SYSTEM ANALYSIS & DESIGN

UNIT-4

System Testing:

System Testing includes testing of a fully integrated software system. Generally, a computer system is made with the integration of software. The software is developed in units and then interfaced with other software and hardware to create a complete computer system.

To check the end-to-end flow of an application or the software as a user is known as System testing. In this, we navigate (go through) all the necessary modules of an application and check if the end features or the end business works fine, and test the product as a whole system.

Types of System Testing:

1. Regression Testing:

Regression testing is performed under system testing to confirm and identify that if there's any defect in the system due to modification in any other part of the system. It makes sure, any changes done during the development process have not introduced a new defect and also gives assurance; old defects will not exist on the addition of new software over the time.

2. Load Testing:

Load testing is performed under system testing to clarify whether the system can work under real-time loads or not.

3. Functional Testing:

Functional testing of a system is performed to find if there's any missing function in the system. Tester makes a list of vital functions that should be in the system and can be added during functional testing and should improve quality of the system.

4. Recovery Testing:

Recovery testing of a system is performed under system testing to confirm reliability, trustworthiness, accountability of the system and all are lying on recouping skills of the system. It should be able to recover from all the possible system crashes successfully.

In this testing, we will test the application to check how well it recovers from the crashes or disasters.

5. Migration Testing:

Migration testing is performed to ensure that if the system needs to be modified in new infrastructure so it should be modified without any issue.

6. Usability Testing:

The purpose of this testing to make sure that the system is well familiar with the user and it meets its objective for what it supposed to do.

7. Software and Hardware Testing:

This testing of the system intends to check hardware and software compatibility. The hardware configuration must be compatible with the software to run it without any issue. Compatibility provides flexibility by providing interactions between hardware and software.

Advantages of System Testing:

- Verifies the system against the business, functional and technical requirements of the end users.
- It helps in getting maximum bugs before acceptance testing.
- System testing increases the confidence level of the team in the product before the product goes for acceptance testing.
- It is the first testing level in which the whole system is under test from end to end. So, it helps in finding important defects which, unit and integration testing could not detect.
- This testing phase uses the test environment which is similar to the real business environment or production environment. Consequently, it helps in boosting the confidence of users into the product.
- It is a black box testing hence testers do not need programming knowledge to perform it.

Disadvantages of System Testing:

- This testing is time consuming process than another testing techniques since it checks the entire product or software.
- The cost for the testing will be high since it covers the testing of entire software.
- It needs good debugging tool otherwise the hidden errors will not be found.

Unit Testing:

Unit testing involves the testing of each unit or an individual component of the software application. It is the first level of functional testing. The aim behind unit testing is to validate unit components with its performance.

A unit is a single testable part of a software system and tested during the development phase of the application software.

Whenever the application is ready and given to the Test engineer, he/she will start checking every component of the module or module of the application independently or one by one, and this process is known as Unit testing or components testing.

Advantages:

- Unit testing uses module approach due to that any part can be tested without waiting for completion of another parts testing.
- The developing team focuses on the provided functionality of the unit and how functionality should look in unit test suits to understand the unit API.
- Unit testing allows the developer to refactor code after a number of days and ensure the module still working without any defect.

Disadvantages:

- It cannot identify integration or broad level error as it works on units of the code.
- In the unit testing, evaluation of all execution paths is not possible, so unit testing is not able to catch each and every error in a program.
- It is best suitable for conjunction with other testing activities

Black box testing:

Black box testing is a technique of software testing which examines the functionality of software without peering into its internal structure or coding. The primary source of black box testing is a specification of requirements that is stated by the customer.

Generic steps of black box testing:

1. The black box test is based on the specification of requirements, so it is examined in the beginning.
2. In the second step, the tester creates a positive test scenario and an adverse test scenario by selecting valid and invalid input values to check that the software is processing them correctly or incorrectly.
3. In the third step, the tester develops various test cases such as decision table, all pairs test, equivalent division, error estimation, cause-effect graph, etc.
4. The fourth phase includes the execution of all test cases.
5. In the fifth step, the tester compares the expected output against the actual output.
6. In the sixth and final step, if there is any flaw in the software, then it is cured and tested again.

Advantages:

- Well suited and efficient for large code segments.
- Code access is not required.

- Clearly separates user's perspective from the developer's perspective through visibly defined roles.
- Large numbers of moderately skilled testers can test the application with no knowledge of implementation, programming language, or operating systems.

Disadvantages:

- Limited coverage, since only a selected number of test scenarios is actually performed.
- Inefficient testing, due to the fact that the tester only has limited knowledge about an application.
- Blind coverage, since the tester cannot target specific code segments or error-prone areas.
- The test cases are difficult to design.

White Box Testing:

The box testing approach of software testing consists of black box testing and white box testing. We are discussing here white box testing which also known as glass box testing, structural testing, clear box testing, open box testing and transparent box testing.

It tests internal coding and infrastructure of a software focus on checking of predefined inputs against expected and desired outputs. It is based on inner workings of an application and revolves around internal structure testing. In this type of testing programming skills are required to design test cases. The primary goal of white box testing is to focus on the flow of inputs and outputs through the software and strengthening the security of the software.

Generic steps of white box testing:

1. Design all test scenarios, test cases and prioritize them according to high priority number.
2. This step involves the study of code at runtime to examine the resource utilization, not accessed areas of the code, time taken by various methods and operations and so on.
3. In this step testing of internal subroutines takes place. Internal subroutines such as nonpublic methods, interfaces are able to handle all types of data appropriately or not.
4. This step focuses on testing of control statements like loops and conditional statements to check the efficiency and accuracy for different data inputs.
5. In the last step white box testing includes security testing to check all possible security loopholes by looking at how the code handles security.

Advantages of White box testing:

- White box testing optimizes code so hidden errors can be identified.
- Test cases of white box testing can be easily automated.
- This testing is more thorough than other testing approaches as it covers all code paths.
- It can be started in the SDLC phase even without GUI.

Disadvantages of White box testing:

- White box testing is too much time consuming when it comes to large-scale programming applications.
- White box testing is much expensive and complex.
- It can lead to production error because it is not detailed by the developers.
- White box testing needs professional programmers who have a detailed knowledge and understanding of programming language and implementation.

Difference B/W Black box testing & White box testing:

S.No.	Black Box Testing	White Box Testing
1.	It is a way of software testing in which the internal structure or the program or the code is hidden and nothing is known about it.	It is a way of testing the software in which the tester has knowledge about the internal structure or the code or the program of the software.
2.	Implementation of code is not needed for black box testing.	Code implementation is necessary for white box testing.
3.	It is mostly done by software testers.	It is mostly done by software developers.
4.	No knowledge of implementation is needed.	Knowledge of implementation is required.
5.	It can be referred to as outer or external software testing.	It is the inner or the internal software testing.
6.	It is a functional test of the software.	It is a structural test of the software.
7.	This testing can be initiated based on the requirement specifications document.	This type of testing of software is started after a detail design document.
8.	No knowledge of programming is required.	It is mandatory to have knowledge of programming.
9.	It is the behavior testing of the software.	It is the logic testing of the software.
10.	It is applicable to the higher levels of testing of software.	It is generally applicable to the lower levels of software testing.
11.	It is also called closed testing.	It is also called as clear box testing.
12.	It is least time consuming.	It is most time consuming.

Verification & Validation:

Verification:

Verification is the process of checking that a software achieves its goal without any bugs. It is the process to ensure whether the product that is developed is right or not. It verifies whether the developed product fulfills the requirements that we have. Verification is static testing.

Validation:

Validation is the process of checking whether the software product is up to the mark or in other words product has high level requirements. It is the process of checking the validation of product i.e., it checks what we are developing is the right product. it is validation of actual and expected product. Validation is the dynamic testing.

The difference between Verification and Validation is as follow:

Verification	Validation
It includes checking documents, design, codes and programs.	It includes testing and validating the actual product.
Verification is the static testing.	Validation is the dynamic testing.
It does *not* include the execution of the code.	It includes the execution of the code.
Methods used in verification are reviews, walkthroughs, inspections and desk-checking.	Methods used in validation are Black Box Testing, White Box Testing and non-functional testing.
It checks whether the software conforms to specifications or not.	It checks whether the software meets the requirements and expectations of a customer or not.
It can find the bugs in the early stage of the development.	It can only find the bugs that could not be found by the verification process.
The goal of verification is application and software architecture and specification.	The goal of validation is an actual product.
Quality assurance team does verification.	Validation is executed on software code with the help of testing team.
It comes before validation.	It comes after verification.
It consists of checking of documents/files and is performed by human.	It consists of execution of program and is performed by computer.

Integration testing:

Integration testing is the process of testing the interface between two software units or modules. It focuses on determining the correctness of the interface. The purpose of integration testing is to expose faults in the interaction between integrated units. Once all the modules have been unit tested, integration testing is performed.

Integration test approaches:

1. Big-Bang Integration Testing
2. Bottom-Up Integration Testing
3. Top-Down Integration Testing
4. Mixed Integration Testing

Implementation:

Implementation is a process of ensuring that the information system is operational. It involves-

- Constructing a new system from scratch
- Constructing a new system from the existing one.

Implementation allows the users to take over its operation for use and evaluation. It involves training the users to handle the system and plan for a smooth conversion.

Implementation planning:

An implementation plan is a document that outlines the steps your team should take to accomplish a shared goal or initiative. Implementation planning is the counterpart to a strategic plan.

The purpose of an implementation plan is to ensure that your team can answer the who, what, when, how, and why of a project before moving into the execution phase. In simple terms, it's the action plan that turns your strategy into specific tasks.

Steps of Implementation planning:

1. Define goals

The first step in the implementation process is defining your goals. Determine what you hope to accomplish when your project is complete, like whether you hope to win over a new marketing client or revamp your internal content strategy. Starting with your project objectives in mind can help flesh out your project plan.

2. Conduct research

Once you have a broad idea of the project goals you want to achieve, you can hone in on these goals by conducting research such as interviews, surveys, focus groups, or observations. Your research should come from key experts in your field. These experts may be team members or external stakeholders. Your research outcomes should include a list of what your project timeline, budget, and personnel may look like.

3. Map out risks

In step three, you'll map out all the potential risks you may face in your project. Risks can include anything from paid time off and holidays to budget constraints and loss of personnel.

A great way to map out your risks is by using a risk register. This tool will help you prioritize project risks and prepare for them accordingly. You can also conduct a SWOT analysis, which will identify any weaknesses or threats affecting your project.

4. Schedule milestones

Scheduling your project milestones is an important step in the planning process because these checkpoints help you track your progress during execution. Milestones serve as metrics—they are a way to measure how far you've come in your project and how far you have left to go.

5. Assign responsibilities and tasks

Every action plan must include a list of responsibilities with team members assigned to each one. By assigning responsibilities, you can assess the performance of each team member and monitor progress more closely.

Assigning responsibilities is different from assigning individual tasks. One team member may be responsible for overseeing the project review, while you may assign three other team members to handle the delivery and communication of the project to various teams for review.

6. Allocate resources

Resource allocation is one of the best ways to reduce risk. If you can plan out what resources you need for your project and ensure those resources will be available, you'll avoid the risk of running out of resources mid-project. If you notice that you don't have enough resources in this step of the implementation process, you can adjust your project accordingly before it kicks off. Resources may include money, personnel, software, equipment, and other physical or technical materials.

Conversion:

It is a process of migrating from the old system to the new one. It provides understandable and structured approach to improve the communication between management and project team.

Conversion Plan:

It contains description of all the activities that must occur during implementation of the new system and put it into operation. It anticipates possible problems and solutions to deal with them.

Conversion Methods:

The four methods of conversion are −

1. **Parallel Conversion:** Old and new systems are used simultaneously.
2. **Direct Cutover Conversion:** New system is implemented and old system is replaced completely.
3. **Pilot Approach:** Supports phased approach that gradually implements system across all users.

4. **Phase-In Method:** Working version of system implemented in one part of organization based on feedback, it is installed throughout the organization all alone or stage by stage.

Input Design:

In an information system, input is the raw data that is processed to produce output. During the input design, the developers must consider the input devices such as PC, MICR, OMR, etc.

Objectives for Input Design:

- To design data entry and input procedures
- To reduce input volume
- To design source documents for data capture or devise other data capture methods
- To design input data records, data entry screens, user interface screens, etc.
- To use validation checks and develop effective input controls.

Data Input Methods:

It is important to design appropriate data input methods to prevent errors while entering data. These methods depend on whether the data is entered by customers in forms manually and later entered by data entry operators, or data is directly entered by users on the PCs.

Some of the popular data input methods are –

1. Batch input method (Offline data input method)
2. Online data input method
3. Computer readable forms
4. Interactive data input

Output Design:

The design of output is the most important task of any system. During output design, developers identify the type of outputs needed, and consider the necessary output controls and prototype report layouts.

Objectives of Output Design:

- To develop output design that serves the intended purpose and eliminates the production of unwanted output.
- To develop the output design that meets the end users' requirements.
- To deliver the appropriate quantity of output.
- To form the output in appropriate format and direct it to the right person.
- To make the output available on time for making good decisions.

Types of outputs:

1. External Outputs

Manufacturers create and design external outputs for printers. External outputs enable the system to leave the trigger actions on the part of their recipients or confirm actions to their recipients.

Some of the external outputs are designed as turnaround outputs, which are implemented as a form and re-enter the system as an input.

2. Internal outputs

Internal outputs are present inside the system, and used by end-users and managers. They support the management in decision making and reporting.

There are three types of reports produced by management information –

Detailed Reports – They contain present information which has almost no filtering or restriction generated to assist management planning and control.

Summary Reports – They contain trends and potential problems which are categorized and summarized that are generated for managers who do not want details.

Exception Reports – They contain exceptions, filtered data to some condition or standard before presenting it to the manager, as information.

Forms Design:

Both forms and reports are the product of input and output design and are business document consisting of specified data. The main difference is that forms provide fields for data input but reports are purely used for reading. For example, order forms, employment and credit application, etc.

During form designing, the designers should know −

- who will use them
- where would they be delivered
- the purpose of the form or report

During form design, automated design tools enhance the developer's ability to prototype forms and reports and present them to end users for evaluation.

Objectives/Requirements of Good Form Design:

- To keep the screen simple by giving proper sequence, information, and clear captions.
- To meet the intended purpose by using appropriate forms.
- To ensure the completion of form with accuracy.

- To keep the forms attractive by using icons, inverse video, or blinking cursors etc.
- To facilitate navigation.

Types of Forms:

1. Flat Forms

It is a single copy form prepared manually or by a machine and printed on a paper. For additional copies of the original, carbon papers are inserted between copies.

It is a simplest and inexpensive form to design, print, and reproduce, which uses less volume.

2. Unit Set/Snap out Forms

These are papers with one-time carbons interleaved into unit sets for either handwritten or machine use.

Carbons may be either blue or black, standard grade medium intensity. Generally, blue carbons are best for handwritten forms while black carbons are best for machine use.

3. Continuous strip/Fanfold Forms

These are multiple unit forms joined in a continuous strip with perforations between each pair of forms.

It is a less expensive method for large volume use.

4. No Carbon Required (NCR) Paper

They use carbonless papers which have two chemical coatings (capsules), one on the face and the other on the back of a sheet of paper.

When pressure is applied, the two capsules interact and create an image.

Hardware /Software Selection:

Selecting hardware and software for implementing information system in an organization is a serious and time-consuming process that passes through several phases.

Major phases in selection:

The selection process should be viewed as a project, and a project team should be organized with management support. Several steps make up the selection process.

1. Requirements analysis
2. System specifications

3. Request for proposal (RFP)
4. Evaluation and validation
5. Vendor selection
6. Post-installation review

Software selection:

Software selection is a critical aspect for system development. There are 2 ways of acquiring the software.

1. Custom -made
2. Packages

Criteria for Software selection:

Reliability – It is the probability that the software will executed in a specific period of time without any failures.

Functionality – It is the definition of the facilities, performance and other factors that the user requires in the finished product.

Capacity – Capacity refers to the capability of the software package to handle the users requirements for size of files, number of data elements, and reports.

Flexibility – It is a measure of effort required to modify an operational program.

Usability – This criteria refers to the effort required to operate, prepare the input, and interpret the output of a program.

Security – It is a measure of the likelihood that a system's user can accidentally or intentionally access or destroy unauthorized data.

Performance – It is a measure of the capacity of the software package to do what it is expected to do.

Serviceability –This criteria focuses on documentation and vendor support.

Ownership – Who owns the software ,and to consider whether he has the right to access the software, or he can sell or modify the software.

Minimal costs – Cost is a major consideration in deciding between in-house and vendor software.

Hardware Selection Criteria:

- Hardware must support current software as well as software planned for procurement over the next planning interval.
- Hardware must be compatible with existing or planned networks

- Hardware must be upgradeable and expandable to meet the needs of the next planning interval
- Hardware warranties must be of an appropriate length
- Hardware maintenance must be performed by [local/remote vendor, in-house personnel]
- Whenever feasible, hardware standards will dictate procurement of like brands and configurations to simplify installation and support
- Routine assessments of installed infrastructure will feed an upgrade/replace decision process

The Make or Buy Decision:

The make-or-buy decision is the act of making a strategic choice between producing an item internally (in-house) or buying it externally (from an outside supplier). In a make-or-buy decision, the most important factors to consider are part of quantitative analysis, such as the associated costs of production and whether the business has the capacity to produce at required levels. Make or buy decision is always a valid concept in business.

Four Numbers You Should Know

When you are supposed to make a make-or-buy decision, there are four numbers you need to be aware of. Your decision will be based on the values of these four numbers. Let's have a look at the numbers now. They are quite self-explanatory.

- The volume
- The fixed cost of making
- Per-unit direct cost when making
- Per-unit cost when buying

Reasons for Making:

There are number of reasons a company would consider when it comes to making in-house. Following are a few:

- Cost concerns
- Desire to expand the manufacturing focus
- Need of direct control over the product
- Intellectual property concerns
- Quality control concerns
- Supplier unreliability
- Lack of competent suppliers
- Volume too small to get a supplier attracted
- Reduction of logistic costs (shipping etc.)
- To maintain a backup source
- Political and environment reasons

- Organizational pride

Reasons for Buying:

Following are some of the reasons companies may consider when it comes to buying from a supplier:

- Lack of technical experience
- Supplier's expertise on the technical areas and the domain
- Cost considerations
- Need of small volume
- Insufficient capacity to produce in-house
- Brand preferences
- Strategic partnerships

Maintenance:

Maintenance is fundamental service for the manufacturing industry to increase the performance of production facilities from both economic and environmental perspectives. The results obtained from the evaluation process help the organization to determine whether its information systems are effective and efficient or otherwise.

The process of monitoring, evaluating, and modifying of existing information systems to make required or desirable improvements may be termed as System Maintenance. System maintenance is an ongoing activity, which covers a wide variety of activities, including removing program and design errors, updating documentation and test data and updating user support.

For the purpose of convenience, maintenance may be categorized into three classes, namely:

1. Corrective
2. Adaptive
3. Perfective.

Documentation:

Documentation is one of the systems which is used to communicate, instruct and record the information for any reference or operational purpose. They are very useful for representing the formal flow of the present system. It is important that prepared document must be updated on regular basis to trace the progress of the system easily.

After the implementation of system if the system is working improperly, then documentation helps the administrator to understand the flow of data in the system to correct the flaws and get the system working.

Importance of Documentation:

- It can reduce system downtime, cut costs, and speed up maintenance tasks.
- It provides the clear description of formal flow of present system and helps to understand the type of input data and how the output can be produced.
- It provides effective and efficient way of communication between technical and nontechnical users about system.
- It facilitates the training of new user so that he can easily understand the flow of system.
- It helps the user to solve the problems such as troubleshooting and helps the manager to take better final decisions of the organization system.
- It provides better control to the internal or external working of the system.

Types of Documentations:

When it comes to System Design, there are following four main documentations −

1. Program Documentation:

- It describes inputs, outputs, and processing logic for all the program modules.
- The program documentation process starts in the system analysis phase and continues during implementation.
- This documentation guides programmers, who construct modules that are well supported by internal and external comments and descriptions that can be understood and maintained easily.

2. Operations Documentation:

Operations documentation contains all the information needed for processing and distributing online and printed output. Operations documentation should be clear, concise, and available online if possible.

It includes the following information −

- Program, systems analyst, programmer, and system identification.
- Scheduling information for printed output, such as report, execution frequency, and deadlines.
- Input files, their source, output files, and their destinations.
- E-mail and report distribution lists.
- Special forms required, including online forms.
- Error and informational messages to operators and restart procedures.
- Special instructions, such as security requirements.

3. User Documentation:

It includes instructions and information to the users who will interact with the system. For example, user manuals, help guides, and tutorials. User documentation is valuable in training

users and for reference purpose. It must be clear, understandable, and readily accessible to users at all levels.

The users, system owners, analysts, and programmers, all put combined efforts to develop a user's guide.

A user documentation should include –

- A system overview that clearly describes all major system features, capabilities, and limitations.
- Description of source document content, preparation, processing, and, samples.
- Overview of menu and data entry screen options, contents, and processing instructions.
- Security and audit trail information.
- Explanation of responsibility for specific input, output, or processing requirements.
- Procedures for requesting changes and reporting problems.
- Examples of exceptions and error situations.
- Frequently asked questions (FAQs).
- Explanation of how to get help and procedures for updating the user manual.

4. System Documentation:

System documentation serves as the technical specifications for the IS and how the objectives of the IS are accomplished. Users, managers and IS owners need never reference system documentation. System documentation provides the basis for understanding the technical aspects of the IS when modifications are made.

- It describes each program within the IS and the entire IS itself.
- It describes the system's functions, the way they are implemented, each program's purpose within the entire IS with respect to the order of execution, information passed to and from programs, and overall system flow.
- It includes data dictionary entries, data flow diagrams, object models, screen layouts, source documents, and the systems request that initiated the project.
- Most of the system documentation is prepared during the system analysis and system design phases.

Disaster Recovery Plan (DRP):

A disaster recovery plan (DRP) is a documented, structured approach with instructions for responding to unplanned incidents.

This step-by-step plan consists of the precautions to minimize the effects of a disaster so the organization can continue to operate or quickly resume mission-critical functions. Typically, disaster recovery planning involves an analysis of business processes and continuity needs.

Recovery strategies:

Recovery strategies define an organization's plans for responding to an incident, while disaster recovery plans describe how the organization should respond.

In determining a recovery strategy, organizations should consider such issues as:

- Budget
- Resources -- people and physical facilities
- Management's position on risks
- Technology
- Data
- Suppliers

Disaster recovery planning steps:

The disaster recovery plan process involves more than simply writing the document.

In advance of the writing, a risk analysis and business impact analysis help determine where to focus resources in the disaster recovery planning process.

Disaster recovery plans are living documents. Involving employees -- from management to entry-level -- helps to increase the value of the plan.

Creating a disaster recovery plan:

An organization can begin its DR plan with a summary of vital action steps and a list of important contacts, so the most essential information is quickly and easily accessible.

The plan should define the roles and responsibilities of disaster recovery team members and outline the criteria to launch the plan into action. The plan then specifies, in detail, the incident response and recovery activities.