System Development Life Cycle Methodology

Learning Objectives:

- To introduce the general concepts of various approaches of systems development, their framework, advantages and disadvantages;
- To explain in detail the phases involved in Systems Development Life Cycle (SDLC);
- To understand the key issues while acquiring or developing system for achieving goals set;
- To discuss in detail various System Development Tools like – DFD, Decision Tree, Flowcharts etc.; and
- To understand the auditors' role in SDLC.

2.1 Introduction

Information systems serve many different purposes, ranging from the processing of business transactions - to provide information needed to decide recurring issues, assisting senior officials with difficult strategy formulation, and linking office information and corporate data. But how do such complex information systems come into existence? Of course, through people. Technology has developed at a rapid pace but the most important aspect of any system is human know-how and the use of ideas to harness the computer so that it performs the required tasks. This process is essentially what system development is all about. To be of any use, a computer-based information system must function properly, be easy to use, and suit the organization for which it has been designed. If a system helps people to work more efficiently they will use it. If not, they will surely avoid it.

2.2 System Development Process

In business, systems 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: System Analysis and System Design.

- **System Analysis** is the process of gathering and interpreting facts, diagnosing problems, and using the information to recommend improvements to the system.

- **System Design** is the process of planning a new business system or one to replace or complement an existing system.

But before planning can be done, one must thoroughly understand the old system and determine how computers can be used (if at all) to make its operation more effective.
2.2 Information Systems Control and Audit

Example: Consider stockroom operations of a clothing store. What measures can be taken to control its inventory and gain access to more up-to-date information about stock levels and reordering in a better way.

Solution: The Stores Manager asks a System Analyst to organize the stockroom operations. Before an analyst can design a system to capture data, update files and produce reports, he needs to know more about:

- how the store currently operates,
- what forms are being used to store information manually, such as requisitions, purchase orders and invoices etc,
- what reports are being produced and how they are being used, etc.

To proceed, an analyst seeks information about lists of reorder notices, outstanding purchase orders, records of stock on hand, and other reports. He tries to understand how the existing system works and more specifically what the flow of information through the system looks like and assesses as carefully as possible, what the future need of the system will be and what changes should be considered to meet these needs. He may recommend alternatives for improving the situation which then management decides to accept or reject. The plan includes all system design features, file specifications, operating procedures, and design features, and equipment and personnel requirements. The system design is like the blue print for a building, it specifies all the features that should be there in the finished product.

2.2.1 Achieving System Development Objectives: There are many reasons why organizations fail to achieve their systems development objectives. Some of them are as follows:

- Lack of senior management support and involvement in information systems development. 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.

- Shifting user needs. User requirements for information technology are constantly changing. As these changes accelerate, there will be more requests for 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 purposes might change since the development process began.

- Development of strategic systems. Because strategic decision making is unstructured, the requirements, specifications, and objectives for such development projects are difficult to define.

- New technologies. When an organization tries to create a competitive advantage by applying advance Information technology, it generally finds that attaining system development objectives is more difficult because personnel are not as familiar with the technology.
Lack of standard project management and systems 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.

Overworked or under-trained development staff. In many cases, systems developers often lack sufficient education background. Furthermore, many companies do little to help their development personnel stay technically sound. Currently in these organizations, a training plan and training budget do not exist.

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.

Lack of user participation. Users must participate in the development effort to define their requirements, feel ownership for project success, and work to resolve development problems. User participation also helps reduce user resistance to change.

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.

To overcome these and other problems, organizations must execute the systems development process efficiently and effectively.

2.2.2 System Development Team: Several people in the organization are responsible for systems development. In large systems, the worth of a particular project is typically decided by a top management level steering committee, usually consisting of a group of key Information Systems services users that acts as a review body for Information Systems plans and applications development. The steering committee ensures that ongoing systems development activities are consistently aimed at satisfying the information requirements of managers and users within the organization. A project management team generally consists of both computer professionals and key users. System analysts are subsequently assigned to determine user requirements, design the system and assist in development and implementation activities. In any systems organization, however, systems designers take a lead role during the design, development and implementation stages.

In end-user developed systems, the end-user is ultimately responsible for the system. Generally, the end-user seeks guidance from information centre personnel while developing the system.

2.2.3 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.
2.3 Systems Development Methodology

A system development methodology is a formalized, standardized, 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. Each of the available methodologies is best suited to specific kinds of projects, based on various technical, organizational, project and team considerations. The methodology is characterized by the following:

- 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.

- 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.

- 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.

- The system must be tested thoroughly prior to implementation to ensure that it meets users’ needs.

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

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

- 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.

Approaches to System Development: Since organizations vary significantly in the way they automate their business procedures, and since each new type of system usually differs from any other, several different system development approaches are often used within an organization. All these approaches are not mutually exclusive, which means that it is possible to perform some prototyping while applying the traditional approach. These approaches are as follows:

(i) Waterfall: Linear framework type

(ii) Prototyping: Iterative framework type

(iii) Incremental: Combination of linear and iterative framework type

(iv) Spiral: Combination linear and iterative framework type

(v) Rapid Application Development (RAD): Iterative Framework Type

(vi) Agile Methodologies

2.3.1 The Traditional / Waterfall Approach / Sequential Approach: The waterfall approach is a traditional development approach in which each developer in a development team works in
different phases. These phases include requirement analysis, specifications and design requirements, coding, final testing, and release. The waterfall approach is used on small projects because it eliminates testing to identify problems early in the process.

In the traditional approach of system development, activities are performed in sequence. Fig. 2.3.1 shows examples of the tasks performed during each phase of the traditional approach. When the traditional approach is applied, an activity is undertaken only when the prior step is fully completed.

**Overview of WaterFall Model**

**Fig. 2.3.1 : Steps in Traditional Approach**

**Framework type :** Linear

**Basic Principles**

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

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

(iii) 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.
2.6 Information Systems Control and Audit

Strengths
(i) Ideal for supporting less experienced project teams and project managers or project teams whose composition fluctuates.
(ii) An orderly sequence of development steps and design reviews help ensure the quality, reliability, adequacy and maintainability of the developed software.
(iii) Progress of system development is measurable.
(iv) Conserves resources.

Weaknesses
(i) Inflexible, slow, costly, and cumbersome due to significant structure and tight controls.
(ii) Project progresses forward, with only slight movement backward.
(iii) Little room for use of iteration, which can reduce manageability if used.
(iv) Depends upon early identification and specification of requirements, yet users may not be able to clearly define what they need early in the project.
(v) Requirement inconsistencies, missing system components and unexpected development needs are often discovered during design and coding.
(vi) Problems are often not discovered until system testing.
(vii) System performance cannot be tested until the system is almost fully coded, and under capacity may be difficult to correct.
(viii) Difficult to respond to changes. Changes that occur later in the life cycle are more costly and are thus discouraged.
(ix) Produces excessive documentation and keeping it updated as the project progresses is time-consuming.
(x) Written specifications are often difficult for users to read and thoroughly appreciate.
(xi) Promotes the gap between users and developers with clear vision of responsibility.

2.3.2 The Prototyping Model: The traditional approach sometimes may take years to analyze, design and implement a system. In order to avoid such delays, organizations are increasingly using prototyping techniques to develop smaller systems such as DSS, MIS and Expert systems. The goal of prototyping approach is to develop a small or pilot version called a prototype of part or all of a system. A prototype is a usable system or system component that is built quickly and at a lesser cost, and with the intention of being modifying or replacing it by a full-scale and fully operational system. As users work with the prototype, they make suggestions about the ways to improve it. These suggestions are then incorporated into another prototype, which is also used and evaluated and so on. 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.
Framework type: Iterative.

Basic Principles: Prototyping can be viewed as a series of four steps, depicted in Fig. 2.3.2 wherein Implementation and Maintenance phases take place once the prototype model is tested and found to meet users’ requirements.

Step 1 - Identify Information System Requirements: In traditional approach, the system requirements have 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.

Step 2 - Develop the Initial Prototype: In this step, the designers create an initial base model and give little or no consideration to internal controls, but instead emphasize such 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.

Step 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.

Step 4 - Obtain User Signoff of the Approved Prototype: At the end of Step 3, 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.

Strengths

(i) Improves both user participation in system development and communication among project stakeholders.
2.8 Information Systems Control and Audit

(ii) 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.

(iii) Potential exists for exploiting knowledge gained in an early iteration as later iterations are developed.

(iv) Helps to easily identify confusing or difficult functions and missing functionality.

(v) May generate specifications for a production application.

(vi) Encourages innovation and flexible designs.

(vii) Provides quick implementation of an incomplete, but functional, application.

(viii) Prototyping requires intensive involvement by the system users. Therefore, it typically results in a better definition of these users’ needs and requirements than does the traditional systems development approach.

(ix) A very short time period (e.g., a week) is normally required to develop and start experimenting with a prototype. This short time period allows system users to immediately evaluate proposed system changes.

(x) 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.

Weaknesses

(i) Approval process and control are not strict.

(ii) 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.

(iii) Requirements may frequently change significantly.

(iv) Identification of non-functional elements us difficult to document.

(v) Designers may prototype too quickly, without sufficient upfront user needs analysis, resulting in an inflexible design with narrow focus that limits future system potential.

(vi) Prototype may not have sufficient checks and balances incorporated.

(vii) 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.

(viii) 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.
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.

In spite of above listed limitations, to some extent, systems analysis and development has been greatly improved by the introduction of prototyping. Prototyping enables the user to take an active part in the systems design, with the analyst acting in an advisory role. Prototyping makes use of the expertise of both the user and the analyst, thus ensuring better analysis and design, and prototyping is a crucial tool in that process.

2.3.3 The Incremental Model

Framework Type: Combination Linear and Iterative.

Basic Principles: The Incremental build 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. It also creates a large initial capital outlay with the subsequent long wait avoided. This model of development also helps ease the traumatic effect of introducing completely new system all at once.

(i) 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.

   or

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

   or

(iii) 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 (ie. Working system).

Fig. 2.3.3 depicts the Incremental Model.
2.10 Information Systems Control and Audit

Strengths

(i) Potential exists for exploiting knowledge gained in an early increment as later increments are developed.

(ii) 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.

(iii) Stakeholders can be given concrete evidence of project status throughout the life cycle.

(iv) More flexible – less costly to change scope and requirements.

(v) Helps to mitigate integration and architectural risks earlier in the project.

(vi) Allows delivery of a series of implementations that are gradually more complete and can go into production more quickly as incremental releases.

(vii) Gradual implementation provides the ability to monitor the effect of incremental changes, isolated issues and make adjustments before the organization is negatively impacted.

Weaknesses

(i) 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.

(ii) Each phase of an iteration is rigid and do not overlap each other.

(iii) Problems may arise pertaining to system architecture because not all requirements are gathered up front for the entire software life cycle.

(iv) Since some modules will be completed much earlier than others, well-defined interfaces are required.

(v) Difficult problems tend to be purchased to the future to demonstrate early success to management.

2.3.4 Spiral Model

Framework Type: Combination Linear and Iterative.

Basic Principles: The Spiral model is a software development process combining elements of both design and prototyping-in-stages, in an effort to combine advantages of top-down and bottom-up concepts. Also known as the Spiral Lifecycle, it is a Systems Development Method (SDM) which combines the features of the prototyping model and the waterfall model. The spiral model is intended for large, expensive and complicated projects.

(i) 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.

(ii) 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.

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

(iv) A second prototype is evolved by a fourfold procedure:
- 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;
- constructing and testing the second prototype.

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.

**Strengths**

(i) Enhance risk avoidance.

(ii) Useful in helping to select the best methodology to follow for development of a given software iteration based on project risk.

(iii) 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.

![Spiral Model](Fig. 2.3.4 : Spiral Model)
2.12 Information Systems Control and Audit

Weaknesses

(i) Challenges to determine the exact composition of development methodologies to use for each iteration around the Spiral.

(ii) Highly customized to each project, and thus is quite complex, limiting reusability.

(iii) A skilled and experienced project manager required to determine how to apply it to any given project.

(iv) No established controls for moving from one cycle to another cycle. Without controls, each cycle may generate more work for the next cycle.

(v) No firm deadlines - cycles continue with no clear termination condition, so there is an inherent risk of not meeting budget or schedule.

2.3.5 Rapid Application Development (RAD)

Framework Type: Iterative.

Basic Principles: 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.

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

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

(iii) 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.

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

(v) 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.

(vi) 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.

(vii) Active user involvement is imperative.

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

(ix) Produces documentation necessary to facilitate future development and maintenance.

(x) Standard systems analysis and design techniques can be fitted into this framework.
**Strengths**

(i) The operational version of an application is available much earlier than with Waterfall, Incremental, or Spiral frameworks.

(ii) Because RAD produces systems more quickly and to a business focus, this approach tends to produce systems at lower cost.

(iii) Quick initial reviews are possible.

(iv) Constant integration isolate problems and encourage customer feedback.

(v) 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.

(vi) Concentrates on essential system elements from user viewpoint.

(vii) Provides the ability to rapidly change system design as demanded by users.

(viii) Produces a tighter fit between user requirements and system specifications.

(ix) Generally produces a dramatic savings in time, money and human effort.

**Weaknesses**

(i) More speed and lower cost may lead to a lower overall system quality.

(ii) Danger of misalignment of developed system with the business due to missing information.

(iii) Project may end up with more requirements than needed (gold-plating).

(iv) Potential for feature creep where more and more features are added to the system over the course of development.

(v) Potential for inconsistent designs within and across systems.

(vi) Potential for violation of programming standards related to inconsistent naming conventions and inconsistent documentation,

(vii) Difficulty with module reuse for future systems.

(viii) Potential for designed system to lack scalability.

(ix) Potential for lack of attention to later system administration needs built into system.

(x) High cost of commitment on the part if key user personnel.

(xi) Formal reviews and audits are more difficult to implement than for a complete system.

(xii) Tendency for difficult problems to be pushed to the future to demonstrate early success to management.

(xiii) Since some modules will be completed much earlier than others, well –defined interfaces are required.
2.3.6 Agile Methodologies

This is a group 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.

Basic Principles

Following are the key principles of this methodology:

- Customer satisfaction by rapid delivery of useful software;
- Welcome changing requirements, even late in development;
- Working software is delivered frequently (weeks rather than months);
- Working software is the principal measure of progress;
- Sustainable development, able to maintain a constant pace;
- Close, daily co-operation between business people and developers;
- Face-to-face conversation is the best form of communication (co-location);
- Projects are built around motivated individuals, who should be trusted;
- Continuous attention to technical excellence and good design;
- Simplicity;
- Self-organizing teams; and
- Regular adaptation to changing circumstances.

Strengths

- Agile methodology has the concept of an adaptive team, which is able to respond to the changing requirements.
- The team does not have to invest time and effort and finally find that by the time they delivered the product, the requirement of the customer has changed.
- Face to face communication and continuous inputs from customer representative leaves no space for guesswork.
- The documentation is crisp and to the point to save time.
- The end result is the high quality software in least possible time duration and satisfied customer.

Weaknesses

- 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.
- There is lack of emphasis on necessary designing and documentation.
- 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.
- Agile requires more re-work. Because of 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.
• The project can easily get taken off track if the customer representative is not clear about the final outcome that they want.

• Only senior programmers are capable of taking the kind of decisions required during the development process. Hence, it has no place for newly appointed programmers, unless combined with experienced resources.

• Agile lacks the attention to outside integration. Because Agile teams often do not invest the time in identifying and designing the integration points with other systems in advance, the need for an integration point can become a last-minute surprise that often requires re-work, additional time, removal from scope, or a poor-quality product.

2.4 System Development Life Cycle (SDLC)

The System Development Life Cycle (SDLC) framework provides system designers and developers to follow a sequence of activities. It consists of a set of steps or phases in which each phase of the SDLC uses the results of the previous one.

The SDLC is document driven which means that at crucial stages during the process, documentation is produced. A phase of the SDLC is not complete until the appropriate documentation or artifact is produced. These are sometimes referred to as deliverables. A deliverable may be a substantial written document, a software artifact, a system test plan or even a physical object such as a new piece of technology that has been ordered and delivered. This feature of the SDLC is critical to the successful management of an IS project.

The SDLC can also be viewed from a more process oriented perspective. This emphasizes the parallel nature of some of the activities and presents activities such as system maintenance as an alternative to a complete re-design of an existing system. The advantages of this system are as follows:

• Better planning and control by project managers.

• Compliance to prescribed standards ensuring better quality.

• Documentation that SDLC stresses on is an important measure of communication and control.

• The phases are important milestones and help the project manager and the user for review and signoff.

From the perspective of the IS Audit, the following are the possible advantages:

(i) The IS auditor can have clear understanding of the various phases if the SDLC on the basis of the detailed documentation created during each phase of the SDLC.

(ii) The IS Auditor on the basis of his examination, can state in his report about the compliance by the IS management of the procedures, if any, set by the management.

(iii) The IS Auditor, if has a technical knowledge and ability of the area of SDLC, can be a guide during the various phases of SDLC.

(iv) The IS auditor can provide an evaluation of the methods and techniques used through the various development phases of the SDLC.

Risks Associated with SDLC: Some of the shortcomings of the SDLC are as follows:

(i) The development team may find it cumbersome.
(ii) The users may find that the end product is not visible for a long time.
(iii) The rigidity of the approach may prolong the duration of many projects.
(iv) IT may not be suitable for small and medium sized projects.

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 of as a set of activities that analysts, designers and users carry out to develop and implement an information system. In most business situations, these activities are all 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.

Table 2.4.1 list all the phases involved in the System Development Life Cycle.

<table>
<thead>
<tr>
<th>PHASE NO.</th>
<th>PHASE NAME</th>
<th>NATURE OF ACTIVITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Preliminary Investigation</td>
<td>Determining and evaluating the strategic benefits of the system and ensure that the solution fits the business strategy. Includes cost-benefit analysis of the proposed system.</td>
</tr>
<tr>
<td>2.</td>
<td>Systems Requirements Analysis</td>
<td>Analyzing the type of the system on the basis of the users requirements.</td>
</tr>
<tr>
<td>3.</td>
<td>Systems Design</td>
<td>Designing the system in terms of user interface data storage and data processing functions on the basis of the requirement phase by developing the system flowcharts, system and data flow diagrams, screens and reports.</td>
</tr>
<tr>
<td>4.</td>
<td>Systems Development/Programming</td>
<td>Programming the system as designed and conduct the continuous testing and debugging.</td>
</tr>
<tr>
<td>5.</td>
<td>Systems Testing</td>
<td>Various kinds of testing is conducted before the developed system is implemented. This includes Unit Testing, Integration Testing and System Testing etc.</td>
</tr>
<tr>
<td>6.</td>
<td>Systems Implementation</td>
<td>Final Testing and quality of controls audit, acceptance by management and user before migration of the system to the live environment and data conversion from legacy system to the new system.</td>
</tr>
<tr>
<td>7.</td>
<td>Post Implementation Review and Maintenance</td>
<td>Continuous evaluation of the system as it functions in the live environment and its updation. Maintenance includes continuous evaluation of the system as it functions in the live environment and its updation.</td>
</tr>
</tbody>
</table>

Table 2.4.1 : Phases in System Development Life Cycle
2.5 The Preliminary Investigation

Objective: 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.

A preliminary investigation is normally initiated by some sort of system request. The steps involved in the preliminary investigation phase are as follows:

(i) Identification of Problem
(ii) Identification of objective
(iii) Delineation of scope
(iv) Feasibility Study

The following issues are typically addressed in the Feasibility Study:

(i) Determine whether the solution is as per the business strategy.
(ii) Determine whether the existing system can rectify the situation without a major modification.
(iii) Define the time frame for which the solution is required.
(iv) Determine the approximate cost to develop the system.
(v) Determine whether the vendor product offers a solution to the problem.


2.5.1 Identification of Problem: The first step in an application development is to define the problem clearly and precisely which is done only after 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.

For instance, personnel in a functional area may feel that an existing system is outdated or a manager might want access to specific new information that he claims will lead to better decisions. Shifting business requirements, changing organizational environments, and evolving information technology may render systems ineffective or inefficient. Whatever may be the reason, managers and users may feel compelled to submit a request for a new system to the IS department. If the need seems genuine, a system analyst is assigned to make a preliminary investigation who submits all proposals to the steering committee for evaluation to identify those projects that are most beneficial to the organization.

Thus it can be concluded that the purpose of the preliminary investigation is to evaluate the project request. It is neither a designed study, nor it includes the collection of details to completely describe the business system. Rather it relates to collection of information that permits committee members to evaluate the merits of the project request and make an informed judgment about the feasibility of the proposed project.
The analyst working on the preliminary investigation should accomplish the following objectives:

- Clarify and understand the project request.
- Determine the size of the project.
- Determine the technical and operational feasibility of alternative approaches.
- Assess costs and benefits of alternative approaches.
- Report findings to the management with recommendation outlining the acceptance or rejection of the proposal.

2.5.2 Identification of Objective: After the identification of the problem, it is easy to work out 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 its objective was “to introduce a system wherein intending passengers could book a ticket from source to destination, faster than in real-time.”

2.5.3 Delineation of Scope: The scope of a solution defines its boundaries. It should be clear and comprehensible to the user management stating 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.

The following questions should be answered while stating the scope:

(i) **Functionality requirements**: What functionalities will be delivered through the solution?
(ii) **Data to be processed**: What data is required to achieve these functionalities?
(iii) **Control requirements**: What are the control requirements for this application?
(iv) **Performance requirements**: What level of response time, execution time and throughput is required?
(v) **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?
(vi) **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.
(vii) **Reliability requirements**: Reliability of an application is measured by its ability to remain uncorrupted in the face of inadvertent / deliberate misuse. The reliability required for an application depends on its criticality and the user profile.

While eliciting information to delineate the scope, few aspects need to be kept in mind:

- Different users will represent the problem and required solution in different ways. The system developer should elicit the need from the initiator of the project alternately called champion or executive sponsor of the project, addressing his concerns should be the basis of the scope.
• While the initiator of the project may be a member of the senior management, the actual users may be from the operating levels in an organization. An understanding of their profile helps in designing appropriate user interface features.

• While presenting the proposed solution for a problem, the development organization has to clearly quantify the economic benefits to the user organization. The information required has to be gathered at this stage. For example - when a system is proposed for Road tax collection, data on the extent of collection and defaults is required to quantify benefits that will result to the Transport Department.

• It is also necessary to understand the impact of the solution on the organization- its structure, roles and responsibilities. Solutions which have a wide impact are likely to meet with greater resistance. ERP implementation in organizations is a classic example of change management requirement. Organizations that have not been able to handle this have had a very poor ERP implementation record, with disastrous consequences.

• While economic benefit is a critical consideration when deciding on a solution, there are several other factors that have to be given weight-age too. These factors have to be considered from the perspective of the user management and resolved. For example- in a security system, how foolproof it is, may be a critical a factor like the economic benefits that entail.

The two primary methods with the help of which the scope of the project can be analyzed are 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. For example, to review an inventory system proposal, the analyst will try to know how the inventory department operates and who are the managers and supervisors. Analysts can usually learn these details by examining organization charts and studying written operating procedures.

(ii) **Conducting Interviews** : Written documents tell the analyst how the systems should operate, but they may not include enough details to allow a decision to be made about the merits of a systems proposal, nor do they present users' views about current operations. To learn these details, analysts use 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.

2.5.4 **Feasibility Study**: After possible solution options are identified, project feasibility - 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 :

• **Technical** : Is the technology needed available?
2.20 Information Systems Control and Audit

- **Financial**: Is the solution viable financially?
- **Economic**: Return on Investment?
- **Schedule / Time**: Can the system be delivered on time?
- **Resources**: Are human resources reluctant for the solution?
- **Operational**: How will the solution work?
- **Behavioral**: Is the solution going to bring any adverse effect on quality of work life?
- **Legal**: Is the solution valid in legal terms?

(i) **Technical Feasibility**: 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:

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

Some of the technical issues to be considered are given in the Table 2.5.1 below.

<table>
<thead>
<tr>
<th>Design Considerations</th>
<th>Design Alternatives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communications channel configuration</td>
<td>Point to point, multidrop, or line sharing</td>
</tr>
<tr>
<td>Communications channels</td>
<td>Telephone lines, coaxial cable, fiber optics, microwave, or satellite</td>
</tr>
<tr>
<td>Communications network</td>
<td>Centralized, decentralized, distributed, or local area</td>
</tr>
<tr>
<td>Computer programs</td>
<td>Independent vendor or in-house</td>
</tr>
<tr>
<td>Data storage medium</td>
<td>Tape, floppy disk, hard disk, or hard copy</td>
</tr>
<tr>
<td>Data storage structure</td>
<td>Files or database</td>
</tr>
<tr>
<td>File organization and access</td>
<td>Direct access or sequential files</td>
</tr>
<tr>
<td>Input medium</td>
<td>Keying, OCR, MICR, POS, EDI, or voice recognition</td>
</tr>
</tbody>
</table>
Table 2.5.1: Technical issues while Systems Design

(ii) **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.

(iii) **Economic Feasibility/Cost-Benefit Analysis**: 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.

(iv) The cost if nothing changes (i.e., the proposed system is not developed)

**Estimating costs and benefits**: After possible solution options are identified, an analyst should make a primary estimate of each solution's costs and benefits.

**Cost**: System costs can be subdivided into **Development, Operational** and **Intangible costs**.

- **Development costs** for a computer based information system include costs of the system development process such as - salaries of the system analysts and computer programmers who design and program the system; cost of converting and preparing data files and preparing systems manual and other supportive documents; cost of preparing new or expanded computer facilities; cost of testing and documenting the system, training employees, and other start up costs.
2.22 Information Systems Control and Audit

- **Operating costs** of a computer based information system include - hardware/software rental or depreciation charges; salaries of computer operators and other data processing personnel who will operate the new system; salaries of system analysts and computer programmers who perform the system maintenance function; cost of input data preparation and control; cost of data processing supplies; and Cost of maintaining proper physical facilities including power, light, heat, air conditioning, building rental or other facility charges and equipment and building maintenance charges, overhead charges of the business firm.

- **Intangible costs** are costs 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. Customer sales and goodwill may be lost by errors made during the installation of a new system. Such costs are difficult to measure in rupees but are directly related to the introduction and operation of information system.

**Benefits**: The benefits which result from developing new or improved information systems that utilize EDP can be subdivided into tangible and intangible benefits. Tangible benefits are those that can be accurately measured and are directly related to the introduction of a new system, such as decrease in data processing cost. Intangible benefits such as improved business image are harder to measure and define.

(iv) **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. For example, if a design team projects that it will take 16 months for a particular system design to become fully functional, the steering committee may reject the proposal in favor of a simpler alternative that the company can implement in a shorter time frame.

(v) **Resources Feasibility**: This focuses on human resources. Implementing sophisticated software solutions becomes difficult in non-metro locations because of the reluctance of skilled personnel to move to such locations.

(vi) **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 are stated below:

- Is there sufficient support for the system from management and from users?
- Are current business methods acceptable to users?
- Have the users been involved in planning and development of the project?
- 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?
- Will individual performance be poorer after implementation than before?

This analysis may involve a subjective assessment of the political and managerial environment in which the system will be implemented. In general, the greater the requirements
for change in the user environment in which the system will be installed, the greater the risk of implementation failure.

(vii) **Behavioral Feasibility**: It refers to the systems which will 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.

(viii) **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 violates the local legal requirements should also be rejected. For example, a revised system should comply with all applicable federal and state statutes about financial reporting requirements, as well as the company’s contractual obligations.

2.5.5 **Reporting Results to Management**: After the analyst articulates the problem and its scope, 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 cover letter that summarizes the results and makes the recommendation regarding further procedures. From the analyst's report, management should determine what to do next. Not all projects submitted for evaluation and review are accepted. Requests that fail to pass feasibility test are not pursued further unless they are reworked and resubmitted as new proposals. In some cases, only a part of the project is actually unworkable and the steering committee may decide to combine the workable part of the project with another feasible proposal. In certain other cases, primary investigation produces new information to suggest that improvements in management and supervision, and not the development of information systems are the actual solutions to the reported problems.

2.6 **System Requirement Analysis**

**Objectives**: 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 activities are performed in this phase:

- 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 verify the requirements are complete, consistent, unambiguous, verifiable, modifiable, testable and traceable.
- To gather data or find facts using tools like - interviewing, research/document collection, questionnaires, observation.
- To model activities such as developing models to document Data Flow Diagrams, E-R Diagrams.
- To document activities such as interview, questionnaires, reports etc. and development of a system (data) dictionary to document the modeling activities.

**Document/Deliverable**: A systems requirements report.
2.24 Information Systems Control and Audit

2.6.1 Fact finding Techniques: Every system is built to meet some set of needs, for example, the need of the organization for lower operational costs, better information for managers, smooth operations for users or better levels of service to customers. To assess these needs, the analysts often interact extensively with the people, who will be benefited from the system, in order to determine what are their actual requirements. Various fact-finding techniques, which are used by the system analyst for determining these needs/requirements, are briefly discussed below:

(i) 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, program codes for the applications associated with the current system, etc. Documents are a very good source of information about user needs and the current system.

(ii) Questionnaires: Users and managers are asked to complete questionnaire about the information system 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.

(iii) Interviews: Users and managers may also be interviewed to extract information in depth. The data gathered through interviews often provide systems developer with a complete picture of the problems and opportunities. Interviews also give analyst the opportunity to note user reaction first-hand and to probe for further information.

(iv) Observation: 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.

2.6.2 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. Survey of existing methods, procedures, data flow, outputs, files, input and internal controls should be intensive in order to fully understand the present system and its related problems.

The following areas should be studied in depth:

(i) Review historical aspects: A brief history of the organization is a logical starting point for an analysis of the present system. The historical facts should identify the major turning points and milestones that have influenced its growth. A review of annual reports and organization chart can identify the growth of management levels as well as the development of various functional areas and departments. The system analyst should investigate what system changes have occurred in the past including operations that have been successful or unsuccessful with computer equipments and techniques.

(ii) Analyze 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 should be aware of the various sources from where the data are initially captured, keeping in view the fact that outputs for one area may serve as an input for another area. 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, he will be able to determine how these inputs fit into the framework of the present system.

(iii) **Review data files maintained**: 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. Information on common data files and their size will be an important factor, which will influence the new information system. This information may be contained in the systems and procedures manuals. 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.

(iv) **Review 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. A procedure review is an intensive survey of the methods by which each job is accomplished, the equipment utilized and the actual location of the operations. 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. He must review the types of data communication equipments including data interface, data links, modems, dial-up and leased lines and multiplexers. The system analyst must understand how the data-communications network is used in the present system so as to identify the need to revamp the network when the new system is installed.

(v) **Analyze 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 it is kept on file, etc. must be investigated. Often many reports are a carry-over from earlier days and have little relevance to current operations. Attempt should be made to eliminate all such reports in the new system.

(vi) **Review internal controls**: A detailed investigation of the present information system is not complete until internal control is reviewed. Locating the control points helps the analyst to visualize the essential parts and framework of a system. 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.
Model the existing physical system and logical 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 process must be properly documented. The flow charting and diagramming of present information not only organizes the facts, but also helps disclose gaps and duplication in the data gathered. It allows a thorough comprehension of the numerous details and related problems in the present operation.

Undertake overall analysis of present 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 benefits and costs and each of these must be investigated thoroughly.

2.6.3 Systems Analysis of Proposed Systems: After each functional area of the present information system has been carefully analyzed, the proposed system specifications must be clearly defined which are determined from the desired objectives set forth at the first stage of the study. Likewise consideration should be given to the strengths and shortcomings of the present system. The required systems specifications which should be in conformity with the project's objectives are as follows:

- Outputs produced with great emphasis on timely managerial reports that utilize the 'management by exception' principle.
- Database maintained with great accent on online processing capabilities.
- Input data prepared directly from original source documents for processing by the computer system.
- Methods and procedures that show the relationship of inputs and outputs to the database, utilizing data communications where deemed appropriate.
- Work volumes and timings carefully considered for present and future periods including peak periods.

The starting point for compiling these specifications is output. After outputs have been determined, it is possible to infer what inputs, database, methods, procedures and data communications must be employed. The output-to-input process is recommended since outputs are related directly to the objectives of the organization. The future workload of the system must be defined for inputs, database and outputs in terms of average and peak loads, cycles and trends.

2.6.4 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 to:

- conceptualize, clarify, document and communicate the activities and resources involved in the organization and its information systems;
- analyze present business operations, management decision making and information processing activities of the organization;
- Propose and design new or improved information systems to solve business problems or pursue business opportunities that have been identified.
Many systems development tools take the form of diagrams and other graphic representations. The major tools used for system development can be grouped into four categories based on the systems features. These are as follows:

(I) 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. A data flow diagram uses a 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 system component matrix provides a matrix framework to document the resources used, the activities performed and the information produced by an information system.

(II) 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. Dialogue flow diagrams analyze the flow of dialogue between computers and people. They document the flows among different display screens generated by alternative end user responses to menus and prompts.

(III) 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.
- Grid charts help in identifying the use of each type of data element in input/output or storage media of a system.

(IV) Detailed system process: These tools are used to help the programmer 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. Structure charts document the purpose, structure and hierarchical relationships of the modules in a program.

We will now describe some of these tools in detail.

(a) Structured English: Structured English, also known as Program Design Language (PDL) or Pseudo Code, 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. Program logic that helps to attain precision and natural language that helps in getting the convenience of spoken languages.

(b) Flowcharts: Flowcharting is a graphic technique that can be used by analysts to represent the inputs, outputs and processes of a business in a pictorial form. It is a common type of chart, that represents an algorithm or process showing the steps as boxes of various kinds, and their order by connecting these with arrows. 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 shown in Table 2.6.1. These four symbols are combined to show how data are processed.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Name</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Data Sources and destinations</td>
<td>The people and organizations that send data to and receive data from the system are represented by square boxes called Data destinations or Data Sinks.</td>
</tr>
<tr>
<td></td>
<td>Data flows</td>
<td>The flow of data into or out of a process is represented by curved or straight lines with arrows.</td>
</tr>
<tr>
<td></td>
<td>Transformation process</td>
<td>The processes that transform data from inputs to outputs are represented by circles, often referred to as bubbles.</td>
</tr>
<tr>
<td></td>
<td>Data stores</td>
<td>The storage of data is represented by two horizontal lines.</td>
</tr>
</tbody>
</table>

Table 2.6.1 : Data Flow Diagram Symbols

(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. Decision tables are necessitated by the fact that branches of the flowchart multiply at each diamond (comparison symbol) and may easily run into scores and even hundreds. If, therefore, the programmer attempts to draw a flowchart directly, he is liable to miss some of the branches. The four parts of the decision table are as follows:

(i) **Condition Stub** - which comprehensively lists the comparisons or conditions;

(ii) **Action Stub** - which comprehensively lists the actions to be taken along the various program branches;

(iii) **Condition entries** - which list in its various columns the possible permutations of answer to the questions in the conditions stub; and

(iv) **Action entries** - which 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** : The data flow diagram and system flow charts that users review are
commonly generated by systems developers using the on-screen drawing modules found in **CASE** (Computer-Aided-Software Engineering) software packages. **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.

(g) **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.

Table 2.6.5 illustrates the use of a system component matrix to document the basic components of a sales processing and analysis system in an organization.

<table>
<thead>
<tr>
<th>Information system Activities</th>
<th>Hardware Resources</th>
<th>Software Resources</th>
<th>People Resources</th>
<th>Data Resources</th>
<th>Information Products</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>POS Terminals</td>
<td>Bar tags, Mag Stripe Cards</td>
<td>Data Entry program</td>
<td>Data Entry procedures</td>
<td>Sales clerks, customers</td>
</tr>
<tr>
<td>Processing</td>
<td>Mainframe Computer</td>
<td>Sales processing program, sales analysis program</td>
<td>Sales transaction procedures</td>
<td>Computer operators</td>
<td>Sales clerks managers</td>
</tr>
<tr>
<td>Output</td>
<td>POS Terminals, Management Workstations</td>
<td>Paper reports and receipts</td>
<td>Report generator program, Graphic program</td>
<td>Output use and distribution procedures</td>
<td>Sales clerks managers, customers</td>
</tr>
<tr>
<td>Storage</td>
<td>Magnetic Disk Drives, Magnetic disk</td>
<td>Database management</td>
<td>Computer operators</td>
<td></td>
<td>Customer, inventory</td>
</tr>
</tbody>
</table>

© The Institute of Chartered Accountants of India
2.30 Information Systems Control and Audit

<table>
<thead>
<tr>
<th>Control</th>
<th>POS terminals, Management workstations, Paper documents and control reports</th>
<th>Performance monitor program, security monitor program</th>
<th>Control procedures</th>
<th>Computer operators control clerks</th>
<th>Sales clerks managers customers</th>
<th>Customer, inventory and sales database</th>
<th>Data entry display, sales receipts, Error display and signals</th>
</tr>
</thead>
</table>

**Table 2.6.2**: An example of a system component matrix for a sales processing and analysis system. Note how it emphasizes the basic activities needed, resources used and products produced by this information system.

(h) **Data Dictionary**: A data dictionary is a computer file that 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 computer record in the data dictionary. Similarly, when new computer programs are created those access data items in existing files, the data dictionary is updated to indicate the data items these new programs access. Finally, when data fields are deleted from the structure of file records, their corresponding records in the data dictionary are dropped.

Fig. 2.6.1 shows a sample record from a data dictionary which is basically a file about data. Each file record contains information about one data field used in other files.

<table>
<thead>
<tr>
<th>Name of data field</th>
<th>File in which stored</th>
<th>Source document</th>
<th>Size in bytes</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inventory quantity on hand</td>
<td>Inventory master file</td>
<td>Form number ABC 123</td>
<td>4</td>
<td>Numeric</td>
</tr>
</tbody>
</table>

**Fig. 2.6.1**: Example of Data Dictionary

Accountants and auditors can also make good use of a data dictionary. For example, a data dictionary can help establish an audit trail because it can identify the input sources of data items, the computer programs that modify particular data items, and the managerial reports on which the data items are output. When an accountant is participating in the design of a new system, a data dictionary can also be used to plan the flow of transaction data through the system.
Layout form and Screen Generator, Menu Generator, Report generator, Code Generator

**Layout form and Screen Generator** : They are for printed report used to format or “paint” the desired layouts and contact without having to enter complex formatting information. Fig. 2.6.7 shows a Layout screen for the design for a customer order report.

**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.

**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.

**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.

<table>
<thead>
<tr>
<th>Customer Order Report</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date MM/DD/YY</td>
</tr>
<tr>
<td>Order Number</td>
</tr>
<tr>
<td>Customer Name</td>
</tr>
<tr>
<td>Catalog Number</td>
</tr>
<tr>
<td>X</td>
</tr>
<tr>
<td>X</td>
</tr>
<tr>
<td>X</td>
</tr>
<tr>
<td>X</td>
</tr>
<tr>
<td>X</td>
</tr>
<tr>
<td>X</td>
</tr>
</tbody>
</table>

**Fig. 2.6.2 :** Layout screen for the design of a display for a customer order report.

2.6.5 System Specification

At the end of the analysis phase, the systems analyst prepares a document called “Systems Requirement Specifications (SRS)”. A SRS contains the following:

- **Introduction** : Goals and Objectives of the software context of the computer-based system; Information description
- **Information Description** : Problem description; Information content, flow and structure; Hardware, software, human interfaces for external system elements and internal software functions.
2.32 Information Systems Control and Audit

- **Functional Description**: Diagrammatic representation of functions; Processing narrative for each function; Interplay among functions; Design constraints.

- **Behavioral Description**: Response to external events and internal controls

- **Validation Criteria**: Classes of tests to be performed to validate functions, performance and constraints.

- **Appendix**: Data flow/Object Diagrams; Tabular Data; Detailed description of algorithms charts, graphs and other such material.

- **SRS Review**: It contains the following:
  - 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. Only after ensuring that the document represents existing processes accurately, should the user sign the document. This is a technical requirement of the contract between users and development team/organization.

2.6.6 Roles Involved in SDLC

(i) **Steering Committee**
Some of the functions of Steering Committee are as follows:

- To provide overall direction and ensures appropriate representation of affected parties.
- To be responsible for all cost and timetables.
- To conduct a regular review of progress of the project in the meetings of steering committee which may involve co-ordination and advisory functions.
- Taking corrective actions like rescheduling, re-staffing, change in the project objectives and need for redesigning.

(ii) **Project Manager**: A project manager is normally responsible for more than one project and liaisons with the client or the affected functions. He is responsible for delivery of the project within the time and budget and periodically reviews the progress of the project with the project leader and his team.

(iii) **Project Leader**: The project leader is dedicated to a project who has to ensure its completion and fulfillment of objectives. He reviews the project position more frequently than a Project Manager and the entire project team reports to him.

(iv) **Systems Analyst / Business Analyst**: The systems analysts’ main responsibility is to conduct interviews with users and understand their requirements. He is a link between the users and the programmers who converts the users requirements in the system requirements and plays a pivotal role in the Requirements analysis and Design phase.

(v) **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.

(vi) **Programmer / Coder / Developer**: 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, he also tested the program for
debugging activity.

(vii) **Database Administrator (DBA)**: 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.

(viii) **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.

(ix) **Tester**: Tester is a junior level quality assurance personnel attached to a project who
tests programs and subprograms as per the plan given by the module / project leaders
and prepare test reports.

(x) **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.

(xi) **IS Auditor**: As a member of the team, IS Auditor ensures that the application
development also focuses on the control perspective. 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.

### 2.7 Systems Design

After the completion of requirements analysis for a system, systems design activity takes
place for the alternative which is selected by management.

**Objective**: Designs an Information System that best satisfies the user / managerial
requirements. It describes the parts of the system and their interaction, sets out how the
system shall be implemented using the chosen hardware, software and network facilities,
specifies the program and the database specifications and the security plan and further
specify the change control mechanism to prevent uncontrolled entry of new requirements.

**Activities**: Key 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.

**Document / Deliverable:** Creates a ‘blueprint’ for the design with the necessary specifications for the hardware, software, people and data resources.

System design involves first logical design and then physical construction of a system. The logical design of an information system is like an engineering blueprint; it shows major features of the system and how they are related to one another. Physical construction, the activity following logical design, produces program software, files and a working system. Design specifications instruct programmers about what the system should do. The programmers, in turn, write the programs that accept input from users, process data, produce the reports, and store data in the files.

Once the detailed design is completed, the design is then distributed to the system developers for coding. The design phase involves following steps:

(i) Architectural Design;
(ii) Design of the Data / Information Flow;
(iii) Design of the Database;
(iv) Design of the User-interface;
(v) Physical Design; and
(vi) Design and acquisition of the hardware/system software platform.

**2.7.1 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; function 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 which can be used to represent hierarchies as shown in Fig. 2.7.1. 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 tail.

![Functional Decomposition Tool](image_url)
2.7.2 Design of Data / Information flow: The design of the data and information flows 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).

2.7.3 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 greater the need the interdependence, the greater the need for a global database to prevent sub-optimization by subunits.

The design of the database involves four major activities as discussed in Table 2.7.1.

<table>
<thead>
<tr>
<th>Design Activity</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conceptual Modeling</td>
<td>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.</td>
</tr>
<tr>
<td>Data Modeling</td>
<td>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.</td>
</tr>
<tr>
<td>Storage Structure Design</td>
<td>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.</td>
</tr>
<tr>
<td>Physical Layout Design</td>
<td>Decisions must be made on how to distribute the storage structure across specific storage media and locations –for example, the cylinders, tracks, and sectors on a disk and the computers in a LAN or WAN.</td>
</tr>
</tbody>
</table>

Table 2.7.1: Major activities in Database Designing

2.7.4 Design of User-Interface: Design of user – interface 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.

Designing System Outputs

One of the most important feature of an information system for users is the output it generates. Designing computer output should proceed in an organized, well thought out manner. The right output must be developed while ensuring that each output element is designed so that users will find the system easy to use effectively.
2.36 Information Systems Control and Audit

**Input Objectives**: Input design consists of developing specifications and procedures for data preparation, developing steps which are necessary to put transactions data into a usable form for processing, and data-entry, i.e., the activity of putting the data into the computer for processing.

**Output Objectives**: The output from an information system should accomplish one or more of the following objectives:

- Convey information about past activities, current status or projections of the future.
- Signal important events, opportunities, problems or warnings.
- Trigger an action.
- Confirmation of an action.

**Important factors in Input / Output design**: There are certain important factors listed in Table 2.7.2, which should be considered by the system analyst while designing user input/output forms.

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Definition</th>
<th>Input Design</th>
<th>Output Design</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Content</strong></td>
<td>Refers to the actual pieces of data to be gathered to produce the required output to be provided to users.</td>
<td>The analyst is required to consider the types of data that are needed to be gathered to generate the desired user outputs. New documents for collecting such information may be designed.</td>
<td>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.</td>
</tr>
<tr>
<td><strong>Timeliness</strong></td>
<td>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.</td>
<td>Data needs to be inputted to computer in time because outputs cannot be produced until certain inputs are available. Hence, a plan must be established regarding when different types of inputs will enter the system.</td>
<td>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.</td>
</tr>
<tr>
<td><strong>Format</strong></td>
<td>Input format refers to the manner in which data are physically arranged.</td>
<td>After the data contents and media requirements are determined, input formats are designed on the basis of few constraints like - the</td>
<td>Format of information reports for the users should be so devised that it assists in decision-making,</td>
</tr>
</tbody>
</table>
| **Output format** | refers to the arrangement referring to data output on a printed report or in a display screen.  
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 Volume / 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 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 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. |
any one time is known as output volume. involves thousands of records that are handled by a centralized data entry department using key-to-tape or key-to-disk systems.

Table 2.7.2 : Factors affecting Input / Output Form Design

2.7.5 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. During physical design, the primary concern of the auditor is effectiveness and efficiency issues. The auditor should seek evidence that designers follow some type of structured approach like – CASE tools to access their relative performance via simulations when they undertake physical design. 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.

Design Principles

- 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.
- The design should be based on the analysis.
- The software functions designed should be directly relevant to business activities.
- 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.
- The design should be modular.

Modularity

A module is a manageable unit containing data and instructions to perform a well-defined task. Interaction among modules is based on well-defined interfaces. Modularity is measured by two parameters: Cohesion and Coupling.

Cohesion refers to the manner in which elements within a module are linked.

Coupling is a measure of the interconnection between modules. It refers to the number and complexity of connections between ‘calling’ and ‘called’ modules.

In a good modular design, cohesion will be high and coupling low.

2.7.6 Design of the Hardware / System Software Platform: In some cases, the new system requires hardware and system software not currently available in an organization. For example – a DSS might require high-quality graphics output not supported by the existing hardware and software. The new hardware/system software platform required to support the application system will then have to be designed. If different hardware and software are not able to communicate with each, subsequent changes will have to be made and resources expanded in
trying to make the hardware and software compatible to each other. Auditors should be concerned about the extent to which modularity and generality are preserved in the design of the hardware/system software platform.

2.8 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 hardware, software and services.

2.8.1 Acquisition Standards: Management should establish acquisition standards that address the same security and reliability issues as development standards. Acquisition standards should focus on -

- Ensuring security, reliability, and functionality already built into a product.
- Ensuring managers complete appropriate vendor, contract, and licensing reviews and acquiring products compatible with existing systems.
- Including invitations-to-tender and request-for-proposals. Invitations-to-tender involve soliciting bids from vendors when acquiring hardware or integrated systems of hardware and software. Request-for-proposals involve soliciting bids when acquiring off-the-shelf or third-party developed software.
- Establishing acquisition standards to ensure functional, security, and operational requirements to be accurately identified and clearly detailed in request-for-proposals.

2.8.2 Acquiring Systems Components from Vendors: At the end of the design phase, the organization gets a reasonable idea of the types of hardware, software and services it needs for the system being developed. 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.

(I) 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.

(II) 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 are 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.
(III) 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 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 right. 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.

(IV) 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.

- The Performance capability of each proposed System in Relation to its Costs;
- The Costs and Benefits of each proposed;
- The Maintainability of each proposed;
- The Compatibility of each proposed system with Existing Systems; and
- Vendor Support.

(V) 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 as follows:

(i) Checklists: It is the most simple and rather a subjective method for validation and evaluation. The various criteria are put in checklist 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; 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.
For example – Table 2.8.1 illustrates a Point Scoring Analysis list.

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

Table 2.8.1 : Point Scoring Analysis List

(iii) **Public Evaluation Reports** : Several consultancy 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, resort 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 problem for vendor’s proposals** : Benchmarking problems for vendors’ proposals are sample programs that represent at least a part of the buyer’s primary computer work load and include software considerations and can be current applications programs or new programs that have been designed to represent planned processing needs. That is, benchmarking problems are oriented towards testing whether a computer offered by the vendor meets the requirements of the job on hand of the buyer.

(v) **Test 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 judgement can be made. It must be borne in mind, however that various capabilities to be tested would have to be assigned relative weight-age.

### 2.9 Development: Programming Techniques and Languages

**Objective**: To convert the specification into a functioning system.

**Activities**: Application programs are written, tested and documented, conduct system testing.

**Document / Deliverable**: A fully functional and documented system.

A good coded program should have the following characteristics:

- **Reliability**: It refers to the consistence which a program provides over a period of time. However poor setting of parameters and hard coding some data, subsequently could result in the failure of a program after some time.

- **Robustness**: It refers to the process of taking into account all possible inputs and outputs of a program in case of least likely situations.

- **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.

- **Efficiency**: It refers to the performance which should not be unduly affected with the increase in input values.

- **Usability**: It refers to a user-friendly interface and easy-to-understand document required for any program.

- **Readability**: It refers to the ease of maintenance of program even in the absence of the program developer.

#### 2.9.1 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. Different programmers may write a program using different sets of instructions but each giving the same results. Therefore, the coding standards are defined which serves as a method of communication between teams, amongst the team members and users, thus working as a good control. Coding standards minimize the system development setbacks due to programmer turnover. Coding standards provide, simplicity, efficient utilization of storage and least processing time.

#### 2.9.2 Programming Language

Application programs are coded on the form of statements or instructions and the same is converted by the compiler to binary machine for the computer to understand and execute. The programming languages commonly used are as follows:

- High – level general purpose programming language such as COBOL and C language.

- Object oriented languages such as C++, JAVA etc.
• Scripting language like JAVAScript, VBScript.
• Decision Support or Expert System languages like PROLOG.

Choice of Programming Language

The following are among the most important criteria on the basis of which the language to be used should be decided 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.

2.9.3 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:

• Inputting the source program to the compiler,
• Letting the compiler find errors in the program,
• Correcting lines of code that are erroneous, and
• Resubmitting the corrected source program as input to the compiler.

2.9.4 Test the program: 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 all possible exceptions. The test plan should require the execution of all standard processing logic. The program test plan should be discussed with the project manager and/or system users. A log of test results and all conditions successfully tested should be kept. The log will prove invaluable in answering the inevitable question. ‘Did you ever test for this condition?’

2.9.5 Program Documentation: The writing of narrative procedures and instructions for people who will use software is done throughout the program life cycle. Managers and users should carefully review documentation in order to ensure that the software and system behave as the documentation indicates. If they do not, documentation should be revised. User documentation should also be reviewed for understandability i.e. the documentation should be prepared in such a way that the user can clearly understand the instructions.

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

2.10 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 and with a minimum amount of effort. 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 of Testing are as follows:

2.10.1 Unit Testing: In computer programming, 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. or may belong to a base/super class, abstract class or derived/child
class.

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 part of the program
and show that the individual parts 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:

- **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.

- **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.

- **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, often to
  a breaking point, 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. For example, during a sort operation, the available memory can be
  reduced to find out whether the program is able to handle the situation.

- **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.

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

Types of Unit Testing

(I) **Static Analysis Testing**

Some important **Static Analysis Tests** are as follows:

- **Desk Check**: This is done by the programmer himself. He checks for logical syntax
  errors, and deviation from coding standards.

- **Structured walk-through**: The application developer leads other programmers through
  the text of the program and explanation.
- **Code inspection**: The program is reviewed by a formal committee. Review is done with formal checklists. The procedure is more formal than a walk-through.

(II) **Dynamic Analysis Testing**

- **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 valid and invalid inputs and determines the correct output. There is no knowledge of the test object’s internal structure.

![Black Box Testing Diagram]

This method of test design is applicable to all levels of software testing: unit, integration, functional testing, system and acceptance. The higher the level, hence the bigger and more complex the box, the more one is forced to use black box testing to simplify. While this method can uncover unimplemented parts of the specification, one cannot be sure that all existent paths are tested. If a module performs a function which is not supposed to, the black box test does not identify it.

- **White Box Testing**: White box testing 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. Since the tests are based on the actual implementation, if the implementation changes, the tests probably will need to change, too. 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.

- **Gray Box Testing**: Gray box testing 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.

2.10.2 **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 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 manner:

- **Bottom-up Integration**: Bottom-up integration 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. The disadvantage, however is that testing of major decision / control points is deferred to a later period.

- **Top-down Integration**: Top-down integration 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 in to 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. Since decision- making processes are likely to occur in the higher levels of program hierarchy, the top-down strategy emphasizes on major control decision points encountered in the earlier stages of a process and detects any error in these processes. The difficulty arises in the top-down method, because the high-level modules are tested, not with real outputs from subordinate modules, but from stubs.

- **Regression Testing**: Each time a new module is added as part of integration testing, the software 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 errors. The data used for the regression tests should be the same as the data used in the original test.

2.10.3 **System Testing**: System testing 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. These test procedures are often performed in a non-production test environment. The types of testing that might be carried out are as follows:

- **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 properly performed.

- **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, authentication, authorization, availability and non-repudiation. This testing technique also ensures the existence and proper execution of access controls in the new system.

- **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.
• **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.

2.10.4 **Final Acceptance Testing**: Final Acceptance Testing is conducted when the system is just ready for implementation. 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:

• **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 methodology.

• **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:
  - **Alpha Testing**: This is the first stage, often performed by the users within the organization.
  - **Beta Testing**: This is the second stage, generally performed by the external users. This is the last stage of testing, and normally involves sending the product outside the development environment for real world exposure.

2.11 **Systems Implementation**

**Objective**: To implement the new system i.e. put it into production.

**Activities**: The activities involved in System Implementation are as follows:

- Conversion of data to the new system files.
- Training of end users.
- Completion of user documentation.
- System changeover.
- Evaluation of the system at regular intervals.

**Document / Deliverable**: A full functional / documented system in its operational environment.

The process of ensuring that the information system is operational and then allowing users to take over its operation for use and evaluation is called **Systems Implementation**. Implementation includes all those activities that take place to convert from the old system to the new. The new system may be totally new, replacing an existing manual or automatic system or it may be a major modification in an existing system.

2.11.1 **Activities during Implementation Stage**: The activities involved in system implementation stage are as follows:

(I) **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:

- **Site Preparation**: An appropriate location must be found to provide an operating environment for the equipment that will meet the vendor's temperature, humidity, and dust control specifications.

- **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 production environment may be desirable to prove end to end connectivity.

- **Equipment check out**: The equipment must be turned on for testing under normal operating conditions. Not only the routine 'diagnostic test' should be run by the vendor, but also the implementation team should devise and run extensive tests of its own to ensure that equipment is in proper working condition.

(II) **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. Often this is imparted through classes, which are organized by vendor, and through hands-on learning techniques.

(III) **System Implementation Conversion Strategies**: Conversion or changeover is the process of changing from the old system (manual system) to the new system. It requires careful planning to establish the basic approach to be used in the actual changeover. The Four types of implementation strategies are as follows:

(i) **Direct Implementation / Abrupt change-over**: This is achieved through an abrupt takeover – an all or nothing approach. With this strategy, the changeover is done in one operation, completely replacing the old system in one go. Fig 2.11.1 depicts Direct Implementation 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.

![Fig. 2.11.1 : Direct Implementation](image-url)
(ii) **Phased implementation**: With this strategy, implementation can be staged with conversion to the new system taking place by degrees. For example - some new files may be converted and used by employees whilst other files continue to be used on the old system i.e. the new is brought in stages (phases). If each phase is successful then the next phase is started, eventually leading to the final phase when the new system fully replaces the old one as shown in Fig. 2.11.2.

![Phased Implementation Diagram](image)

**Fig. 2.11.2 : Phase Implementation**

(iii) **Pilot implementation**: 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. Fig. 2.11.3 depicts Pilot Implementation.

![Pilot Implementation Diagram](image)

**Fig. 2.11.3 : Pilot Implementation**

(iv) **Parallel running implementation**: 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. Fig. 2.11.4 shows parallel implementation.

![Parallel Implementation Diagram](image)

**Fig. 2.11.4 : Parallel Implementation**

2.11.2 **Activities involved in conversion**: Conversion includes all those activities which must be completed to successfully convert from the previous system to the new information.
2.50 Information Systems Control and Audit

Fundamentally these activities can be classified as follows:

(i) **Procedure conversion**: Operating procedures should be completely documented for the new system that applies to both computer-operations and functional area operations. 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.

(ii) **File conversion**: Because large files of information must be converted from one medium to another, this phase should be started long before programming and testing are completed. The cost and related problems of file conversion are significant whether they involve on-line files (common database) or off-line files.

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.

(iii) **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 and balancing the total results of both systems.

(iv) **Scheduling personnel and equipment**: Scheduling data processing operations of a new information system for the first time is a difficult task for the system manager. As users become more familiar with the new system, however, the job becomes more routine.

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 month should provide sufficient computer time to handle all required processing.

### 2.12 Post Implementation Review and Systems Maintenance

**Objective**: To assess and review the complete working solution.

**Activities**: Some of the Systems maintenance activities are as follows:

- Adding new data elements;
- Modifying reports;
- Adding new reports; and
• Changing calculations.

**Document / Deliverable** : A document stating scope of further improvements, if any like-
- Could further training or coaching improve the degree of benefit being generated?
- Are there further functional improvements or changes that would deliver greater benefit?
- Are specific improvements required in procedures, documentation, support, etc?
- What learning points are there for future projects?

### 2.12.1 Post Implementation Review

A Post Implementation Review answers the question “Did we achieve what we set out to do in business terms?” Some of the purposes served a Post Implementation Review ascertains the degree of success from the project, in particular, the extent to which it met its objectives, delivered planned levels of benefit, and addressed the specific requirements as originally defined.

- 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.

- **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. In fact, many information systems are developed without clearly defined schedules or budgets. Due to the uncertainty and mystique associated with system development, they are not subjected to traditional management control procedures.

- **Operation evaluation** : The evaluation of the information system's operation pertains to whether the hardware, software and personnel are capable to perform their duties. Operation evaluation answers such questions : Operation evaluation is relatively straightforward if evaluation criteria are established in advance. For example, if the systems analyst lays down the criterion that a system which is capable of supporting one hundred terminals should give response time of less than two seconds, evaluation of this aspect of system operation can be done easily after the system becomes operational.

- **Information evaluation** : An information system should also be evaluated in terms of information it provides. This aspect of system evaluation is difficult and it cannot be conducted in a quantitative manner, as is the case with development and operation evaluations. The objective of an information system is to provide information to support the organizational decision system. Therefore, the extent to which information provided by the system is supportive to decision making is the area of concern in evaluating the system.
2.12.2 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.

Most information systems require at least some modification after development. The need for modification arises from a failure to anticipate all requirements during system design and/or from changing organizational requirements. Maintenance can be categorized in the following two ways:

- **Scheduled maintenance**: Scheduled maintenance is anticipated and can be planned for. For example, the implementation of a new inventory coding scheme can be planned in advance.

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

- **Corrective maintenance**: Corrective maintenance deals with fixing bugs in the code or defects found. 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. Examples of corrective maintenance include correcting a failure to test for all possible conditions or a failure to process the last record in a file.

- **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, for example, business rule, government policies, work patterns, software and hardware operating platforms. The need for adaptive maintenance can only be recognized by monitoring the environment.

- **Perfective maintenance**: Perfective maintenance mainly deals with accommodating to 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.

- **Preventive maintenance**: Preventive maintenance concerns 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. As a large program is continuously changed, its complexity, which reflects deteriorating structure, increases unless work is done to maintain or reduce it. This work is known as preventive change.

2.13 Operation Manuals

**Operation Manuals**: A user’s guide, also commonly known as an Operation Manual, is a technical communication document intended to give assistance to people using a particular system. It is usually written by a technical writer, 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. The section of an operation manual after include the following:

- A cover page, a title page and copyright page;
• A preface, containing details of related documents and information on how to navigate the user guide;
• A contents page;
• A guide on how to use at least the main functions of the system;
• A troubleshooting section detailing possible errors or problems that may occur, along with how to fix them;
• A FAQ (Frequently Asked Questions);
• Where to find further help, and contact details;
• A glossary and, for larger documents, an index;

Sample format of any operations manual could be as shown in Fig. 2.13.1.

<table>
<thead>
<tr>
<th>1.0 GENERAL INFORMATION</th>
<th>3.0 RUN DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 SYSTEM OVERVIEW</td>
<td>3.1 RUN INVENTORY</td>
</tr>
<tr>
<td>1.2 Project References</td>
<td>3.2 RUN DESCRIPTION</td>
</tr>
<tr>
<td>1.3 Authorized Use Permission</td>
<td>*3.2.x [Run Identifier]</td>
</tr>
<tr>
<td>1.4 Points of Contact</td>
<td>3.2.x.1 Run Interrupt Checkpoints</td>
</tr>
<tr>
<td>1.4.1 Information</td>
<td>3.2.x.2 Set-Up and Diagnostic Procedures</td>
</tr>
<tr>
<td>1.4.2 Coordination</td>
<td>3.2.x.3 Error Messages</td>
</tr>
<tr>
<td>1.4.3 Help Desk</td>
<td>3.2.x.4 Restart/Recovery Procedures</td>
</tr>
<tr>
<td>1.5 Organization of the Manual</td>
<td></td>
</tr>
<tr>
<td>1.6 Acronyms and Abbreviations</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2.0 SYSTEM OPERATIONS OVERVIEW</th>
<th>2.0 SYSTEM OPERATIONS OVERVIEW</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 System Operations</td>
<td>2.1 System Operations</td>
</tr>
<tr>
<td>2.2 Software Inventory</td>
<td>2.2 Software Inventory</td>
</tr>
<tr>
<td>2.3 Information Inventory</td>
<td>2.3 Information Inventory</td>
</tr>
<tr>
<td>2.3.1 Resource Inventory</td>
<td>2.3.1 Resource Inventory</td>
</tr>
<tr>
<td>2.3.2 Report Inventory</td>
<td>2.3.2 Report Inventory</td>
</tr>
<tr>
<td>2.4 Operational Inventory</td>
<td>2.4 Operational Inventory</td>
</tr>
<tr>
<td>2.5 Processing Overview</td>
<td>2.5 Processing Overview</td>
</tr>
<tr>
<td>2.5.1 System Restrictions</td>
<td>2.5.1 System Restrictions</td>
</tr>
<tr>
<td>2.5.2 Waivers of Operational Standards</td>
<td>2.5.2 Waivers of Operational Standards</td>
</tr>
<tr>
<td>2.5.3 Interfaces with Other Systems</td>
<td>2.5.3 Interfaces with Other Systems</td>
</tr>
<tr>
<td>2.6 Communications Overview</td>
<td>2.6 Communications Overview</td>
</tr>
<tr>
<td>2.7 Security</td>
<td>2.7 Security</td>
</tr>
</tbody>
</table>

* Each run should be under a separate header. Generate new sections and subsections as necessary for each run from 3.2.1 through 3.2.x.

Fig. 2.13.1 : Sample format of Operations Manual

2.14 Auditors’ 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

to assess the controls being provided for the management of the system's operation.

For the first objective to achieve, an auditor will have 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, (such as a formal systems development process) and determining the extent to which compliance is being achieved. For addressing second objective, the auditor is 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 Technology system should meet. A list of such objectives should be provided to the auditee. The same is true for the third objective, viz. system's operational controls. The auditor should provide the 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.

An Auditor may adopt a rating system such as on scale of 1 to 10 in order to give rating to the various phases of SDLC. E.g. in rating a Feasibility Study, auditor can review Feasibility Study Report and different work products of this study phase. An interview with personnel who have conducted this feasibility study can be conducted. Depending on the content and quality of the Feasibility Study report and interviews, an auditor can arrive at a rating between 1 to 10 (10 being best). After deriving such a rating for all the phases, the auditor can form his/her overall opinion about the SDLC phases.

In order to audit technical work products (such as database design or physical design), auditor may opt to include a technical expert to seek his/her opinion on the technical aspects of SDLC. However, auditor will have to give control objectives, directives and in general validate the opinion expressed by technical experts. Some of the control considerations for an auditor are:

- Documented policy and procedures;
- Established Project team with all infrastructure and facilities ;
- Developers/ IT managers are trained on the procedures ;
- Appropriate approvals are being taken at identified mile-stones;
- Development is carried over as per standards, functional specifications;
- Separate test environment for development/ test/ production / test plans;
- Design norms and naming conventions are as per standards and are adhered to;
- Business owners testing and approval before system going live;
- Version control on programs;
- Source Code is properly secured;
- Adequate audit trails are provided in system; and
- 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 were 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. E.g. it could be due to incomplete user requirements. Such reasons can help auditors to evaluate the current situation and offer guidelines for future projects.

**Master Checklist**

The process objectives are:

- To ensure an appropriate acquisition and / or development of information systems including software, and
- To maintain the information systems in an appropriate manner.

The following checklist may be used by the IS Auditors for this purpose:

<table>
<thead>
<tr>
<th>S. No.</th>
<th>Checkpoints</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Whether information system acquisition and / or development policy and procedure documented?</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>Whether system acquisition and / or development policy and procedure approved by the management?</td>
<td></td>
</tr>
</tbody>
</table>
| 3.     | Whether the policy and procedure cover the following:  
  • Problems faced in the existing system and need for replacement  
  • Functionality of new IS  
  • Security needs  
  • Regulatory compliance  
  • Acceptance Criteria  
  • Proposed roles and responsibilities  
  • Transition/ Migration to new IS  
  • Interfaces with legacy systems  
  • Post implementation review  
  • Maintenance arrangements. |        |
<p>| 4.     | Whether policy and procedure documents are communicated / available to the respective users? |        |
| 5.     | Whether policy and procedure documents are reviewed and updated at regular intervals? |        |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
</table>
| 6. | Whether the organization has evaluated requirement and functionalities of proposed IS?  
(Verify the requirement analysis conducted at three levels viz. process level, application level and organization level. Verify the site visit reports and other customer references obtained with respect to functionalities of proposed IS). |
| 7. | Whether the organization carried out feasibility study in respect of the following  
- Financial feasibility  
- Operational feasibility  
- Technical feasibility |
| 8. | Whether the selection of vendor and acquisition terms considers the following:  
- Evaluation of alternative vendors  
- Specification on service levels and deliverables  
- Penalty for delays  
- Escrow mechanism for Source codes  
- Customization  
- Upgrades  
- Regulatory Compliance  
- Support and maintenance. |
| 9. | Whether the organization has identified and assigned roles in development activities to appropriate stakeholders?  
(Verify the assigned roles should be on “need to know” and “need to basis”, and duties of developers and operators are segregated). |
| 10. | Whether the organization has a separate development, test and production environments? |
| 11. | Whether the IS developed plan is prepared and approved by the management?  
(Verify that IS development plan to include:  
- Input data elements,  
- Validations controls viz. Field/ Transactions/ File with appropriate error reporting  
- Process workflow  
- data classifications with security are in place, viz. Read |
<table>
<thead>
<tr>
<th></th>
<th>only for users, Read/ Write for authorized persons</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Output).</strong></td>
</tr>
<tr>
<td>12.</td>
<td>Whether the testing of IS includes:</td>
</tr>
<tr>
<td></td>
<td>• Confirms the compliance to functional requirements</td>
</tr>
<tr>
<td></td>
<td>• Confirms the compatibility with IS infrastructure</td>
</tr>
<tr>
<td></td>
<td>• Identifies bugs and errors and addresses them by analyzing root causes</td>
</tr>
<tr>
<td></td>
<td>Escalating functionality issues at appropriate levels.</td>
</tr>
<tr>
<td>13.</td>
<td>Whether the adequate documentation for:</td>
</tr>
<tr>
<td></td>
<td>• Preserving test results for future reference</td>
</tr>
<tr>
<td></td>
<td>• Preparation of manuals like systems manual, installation manual, user manual</td>
</tr>
<tr>
<td></td>
<td>• Obtaining user sign off / acceptance</td>
</tr>
<tr>
<td>14.</td>
<td>Whether the implementation covers the following?</td>
</tr>
<tr>
<td></td>
<td>• User Departments' involvement and their role</td>
</tr>
<tr>
<td></td>
<td>• User Training</td>
</tr>
<tr>
<td></td>
<td>• Acceptance Testing</td>
</tr>
<tr>
<td></td>
<td>• Role of Vendor and period of Support</td>
</tr>
<tr>
<td></td>
<td>• Required IS Infrastructure plan</td>
</tr>
<tr>
<td></td>
<td>• Risk involved and actions required to mitigate the risks</td>
</tr>
<tr>
<td></td>
<td>• Migration plan</td>
</tr>
<tr>
<td>15.</td>
<td>If the development activities are outsourced, are the outsourcing activities evaluated based on the following practices:</td>
</tr>
<tr>
<td></td>
<td>• What is the objective behind Outsourcing?</td>
</tr>
<tr>
<td></td>
<td>• What are the in-house capabilities in performing the job?</td>
</tr>
<tr>
<td></td>
<td>• What is the economic viability?</td>
</tr>
<tr>
<td></td>
<td>• What are the in-house infrastructure deficiencies and the time factor involved?</td>
</tr>
<tr>
<td></td>
<td>• What are the Risks and security concerns?</td>
</tr>
<tr>
<td></td>
<td>• What are the outsourcing arrangement and fall back method?</td>
</tr>
<tr>
<td></td>
<td>• What are arrangements for obtaining the source code for the software?</td>
</tr>
<tr>
<td></td>
<td>• Reviewing the capability and quality of software development activities by visit to vendor's premises?</td>
</tr>
<tr>
<td></td>
<td>• Review of progress of IS development at periodic intervals.</td>
</tr>
</tbody>
</table>
2.58 Information Systems Control and Audit

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>16.</td>
<td>Whether the organization carried out a post implementation review of new IS?</td>
</tr>
<tr>
<td>17.</td>
<td>Whether a process exists for measuring vendors' performance against the agreed service levels?</td>
</tr>
<tr>
<td>18.</td>
<td>Whether the post implementation review results are documented?</td>
</tr>
</tbody>
</table>

References:


© The Institute of Chartered Accountants of India