CHAPTER -5
ACQUISITION, DEVELOPMENT AND IMPLEMENTATION OF SYSTEMS

5.1. System Development

System development refers to the process of examining a business situation with the intent of improving it through better procedures and methods. System development can generally be thought of as having two major components described briefly as follows:

1) System Analysis is the process of gathering and interpreting facts, diagnosing problems, and using the information to recommend improvements to the system

2) System Design is the process of planning and structuring a new business system or one to replace or complement an existing system.

5.2. Achieving System Development Objectives

Achieving the objectives of the system development is essential but many times, such objectives are not achieved as desired. An analysis on 'why organizations fail to achieve their systems development objectives' reveals bottlenecks. Some of the most notable ones are described briefly as follows:

1) User Related Issues: It refers to those issues where user/customer is reckoned as the primary agent. Some of the aspects with regard to this problem are mentioned as follows:

- **a) Shifting User Needs:** User requirements for IT are constantly changing. As these changes accelerate, there will be more requests for Information systems development and more development projects. When these changes occur during a development process, the development team faces the challenge of developing systems whose very purpose might change since the development process began.

- **b) Resistance to Change:** People have a natural tendency to resist change, and information systems development projects signal changes - often radical - in the workplace. When personnel perceive that the project will result in personnel cutbacks, threatened personnel will dig in their heels, and the development project is doomed to failure.

- **c) Lack of User Participation:** Users must participate in the development efforts to define their requirements, feel ownership for project success, and work to resolve development problems. User participation also helps to reduce user resistance to change.

- **d) Inadequate Testing and User Training:** New systems must be tested before installation to determine that they operate correctly. Users must be trained to effectively utilize the new system.

2) Developer Related Issues: It refers to the issues and challenges with regard to the developers. Some of the critical bottlenecks are mentioned as follows:

- **a) Lack of Standard Project Management and System Development Methodologies:** Some organizations do not formalize their project management and system development methodologies, thereby making it very difficult to consistently complete projects on time or within budget.

- **b) Overworked or Under-Trained Development Staff:** In many cases, system developers often lack sufficient educational background and requisite state of the art skills. Furthermore, many companies do a little to help their development personnel stay technically sound, and more so a training plan and training budget do not exist.

3) Management Related Issues: It refers to the bottlenecks with regard to organizational set up, administrative and overall management to accomplish the system development goals. Some of such bottlenecks are mentioned as follows:

- **a) Lack of Senior Management Support and Involvement:** Developers and users of information systems watch senior management to determine ‘which systems development projects are important’ and act accordingly by shifting their efforts away from any project, which is not receiving management attention. In addition, management can see that adequate resources, as well as budgetary control over use of those resources, are dedicated to the project.

b) Development of Strategic Systems: Because strategic decision making is unstructured, the requirements, specifications, and objectives for such development projects are difficult to define.

4) New Technologies: When an organization tries to create a competitive advantage by applying advance technologies, it generally finds that attaining system development objectives is more difficult because personnel are not as familiar with the technology.

In order to overcome these aforementioned issues, organizations must execute a well-planned systems development process efficiently and effectively. Accordingly, a sound system development team is inevitable.

5.3. Accountants' Involvement in Development Work

Most accountants are uniquely qualified to participate in systems development because they may be among the few people in an organization, who can combine knowledge of IT, business, accounting, and internal control, as well as behavior and communications, to ensure that new systems meet the needs of the user and possess adequate internal controls. They have specialized skills - such as accounting and auditing - that can be applied to the development project. For example, an accountant might perform the analysis of a proposed system's costs and benefits.

An accountant can help in various related aspects during system development; some of them are as follows:

1) Return on Investment (referred as ROI): This defines the return, an entity shall earn on a particular investment i.e. capital expenditure. This financial data is a prime consideration for any capital expenditure entity decides to incur. The important data required for this analysis being the cost of project, the expected revenue/benefit for a given period. The analysis ideally needs to be done before the start of the development efforts for better decision making by management. For this analysis following data needs to be generated.

a) Cost: This includes estimates for typical costs involved in the development, which are given as follows:

i) Development Costs: Development Costs for a computer based information system include costs of the system development process, like salaries of developers.

ii) Operating Costs: Operating Costs of a computer based information system including hardware/software rental or depreciation charges; salaries of computer operators and other data processing personnel, who will operate the new system.

iii) Intangible Costs: Intangible Cost that cannot be easily measured. For example, the development of a new system may disrupt the activities of an organization and cause a loss of employee productivity or morale.

b) Benefits: The benefits, which result from developing new or improved information systems that can be subdivided into tangible and intangible benefits. A post implementation analysis is also done to see how the system development effort has benefitted an organization. For example: A large oil company in public sector, few years back implemented an ERP system at a total cost of ` 100 crores. The calculated benefits from the project were ` 40 crores per annum. Above data gives an actual ROI of 40%, which is tremendous for any business. Same also tells the payback period is around 2.5 years.

2) Computing Cost of IT Implementation and Cost Benefit Analysis: For analysis of ROI, accountants need the costs and returns from the system development efforts. For correct generation of data, proper accounting needs to be done. Accountants shall be the person to whom management shall look for the purpose.

3) Skills expected from an Accountant: An accountant, being an expert in accounting field must possess skills to understand the system development efforts and nuances of the same. S/he is expected to have various key skills, including understanding of the business objectives, expert book keeper, and understanding of system development efforts etc.

CHAPTER -5
ACQUISITION, DEVELOPMENT AND IMPLEMENTATION OF SYSTEMS

5.4. Systems Development Methodology

A system development methodology is a formalized, standardized, well-organized and documented set of activities used to manage a system development project. It refers to the framework that is used to structure, plan and control the process of developing an information system.

1) The project is divided into a number of identifiable processes, and each process has a starting point and an ending point. Each process comprises several activities, one or more deliverables, and several management control points. The division of the project into these small, manageable steps facilitates both project planning and project control.

2) Specific reports and other documentation, called Deliverables must be produced periodically during system development to make development personnel accountable for faithful execution of system development tasks.

3) Users, managers, and auditors are required to participate in the project, which generally provide approvals, often called signoffs, at pre-established management control points. Signoffs signify approval of the development process and the system being developed.

4) The system must be tested thoroughly prior to implementation to ensure that it meets users' needs as well as requisite functionalities.

5) A training plan is developed for those who will operate and use the new system.

6) Formal program change controls are established to preclude unauthorized changes to computer programs.

7) A post-implementation review of all developed systems must be performed to assess the effectiveness and efficiency of the new system and of the development process.

5.5. Waterfall Model

The waterfall approach is a traditional development approach in which each phases is carried in sequence or linear fashion. These phases include requirements analysis, specifications and design requirements, coding, final testing, and release. In this traditional approach of system development, activities are performed in sequence. When the traditional approach is applied, an activity is undertaken only when the prior step is fully completed.

The characterizing features of this model have influenced the development community in big way. Some of the key characteristics are the following:

1) Project is divided into sequential phases, with some overlap and splash back acceptable between phases.

2) Emphasis is on planning, time schedules, target dates, budgets and implementation of an entire system at one time.

3) Tight control is maintained over the life of the project through the use of extensive written documentation, as well as through formal reviews and approval/signoff by the user and information technology management occurring at the end of most phases before beginning the next phase.

4) **Strengths:** The fundamental strength of the waterfall model has made it quite popular and handy among the fraternity. Major strengths are given as follows:
- **a)** It is ideal for supporting less experienced project teams and project managers or project teams, whose composition fluctuates.
- **b)** The orderly sequence of development steps and design reviews help to ensure the quality, reliability, adequacy and maintainability of the developed software.
- **c)** Progress of system development is measurable.
- **d)** It enables to conserve resources.

5) **Weaknesses:** Though it is highly useful model, it suffers from various weaknesses too. Experts and practitioners identify a number of weaknesses including the following:
   - **a)** It is criticized to be Inflexible, slow, costly, and cumbersome due to significant structure and tight controls.
   - **b)** Project progresses forward, with only slight movement backward.
   - **c)** There is a little to iterate, which may be essential in situations.
   - **d)** It depends upon early identification and specification of requirements, even if the users may not be able to clearly define 'what they need early in the project'.
   - **e)** Requirement inconsistencies, missing system components and unexpected development needs are often discovered during design and coding.
   - **f)** Problems are often not discovered until system testing.
   - **g)** System performance cannot be tested until the system is almost fully coded, and under capacity may be difficult to correct.
   - **h)** It is difficult to respond to changes, which may occur later in the life cycle, and if undertaken it proves costly and are thus discouraged.
   - **i)** It leads to excessive documentation, whose updation to assure integrity is an uphill task and often time-consuming.
   - **j)** Written specifications ate often difficult for users to read and thoroughly appreciate.
   - **k)** It promotes the gap between users and developers with clear vision of responsibility.

5.6. The Prototyping Model

A prototype is a usable system or system component that is built quickly and at a lesser cost, and with the intention of modifying/replicating/expanding or even replacing it by a full-scale and fully operational system. As users work with the prototype, they learn about the system criticalities and make suggestions about the ways to manage it. These suggestions are then incorporated to improve the prototype, which is also used and evaluated Finally, when a prototype is developed that satisfies all user requirements, either it is refined and turned into the final system or it is scrapped. If it is scrapped, the knowledge gained from building the prototype is used to develop the real system. Prototyping can be viewed as a series of four steps.

The generic phases of this model are explained as follows:

1) **Identify Information System Requirements:** In traditional approach, the system requirements are to be identified before the development process starts. However, under prototype approach, the design team needs only fundamental system requirements to build the initial prototype, the process of determining them can be less formal and time- consuming than when performing traditional systems analysis.

2) **Develop the Initial Prototype:** The designers create an initial base model and give little or no consideration to internal controls, but instead emphasize system characteristics such as simplicity, flexibility, and ease of use. These characteristics enable users to interact with tentative versions of data entry display screens, menus, input prompts, and source documents. The users also need to be able to respond to system prompts, make inquiries of the information system, judge response times of the system, and issue commands.

3) **Test and Revise:** After finishing the initial prototype, the designers first demonstrate the model to users and then give it to them to experiment and ask users to record their likes and dislikes about the system and recommend changes. Using this feedback, the design team modifies the prototype as necessary and then resubmits the revised model to system users for reevaluation. Thus iterative process of modification and reevaluation continues until the users are satisfied.

4) **Obtain User Signoff of the Approved Prototype:** Users formally approve the final version of the prototype, which commits them to the current design and establishes a contractual obligation about what the system will, and will not, do or provide. Prototyping is not commonly used for developing traditional applications such as accounts receivable, accounts payable, payroll, or inventory management, where the inputs, processing, and outputs are well known and clearly defined.

5) **Strengths:** Some of its strengths identified by the experts and practitioners include the following:
   - **a)** It improves both user participation in system development and communication among project stakeholders.
   - **b)** It is especially useful for resolving unclear objectives; developing and validating user requirements; experimenting with or comparing various design solutions, or investigating both performance and the human computer interface.
   - **c)** Potential exists for exploiting knowledge gained in an early iteration as later iterations are developed.
   - **d)** It helps to easily identify, confusing or difficult functions and missing functionality.
   - **e)** It enables to generate specifications for a production application.
   - **f)** It encourages innovation and flexible designs.
   - **g)** It provides for quick implementation of an incomplete, but functional, application.
   - **h)** It typically results in a better definition of these users' needs and requirements than does the traditional systems development approach.
   - **i)** A very short time period is normally required to develop and start experimenting with a prototype. This short time period allows system users to immediately evaluate proposed system changes.
   - **j)** Since system users experiment with each version of the prototype through an interactive process, errors are hopefully detected and eliminated early in the developmental process. As a result, the information system ultimately implemented should be more reliable and less costly to develop than when the traditional systems development approach is employed.

6) **Weaknesses:** Some of the weaknesses identified by the experts and practitioners include the following:
   - **a)** Approval process and control are not strict.
   - **b)** Incomplete or inadequate problem analysis may occur whereby only the most obvious and superficial needs will be addressed, resulting in current inefficient practices being easily built into the new system.
   - **c)** Requirements may frequently change significantly.
   - **d)** Identification of non-functional elements is difficult to document.
   - **e)** Designers may prototype too quickly, without sufficient upfront user needs analysis, resulting in an inflexible design with narrow focus that limits future system potential.
   - **f)** Prototype may not have sufficient checks and balances incorporated.
   - **g)** Prototyping can only be successful if the system users are willing to devote significant time in experimenting with the prototype and provide the system developers with change suggestions. The users may not be able or willing to spend the amount of time required under the prototyping approach.
   - **h)** The interactive process of prototyping causes the prototype to be experimented with quite extensively. Because of this, the system developers are frequently tempted to minimize the testing and documentation process of the ultimately approved information system. Inadequate testing can make the approved system error-prone, and inadequate documentation makes this system difficult to maintain.
   - **i)** Prototyping may cause behavioral problems with system users. These problems include dissatisfaction by users if system developers are unable to meet all user demands for improvements as well as dissatisfaction and impatience by users when they have to go through too many interactions of the prototype.

5.7. The Incremental Model

The Incremental model is a method of software development where the model is designed, implemented and tested incrementally (a little more is added each time) until the product is finished. The product is defined as finished when it satisfies all of its requirements. This model combines the elements of the waterfall model with the iterative philosophy of prototyping.

The product is decomposed into a number of components, each of which are designed and built separately (termed as builds). Each component is delivered to the client when it is complete. This allows partial utilization of product and avoids a long development time.

A few pertinent features are listed as follows:

1) A series of mini-waterfalls are performed, where all phases of the waterfall development model are completed for a small part of the system, before proceeding to the next increment.

2) Overall requirements are defined before proceeding to evolutionary, mini – Waterfall development of individual increments of the system.

3) The initial software concept, requirement analysis, and design of architecture and system core are defined using the Waterfall approach, followed by iterative Prototyping, which culminates in installation of the final prototype.

4) **Strengths**: Some of its strengths identified by the experts and practitioners include the following:
- **a)** Potential exists for exploiting knowledge gained in an early increment as later increments are developed.
- **b)** Moderate control is maintained over the life of the project through the use of written documentation and the formal review and approval/signoff by the user and information technology management at designated major milestones.
- **c)** Stakeholders can be given concrete evidence of project status throughout the life cycle.
- **d)** It is more flexible and less costly to change scope and requirements.
- **e)** It helps to mitigate integration and architectural risks earlier in the project.
- **f)** It allows the delivery of a series of implementations that are gradually more complete and can go into production more quickly as incremental releases.
- **g)** Gradual implementation provides the ability to monitor the effect of incremental changes, isolated issues and make adjustments before the organization is negatively impacted.

5) **Weaknesses:** Some of the weaknesses identified by the experts and practitioners include the following:
- **a)** When utilizing a series of mini-waterfalls for a small part of the system before moving onto the next increment, there is usually a lack of overall consideration of the business problem and technical requirements for the overall system.
- **b)** Each phase of an iteration is rigid and do not overlap each other.
- **c)** Problems may arise pertaining to system architecture because not all requirements are gathered up front for the entire software life cycle.
- **d)** Since some modules will be completed much earlier than others, well-defined interfaces are required.
- **e)** It is difficult to demonstrate early success to management.

5.8. Spiral Model

The Spiral model is a software development process combining elements of both design and prototyping-in-stages. It tries to combine advantages of top-down and bottom-up concepts. It combines the features of the prototyping model and the waterfall model. The spiral model is intended for large, expensive and complicated projects. Game development is a main area where the spiral model is used and needed, that is because of the size and the constantly shifting goals of those large projects. A list of pertinent characterizing features includes the following:

1) The new system requirements are defined in as much detail as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.

2) A preliminary design is created for the new system. This phase is the most important part of "Spiral Model" in which all possible alternatives that can help in developing a cost effective project are analyzed and strategies are decided to use them. This phase has been added specially in order to identify and resolve all the possible risks in the project development. If risks indicate any kind of uncertainty in requirements, prototyping may be used to proceed with the available data and find out possible solution in order to deal with the potential changes in the requirements.

3) A first prototype of the new system in constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.

4) A second prototype is evolved by a fourfold procedure by evaluating the first prototype in terms of its strengths, weaknesses, and risks; defining the requirements of the second prototype; planning and designing the second prototype; and constructing and testing the second prototype.

5) **Strengths:** Some of its strengths identified by the experts and practitioners include the following:

- **a)** It enhances the risk avoidance.
- **b)** It is useful in helping for optimal development of a given software iteration based on project risk.
- **c)** It can incorporate Waterfall, Prototyping, and Incremental methodologies as special cases in the framework, and provide guidance as to which combination of these models best fits a given software iteration, based upon the type of project risk. For example, a project with low risk of not meeting user requirements but high risk of missing budget or schedule targets would essentially follow a linear Waterfall approach for a given software iteration. Conversely, if the risk factors were reversed, the Spiral methodology could yield an iterative prototyping approach.

6) **Weaknesses:** Some of the weaknesses identified by the experts and practitioners include the following:

- **a)** It is challenging to determine the exact composition of development methodologies to use for each iteration around the Spiral.
- **b)** It may prove highly customized to each project, and thus is quite complex and limits reusability.
- **c)** A skilled and experienced project manager is required to determine how to apply it to any given project.
- **d)** No established controls exist for moving from one cycle to another cycle. Without controls, each cycle may generate more work for the next cycle.
- **e)** There are no firm deadlines, cycles continue with no clear termination condition leading to, inherent risk of not meeting budget or schedule.

5.9. Rapid Application Development (RAD) Model

Rapid Application Development (RAD) refers to a type of software development methodology. which uses minimal planning in favor of rapid prototyping. The planning of software developed using RAD is interleaved with writing the software itself. The lack of extensive pre-planning generally allows software to be written much faster, and makes it easier to change requirements. Key features include the following:

1) Key objective is fast development and delivery of a high quality system at a relatively low investment cost,

2) Attempts to reduce inherent project risk by breaking a project into smaller segments and providing more ease-of-change during the development process.

3) Aims to produce high quality systems quickly, primarily through the use of iterative Prototyping (at any stage of development), active user involvement, and computerized development tools. Graphical User Interface (GUI) builders, Computer Aided Software Engineering (CASE) tools, Database Management Systems (DBMS), Fourth generation programming languages, Code generators and object-oriented techniques etc.

4) Key emphasis is on fulfilling the business need while technological or engineering excellence is of lesser importance.

5) Project control involves prioritizing development and defining delivery deadlines or "timeboxes." If the project starts to slip, emphasis is on reducing requirements to fit the timebox, not in increasing the deadline.

6) Generally includes Joint Application Development (JAD), where users are intensely involved in system design, either through consensus building in structured workshops, or through electronically facilitated interaction.

7) Active user involvement is imperative.

8) Iteratively produces production software, as opposed to a throwaway prototype.

9) Produces documentation necessary to facilitate future development and maintenance.

10) Standard systems analysis and design techniques can be fitted into this framework.

11) **Strengths:** Some of the strengths identified by the experts and practitioners include the following:

- **a)** The operational version of an application is available much earlier than with Waterfall, Incremental, or Spiral frameworks.
- **b)** Because RAD produces systems more quickly and to a business focus, this approach tends to produce systems at lower cost.
- **c)** Quick initial reviews are possible.
- **d)** Constant integration isolates problems and encourages customer feedback.
- **e)** It holds a great level of commitment from stakeholders, both business and technical, than Waterfall, Incremental, or spiral frameworks. Users are seen as gaining more of a sense of ownership of a system, while developer are seen as gaining more satisfaction from producing successful systems quickly.
- **f)** It concentrates on essential system elements from user viewpoint.
- **g)** It provides for the ability to rapidly change system design as demanded by users.
- **h)** It leads to a tighter fit between user requirements and system specifications.

12) **Weaknesses:** Some of the weaknesses identified by the experts and practitioners include the following:
   - **a)** Fast speed and lower cost may affect adversely the system quality.
   - **b)** The project may end up with more requirements than needed (gold-plating).
   - **c)** Potential for feature creep where more and more features are added to the system over the course of development.
   - **d)** It may lead to inconsistent designs within and across systems.
   - **e)** It may call for violation of programming standards related to inconsistent naming conventions and inconsistent documentation,
   - **f)** It may call for lack of attention to later system administration needs built into system.
   - **g)** Formal reviews and audits are more difficult to implement than for a complete system.
   - **h)** Tendency for difficult problems to be pushed to the future to demonstrate early success to management.
   - **i)** Since some modules will be completed much earlier than others, well–defined interfaces are required.

5.10. Agile Model

This is an organized set of software development methodologies based on the iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. It promotes adaptive planning, evolutionary development and delivery; time boxed iterative approach and encourages rapid and flexible response to change. It is a conceptual framework that promotes foreseen interactions throughout the development life cycle. Following are considered as the key features of the agile methodologies:

1) Customer satisfaction by rapid delivery of useful software;

2) Welcome changing requirements, even late in development;

3) Working software is delivered frequently (weeks rather than months);

4) Working software is the principal measure of progress;

5) Sustainable development, able to maintain a constant pace;

6) Close, daily co-operation between business people and developers;

7) Face-to-face conversation is the best form of communication (co-location);

8) Projects are built around motivated individuals, who should be trusted;

9) Continuous attention to technical excellence and good design;

10) Simplicity; Self-organizing teams; and

11) Regular adaptation to changing circumstances.

12) **Strengths:** Some of the strengths identified by the experts and practitioners include the following:
   - **a)** Agile methodology has the concept of an adaptive team, which enables to respond to the changing requirements.
   - **b)** The team does not have to invest time and efforts and finally find that by the time they delivered the product, the requirement of the customer has changed.
   - **c)** Face to face communication and continuous inputs from customer representative leaves a little space for guesswork.
   - **d)** The end result is generally the high quality software in least possible time duration and satisfied customer.

13) Weaknesses: Some of the weaknesses identified by the experts and practitioners include the following:

- **a)** In case of some software deliverables, especially the large ones, it is difficult to assess the efforts required at the beginning of the software development life cycle.
- **b)** There is lack of emphasis on necessary designing and documentation.
- **c)** Agile increases potential threats to business continuity and knowledge transfer. By nature, Agile projects are extremely light on documentation because the team focuses on verbal communication with the customer rather than on documents or manuals.
- **d)** Agile requires more re-work and due to the lack of long-term planning and the lightweight approach to architecture, re-work is often required on Agile projects when the various components of the software are combined and forced to interact.
- **e)** The project can easily get taken off track if the customer representative is not clear about the final outcome.
- **f)** Agile lacks the attention to outside integration.

System Development Life Cycle (SDLC)

The process of system development starts when management or sometimes system development personnel realize that a particular business system needs improvement. The System Development Life Cycle method can be thought as a set of activities that analysts, designers and users carry out to develop and implement an information system. In most of the business situations, these activities are closely related, usually inseparable and even the order of the steps in these activities may be difficult to determine. Different parts of a project can be in various phases at the same time, with some components undergoing analysis while others are at advanced design stages.

System Development Life Cycle

PHASE	PHASE NAME	NATURE OF ACTIVITY
1.	Preliminary Investigation	Determining and evaluating the strategic feasibility of the system and e that the solution fits the business strategy.
2.	Systems Requirer Analysis	Analyzing the typical system requirements, in view if its functional deliverables etc.
3.	Systems Design	Designing the system in terms of user interface, data storage and processing functions on the basis of the requirement phase by developin system flowcharts, system and data flow diagrams, screens and reports.
4.	Systems Developr	Developing the system as per the system designed in view of its ade implementation to lead to fulfillment of requirements to the satisfaction the stakeholders.
5.	Systems Testing	Requisite testing to ensure the valid and reliable implementations.
6.	Systems Implementation	Operationalization of the developed system for the acceptance management and user before migration of the system to the environment and data conversion from legacy system to the new syste
7.	Post Implement Review and Maintenance	Continuous evaluation of the system as it functions in the live environ and its updation / maintenance.

CHAPTER -5
ACQUISITION, DEVELOPMENT AND IMPLEMENTATION OF SYSTEMS

5.11. Preliminary Investigation

It is predominantly aimed to determine and analyze the strategic benefits in implementing the system through evaluation and quantification of - productivity gains; future cost avoidance; cost savings, and Intangible benefits like improvement in morale of employees. The deliverable of the preliminary investigation includes a report including feasibility study observations.
Thereby, it largely enables the requirements engineer to tackle the issues and Feasibility Study for the following:

- Determine whether the solution is as per the business strategy:
- Determine whether the existing system can rectify the situation without a major modification;
- Define the time frame for which the solution is required;
- Determine the approximate cost to develop the system; and
- Determine whether the vendor product offers a solution to the problem.

1) **Identification of Problem:** The first step in an application development is to define the problem clearly and precisely, which is done only after the critical study of the existing system and several rounds of discussions with the user group. Then its prevalence within the organization has to be assessed. A problem that has a considerable impact on the organization is likely to receive immediate management attention. User involvement will also be high, if they are convinced that the proposed solution will resolve the problem.
   The analyst working on the preliminary investigation should accomplish the following objectives:
   a) Clarify and understand the project request;
   b) Determine the size of the project;
   c) Determine the technical and operational feasibility of alternative approaches;
   d) Assess costs and benefits of alternative approaches; and
   e) Report findings to the management with recommendation outlining the acceptance or rejection of the proposal.

2) **Identification of Objectives:** After the identification of the problem, it is easy to work out and precisely specify the objectives of the proposed solution. For instance, inability to provide a convenient reservation system, for a large number of intending passengers was the problem of the Railways. So, one of the objectives was 'to introduce a system wherein intending passengers could book a ticket from source to destination, faster than in real-time'.

3) **Delineation of Scope:** The scope of a solution defines its typical boundaries. It should be clear and comprehensible to the user management stating the extent and 'what will be addressed by the solution and what will not'. Often, the scope becomes a contentious issue between development and user organizations. Hence, outlining the scope in the beginning is essential and proves quite handy. The typical scope determination may be performed on the following dimensions:
   a) Functionality Requirements: What functionalities will be delivered through the solution?
   b) Data to be Processed: What data is required to achieve these functionalities?
   c) Control Requirements: What are the control requirements for this application?
   d) Performance Requirements: What level of response time, execution time and throughput is required?
   e) Constraints: What are the conditions the input data has to conform to? For example, what is the maximum number of characters that a name can have in a database?
   f) Interfaces: Is there any special hardware/software that the application has to interface with? For example-Payroll application may have to capture from the attendance monitoring system that the company has already installed. Then the solution developer has to understand the format of data, frequency mode of data transfer and other aspects of the software.
   g) Reliability requirements: Reliability of an application is measured by its ability to remain uncorrupted in the face of inadvertent / deliberate misuse and probability of failure-free operations. The reliability required for an application depends on its criticality and the user profile.

h) **Two primary methods** with the help of which the scope of the project can be analyzed are given as follows:

i) **Reviewing Internal Documents:** The analysts conducting the investigation first try to learn about the organization involved in, or affected by, the project.

ii) **Conducting Interviews:** Interviews allow analysts to know more about the nature of the project request and the reasons for submitting it. Usually, preliminary investigation interviews involve only management and supervisory personnel.

4) **Feasibility Study:** After possible solution options are identified, project feasibility i.e. the likelihood that these systems will be useful for the organization is determined. A feasibility study is carried out by the system analysts, which refers to a process of evaluating alternative systems through cost/benefit analysis so that the most feasible and desirable system can be selected for development. The Feasibility Study of a system is evaluated under following dimensions described briefly as follows:

a) **Technical:** Is the technology needed available?
b) **Financial:** Is the solution viable financially?
c) **Economic:** Return on Investment?
d) **Schedule/Time:** Can the system be delivered on time?
e) **Resources:** Are human resources reluctant for the solution?
f) **Operational:** How will the solution work?
g) **Behavioral:** Is the solution going to bring any adverse effect on quality of work life?
h) **Legal:** Is the solution valid in legal terms?

The detailed description of each dimension is given as follows:

a) **Technical Feasibility:** It may try to answer, whether implementation of the project viable using current technology? It is concerned with issues pertaining to hardware and software. Essentially, an analyst ascertains whether the proposed system is feasible with existing or expected computer hardware and software technology. The technical issues usually raised during the feasibility stage of investigation include the following:

i) Does the necessary technology exist to do what is suggested (and can it be acquired)?
ii) Does the proposed equipment have the technical capacity to hold the data required to use the new system.
iii) Can the proposed application be implemented with existing technology?
iv) Will the proposed system provide adequate responses to inquiries, regardless of the number or location of users?
v) Can the system be expanded if developed?
vi) Are there technical guarantees of accuracy, reliability, ease of access, and data security?

Some of the technical issues to be considered are given below.

Design Considerations	Design Alternatives
Communications Channel configuration	Point to point, multidrop, or line sharing
Communications Channel	Telephone lines, coaxial cable, fiber optics, microwave, or satellite
Communications network	Centralized, decentralized, distributed, or local area
Computer programs	Independent vendor or in-house
Data storage medium	Tape, floppy disk, hard disk, or hard copy
Data storage structure	Files or database

b) Financial Feasibility: The solution proposed may be prohibitively costly for the user organization. For example, Monitoring the stock through VSAT network connecting multiple locations may be acceptable for an organization with high turnover. But this may not be a viable solution for smaller ones.

c) Economic Feasibility: It includes an evaluation of all the incremental costs and benefits expected if the proposed system is implemented. After problems or opportunities are identified, the analysts must determine the scale of response needed to meet the user's requests for a new system as well as the approximate amount of time and money that will be required in the effort. The financial and economic questions raised by analysts during the preliminary investigation are for the purpose of estimating the following:

i) The cost of conducting a full systems investigation;
ii) The cost of hardware and software for the class of applications being considered;
iii) The benefits in the form of reduced costs or fewer costly errors; and
iv) The cost if nothing changes (i.e. the proposed system is not developed).

d) Schedule or Time Feasibility: Schedule feasibility involves the design team's estimating how long it will take a new or revised system to become operational and communicating this information to the steering committee.

e) Resources Feasibility: This focuses on human resources. Implementing sophisticated software solutions becomes difficult at specific locations because of the reluctance of skilled personnel to move to such locations.

f) Operational Feasibility: It is concerned with ascertaining the views of workers, employees, customers and suppliers about the use of computer facility. A system can be highly feasible in all respects except the operational and fails miserably because of human problems. Some of the questions, which help in testing the operational feasibility of a project, may include the following:

i) Is there sufficient support for the system from management and from users?
ii) Are current business methods acceptable to users?
iii) Have the users been involved in planning and development of the project?
iv) Will the proposed system cause harm? Will it produce poorer results in any respect or area? Will loss of control result in any areas? Will accessibility of information be lost?
v) Will individual performance be poorer after implementation than before?

g) Behavioral Feasibility: It refers to the systems, which is to be designed to process data and produce the desired outputs. However, if the data input for the system is not readily available or collectable, then the system may not be successful.

h) Legal Feasibility: Legal feasibility is largely concerned with whether there will be any conflict between a newly proposed system and the organization's legal obligations. Any system, which is liable to violate the local legal requirements, should also be rejected. For example, a revised system should comply with all applicable statutes about financial and statuary reporting requirements, as well as the company's contractual obligations.

5) Reporting Results to Management: After the analyst articulates the problem, defines the same along with its scope, s/he provides one or more solution alternatives and estimates the cost and benefits of each alternative and reports these results to the management. The report should be accompanied by a short covering letter of intent that summarizes the results and makes the recommendation regarding further procedures.

CHAPTER -5
ACQUISITION, DEVELOPMENT AND IMPLEMENTATION OF SYSTEMS

5.12. System Requirements Analysis

This phase includes a thorough and detailed understanding of the current system, identifies the areas that need modification to solve the problem, the determination of user/managerial requirements and to have fair idea about various systems development tools.
The following objectives are performed in this phase in order to generate the deliverable, Systems Requirements Specification (SRS):

- To identify and consult the stake owners to determine their expectations and resolve their conflicts;
- To analyze requirements to detect and correct conflicts and determine priorities;
- To gather data or find facts using tools like - interviewing, research/document collection, questionnaires, observation;
- To verify that the requirements are complete, consistent, unambiguous, verifiable, modifiable, testable and traceable;
- To model activities such as developing models to document Data Flow Diagrams, E-R Diagrams; and
- To document activities such as interview, questionnaires, reports etc. and development of a system (data) dictionary to document the modeling activities.

1) **Fact Finding:** Various fact-finding techniques/tools are used by the system analyst for determining these needs/requirements are briefly discussed below:
   a) **Documents:** Document means manuals, input forms, output forms, diagrams of how the current system works, organization charts showing hierarchy of users and manager responsibilities, job descriptions for the people, who work with the current system, procedure manuals, etc. Documents are a very good source of information about user needs and the current system.

   b) **Questionnaires:** Users and managers are asked to complete questionnaire about the information systems when the traditional system development approach is chosen. The main strength of questionnaires is that a large amount of data can be collected through a variety of users quickly. Also, if the questionnaire is skillfully drafted, responses can be analyzed rapidly with the help of a computer.

   c) **Interviews:** Users and managers may also be interviewed to extract information in depth. The data gathered through interviews often provide system developers with a larger picture of the problems and opportunities. Interviews also give analyst the opportunity to observe and record first-hand user reaction and to probe for further information.

   d) **Observation:** In general and particularly in prototyping approaches, observation plays a central role in requirement analysis. Only by observing how users react to prototypes of a new system, the system can be successfully developed.

2) **Analysis of the Present System:** Detailed investigation of the present system involves collecting, organizing and evaluating facts about the system and the environment in which it operates. There should be enough information assembled so that a qualified person can understand the present system without visiting any of the operating departments.
   a) **Reviewing Historical Aspects:** A brief history of the organization is a logical starting point for an analysis of the present system. The historical facts enable to identify the major turning points and milestones that have influenced its growth. The system analyst should investigate ‘what system changes have occurred in the past including operations’ that have been successful or unsuccessful with computer equipment and techniques.

   b) **Analyzing Inputs:** A detailed analysis of present inputs is important since they are basic to the manipulation of data. Source documents are used to capture the originating data for any type of system. The system analyst must understand the nature of each form, ‘what is

contained in it', 'who prepared it', 'from where the form is initiated', 'where it is completed', the distribution of the form and other similar considerations. If the analyst investigates these questions thoroughly, s/he will be able to determine how these inputs fit into the framework of the present system.

c) **Reviewing Data Files:** The analyst should investigate the data files maintained by each department, noting their number and size, where they are located, who uses them and the number of times per given time interval, these are used. The system analyst should also review all on-line and off-line files, which are maintained in the organization as it will reveal information about data that are not contained in any outputs. The related cost of retrieving and processing the data is another important factor that should be considered by the systems analyst.

d) **Reviewing Methods, Procedures and Data Communications:** Methods and procedures transform input data into useful output. A method is defined as a way of doing something; a procedure is a series of logical steps by which a job is accomplished. Its basic objective is to eliminate unnecessary tasks or to perceive improvement opportunities in the present information system. A system analyst also needs to review and understand the present data communications used by the organization. S/he must review the types of data communication equipment including data interface, data links, modems, dial-up and leased lines and multiplexers.

e) **Analyzing Outputs:** The outputs or reports should be scrutinized carefully by the system analysts in order to determine 'how well they will meet the organization's needs. The analysts must understand what information is needed and why, who needs it and when and where it is needed. Additional questions concerning the sequence of the data, how often the form reporting is used, how long is it kept on file, etc. must be investigated.

f) **Reviewing Internal Controls:** A detailed investigation of the present information system is not complete until internal control mechanism is reviewed. An examination of the present system of internal controls may indicate weaknesses that should be removed in the new system. The adoption of advanced methods, procedures and equipments might allow much greater control over the data.

g) **Modeling the Existing System:** As the logic of inputs, methods, procedures, data files, data communications, reports, internal controls and other important items are reviewed and analyzed in a top down manner; the processes must be properly documented. The flow charting and diagramming of present information not only organizes the facts, but also helps to disclose gaps and duplication in the data gathered.

h) **Undertaking Overall Analysis of the Existing system:** Based upon the aforesaid investigation of the present information system, the final phase of the detailed investigation includes the analysis of the present work volume; the current personnel requirements; the present costs-benefits of each of these must be investigated thoroughly.

3) **System Analysis of Proposed Systems:** After a thorough analysis of each functional area of the present information system, the proposed system specifications must be clearly defined. The required systems specifications should be in conformity with the project's objectives articulated and in accordance with the following:
   a) Outputs are produced with great emphasis on timely managerial reports that utilize the management by exception' principle.
   b) Databases are maintained with great accent on online processing capabilities.
   c) Input data is prepared directly from original source documents for processing by the computer system.

d) Methods and procedures that show the relationship of inputs and outputs to the database, utilize data communications as, when and where deemed appropriate.

e) Work volumes and timings are carefully considered for present and future periods including peak periods.

f) The starting point for compiling these specifications is output. The output-to-input process is recommended since outputs are related directly to the objectives of the organization.

4) System Development Tools: Many tools and techniques have been developed to improve current information systems and to develop new ones. Such tools help end users and systems analysts primarily for the following:

a) To conceptualize, clarify, document and communicate the activities and resources involved in the organization and its information systems;

b) To analyze present business operations, management decision making and information processing activities of the organization; and

c) To propose and design new or improved information systems to solve business problems or pursue business opportunities that have been identified.

The major tools used for system development specification or representations can be classified into four categories based on the systems features. These are described briefly as follows:

a) System Components and Flows: These tools help the system analysts to document the data flow among the major resources and activities of an information system. System flow charts are typically used to show the flow of data media as they are processed by the hardware devices and manual activities.

b) User Interface: Designing the interface between end users and the computer system is a major consideration of a system analyst while designing the new system. Layout forms and screens are used to construct the formats and contents of input/output media and methods.

c) Data Attributes and Relationships: The data resources in information system are defined, catalogued and designed by this category of tools. A Data Dictionary catalogs the description of the attributes (characteristics) of all data elements and their relationships to each other as well as to external systems. Entity-relationship diagrams are used to document the number and type of relationship among the entities in a system. File layout forms document the type, size and names of the data elements in a system.

d) Detailed System Processes: These tools are used to help the programmer to develop detailed procedures and processes required in the design of a computer program. Decision trees and decision tables use a network or tabular form to document the complex conditional logic involved in choosing among the information processing alternatives in a system.

2) CASE tools are in use for typical representation, specifications and system modeling. A set of the prominent tools have been already mentioned categorically and some are being described in detail as follows:

a) Structured English: Structured English, also known as Program Design Language (PDL), is the use of the English language with the syntax of structured programming. Thus, Structured English aims at getting the benefits of both the programming logic and natural language. A better structured, universal and precise tool is referred to as pseudo code.

b) Flowcharts: Flowcharting is a pictorial representation technique that can be used by analysts to represent the inputs, outputs and processes of a business process. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields.

c) Data Flow Diagrams: A Data Flow Diagram uses few simple symbols to illustrate the flow of data among external entities (such as people or organizations, etc.), processing activities and data storage elements. A DFD is composed of four basic elements: Data Sources and Destinations, Data Flows, Transformation processes, and Data stores

d) Decision Tree: A Decision Tree or tree diagram is a support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. Decision tree is commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal and to calculate conditional probabilities.

e) Decision Table: A Decision Table is a table, which may accompany a flowchart, defining the possible contingencies that may be considered within the program and the appropriate course of action for each contingency. The four parts of the decision table are given as follows:

- **i) Condition Stub** – This comprehensively lists the comparisons or conditions;
- **ii) Action Stub** – This comprehensively lists the actions to be taken along various program branches;
- **iii) Condition entries** – This list in its various columns the possible permutations of answer to the questions in the conditions stub); and
- **iv) Action entries** – This lists in its columns corresponding to the condition entries the actions contingent upon the set of answers to questions of that column.

f) CASE Tools: CASE refers to the automation of anything that humans do to develop systems and support virtually all phases of traditional system development process. For example, these packages can be used to create complete and internally consistent requirements specifications with graphic generators and specifications languages. An ideal CASE system would have an integrated set of tools and features to perform all aspects in the life cycle. Some of the features that various CASE products possess are - Repository / Data Dictionary; Computer aided Diagramming Tools; Word Processing; Screen and Report generator; Prototyping; Project Management; Code Generation; and Reverse Engineering.

i) System Components Matrix: A System Component Matrix provides a matrix framework to document the resources used, the activities performed and the information produced by an information system. It can be used as an information system framework for both systems analysis and system design and views the information system as a matrix of components that highlights how the basic activities of input, processing, output, storage and controls are accomplished in an information system, and how the use of hardware, software and people resources can convert data resources into information products.

ii) Data Dictionary: A data dictionary contains descriptive information about the data items in the files of a business information system. Thus, a data dictionary is a computer file about data. Each computer record of a data dictionary contains information about a single data item used in a business information system. This information may include - the identity of the source document(s) used to create the data item; the names of the computer files that store the data item; the names of the computer programs that modify the data item; the identity of the computer programs or individuals permitted to access the data item for the purpose of file maintenance, upkeep, or inquiry; the identity of the computer programs or individuals not permitted to access the data item etc. As new data fields are added to the record structure of a business file, information about each new data item is used to create a new record in the data dictionary.

iii) User Interface Layout and Forms: Several type layout forms for both soft and hard copy are used to model input/output components of an automated information system. Some of the prominent and inevitable ones are described briefly as follows:

(1) Layout form and Screen Generator: These are for printed report used to format or "paint" the desired layouts and contact without having to enter complex formatting information.

(2) Menu Generator: Menu generator outlines the functions, which the system is aimed to accomplish. Menu may be linked to other submenus that will enable the user to understand how the screens and sub-screens will be used for data entry or inquiry.

(3) Report Generator: Report generator has capacity of performing similar functions as found in screen generators. In addition, it can also indicate totals, paging, sequencing and control breaks in creating samples of the desired report.

(4) Code Generator: Code generator allows the analyst to generate modular units of source code from the high level specifications provided by the system analyst and play significant role in systems development process.

3) **Systems Specification:** At the end of the analysis phase, the systems analyst prepares a document called Systems Requirement Specifications (SRS). A well documented SRS may normally contains the following sections:
   - **a) Introduction:** Goals, Objectives, software context, Scope and Environment of the computer-based system.
   - **b) Information Description:** Problem description; Information content, flow and structure; Hardware, software, human interfaces for external system elements and internal software functions.
   - **c) Functional Description:** Diagrammatic representation of functions; Processing narrative for each function; Interplay among functions; Design constraints.
   - **d) Behavioral Description:** Response to external events and internal controls.
   - **e) Validation Criteria:** Classes of tests to be performed to validate functions, performance and constraints.
   - **f) Appendices:** Data flow/Object Diagrams; Tabular Data; Detailed description of algorithms charts, graphs and other such material.
   - **g) SRS Review:** The development team makes a presentation and then hands over the SRS document to be reviewed by the user or customer. The review reflects the development team's understanding of the existing processes. This is a technical requirement of the contract between users and development team/organization.

4) **Roles Involved in SDLC:** A variety of tasks during the SDLC are performed by special teams/committees/individuals based on requisite expertise as well as skills. Some of the generic roles are described as follows:
   - **a) Steering Committee:** It is a special high power committee of experts to accord approvals for go-ahead and implementations. Some of the functions of Steering Committee are given as follows:
     - **i)** To provide overall directions and ensures appropriate representation of affected parties;
     - **ii)** To be responsible for all cost and timetables;
     - **iii)** To conduct a regular review of progress of the project in the meetings of steering committee, which may involve co-ordination and advisory functions; and
     - **iv)** To undertake corrective actions like rescheduling, re-staffing, change in the project objectives and need for redesigning.

   - **b) Project Manager:** A project manager is normally responsible for more than one project and liaisonsing with the client or the affected functions. S/he is responsible for delivery of the project deliverables within the time/budget and periodically reviews the progress of the project with the project leader and his/her team.

   - **c) Project Leader:** The project leader is dedicated to a project, who has to ensure its completion and fulfillment of objectives. S/he reviews the project status more frequently than a Project Manager and the entire project team reports to him/her.

d) Systems Analyst / Business Analyst: The systems analysts' main responsibility is to conduct interviews with users and understand their requirements. S/he is a link between the users and the designers/programmers, who convert the users' requirements in the system requirements and plays a pivotal role in the Requirements analysis and Design phase.

e) Module Leader/Team Leader: A project is divided into several manageable modules, and the development responsibility for each module is assigned to Module Leaders. For example, while developing a financial accounting application – Treasury, Accounts payable, Accounts receivable can be identified as separate modules and can be assigned to different module leaders. Module leaders are responsible for the delivery of tested modules within the stipulated time and cost.

f) Programmer/Developers: Programmers is a mason of the software industry, who converts design into programs by coding using programming language. Apart from developing the application in a programming language, they also test the program for debugging activity to assure correctness and reliability.

g) Database Administrator: The data in a database environment has to be maintained by a specialist in database administration so as to support the application program. The DBA handles multiple projects; ensures the integrity and security of information stored in the database and also helps the application development team in database performance issues. Inclusion of new data elements has to be done only with the approval of the database administrator.

h) Quality Assurance: This team sets the standards for development, and checks compliance with these standards by project teams on a periodic basis. Any quality assurance person, who has participated in the development process, shall not be viewed as 'independent' to carry out quality audits.

i) Testers: Testers are a junior level quality assurance personnel attached to a project, who test programs and subprograms as per the plan given by the module / project leaders and prepare test reports.

j) Domain Specialist: Whenever a project team has to develop an application in a field that's new to them, they take the help of a domain specialist. For example, if a team undertakes application development in Insurance, about which they have little knowledge, they may seek the assistance of an Insurance expert at different stages. This makes it easier to anticipate or interpret user needs. A domain specialist need not have knowledge of software systems.

k) IS Auditor: As a member of the team, IS Auditor ensures that the application development also focuses on the control perspective. S/he should be involved at the Design Phase and the final Testing Phase to ensure the existence and the operations of the Controls in the new software.

5) Internal Controls: Requirements phase is the most important phases of SDLC. The issue of controls is very important here also. Some of the key control aspects at this stage may be taken care by the following queries:

a) Whether present system analysis has been properly done?
b) Whether appropriate domain, were expert was engaged?
c) Whether all user requirements of proposed system have been considered?
d) Whether SRS document has been properly made and vetted by Users, Domain Experts, System Analysts?

CHAPTER -5
ACQUISITION, DEVELOPMENT AND IMPLEMENTATION OF SYSTEMS

5.13. System Designing

After the completion of requirements analysis for a system, systems designing activity takes place for the most feasible and optimal alternative, which is selected by management. The objective is to design an Information System that best satisfies the users/managerial requirements. It describes the parts of the system and their interaction. It sets out how the system shall be implemented using the chosen hardware, software and network facilities.

The key and generic design phase activities include describing inputs and outputs such as screen design and reports; determining the processing steps and computation rules for the new solution; determining data file or database system file design; preparing the program specifications for the various types of requirements or information criteria defined; and Internal/external controls.

1) **Architectural Design:** Architectural design deals with the organization of applications in terms of hierarchy of modules and sub-modules. At this stage, we identify major modules; functions and scope of each module; interface features of each module; modules that each module can call directly or indirectly and Data received from / sent to / modified in other modules. The architectural design is made with the help of a tool called Functional Decomposition. It has three elements – Module, Connection, and Couple. The module is represented by a box and connection between them by arrows. Couple is data element that moves from one module to another and is shown by an arrow with circular tai

2) **Design of Data/Information flow:** The design of the data and information flow is a major step in the conceptual design of the new system. In designing the data / information flow for the proposed system, the inputs that are required are - existing data / information flows, problems with the present system, and objective of the new system. All these have been identified in the analysis phase and documented in Software Requirements Specification (SRS).

3) **Design of Database:** Design of the database involves determining its scope ranging from local to global structure. The scope is decided on the basis of interdependence among organizational units. The design of the database involves four major activities,

Major Activities in Database Designing

Design Activity	Explanation
Conceptual Modeling	These describe the application domain via entities/objects, attributes of these entities/objects and static and dynamic constraints on these entities/objects, their attributes, and their relationships.
Data Modeling	Conceptual Models need to be translated into data models so that they can be accessed and manipulated by both high-level and low- level programming languages.
Storage Structure Design	Decisions must be made on how to linearize and partition the data structure so that it can be stored on some device. For example- tuples (row) in a relational data model must be assigned to records, and relationships among records might be established via symbolic pointer addresses.
Physical Layout Design	Decisions must be made on how to distribute the storage structure across specific storage media and locations for example, the cylinders, tacks, and sectors on a disk and the computers in a LAN or WAN.

4) **User Interface Design:** It involves determining the ways in which users will interact with a system. The points that need to be considered while designing the user interface are - source documents to capture raw data, hard-copy output reports, screen layouts for dedicated source-document input, inquiry screens for database interrogation, graphic and color displays, and requirements for special input/output device. Important factors in Input/Output design are

Factors affecting Input / Output Form Designs

Definition	Input Design	Output Design
Content		
Refers to the actual pieces of data to be gathered to produce the required output to be provided to users.	The analyst is required to consider the types of data that are needed to be gathered to generate the desired user outputs.	The contents of a weekly output report to a sales manager might consist of sales person's name, sales calls made by each sales person during the week, and the amount of each product sold by each salesperson to each major client category.
Timelines Content		
Timeliness refers to when users need outputs, which may be required on a regular, periodic basis - perhaps daily, weekly, monthly, at the of quarter or annually.	Data needs to be inputted to computer in time because outputs cannot be produced until certain inputs are available.	A sales manager, may be requiring a weekly sales report. Other users, such as airline agents, require both real-time information and rapid response times in order to render better client service.
Format		
Input format refers to the manner in which data are physically arranged. Output format refers to the arrangement referring to data output on a printed report or in a display screen.	After the data contents and media requirements are determined, input formats are designed on the basis of few constraints like - the type and length of each data field as well as any other special characteristics (number decimal places etc.).	Format of information reports for the users should be so devised that it assists in decision- making, identifying and solving problems, planning and initiating corrective action and searching.
Media		
Input-output medium refers to the physical device used for input, storage or output.	This includes the choice of input media and subsequently the devices on which to enter the data. Various user input alternatives may include display workstations, magnetic tapes, magnetic disks, key- boards, optical character recognition, pen-based computers and voice input etc. A suitable medium may be selected depending on the application to be computerized.	A variety of output media are available in the market these days which include paper, video display, microfilm, magnetic tape/disk and voice output.
Form		

Form refers to the way the information is inputted in the input form and the content is presented to users in various output forms - quantitative, non-quantitative, text, graphics, video and audio.	Forms are pre-printed papers that require people to fill in responses in a standardized way. Forms elicit and capture information required by organizational members that often will be input to the computer. Through this process, forms often serve as source documents for the data entry personnel.	The form of the output should be decided keeping in view the requirements for the concerned user. For example - Information on distribution channels may be more understandable to the concerned manager if it is presented in the form of a map, with dots representing individual outlets for stores.
Input/Output Volume		
Input volume refers to the amount of data that has to be entered in the computer system at any one time. The amount of data output required at any one time is known as output volume	In some decision- support systems and many real-time processing systems, input volume is light. In batch-oriented transaction processing systems, input volume could be heavy which involves thousands of records that are handled by a centralized data entry department using key- to-tape or key-	It is better to use high-speed printer or a rapid-retrieval display unit, which are fast and frequently used output devices in case the volume is heavy.

5) **Physical Design:** For the physical design, the logical design is transformed into units, which in turn can be decomposed further into implementation units such as programs and modules. Some of the issues addressed here are type of hardware for client application and server application, Operating systems to be used, type of networking, processing – batch – online, real – time; frequency of input, output; and month-end cycles / periodical processing.

Some of the generic design principles being applied to develop the design of typical information systems include the following:

a) There is a tendency to develop merely one design and consider it the final product. However, the recommended procedure is to design two or three alternatives and choose the best one on pre-specified criteria.
b) The design should be based on the analysis.
c) The software functions designed should be directly relevant to business activities.
d) The design should follow standards laid down. For instance, the user interface should have consistent color scheme, menu structure, location of error message and the like.
e) The design should be modular, with high cohesion and low coupling.

6) **Internal Design Controls:** From internal control point of view, this phase is also an important phase as all internal controls are placed in system during this phase. The key control aspects at this stage include the following:

a) Whether management reports of stage I and stage II, were referred by System Designer?
b) Whether all control aspects have been properly covered?
c) Whether controls put in place in system, appear in the documentation done at this stage?
d) Whether a separate review of design document has been done by internal auditor?

CHAPTER -5
ACQUISITION, DEVELOPMENT AND IMPLEMENTATION OF SYSTEMS

5.14. System Acquisition

After a system is designed either partially or fully, the next phase of the systems development starts, which relates to the acquisition of operating infrastructure including hardware, software and services.

1) **Acquisition Standards:** Management should establish acquisition standards that address the security and reliability issues as per current state-of-the art development standards. Acquisition standards should focus on the following:
   - **a)** Ensuring security, reliability, and functionality already built into a product;
   - **b)** Ensuring managers complete appropriate vendor, contract, and licensing reviews and acquiring products compatible with existing systems;
   - **c)** Invitations-to-tender soliciting bids from vendors when acquiring hardware or integrated systems of hardware and software;
   - **d)** Request-for-proposals soliciting bids when acquiring off-the-shelf or third-party developed software; and
   - **e)** Establishing acquisition standards to ensure functional, security, and operational requirements to be accurately identified and clearly detailed in request-for-proposals.

2) **Acquiring Systems Components from Vendors:** Acquiring the appropriate hardware and software is critical for the success of the whole project. The organization can discover new hardware and software developments in various ways. Management also decides whether the hardware is to be purchased, leased from a third party or to be rented. A sub-committee of experts under the steering committee, is constituted. The sub- committee is mandated to ensure timely and effective completion of this stage.

   The following considerations are valid for both acquisition of hardware and software:

   - **a) Vendor Selection:** This step is a critical step for success of process of acquisition of systems. It is necessary to remember that vendor selection is to be done prior to sending RFP. The result of this process is that 'RFP are sent only to selected vendors'. For vendor selection, following things are kept in mind including the background and locational advantage of the vendor, the financial stability of vendor, the market feedback of vendor performance, in terms of price, services etc.

   - **b) Geographical Location of Vendor:** The issue to look for whether the vendor has local support persons. Otherwise, the proposals submitted by vendor not as per RFP requirements need to rejected, with no further discussion on such rejected proposals. This stage may be referred to as 'technical validation', that is to check the proposals submitted by vendors, are technically complying with RFP requirements.

   - **c) Presentation by Selected Vendors:** All vendors, whose proposals are accepted after "technical validation", are allowed to make presentation to the System Acquisition Team. The team evaluates the vendor's proposals by using techniques.

   - **d) Evaluation of Users Feedback:** The best way to understand the vendor systems is to analyze the feedback from present users. Present users can provide valuable feedback on system, operations, problems, vendor response to support calls.

Besides these, some specific considerations for hardware and software acquisition are described as follows:

- **a)** The benchmark tests to be done for proposed machine. For hardware's, there are specified standard benchmark tests defined based on the nature of hardware. These need to be applied to proposed equipment.

- **b)** Software considerations that can be current applications programs or new programs that have been designed to represent planned processing needs.

c) The benchmarking problems are oriented towards testing whether a computer offered by the vendor meets the requirements of the job on hand of the buyer.

d) The benchmarking problems would then comprise long jobs, short jobs, printing jobs, disk jobs, mathematical problems, input and output loads etc., in proportion typical of the job mix.

e) If the job is truly represented by the selected benchmarking problems, then this approach can provide a realistic and tangible basis for comparing all vendors' proposals.

f) Tests should enable buyer to effectively evaluate cross performance of various systems in terms of hardware performance (CPU and input/output units), compiler language and operating system capabilities, diagnostic messages, ability to deal with certain types of data structures and effectiveness of software utilities.

g) Benchmarking problems, however, suffer from a couple of disadvantages. It takes considerable time and efforts to select problems representative of the job mix which itself must be precisely defined. It also requires the existence of operational hardware, software and services of systems. Nevertheless, this approach is very popular because it can test the functioning of vendors' proposal. The manager can extrapolate in the light of the results of benchmarking problems, the performance of the vendors' proposals on the entire job mix.

3) Other Acquisition Aspects and Practices: On addition to the above, there are several other acquisition aspects and practices also, which are given as follows:

a) Hardware Acquisition: In case of procuring such machinery as machine tools, transportation equipment, air conditioning equipment, etc., the management can normally rely on the time tested selection techniques and the objective selection criteria can be delegated to the technical specialist. The management depends upon the vendor for support services, systems design, education and training etc., and expansion of computer installation for almost an indefinite period; therefore, this is not just buying the machine and paying the vendor for it but it amounts to an enduring alliance with the supplier.

b) Software Acquisition: Once user output and input designs are finalized, the nature of the application software requirements must be assessed by the systems analyst. This determination helps the systems development team to decide 'what type of application software products is needed' and consequently, the degree of processing that the system needs to handle. This helps the system developers in deciding about the nature of the systems software and computer hardware that will be most suitable for generating the desired outputs, and also the functions and capabilities that the application software must possess. At this stage, the system developers must determine whether the application software should be created in-house or acquired from a vendor.

c) Contracts, Software Licenses and Copyright Violations: Contracts between an organization and a software vendor should clearly describe the rights and responsibilities of the parties to the contract. The contracts should be in writing with sufficient detail to provide assurances for performance, source code accessibility, software and data security, and other important issues. Software license is a license that grants usage permission to do things with computer software. The usual goal is to authorize activities, which are prohibited by default by copyright law, patent law, trademark law and any other intellectual property rights. The reason for the license, essentially is that virtually all intellectual property laws were enacted to encourage disclosure of the intellectual property. Copyright laws protect proprietary as well as open-source software. The use of unlicensed software or violations of a licensing agreement expose organizations to possible litigation.

d) Validation of Vendors' proposals: The contracts and software licensing process consists of evaluating and ranking the proposals submitted by vendors and is quite difficult, expensive and time consuming, but in any case it has to be gone through. This problem is made difficult by the fact that vendors would be offering a variety of configurations. The following factors have to be considered towards rigorous evaluation.

i) The Performance capability of each proposed System in Relation to its Costs;
ii) The Costs and Benefits of each proposed;
iii) The Maintainability of each proposed;
iv) The Compatibility of each proposed system with Existing Systems; and
v) Vendor Support.

e) Methods of Validating the proposal: Large organizations would naturally tend to adopt a sophisticated and objective approach to validate the vendor's proposal. Some of the validation methods are given as follows:

i) Checklists: It is the most simple and a subjective method for validation and evaluation. The various criteria are put in check list in the form of suitable questions against which the responses of the various vendors are validated. For example, Support Service Checklists may have parameters like Performance; System development, Maintenance, Conversion, Training, Back-up, Proximity, Hardware and Software.

ii) Point-Scoring Analysis: Point-scoring analysis provides an objective means of selecting a final system. There are no absolute rules in the selection process, only guidelines for matching user needs with software capabilities. Thus, even for a small business, the evaluators must consider such issues as the company's data processing needs, its in- house computer skills, vendor reputations, software costs, and so forth.

Point Scoring Analysis List

Software Evaluation Criteria	**Possible points**	**Vendor A**	**Vendor B**	**Vendor C**
Does the software meet all mandatory specifications?	10	7	9	6
Will program modifications, if any, be minimal to meet company needs?	10	8	9	7
Does the software contain adequate controls?	10	9	9	8
Is the performance (speed, accuracy, reliability, etc.) adequate?	10	7	9	6
Are other users satisfied with the software?	8	6	7	5
Is the software user-friendly?	10	7	8	6
Can the software be demonstrated and test-driven?	9	8	8	7
Does the software have an adequate warranty?	8	6	7	6
Is the software flexible and easily maintained?	8	5	7	5
Is online inquiry of files and records possible?	10	8	9	7
Will the vendor keep the software up to date?	10	8	8	7
Totals	123	94	106	85

iii) Public Evaluation Reports: Several consultancy as well as independent agencies compare and contrast the hardware and software performance for various manufacturers and publish their reports in this regard. This method has been frequently and usefully employed by several buyers in the past. For those criteria, however, where published reports are not available, reports would have to be made to other methods of validation. This method is particularly useful where the buying staff has inadequate knowledge of facts.

iv) **Benchmarking Problems related Vendor's Solutions:** Benchmarking problems related to vendors' proposals are accomplished by sample programs that represent at least a part of the buyer's primary work load and include considerations and can be current applications that have been designed to represent planned processing needs. That is, benchmarking problems are oriented towards testing whether a solution offered by the vendor meets the requirements of the job on hand of the buyer.

v) **Testing Problems:** Test problems disregard the actual job mix and are devised to test the true capabilities of the hardware, software or system. For example, test problems may be developed to evaluate the time required to translate the source code (program in an assembly or a high level language) into the object code (machine language), response time for two or more jobs in multi-programming environment, overhead requirements of the operating system in executing a user program, length of time required to execute an instruction, etc. The results, achieved by the machine can be compared and price performance judgment can be made. It must be borne in mind, however that various capabilities to be tested would have to be assigned relative weight- age.

5.15. System Development Programming Techniques and Languages

1) A good coded application and programs should have the following characteristics:

a) Reliability: It refers to the consistency with which a program operates over a period of time. However, poor setting of parameters and hard coding of some data, subsequently could result in the failure of a program after some time.

b) Robustness (Strength): It refers to the applications' strength to uphold its operations in adverse situations by taking into account all possible inputs and outputs of a program in case of least likely situations.

c) Accuracy: It refers not only to 'what program is supposed to do', but should also take care of 'what it should not do'. The second part becomes more challenging for quality control personnel and auditors.

d) Efficiency: It refers to the performance per unit cost with respect to relevant parameters and it should not be unduly affected with the increase in input values.

e) Usability: It refers to a user-friendly interface and easy-to-understand internal/external documentation.

f) Readability: It refers to the ease of maintenance of program even in the absence of the program developer.

2) Other related aspects of this phase are given as follows:

a) Program Coding Standards: The logic of the program outlined in the flowcharts is converted into program statements or instructions at this stage. For each language, there are specific rules concerning format and syntax. Syntax means vocabulary, punctuation and grammatical rules available in the language manuals that the programmer has to follow strictly and pedantically. Coding standards minimize the system development setbacks due to programmer turnover. Coding standards provide simplicity, interoperability, compatibility, efficient utilization of resources and least processing time.

b) Programming Language: Application programs are coded in the form of statements or instructions and the same is converted by the compiler to object code for the computer to understand and execute. The programming languages commonly used are given as follows:

i) High level general purpose programming languages such as COBOL and C;

ii) Object oriented languages such as C++, JAVA etc.;
iii) Scripting language such as JAVAScript, VBScript; and
iv) Decision Support or Logic Programming languages such as LISP and PROLOG.

In general, language selection may be made on the basis of application area; algorithmic complexity; environment in which software has to be executed; performance consideration; data structure complexity; knowledge of software development staff; and capability of in-house staff for maintenance.

c) Program Debugging: Debugging is the most primitive form of testing activity, which refers to correcting programming language syntax and diagnostic errors so that the program compiles cleanly. A clean compile means that the program can be successfully converted from the source code written by the programmer into machine language instructions. Debugging can be a tedious task consisting of following four steps:
i) Giving input the source program to the compiler,
ii) Letting the compiler to find errors in the program,
iii) Correcting lines of code that are erroneous, and
iv) Resubmitting the corrected source program as input to the compiler.

d) Testing the Programs: A careful and thorough testing of each program is imperative to the successful installation of any system. The programmer should plan the testing to be performed, including testing of all the possible exceptions. The test plan should require the execution of all standard processing logic based on chosen testing strategy/techniques. A log of test results and all conditions successfully tested should be kept. The log will prove invaluable in finding the faults and debugging.

e) Program Documentation: The writing of narrative procedures and instructions for people, who will use software is done throughout the program life cycle. The documentation should be prepared in such a way that the user can clearly understand the instructions.

f) Program Maintenance: The requirements of business data processing applications are subject to periodic change. This calls for modification of various programs. There are usually separate categories of programmers called maintenance programmers, who are entrusted with this task.

5.16. System Testing

Testing is a process used to identify the correctness, completeness and quality of developed computer software. Testing should systematically uncover different classes of errors in a minimum amount of time with a minimum amount of efforts. The data collected through testing can also provide an indication of the software's reliability and quality. However, testing cannot show the absence of defect, it can only show that software defects are present. Different levels/facets of Testing are described as follows.

1) Unit Testing: Unit testing is a software verification and validation method in which a programmer tests if individual units of source code are fit for use. A unit is the smallest testable part of an application, which may be an individual program, function, procedure, etc. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended. The goal of unit testing is to isolate each component of the program and show that they are correct. A unit test provides a strict, written contract that the piece of code must satisfy.

There are five categories of tests that a programmer typically performs on a program unit. Such typical tests are described as follows:

a) **Functional Tests:** Functional Tests check 'whether programs do, what they are supposed to do or not'. The test plan specifies operating conditions, input values, and expected results, and as per this plan, programmer checks by inputting the values to see whether the actual result and expected result match.

b) **Performance Tests:** Performance Tests should be designed to verify the response time, the execution time, the throughput, primary and secondary memory utilization and the traffic rates on data channels and communication links.

c) **Stress Tests:** Stress testing is a form of testing that is used to determine the stability of a given system or entity. It involves testing beyond normal operational capacity in order to observe the results. These tests are designed to overload a program in various ways. The purpose of a stress test is to determine the limitations of the program.

d) **Structural Tests:** Structural Tests are concerned with examining the internal processing logic of a software system. For example, if a function is responsible for tax calculation, the verification of the logic is a structural test.

e) **Parallel Tests:** In Parallel Tests, the same test data is used in the new and old system and the output results are then compared.

In terms of techniques, Unit Testing is classified as Static Analysis Testing and Dynamic Testing. Such typical testing techniques are elaborated as follows:

a) **Static Testing:** Static Analysis Tests are conducted on source programs and do not normally require executions in operating conditions. Typical static analysis techniques include the following:
   i) **Desk Check:** This is done by the programmer him/herself. S/he checks for logical syntax errors, and deviation from coding standards.
   ii) **Structured Walk Through:** The application developer leads other programmers to scan through the text of the program and explanation to uncover errors.
   iii) **Code Inspection:** The program is reviewed by a formal committee. Review is done with formal checklists.

b) **Dynamic Analysis Testing:** Such testing is normally conducted through execution of programs in operating conditions. Typical techniques for dynamic testing and analysis include the following**:**
   i) **Black Box Testing:** Black Box Testing takes an external perspective of the test object, to derive test cases. These tests can be functional or non-functional, though usually functional. The test designer selects typical inputs including simple, extreme, valid and invalid input-cases and executes to uncover errors. There is no knowledge of the test object's internal structure.
   This method of test design is applicable to all levels of software testing i.e. unit, integration, functional testing, system and acceptance. The higher the level the more one is forced to use black box testing to simplify.

   ii) **White Box Testing:** It uses an internal perspective of the system to design test cases based on internal structure. It requires programming skills to identify all paths through the software. The tester chooses test case inputs to exercise paths through the code and determines the appropriate outputs. It is applicable at the unit, integration and system levels of the testing process, it is typically applied to the unit. While it normally tests paths within a unit, it can also test paths between units during integration, and between subsystems during a system level test. After obtaining a clear picture of the internal workings of a product, tests can be conducted to ensure that the internal operation of the product conforms to specifications and all the internal components are adequately exercised.

iii) **Gray Box Testing:** It is a software testing technique that uses a combination of black box testing and white box testing. In gray box testing, the tester applies a limited number of test cases to the internal workings of the software under test. In the remaining part of the gray box testing, one takes a black box approach in applying inputs to the software under test and observing the outputs.

2) **Integration Testing:** Integration testing is an activity of software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before system testing with an objective to evaluate the validity of connection of two or more components that pass information from one area to another. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing. This is carried out in the following two manners:
   a) **Bottom-up Integration:** It is the traditional strategy used to integrate the components of a software system into a functioning whole. It consists of unit testing, followed by sub- system testing, and then testing of the entire system. Bottom-up testing is easy to implement as at the time of module testing, tested subordinate modules are available.

   b) **Top-down Integration:** It starts with the main routine, and stubs are substituted, for the modules directly subordinate to the main module. An incomplete portion of a program code that is put under a function in order to allow the function and the program to be compiled and tested, is referred to as a stub. A stub does not go into the details of implementing details of the function or the program being executed. Once the main module testing is complete, stubs are substituted with real modules one by one, and these modules are tested with stubs. This process continues till the atomic modules are reached.

   c) **Regression Testing:** Each time a new module is added or any modification made in the software, it changes. New data flow paths are established, new I/O may occur and new control logic is invoked. These changes may cause problems with functions that previously worked flawlessly. In the context of the integration testing, the regression tests ensure that changes or corrections have not introduced new faults. The data used for the regression tests should be the same as the data used in the original test.

3) **System Testing:** It is a process in which software and other system elements are tested as a whole. System testing begins either when the software as a whole is operational or when the well-defined subsets of the software's functionality have been implemented. The purpose of system testing is to ensure that the new or modified system functions properly. The types of testing that might be carried out are as follows:
   a) **Recovery Testing:** This is the activity of testing 'how well the application is able to recover from crashes, hardware failures and other similar problems'. Recovery testing is the forced failure of the software in a variety of ways to verify that recovery is liable to be properly performed, in actual failures

   b) **Security Testing:** This is the process to determine that an Information System protects data and maintains functionality as intended or not. The six basic security concepts that need to be covered by security testing are – confidentiality, integrity, availability authentication, authorization, and non-repudiation.

   c) **Stress or Volume Testing:** Stress testing is a form of testing that is used to determine the stability of a given system or entity. It involves testing beyond normal operational capacity, often to a breaking point, in order to observe the results. Stress testing may be performed by testing the application with large quantity of data during peak hours to test its performance.

d) Performance Testing: In the computer industry, software performance testing is used to determine the speed or effectiveness of a computer, network, software program or device. This testing technique compares the new system's performance with that of similar systems using well defined benchmarks.

4) Final Acceptance Testing: During this testing, it is ensured that the new system satisfies the quality standards adopted by the business and the system satisfies the users. Thus, the final acceptance testing has two major parts:

a) Quality Assurance Testing: It ensures that the new system satisfies the prescribed quality standards and the development process is as per the organization's quality assurance policy, methodology and prescriptions.

b) User Acceptance Testing: It ensures that the functional aspects expected by the users have been well addressed in the new system. There are two types of the user acceptance testing described as follows:

i) Alpha Testing: This is the first stage, often performed by the users within the organization by the developers, to improve and ensure the quality/functionalities as per users satisfaction.

ii) Beta Testing: This is the second stage, generally performed after the deployment of the system. It is performed by the external users, during the real life execution of the project. It normally involves sending the product outside the development environment for real world exposure and receives feedback for analysis and modifications, if any.

5) Internal Testing Controls: There are several controls that can be exercised internally to assure the testing phase quality and efficiency. Though it varies from one organization to another, some of the generic key control aspects appear to be addressed by the responses to following queries:

a) Whether the test-suite prepared by the testers includes the actual business scenarios?
b) Whether test data used covers all possible aspects of system?
c) Whether CASE tools like 'Test Data Generators' have been used?
d) Whether test results have been documented?
e) Whether test have been performed in their correct order?
f) Whether modifications needed based on test results have been done?
g) Whether modifications made have been properly authorized and documented.

5.17. System Implementation

1) Some of the generic activities involved in system implementation stage are described briefly as follows:

a) Equipment Installation: The hardware required to support the new system is selected prior to the implementation phase. The necessary hardware should be ordered in time to allow for installation and testing of equipment during the implementation phase. An installation checklist should be developed at this time with operating advice from the vendor and system development team. In those installations, where people are experienced in the installation of the same or similar equipment, adequate time should be scheduled to allow completion of the following activities:

b) Site Preparation: An appropriate location/ambiance as prescribed and the typical equipment must be availed to provide an operating environment for the equipment that will meet the vendor's temperature, humidity and dust control specifications etc.

c) Installation of New Hardware / Software: The equipment must be physically installed by the manufacturer, connected-to the power source and wired to communication lines, if required. If the new system interfaces with the other systems or is distributed across multiple software platforms, some final commissioning tests of the operating environment may be desirable to prove end to end connectivity.

d) Equipment Checkout: The equipment must be turned on for testing under normal operating conditions. Though the routine 'diagnostic tests' should be run by the vendor, the implementation in-house team should devise and run extensive tests of its own to ensure that equipment functionalities in actual working conditions.

2) Training Personnel: A system can succeed or fail depending on the way it is operated and used. Therefore, the quality of training received by the personnel involved with the system in various capacities helps or hinders the successful implementation of information system. Thus, training is a major component of systems implementation. When a new system is acquired, which often involves new hardware and software, both users and computer professionals generally need some type of training. Such training structure should be highly formalized and be based business process executions with actual data.

3) System Change-Over Strategies: Conversion or changeover is the process of changing over or shifting over from the old system (may be the manual system) to the new system. It requires careful planning to establish the basic approach to be used in the actual changeover, as it may put many resources/assets/operations at risk. The Four types of popular implementation strategies are described as follows:

a) Direct Implementation / Abrupt Change-Over: This is achieved through an abrupt takeover – an all or no approach. With this strategy, the changeover is done in one operation, completely replacing the old system in one go. which usually takes place on a set date, often after a break in production or a holiday period so that time can be used to get the hardware and software for the new system installed without causing too much disruption.

b) Phased Changeover: With this strategy, implementation can be staged with conversion to the new system taking place gradually. For example, some new files may be converted and used by employees whilst other files continue to be used on the old system i

c) Pilot Changeover: With this strategy, the new system replaces the old one in one operation but only on a small scale. Any errors can be rectified or further beneficial changes can be introduced and replicated throughout the whole system in good time with the least disruption. For example - it might be tried out in one branch of the company or in one location. If successful then the pilot is extended until it eventually replaces the old system completely.

d) Parallel Changeover: This is considered the most secure method with both systems running in parallel over an introductory period. The old system remains fully operational while the new systems come online. With this strategy, the old and the new system are both used alongside each other, both being able to operate independently. If all goes well, the old system is stopped and new system carries on as the only system.

4) Such requisite changeover or Conversion includes all those activities, which must be completed to successfully convert from the previous system to the new information system. Fundamentally these technical activities can be classified as follows:

a) Procedure Conversion: Operating procedures should be carefully completed with sufficient-enough documentation for the new system. Before any parallel or conversion activities can start, operating procedures must be clearly spelled out for personnel in the functional areas undergoing changes. Information on input, data files, methods, procedures, output, and internal control must be presented in clear, concise and understandable terms for the average reader. Written operating procedures must be supplemented by oral communication during the training sessions on the system change.

b) File Conversion: Because large files of information must be converted from one medium to another. In order for the conversion to be as accurate as possible, file conversion programs must be thoroughly tested. Adequate control, such as record counts and control totals, should be required output of the conversion program. The existing computer files should be kept for a period of time until sufficient files are accumulated for back up. This is necessary in case, the files must be reconstructed from scratch after a "bug" is discovered later in the conversion routine.

c) System conversion: After on-line and off-line files have been converted and the reliability of the new system has been confirmed for a functional area, daily processing can be shifted from the existing information system to the new one. All transactions initiated after this time are processed on the new system. System development team members should be present to assist and to answer any questions that might develop. Consideration should be given to operating the old system for some more time to permit checking, matching and balancing the total results of both systems.

d) Scheduling Personnel and Equipment: Schedules should be set up by the system manager in conjunction with departmental managers of operational units serviced by the equipment. The master schedule for next period/month should provide sufficient computer time to handle all required processing.

5.18. Post Implementation Review and Systems Maintenance

In order to assess, review and assure the complete working solution, a number of activities may be planned. A well-formalized review must be undertaken including some of the systems maintenance activities, such as adding new data elements, modifying reports, adding new reports; and changing calculations.

1) Post Implementation Review: A Post Implementation Review answers the question “Did we achieve what we set out to do in business terms?” It examines the efficacy of all elements of the working business solution to see if further improvements can be made to optimize the benefit delivered. A Post-Implementation Review should be scheduled some time after the solution has been deployed. Typical periods range from 6 weeks to 6 months, depending on the type of solution and its environment. There are two basic dimensions of Information system that should be evaluated. The first dimension is concerned with whether the newly developed system is operating properly. The other dimension is concerned with whether the user is satisfied with the information system with regard to the reports supplied by it. Typical evaluations include the following:

a) Development Evaluation: Evaluation of the development process is primarily concerned with whether the system was developed on schedule and within budget. It requires schedules and budgets to be established in advance and that records of actual performance and cost be maintained. However, it may be noted that very few information systems have been developed on schedule and within budget.

b) Operational Evaluation: The evaluation of the information system's operation pertains to whether the hardware, software and personnel are capable to perform their duties. It tries to answer the questions related to functional aspects of the system. Such an evaluation is relatively straightforward if evaluation criteria are established in advance.

c) Information Evaluation: An information system should also be evaluated in terms of information it provides or generates. This aspect of system evaluation is difficult and it cannot be conducted in a quantitative manner, as is the case with development and operational evaluations.

2) **System Maintenance:** Maintaining the system is an important aspect of SDLC. As key personnel change positions in the organization, new changes will be implemented, which will require system updates at regular intervals. Most of the information systems require at least some modification after development. The need for modification arises from a failure to anticipate/capture all the requirements during system analysis/design and/or from changing organizational requirements. Maintenance can be categorized in the following ways:
   a) **Scheduled Maintenance:** Scheduled maintenance is anticipated and can be planned for operational continuity and avoidance of anticipated risks. For example, the implementation of a new inventory coding scheme can be planned in advance, security checks may be promulgated etc.

   b) **Rescue Maintenance:** Rescue maintenance refers to previously undetected malfunctions that were not anticipated but require immediate troubleshooting solution. A system that is properly developed and tested should have few occasions of rescue maintenance.

   c) **Corrective Maintenance:** Corrective maintenance deals with fixing bugs in the code or defects found during the executions. A defect can result from design errors, logic errors coding errors, data processing and system performance errors. The need for corrective maintenance is usually initiated by bug reports drawn up by the end users.

   d) **Adaptive Maintenance:** Adaptive maintenance consists of adapting software to changes in the environment, such as the hardware or the operating system. The term environment in this context refers to the totality of all conditions and influences, which act from outside upon the system,

   e) **Perfective Maintenance:** Perfective maintenance mainly deals with accommodating to the new or changed user requirements and concerns functional enhancements to the system and activities to increase the system's performance or to enhance its user interface.

   f) **Preventive Maintenance:** Preventive maintenance concerns with the activities aimed at increasing the system's maintainability, such as updating documentation, adding comments, and improving the modular structure of the system. The long-term effect of corrective, adaptive and perfective changes increases the system's complexity.

5.19. Operation Manuals

It is typical user's guide, also commonly known as Operations Manual. Moreover, it may be a technical communication document intended to give assistance to people using a particular system. It is usually written by technical writers, although user guides are written by programmers, product or project managers, or other technical staff, particularly in smaller companies. Operation manuals are most commonly associated with electronic goods, computer hardware and software.

5.20. Auditor's Role in SDLC

The audit of systems under development can have three main objectives.

- To provide an opinion on the efficiency, effectiveness, and economy of project management.
- To assess the extent to which the system being developed provides for adequate audit trails and controls to ensure the integrity of data processed and stored;
- And the effectiveness of controls being enacted for the management of the system's operation.

In order to achieve these goals, an auditor has to attend project and steering committee meetings and examine project control documentation and conducting interviews. This is to ensure 'what project control standards are to be complied with, and determining the extent to which compliance is being achieved. For addressing the second objective, the auditor can examine system documentation such as functional specifications to arrive at an opinion on controls. The auditor's opinion will be based on the degree to which the system satisfies the general control objectives that any information system should meet. A list of such objectives should be provided to the auditor. The auditor should

provide a list of the standard controls, over such operational concerns as response time, CPU usage, and random access space availability that the auditor has used as assessment criteria.

Some of the control considerations for an auditor include the following:

1) Documented policy and procedures;
2) Established Project team with all infrastructure and facilities ;
3) Developers/ IT managers are trained on the procedures ;
4) Appropriate approvals are being taken at identified mile-stones;
5) Development is carried over as per standards, functional specifications;
6) Separate test environment for development/ test/ production / test plans;
7) Design norms and naming conventions are as per standards and are adhered to;
8) Business owners testing and approval before system going live;
9) Version control on programs;
10) Source Code is properly secured;
11) Adequate audit trails are provided in system; and
12) Appropriateness of methodologies selected

Further, Post-Implementation Review is performed to determine whether the system adequately meets earlier identified business requirements and needs (in feasibility studies or Requirements Specifications). Auditors should be able to determine if the expected benefits of the new system are realized and whether users are satisfied with the new system. In post implementation review, auditors need to review which of the SDLC phases have not met desired objectives and whether any corrective actions were taken. If there are differences between expectations and actual results, auditors need to determine the reasons for the same. Such discrepancies may be due to incomplete user requirements or any other shortcoming. Such reasons can help auditors to evaluate the current situation and offer guidelines for future projects.