UNIT -III

Testing & Implementation: Introduction to Testing, Understanding Testing, Applying Testing. Challenges and Opportunities in Applying Verification and Validation.

(ii)

Implementation : Understanding Implementation. Applying Implementation Planning with example, Challenges and Opportunities of Implementation Planning

14

Testing, Implementation & Maintenance

Introduction

The techniques and guidelines introduced in the preceding sessions if properly followed will produce low-failure systems. However, even if techniques are followed, the analyst must not assume that the necessary quality standards have been met. Quality Assurance is the review of software products and related documentation for completeness, correctness, reliability, and maintainability.

Managing Quality Assurance

Quality assurance is the review of software products and related documentation for completeness, correctness, reliability and maintainability.

It also includes assurance, that the system meets the specification and the requirements for its intended use and performance.

Quality assurance can be done by:

- Testing
- Verification And Validation.

Testing

Testing is generally done at two levels – testing of individual modules and testing of the entire system (systems testing).

During systems testing, the system is used experimentally to ensure that the software does not fail, i.e., that it will run according to its specifications and in the way users expect.

Special test data are input for processing, and the results examined. A limited number of users may be allowed to use the system so analysts can see whether they use it in unforeseen ways. It is preferable to discover any surprises before the organization implements the system and depends on it.

Testing is done throughout systems development at various stages (not just at the end). It is always a good practice to test the system at many different levels

at various intervals, that is, sub-systems, program modules as work progresses and finally the system as a whole If this is not done, then the poorly tested system can fail after installation.

As you may already have gathered, testing is a very tedious and time-consuming job. For a test to be successful the tester, should try and make the program fail. The tester maybe an analyst, programmer, or specialist trained in software testing. One should try and find areas in which the program can fail. Each test case is designed with the intent of finding errors in the way the system will process it.

Thorough testing of programs do not guarantee the reliability of systems. It is done to assure that the system runs error free.

Testing Strategies

There are two general strategies for testing software. This section examines both the strategies of code testing and specification testing.

Code Testing

The code testing strategy examines the logic of the program. For this testing method, the analyst develops test cases that result in executing every instruction in the program or module, that is, every path through the program is tested.

A path is a specific combination of conditions that is handled by the program. The testing of every path in the program is not always possible as there could be several thousands, and financial and time limitations will not permit this. Generally, all the frequently used paths undergo testing.

Specification Testing

To perform specification testing, the analyst examines the program specifications wherein states what the program should do and how it should perform under various conditions. Then, test cases are developed for each condition or combination of conditions and submitted for processing.

By examining the results, the analyst can determine whether the program performs according to its specified requirements. This testing does not look into the program to study the code or path, it looks at the program as a whole. The assumption here is that, if the program meets the specifications it will not fail.

Type of Test Data

There are two very different sources of test data:

- Live
- Artificial

Both have advantages and disadvantages.

Using live test data

Live test data are those that are actually extracted from organization files. This shows you how the system will perform on typical data. Although, the data may be the best, it is difficult to obtain sufficient amounts to conduct extensive testing. All combinations and conditions of the system are not tested with this data. This may not contain values that may cause system failure.

Using Artificial Data

Artificial test data are solely for test purposes. They are to be generated to test all combinations of formats and values. They are generated using the utility programs of the information systems. Using this type of data all logic and control paths through the program can be tested.

For best results, the artificial test data should be generated by persons other than those who wrote the programs. Automated test data generators are also available.

Levels of Testing

As already mentioned, testing is carried out at different levels and at various intervals.

Unit Testing

This involves the tests carried out on modules/programs which make up a system. This is also called as program testing. The units in a system are the modules and routines that are assembled and integrated to perform a specific function.

In a large system, many modules at different levels are needed. Unit testing focuses first on the modules, independently of one another, to locate errors.

The programs should be tested for correctness of logic applied and should detect errors in coding. For example in the payroll system, all the calculations should be tested by feeding the system with all combinations of data.

Valid and invalid data should be created and the programs should be made to process this data to catch errors. For example in the payroll system, the employee no. consists of three digits, so during testing one should ensure that the programs do not accept anything other than a 3 digit code for the employee no.

Another e.g. for valid and invalid data check is that, in case a three digit no. is entered during the entry of transaction, and that number does not exist in the master file, or if the number entered is an exit case, then the program should not allow the entry of such cases.

All dates that are entered should be validated. No program should accept invalid dates. The checks that need to be incorporated are : in the month of Feb the date cannot be more than 29. For the months having 30 days one chould not be allowed to enter 31.

All conditions present in the program should be tested. Before proceeding one must make sure that all the programs are working independently.

Systems Testing

When unit tests are satisfactorily concluded, the system as a complete entity must be tested. At this stage, end-users and operators become actively involved in testing. While testing one should also test to find discrepancies between the system and its original objective, current specifications and systems documentation.

For example, one module may expect the data item for employee number to be numeric field, while other modules expect it to be a character data item. The system itself may not report this error, but the output may show unexpected results. A record maybe created and stored in one module, using the employee number as a numeric field. If this is later sought on retrieval with the expectation that it will be a character field, the field will not be recognized and the message REQUESTED RECORD NOT FOUND will be displayed.

Systems testing must also verify that file sizes are adequate and that indexes have been built properly. Sorting and reindexing procedures assumed to be present in lower-level modules must be tested at the systems level to see that they in fact exist and achieve the results modules expect.

Special Systems Tests

There are other tests that are in a special category as they do not focus on the normal running of the system. They are listed below :

(a) Peak Load Test

This is used to determine whether the system will handle the volume of activities that occur when the system is at peak of its processing demand. For instance when all terminals are active at the same time.

This test applies mainly for on-line systems. For example, in a banking system, analyst want to know what will happen if all tellers sign on at their terminals at the same time before start of the business day. Will the system handle them one at a time without incident, will it attempt to handle all of them at once and be so confused that it 'locks up' and must be restarted, or will terminal addresses be lost? The only way sure way to find out is to test for it.

(b) Storage Testing

This test is to be carried out to determine the capacity of the system to store transaction data on a disk or in other files. Capacities here are measured in terms of the number of records that a disk will handle or a file can contain. If this test is not carried out then there are possibilities that during installation one may discover that, there is not enough storage capacity for transactions and master file recordes.

(c) Performance Time Testing

This test refers to the response time of the system being installed. Performance time testing is conducted prior to implementation to determine how long it takes to receive a response to a inquiry, make a backup copy of a file, or send a transmission and receive a response.

This also includes test runs to time indexing or resorting of large files of the size the system will have during a typical run or to prepare a report. A system may run well with only a handful of test transactions, may be unacceptably slow when fully loaded. This should be done using the entire volume of live data.

(d) Recovery Testing

Analyst must never be too sure of anything. He must always be prepared for the worst. One should assume that the system will fail and data will be damaged or lost. Even though plans and procedures are written to cover these situations, they also must be tested.

(e) Procedure Testing

Documentation and run manuals telling the user how to perform certain functions are tested quite easily by asking the user to follow them exactly through a series of events.

It is surprising how not including instructions about aspects such as, when to depress the enter key, removing the diskettes before putting off the power and so on, could cause problems. This type of testing brings out what is not mentioned in the documentation, and also the errors in them.

(f) Human Factors

In case during processing, the screen goes blank, the operator may start to wonder as to what is happening can he could just about do anything such as press the enter key a number of times, or switch off the system and so on, but if a message is displayed saying that the processing is in progress and asking the operator to wait, then these types of problems can be avoided.

Thus, during this test we determine how users will use the system when processing data or preparing reports.

Verification and Validation

In the context of testing, "*Verification and Validation*" are very widely and commonly used terms. Most of the times, we consider the terms same, but actually the terms are quite different.

The terms Verification and Validation are commonly used in software engineering to mean two different types of analysis. The usual definitions are:

- ***Validation: Are we building the right system?***
- ***Verification: Are we building the system right?***

In other words, validation is concerned with checking that the system will meet the customer's actual needs, while verification is concerned with whether the system is well-engineered, error-free, and so on. Verification will help to determine whether the software is of high quality, but it will not ensure that the system is useful.

Verification testing runs the system in a simulated environment using simulated data. This simulated test is sometimes called alpha testing. The simulated test is primarily looking for errors and omissions regarging end users and design specifications that were specified in the earlier phases but not fulfilled during construction.

Validation refers to the process of using software in a live environment in order to find errors. The feedback from the validation phase generally produces changes in the software to deal with errors and failures that are uncovered. Then a set of user sites is selected that puts the system into use on a live basis. They are called beta test sites.

The beta test sites use the system in day-to-day activities. They process live transactions and produce normal system output. The system is live in every sense of the word, except that the users are aware they are using a system that can fail. But the transactions that are entered and the persons using the system are real. Validation may continue for several months. During the course of validating the system, failure may occur and the software will be changed. Continued use may produce additional failures and need for still more changes.

Verification	Validation
1. Evaluates the intermediary products to check whether it meets the specific requirements of the particular phase	Evaluates the final products to check whether it meets the business needs.
2. Checks whether the product is built as per the specified requirement and design specification	It determines whether the software is fit for use and satisfy the business need.
3. Checks "Are we building the product right ?	Checks "Are we building the right product"?
4. This is done without executing the software	Is done with executing the software
5. Involves all the static testing techniques	Includes all the dynamic testing techniques.

6. Examples includes reviews, inspection and walkthrough	Example includes all types of testing like smoke, regression, functional, systems and UAT
7. Verification is a static practice of verifying documents, design, code and program.	Validation is a dynamic mechanism of validating and testing the actual product.
8. It does not involve executing the code.	It always involves executing the code.
9. It is human based checking of documents and files.	It is computer based execution of program.
10. Verification uses methods like inspections, reviews, walk theroughs, and Desk-checking etc.	Validation uses methods like black box (functional) testing, gray box testing, and white box (structural) testing etc.
11. Verification is to check whether the software conforms to specifications.	Validation is to check whether software meets the customer expectations and requirements.
12. Target is requirements specification, application and software architecture, high level, complete design, and database design etc.	Target is actual product-a unit, a module, a bent of integrated modules, and effective final product.
13. Verification is done by QA team to ensure that the software is as per the specifications in the SRS document.	Validation is carried out with the involvement of testing team.
14. It generally comes first-done before validation	It generally follows after verification.

Example of verification and validation

Suppose we have the specifications related to the project than by checking that specifications without executing to see whether the specifications are up to the mark or not is what we have done in verification.

Similarly Validation of the software is done to make sure that the software always meets the requirements of the customer by executing the specifications of the project and product.

Note that the customer and end users are concerned in validation of the software.

It is also crucial to differentiate between end users, and customers. Considering example, if you are developing a library monitoring system, the librarian is the client and the person who issue the books, collect fines etc. are comes under the category of the end users.

Applying Testing by Software Testing Life Cycle (STLC)

Software Testing Life Cycle (STLC) defines the steps/ stages/ phases in testing of software. However, there is no fixed standard STLC in the world and it basically varies as per the following:

- Software Development Life Cycle
- Whims of the Management

Nevertheless, Software Testing Life Cycle, in general, comprises of the following phases:

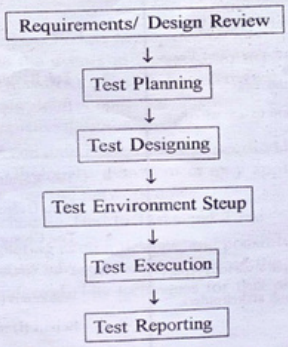

Fig. : Software Testing Life Cycle

Phase	Activity	Deliverables	Necessity
Requirements/ Design Review	You review the software requirements/ design (Well, if they exist.)	• 'Review Defect' Reports	Curiosity
Test Planning	Once you have gathered a general idea of what needs to be tested, you 'plan' for the tests.	• Test Plan • Test Estimation • Test Schedule	Farsightedness
Test Designing	You design/ detail your tests on the basis of detailed requirements/design of the software (sometimes, on the basis of your imagination).	• Test Cases / Test Scripts/Test Data • Requirements Traceability Matrix	Creativity
Test Environment Setup	You setup the test environment (server/ client/ network, etc) with the goal of replicating the end-users' environment.	• Test Environment	Rich company
Test Execution	You execute your Test Cases/ Scripts in the Test Environment to see whether they pass.	• Test Results (Incremental) • Defect Reports	Patience
Test Reporting	You prepare various report for various stakeholders.	• Test Results (Final) • Test/ Defect Metrics • Test Closure Report • Who Worked Late & on Weekends (WWLW) Report [Depending on how fussy your Management is]	Diplomacy

Note that the STLC phases mentioned above do not necessarily have to be in the order listed; some phases can sometimes run in parallel (For instance, Test Designing and Test Execution). And, in extreme cases, the phases might also be reversed (For instance, when there is Cursing prior to Testing).

Interestingly, no matter how well-defined a Software Testing Life Cycle you have in your project or organization, there are chances that you will invariably witness the following widely-popular cycle:

- Testing
- Cursing

In this type of STLC, you skip phases like design review, test planning, etc – in the hope that the skipping will save you some time and/or cost [But, it never does].

Implementation

After proper testing and validation, the question arises whether the system can be implemented or not. Implementation includes all those activities that take place to convert from old system to the new.

The new system may be totally new, replacing an existing manual or automated system, or it may be a major modification to an existing system. N either case, proper implementation is essential to provide a reliable system to meet organization requirements.

Plan the Implementation

After proper testing and validation, the question aeries whether the system can be implemented or not. Implementation includes all those activities that take place to convert from old system to the new.

The new system may be totally new, replacing an existing manual or automated system, or it may be a major modification to an existing system. Either case, proper implementation is essential to provide a reliable system to meet organization requirements.

The three main phases in implementation take place in series; these are the initial installation; the test of the system as a whole; and the evaluation, maintenance, and control of the system. On the other hand, many implementa-tion activities should be undertaken in parallel to reduce implementation time. For example, acquisition of data for the database and forms design for collection and dissemination of information may be carried out in parallel. Training of personnel and preparation of software may be in parallel with each other and with other implementation activities.

It is apparent, then, that the first step in the implementation procedure is to plan the implementation. Although some analysts include the planning of the implementation with the design of the system, we believe that it is operationally

significant to include it in the implementation stage, for several reasons. First, the planning and the action to implement the plan should be bound closely together. Planning is the first step of management, not the last. Further, the MIS design and the urgent need for the system at the time the design is completed will weigh heavily on the plan for implementation. And, finally, the planning process is a function of line management, at least as far as key decisions or alternative plans are concerned. The systems analyst may prepare plans to assist managers, but managers must have the last say. At the same time, managers require the services of the systems analyst to detail plans. The managers prefer to make decisions based upon the most recent information: the MIS specifications, the proposed plans of the systems analyst, and the current operating situation. The planning for the project of implementation should follow the proce-dures for project planning described in Chapter 6. Once the conversion method has been described, the specific steps are as we shall delineate here.

Identify the Implementation Tasks

The major implementation tasks, or milestones, usually consist of

1. Planning the implementation activities
2. Acquiring and laying out facilities and offices
3. Organizing the personnel for implementation
4. Developing procedures for installation and testing
5. Developing the training program for operating personnel
6. Completing the system's software
7. Acquiring required hardware
8. Generating files
9. Designing forms
10. Testing of the entire system
11. Completing cutover to the new system
12. Documenting the system
13. Evaluating the MIS
14. Providing system maintenance (debugging and improving)

The plans should list all subtasks for each of these major tasks so that individuals in the organization may be assigned specific responsibilities.

Establish Relationships Among Tasks

For small projects, the order of performance may simply be described in text form. However, even in small projects, a Gantt chart or network diagram makes visualization of the plan and schedule much clearer! In large projects, many

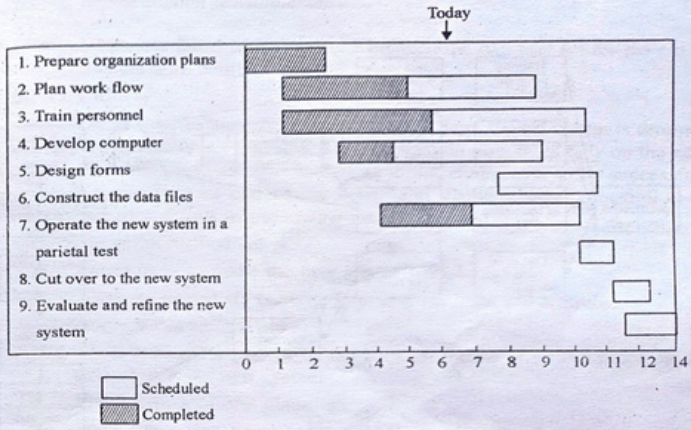

Fig. 14.2 : Gantt Chart for MIS Implementation

concurrent and sequential activities are interrelated, so that a network diagram must be employed in any good plan. Figure 9-2 shows a Gantt chart and Figure 9-3 shows a network diagram (condensed) for illustrating task relationships.

Establish a Schedule

A first estimate of the schedule is prepared by having the system designers estimate the times between the events in the program network. The critical path (longest time through the network) can then be calculated. The end date is thus established once the starting date is specified. Figures 9-2 and 9-3 indicate how times are shown for the implementation activities.

The actual desired end date is then usually specified by management on the basis of this information. Obviously, management may apply pressure or provide additional personnel to shorten the network tunes.

Prepare a Cost Schedule Tied to Tasks and Time

The cost for completing each milestone, and possibly each task required to complete a milestone, should be established as part of the plan; then the rate of expenditures should be budgeted. The techniques for this phase of planning were covered in Chapter 6.

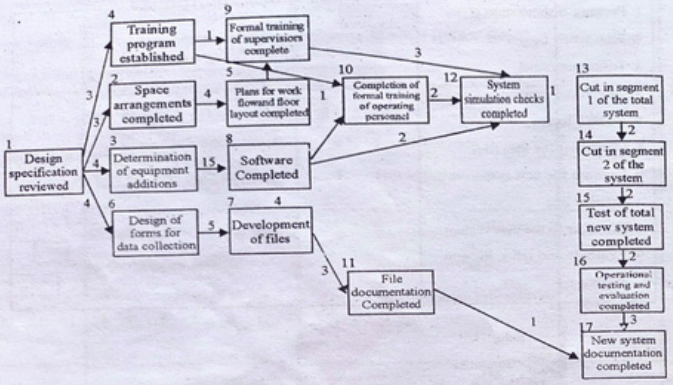

Fig. 14.3 : Network Diagram for System implementation

Establish a Reporting and Control System

Reporting and control of the work in progress may be obtained by weekly meetings of the key people involved or by brief written progress reports. The financial personnel must make certain that report formats allow them to show cost and technical progress relationships as well as cost and time relationships. When large numbers of people are both conducting regular operations and introducing new equipment, arrangements, and operations, some confusion is inevitable. The object of the control system is to minimize this confusion and the associated delays and costs.

Training

A well designed system, if not operated and used properly could fail. Training the users is important, as if not done well it could prevent the successful implementation of an information system.

Throughout the systems development life cycle the user has been involved. By this stage the analyst should posses an accurate idea of the users that need to be trained. They must know what their roles will be, how they can use the system and what the system will do and will not do.

Both system operators and users need training. During their training, they need to be given a trouble shooting list that identifies possible problems and identifies – remedies for the problem. They should be advised of the common mal functions that may arise and how to solve them.

The training should cover :

- familiarization with the processing system itself (that is the equipment used for data entry or processing)
- Training in using the application i.e. the software
- Good documentation is essential, but this cannot replace training.

There is no substitute for hands on operation of the system while learning its use.

Train the operating personal

A program should be developed to impress upon management and support personnel the nature and goals of the MIS and to train operating personnel in their new duties. In the case of management, many of whom participate in the development of the system, two short seminars are usually adequate. If the first meeting is held at a time when the detailed design is well along, some valuable proposals may be offered that can then be incorporated in the design. Another meeting near the end of the implementation stage may review the benefits of the system and the roles of the executives.

Particular attention should be paid to the training of first-line supervisors. They must have a thorough understanding of what the new MIS is like and what it is supposed to do. Because, in essence, they oversee the operation of the system, they must learn how it will operate. They are faced with many changes in their work and they must obtain acceptance of changes by their subordinates. Supervisors will therefore have an intense interest in the answers to

1. What new skills must we and our people learn?
2. How many people do we gain or lose?
3. What changes in procedures do we make?
4. What are the new forms? Are there more or fewer?
5. What jobs will be upgraded or downgraded?
6. How will our performance be measured?

Certain professional support personnel — such as computer center personnel, marketing researchers, production planners, and accounting personnel who pro-vide input to the MIS or are concerned with processing data and information — should also attend one or several orientation meetings. Because these people will be working with only a small part of the MIS, the seminars should be designed to provide them with an understanding of the complete system. This will furnish direction for their own jobs and give them a perspective that may reduce the likelihood of blunders.

Finally, longer and more formal training programs should be established for people who perform the daily operational tasks of the MIS. These are the clerks, the computer operators, the input and output machine operators, file mainte-nance personnel, and possibly printing production and graphic arts personnel.

In most medium and large companies, a training specialist arranges such programs. The specialist schedules classes, arranges for facilities, and assists the technical people (in this case, the systems analysts) in developing course content and notes for distribution. In small companies, the MIS manager will probably have to develop the training program.

Conversion

Conversion is the process of changing from the old information system to the new or modified one. Conversions should be accomplished quickly as delays and long conversion periods cause frustration and the tasks of all involved including the analyst and user becomes more difficult.

There are four methods available for conversion. There is no single best way to proceed with conversion. Each method should be considered in light of the opportunities that it offers and problems that it may cause. Some situations dictate the use of one method over others, even though other methods maybe more beneficial. Each of the four conversion methods are discussed briefly below :

Parallel Conversion

As the name implies, this refers to running the old system and the new system at the same time in parallel. This approach is most frequently used. This is the most secure method of converting from an old system to a new or modified one.

Both systems are run simultaneously for a specific period of time. When the new system is proven to be functioning as it should, then the old one is stopped. This method is best used when a computerized system replaces a manual one.

Advantages of this method are :

- Offers greatest security. In the case of problems or errors in the new system, then the old system is there as backup.
- Users are more at ease as they do not have to make an abrupt change to the new system.

Disadvantages of this method are :

- Doubles the operating costs
- Burdens employees involved with double workload
- In case the old system is not manual, then it is difficult to make comparisons with old and new outputs.
- Supposedly the new in an improvement on the old, therefore the outputs should differ.

- Employees faced with the choice between the two may opt for the old as they are more familiar with it.

Direct Cutover

Direct Cutover means that on a specified date, the old system is dropped and the new system is put into use. The organization now relies fully on the new system. For direct cutover also known as direct changeover to be successful, extensive testing is to be carried out beforehand. Direct cutover is best used in cases where some delays in processing can be tolerated.

Advantages of this method are :-

- Forces users to make the new system work
- There are immediate benefits from new methods and controls.

Disadvantages in this method are :

- There is no other system to fall back on incase of serious problems or difficulties in the new system.
- Requires most careful planning.
- Users may resent being forced into using an unfamiliar system without recourse.
- There is no adequate way to compare new results with the old.

❖ Pilot System

In this method a working version of the system is implemented in one part of the organization for example a department. The users in this area typically know that they are piloting a new system and that changes can be made to improve the system.

Once the required changes are carried out and the system is complete, then it is implemented throughout the organization either all at once or phase by phase. Pilot approach is best used when new systems involve new techniques or drastic changes in the organization.

Advantages of this method are :

- Provides experience to the users and operators
- Provides live test data before implementation

Disadvantages in this method are :

Incase implementation is not handled properly then users may develop the impression that the system is not error free and may think it unreliable.

❖ Phase – In - Method

The Phase – in – method is used when it is not possible to install a new system throughout the organization all at once. Only one phase of the system is implemented at a time. The file conversions, training of personnel, or arrival of

equipment may not take place all at once. This may force the staging of implementation over a period of time. This could be weeks or months. Some users may start taking advantage of the new system earlier.

Advantages of this method are :

- Allows some users to take advantage of the system early
- Allow training and installation without unnecessary use of resources.

Disadvantages in this method are :

- A long phase-in causes user problems whether the project goes well or not.

Conversion Plan

This plan should be formulated in consultation with the users. The conversion plan includes a description of all activities that must occur to implement the new system and put it into operation. This includes identifications of persons responsible and timetable for each activity that is to be carried out.

During the planning of conversion, the analyst should form a list containing all tasks, including the following :-

1. List all files for conversion
2. Identify all data required to build new files during conversion
3. List all new documents and procedures that go into use during conversion
4. Identify all controls to be used during conversion. Establish procedures for cross – checking the old and the new systems.
5. Determine how team members will know if so mething has not been completed properly.
6. Assign responsibility for each activity.
7. Verify conversion schedules.

The conversion plan should anticipate possible problems and ways to deal with them.

Systems Maintenance

A system should be created whose design is comprehensive and farsighted enough to serve current and projected user needs for several years to come. Part of the analyst's expertise should be in projecting what those needs might be, and then building flexibility and adaptability into the system. The better the system design, the easier it will be to maintain and the maintenance cost will be low. Reducing the maintenance costs is a major concern, since software maintenance can prove to be

very expensive. It is important to detect software design errors early on, as it is less costly than if errors remain unnoticed until maintenance is necessary.

Maintenance is performed most often to improve the existing software rather than to respond to a crisis or system failure. As user requirements change, software and documentation should be changed as part of the maintenance work. Maintenance is also done to update software in response to the change made in an organization. This work is not as substantial as enhancing the software, but it must be done. The system could fail if the system is not properly maintained.

Documentation

Documentation or Procedure Manuals explains how the system is designed and operates. Access to procedure manuals is necessary for new people learning the system, as well as a reminder to those who use the program infrequently. They may contain background comments, steps to accomplish different processes, instructions on how to recover from problems, and what to do next if something isn't working (trouble shooting).

To be useful, manuals must be up to date. A good manual will be used repeatedly as a reference. As such, it needs to be organized in a logical way with careful thought given to the circumstances that would call forth use of the manual.

In general a good manual should be :

- Well organized
- Should be easy to locate needed information
- All cases should be included
- Manuals should be written in plain English.
- Besides the manual's organization and clarity, careful thought should be given to the kinds of people who will be using the manual.

Documentation is to be done at various stages of the SDLC.

1. Feasibility Report

This report is the output of the feasibility study which is carried out at the onset of the system. It tells us that the system requested is feasible or not. The major purposes of this report are :

1. Identify the responsible users and develop an initial "scope" of the system.
2. Identify current deficiencies in the user's environment
3. Establish goals and objectives for the new system.

4. Determine whether it is feasible to automate the system and, if so, suggest some acceptable scenarios.

This report contains a brief description of the current system and an outline of the proposed system.

2. Functional Specifications

This is the output of the requirements analysis phase of systems development life cycle. The feasibility study forms the basis of this report. This document contains the following detail :

Describes 'what' system should do

It contains :

- Detailed DFDs : All levels of Physical and logical DFDs
- Data dictionary : detailing the data in the system
- Input formats
  - Copy of the input documents and screen output
  - Specifications such as general format of all the screens
  - Source of Data – Which specifies type of input form needed by the system
  - Available From – Gives sources from where inputs are to be collected
- Output specifications
  - List of reports required and their details such as :
  - Report name and object of report
  - Frequency – how often it is used
  - Period – period for which information is required
  - Due Date – when the report is required
  - Prescribed arrangement – Specification of the order in which information is required
  - Unit – Specifies units of columns of the report
  - Distribution – Distribution of the report to various department persons.
- Process specification – Decision trees, structured English, and decision tables that are used to describe the logic used in the processes.

3. System Design Note

This should give the details of the design of the system. The topics that should be covered in this document are :

- Scope of the system
  - Systems limitations, systems objectives, major functions and constraints.
- Structure charts
- Program specifications – a short description of what the program does, what are inputs to the program, outputs of the program, calculations if any, program code listings, the program codes could contain comments to explain complex sections of the code.
- Input layouts
- Output layouts
- Data dictionary

4. User Manual

This is a very important document. If the user / operator does not use the system in the proper manner, then the system could fail. This should detail out the procedural steps to be followed right from the start to the finish. It should also tell the users the difficulties that could crop up and how to overcome them.

The back-up procedures should also be included so as no valuable data is lost. All screens should be included and should contain details do what is to be entered at each stage.

The user manual should generally contain the following :

- Introduction
  - What the system does
  - System functions
  - Users of the system
- System developers
- System and configuration required
  - limitations
  - size limitations
  - assumptions

- Principles And Procedures
  - General
  - Outputs
  - Inputs
    - General procedures
    - Start up / sign on
    - Backup
    - Shutdown
    - Formatting disks
- Tutorial -- step-by-step walkthrough of example functions to be illustrated in the example
- Reference
  - For each function
    - Function description
    - How and when used
    - Structure of command or screen
    - What happens
    - How to use
    - Errors and what to do
    - Examples
- For each error
  - How recognized
  - Meaning
  - What to do about it
- Appendix

How to install the system before first use.

Structured Walkthrough / Formal Technical Reviews

A structured walkthrough is a planned review of a system or its software by persons involved in the development effort. The purpose of walkthroughs is to find areas where improvement can be made in the system or the development process. A walkthrough should be viewed by the programmers and analysts as an opportunity to receive assistance. As users and developers are involved in walkthroughs, the concept recognizes that systems development is a team process.

The structured walkthrough should be used throughout the systems development process as a constructive and cost-effective management tool, after the detailed investigation (analysis review) following design (design review), and during program development (code review and testing review).

All walkthroughs includes documentation that participants read and study prior to the actual walkthrough.

- **Analysis Review**

This is conducted to examine the functional specifications of the system, which is prepared after the analysis phase of the SDLC. This walkthrough is aimed at examining the functions, activities, and processes that the new system will handle.

It emphasizes the information and processing requirements the proposed design should handle. If there are inconsistencies among the requirements stated by the users and those the analyst is proposing to meet through the new system or if there are vague specifications, the walkthrough should uncover them so they can be dealt with.

- **Design Review**

Design reviews, as the name suggests, focus on design specification for meeting previously identified systems requirements. The information supplied about the design prior to the session can be communicated using structured charts, N-S flowcharts, screen designs, input formats, output formats, document layouts.

The purpose of this walkthrough is to determine whether the proposed design will meet the requirements effectively and efficiently. If the participants find discrepancies between the design and requirements, they will point out and discuss them.

- **Code Review**

A code review is a structured walkthrough conducted to examine the program code developed in a system along with its documentation. Is is used for new systems and systems under maintenance. This does not deal with an entire software system, but rather with individual modules or major components in a program.

- **Post-implementation Revierw**

After the system is implemented and conversion is complete, a review of the system is usually conducted by users and analysts alike. Not only is this a normal practice, but it should be a formal process to determine how well the system is working, how it has been accepted, and whether adjustments are needed.

The review is also important to gather information for the maintenance of the system. Since no system is really ever complete, it will be maintained as changes are required because of internal developments, such as new legal requirements, industry standards, or competition. The post-implementation review provides the first source of information for maintenance requirements.

Questions

Very Short Questions :

1. What is Iode Testing ?
2. What is verification ?
3. What is velidation ?
4. What is parallel conversion method ?

Short Questions :

1. What is testing ?
2. What is the difference between Live and Artifical data ?
3. What is the diffence between veritication and validations ?
4. Explain forma Technical Reviews in brief ?
5. Brief the STLC ?

Long Question :

1. What is the Testing ? Discuss all type of testing in details.
2. What do you mean by Maintenance ?
3. Short note on :
   (a) Varification and Validation
   (b) Documentation
   (c) Testing
   (d) Maintenance
   (e) Implimentation
   (f) Conversion