set the expectations for the stakeholders or we set the baselines for them. If you have established the baselines the test reporting is consistent to the management and you can avoid subjective assessment of testing.

Definitions and Formulas for Calculating Metrics :

1) **Percentage test cases executed :** This metric is used to obtain the execution status of the test cases in terms of %ge.

   %ge Test cases Executed = (No. of Test cases executed / Total no. of test cases written) * 100.

   So, from the above data,

   %ge Test cases Executed = (65 / 100) * 100 = 65 %

2) **Percentage test cases not executed :** This metric is used to obtain the pending execution status of the test cases in terms of %ge.

   %ge test cases not executed = (No. of test cases not executed / Total no. of test cases written) * 100.

   So, from the above data,

   %ge Test cases blocked = (35 / 100) * 100 = 35 %

Board Question

1. *Define metrics and measurements. Explain need of software measurement.*

MSBTE : Summer-15, Winter-17, Marks 4

5.5.4 Product Metrics and Process Metrics, Object Oriented Metrics in Testing

5.5.4.1 Product metrics and Process metrics

MSBTE : Winter-16

Software metrics can be classified into three categories

- **Product metrics** – Describes the characteristics of the product such as size, complexity, design features, performance, and quality level.
- **Process metrics** – These characteristics can be used to improve the development and maintenance activities of the software.
- **Project metrics** – This metrics describe the project characteristics and execution. Examples include the number of software developers, the staffing pattern over the life cycle of the software, cost, schedule, and productivity.

Some metrics belong to multiple categories. For example, the in-process quality metrics of a project are both process metrics and project metrics.

Software quality metrics are a subset of software metrics that focus on the quality aspects of the product, process, and project. These are more closely associated with process and product metrics than with project metrics.

Software quality metrics can be further divided into three categories –

- Product quality metrics
- In-process quality metrics
- Maintenance quality metrics

1. Product Quality Metrics

This metrics include the following –

- Mean Time to Failure
- Defect Density
- Customer Problems
- Customer Satisfaction.

Mean Time to Failure : It is the time between failures. This metric is mostly used with safety critical systems such as the airline traffic control systems, avionics, and weapons.

Defect Density : It measures the defects relative to the software size expressed as lines of code or function point, etc. i.e., it measures code quality per unit. This metric is used in many commercial software systems.

Customer Problems : It measures the problems that customers encounter when using the product. It contains the customer's perspective towards the problem space of the software, which includes the non-defect oriented problems together with the defect problems.

The problems metric is usually expressed in terms of Problems per User-Month (PUM).

PUM = Total Problems that customers reported (true defect and non-defect oriented problems) for a time period + Total number of license months of the software during the period

Where,

Number of license-month of the software = Number of install license of the software × Number of months in the calculation period

PUM is usually calculated for each month after the software is released to the market, and also for monthly averages by year.

Customer Satisfaction : Customer satisfaction is often measured by customer survey data through the five-point scale –

- Very satisfied
- Satisfied
- Neutral
- Dissatisfied
- Very dissatisfied

Satisfaction with the overall quality of the product and its specific dimensions is usually obtained through various methods of customer surveys. Based on the five-point-scale data, several metrics with slight variations can be constructed and used, depending on the purpose of analysis. For example –

- Percent of completely satisfied customers
- Percent of satisfied customers
- Percent of dis-satisfied customers
- Percent of non-satisfied customers

Usually, this percent satisfaction is used.

In-process quality metrics : In-process quality metrics deals with the tracking of defect arrival during formal machine testing for some organizations. This metric includes –

- Defect density during machine testing
- Defect arrival pattern during machine testing
- Phase-based defect removal pattern
- Defect removal effectiveness.

Defect density during machine testing : Defect rate during formal machine testing (testing after code is integrated into the system library) is correlated with the defect rate in the field. Higher defect rates found during testing is an indicator that the software has experienced higher error injection during its development process, unless the higher testing defect rate is due to an extraordinary testing effort.

This simple metric of defects per KLOC or function point is a good indicator of quality, while the software is still being tested. It is especially useful to monitor subsequent releases of a product in the same development organization.

Defect arrival pattern during machine testing

The overall defect density during testing will provide only the summary of the defects. The pattern of defect arrivals gives more information about different quality levels in the field. It includes the following –

- The defect arrivals or defects reported during the testing phase by time interval (e.g., week). Here all of which will not be valid defects.
- The pattern of valid defect arrivals when problem determination is done on the reported problems. This is the true defect pattern.
- The pattern of defect backlog overtime. This metric is needed because development organizations cannot investigate and fix all the reported problems immediately. This is a workload statement as well as a quality statement. If the defect backlog is large at the end of the development cycle and a lot of fixes have yet to be integrated into the system, the stability of the system (hence its quality) will be affected. Retesting (regression test) is needed to ensure that targeted product quality levels are reached.

Phase-based defect removal pattern : This is an extension of the defect density metric during testing. In addition to testing, it tracks the defects at all phases of the development cycle, including the design reviews, code inspections, and formal verifications before testing.

Because a large percentage of programming defects is related to design problems, conducting formal reviews, or functional verifications to enhance the defect removal capability of the process at the front-end reduces error in the software. The pattern of phase-based defect removal reflects the overall defect removal ability of the development process.

With regard to the metrics for the design and coding phases, in addition to defect rates, many development organizations use metrics such as inspection coverage and inspection effort for in-process quality management.

Defect removal effectiveness

It can be defined as follows –

"DRE=(Defect_removed_during_a_development_phase/Defects_latent_in_the_product) × 100 %

This metric can be calculated for the entire development process, for the front-end before code integration and for each phase. It is called early defect removal when used for the front-end and phase effectiveness for specific phases. The higher the value of the metric, the more effective the development process and the fewer the defects passed to the next phase or to the field. This metric is a key concept of the defect removal model for software development.

Maintenance Quality Metrics : Although much cannot be done to alter the quality of the product during this phase, following are the fixes that can be carried out to eliminate the defects as soon as possible with excellent fix quality.

- Fix backlog and backlog management index
- Fix response time and fix responsiveness
- Percent delinquent fixes
- Fix quality.

Fix backlog and backlog management index : Fix backlog is related to the rate of defect arrivals and the rate at which fixes for reported problems become available. It is a simple count of reported problems that remain at the end of each month or each week. Using it in the format of a trend chart, this metric can provide meaningful information for managing the maintenance process.

Backlog Management Index (BMI) is used to manage the backlog of open and unresolved problems.

BMI=(Number_of_problems_closed_during_the_month/Number_of_problems_arrived_during_the_month) × 100 %

If BMI is larger than 100, it means the backlog is reduced. If BMI is less than 100, then the backlog increased.

Fix response time and fix responsiveness : The fix response time metric is usually calculated as the mean time of all problems from open to close. Short fix response time leads to customer satisfaction.

The important elements of fix responsiveness are customer expectations, the agreed-to fix time, and the ability to meet one's commitment to the customer.

Percent delinquent fixes

It is calculated as follows –

Percent Deliquent fixes = (Number of fixes that exceeded the response time criteria by severity level / Number of fixes delivered in a specified time) × 100 %

Fix Quality : Fix quality or the number of defective fixes is another important quality metric for the maintenance phase. A fix is defective if it did not fix the reported problem, or if it fixed the original problem but injected a new defect. For mission-critical software, defective fixes are detrimental to customer satisfaction. The metric of percent defective fixes is the percentage of all fixes in a time interval that is defective.

A defective fix can be recorded in two ways : Record it in the month it was discovered or record it in the month the fix was delivered. The first is a customer measure; the second is a process measure. The difference between the two dates is the latent period of the defective fix.

Usually the longer the latency, the more will be the customers that get affected. If the number of defects is large, then the small value of the percentage metric will show an optimistic picture. The quality goal for the maintenance process, of course, is zero defective fixes without delinquency.

Board question

1. *Explain three types of product metrics.*

MSBTE : Winter-16, Marks 4

2. Progress Metrics

- The software test metrics can be used for different purposes. One of them was tracking progress. Usually when we track progress, it is related to time, or other unit that indicates a schedule. Most often we use progress metrics to track planned versus actual over time.
- What we track depends on our role. If we are financial people, then we track money spent. But for software quality assurance, we want to track progress of such things as defects, test cases, man-hours, etc.
- Basically anything that is related to results or effort spent to get those results. For instance, here are a few :

Man-hours/test case executed : The natural tendency in driving costs down is to force this as low as possible, but remember the faster they go, and the more tests that are executed, does not translate into higher quality software.

Planned hours/actual hours : We want to track the effort we plan versus what we spend not only just to see if we are planning our resources well, but to see deviations, and then think and find out why those deviations exist which could point to problems.

If we find out that planned versus actual deviates on certain days of the week, or that deviations occur only from certain testers, and those testers are working on specific parts of the software, this is useful information.

Test cases executed/planned : This just keeps us on track to make sure we get the bare minimum done in terms of getting our test cases executed.

If we find that it takes too long, on a repeated basis, then something needs to be changed. Or if we go faster than normal on a regular basis, then this may point to a problem with the test cases (especially if they find no defects).

Test cases executed/defects found : This is a metric that indicates how good our test cases are at finding defects. Test cases run with no defects found or with a low ratio does not mean there are no defects in the software.

Here is a generic chart in Fig. 5.5.4 which shows planned versus actual in terms of test cases executed.

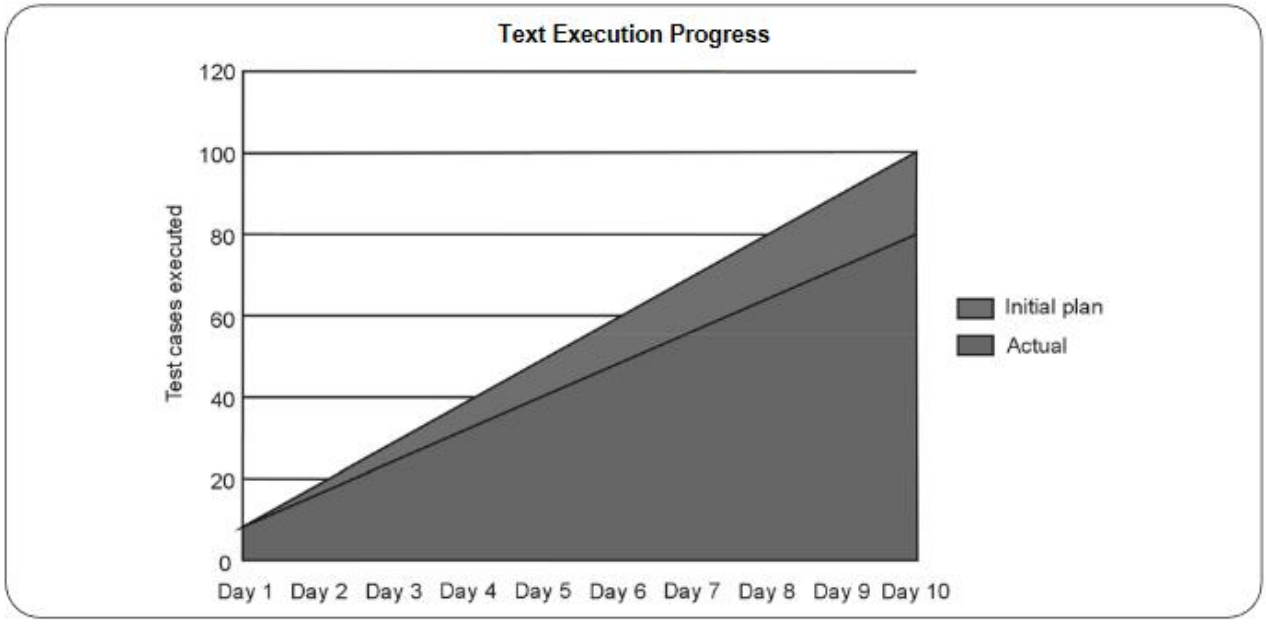

Fig. 5.5.4 Executed versus planned test cases

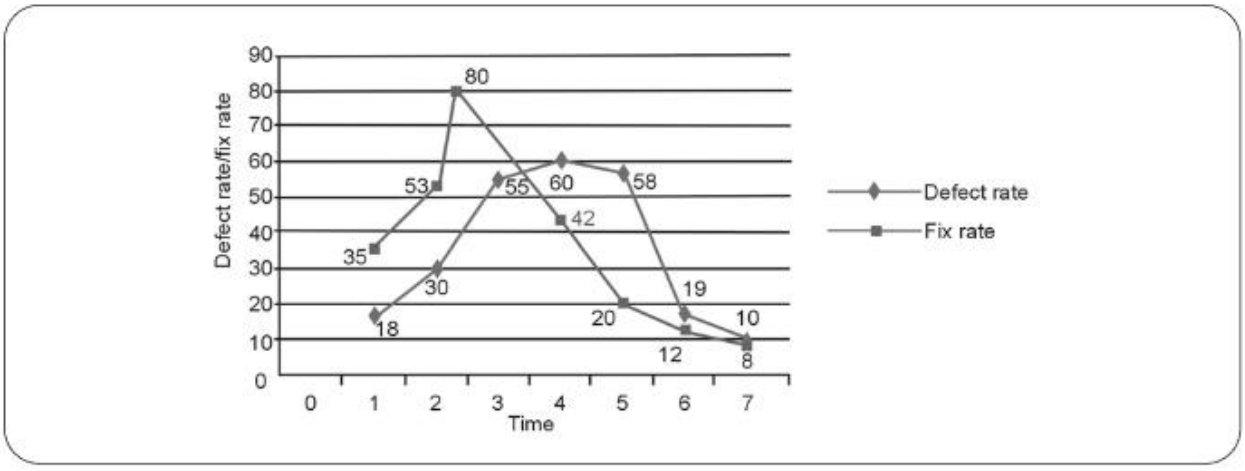

Fig. 5.5.5 Defect fix rate

The chart for the defect fix rate is as shown in Fig. 5.5.5.

Productivity Matrix : Productivity metrics are ways to measure how much is produced for an input such as an hour of work. They are commonly used to manage and improve performance.

Productivity was defined as how many 'simple tasks' can be delivered by day. It can be computed at various levels within a project: individual, profile, task phase or project.

The definition 'simple task' will always raise heated conversations; we finally define it as the amount of time required by a resource to deliver something and settle it to five hours (of course some simple tasks take less or more, but we settled for that number).

Given our definition 'simple tasks that can be delivered by day - 8 hours-' the formula was defined as :

Productivity = ((Planned Effort / 5) / Actual Effort) *8

The following are common examples of productivity metrics.

Lines of Code Per Day : The amount of source code produced per software developer per day. This measure isn't particularly accurate as much code is autogenerated or cut and pasted.

Function Points Per Week : Measuring the productivity of a development team by how many function points they implement in a month or week.

Story Points : Development productivity can also be measured by story points per week.

Units Per Hour : The number of units completed in an hour by a production line.

Average Handle Time : The average time to handle a customer service request such as a refund. Used alone this can be a dangerous metric that encourages poor service. Typically, this is a secondary measurement after quality metrics such as customer satisfaction.

Revenue Per Employee : A broad metric to measure the productivity of an organization. Calculated by dividing revenue by total employees.

Revenue Per Salesperson : A common sales productivity metric is simply to look at how much revenue was generated per salesperson.

Labor Productivity : Productivity can be measured for a nation, region or industry by calculating GDP or revenue per hour worked. Generally speaking, productivity increases over time due to technological and process improvements.

3. Project metrics :

- The first application of project metrics occurs during estimation–Metrics from past projects are used as a basis for estimating time and effort.

- As a project proceeds, the amount of time and effort expended are compared to original estimates.
- As technical work commences, other project metrics become important–production rates are measured (represented in terms of models created, review hours, function points, and delivered source lines of code)–Error uncovered during each generic framework activity (i.e, communication, planning, modeling, construction, deployment) are measured.

Use of Project Metrics

- Project metrics are used to–Minimize the development schedule by making the adjustments necessary to avoid delays and mitigate potential problems and risks–Assess product quality on an ongoing basis and, when necessary, to modify the technical approach to improve quality.
- In summary–As quality improves, defects are minimized–As defects go down, the amount of rework required during the project is also reduced–As rework goes down, the overall project cost is reduced
- Project metrics can be consolidated to create process metrics for an organization

5.5.4.2 Object Oriented Metrics

It is widely accepted that object oriented development requires a different way of thinking than traditional structured development1 and software projects are shifting to object oriented design.

The main advantage of object oriented design is its modularity and reusability. Object oriented metrics are used to measure properties of object oriented designs.

Compared to structural development, object oriented design is a comparatively new technology. The metrics, which were useful for evaluating structural development, may perhaps not affect the design using OO language.

As for example, the "Lines of Code" metric is used in structural development whereas it is not so much used in object oriented design. Very few existing metrics (so called traditional metrics) can measure object oriented design properly.

Lines of code and functional point metrics can be used for estimating object-oriented software projects. However, these metrics are not appropriate in the case of incremental software development as they do not provide adequate details for effort and schedule estimation. Thus, for object-oriented projects, different sets of metrics have been proposed. These are listed below.

- **Number of scenario scripts :** Scenario scripts are a sequence of steps, which depict the interaction between the user and the application. A number of scenarios is directly related to application size and number of test cases that are developed to test the software, once it is developed. Note that scenario scripts are analogous to use-cases.
- **Number of key classes :** Key classes are independent components, which are defined in object-oriented analysis. As key classes form the core of the problem domain, they indicate the effort required to develop software and the amount of 'reuse' feature to be applied during the development process.
- **Number of support classes :** Classes, which are required to implement the system but are indirectly related to the problem domain, are known as support classes. For example, user interface classes and computation class are support classes. It is possible to develop a support class for each key class. Like key classes, support classes indicate the effort required to develop software and the amount of 'reuse' feature to be applied during the development process.
- **Average number of support classes per key class :** Key classes are defined early in the software project while support classes are defined throughout the project. The estimation process is simplified if the average number of support classes per key class is already known.

- **Number of subsystems :** A collection of classes that supports a function visible to the user is known as a subsystem. Identifying subsystems makes it easier to prepare a reasonable schedule in which work on subsystems is divided among project members.

The afore-mentioned metrics are collected along with other project metrics like effort used, errors and defects detected, and so on. After an organization completes a number of projects, a database is developed, which shows the relationship between object-oriented measure and project measure. This relationship provides metrics that help in project estimation.

Board questions

1. *State the different metrics types with its classification.*
   MSBTE : Summer-16, Marks 4
2. *Define software metrics. Describe product Vs process and objective Vs subjective metrics.*
   MSBTE : Summer-17, Marks 4
3. *What do you mean by software metrics ? List any three types of metrics.*
   MSBTE : Summer-18, Marks 4
4. *Define the term metrics and measurements. Explain need of software measurement.*
   MSBTE : Winter-18, Marks 4

Notes

SOLVED SAMPLE TEST PAPER - I

Software Testing

T.Y. Diploma (Sem - V) Computer Engg. Group (CO/CM/CW)

Time : 1 Hour] **[Total Marks : 20**

Instructions :

1) All questions are compulsory.

2) Illustrate your answers with neat sketches wherever necessary.

3) Figures to the right indicate full marks.

4) Assume suitable data if necessary.

5) Preferably, write the answers in sequential order.

Q.1 Attempt any FOUR. **[8]**

a) *Explain the terms mistake, defect, fault and failure in relation with software testing.* **(Refer section 1.2)**

b) *Enlist features of test cases* **(Refer section 1.3.2)**

c) *What is GUI testing ?* **(Refer section 2.6.2)**

d) *What is bi-directional integration testing ?* **(Refer section 2.3.3)**

e) *What kind of errors can be identified during unit testing ?* **(Refer section 2.2.1)**

f) *What is size estimation ?* **(Refer section 3.1.7(1))**

Q.2 Attempt any THREE. **[12]**

a) *Explain driver and stub in detail.* **(Refer section 2.2.3)**

b) *What factors shall be considered while selecting resource requirements ?* **(Refer section 3.1.5)**

c) *Difference between top down integration and bottom up integration.* **(Refer section 2.2.2)**

d) *Explain requirement based testing.* **(Refer section 1.7.1)**

e) *How to prepare a test plan ? Explain in detail.* **(Refer section 3.1.1)**

f) *Differentiate between verification and validation (Any four points).* **(Refer section 1.4)**

❑❑❑

SOLVED SAMPLE TEST PAPER - II

Software Testing

T.Y. Diploma (Sem - V) Computer Engg. Group (CO/CM/CW)

Time : 1 Hour] **[Total Marks : 20**

Instructions :

1) All questions are compulsory.

2) Illustrate your answers with neat sketches wherever necessary.

3) Figures to the right indicate full marks.

4) Assume suitable data if necessary.

5) Preferably, write the answers in sequential order.

Q.1 Attempt any FOUR. **[8]**

a) *Enlist the states in defect life cycle.* **(Refer section 4.2.1)**

b) *Give any two advantages of manual testing.* **(Refer section 5.1.2)**

c) *Enlist different types of defects.* **(Refer section 4.1.1)**

d) *Explain the need for testing tool (Any two points).* **(Refer section 5.1.5)**

e) *Enlist the techniques for finding defects.* **(Refer section 4.3.2)**

f) *Specify the any two criteria that should be considered before selecting any testing tool.* **(Refer section 5.3)**

Q.2 Attempt any THREE. **[12]**

a) *Differentiate between manual testing and automation testing (Any four points).* **(Refer section 5.1.4)**

b) *What do you mean by 'Defect Impact' ? Explain how to estimate the defect impact.* **(Refer section 4.3)**

c) *Explain the importance of software metrics.* **(Refer section 5.5.1)**

d) *Explain defect management process in detail.* **(Refer section 4.1.2)**

e) *What are types of manual testing metrics.* **(Refer section 5.5.3)**

f) *Explain the defect template with its attributes.* **(Refer section 4.2.2)**

❑❑❑

SOLVED SAMPLE QUESTION PAPER

Software Testing

T.Y. Diploma (Sem - V) Computer Engg. Group (CO/CM/CW)

Time : 3 Hours] **[Total Marks : 70**

Instructions :

1) All questions are compulsory.

2) Illustrate your answers with neat sketches wherever necessary.

3) Figures to the right indicate full marks.

4) Assume suitable data if necessary.

5) Preferably, write the answers in sequential order.

Q.1 Attempt any FIVE of the following. **[10]**

a) *What is software testing ?* **(Refer section 1.1.1)**

b) *Enlist various levels of testing.* **(Refer section 2.1)**

c) *What is the purpose of Test plan ?* **(Refer section 3.1)**

d) *Give any two possible causes of defects.* **(Refer section 4.1)**

e) *What is manual testing ?* **(Refer section 5.1)**

f) *What is performance testing ?* **(Refer section 2.4.1)**

g) *Define – Static and dynamic testing.* **(Refer section 1.5)**

Q.2 Attempt any THREE of the following. **[12]**

a) *Give difference between quality assurance and quality control. (Any four)* **(Refer section 1.4.5)**

b) *Describe top-down integration testing with labelled diagram.* **(Refer section 2.3.1)**

c) *Why is it essential to setup criteria for testing ? List any three criteria in different situations.* **(Refer section 3.1.3)**

d) *Explain requirement defects and design defects.* **(Refer section 4.1.1)**

Q.3 Attempt any THREE of the following. **[12]**

a) *State any eight limitations of manual testing.* **(Refer section 5.1.3)**

b) *Explain the 'Test Infrastructure' components with diagram.* **(Refer section 3.2.1)**

c) *Explain static and dynamic testing tools in details.* **(Refer section 5.2.3)**

d) *With the help of neat diagram, describe unit testing.* **(Refer section 2.2)**

Q.4 Attempt any THREE of the following. **[12]**

a) *Describe V-model with labelled diagram.* **(Refer section 1.4.2)**

b) *Explain defect life cycle to identity status of defect with proper labelled diagram.* **(Refer section 4.2.1)**

c) *Explain people management in test planning.* **(Refer section 3.2.2)**

d) *What is load testing and stress testing ? Describe with respect to system testing.* **(Refer section 2.4)**

e) *Enlist factors considered for selecting a testing tool for test automation.* **(Refer section 5.3)**

Q.5 Attempt any TWO of the following. **[12]**

a) *What is walkthrough ? Give advantages and disadvantages of it.* **(Refer section 1.6.1.2)**

b) *Write important six test cases for the 'Login Form' of the Facebook website.* **(Refer example 3.3.2)**

c) *Describe defect template with its attributes.* **(Refer section 4.2.2)**

Q.6 Attempt any TWO of the following. **[12]**

a) *Draw the flow graph for finding maximum of three numbers and derive the test cases using cyclomatic complexity.* **(Refer example 1.6.1)**

b) *Differentiate between alpha testing and beta testing (Any six point).* **(Refer section 2.5.2)**

c) *Define metrics and measurements. Explain need of software measurement.* **(Refer section 5.5)**

❑❑❑