MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION
(Autonomous)
(ISO/IEC - 27001 - 2005 Certified)
WINTER-15EXAMINATION
Model Answer Paper

Subject Code: 17624 **Subject Name: Software Testing**

Important Instructions to examiners:
1) The answers should be examined by key words and not as word-to-word as given in themodel answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may tryto assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling errors should not be given moreImportance (Not applicable for subject English and Communication Skills).
4) While assessing figures, examiner may give credit for principal components indicated in thefigure. The figures drawn by candidate and model answer may vary. The examiner may give credit for anyequivalent figure drawn.
5) Credits may be given step wise for numerical problems. In some cases, the assumed constantvalues may vary and there may be some difference in the candidate's answers and model answer.
6) In case of some questions credit may be given by judgment on part of examiner of relevant answer based on candidate's understanding.
7) For programming language papers, credit may be given to any other program based on equivalentconcept.

1. **a) Attempt any THREE of the following:** **Marks 12**

(i) **What is static testing? State advantages and disadvantages of static testing two each.**

(Explanation - 2 Marks, two advantages - 1 Mark, two disadvantages - 1 Marks)

Ans:

Static testing is a type testing which requires only source code of the product, not the binaries or executable. Static testing does not involve executing the program on computers but involves select people going through the code to find out whether

1. The code works according functional requirement.
2. The code has been written in accordance with the design developed earlier in the projectlife cycle.
3. The code for any functionality has been missed out.
4. The code handles errors properly.

Advantages:

1. Static testing start early in the life cycle so early feedback on quality issues can be established.

2. As the defects are getting detected at early stage so the rework cost most often relatively low.
3. Development productivity is likely to increase because of the less rework effort.
4. It helps to produce a better product.

Disadvantages:

1. It is time-consuming.
2. The logistics and scheduling can become an issue since multiple peoples are involved.
3. It is not always possible to go through every line of code.
4. Required High- skills.

(ii) **State and explain top-down approach of integration testing with diagram.**

(Explanation -2 Marks, diagram-2 Marks)

Ans: The strategy in top-down integration is look at the design hierarchy from top to bottom. Start with the high - level modules and move downward through the design hierarchy.

Modules subordinate to the top modules are integrated in the following two ways:

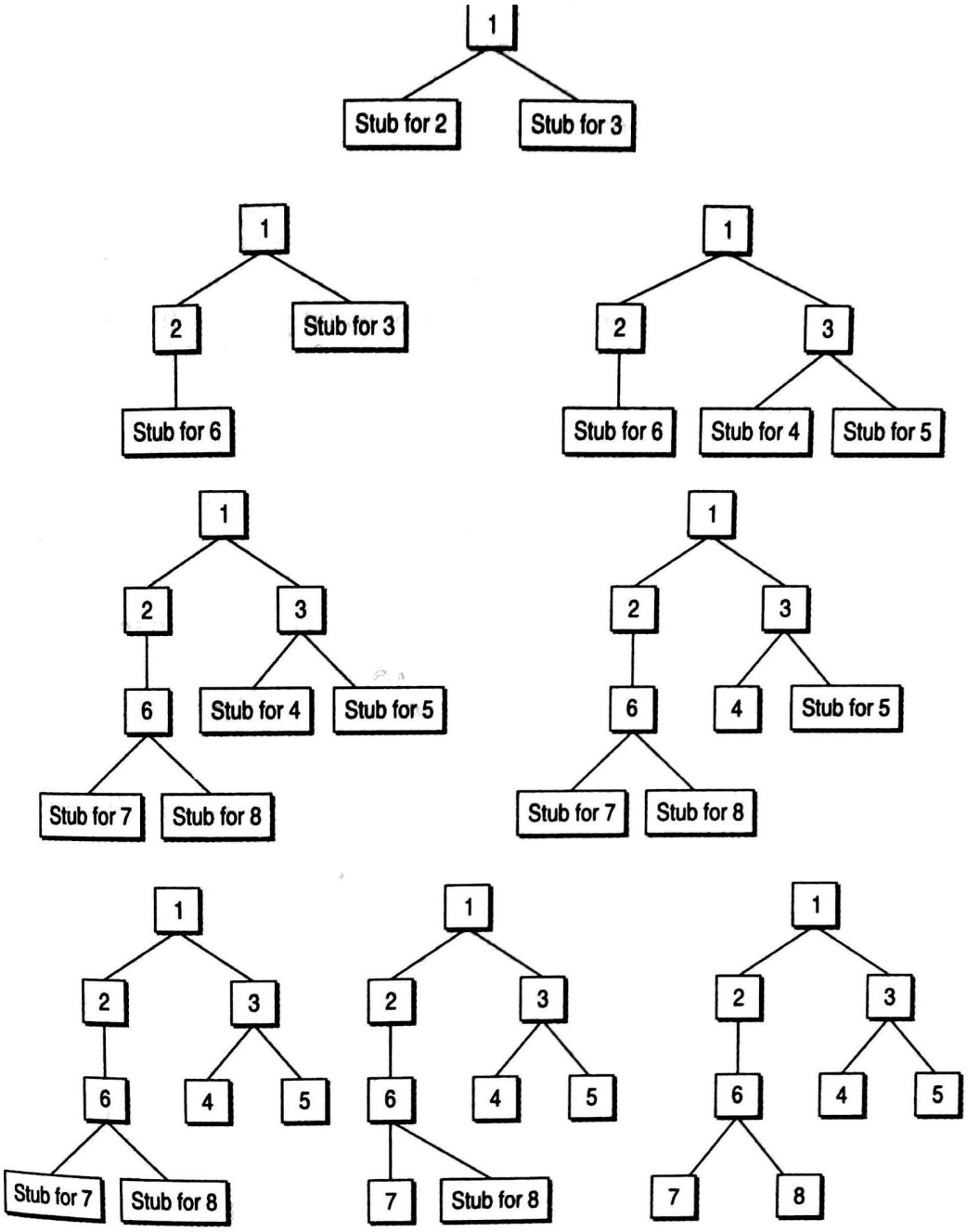
1. Depth first Integration: In this type, all modules on major control path of the design hierarchy are integrated first. In this example shown in fig. modules 1, 2, 6, 7/8 will be integrated first. Next, modules 1, 3, 4/5 will be integrated.

2. Breadth first Integration: In this type, all modules directly subordinate at each level, moving across the design hierarchy horizontally, are integrated first. In the example shown in fig.modules 2 and 3 will be integrated first. Next, modules 6,4 and 5 will be integrated . Modules 7 and 8 will be integrated last.

Procedure:

The procedure for Top-Down integration process is discussed in the following steps:

1. Start with the top or initial module in the software. Substitute the stubs for all the subordinate of the top module. Test the top module.
2. After testing the top module, stubs are replaced one at a time with the actual modules for integration.
3. Perform testing on this recent integrated environment.
4. Regression testing may be conducted to ensure that new errors have not appeared.
5. Repeat steps 2-4 for whole design hierarchy.



(iii) **Which features are included in test approach while planning test?**

(4 Marks)

Ans:

Like any project, the testing also should be driven by a plan. The test plan acts as the anchor for the execution, tracking and reporting of the entire testing project. Activities of test plan:

1. Scope Management: Deciding what features to be tested and not to be tested.
2. Deciding Test approach /strategy: Which type of testing shall be done like configuration, integration, localization etc.
3. Setting up criteria for testing: There must be clear entry and exit criteria for different phases of testing. The test strategies for the various features and combinations determined how these features and combinations would be tested.
4. Identifying responsibilities, staffing and training needs
5. Identifying resource requirements
6. Identifying test deliverables
7. Testing tasks: size and effort estimation

(iv) **Which parameters are considered while writing good defect report? Also write contents of defecttemplate.**

(Parameters- 2 Marks, contents of defect report-2 Marks)

Ans:

A defect report documents an anomaly discovered during testing. It includes all the information needed to reproduce the problem, including the author, release/build number, open/close dates, problem area, problem description, testenvironment, defect type, how it was detected, who detected it, priority, severity, status, etc.

After uncovering a defect (bug), testers generate a formal defect report. The purpose of a defect report is to state the problem as clearly as possible so that developers can replicate the defect easily and fix it.

DEFECT REPORT TEMPLATE

In most companies, a defect reporting tool is used and the elements of a report can vary. However, in general, a defect report can consist of the following elements.

ID	Unique identifier given to the defect. (Usually Automated)
Project	Project name.
Product	Product name.
Release Version	Release version of the product. (e.g. 1.2.3)
Module	Specific module of the product where the defect was detected.
Detected Build Version	Build version of the product where the defect was detected (e.g. 1.2.3.5)
Summary	Summary of the defect. Keep this clear and concise.
Description	Detailed description of the defect. Describe as much as possible but without repeating anything or using complex words. Keep it simple but comprehensive.
Steps to Replicate	Step by step description of the way to reproduce the defect. Number the steps.
Actual Result	The actual result you received when you followed the steps.
Expected Results	The expected results.

Attachments	Attach any additional information like screenshots and logs.
Remarks	Any additional comments on the defect.
Defect Severity	Severity of the Defect.
Defect Priority	Priority of the Defect.
Reported By	The name of the person who reported the defect.
Assigned To	The name of the person that is assigned to analyze/fix the defect.
Status	The status of the defect.
Fixed Build Version	Build version of the product where the defect was fixed (e.g. 1.2.3.9)

b) Attempt any ONEof the following: Marks 06

(i) **Describe V-model with labeled diagram. State it's any two advantages and disadvantages. Also write where it is applicable.**

(Explanation with diagram-3 Marks, two Advantages- 1Mark, twoDisadvantages-1 Mark, explanation of where it is applicable-1 Mark)

Ans:

V model means verification and validation model. It is sequential path of execution of processes. Each phase must be completed before the next phase begins.

Under V-model, the corresponding testing phase of the development phase is planned in parallel. So there is verification on one side of V & validation phase on the other side of V.

Verification Phase:

1. **Overall Business Requirement:** In this first phase of the development cycle, the product requirements are understood from customer perspective. This phase involves detailed communication with the customer to understand his expectations and exact requirements. The acceptance test design planning is done at this stage as business requirements can be used as an input for acceptance testing.
2. **Software Requirement**:Once the product requirements are clearly known, the system can be designed. The system design comprises of understanding & detailing the complete hardware, software & communication set up for the product under development. System test plan is designed based on system design. Doing this at earlier stage leaves more time for actual test execution later.
3. **High level design:** High level specification are understood & designed in this phase. Usually more than one technical approach is proposed & based on the technical & financial feasibility, the final decision is taken. System design is broken down further into modules taking up different functionality.
4. **Low level design:** In this phase the detailed integral design for all the system modules is specified. It is important that the design is compatible with the other modules in the system & other external system. Components tests can be designed at this stage based on the internal module design,

5. **Coding:** The actual coding of the system modules designed in the design phase is taken up in the coding phase. The base suitable programming language is decided base on requirements. Coding is done based on the coding guidelines & standards.

Validation:

1. **Unit Testing:** Unit testing designed in coding are executed on the code during this validation phase. This helps to eliminate bugs at an early stage.
2. **Components testing:** This is associated with module design helps to eliminate defects in individual modules.
3. **Integration Testing:** It is associated with high level design phase & it is performed to test the coexistence & communication of the internal modules within the system
4. **System Testing:** It is associated with system design phase. It checks the entire system functionality & the communication of the system under development with external systems. Most of the software & hardware compatibility issues can be uncovered using system test execution.

Acceptance Testing: It is associated with overall & involves testing the product in user environment. These tests uncover the compatibility issues with the other systems available in the user environment. It also uncovers the non-functional issues such as load & performance defects in the actual user environment.

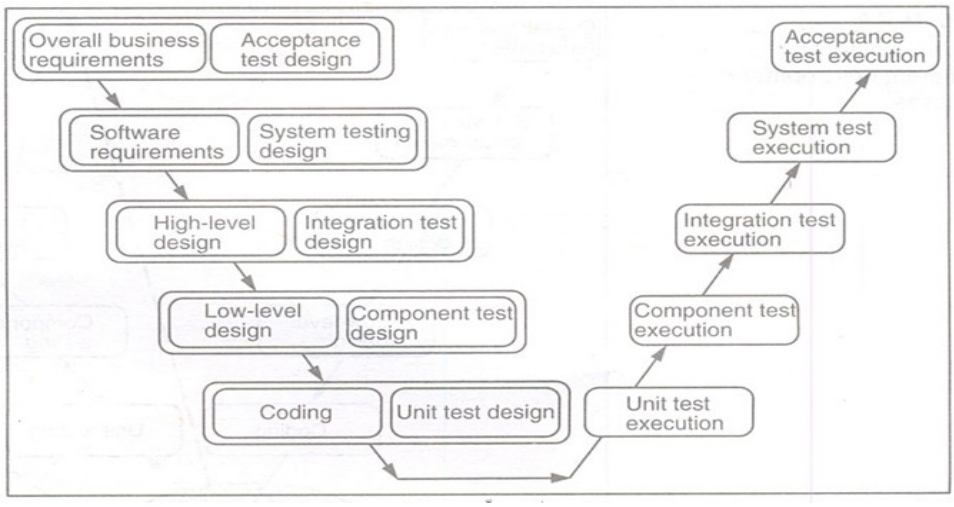
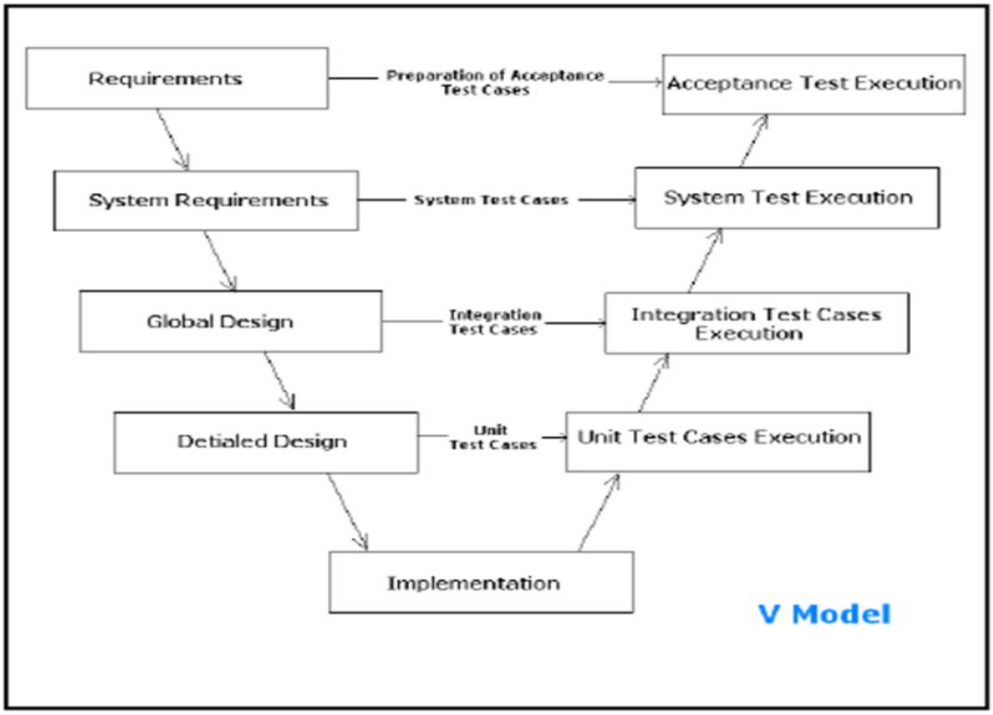


Fig: V-Model

OR



Advantages of V-model:

- Simple and easy to use.
- Testing activities like planning, test designing happens well before coding. This saves a lot of time. Hence higher chance of success over the waterfall model.
- Proactive defect tracking – that is defects are found at early stage.
- Avoids the downward flow of the defects.
- Works well for small projects where requirements are easily understood.

Disadvantages of V-model:

- Very rigid and least flexible.
- Software is developed during the implementation phase, so no early prototypes of the software are produced.
- If any changes happen in midway, then the test documents along with requirement documents has to be updated.

Where to use the V-model:

- The V-shaped model should be used for small to medium sized projects where requirements are clearly defined and fixed.

- The V-Shaped model should be chosen when ample technical resources are available with needed technical expertise.

(ii) **Explain concept of application of equivalence partitioning. How it is useful in result analysis of diploma?**

(Explanation of equivalence partitioning- 3 Marks, example of result analysis - 3 Marks)

Ans: Equivalence partitioning is a software technique that involves identifying a small set of representative input values that produce as much different output condition as possible. This reduces the number of permutation & combination of input, output values used for testing, thereby increasing the coverage and reducing the effort involved in testing.

The set of input values that generate one single expected output is called a partition. When the behavior of the software is the same for a set of values, then the set is termed as equivalence class or partition.

Example

Marks	**result**
Under 40	fail
40-100	pass

Based on the equivalence portioning technique, the equivalence partitions that are based on marks are given below:

Above	40 marks in subject	(valid input)
Between	40 and 100 marks	(valid input)
Below	40 marks	(invalid input)
	Negative marks	(Invalid Input)

2. **Attempt any <u>FOUR</u> of the following :** **Marks 16**

a) **List all objectives of testing.**

(Six objectives- 4 Marks)
Ans:

1. Finding defects which may get created by the programmer while developing the software.
2. Gaining confidence in and providing information about the level of quality.
3. To prevent defects.
4. To make sure that the end result meets the business and user requirements.
5. To ensure that it satisfies the BRS that is Business Requirement Specification and SRS that is System Requirement Specifications.
6. To gain the confidence of the customers by providing them a quality product.

b) **State purpose of code coverage. How it is used in analyzing coding of software?**

(Purpose– 2Marks, explanation -2 Marks)
Ans:

Subject Code: 17624 **Subject Name: Software Testing**

Purpose:

Code coverage is used to find out the percentage of code that is covered by testing.
It is used for analyzing the coding software: As product is realized in terms of program code we can run test cases to exercise the different parts of the code gets tested. Code coverage testing involves designing and executing test cases and finding out the percentage of the code that is covered by testing. It is found by adopting a technique called instrumentation of the code. This instrumented code can monitor and keep audit of what portions of code are covered. The tools also allow reporting on the portions of the code that are covered frequently, so that the critical or most-often portions of the code can be identified.
Code coverage testing is made up of the following types of coverage:
1. Statement coverage
2. Path coverage
3. Condition coverage
4. Function coverage
5. Branch coverage

c) **List any four advantages of acceptance test before launching any software.**

(Any 4 advantages- 4 Marks, 1 Mark each)
Ans*:*
1. Acceptance testing is phase after system testing that is normally done by the customer or representatives of the customer. Due to that customer themselves to quickly judge the quality of the product.
2. Determine whether the software is fit for the user.
3. Making users confident about product.
4. Determine whether a software system satisfies its acceptance criteria.
5. Enables the buyer to determine whether to accept the system or not

d) **What is test case? Which parameters are to be considered while documenting test cases?**

(Explanation of test case- 2 Marks, parameters-2 Marks)
Ans: Test case is a well-documented procedure designed to test the functionality ofthe feature in the system.
For designing the test case, it needs to provide set of inputs and its corresponding expected outputs.
Parameters:
1. Test case ID: is the identification number given to each test case.
2. Purpose: defines why the case is being designed.
3. Precondition: for running in the system can be defined, if required, in the test case.
4. Input: should not hypothetical. Actual inputs must be provided, instead of general inputs.

5. Expected outputs which should be produced when there is no failure.

e) **Explain defect life cycle to identify status of defect with proper labelled diagram.**

(Explanation -2 Marks, Diagram-2 Marks)

Ans:

Defect/Bug Life cycle:

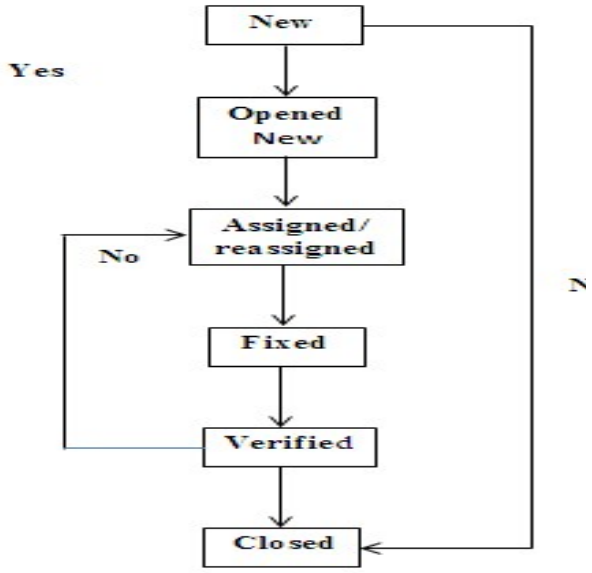
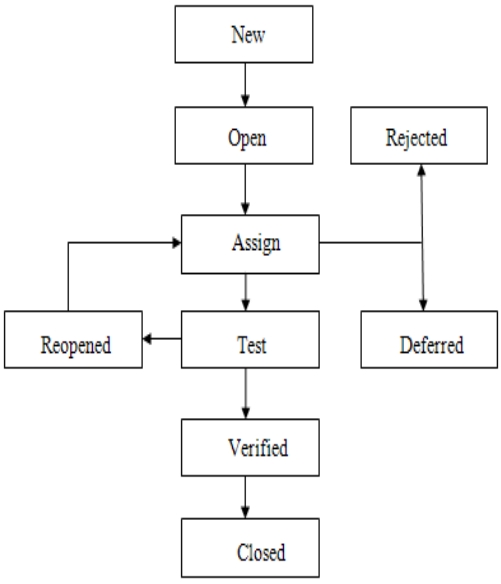


Fig: Defect life cycle

OR



1. **New:** When a defect is logged and posted for the first time. It's state is given as new.
2.Assigned: After the tester has posted the bug, the lead of the tester approves that the bugis genuine and he assigns the bug to corresponding developer and the developer team. It's state given as assigned.
3.Open: At this state the developer has started analyzing and working on the defect fix.
4.Fixed: When developer makes necessary code changes and verifies the changes then he/she Can make bug status as 'Fixed' and the bug is passed to testing team.

5.Pending retest: After fixing the defect the developer has given that particular code for retesting to the tester. Here the testing is pending on the testers end. Hence its status is pending retest.
6.Retest: At this stage the tester do the retesting of the changed code which developer has Given to him to check whether the defect got fixed or not.
7.Verified: The tester tests the bug again after it got fixed by the developer. If the bug is not present in the software, he approves that the bug is fixed and changes the status to "verified".
8. Reopen: If the bug still exists even after the bug is fixed by the developer, the tester changes the status to "reopened". The bug goes through the life cycle once again.
9.Closed: Once the bug is fixed, it is tested by the tester. If the tester feels that the bug no longer exists in the software, he changes the status of the bug to "closed". This state means that the bug is fixed, tested and approved.

10. Duplicate: If the bug is repeated twice or the two bugs mention the same concept of the bug, then one bug status is changed to "duplicate**".**
11. Rejected: If the developer feels that the bug is not genuine, he rejects the bug. Then the state of the bug is changed to "rejected".
12.**Deferred:** The bug, changed to deferred state means the bug is expected to be fixed in next releases. The reasons for changing the bug to this state have many factors. Some of them are priority of the bug may be low, lack of time for the release or the bug may not have major effect on the software.
13.Not a bug: The state given as "Not a bug" if there is no change in the functionality of the application. For an example: If customer asks for some change in the look and field of the application like change of color of some text then it is not a bug but just some change in the looks of the application.

f) **Enlist factors considered for selecting a testing tool for test automation.**

(Any four Factors- 4 Marks)

[Note: Criteria or guidelines can be considered as an answer]

Ans:
Criteria for Selecting Test Tools:
The Criteria's for selecting Test Tools are,
1. Meeting requirements;
2. Technology expectations;
3. Training/skills;
4. Management aspects.

1. Meeting requirements-
There are plenty of tools available in the market but rarely do they meet all the requirements of a given product or a given organization. Evaluating different tools for different requirements involve significant effort, money, and time. Given of the plethora of choice available, huge delay is involved in selecting and implementing test tools.

2. Technology expectations-
Test tools in general may not allow test developers to extends/modify the functionality of the framework. So extending the functionality requires going back to the tool vendor and involves additional cost and effort. A good number of test tools require their libraries to be linked with product binaries.

3. Training/skills-
While test tools require plenty of training, very few vendors provide the training to the required level. Organization level training is needed to deploy the test tools, as the user of the test suite are not only the test team but also the development team and other areas like configuration management.

4. Management aspects-

A test tool increases the system requirement and requires the hardware and software to be upgraded. This increases the cost of the already- expensive test tool.

OR

Guidelines for selecting a tool:

1. The tool must match its intended use. Wrong selection of a tool can lead to problems like lower efficiency and effectiveness of testing may be lost.
2. Different phases of a life cycle have different quality-factor requirements. Tools required at each stage may differ significantly.
3. Matching a tool with the skills of testers is also essential. If the testers do not have proper training and skill then they may not be able to work effectively.
4. Select affordable tools. Cost and benefits of various tools must be compared before making final decision.
5. Backdoor entry of tools must be prevented. Unauthorized entry results into failure of tool and creates a negative environment for new tool introduction.

3. **Attempt any <u>FOUR</u> of the following : Marks 16**

a) **Which are different methods of object oriented testing? Explain any one in detail.**

(Methods of Object Oriented Testing - 1 mark, description of any one testing - 3 Marks)

Ans:

Traditional testing techniques can be adopted in Object Oriented environment by using the following techniques:

- Method testing
- Class testing
- Interaction testing
- System testing
- Acceptance testing

Integration Testing: Object Orientation does not have a hierarchical control structure so conventional top-down and bottom up integration tests have little meaning. Integration testing can be applied in three different incremental strategies:

- Thread-based testing, which integrates classes required to respond to one input or event.

- Use-based testing, which integrates classes required by one use case. • Cluster testing, which integrates classes required to demonstrate one collaboration. Integration testing is performed using the following methods:

- For each client class, use the list of class methods to generate a series of random test sequences. Methods will send messages to other server classes.

- For each message that is generated, determine the collaborating class and the corresponding method in the server object.

- For each method in the server object (that has been invoked by messages sent from the client object), determine the messages that it transmits.

- For each of the messages, determine the next level of methods that are invoked and add these into the test sequence.

b) **Describe inspection process performed under white box testing.**

(Inspection process under white box testing - 4 Marks)

Ans:

Inspections are the most formal type of reviews in white box testing.

They are highly structured and require training for each participant.

Inspections are different from peer reviews and walkthroughs in that the person who presents the code, the presenter or reader, isn't the original programmer. These forces someone else to learn and understand the material being presented, potentially giving a different slant and interpretation at the inspection meeting. The other participants are called inspectors. Each is tasked with reviewing the code from a different perspective, such as a user, a tester, or a product support person. This helps bring different views of the product under review and very often identifies different bugs. One inspector is even tasked with reviewing the code backward—that is, from the end to the beginning—to make sure that the material is covered evenly and completely.

c) **Write steps to prepare test plan. Also write features to be tested.**

(Steps to prepare test plan, features to be tested - 4Marks)

Ans:

Steps to prepare test plan:

Like any project, the testing also should be driven by a plan. The test plan acts as the anchor for the execution, tracking and reporting of the entire testing project. Activities of test plan:

1. Scope Management: Deciding what features to be tested and not to be tested.

2. Deciding Test approach /strategy: Which type of testing shall be done like configuration, integration, localization etc.

3. Setting up criteria for testing: There must be clear entry and exit criteria for different phases of testing. The test strategies for the various features and combinations determined how these features and combinations would be tested.

4. Identifying responsibilities, staffing and training needs.

5. Identifying resource requirements

6. Identifying test deliverables.

7. Testing tasks: size and effort estimation.

Features to be tested are:

1. Lists/tables: Files, features and functions, inputs and outputs, error messages

2. Outlines :E.g., function list, top-level/user-visible functions, sub-functions (options or submenus), entry and exit conditions on fully parameterized methods

3. Matrices: List function/operation vs. test conditions, e.g., the save operation with the following conditions: disk full, almost full, write-protected

4. Notes : How to run test, expected results, special instructions, one-shot or regression test, what test is looking for, assumptions in the test

d) **What are the points considered while estimating impact of a defect? Also explain techniques to find defect in short.**

(Points considered while estimation of impacts-2 Marks, techniques to find defects - 2 Marks)

Ans:

Points to be considered while estimating the impacts of a defect are :

i. There is a strong relationship between the number of test cases and the number of function points.

ii. There is a strong relationship between the number of defects and the number of test cases and number of function points.

iii. The number of acceptance test cases can be estimated by multiplying the number of function points by 1.2.

iv. Acceptance test cases should be independent of technology and implementation techniques.

v. If a software project was 100 function points the estimated number of test cases would be 120.

vi. To estimate the number of potential defects is more involved.

Techniques to find defects are:

a) Quick Attacks: The quick-attacks technique allows you to perform a cursory analysis of a system in a very compressed timeframe. Even without a specification, you know a little bit about the software, so the time spent is also time invested in developing expertise. The skill is relatively easy to learn, and once you've attained some mastery your quick-attack session will probably produce a few bugs. Finally, quick attacks are quick. They can help you to make a rapid assessment. You may not know the requirements, but if your attacks yielded a lot of bugs, the programmers probably aren't thinking about exceptional conditions, and it's also likely that they made mistakes in the main functionality. If your attacks don't yield any defects, you may have some confidence in the general, happy-path functionality

b) Equivalence and Boundary Conditions: Boundaries and equivalence classes give us a technique to reduce an infinite test set into something manageable. They also provide a mechanism for us to show that the requirements are "covered".

c) Common Failure Modes: The heart of this method is to figure out what failures are common for the platform, the project, or the team; then try that test again on this build. If your team is new, or you haven't previously tracked bugs, you can still write down defects that "feel" recurring as they occur and start checking for them.

d) State-Transition Diagrams: Mapping out the application provides a list of immediate, powerful test ideas. Model can be improved by collaborating with the whole team to find "hidden" states transitions that might be known only by the original programmer or specification author. Once you have the map, you can have other people draw their own diagrams, and then compare theirs to yours. The differences in those maps can indicate gaps in the requirements, defects in the software, or at least different expectations among team members. e) Use Cases and Soap Opera Tests: Use cases and scenarios focus on software in its role to enable a human being to do something. Use cases and scenarios tend to resonate with business customers, and if done as part of the requirement process, they sort of magically generate test cases from the requirements. They make sense and can provide a straightforward set of confirmatory tests. Soap opera tests offer more power, and they can combine many test types into one execution.

f) Code-Based Coverage Models: Imagine that you have a black-box recorder that writes down every single line of code as it executes. Programmers love code coverage. It allows them to attach a number an actual, hard, real number, such as 75% to the performance of their unit tests, and they can challenge themselves to improve the score. Meanwhile, looking at the code that isn't covered also can yield opportunities for improvement and bugs.

g) Regression and High-Volume Test Techniques: People spend a lot of money on regression testing, taking the old test ideas described above and rerunning them over and over. This is generally done with either expensive users or very expensive programmers spending a lot of time writing and later maintaining those automated tests.

e) **Which are features for selecting static test tools? Also list any two available test tools (static).**

(Any 3 features for selecting static test tools - 3 Marks, listing of two available tools - 1 Mark)

Ans:

Features for selecting static test tools:

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).

Available static test tools are: 1. code coverage analyzer 2. Interface Analyzer

4. **a) Attempt any <u>THREE</u> of the following : Marks12**

(6.a.i) **Which are the different causes of software defects?**

(*Any 4 causes -1 mark each) (Diagram optional)*

Ans:

Different causes of software defects are as given below:

In software defects occur due to various reasons.

1. One of the extreme causes is the specification.

2. Specifications are the largest producer of defects.

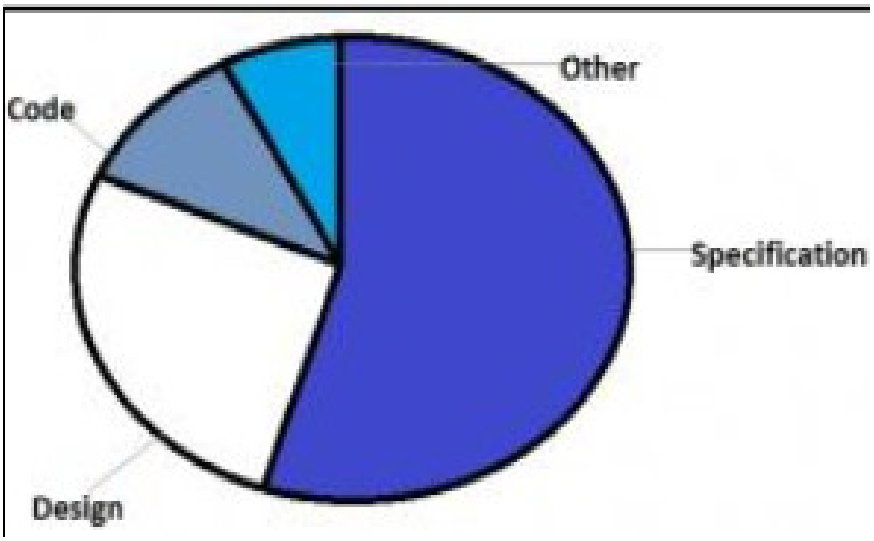
3. Either specifications are not written, specifications are not thorough enough, constantly changing or not communicated well to the development team.

4. Another bigger reason is that software is always created by human beings.

5. They know numerous things but are not expert and might make mistakes.

6. Further, there are deadlines to deliver the project on time. So increasing pressure and workload conduct in no time to check, compromise on quality and incomplete systems. So this leads to occurrence of defects in softwares.

Following diagram depicts the causes of defects in softwares:



(ii)What is white box testing? Classify static white box testing. State any one situation where white box testing used.

(White box testing - 1 Mark, classification - 1 mark, usages of static white box testing :

2 Marks)

Ans:

White box testing:

This is also known as glass box, clear box, and open box testing. In white box testing, test cases are created by looking at the code to detect any potential failure scenarios. The suitable input data for testing various APIs and the special code paths that need to be tested by analysing the source code for the application block. Therefore, the test plans need to be updated before starting white box testing and only after a stable build of the code is available. White box testing assumes that the tester can take a look at the code for the application block and create test cases that look for any potential failure scenarios. During white box testing, analyze the code of the application block and prepare test cases for testing the functionality to ensure that the class is behaving in accordance with the specifications and testing for

robustness. A failure of a white box test may result in a change that requires all black box testing to be repeated and white box testing paths to be reviewed and possibly changed.

Classification of white box testing:

i. Static Testing- Inspections, Structured Walkthroughs, Technical Review.

ii. Structural Testing-Code Functional Testing, Code Coverage Testing, Code Complexity Testing.

A simple website application as an example to explain the white box testing. . The end user will simply access the website, Login &Logout; this is very simple and day 2 days life example. As end users point of view users able to access the website from GUI, but inside there are lots of things going on to check the internal things are going right or not, the white box testing method is used. To explain this we have to divide this part in two steps. This is all is being done when the tester is testing the application using White box testing techniques.

(iii)Why is it essential to setup criteria for testing? List any three criteria in different situations.

(Need to setup criteria for testing - 2 Marks, 3 criteria's - 2 Marks)

Ans: There is a need to setup criteria for testing because:

i. An early start to testing reduces the cost, time to rework and error free software that is delivered to the client.

ii. Software Development Life Cycle (SDLC) testing can be started from the Requirements Gathering phase and lasts till the deployment of the software.

iii. It also depends on the development model that is being used. For example in Water fall model formal testing is conducted in the Testing phase, but in incremental model, testing is performed at the end of every increment/iteration and at the end the whole application is tested.

iv. Testing is done in different forms at every phase of SDLC like during Requirement gathering phase, the analysis and verification of requirements are also considered testing.

v. Reviewing the design in the design phase with intent to improve the design is also considered as testing.

vi. Testing performed by a developer on completion of the code is also categorized as Unit type of testing.

Any 3 criteria's in different situation are :

1. Start with the static white box testing procedure when the specifications of the software to be developed are ready.

2. Use the code coverage analyzer to test whether the whole code is getting executed and covered.

3. Perform unit testing as soon as one of the unit or sub module in the software is ready.

(iv)State limitations of manual testing. Write any four.

(Any 4 appropriate limitations of manual testing - 4Marks; 1 Mark each)

Ans:

Limitations of Manual Testing are as given below:

i. Manual testing is slow and costly.

ii. It is very labor intensive; it takes a long time to complete tests.

iii. 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.

iv. 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.

v. Lack of training is the common problem, although not unique to manual software testing.

vi. GUI objects size difference and color combinations are not easy to find in manual testing.

vii. Not suitable for large scale projects and time bound projects.

Batch testing is not possible, for each and every test execution Human user interaction is mandatory.

viii. Comparing large amount of data is impractical.

ix. Processing change requests during software maintenance takes more time.

b) Attempt any ONE of the following: Marks 06

(i) Describe use of load testing and security testing in performance testing of facility of online result display of MSBTE.

(Loadtesting - 3 Marks, security testing - 3 Marks)

Ans: In case of testing of facility of online result display of MSBTE, Performance testing, will be a non-functional testing technique performed to determine the system parameters in terms

of responsiveness and stability under various workload on the website. It will measures the quality attributes of the

Load testing - It is the simplest form of testing conducted to understand the behavior of the system under a specific load. Load testing will result in measuring number of critical transactions and load on the database, application server, etc., are also monitored. It will show how the software will respond if the number of students checking the result at the same time are more than specified.

Response time of the system, throughput, resourceutilization,and Maximum user load will be checked by this type of testing.

Security Testing: Security testing is a testing technique to determine if an information system protects data and maintains functionality as intended. It also aims at verifying 6 basic principles as Confidentiality of the MSBTE online result display system, Integrity of the software, authentication of the user who logs in to the system to check the result with the help of valid seat number and enrollment number, authorization, availability , and Non-repudiation of the system.

(ii) Explain defect management process with proper diagram.

(Defect management process Diagram - 3 Marks, description - 3 Marks)

Ans:

Defect Management Process diagram:

Defect Prevention -- Implementation of techniques, methodology and standard processes to reduce the risk of defects.

Deliverable Baseline -- Establishment of milestones where deliverables will be considered complete and ready for further development work. When a deliverable is baseline, any further changes are controlled. Errors in a deliverable are not considered defects until after the deliverable is baseline.

Defect Discovery -- Identification and reporting of defects for development team acknowledgment. A defect is only termed discovered when it has been documented and acknowledged as a valid defect by the development team member(s) responsible for the component(s) in error.

Defect Resolution -- Work by the development team to prioritize, schedule and fix a defect, and document the resolution. This also includes notification back to the tester to ensure that the resolution is verified.

Process Improvement -- Identification and analysis of the process in which a defect originated to identify ways to improve the process to prevent future occurrences of similar defects. Also the validation process that should have identified the defect earlier is analyzed to determine ways to strengthen that process.

Management Reporting -- Analysis and reporting of defect information to assist management with risk management, process improvement and project management.

5. Attempt anyTWOof the following: Marks 16

Subject Code: 17624 **Subject Name: Software Testing**

a) **What is decision table? How to use decision tables for creating test-design for credit card example?**

(Definition – 4Marks, Example - 4 Marks)

Ans:

- A **decision table** is a good way to deal with combinations of things (e.g. inputs). This technique is sometimes also referred to as a cause-effect' table. The first task is to identify a suitable function or subsystem which reacts according to a combination of inputs or events. The system should not contain too many inputs otherwise the number of combinations will become unmanageable. It is better to deal with large numbers of conditions by dividing them into subsets and dealing with the subsets one at a time. Once you have identified the aspects that need to be combined, then you put them into a table listing all the combinations of True and False for each of the aspects.

Conditions	Rule 1	Rule 2	Rule 3	Rule 4:
Pin Number	T	T	T	F
Payment Detail	T	F	F	T
Overdue details	F	T	T	F

b) Enlist components of usability testing. Which features of software were tested in usability test?

(Components -4Marks, features -4Marks)

Ans:

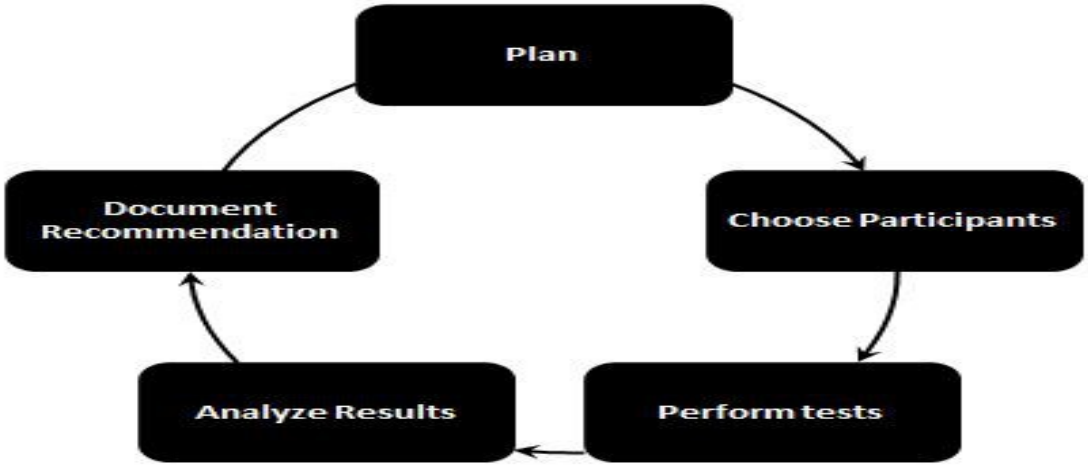
i. Usability testing, a non-functional testing technique that is a measure of how easily the system can be used by end users.

ii. It is difficult to evaluate and measure but can be evaluated based on the below parameters:

- Levels of Skill required learn/use the software. It should maintain the balance for both novice and expert user.
- Time required to get used to in using the software.

- The measure of increase in user productivity if any.
- Assessment of a user's attitude towards using the software.
- Usability testing, a non-functional testing technique that is a measure of how easily the system can be used by end users.
    - It is difficult to evaluate and measure but can be evaluated based on the below parameters:
    - Levels of Skill required learn/use the software. It should maintain the balance for both novice and expert user.
- Time required to get used to in using the software.
    - The measure of increase in user productivity if any.
    - Assessment of a user's attitude towards using the software.

Usability Testing Process:



Usability Testing Process

c) How test case specifications useful in designing test cases?

(Specifications (minimum 4) -4Marks, their justification -4Marks)

Ans: The test case specifications should be developed from the test plan and are the second phase of the test development life cycle. The test specification should explain "how" to implement the test cases described in the test plan. Test case specifications are useful as it enlists the specification details of the items.

Test Specification Items are must for each test specification should contain the following items:

1. **Case No**.: The test case number should be a three digit identifier of the following form: c.s.t, where: c- is the chapter number, s- is the section number, and t- is the test case number.
2. **Title:** is the title of the test.
3. **Programme:** is the program name containing the test.

4. **Author:** is the person who wrote the test specification.
5. **Date:** is the date of the last revision to the test case.
6. **Background:** (Objectives, Assumptions, References, Success Criteria): Describes in words how to conduct the test.
7. **Expected Error(s):** Describes any errors expected
8. **Reference(s):** Lists reference documentation used to design the specification.

9. **Data:** (Tx Data, Predicted Rx Data): Describes the data flows between the Implementation under Test (IUT) and the test engine.
10. **Script:** (Pseudo Code for Coding Tests): Pseudo code (or real code) used to conduct the test.

6. Attempt any FOUR ofthe following: Marks 16

a) **State any four objectives of user documentation testing. How these are useful in planning user documentation test?**

(One Mark each for objectives)

Ans:

Documentation testing is a non-functional type of software testing.

1. It is a type of non-functional testing.

2. Any written or pictorial information describing, defining, specifying, reporting, or certifying activities, requirements, procedures, or results'. Documentation is as important to a product's success as the product itself. If the documentation is poor, non-existent, or wrong, it reflects on the quality of the product and the vendor.

3. As per the IEEE Documentation describing plans for, or results of, the testing of a system or component, Types include test case specification, test incident report, test log, test plan, test procedure, test report. Hence the testing of all the above mentioned documents is known as documentation testing.

4. This is one of the most cost effective approaches to testing. If the documentation is not right: there will be major and costly problems. The documentation can be tested in a number of different ways to many different degrees of complexity. These range from running the documents through a spelling and grammar checking device, to manually reviewing the documentation to remove any ambiguity or inconsistency.

5. Documentation testing can start at the very beginning of the software process and hence save large amounts of money, since the earlier a defect is found the less it will cost to be fixed.

b) **Describe Graphical User Interface (GUI) testing and its important traits.**

(Concept of GUI Testing - 2 Marks, traits/Guidelines - 2 Marks)

Ans:

i. GUI testing is a testing technique in which the application's user interface is tested whether the application performs as expected with respect to user interface behaviour.

ii. GUI Testing includes the application behaviour towards keyboard and mouse movements and how different GUI objects such as toolbars, buttons, menu bars, dialog boxes, edit fields, lists, behaviour to the user input.

GUI Testing Guidelines

- Check Screen Validations
- Verify All Navigations
- Check usability Conditions
- Verify Data Integrity
- Verify the object states
- Verify the date Field and Numeric Field Formats

GUI Automation Tools

Following are some of the open source GUI automation tools in the market:

Product	Licensed Under	URL
AutoHotkey	GPL	http://www.autohotkey.com/
Selenium	Apache	http://docs.seleniumhq.org/
Sikuli	MIT	http://sikuli.org
Robot Framework	Apache	www.robotframework.org
Water	BSD	http://www.watir.com/
Dojo Toolkit	BSD	http://dojotoolkit.org/

GUI Automation

c) **How performance testing is performed? List steps involved in it?**

(Performance Testing - 2Marks, steps involved - 2 Marks)

Ans:

i. Performance testing, a non-functional testing technique performed to determine the system parameters in terms of responsiveness and stability under various workload.

ii. Performance testing measures the quality attributes of the system, such as scalability, reliability and resource usage.

Techniques are:

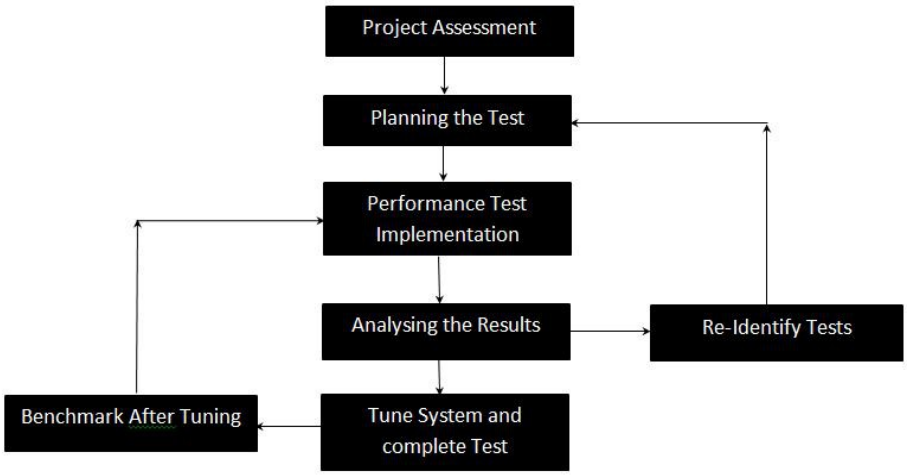
- **Load testing -** It is the simplest form of testing conducted to understand the behaviour of the system under a specific load. Load testing will result in measuring important business critical transactions and load on the database, application server, etc., are also monitored.

- **Stress testing -** It is performed to find the upper limit capacity of the system and also to determine how the system performs if the current load goes well above the expected maximum.

- **Soak testing -** Soak Testing also known as endurance testing, is performed to determine the system parameters under continuous expected load. During soak tests the parameters such as memory utilization is monitored to detect memory leaks or other performance issues. The main aim is to discover the system's performance under sustained use.

- **Spike testing -** Spike testing is performed by increasing the number of users suddenly by a very large amount and measuring the performance of the system. The main aim is to determine whether the system will be able to sustain the workload.

Performance Testing Process:



d) **Which are the various hardware and software required / recommended by project manager?**

(List selection criteria and description in short - 1 Mark each,)

Ans:

1. At the most basic level, project management products will help your organization to manage projects from start to finish, and allow employees at different levels to have an input into the process.

2. Project management software has been around for a number of years now and as a result, it does far more than just manage the projects themselves.

3. Project applications can also carry out scheduling, cost control and budget management, resource allocation, collaboration, communication, quality management and documentation or administration.

4. The aim with these is to handle all aspects and complexities of larger projects and help keep costs down.

e) **Explain defect classification.**

(Explanation of defect - 2 Marks, 2 Marks for Classification any four minimum)

Ans:

A Software Defect / Bug is a condition in a software product which does not meet a software requirement (as stated in the requirement specifications) or end-user expectations (which may not be specified but are reasonable). In other words, a defect is an error in coding or logic that causes a program to malfunction or to produce incorrect/unexpected results.

•A program that contains a large number of bugs is said to be buggy.

•Reports detailing bugs in software are known as bug reports.

•Applications for tracking bugs are known as bug tracking tools.

•The process of finding the cause of bugs is known as debugging.

•The process of intentionally injecting bugs in a software program, to estimate test coverage by monitoring the detection of those bugs, is known as debugging.

CLASSIFICATION

Software Defects/ Bugs are normally classified as per:

•Severity / Impact

•Probability / Visibility

•Priority / Urgency

•Related Dimension of Quality

•Related Module / Component

•Phase Detected

•Phase Injected

OR

Defect Classification:

Requirements and specification defect: Requirement related defects arise in a product when one fails to understand what is required by the customer. These defects may be due to customer gap, where the customer is unable to define his requirements, or producer gap, where developing team is not able to make a product as per requirements. Defects injected in early phases can persist and be very difficult to remove in later phases. Since any requirements documents are written using natural language representation, there are very often occurrences of ambiguous, contradictory, unclear, redundant and imprecise requirements. Specifications are also developed using natural language representations.

Design Defects: Design defects occur when system components, interactions between system components, interactions between the outside software/hardware, or users are incorrectly designed. This covers in the design of algorithms, control, logic/ data elements, module interface descriptions and external software/hardware/user interface descriptions. Design defects generally refer to the way of design creation or its usage while creating a product. The customer may or may not be in a position to understand these defects, if structures are not correct. They may be due to problems with design creation and implementation during software development life cycle.

Coding Defects: Coding defects may arise when designs are implemented wrongly. If there is absence of development/coding standards or if they are wrong, it may lead to coding defects. Coding defects are derived from errors in implementing the code. Coding defect classes are closely related to design defect classes especially if pseudo code has been used for detailed design. Some coding defects come from a failure to understand programming language constructs, and miscommunication with the designers. Others may have transcription or omission origins. At times it may be difficult to classify a defect as a design or as a coding detect.

Testing Defect: Testing defect are defects introduced in an application due to wrong testing , or defects in the test artifact leading to wrong testing. Defects which cannot be reproduced , or

are not supported by requirement or are duplicate may represent a false call .In this defects includes

1. Test-design defect: test-design defect refers to defects in test artifacts. there can be defects in test plans, test scenarios, test cases and test data definition which can lead to defect in software.

2. Test-environment defect: this defect may arise when test environment is not set properly. Test environment may be comprised of hardware, software, simulator and people doing testing.

3. Test-tool defects: any defects introduced by a test tool may be very difficult to find and resolve, as one may have to find the defect using manual test as against automated tools.

f) **Which different benefits help to recommend automated testing? Write advantages of switching to automated testing.**

(Need / Benefit – 2 Marks, Advantages – 2 Marks)

Ans:

NEED of automated testing

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.

1. Automated Software Testing Saves Time and Money

i. Software tests have to be repeated often during development cycles to ensure quality. Every time source code is modified software tests should be repeated.
ii. For each release of the software it may be tested on all supported operating systems and hardware configurations.
iii. Manually repeating these tests is costly and time consuming. Once created, automated tests can be run over and over again at no additional cost and they are much faster than manual tests.
iv. Automated software testing can reduce the time to run repetitive tests from days to hours.
v. A time savings that translates directly into cost savings.

2. Testing Improves Accuracy

i. Even the most conscientious tester will make mistakes during monotonous manual testing.
ii. Automated tests perform the same steps precisely every time they are executed and never forget to record detailed results.

3. Increase Test Coverage

i. Automated software testing can increase the depth and scope of tests to help improve software quality.
ii. Lengthy tests that are often avoided during manual testing can be run unattended.
iii. They can even be run on multiple computers with different configurations.
iv. Automated software testing can look inside an application and see memory contents, data tables, file contents, and internal program states to determine if the product is behaving as expected.

v. Automated software tests can easily execute thousands of different complex test cases during every test run providing coverage that is impossible with manual tests.

vi. Testers freed from repetitive manual tests have more time to create new automated software tests and deal with complex features.