BCA515: System Design and Analysis

Unit-5

System Implementation:

1. Need of Testing,
2. Test Plan,
3. Quality Assurance,
4. Trends in Testing,
5. Audit Trails,
6. Post Implementation Review.

Security and Recovery in System Development:

1. System Security,
2. Threats to System Security,
3. Control Measures,

Disaster/Recovery Plannings:

1. Ethics in System Development.

SYSTEM IMPLEMENTATION:

No program or system design is perfect; communication between the user and the designer is not always complete or clear, and time is usually short. The result is errors and more errors. The number and nature of errors in a new design depend on several factors:

1. Communications between the user and the designer.

2. The programmer's ability to generate a code that reflect exactly the system specifications.

3. The time frame for the design.

Theoretically, a newly designed system should have all the pieces in working order, but in reality, each piece work independently. Now is the time to put all the pieces into one system and test it to determine whether it meets the user's requirements. This is the last chance to detect and correct errors before the system is installed for user acceptance testing. The purpose of system testing is to consider all the likely variations to which it will be subjected and then push the system to its limits. It is a tedious but necessary step in system development.

The process of system testing and the steps taken to validate and prepare a system for final implementation. Following are the basic terms:

1. **Unit testing** is testing changes made in an existing or a new program.

2. **Sequential or series testing** is checking the logic of one or more programs in a candidate system, where the output of one program will affect the processing done by another program.

3. **System testing** is executing a program to check logic changes made in it and with the intention of finding errors - making the program fail. Effective testing does not guarantee reliability. Reliability is a design consideration.

4. **Positive testing** is making sure that the new programs do in fact process certain transactions according to specifications.

5. **Acceptance testing** is running the system with live data by the actual user.

I) SYSTEM TESTING (NEED)

Testing is vital to the success of the system. System testing makes a logical assumption that if all the parts of the system are correct, the goal will be successfully achieved. Inadequate testing or non-testing leads to errors that may not appear until months later. This creates to problems:

1. The time lag between the cause and the appearance of the problem (the longer the time interval, the more complicated the problem has become), and

2. The effect of system errors on files and records within the system. A small system errors can conceivably explode into a much larger problem. Effective testing early in the process translates directly into long-term costs savings from a reduced number of errors.

Another reason for system testing is its utility as a user-oriented vehicle before implementation. The best program is worthless if it does not meet user needs. Unfortunately, the user's demands are often compromised by efforts to facilitate program or design efficiency in terms of processing time for memory utilisation. Often the computer technician and the user have communication barriers due to different backgrounds, interests, priorities, and perhaps languages. The system tester (designer, programmer, or user) who has developed some computer mastery can bridge this barrier.

WHAT DO WE TEST FOR?

The first test of a system is to see whether it produces the correct outputs. No other test can be more crucial. Following this step, a variety of other tests are conducted:

1. *Online response*: Online systems must have a response time that will not cause a hardship to the user. One way to test this is to input transactions on as many CRT screens as would normally be used in peak hours and time the response to each online function to establish a true performance level.

2. *Volume*: In this test, we create as many records as would normally be produced to verify that the hardware and software will function correctly. The user is usually asked to provide test data for volume testing.

3. *Stress testing*: The purpose of stress testing is to prove that the candidate system does not malfunction under peak loads. Unlike volume testing, where time is not a factor, we subject the system to a high volume of data over a short time period. This simulates an online environment where a high volume of activities or cause in spurts.

4. *Recovery and security*: A forced system failure is induced to test a backup recovery procedure for file integrity. Inaccurate data entered to see how the system responds in terms of error detection and protection. Related to file integrity is a test to demonstrate that data and programs are secure from unauthorized access.

5. *Usability documentation and procedure*: The usability test verifies the user-friendly nature of the system. This relates to normal operating and error-handling procedures, for example. One aspect of user-friendliness is accurate and complete documentation for the step the user is asked to use only the documentation and procedures as a guide to determine whether the system can be run smoothly.

II) THE TEST PLAN

The first step in system testing is to prepare a plan that will test all aspects of the system in a way that promotes its credibility among potential uses. There is psychology in testing:

1. Programmers usually do a better job in unit testing because they are expected to document and report on the method and extent of their testing.

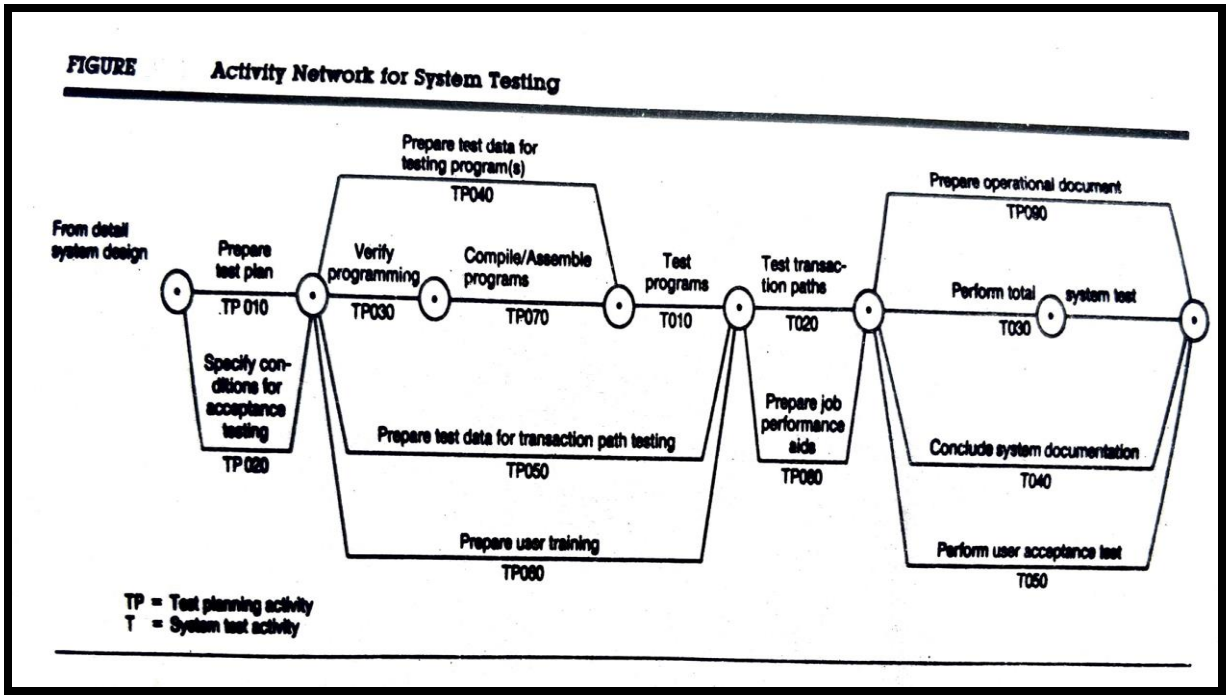
2. Uses are involved, which means communication is improved between users and the designer group.

3. Programmers are involved when they become aware of user problems and expectations. The user also becomes more aware (and appreciative) of the complexity of programming and testing. The upshot of all this is a more reliable realistic and cooperative user for successful testing.

Activity Network for System Testing

A test plan entails the following activities (see Figure):

1. Prepare test plan.
2. Specify conditions for user acceptance testing.
3. Prepare test data for program testing.
4. Prepare test data for transaction path testing.
5. Plan user training.
6. Compile/assemble programs.
7. Prepare job performance aids.
8. Prepare operational documents.



1. Prepare Test Plan

A workable test plan must be prepared in accordance with established design specifications. It includes the following items:

1. Outputs expected from the system.
2. Criteria for evaluating outputs.
3. A volume of test data.
4. Procedure for using test data.
5. Personnel and training requirements.

2. Specify conditions for user acceptance testing

Planning for user acceptance testing calls for the analyst and the user to agree on the conditions for the test. Many of these conditions may be derived from the test plan. Others are an agreement on the test schedule, the test duration, and the persons designated for the test. The start and termination dates for the test should also be verified in advance.

3. Prepare test data for program testing

As each program is coded, test data are prepared and documented to ensure that all aspects of the program are properly tested. After the testing, the data are field for future reference.

4. Prepare test data for transaction path testing

This activity develops the data required for testing every condition and transaction to be introduced into the system. The path of each transaction from origin to destination is carefully tested for reliable results. The test verifies that the test data virtually comparable to live data used after conversion.

5. Plan user testing

User training is designed to prepare the user for testing and converting the system. User involvement and training take place parallel with the programming for three reasons:
1. The system group has time available to spend on training while the programs are being written.
2. Initiating a user-training program gives the systems group clearer image of the user's interest in the new system.
3. A trained user participates more effectively in system testing.

For a user training, preparation of a checklist is useful. Included are provisions for developing training materials and other documents to complete the training activity. In effect, the checklist calls for a commitment of personnel, facilities, and efforts for implementing the candidate system.
The training plan is followed by preparation of the user training manual and other text materials. Facility requirements and the necessary hardware are specified and documented. A common procedure is to train supervisors and department heads who, in turn, train their staff as they see fit. The reasons are obvious:

1. User supervisors are knowledgeable about the capabilities of their staff and the overall operation.
2. Staff members usually respond more favourably and accept instructions better from supervisors than from outsiders.

3. Familiarity of users with their particular problems (bugs) makes them better candidates for handling user training then the systems analyst. The analyst gets feedback to ensure that proper training is provided.

FIGURE 12-2 User Training Checklist

Company _____ Analyst _____

Project Name _____ Date _____

Activity	Start Date	Completion Date	Staff in Charge	Department in Charge	Comments
1. Notification					
Announcement to officers	10/06	10/20	P. Solen	Personnel Mgr.	
Announcement to employees	10/06	10/20	J. Hill	Auditing	
Coordinated customer activities	10/06	10/29	D. Stang	Cashier	
Coordinate computer service	10/06	10/29	C. Sibley	Sr. Vice Pres.	
2. Procedures					
Interdepartmental	10/14	11/01	A. Blake	Systems	
Interdepartmental	10/14	11/01	J. Hill	Auditing	
3. Forms					
Design	11/01	11/14	A. Blake	Systems	
Printing	11/01	11/20	A. Blake	Systems	
4. Equipment					
Terminals	11/01	12/15			
5. Training and orientation					
Manuals	12/01	21/16	A. Blake	Systems	
Training aids	12/01	12/16	A. Blake	Systems	
Special workshops	12/10	12/14	A. Blake	Systems	
6. Lobby layout	12/10	12/30	D. Stang	President	
7. Supplies	12/10	12/15	M. Steed	Purchasing Agent	
8. Personnel					
Transfers	12/10	12/12	P. Solen	Personnel Mgr.	
New hires	12/15	12/30	P. Solen	Personnel Mgr.	

Approved by: _____ _____

(Project leader) (Systems Department)

6. Compile/Assemble Programs

All programs have to be compiled/assembled for testing. Before this, however, a complete program description should be available. Included is the purpose of the program, its use, the programmer(s) who prepared it, and the amount of the computer time it takes to run it. Program and system flow charts of the project should also be available for reference.

In addition to these activities,

desk checking the source code uncovers programming errors or inconsistencies. Before actual program testing, a run order schedule and test scheme are finalized.

A ***run order schedule*** specifies the transactions to the test and the order in which they should be tested. High priority transactions that make special demands on the candidate system are tested first.

In contrast, a ***test scheme*** specifies how program software should be debugged.

A common approach, called ***bottom-up programming,*** tests small-scale program modules, which are linked to a higher-level-module, and so on until the program is completed.

An alternative is the ***top-down*** approach, where the general program is tested first, followed by the addition of program modules, one level at a time to the lowest level.

7. **Prepare Job Performance Aids**

   In this activity the materials to be used by personnel to run the system are specified and scheduled. This includes a display of materials such as program codes, a list of input codes attached to the CRT terminal, and a posted instruction scheduled to load the disk drive. These aids reduce the training time and employ personnel at lower positions.

8. **Prepare Operational Documents**

   During the test plan stage, all operational documents are finalized including copies of the operational formats required by the candidate system. Related to operational documentation is a section on the experience, training, and educational qualifications of personnel for the proper operation of the new system.

SYSTEM TESTING

The purpose of system testing is to identify and correct errors in the candidate system. As important as this phase is, it is on that is frequently compromised. Typically, the project is behind schedule or the user is eager to go directly to conversion.

In system testing, performance and accepted standards are developed. Substandard performance or service interruptions that result in system failure are checked during the test. The following performance criteria are used for system testing:

1. **Turnaround time** is the elapsed time between the receipt of the input and the availability of the output. In online systems, higher priority processing is handled during peak hours, while low-priority processing is done later in the day or during the night shift. The objective is to decide on and evaluate all the factors that might have a bearing on the turnaround time for handling on applications.
2. **Backup** relates to procedures to be used when the system is down. Backup plan might call for the use of another computer. The software for the candidate system must be tested for compatibility with a backup computer.

   In case of a partial system breakdown, provisions must be made for dynamic reconfiguration of the system. For example, in an online environment, when the printer breaks down, a provisional plan might call for automatically “dumping” the output on tape until the service is restored.
3. **File protection** pertains to storing files in a separate area of protection against fire, flood, or natural disaster. Plans should also be established for reconstructing files damaged through a hardware malfunction.
4. **The human factor** applies to the personnel of the candidate system. During system testing, lighting, air conditioning, noise, and other environmental factors are evaluated with people’s desks, chairs, CRTs, etc. Hardware should be designed to match human comfort. This is referred to as ergonomics. It is becoming an extremely important issue in system development.

Types of System Tests

After test plan has been developed, system testing begins by testing program modules separately, followed by testing "bundled" models as a unit. A program module may function perfectly in isolation but fail when interfaced with other modules. The approach is to test each entity with successively larger ones, up to the system test level.

System testing consists of the following steps:

1. Program(s) testing.
2. String testing.
3. System testing.
4. System documentation.
5. User acceptance testing.

Each step is briefly explained here.

Program Testing

A program represents the logical elements of a system. For a program to run satisfactory, it must compile and test data correctly and tie in properly with other programs. Achieving an error free

program is the responsibility of the programmer. Program testing checks for two types of error: syntax and logic.

A **syntax error** is a program is statement that violates one or more rules of the language in which it is written. An improperly defined field dimensions or omitted key words are common syntax errors. These errors are shown through error messages generated by the computer.

A **logic error**, on the other hand, deals with incorrect data fields, out-of-range items, and invalid combinations. Since diagnostics do not detect logic errors, the programmer must examine the output carefully for them.

When a program is tested, the actual output is compared with the expected output. When there is a discrepancy, the sequence of instructions must be traced to determine the problem. The process is facilitated by breaking the program down into self-contained portions, each of which can be checked at certain key points. The idea is to compare program values against desk-calculated values to isolate the problem.

String Testing

Programs are invariably related to one another and interact in a total system. Each program is tested to see whether it conforms to related programs in the system. Each portion of the system is tested against the entire module with both test and live data before the entire system is ready to be tested.

System Testing

System testing is designed to uncover weakness that were not found in earlier tests. This includes forced system failure and validation of the total system as it will be implemented by its users in the operational environment. Generally, it begins with low volumes of transactions based on live data. The volume is increased until the maximum level for each transactions type is reached. The total system is also tested for recovery and fall back after various major failures to ensure that no data are lost during the emergency. All this is done with the old system is still in operation. After the candidate system passes the test, the old system is discontinued.

System Documentation

All design and test documentation should be finalized and entered in the library for future reference. The library is the central location for maintenance of the new system. The format, organisation, and language of each documentation should be in line with system with standards.

User Acceptance Testing

An acceptance test has the objective of selling the user on the validity and reliability of the system. It verifies that the systems procedures operate to system specifications and that the integrity of vital data is maintained. Performance of an acceptance test is actually the users show. User motivation and knowledge are critical for successful performance of the system. Then a comprehensive test report is prepared. The report indicates the system's tolerance, performance range, error rate, and accuracy.

III) QUALITY ASSURANCE

The amount and complexity of software produced today stagger the imagination. Software development strategies have not kept pace, however, and software products fall short of meeting application objectives. Consequently, controls must be developed to ensure a quality products. Basically, quality assurance defines the objectives of the project and reviews the overall activities so that errors are corrected early in the development process. Steps are taken in each phase to ensure that there are no errors in the final software.

Quality Assurance Goals in the Systems Life Cycle

The software life cycle includes various stages of development, and each stage has the goal of quality assurance. The goals and their relevance to the quality assurance of the system are summarised next.

Quality Factors Specifications

The goal of this stage is to define the factors that contribute to the quality of the candidate system. Several factors determine the quality of a system:

1. *Correctness*-
The extent to which a program meets system specifications and user objectives.
2. *Reliability*-
The degree to which the system performs its intended functions over a time.
3. *Efficiency*-
The amount of computer resources required by a program to perform a function.
4. *Usability*-
The effort required to learn and operate a system.
5. *Maintainability*-
The ease with which program errors are located and connected.
6. *Testability*-
The effort required to test a program to ensure its correct performance.
7. *Portability*-
The ease of transporting a program from one hardware configuration to another.
8. *Accuracy*-
the required precision in input editing, computations, and output.
9. *Error tolerance*-
Error detection and correction versus error avoidance.
10. *Expandability*-
Ease of adding or expanding the existing database.
11. *Access control and Audit*-
Control of access to the system and the extent to which that access can be audited.
12. *Communicativeness*-
How descriptive or useful the inputs and outputs of the system are.

Software Requirements Specifications

The quality assurance goal of this stage is to generate the requirements document that provides the technical specification for the design and development of the software. This document enhances the systems quality by formalizing communication between the system developer and the user and provides the proper information for accurate documentation.

Software Design Specifications

The software design document defines the overall architecture of the software that provides the functions and features described in the software requirements document. In addresses the question; How it will be done? The document describes the logical subsystems and their respective physical modules. It ensures that all conditions are covered.

Software Testing and Implementation

The quality assurance goal of the testing phase is to ensure that completeness and accuracy of the system and minimize the retesting process. In the implementation phase, the goal is to provide a logical order for the creation of the modules and, in turn, the creation of the system.

Maintenance and Support

This phase provides the necessary software adjustment for the system to continue to comply with the original specifications. The quality assurance goal is to develop a procedure for correcting errors and enhancing software. This procedure improves quality assurance by encouraging complete reporting and logging of problems, ensuring that reported problems are promptly forwarded to the appropriate group for resolution, and reducing redundant effort by making known problem reports available to any department that handles complaints.

Levels of Quality Assurance

There are three levels of quality assurance: testing, validation, and certification.

In system testing, the common view is to eliminate program errors. This is extremely difficult and time-consuming, since designers cannot prove 100% accuracy. Therefore, all that can be done is to put the system through a "fail test “cycle- determine what will make it fail. A successful test, then is one that finds errors. The test strategies discussed earlier are used in system testing.

System validation checks the quality of the software in both simulated and live environments. First the software goes through a phase (often referred to as Alpha testing) in which errors and failures based on simulated user requirements are verified and studied. The modified software is then subjected to phase two (called beta testing) in the actual user site or a live environment. The system is used regularly with live transactions. After a scheduled time, failures and errors are documented and final correction and enhancements are made before the package is released for use.

The third level of System validation checks the quality of the software in both simulated and live environments. First the software goes through a phase (often referred to as Alpha testing) in which errors and failures based on simulated user requirements are verified and studied. The modified software is then subjected to phase two (called beta testing) in the actual user site or a live environment. The system is used regularly with live transactions. After a scheduled time, failures and errors are documented and final correction and enhancements are made before the package is released for use.

The third level of quality assurance is to certify that the program or software package is current and conforms to standards. With a growing trend towards purchasing ready to use software, certification has become more important. A package that is certified goes through a team of specialists who test, review, and determine how well it meets the vendors claims. Certifications is actually issued after the package passes the test. Certification, however, does not assure the user that it is the best package to adapt; it only attests that it will perform what the vendor claims.

In summary, the quality of an information system depends on its design, testing, and implementation. One aspect of system quality is its reliability or the assurance that it does not produce costly failures the strategy. The strategy of error tolerance (detection and correction) rather than error avoidance is the basis for successful testing and quality assurance.

IV) TRENDS IN TESTING

In the future, we can except unparalleled growth in the development and use of automated tools and software aids for testing. One such tool is the functional tester, which determines whether determines whether the hardware is operating up to a minimal standard. It is a computer program that controls the complete hardware configuration and verifies that is functional. For example, it can test computer memory by performing read/write tests, and it tests each peripheral device individually.

The functional testing is of great Value when minute hardware problems are disguised as software bugs. For example, hardware faults are usually repeatable, whereas software bugs are generally erratic. Problems arise when the delicate interaction between hardware and software causes the hardware problem to appear as an electric software bug. A functional tester determines immediately that the problem is in the hardware. This saves considerable time during testing.

Another software aid is the debug monitor. It is a computer program that regulates and modifies the applications software that is being tested. It can also control the execution of functional tests and automatically patch or modify the application program being tested.

The use of these tools will increase as systems grow in size and complexity and as the verification and insurance of reliable software become increasingly important.

ROLE OF THE DATA PROCESSING AUDITOR

The planned test of any system ought to include a thorough auditing technique and introduce control elements unique to the system. The data processing (DP) auditor should be involved in most phases of the system life cycle, especially system testing. In the past, auditors have audited systems after they have been installed. Then the cost is often too prohibitive to go back and modify the system to incorporate adequate control. Therefore, audit controls must be built into the system design and tested with the cooperation of both the analyst and the DP auditor. The auditor's role is to judge the control and make recommendations to the system team in the charge of the project. The user department should participate in reviewing the control specifications for the system to ensure that adequate control has been provided.

For testing programs, test data must include transactions that are specifically designed to violate control procedure incorporated in the program as well as valid transactions to test their acceptance by the system. The control setup must be carefully examined. At the time programs are tested, all required files are accumulated and set out in the proper order and format for final testing.

V) THE AUDIT TRAIL

An important function of control is to provide the audit trail. In designing tests, the auditor is concerned about the changing nature of the audit trail. In an online environment, the form, content, and accessibility of records are such that the auditor often has difficulty following a single transaction completely through the system. Some of the following changes in the audit trail confront the auditor:

1. Source documents are no longer used by the system after they are transcribed onto a machine-readable medium. They are often filled in the areas or ways that make later access difficult. In direct data entry, traditional source documents are simply eliminated.

2. Files stored on tape or disk can be read only by a computer, which limits the auditing function. A data dump is possible, though, to compare the data against a data map.

3. Processing activities are difficult to observe, since most of them are within the system. It is possible, however, to get a trace when required.

One way to maintain a viable audit trail is to keep a detailed file of the transactions as they occur. The file can then be the input for an audit program that extracts the transactions for selected accounts and prints them so that the auditor can trace the status of an account in detail.

Given the important role of the auditor, he/ she is expected to be part of the system development team, which includes the user. As an independent advisor, the role is to judge the controls and make recommendations to the team. Three important steps are considered in the evaluations:

1. Define the control objectives as separate design and test requirements. Input preparation and transmission by the user are important control areas that are viewed with an emphasis on audit trails, error-correction procedures, and adequate documentation during testing. These areas should always be present and well documented.

2. Re-examine budget costs to see whether system testing is within the limits.

3. Review specifications. The auditor should evaluate program acceptance test specifications and assist the system programmer in developing test standards, various levels of testing, and actual test conditions. He/she should also evaluate the actual system acceptance test to ensure an acceptable level of confidence and reliability.

In summary, it is the auditor's responsibility to build controls into candidate systems to ensure integrity reliability, and confidence of the users at all levels. The auditor should be called in during design as well as testing so that any suggestions he/she has can be considered before implementation, when changes are less costly. Including the auditor in the system development team makes it easier for him/ her to monitor testing procedures and consider the acceptance of new controls to replace those changes by the new design.

VI) POST IMPLEMENTATION REVIEW

Operational systems are quickly taken for granted. Every system requires periodic evolution after implementation. A post-implementation review measures the system performance against predefined requirements. Unlike system testing, which determines where the system fails so that the necessary adjustments can be made, a post implementation review determines how will the system continues to meet performance specifications. It is after the fact- after design and conversion are complete. It also provides information to determine whether major redesign is necessary.

A post implementation review is an evolution of a system in terms of the extent to which the system accomplishes stated objectives and actual project costs exceed initial estimates. It is usually a review of major problems that need converting and those that surfaced during the implementation phase. The primary responsibility for initiating the review lies with the user organisation, which assigns special staff for this purpose.

Request for review

The initiating study begins with the review team, which gathers and reviews requests for evaluation. It also files discrepancy notices after the system has been accepted. Unexpected change in the system that affects the user or system performance is a primary factor that prompts system review. Once a request is field, the user is asked how well the system is functioning to specifications or how well the measured benefits have been realised. Suggestions regarding changes and improvements are also sought. This phase sets the stage for a formal post-implementation review.

A Review Plan

The review team prepares a formal review plan round objectives off the review the type of evolution to be carried out, and the time schedule required. An overall plan covers the following areas (see figure):

1. *Administrative plan* -

Review area objectives, operating costs, actual operating performance, and benefits.

2. *Personnel requirements plan*-

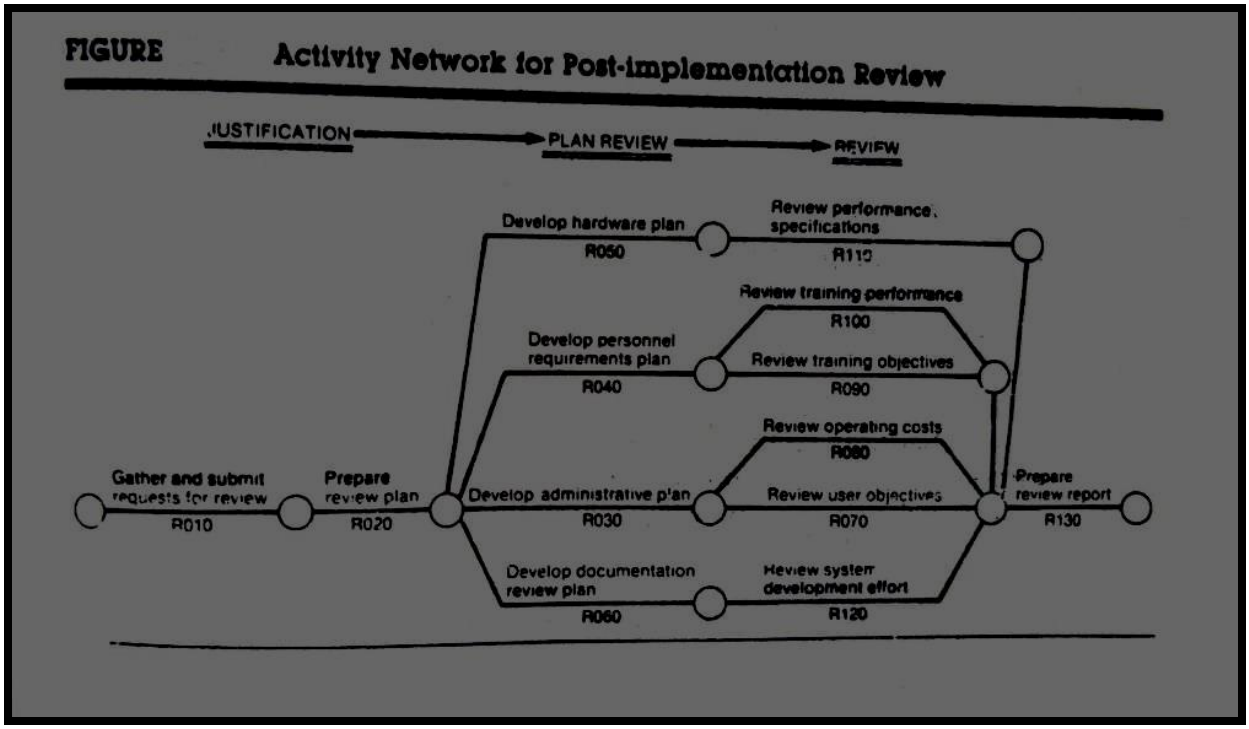
Review performance objectives and training performance to date.

3. *Hardware plan*- Review performance specifications.

4. *Documentation review plan*-

Review the system development effort.

Once drafted, the review should be verified and approved by the requester or the end-user.



Administrative Plan

The review group probes the fact of the operational system on the the administrative procedures of the user. The following activities are reviewed:

1. User objectives: this is an extremely critical areas since it is possible that over time either the system fails to meet the user's initial objectives for the user objectives change as a reflection of the changes in the organisational objectives. We need to think in terms of problems and off further opportunities. The result of the evolution are documented for future reference.

2. Operating cost and benefits: under the administrative plan, the cost structure of the system is closely reviewed. This includes a review of all cost and savings, a review and update of the non cost benefits of the system, and a current budget designed to manipulate the costs and savings of the system.

Personnel Requirement Plan

This plan evaluates all activities involving system personnel and staff as they deal directly with the system. The emphasis is on productivity, morale, and job satisfaction. After the plan is developed, the review group evaluates the following:

1. Personnel performance objectives compared with the current performance levels:

Turnover, tardiness, and absenteeism are also evaluated. The results are documented and made available to the maintenance group for follow-up.

2. Training performance:
Through testing, interviews, and other data gathering techniques, the review group attempts to answer questions about the adequacy of the training materials.

Hardware Plan

The hardware of the new system is also reviewed, including terminals, CRT screens, software programs, and the communication network. The primary target is a comparison of current performance specifications with design specifications. The outcome of the evaluation indicates any differences between expectations and realized results. It also points to any necessary modifications to be made.

Documentation Review Plan

The reason for developing a documentation review plan is to evaluate the accuracy and completeness of the documentation compiled to date and its conformity with pre-established documentation standards. Irregularities prompt action where changes in documentation would improve the format and content.