CHAPTER 2

UNIT - 2

Fundamentals of Testing

Syllabus Topic : Introduction, What is Testing?

2.1 Introduction

2.1.1 What is Testing?

Q. 2.1.1 What is testing? Why is testing needed? (Ref. Secs. 2.1.1 and 2.1.2) (5 Marks)

- "When we are testing something we are checking whether it is OK".
- Software testing is a process used to identify the correctness, completeness and quality of developed software, includes a set of activities conducted with the intent of finding errors in software so that it could be corrected before the product is released to the end users.
- In simple words, software testing is an activity to check that the software system is defect free.
- Testing is a process of all life cycle activities to determine that a product satisfies the specified requirements (we check all the requirements mentioned in the Software Requirement Specification document are converted into functionalities in the software) and they fit the purpose (We check that it does whatever it is designed for).
- We think software testing as a means of detecting faults or defects that will lead to failure when put to use. Finding defects helps in understanding the risks associated with using the software and fixing the defect improves the quality of the software.
- Identifying defects has another benefit. With root cause analysis of the defect it helps to improve the development processes and make fewer mistakes.

Syllabus Topic : Necessity of Testing

2.1.2 Necessity of Testing

Q. 2.1.2 What is testing? Why is testing needed? (Ref. Sec. 2.1.1 and 2.1.2) (5 Marks)

- Testing is necessary because we all make mistakes. Some of those mistakes are unimportant, but some of them are expensive or dangerous.
- We need to check everything and anything we produce because things can always go wrong - humans make mistakes all the time.
- Software bugs can potentially cause monetary and even human loss.
- It is important to ensure that the application should not result into any failures.
- Software testing is important for the following reasons :
  1. It is required to point out the defects and errors that were made during the development of the software.
  2. It helps businesses understand an actual and expected outcome so that they can improve the quality of their products.
  3. Quality products increase the confidence of the customer for a product.
  4. Ensures customer about the product's reliability.
  5. Improves the consistency and performance of the software.

Syllabus Topic : Fundamental Test Process

2.2 Fundamental Test Process

Q. 2.2.1 Explain the fundamental process of testing. (Ref. Sec. 2.2) (5 Marks)

The Fundamental Test Process comprises five activities :

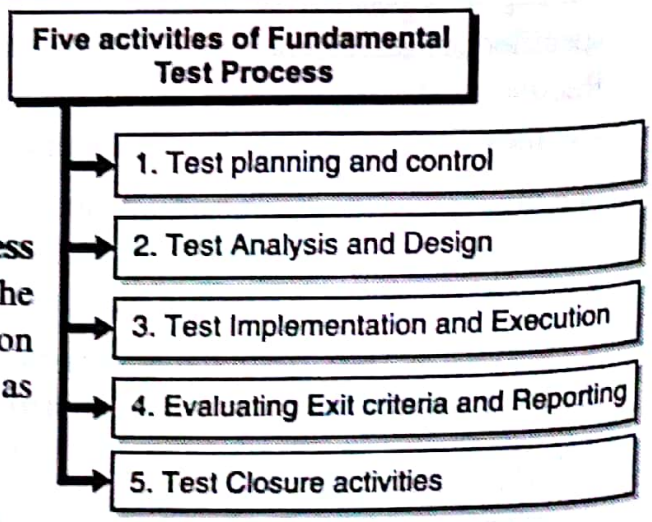

Fig. 2.2.1

➜ **(i) Test planning and control**

Test Planning is a **Fundamental Test Process** which defines the objective and goal of the testing process. All good testing is based upon good test planning. Tasks of this step are as follows :

- Deciding the scope and risk of testing
- Deciding the overall approach of testing

- Scheduling test analysis and design process
- Assigning resources to different activities

Test control has the following major tasks :

- To measure and analyse the results of reviews and testing.
- To monitor and document progress, test coverage and exit criteria.
- To provide information on testing.
- To initiate corrective actions.
- To make decisions.

➜ (ii) Test Analysis and Design

Test analysis and Test Design has the following major tasks :

- To review the test basis. The test basis is the information on which test cases are based, such as requirements, design specifications, product risk analysis, architecture and interfaces
- To identify test conditions
- To design the tests
- To design the test environment set-up and identify the required infrastructure and tools

➜ (iii) Test Implementation and Execution

Test implementation and execution is the Process in which actual work is done. The tasks includes :

- Creating test suites from test executions
- Checking test environments, updating traceability between test basis and test cases
- Executing test procedures using test execution tools
- Checking actual result with expected results
- Reporting errors & creating Incident reports

➜ (iv) Evaluating Exit criteria and Reporting

Based on the risk assessment of the project we will set the criteria for each test level against which we will measure the "enough testing". These criteria vary from project to project and are known as **exit criteria**. Exit criteria come into picture, when :

- Maximum test cases are executed with certain pass percentage.
- Bug rate falls below certain level.
- When achieved the deadlines.

Evaluating exit criteria has the following major tasks :

- To check the test logs against the exit criteria specified in test planning.
- To assess if more test are needed or if the exit criteria specified should be changed.
- To write a test summary report for stakeholders.

➜ (v) Test Closure activities

Test closure activities are done when software is delivered. The testing can be closed for the other reasons also like :

- When all the information has been gathered which are needed for the testing.
- When a project is cancelled.
- When some target is achieved.
- When a maintenance release or update is done.

Test closure activities have the following major tasks :

- To check which planned deliverables are actually delivered and to ensure that all incident reports have been resolved.
- To finalize and archive test ware such as scripts, test environments, etc. for later reuse.
- To handover the test ware to the maintenance organization. They will give support to the software.
- To evaluate how the testing went and learn lessons for future releases and projects.

Syllabus Topic : The Psychology of Testing

2.3 The Psychology of Testing

Q. 2.3.1 What is the psychology behind testing? (Ref. Sec. 2.3) (5 Marks)

- We will discuss the psychological factors that affect testing.
- Human beings reaction in this complex world of happenings varies widely with respect to situations, surroundings, emotions, need, requirements, time frame, money, visualization, belief, education, knowledge, expertise, intuition and so on.
- Such complex is the nature of human being and certainly there's no exception at work place too.
- The quality of the job done by the software tester is directly proportional to his or her psychological maturity and profoundness acquired, adopted and developed with age and experience.

- The mindset of a tester is different from a developer. When we build something we are working positively to solve problems and realize a product that meets some need.
- However, when we test a product, we are looking for a defect and thus are critical of it. The quality of the job done by the software tester is directly proportional to his or her psychological maturity and profoundness acquired, adopted and developed with age and experience.
- This degree of independence avoids author bias and is often more effective at finding defects and failures. There is several level of independence in software testing which is listed here from the lowest level of independence to the highest :
  - Tests by the person who wrote the item.
  - Tests by another person within the same team, like another programmer.
  - Tests by the person from some different group such as an independent test team.
  - Tests by a person from a different organization or company, such as outsourced testing or certification by an external body.
- We all make mistakes and we sometimes get annoyed and upset or depressed when someone points them out. So, when as testers we run a test which is a good test from our viewpoint because we found the defects and failures in the software.
- But at the same time we need to be very careful as how we react or report the defects and failures to the programmers. We are pleased because we found a good bug but how will the requirement analyst, the designer, developer, project manager and customer react.
  - The people who build the application may react defensively and take this reported defect as personal criticism.
  - The project manager may be annoyed with everyone for holding up the project.
  - The customer may lose confidence in the product because he can see defects.
- Because testing can be seen as destructive activity we need to take care while reporting our defects and failures as objectively and politely as possible.

Syllabus Topic : Historical Perspective of Testing

2.3.1 Historical Perspective of Testing

Q. 2.3.2 What is the history behind testing? (Ref. Sec. 2.3.1) (5 Marks)

- The origin of software testing can be traced back to the fifties, when the primary method of testing was debugging.
- In late seventies, the approach evolved to one of the destruction. The testers would break down the codes to find the gaps in it. This method was effective until the prevention oriented methodologies were discovered which made software application more robust.

- In 1979, Glenford J Myers distinguished between debugging which means identifying and eliminating bugs in software code and testing the software in real world.

Syllabus Topic : Definition of Testing

2.3.2 Definition of Testing

Q. 2.3.3 Define Testing. List and explain the approaches to testing. (Ref. Sec. 2.3.2 and 2.4) **(5 Marks)**

- **Software testing** is a process, to evaluate the functionality of a software application with an intent to find whether the developed software met the specified requirements or not and to identify the defects to ensure that the product is defect free in order to produce the quality product.
- Software testing is primarily a broad process that is composed of several interlinked processes. The primary objective of software testing is to measure software health along with its completeness in terms of core requirements.
- Software testing involves examining and checking software through different testing processes. The objectives of these processes can include :
  - Verifying software completeness in regards to functional/business requirements
  - Identifying technical bugs/errors and ensuring the software is error-free
  - Assessing usability, performance, security, localization, compatibility and installation

Syllabus Topic : Approaches to Testing

2.4 Approaches to Testing

Q. 2.4.1 Define Testing. List and explain the approaches to testing. (Ref. Sec. 2.3.2 and 2.4) **(5 Marks)**

- We start by 'testing-in-the-small' and head toward 'testing-in-the-large'.
- For conventional software :
  - Our initial focus is the module (component)
  - Integration of modules follows :

(A) Unit Testing

Unit testing focuses verification effort on the smallest unit of software design – the software component or module.

☞ Unit Test Considerations

The module interface is tested to ensure that information properly flows into and out of the program unit under test.

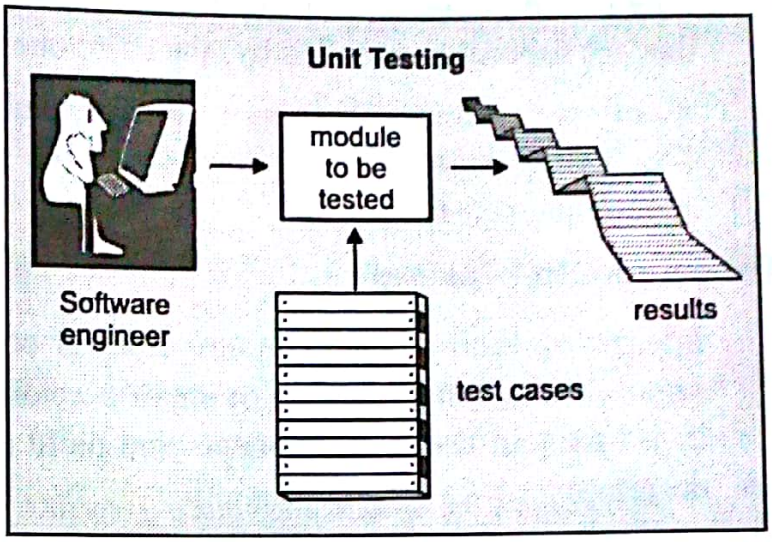

Fig. 2.4.1

- The local data structure is checked to ensure that the data stored temporarily retains its Integrity during all steps in an algorithm's execution.
- Boundary conditions are tested to make sure that the module operates properly at boundaries established to restrict processing.
- All independent paths (basis paths) through the control structure are exercised to ensure that all statements in a module have been executed at least once.
- And finally, all error handling paths are tested.

☞ Unit Test Procedures

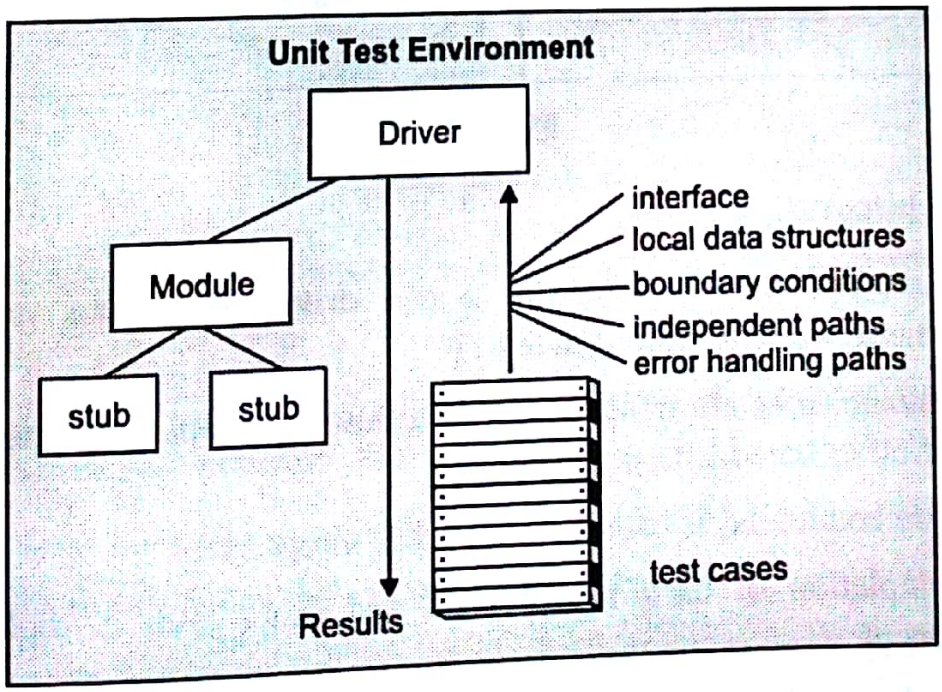

Fig. 2.4.2

- Each test case should be combined with a set of expected results. Because a component is not a stand-alone program, driver and/or stub software must be developed for each unit test.
- A driver is nothing more than a "main program" that accepts test case data, passes such data to the component (to be tested), and prints relevant results. Stubs replace modules that are subordinate (called by) the component to be tested.
- A stub or "dummy subprogram" uses the subordinate modules interface, may do minimal data manipulation, prints verification of entry, and returns control to the module undergoing testing.

(B) Integration Testing

Integration testing is a technique used for constructing the program structure while at the same time carrying out tests to uncover errors related with interfacing. The main objective is to take unit tested components and build a program structure that has been dictated by design.

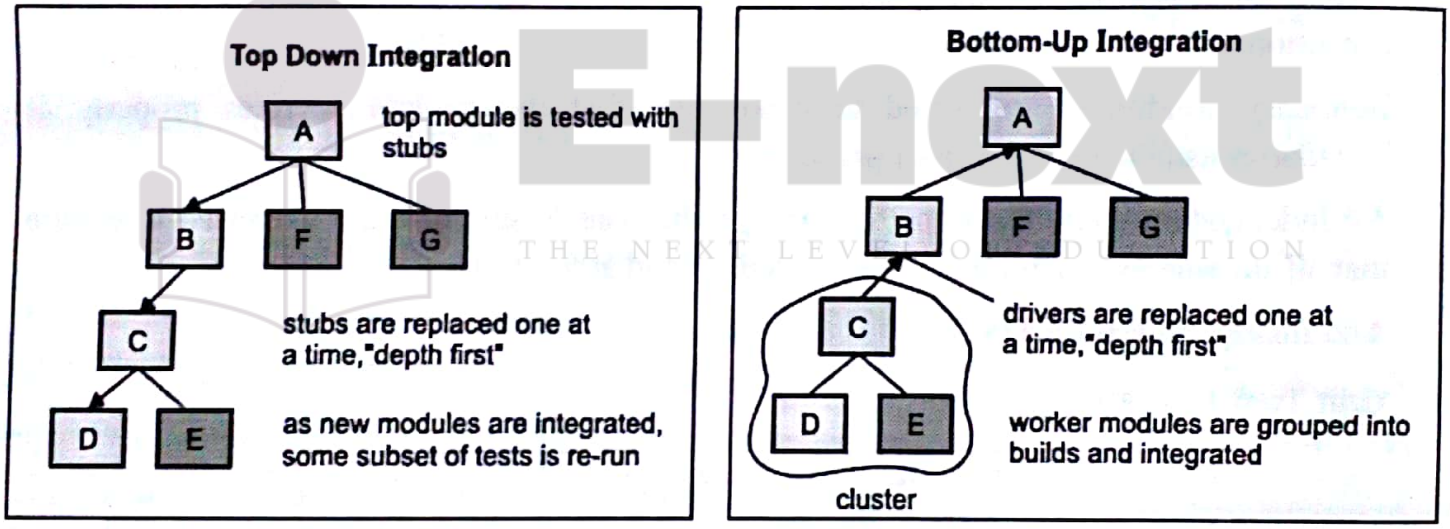

Fig. 2.4.3

(i) Top-down integration testing

- Main control module used as a test driver and stubs are substitutes for components directly subordinate to it.
- Subordinate stubs are replaced one at a time with real components (following the depth-first or breadth-first approach).
- Tests are conducted as each component is integrated.
- On completion of each set of tests and other stub is replaced with a real component.
- Regression testing may be used to ensure that new errors not introduced.

(ii) Bottom-up integration testing

- Low level components are combined in clusters that perform a specific software function.
- A driver (control program) is written to coordinate test case input and output.
- The cluster is tested.
- Drivers are removed and clusters are combined moving upward in the program structure.
- Regression testing (check for defects propagated to other modules by changes made to existing program)
- 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.

(C) Validation Testing

- Ensure that each function or performance characteristic conforms to its specification.
- Deviations (deficiencies) must be negotiated with the customer to establish a means for resolving the errors.
- Configuration review or audit is used to ensure that all elements of the software configuration have been properly developed, cataloged, and documented to allow its support during its maintenance phase.

(D) Acceptance Testing

- Making sure the software works correctly for intended user in his or her normal work environment.
- Alpha test (version of the complete software is tested by customer under the supervision of the developer at the developer's site).
- Beta test (version of the complete software is tested by customer at his or her own site without the developer being present).

Syllabus Topic : Testing during Development Life Cycle

2.5 Testing during Development Life Cycle

Q. 2.5.1 Explain how is testing carried out during SDLC? (Ref. Sec. 2.5) (5 Marks)

- It is a framework that defines activities that are performed throughout the software development process. The development process adopted for a project will depend on project aims and goals.
- There are numerous development life cycles that have been developed in order to achieve different required objectives.
- The most appropriate development process should be applied to each project. The models specify the various stages of the process and the order in which they are carried out.
- Phases of SDLC are as follows :

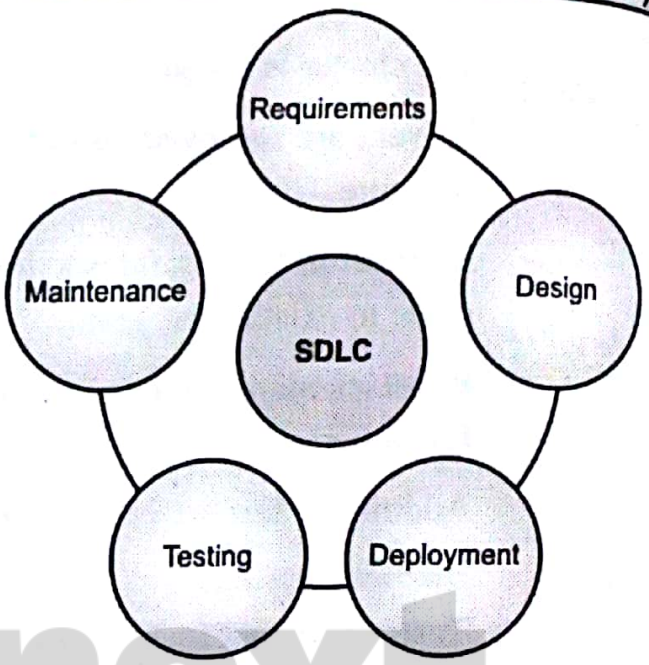

Fig. 2.5.1

1. Requirement Gathering

- This is the most important phase in software development life cycle. In this phase requirement for the software are collected from the customer/client. These requirements are provided in a document called Business Requirement Specification (BRS) or System Requirement Specification (SRS).
- All details and specifications of the product must be discussed with the customer. The development team analyses the requirements keeping in view the design and coding of the software.
- The requirements so gathered are then analyzed for their validity and possibility of incorporating them into the software system. The aim of requirement analysis is to capture the detail of each requirement so that everyone understands how each requirement is to be worked.

2. Design

- It has two steps : HLD - High Level Design - It gives the architecture of the software product to be developed and is done by architects and senior developers LLD - Low Level Design - It is done by senior developers.
- It describes how each and every feature in the product should work and how every component should work. Here, only the design will be there and not the code.

- The outcome from this phase is High Level Document and Low Level Document which works as an input to the next phase.

3. Coding

- It means translating the design into a computer readable language. Development team does the actual coding based on designed software and writes unit tests for each component to test the new codes written by them.
- This is the longest phase of the software development life cycle.

4. Testing

- After the code is developed, testing is carried out to verify the entire requirement specified by customer has been implemented.
- The aim of tester is to find out the gaps or defects within the system and also to verify that the software works as expected according to the requirements. It includes Unit testing, Integration testing and System testing.

5. Deployment

- After successful testing, the product is delivered/deployed to the customer for their use. The size of the project will determine the complexity of the deployment.
- The users can be trained on, or aided with the documentation on how to operate the software.
- A small round of testing is also performed on production to make sure of any environmental issues and any impact of new release.

Syllabus Topic : Requirement Traceability Matrix

2.6 Requirement Traceability Matrix (RTM)

Q. 2.6.1 What is Requirement Traceability Matrix? (Ref. Sec. 2.6) (5 Marks)

- Requirements Traceability Matrix (RTM) is used to trace the requirements to the tests that are needed to verify whether the requirements are fulfilled.
- The purpose of the Requirements Traceability Matrix is to ensure that all requirements defined for a system are tested.
- The requirements traceability matrix is usually developed in concurrence with the initial list of requirements (either the User Requirements Specification or Functional Requirements Specification).
- Assume we have total 5 requirements and total test cases are 10. Below is an example of RTM.

Table 2.6.1

Requirement Traceability		Business Requirement				
		BID001	BID002	BID003	BID004	BID005
Test Cases	TID001	X				
	TID002	X				
	TID003	X				
	TID004		X			
	TID005		X			
	TID006			X		
	TID007				X	
	TID008				X	X
	TID009					X
	TID0010					X

- Whenever a new test case is written it should be updated in this matrix.
- **Forward Traceability :** Mapping requirements to test cases is called Forward Traceability Matrix. It is used to ensure whether the project progresses in the desired direction. It makes sure that each requirement is tested thoroughly.
- **Backward or Reverse Traceability :** Mapping test cases to requirements is called Backward Traceability Matrix. It is used to ensure whether the current product remains on the right track. It makes sure that we are not expanding the scope of the project by adding functionality that is not specified in the requirements.

Syllabus Topic : Essentials of Software Testing

2.7 Essentials of Software Testing

Q. 2.7.1 What are essentials prerequisites for testing? (Ref. Sec. 2.7) (5 Marks)

Five essentials of software testing are as follows :

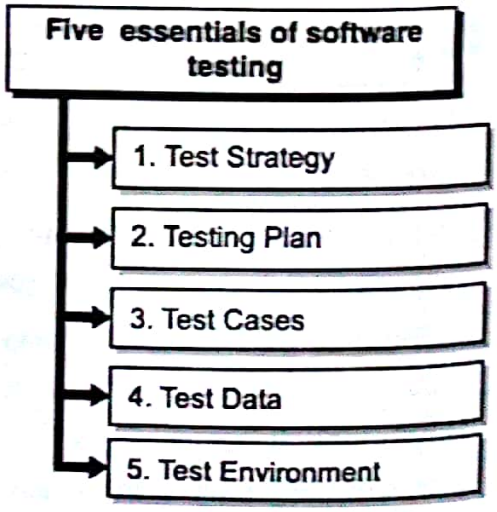

Fig. 2.7.1

1. Test Strategy

- Planning a strategy is an essential component in the process of attaining a successful result. One needs to take into account the possible number of risks involved, degree of accuracy of test results etc.
- A test strategy thus helps to uncover the facts like which type of testing best suits our requirements, the sequence in which the tests are to be performed etc.

- Basically, a test strategy helps to sort our tasks in a wise manner and that results in less wastage of time and efforts.

2. Testing Plan

- A test plan is a layout of how a project's testing activities are to be carried out. It is about classifying as to who will be assigned which task, in which order the tasks shall begin, the optimum time allocated to each task etc.
- A test plan is exercised after the test strategy is in place.

3. Test Cases

- Test cases are the set of conditions that needs to be checked to fulfil criteria. Test cases are generally prepared followed by the test strategy.
- A test case or test script is intended to check if the software functions as per the given set of requirements.
- A test case should be robust enough in order to ascertain that our expected results are delivered. A good test case also brings out the defects in the system.

4. Test Data

- Data is crucial. Test data is simply the input values to be passed to the system under test.
- For instance, our test data can be customer's phone number, customer's address, postal address etc.

5. Test Environment

- The environment here refers to the platform used for testing an application. The environment could be the operating system used, the automation tool used or servers/networks.

Syllabus Topic : Workbench

2.7.1 Workbench

Q. 2.7.2 What is a workbench? (Ref. Sec. 2.7.1) (5 Marks)

A workbench concept is a method of scheduling that how a particular action has to be executed. Workbench concept normally contains the following actions mentioned as below :

➜ **1. Input**

Input form is the very first phase of any workbench technique. As we all know that all chores desire some specific access standards or input and output arguments. So for whole workbench we require determined inputs. In the workbench technique, input form or entrance criteria is the foremost stage.

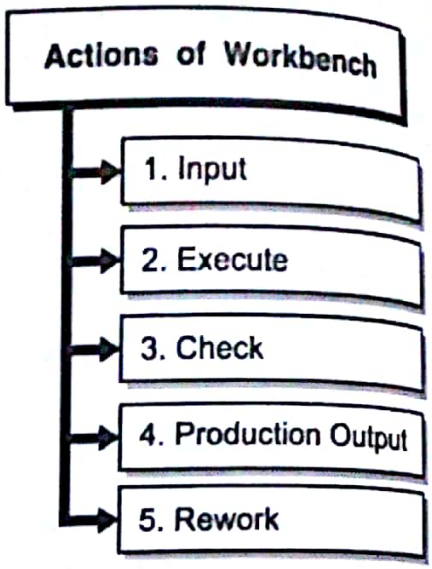

Fig. 2.7.2

➜ **2. Execute**

Execute activity is the second phase of the workbench approach. This phase is very important. This phase perform very prime objective of the workbench technique. This converts input forms into the predictable outputs.

➜ **3. Check :** This is the third phase of workbench technique. This phase confirm that the received output after the execution step accomplishes the wanted results.

➜ **4. Production Output :** Production output is the fourth phase of workbench technique. If the check is finished properly then the production output turns into the final phase of workbench technique.

➜ **5. Rework :** Throughout the third phase that is 'Check' activity, if the output is not as per expected then we require once more to begin from the second phase that is 'Execute' steps.

Syllabus Topic : Important Features of Testing Process

2.7.2 Important Features of Testing Process

Q. 2.7.3 Explain the important features of testing. (Ref. Sec. 2.7.2) **(5 Marks)**

➜ **1. Easy to operate**

High quality software can be tested in a better manner. This is because if the software is designed and implemented considering quality, then comparatively fewer errors will be detected during the execution of tests.

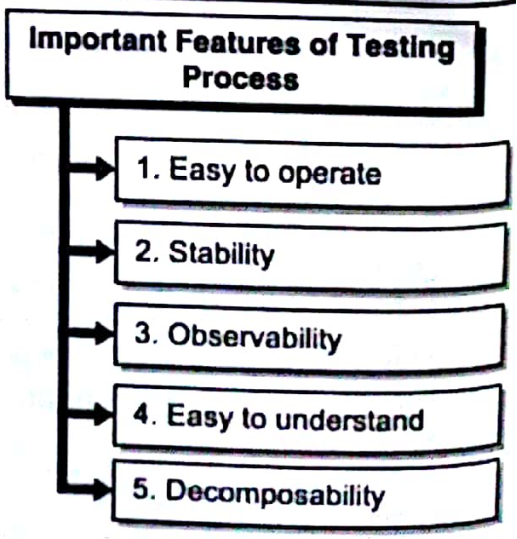

Fig. 2.7.3

➜ **2. Stability**

Software becomes stable when changes made to the software are controlled and when the existing tests can still be performed.

3. Observability

Testers can easily identify whether the output generated for certain input is accurate simply by observing it.

4. Easy to understand

- Software that is easy to understand can be tested in an efficient manner. Software can be properly understood by gathering maximum information about it.
- For example, to have a proper knowledge of the software, its documentation can be used, which provides complete information of the software code thereby increasing its clarity and making the testing easier.

5. Decomposability

By breaking software into independent modules, problems can be easily isolated and the modules can be easily tested.

Syllabus Topic : Misconceptions about Testing

2.7.3 Misconceptions about Testing

Q. 2.7.4 What are the misconceptions behind testing? (Ref. Sec. 2.7.3) (5 Marks)

1. Testing can find all bugs

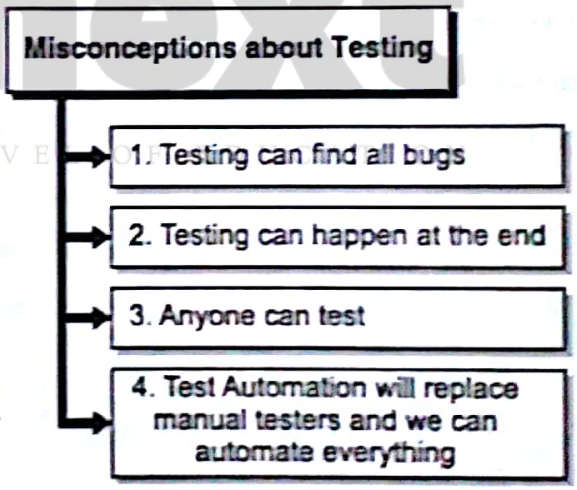

Fig. 2.7.4

- "Exhaustive testing is impossible" this is the one of the basic principles of testing. Testing all the input combinations and preconditions is not feasible. The bugs will remain in the software even after the testing is done.
- Testing can show the presence of defects but cannot prove that there are no defects. Even if testing cannot find bugs it's not the proof that the software is bug-free.
- As a tester one can focus his tests and uncover the defects based on project risks, priorities and project constraints (like time, budget).

2. Testing can happen at the end

- Testing happens all the time throughout the life cycle of a software. Just because buttons and links aren't being clicked doesn't mean that testing isn't happening.
- Before coding is completed, testers can work with team members and communicate the things they'll be looking for.
- They can go over requirements, ask questions about use cases and user stories, and setup prerequisites. All of this is testing.

➜ **3. Anyone can test**

- As any other skill, testing needs experience, practice, and learning. Being a real good tester requires a specific mindset.
- Software testing is a challenging job it demands traits like understanding testing methodologies, creativity, problem solving, planning, eye to detail, patience, communication and lot more. It is never easy to come up with scenarios to break the system.
- This myth is still deep-rooted in the minds of people. You often see the developers who failed to deliver or the freshers with no formal training being pushed to testing by the management assuming that anyone can test.
- Software testing is a passion and everyone cannot excel in it. It's our responsibility as a tester to convince the management that they are wrong.

➜ **4. Test Automation will replace manual testers and we can automate everything**

- Automation tools are available to assist manual testers to automate their repeated tests and not to replace them. Everything cannot be automated.
- Automation is not feasible when the requirements are changing and it's never wise to automate the feature that needs human intelligence and intuition.
- An automation tool cannot tell if the look and feel is good, if the app is usable etc.

Syllabus Topic : Principles of Software Testing

2.7.4 Principles of Software Testing

Q. 2.7.5 What are principles of testing? (Ref. Sec. 2.7.4) **(5 Marks)**

➜ **1. Testing shows presence of defects**

Testing shows that there are defects present but it does not imply that there are no defects.

➜ **2. Exhaustive testing is not possible**

Testing everything i.e. all the combinations of inputs is not possible. Testing the product should be accomplished considering the risk factor and priorities

➜ **3. Early Testing**

Early testing helps identify issues prior to the development stage, which eases error correction and helps reduce cost

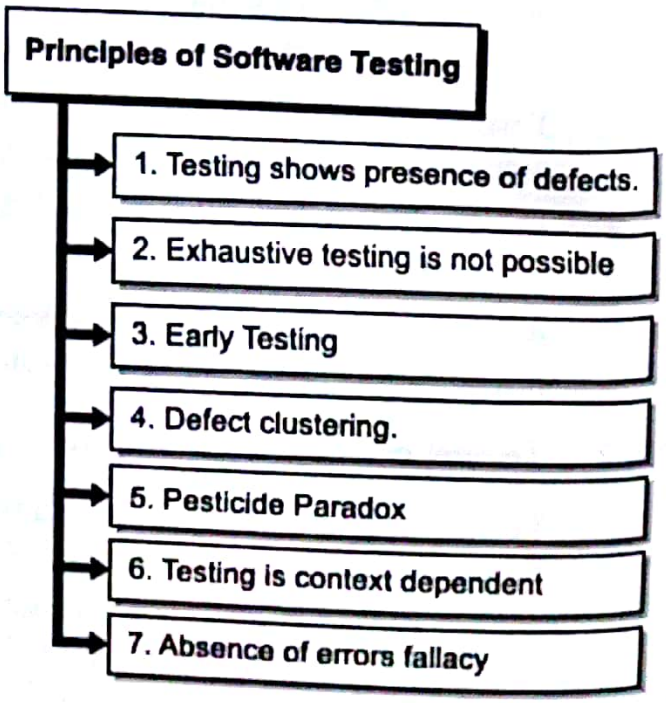

Fig. 2.7.5

4. Defect clustering

Normally a defect is clustered around a set of modules or functionalities. Once they are identified, testing can be focused on the defective areas, and yet continue to find defects in other modules simultaneously.

5. Pesticide Paradox

If the same tests are repeated over and over again then eventually the test will no longer find and defects. To overcome this the tests should be regularly revised.

6. Testing is context dependent

Testing is done differently depending on the type of software. For e.g. software which deals with money should be tested keeping security in mind.

7. Absence of errors fallacy

Just identifying and fixing issues does not really help in setting user expectations. Even if testing is performed to showcase the software's reliability, it is better to assume that none of the software products are bug-free.

Syllabus Topic : Salient Features of Good Testing

2.7.5 Salient Features of Good Testing

Q. 2.7.6 How do we know testing is good testing? (Ref. Sec. 2.7.5) (5 Marks)

1. Be Skeptical

- Don't believe that the build given by the developers is a bug-free or quality outcome. Question everything. Accept the build only if you test and find it defect free.
- Don't believe anyone whatever is the designation they hold, just apply your knowledge and try to find the errors.
- You need to follow this until the last phase of the testing cycle.

2. Don't Compromise on Quality

- Don't compromise after certain testing stages. There is no limit for testing until you produce a quality product.

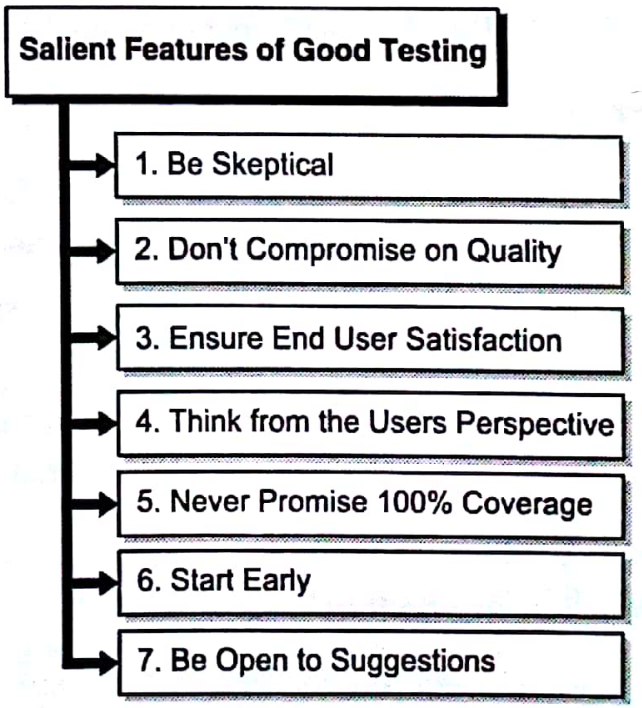

Fig. 2.7.6

- Quality is the word made by software testers to achieve more effective testing. Compromising at any level leads to a defective product, so don't do that at any point.

➜ 3. Ensure End User Satisfaction

- Always think what can make an end user happy. How they can use the product with ease.
- Don't stop by testing the standard requirements alone. The end user can be happy only when you provide an error-free product.

➜ 4. Think from the Users Perspective

- Every product is developed for the customers. Customers may or may not be technical persons. If you don't consider the scenarios from their perspective you will miss many important bugs. So put yourself in their shoes.
- Know your end users first their age, education even the location can matter most while using the product.
- Make sure to prepare your test scenarios and test the data accordingly. After all, the project is said to be successful only if the end user is able to use the application successfully.

➜ 5. Never Promise 100% Coverage

- Saying 100% coverage on paper is easy but practically it is impossible. So never promise to anyone including your clients about total test coverage.
- In business there is a philosophy - "Under promise and over deliver." So don't the goal for 100% coverage but focus on the quality of your tests.

➜ 6. Start Early

- Don't wait until you get your first build for testing. Start analyzing the requirements, preparing test cases, test plan and test strategy documents in the early design phase.
- Starting early to test helps to visualize the complete project scope and hence planning can be done accordingly.
- Most of the defects can be detected in early design and analysis phase saving huge time and money. Early requirement analysis will also help you to question the design decisions.

➜ 7. Be Open to Suggestions

- Listen to everyone even though you are an authority on the project having in-depth project knowledge.

- There is always scope for improvements and getting suggestions from the fellow software testers is a good idea.
- Everyone's feedback to improve the quality of the project would certainly help you to release a bug-free software.

Syllabus Topic : Test Policy, Test Strategy or Test Approach

2.8 Test Policy, Test Strategy or Test Approach

Q. 2.8.1 Differentiate between test policy and test strategy. (Ref. Sec. 2.8) (5 Marks)

Test Policy

- Test policy is a document described at the organization level and gives the organizational insight for the test activities.
- It is determined by the senior management of the organization and defines the test principles that the organization has adopt. The test policy is very high-level document and at the top of the test documentation structure.
- Organizations may prefer to publish their test policy in a sentence, as well as a separate document. Also they may use this policy in both development and maintenance projects.
- The test policy shall describe the followings :
  - Clear answer to the question of "What does testing means for the organization"
  - Test objectives that the organization have
  - The definition of the testing process used by the organization to increase the quality of the software developed
  - How the organization will measure the effectiveness and efficiency of the test while achieving goals
  - How the organization will improve its test processes

Test Strategy

- Test strategy document is prepared at the program level and includes general test strategy, management principles, processes and approaches for the tests to be performed for software in detail.
- The test strategy document is also a high level document and is usually written by the test manager and the project manager in the top level organization. It is generally prepared in large scale projects and does not need much updating.
- In small scale projects, test strategies and test approach may be included in the test plan, and also the test strategy document may not be written separately.

- Test approach and test activities included in the test strategy document must be consistent with the test policies of the organization.
- The test strategy document may applicable for a program / system that contains multiple projects and describes :
  - Objective / scope of testing
  - In-scope / out of scope items for testing
  - Test levels (Unit, System, Integration, System Integration)
  - Test types (Functional / Non-Functional)
  - Entry / Exit / Stop / Resumption Criteria for testing (for different levels / phases)
  - Risks to be addressed
  - Test environment
  - Test case design methodology
  - Test methodology (Top-down / bottom-up / risk based)
  - Test control and reporting
  - Test automation approach
  - Test tools to be used
  - Defect management approach
  - Defect classification
  - Retesting & regression approach

Syllabus Topic : Test Planning, Testing Process and Number of Defects Found In Testing

2.8.1 Test Planning, Testing Process and Number of Defects Found in Testing

Q. 2.8.2 What is testing planning and test process? (Ref. Sec. 2.8.1) (5 Marks)

Test Planning

- Test plan is a document prepared at the project level. In general it defines work products to be tested, how they will be tested (test cases) and test type distribution among testers.
- Test plan also includes test environment and test tools to be used during the project, the persons responsible for the tests and their responsibilities, test levels and test types, test schedule planned for test runs, and the principles of management and reporting of errors / bugs.
- Test plan is usually prepared by the test manager or test leader in the test organization and shared with the entire team in the project. It is a living document throughout the project and should be kept under revision control as it's updated.

- The information in the test plan document must be consistent with the organization's test policy and test strategy.
- The test plan may describe the followings :
  - All test strategies specific to the project
  - Test estimations and test schedule
  - Test organization / roles / responsibilities
  - Test deliverables
  - Test reporting principles

Testing process

- Testing is a process rather than a single activity. This process starts from test planning then designing test cases, preparing for execution and evaluating status till the test closure.
- Test processes are a vital part of Software Development Life Cycle (SDLC) and consist of various activities, which are carried out to improve the quality of the software product.
- From planning to execution, each stage of the process is systematically planned and requires discipline to act upon them. These steps and stages are extremely important, as they have their own entry criteria and deliverable, which are combined and evaluated to get expected results and outcomes.
- Therefore, we can say that the quality and effectiveness of the software testing is primarily determined by the quality of the test processes used by the software testers.
- Moreover, by following a fundamental test process, testers can simplify their work and keep a track of every major and minor activity.

Number of Defects Found In Testing

- Number of defects found is high in initial phases of testing. The defect count decreases with each testing cycle.
- The cost of finding and fixing the defects rises considerably across the life cycle.

Syllabus Topic : Test Team Efficiency

2.8.2 Test Team Efficiency

Q. 2.8.3 Write a short on Test Team efficiency. (Ref. Sec. 2.8.2) (5 Marks)

- There is no doubt that tracking team member performance is important. Talented team members will want feedback to help them grow and improve.
- It makes sense to keep track of what's going on at your business. However, quantifying and measuring performance is easier said than done.

- It is important to measure efficiency of team members. They should be able to complete their work on time.
- They should have a good handle on the limitations provided by the time and resources available and should be able to prioritize to get things done as efficiently as possible.
- It's nice when those you work with ask what's needed and where they can help. It's even nicer when they see a need and take steps to meet it on their own.
- An employee that takes initiative is definitely a sign of team satisfaction and engagement.
- Looking at team members who take initiative is also important for growing businesses and for rapidly changing workplaces that require people who can adapt and be proactive.
- Initiative-taking is definitely a difficult metric to measure, but a good place to start would be by keeping track of the times you see a team member taking initiative, either with a nifty app or with good old-fashioned pen and paper.
- The quality of work your team members put out is perhaps the most important but it is also the most difficult to define.

Syllabus Topic : Mutation Testing

2.8.3 Mutation Testing

Q. 2.8.4 What is mutation testing? (Ref. Sec. 2.8.3) (5 Marks)

- Mutation Testing is a type of software testing where we mutate (change) certain statements in the source code and check if the test cases are able to find the errors.
- The mutations introduced to source code are designed to imitate common programming errors.
- A good unit test suite typically detects the program mutations and fails automatically. It is a type of White Box Testing which is mainly used for Unit Testing.
- The changes in mutant program are kept extremely small, so it does not affect the overall objective of the program.
- Mutation testing has the following advantages :
  - Program code fault identification
  - Effective test case development
  - Detection of loopholes in test data
  - Improved software program quality
  - Elimination of code ambiguity

Example

Read Marks
If Marks>60
Print "First Class"
End if
Change ">" with "<"

Syllabus Topic : Challenges In Testing

2.8.4 Challenges In Testing

Q. 2.8.5 What are the different challenges in testing? (Ref. Sec. 2.8.4) (5 Marks)

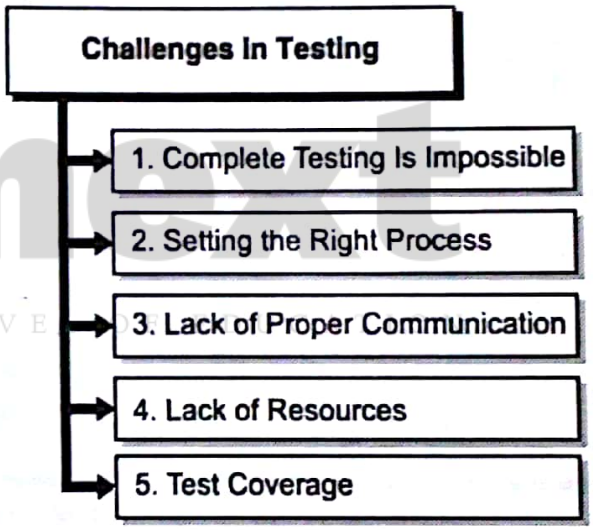

Fig. 2.8.1

➜ 1. Complete Testing Is Impossible

- For any application, there are many different test combinations and resulting scenarios.
- Trying to test all of them is an unrealistic goal. Having a clear focus on your target users and business goals will help you in prioritizing the app features that need your attention.

➜ 2. Setting the Right Process

- The company process for testing should be aligned in such way that the process is efficient in capturing the bugs in time.
- The process should be clearly defined to avoid misunderstandings. Setting up a uniform process to get everyone on the same page becomes harder for teams that are located in different offices in different time zones.

➜ 3. Lack of Proper Communication

- Right from the start of the project, it is important to make the business requirement and testing goals clear.
- Any change in app features or requirement must be updated to the QA team in time. This further helps in fixing defects that are detected.
- A good communication channel between various teams like development & QA will help sync the testing efforts with the quality goals set by the business.

➜ **4. Lack of Resources**

- Skilled testers with good knowledge of the business domain can help you create more effective test scenarios and scripts.
- Apart from getting domain specific testing experts on board, you also need to equip them with the right testing tools.
- The right tools can make it easier for QA teams to execute the project and also delivering results faster.

➜ **5. Test Coverage**

- In today's competitive world, enterprises need an effective testing strategy to ensure that their app works well, everywhere all the time.
- Covering different geographies and configurations for testing applications is challenging for teams that rely solely on a traditional QA approach. Lab-based testing with limited resources can never recreate a real world environment.
- Getting an accurate insight in these cases thus becomes more difficult. Though a 100% coverage might not be possible, you can maximize your QA efforts by relying on crowd-sourced testing platforms.
- Other challenges faced in software testing include test automation, regression testing, and app complexity.
- While a 100% bug-free software product may be impossible, planning ahead for these critical software testing challenges can help you succeed in your QA efforts to make your application better than yesterday.

Syllabus Topic : Test Team Approach

2.8.5 Test Team Approach

Q. 2.8.6 Who are members of a test team? (Ref. Sec. 2.8.5) (5 Marks)

- Proactive communication and a well-equipped test team are able to deliver suitable solutions. Effective communication with clients holds the key to successful project completion.
- **Testing Team :** The role and responsibility of each member is clearly defined at every stage of the testing process.
- **Program Manager : He is responsible for** planning and execution of the project and to ensure the success of a project by minimizing risk throughout the lifetime of a project. He is also responsible for writing the product specification, managing the schedule and making critical decisions.

- **QA Lead :** He is mentors team members to improve QA effectiveness. He implements industry's best practices.
- **Test Analysts :** Responsible for executing tests, gathering and managing test data and evaluate the outcome of each test.
- **Test Engineer :** Responsible for writing and executing test cases and reporting test defects. They are also responsible for determining the best way a test can be performed in order to achieve 100% test coverage.

Syllabus Topic : Process Problems Faced by Testing

2.9 Process Problems Faced by Testing

Q. 2.9.1 What are the process related problems that affect testing? (Ref. Sec. 2.9) **(5 Marks)**

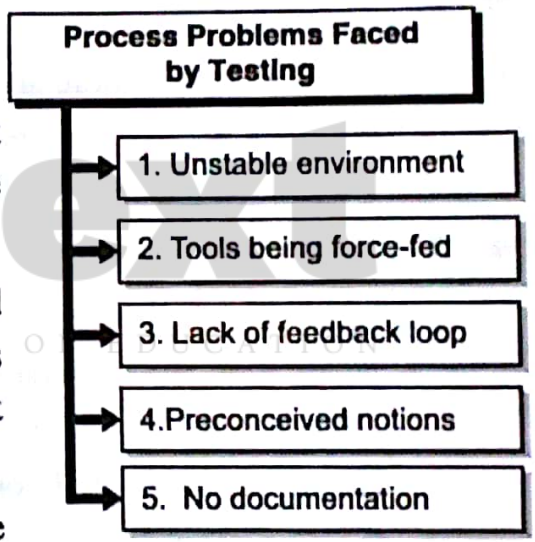

Fig. 2.9.1

1. Unstable environment

- Usually, QA teams suffer from inferior environment set up that we have to really be ready to make the most of what we have.
- **For example :** The server that gets overloaded and needs a restart few times during testing, the logs that need clearing often to make sure that there isn't an overflow, etc.
- Bring these problems to the forefront and make sure you get environment support during testing.
- For commonly happening cases, get the access to the servers with the steps to do some simple maintenance, such as restart, clearing queues, etc.

2. Tools being force-fed

- Sometimes we know that a tool is not fit for the job. We have no choice but to continue using it because the clients/teams already have licenses and would not want to go for a new one until the current license runs out.
- I had to test a Mainframes application on HP QTP without the Terminal Emulator add-in. In this case, I had the tool but not the correct configuration. There was little I could do about it, so I had to switch between Normal and low-level recording modes as a workaround.
- It is not fun, but you learn alternatives. Or at least, you will reach to a definite conclusion as to whether the alternatives actually work or not.

➜ **3. Lack of feedback loop**

- Sometimes you go days at an end working on and obsessing over a deliverable only to find out at that it wasn't supposed to this way.
- Or you work from a remote location with your team located elsewhere that you feel isolated and have no one to bounce your ideas off of.
- Or you receive feedback that is not exactly helpful. Let's say you created a process document and they said it was good.
- You don't see the process document published or put to use and you are left wondering what happened to it. So, the feedback 'good' did not do any good here and is almost a non-feedback.
- Seek honest feedback and create a community to discuss your ideas. Not often the easiest to do, but without the positive reinforcement that this step provides you are left demotivated.

➜ **4. Preconceived notions**

- Well, we know there are many prejudices in the workplace around gender, nationality etc. I am not going to go into specifics here but unless we start looking at the world as a global village and everyone equal, the world, and the workplace both become toxic.

➜ **5. No documentation**

- Many teams still believe in verbal communication and keep little reference material about how the software became what it is today. Rapid development cycles only made this more intense.
- However, this is really one of those cases of challenges becoming opportunities.
- Engage in conversations with your development, business analysis or technical teams. Research the application; set up references looking at similar applications and their standards. Understand the end-user perspective. Get adventurous with exploratory testing.

Syllabus Topic : Cost Aspect of Testing

2.9.1 Cost Aspect of Testing

Q. 2.9.2 Explain the cost aspect of testing. (Ref. Sec. 2.9.1) (5 Marks)

- Measuring the cost of testing is an important step toward justifying any software testing initiative. All software testing expenditures are justified by comparing the benefits accrued with the cost.
- The benefits could be in terms of quality measurement and failure prevention or earlier detection etc.

- The cost of the test phase is not the cost of testing for the project. Some testing work is carried out in other phases (design testing, unit testing, etc.) and a lot of work is performed during the systems test that is not testing (for example, documentation, debugging, deficiency analysis and removal, conversion, and training).
- Careful analysis usually reveals that actual software testing costs usually lies between 15 to 25 percent of the total project cost. Lot many testing managers feel that the software testing costs are much higher to the tune of 50 percent of the cost of a project.
- They mistakenly treat all project expenditure after programming as a testing cost. While it is true, for many projects, that a good way to estimate final total cost is to take actual expense through the programming phase and double it, it is not true that most of the cost after programming is the result of software testing.
- Determining software testing costs is a crucial first step to planning any improvement initiative and justifying the investment. An estimate of the amount being spent to test and measure quality, as well as the cost of rework or corrections, is fundamental.
- **Cost of Quality** is a cost associated with software testing. The "cost of quality" isn't the price of creating a quality product or service. It's the cost of NOT creating a quality product or service. Every time work is redone, the cost of quality increases. Obvious examples include :
  - The reworking of a manufactured item.
  - Making corrections after a product is installed
  - Replacing a faulty product
- The Cost of Quality includes prevention, appraisal and correction or repair costs.
- Cost of Control (Also known as Cost of Conformance)

Prevention Cost

- The cost arises from efforts to prevent defects.
- Example : Quality Assurance costs

Appraisal Cost

- The cost arises from efforts to detect defects.
- Example : Quality Control costs

- Cost of Failure of Control (Also known as Cost of Non-Conformance)

Internal Failure Cost

- The cost arises from defects identified internally and efforts to correct them.
- Example : Cost of Rework (Fixing of internal defects and re-testing)

External Failure Cost

- The cost arises from defects identified by the client or end-users and efforts to correct them.
- Example : Cost of Rework (Fixing of external defects and re-testing) and any other costs due to external defects (Product service/liability/recall, etc)

Formula / Calculation

$$\text{Cost of Quality (COQ)} = \text{Cost of Control} + \text{Cost of Failure of Control}$$

Where,

Cost of Control = Prevention Cost + Appraisal Cost

and

Cost of Failure of Control = Internal Failure Cost + External Failure Cost

Syllabus Topic : Establishing Testing Policy, Methods, Structured Approach to Testing

2.9.2 Establishing Testing Policy, Methods, Structured Approach to Testing

Q. 2.9.3 What are the contents of a test policy? (Ref. Sec. 2.9.2) (5 Marks)

- A Test Policy is a high level document and is at the top of the hierarchy of the Test Documentation structure.
- The purpose of the Test Policy document is to represent the testing philosophy of the company as a whole and to provide a direction which the testing department should adhere to and follow. It should apply to both new projects and maintenance work.
- Setting an appropriate test policy by senior managers provides a robust framework within which testing practitioners can then operate. This will help to ensure the maximisation of the strategic value inherent in every project.

2.9.3 Contents of a Test Policy Document

Definition of Testing

- Organisations needs to be clear why they are testing. This will influence the remainder of the policy document and also the detailed testing techniques that are selected by test managers at the programme and project level.
- From the understanding of why testing is required it is possible to specify what the purpose of testing is within the organisation. Without this fundamental linkage the test effort is destined to fail.
- **Example :** "ensuring the software fulfils its requirements"

Description of the test process

- It is vital to establish a solid view towards the test process. We should address questions like, which phases and subtasks will the test process include.
- Which roles will be involved and the document structure associated with each tasks, as well as what test levels need to be considered.
- **Example :** "all test plans are written in accordance with company policy"

Test Evaluation

- How are we going to evaluate the results of testing, what measures will we use to ensure test effectiveness in the project?
- **Example :** "effect on business of finding a fault after its release"

Quality Level to be achieved

- Which quality criteria are going to be tested and which quality level is the system required achieving prior to its release with regards to these criteria?
- **Example :** "no outstanding high severity faults prior to products release"

Approach to Test Process Improvement

- How often and when are we going to assess the usefulness of the current processes in place and what elements need improving and techniques that shall be used to improve the processes.
- **Example :** "project review meetings to be held after project completion"

Syllabus Topic : Categories of Defect

2.10 Categories of Defect

Q. 2.10.1 What is a defect? What are the categories of defects? (Ref. Sec. 2.10) **(5 Marks)**

- A Software DEFECT / BUG is a condition in a software product which does not meet a software requirement or end-user expectation. In other words, a defect is an error in coding or logic that causes a program to malfunction or to produce incorrect/unexpected results.
- When actual result deviates from the expected result while testing a software application or product then it results into a defect. Hence, any deviation from the specification mentioned in the product functional specification document is a defect.

- Defects are classified from the QA team perspective as **Priority** and from the development perspective as **Severity.** These are two major classifications that play an important role in the timeframe and the amount of work that goes in to fix defects.

2.10.1 Defect severity and Priority

- **Priority** of a defect indicates the urgency with which it would need to be fixed. The priority status is usually set by the QA team while raising the defect.
- Defect priority is categorized into three classes :

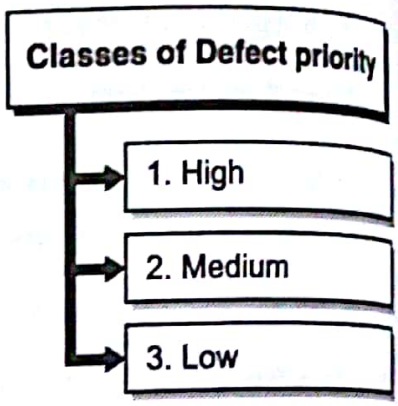

Fig. 2.10.1

➜ **1. High**

This defect requires immediate solution as it affects the testing process. They should be resolved within 24 hours. These defects are generally when the entire functionality fails and no testing can be done.

➜ **2. Medium**

These defects are fixed after all the critical defects are fixed. When a defect causes a program to not function as expected but it does not lead to entire blockage of functionality then the defect can be categorized as medium priority. These defects or issues should be resolved before the release is made.

➜ **3. Low**

An issue which has no impact on the customer business comes under low priority.

- **Severity** is defined as the degree of impact a defect has on the development or operation of a component application being tested.
- Defect severity is categorized into four classes :

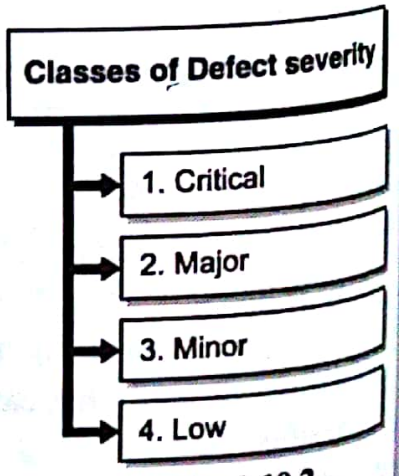

Fig. 2.10.2

➜ **1. Critical**

- This defect affects critical functionality of the software. It indicates complete shut-down of the process, nothing can proceed further
- E.g. : Restarting of the phone, complete failure of a feature etc

➜ **2. Major**

- Major defects are of high severity which affects functionality of the software. There is a workaround but is not easy and obvious for users to understand.

- **E.g.:** functionality is not working from a module but the same functionality can be accessed from other module after performing some number of additional steps from that module

3. Minor

- This defect affects minor functionality which is not very critical. It cause some undesirable behavior, but the system is still functional
- **E.g.:** functionality is not working from a module but the same functionality can be accessed from other module easily i.e. Minor defects have easy workaround

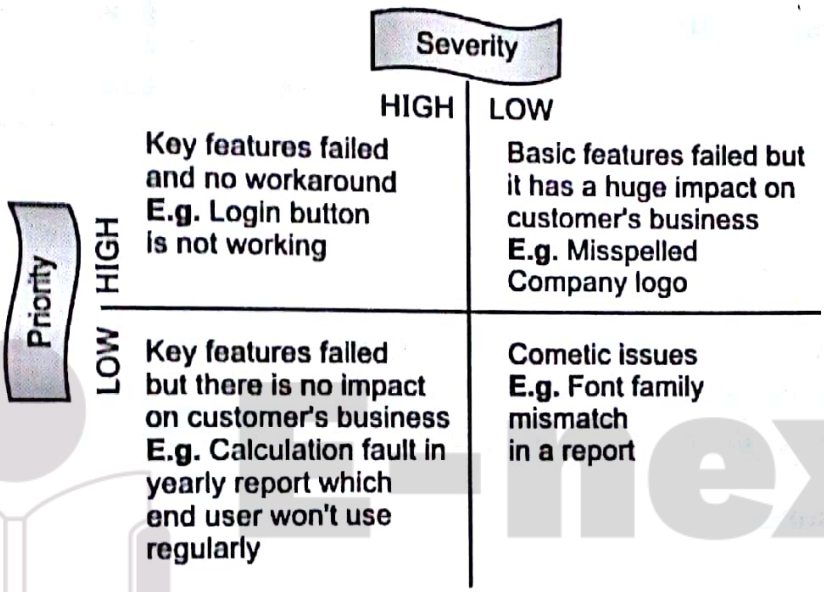

Fig. 2.10.3

4. Low

- These defects do not affect functionality. These are generally cosmetic defects and does not impact productivity or efficiency of the software
- E.g. : Spelling mistake, grammatical error etc.

Syllabus Topic : Defect, Error, or Mistake in Software

2.10.2 Defect, Error or Mistake in Software

Q. 2.10.2 Define Defect, Error and mistake (Ref. Sec. 2.10.2) (5 Marks)

2.10.2.1 Error (Mistake)

- Error or mistake is a human action that can produces an incorrect result.
- Errors can be present in the software due to the following reasons.

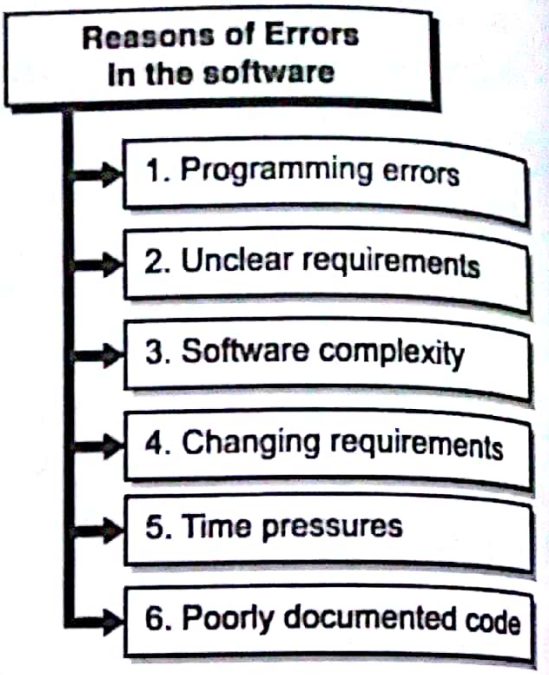

Fig. 2.10.4

➜ **1. Programming errors**

Programmers can make mistakes while developing the source code.

➜ **2. Unclear requirements**

The user is not clear about the desired requirements or the developers are unable to understand the user requirements in a clear and concise manner.

➜ **3. Software complexity**

The greater the complexity of the software, the more the scope of committing an error (especially by an inexperienced developer).

➜ **4. Changing requirements**

The users usually keep on changing their requirements, and it becomes difficult to handle such changes in the later stage of development process. Therefore, there are chances of making mistakes while incorporating these changes in the software.

➜ **5. Time pressures**

Maintaining schedule of software projects is difficult. When deadlines are not met, the attempt to speed up the work causes errors.

➜ **6. Poorly documented code**

If the code is not well documented or well written, then maintaining and modifying it becomes difficult. This causes errors to occur.

- Testing is the process of identifying defects, where a defect is any variance between actual and expected results. "A mistake in coding is called Error, error found by tester is called Defect, defect accepted by development team then it is called Bug, build does not meet the requirements then it Is Failure."

2.10.2.2 Defect

- **Defect** can be simply defined as a variance between expected and actual. Defect is an error found AFTER the application goes into production.
- Defect can be divided as follows :

1. Wrong

When requirements are implemented not in the right way. This defect is a variance from the given specification. It is wrong!

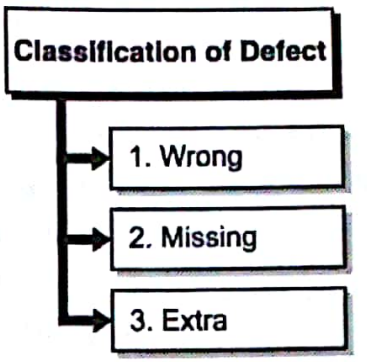

Fig. 2.10.5

2. Missing

A requirement of the customer which was not fulfilled. This is a variance from the specifications, an indication that a specification was not implemented, or a requirement of the customer was not noted correctly.

3. Extra

A requirement incorporated into the product that was not given by the end customer. This is always a variance from the specification, but may be an attribute desired by the user of the product. However, it is considered a defect because it's a variance from the existing requirements.

Syllabus Topic : Developing Test Strategy

2.11 Developing Test Strategy

Q. 2.11.1 List and explain components of test strategy document. (Ref. Sec. 2.11) (5 Marks)

- A Test Strategy document is a high level document and normally developed by project manager. This document defines "Software Testing Approach" to achieve testing objectives.
- The Test Strategy is normally derived from the Business Requirement Specification document.
- The Test Strategy document is a static document meaning that it is not updated too often. It sets the standards for testing processes and activities and other documents such as the Test Plan draws its contents from those standards set in the Test Strategy Document.
- Some companies include the "Test Approach" or "Strategy" inside the Test Plan, which is fine and it is usually the case for small projects.
- However, for larger projects, there is one Test Strategy document and different number of Test Plans for each phase or level of testing.

2.11.1 Components of the Test Strategy Document

- Scope and Objectives : In this section, we will mention the scope of testing activities (what to test and why to test).
- Roles and responsibilities : This section describes the roles and responsibilities of Project Manager, Project Lead, individual testers.
- Communication and status reporting : This section describes the tools used for communication and reporting of the testing activities status
- Test deliverables : This section lists out the deliverables that need to produce before, during and at the end of testing.
- Test automation and tools : This section describes the tools that should be used for automating testing.
- Testing measurements and metrices : This section describes the metrics that should be used in the project to analyze the project status.
- Risks and mitigation: Identify all the testing risks that will affect the testing process and specify a plan to mitigate the risk.
- Defect reporting and tracking: This section outlines how defects and issues will be tracked using a reporting tool.
- Change and configuration management: List the name of the tool to be used for configuration management Training plan

Syllabus Topic : Developing Testing Methodologies (Test Plan), Testing Process

2.11.2 Developing Testing Methodologies (Test Plan), Testing Process

Q. 2.11.2 What is a test plan? (Ref. Sec. 2.11.2) **(5 Marks)**

- The Test Plan document on the other hand, is derived from the Product Description, Software Requirement Specification SRS, or Use Case Documents.
- The Test Plan document is usually prepared by the Test Lead or Test Manager and the focus of the document is to describe what to test, how to test, when to test and who will do what test.
- It is not uncommon to have one Master Test Plan which is a common document for the test phases and each test phase will have their Test Plan documents.
- There is much debate, as to whether the Test Plan document should also be a static document like the Test Strategy document mentioned above or should it be updated every often to reflect changes according to the direction of the project and activities.
- My own personal view is that when a testing phase starts and the Test Manager is "controlling" the activities, the test plan should be updated to reflect any deviation from the original plan. After all, Planning and Control are continuous activities in the formal test process.

2.11.3 Components of the Test Plan Document

- Test Plan id – Introduction
- Test items – Features to be tested
- Features not to be tested – Test techniques
- Testing tasks – Suspension criteria
- Features pass or fail criteria – Test environment (Entry criteria, Exit criteria)
- Test deliverables – Staff and training needs
- Responsibilities – Schedule

This is a standard approach to prepare test plan and test strategy documents, but things can vary company-to-company.

Syllabus Topic : Test Methodologies/Approaches

2.11.4 Test Methodologies/Approaches

Q. 2.11.3 Define Test Methodologies. (Ref. Sec. 2.11.4) (5 Marks)

- Software testing methodologies are the different approaches and ways of ensuring that a software application in particular is fully tested.
- Software testing methodologies encompass everything from unit testing individual modules, integration testing an entire system to specialized forms of testing such as security and performance.
- As software applications get ever more complex and intertwined and with the large number of different platforms and devices that need to get tested, it is more important than ever to have a robust testing methodology for making sure that software products/systems being developed have been fully tested to make sure they meet their specified requirements and can successfully operate in all the anticipated environments with the required usability and security.

Syllabus Topic : Attitude towards Testing (Common People Issues)/ People Challenges in Software Testing

2.11.5 Attitude towards Testing (Common People Issues)/ People Challenges in Software Testing

Q. 2.11.4 Why is it important to increase awareness of management towards testing? (Ref. Sec. 2.11.5) (5 Marks)

Cross-Cultural differences

- Although corporate culture is fairly non-ethnic, where we are from impacts our behavior and understanding.
- **Example :** "Hi, How are you?" is a common greeting in the US. It does not necessarily mean they want to know exactly what you are feeling at the moment. However, when I was new in the US, I used to think, "I was just in a meeting with this person a moment ago. What would change in such a little time?" :) Good for me, I learnt fast.
- Also, in some cultures, talking less indicates quiet contemplation while in others it simply means, it is boring or you have nothing to say.
- When you try to understand these small nuances, you understand people better and can function in a better way.

Lack of proper communication

- Right from the start of the project, it is important to make the business requirement and testing goals clear. Any change in app features or requirement must be updated to the QA team in time.
- This further helps in fixing defects that are detected. A good communication channel between various teams like development & QA will help sync the testing efforts with the quality goals set by the business.

Syllabus Topic : Raising Management Awareness for Testing

2.12 Raising Management Awareness for Testing

Q. 2.12.1 Why is important to raise awareness about testing in management? (Ref. Sec. 2.12) **(5 Marks)**

- Management should be made aware of the importance of testing a product because the funds required will be provided by them.
- Without adequate funding for testing, software quality suffers. Benefits of software testing are enormous, and they have a significant role in entire business.

Quality

It helps increase the quality of a product. The quality of the product is important for the customer. For quality, customers will surely pay more money. Even more important is that with selling high-quality products, you build a strong reputation and brand image, things that are important in the long-term.

Satisfied Customer

- The centre of every business is a happy customer. When selling something, you are aware that everything does not end there. The client can ask for a refund if the product does not suit him.
- If the product is not reliable, you need to invest more money in fixing or replacing it and then you realise that it would be best that quality was controlled from the start. The only conclusion is that it pays off to produce a higher quality product from the start.
- Only when you do the software testing right you can guarantee that your product is valuable and reliable.

Bringing Profit

- Speaking of profit, the testing phase is a part of it. A good product needs less promotion because people will recommend it one to another. Word-of-mouth recommendation is the best and most valuable commercial you can get, and it's the best advertising tool.
- Offering a rigorously tested and quality checked product means having respect for your clients. That will help in retaining old customers and gaining new ones.
- Not only will the testing phase bring profit, but it will cut down existing expenses. In the long run, it will save money because you are selling software that does not need constant fixing.
- It is often seen that compromising on quality ends up with having to spend more money than planned. Secondly, the benefit of software testing is that it allows removing errors and bugs before the products get shipped to the market.
- That will prevent unsatisfied clients and unnecessary expenses that bring customer support. Third, the costs of the service can be reduced by using automated software testing solutions.

User Experience

- User experience is a significant factor when putting some products on the market. Software needs to be simple, understandable, and easy to use. Only testers can assure that.
- Their experience will make sure that the software is designed in a way that is logical and intuitive. If you want great user experience, software needs to be free from bugs and errors, which can be a source of dissatisfaction for users.
- Choosing a good software testing service with a professional team will guarantee the quality of a product and good user experience.

Business Optimisation

- The biggest benefit is that software testing leads to business optimisation.
- Business optimisation means more satisfied clients, customer retention, fewer costs of fixing a product, fewer costs of a customer service, better quality, and more reliable products, improved reputation and brand image.

Syllabus Topic : Skills Required by Tester

2.12.1 Skills Required by Tester

Q. 2.12.2 What are the skills required by testers? (Ref. Sec. 2.12.1) (5 Marks)

1. Understanding of Business Needs

- A great software tester must understand the ultimate goal of the system being built from the business end. The tester, only when knows and understands the bigger picture, can help analyse the system's strength and weaknesses and go beyond the mere testing duty.
- A great software tester must be able to take the ultimate call on a system's ability to go live. He must provide the business with competitively advantageous strategic inputs from testing perspective.

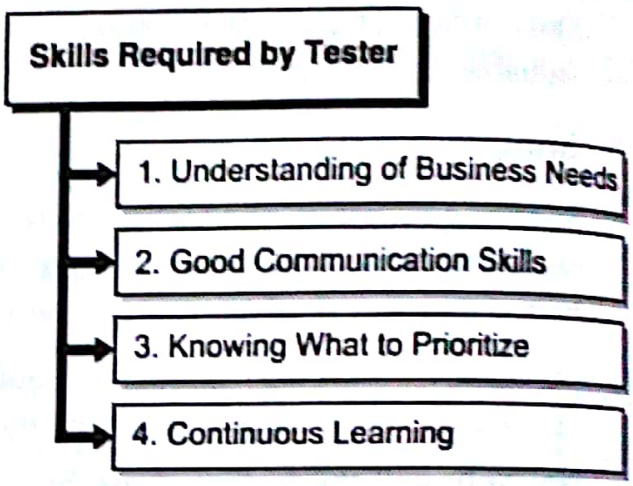

Fig. 2.12.1

2. Good Communication Skills

- A great software tester must definitely possess strong verbal and written communication skills. The tester's ability to collaborate with other programmers, test managers and customers must be impeccable.
- Being a good team player, skills to get across their point to the other party without difficulty and knowing how to put across a defect all are important personality traits of a great software tester.

3. Knowing What to Prioritize

- A great software tester must be able to prioritize the test cases and features that are to be tested. The testing environment, requirements and timelines vary frequently.
- Depending upon these factors, a software tester must be able to interpret, organize and prioritize his activities such that the testing goals are achieved without compromising the quality.

4. Continuous Learning

- As with any IT field, testing is an area that keeps evolving every day! New technologies, tools and concepts are introduced to benefit the testers and end users.
- A great software tester must keep himself updated with the latest technologies and happenings in the field of testing.
- A consistent learning schedule must be a part of every great software tester.

Syllabus Topic : Testing Throughout the Software Life Cycle

2.12.2 Testing Throughout the Software Life Cycle

Q. 2.12.3 Why should testing be carried throughout the software life cycle? (Ref. Sec. 2.12.2) **(5 Marks)**

- When the testing team executes the test cases, they come across a situation where the actual test result is different from the expected result. This variation is termed as a **Defect.**
- Defects must be identified as early as possible in the life cycle of a project so that the cost of fixing the bug reduces. Without a doubt, the best approach to defects is to eliminate them altogether.
- While that would be ideal, it is virtually impossible to develop a 100% bug free system however we can identify and implement ways to prevent defects.
- Defect Prevention should be considered as a critical activity. It begins with studying the already found defects and finding the risk areas.
- Getting the critical risks defined allows people to know the types of defects that are most likely to occur and the ones that can have the greatest system impact. Strategies can then be implemented to prevent them.
- Based on all the defects found, testing team should go back to check the processes implemented throughout the life cycle to understand what originated the defects.
- The processes should then be modified and updated. The valuable insight gained should be used to strengthen the review process.

Syllabus Topic : Software Development Models

2.12.3 Software Development Models

Q. 2.12.4 Explain V-model. (Ref. Sec. 2.12.3) **(5 Marks)**

2.12.3.1 V- Model (V&V Model)

- V-model is also known as **Validation & Verification** model. V-model was developed to address some of the problems experienced in the traditional software development life cycle.
- Defects were found too late as the testing was not involved until the end of the project. V-model provides the guidelines that testing should begin as early as possible in the life cycle; this is also one of the important principles of testing.
- There are many activities related to testing that can be performed before the completion of coding phase.

- These activities can be carried out in parallel with the other development activities. By starting testing early we will be able to find defects and resolve them early as well.

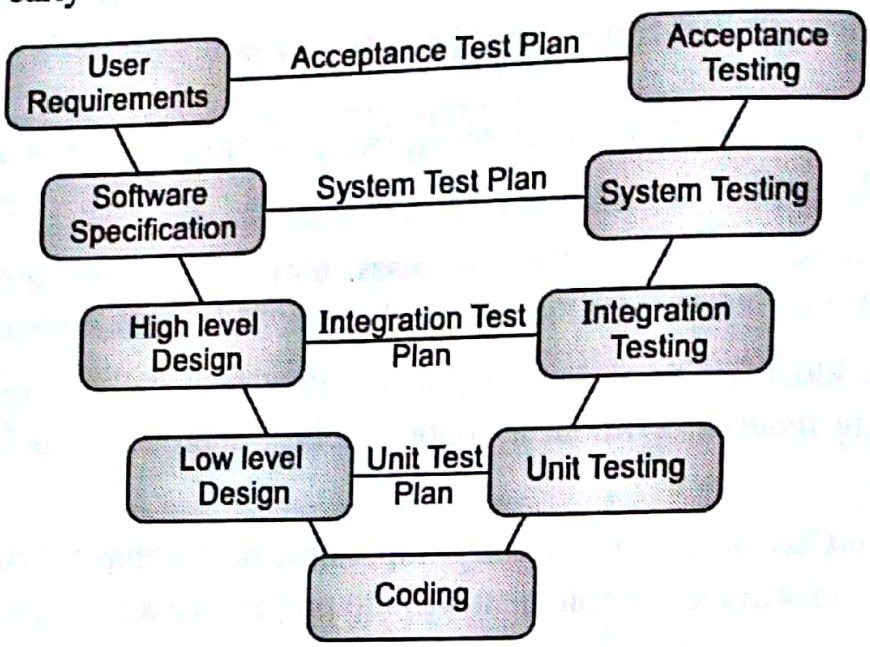

Fig. 2.12.2

- Like SDLC, V-model is also sequential model. Each phase must be completed before the next phase begins. Testing of the product is planned in parallel with a corresponding phase of development in **V-model**. The various phase of V-model are :

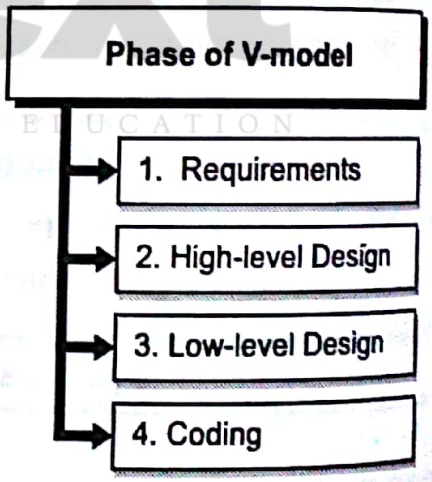

Fig. 2.12.3

➜ 1. Requirements

- In V-Model when the requirements are gathered in the form of BRS or SRS from Customer/Client for development of the project, the same requirements are provided to the testing team to create an acceptance and system test plan.
- The test plan focuses on meeting the functionality specified in the requirements gathering.

➜ 2. High-level Design

- In this phase, based on the high level design, software architecture is created. The modules, their relationships and dependencies, architectural diagrams, database tables, technology details are all finalized in this phase.
- An integration test plan is created in this phase as well in order to test the pieces of the software systems ability to work together.

3. Low-level Design

- In this phase each and every module or the software components are designed individually. Methods, classes, interfaces, data types etc are all finalized in this phase.
- It defines the actual logic for each and every component of the system. In this phase component tests are created.

4. Coding

- In this phase actual coding is done. Unit tests are created in this phase.
- The actual code is tested using the tests written during different phases of this model.

2.12.3.2 Prototyping Model

Description

- It refers to the activity of creating prototypes of software applications, for example, incomplete versions of the software program being developed.
- It is an activity that can occur in software development and it is used to visualize some component of the software to limit the gap of misunderstanding the customer requirements by the development team.
- This also will reduce the iterations may occur in the waterfall approach and hard to be implemented due to the inflexibility of the waterfall approach. So, when the final prototype is developed, the requirement is considered to be frozen.
- It has some types, such as :

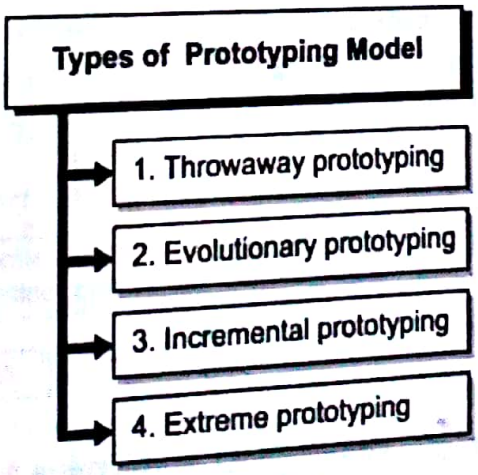

Fig. 2.12.4

1. Throwaway prototyping

Prototypes that are eventually discarded rather than becoming a part of the finally delivered software.

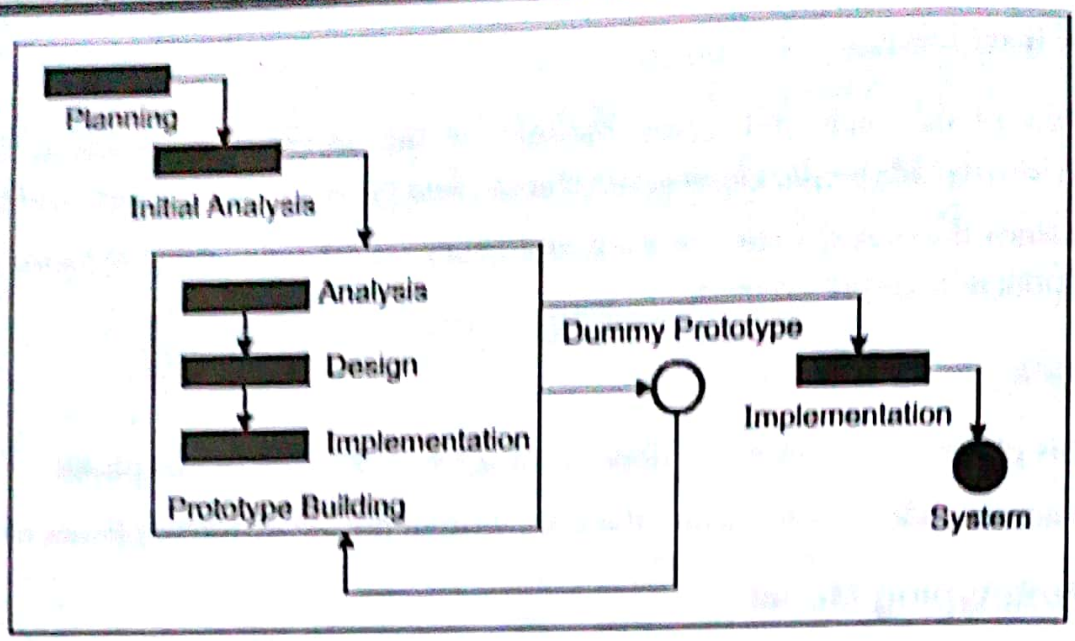

Fig. 2.12.5 : Throw-away Prototyping Methodology

➜ 2. Evolutionary prototyping

Prototypes that evolve into the final system through an iterative incorporation of user feedback.

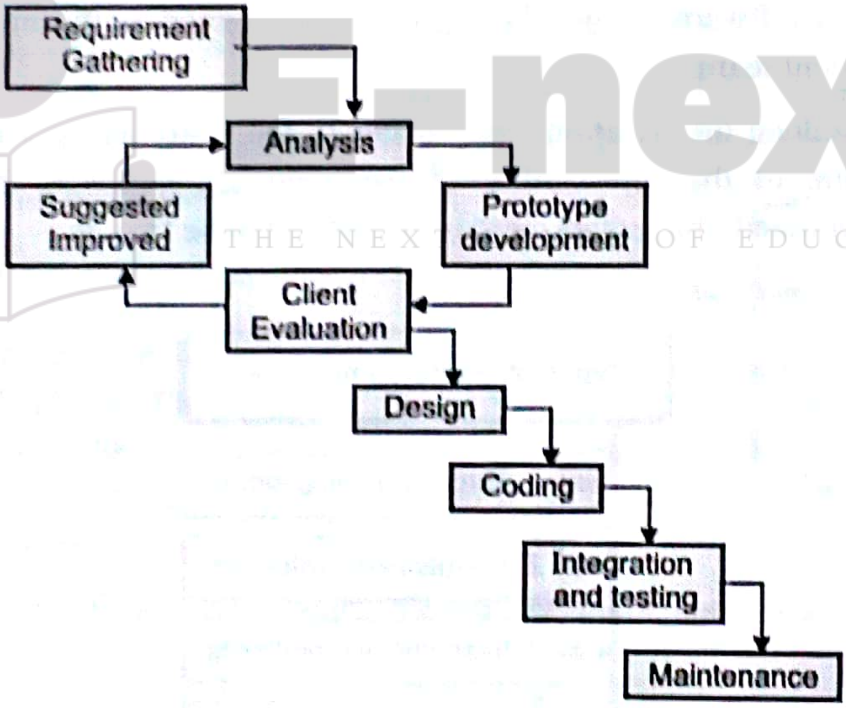

Fig. 2.12.6 : Evolutionary prototyping Model

➜ 3. Incremental prototyping

The final product is built as separate prototypes. In the end, the separate prototypes are merged in an overall design.

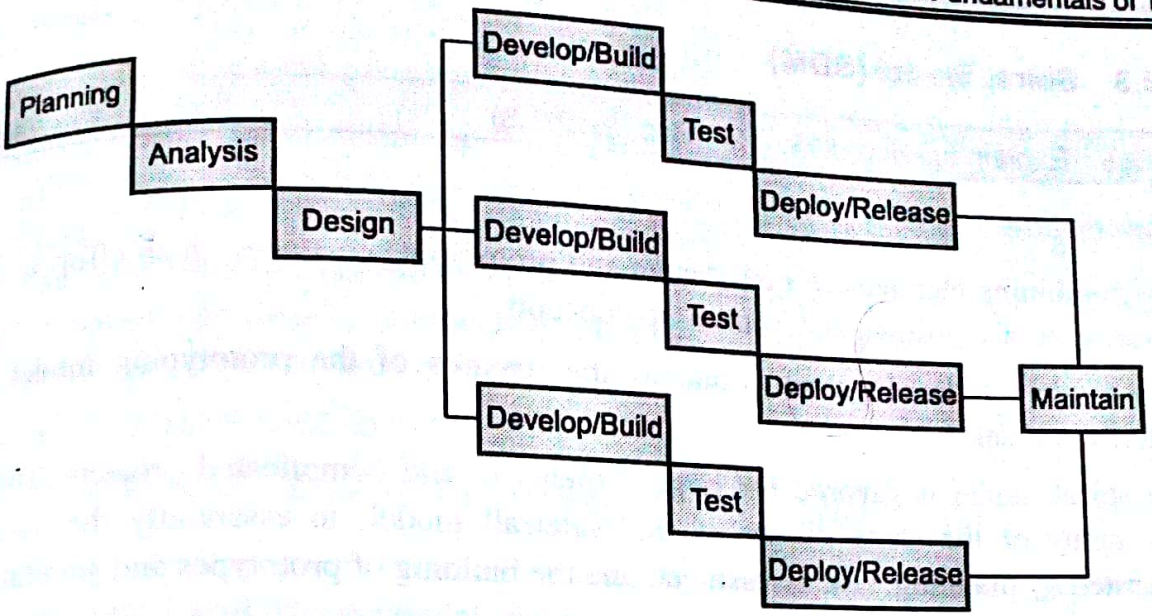

Fig. 2.12.7 : Staged / Incremental Model of SDLC

4. Extreme prototyping

This is used in web applications mainly. Basically, it breaks down web development into three phases, each one based on the preceding one. The first phase is a static prototype that consists mainly of HTML pages. In the second phase, the screens are programmed and fully functional using a simulated services layer. In the third phase, the services are implemented.

The usage

This process can be used with any software developing life cycle model. While this shall be chosen when you are developing a system has user interactions. So, if the system does not have user interactions, such as a system does some calculations shall not have prototypes.

Advantages and Disadvantages

Advantages	Disadvantages
- Reduced time and costs, but this can be a disadvantage if the developer loses time in developing the prototypes. - Improved and increased user involvement.	– Insufficient analysis. User confusion of prototype and finished system. – Developer misunderstanding of user objectives. – Excessive development time of the prototype. – It is costly to implement the prototypes

2.12.3.3 Spiral Model (SDM)

Q. 2.12.5 Explain spiral model. (Ref. Sec. 2.12.3.3) **(5 Marks)**

Description

- It is combining elements of both design and prototyping-in-stages, in an effort to combine advantages of top-down and bottom-up concepts.
- This model of development combines the features of the prototyping model and the waterfall model.
- The spiral model is favored for large, expensive, and complicated projects. This model uses many of the same phases as the waterfall model, in essentially the same order, separated by planning, risk assessment, and the building of prototypes and simulations.

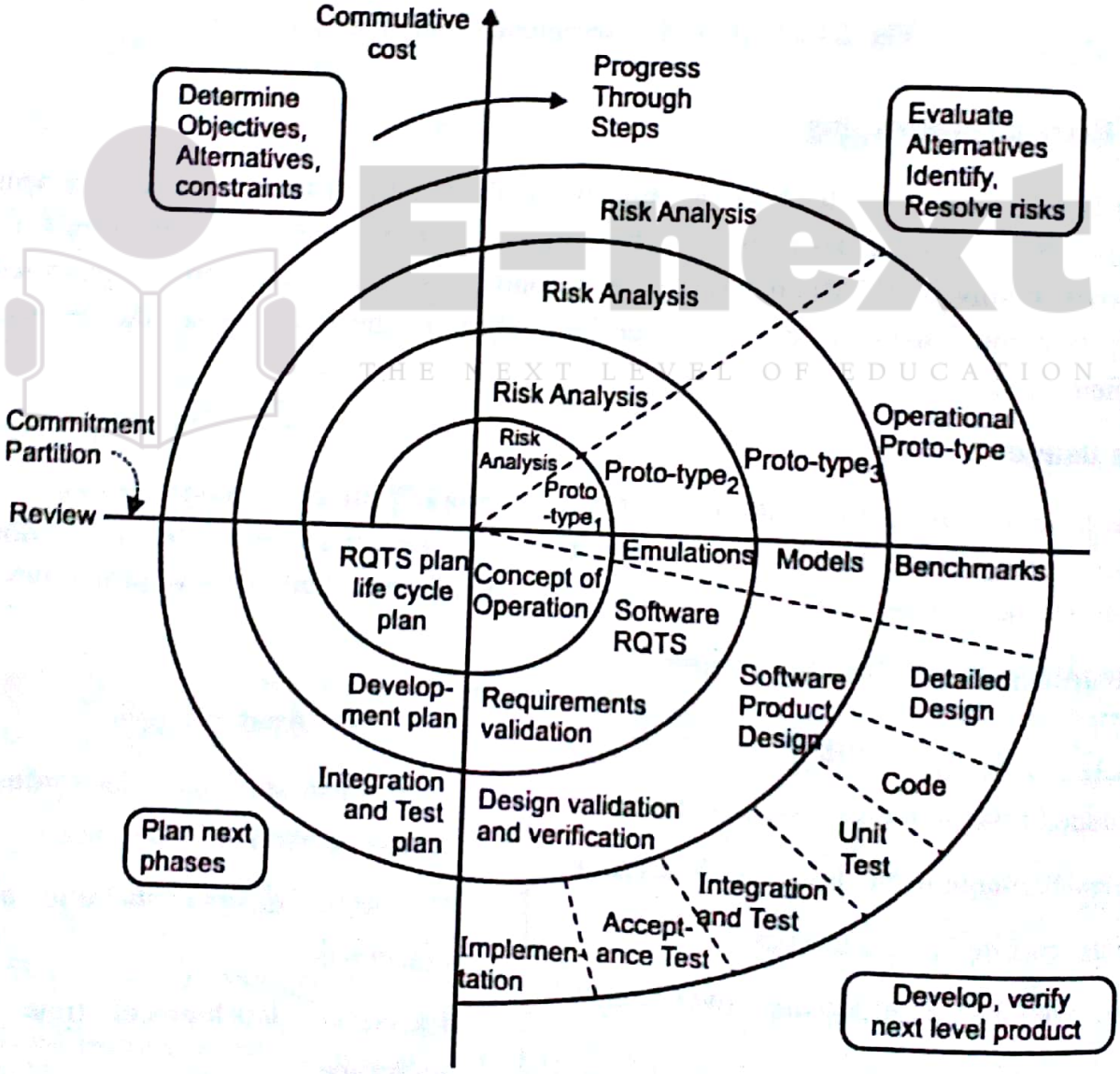

Fig. 2.12.8

The usage

It is used in the large applications and systems which built-in small phases or segments.

Advantages and Disadvantages

Advantages	Disadvantages
- Estimates (i.e. budget, schedule, etc.) become more realistic as work progressed because important issues are discovered earlier. - Early involvement of developers. - Manages risks and develops the system into phases.	– High cost and time to reach the final product. – Needs special skills to evaluate the risks and assumptions. – Highly customized limiting re-usability

2.12.3.4 Iterative and Incremental Model

Description

- It is developed to overcome the weaknesses of the waterfall model. It starts with an initial planning and ends with deployment with the cyclic interactions in between.
- The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental), allowing software developers to take advantage of what was learned during the development of earlier parts or versions of the system.
- It can consist of mini waterfalls or mini V-Shaped model :

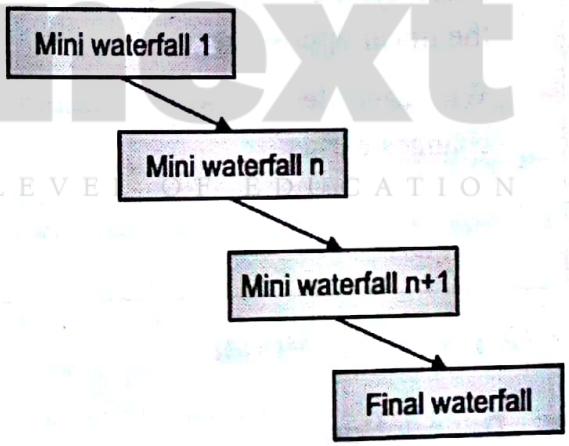

Fig. 2.12.9

The usage

- It is used in shrink-wrap application and large system which built-in small phases or segments.
- Also, can be used in a system has separated components, for example, ERP system. Which we can start with the budget module as a first iteration and then we can start with the inventory module and so forth.

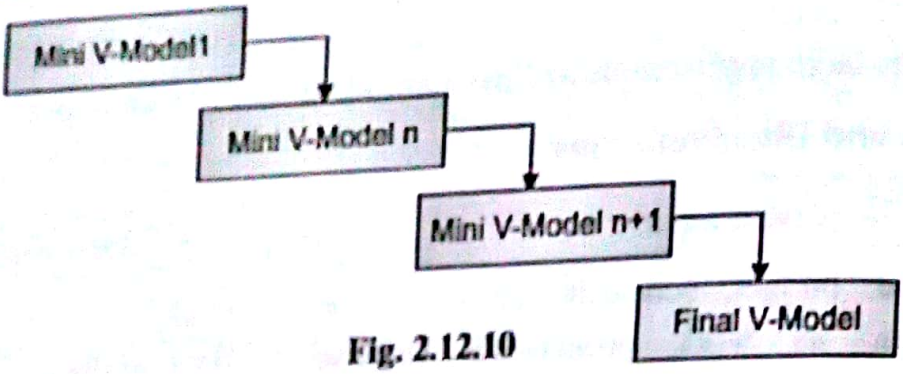

Fig. 2.12.10

☞ **Advantages and Disadvantages**

Advantages	Disadvantages
– Produces business value early in the development lifecycle.	– Requires heavy documentation.
– Better use of scarce resources through proper increment definition.	– Follows a defined set of processes.
– Can accommodate some change requests between increments.	– Defines increments based on function and feature dependencies.
– More focused on customer value than the linear approaches.	– Requires more customer involvement than the linear approaches.
– We can detect project issues and changes earlier.	– Partitioning the functions and features might be problematic.
	– Integration between the iterations can be an issue if it is not considered
	– during the development and project planning.

2.12.3.5 Agile Model

☞ **Description**

– It is based on iterative and incremental development, where requirements and solutions evolve through collaboration between cross-functional teams.

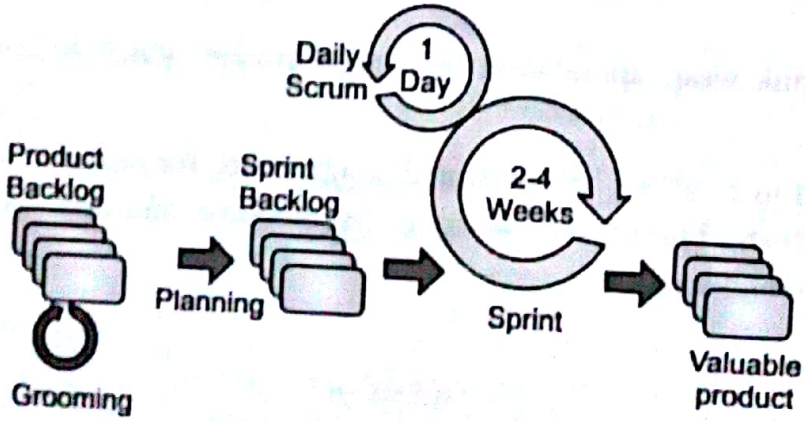

Fig. 2.12.11 : Scrum Agile Model

The usage

It can be used with any type of the project, but it needs more engagement from the customer and to be interactive.

Also, we can use it when the customer needs to have some functional requirement ready in less than three weeks and the requirements are not clear enough.

This will enable more valuable and workable piece for software early which also increases the customer satisfaction.

Advantages and Disadvantages

Advantages	Disadvantages
- Decrease the time required to avail some system features. - Face to face communication and continuous inputs from customer representative leaves no space for guesswork. - The end result is the high-quality software in the least possible time duration and satisfied customer.	– Scalability. – The ability and collaboration of the customer to express user needs. – Documentation is done at later stages. – Reduce the usability of components. – Needs special skills for the team.

Syllabus Topic : Test levels

2.13 Test Levels

Q. 2.13.1 What are test levels? (Ref. Sec. 2.13) **(5 Marks)**

1. Unit test level

- Unit tests (otherwise known as component tests) seek to identify defects in, and verify the functioning of, testable software modules, programs, items, classes etc.
- Unit tests are capable of verifying functionalities, as well as non-functional characteristics.
- The test cases are based on work products, such as the specifications of a component, the software design or the data model. Unit tests are used in order to be certain that the individual components are working correctly.

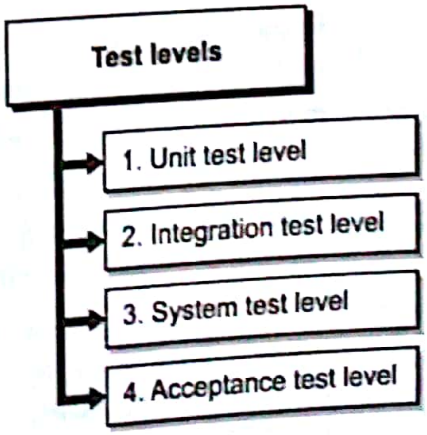

Fig. 2.13.1

- Unit tests mainly make use of white-box testing techniques and are therefore carried out with a knowledge of the code being tested and possibly also with the support of the development environment (the unit test framework, debugging tool, etc.).
- That is why these tests are often carried out by the developer that wrote the code or by a party of a similar calibre. Any defects identified are resolved as soon as they are detected.

➜ 2. Integration test level

- As far as the integration tests are concerned, these set out to test the interfaces between the various components and the interactions with different parts of a system (such as the control system, file system, hardware etc.). Integration tests can exist in more than one level and they can be carried out on test items of various sizes.
- We are able to distinguish between 2 separate levels of integration test :

  Component integration tests : The testing of the interaction between software components. These are carried out following the unit tests.

  System integration tests : The testing of the interaction between various systems. These are carried out following the system tests. The greater the scope of integration, the more difficult it becomes to isolate defects that exist in a specific component or system.
- At any point in the integration, the tests will concentrate solely on integration itself. For example: A new module A is integrated with a module B. During the integration tests, the primary focus will lie on the communication between the modules and not on the functionality of the modules themselves.

➜ 3. System test level

- In the case of system tests, our aim is to test the behaviour of an entire system, as defined within the project. For the system tests, no knowledge is required, a priori of the internal design or logic of the system; the majority of techniques used are primarily black box techniques.
- In the case of the system tests, the environment must correspond as closely as possible with the ultimate production environment, in order to minimise the risk of environment-specific defects.
- System tests are derived from risk specifications, requirement specifications, business processes, use-cases or other high-level definitions and are carried out within the context of functional requirements.
- What is more, system tests will also investigate the non-functional requirements (quality attributes). System tests typically include the following types of test: GUI tests, usability tests, regression tests, performance tests, error-handling tests, maintenance tests, compatibility tests, load tests, security tests, scalability tests, etc.

- The intention is that once the system test has been carried out, the supplier is satisfied that the system complies with all of the necessary requirements and is ready to be handed on to the customer (possibly with the exception of interaction with peer systems).

4. Acceptance test level

- Acceptance tests are often the responsibility of the customers or (end-)users of a system. Other stakeholders can also be involved, however. The purpose of acceptance testing is to gain confidence in the system, parts of the system or specific non-functional properties.
- Locating defects is not the primary aim of acceptance tests. Acceptance tests are a means of checking whether a system is ready for use.
- Acceptance tests can occur at more than just one test level. For example :
  - Acceptance testing of the usability of a component can take place during unit testing.
  - Acceptance testing of a new functional improvement can take place before the system tests.
  - Acceptance tests typically involve the following elements, which may also be regarded as a separate test level :

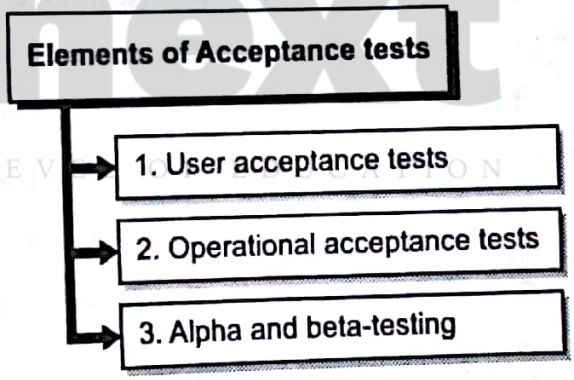

Fig. 2.13.2

1. User acceptance tests

Verifying the readiness of a system for use by business users

2. Operational acceptance tests

Testing the acceptance of system by system administrators. These consist of tests of backup/restore, disaster recovery, user management, maintenance tasks etc.

3. Alpha and beta-testing

This involves obtaining feedback from potential or existing customers, before the product is launched on the market. Alpha-testing takes place within the developer's own organisation. Beta-testing (field-testing) is carried out by individuals in their own working environment.

Syllabus Topic : Test Types, The Targets of Testing

2.13.1 Test Types, The Targets of Testing

Q. 2.13.2 Explain testing types. (Ref. Sec. 2.13.1) (5 Marks)

Functional tests

- A functional test is a kind of black box testing that is performed to confirm that the functionality of an application or system is behaving as expected.
- Functions (or features) are tested by feeding them input and examining the output. Functional testing ensures that the requirements are properly satisfied by the application.
- This type of testing is not concerned with how processing occurs, but rather, with the results of processing. It simulates actual system usage but does not make any system structure assumptions. Functional tests can be carried out at any test level.
  - **Testing based on Requirements :** Contains all the functional specifications which form a basis for all the tests to be conducted.
  - **Testing based on Business scenarios:** Contains the information about how the system will be perceived from a business process perspective.
- Typically, functional testing involves the following steps :
  - Identify functions that the software is expected to perform.
  - Create input data based on the function's specifications.
  - Determine the output based on the function's specifications.
  - Execute the test case.
  - Compare the actual and expected outputs.

Non-functional tests:

- Non-functional testing ensures that a system/application meets the specified performance requirements.
- In non-functional software testing, by performance we do not only mean response time, but several other factors such as security, scalability and usability of the application as well.
- Non-functional testing includes :
  - Reliability testing
  - Efficiency testing
  - Portability testing
  - Usability testing
  - Maintainability testing
  - Baseline testing

- Compliance testing
- Endurance testing
- Performance testing
- Security testing
- Volume testing
- Recovery testing
- Documentation testing
- Load testing
- Compatibility testing
- Scalability testing
- Stress testing
- Internationalization testing and Localization testing

- **Non-functional tests test "how" the system works**

- The sole purpose of non-functional testing is to ensure that the non-functional aspects of the application are tested and the application works well in context to the same.
- The purpose is to cover the testing of all the characteristics of the application which helps to provide an application that meets the business expectation.

Syllabus Topic : Maintenance Testing

2.13.2 Maintenance Testing

Q. 2.13.3 Explain maintenance testing. (Ref. Sec. 2.13.2) (5 Marks)

- **Maintenance testing** is a **test** that is performed to either identify equipment problems, diagnose equipment problems or to confirm that **repair** measures have been effective.
- Maintenance Testing is done on the already deployed software. The deployed software needs to be enhanced, changed or migrated to other hardware.
- The Testing done during this enhancement, change and migration cycle is known as maintenance testing.
- The importance of maintenance, testing and inspection schedules are as follows :
  1. To ensure the reliability and efficiency of the equipment and/or facilities.
  2. To provide smooth and satisfactory operation and performance within the facility without compromising the safety.
  3. To determine critical areas or equipment that needs immediate action.
  4. As the saying goes prevention is better than cure, by doing scheduled maintenance and inspection, all issues concerning the equipment or facilities can be taken into action immediately.

2.14 Exam Pack (Review Questions)

Syllabus Topic : Introduction, What Is Testing?

Q. 1 What is testing? Why is testing needed? *(Ans. : Refer sections 2.1.1 and 2.1.2)* **(5 Marks)**

Syllabus Topic : Necessity of Testing

Q. 2 What is testing? Why is testing needed? *(Ans. : Refer sections 2.1.1 and 2.1.2)* **(5 Marks)**

Syllabus Topic : Fundamental Test Process

Q. 3 Explain the fundamental process of testing. *(Ans. : Refer section 2.2)* **(5 Marks)**

Syllabus Topic : The Psychology of Testing

Q. 4 What is the psychology behind testing? *(Ans. : Refer section 2.3)* **(5 Marks)**

Syllabus Topic : Historical Perspective of Testing

Q. 5 What is the history behind testing? *(Ans. : Refer section 2.3.1)* **(5 Marks)**

Syllabus Topic : Definition of Testing

Q. 6 Define Testing. List and explain the approaches to testing. *(Ans. : Refer section 2.3.2 and 2.4)* **(5 Marks)**

Syllabus Topic : Approaches to Testing

Q. 7 Define Testing. List and explain the approaches to testing. *(Ans. : Refer section 2.3.2 and 2.4)* **(5 Marks)**

Syllabus Topic : Testing during Development Life Cycle

Q. 8 Explain how is testing carried out during SDLC? *(Ans. : Refer section 2.5)* **(5 Marks)**

Syllabus Topic : Requirement Traceability Matrix

Q. 9 What is Requirement Traceability Matrix? *(Ans. : Refer section 2.6)* **(5 Marks)**

Syllabus Topic : Essentials of Software Testing

Q. 10 What are essentials prerequisites for testing? *(Ans. : Refer section 2.7)* **(5 Marks)**

Syllabus Topic : Workbench

Q. 11 What is a workbench? *(Ans. : Refer section 2.7.1)* **(5 Marks)**

Syllabus Topic : Important Features of Testing Process

Q. 12 Explain the important features of testing. *(Ans. : Refer section 2.7.2)* **(5 Marks)**

Syllabus Topic : Misconceptions about Testing

Q. 13 What are the misconceptions behind testing? *(Ans. : Refer section 2.7.3)* **(5 Marks)**

Syllabus Topic : Principles of Software Testing

Q. 14 What re principles of testing? *(Ans. : Refer section 2.7.4)* **(5 Marks)**

Syllabus Topic : Salient Features of Good Testing

Q. 15 How do we know testing is good testing? *(Ans. : Refer section 2.7.5)* **(5 Marks)**

Syllabus Topic : Test Policy, Test Strategy or Test Approach

Q. 16 Differentiate between Test policy and Test strategy. *(Ans. : Refer section 2.8)* **(5 Marks)**

Syllabus Topic : Test Planning, Testing Process and Number of Defects Found in Testing

Q. 17 What is testing planning and test process? *(Ans. : Refer section 2.8.1)* **(5 Marks)**

Syllabus Topic : Test Team Efficiency

Q. 18 Write a short on Test Team efficiency. *(Ans. : Refer section 2.8.2)* **(5 Marks)**

Syllabus Topic : Mutation Testing

Q. 19 What is mutation testing? *(Ans. : Refer section 2.8.3)* **(5 Marks)**

Syllabus Topic : Challenges in Testing

Q. 20 What are the different challenges in testing? *(Ans. : Refer section 2.8.4)* **(5 Marks)**

Syllabus Topic : Test Team Approach

Q. 21 Who are members of a test team? *(Ans. : Refer section 2.8.5)* **(5 Marks)**

Syllabus Topic : Process Problems Faced by Testing

Q. 22 What are the process related problems that affect testing? *(Ans. : Refer section 2.9)* **(5 Marks)**

Syllabus Topic : Cost Aspect of Testing

Q. 23 Explain the cost aspect of testing. *(Ans. : Refer section 2.9.1)* **(5 Marks)**

☞ **Syllabus Topic : Establishing Testing Policy, Methods, Structured Approach to Testing**

Q. 24 What are the contents of a test policy? *(Ans. : Refer section 2.9.2)* **(5 Marks)**

☞ **Syllabus Topic : Categories of Defect**

Q. 25 What is a defect? What are the categories of defects? *(Ans. : Refer section 2.10)* **(5 Marks)**

☞ **Syllabus Topic : Defect, Error, or Mistake In Software**

Q. 26 Define Defect, Error and mistake *(Ans. : Refer section 2.10.2)* **(5 Marks)**

☞ **Syllabus Topic : Developing Test Strategy**

Q. 27 List and explain components of test strategy document. *(Ans. : Refer section 2.11)* **(5 Marks)**

☞ **Syllabus Topic : Developing Testing Methodologies (Test Plan), Testing Process**

Q. 28 What is a test plan? *(Ans. : Refer section 2.11.2)* **(5 Marks)**

☞ **Syllabus Topic : Test Methodologies/Approaches**

Q. 29 Define Test Methodologies. *(Ans. : Refer section 2.11.4)* **(5 Marks)**

☞ **Syllabus Topic : Attitude towards Testing (Common People Issues)/ People Challenges in Software Testing**

Q. 30 Why is it important to increase awareness of management towards testing? *(Ans. : Refer section 2.11.5)* **(5 Marks)**

☞ **Syllabus Topic : Raising Management Awareness for Testing**

Q. 31 Why is important to raise awareness about testing in management? *(Ans. : Refer section 2.12)* **(5 Marks)**

☞ **Syllabus Topic : Skills Required by Tester**

Q. 32 What are the skills required by testers? *(Ans. : Refer section 2.12.1)* **(5 Marks)**

☞ **Syllabus Topic : Testing Throughout the Software Life Cycle**

Q. 33 Why should testing be carried throughout the software life cycle? *(Ans. : Refer section 2.12.2)* **(5 Marks)**

Syllabus Topic : Software Development Models

Q. 34 Explain V-model. *(Ans. : Refer section 2.12.3)* **(5 Marks)**

Q. 35 Explain Spiral model. *(Ans. : Refer section 2.12.3.3)* **(5 Marks)**

Syllabus Topic : Test levels

Q. 36 What are test levels? *(Ans. : Refer section 2.13)* **(5 Marks)**

Syllabus Topic : Test Types, The Targets of Testing

Q. 37 Explain testing types. *(Ans. : Refer section 2.13.1)* **(5 Marks)**

Syllabus Topic : Maintenance Testing

Q. 38 Explain maintenance testing. *(Ans. : Refer section 2.13.2)* **(5 Marks)**