CHAP 2 - SYSTEM DEVELOPMENT LIFE CYCLE METHODOLOGY

- Introduction
- Systems development process
- Achieving system development objectives
- System development team
- Accountants’ involvement in development work
- Systems development methodology
- Approaches to system development
- System development life cycle (SDLC)
- Operation manuals
- Organizational structure of IT department
- Project management structure

INTRODUCTION - What is System development?

Process of examining a business situation to improve it through better procedures & methods. Its 2 major components:

1. **Systems analysis** - process of gathering and interpreting facts, diagnosing problems, and using the information to recommend improvements to the system.
   - Study current operations & problems, information flow
   - Assess future needs, changes required
   - Analyse suitable alternatives
   - Management finalizes

2. **Systems design** - process of planning a new business system or one to replace/complement existing one.

“Before planning, understand old system and determine how computers can be used (if at all) to make its operation more effective.” Example- Before computerizing, understand & analyse stockroom operations (files, reports, stock movement) of a clothing store for better inventory control & up-to-date information.

1. Plan developed to implement recommendations of systems analysis
2. Draw detailed blueprint (system design) specifying features, operating procedures, specifications, equipment and personnel requirements of the new system
Achieving “System Development Objectives” - hurdles

1. **Lack of senior Management support** for and involvement in information systems development. Management to arrange for adequate resources and exercise budgetary control over use of those resources. Efforts as per project importance.

2. **Lack of user Participation**: Users’ involvement necessary to reduce resistance to changes and ensure adequate development.

3. **Shifting User needs**: User requirements for IT change constantly. Changes during an ongoing development process pose a challenge.

4. **Resistance to change**: Users have a natural tendency to resist change. E.g., expectations of personnel cutbacks.

5. **Inadequate testing and user training**: New systems must be tested before installation. Users must be adequately trained.

6. **Overworked or under-trained development staff**: Backlog of work created. Developers may be overworked or lack expertise/edu. Staff updating, training plans and budget may not exist.

7. **Development of strategic systems**: Strategic decision making is unstructured. Requirements, specifications, and objectives - difficult to define.

8. **New technologies**: Use of advanced IT needed for competitive advantage. However personnel may be unfamiliar.

9. **Lack of standard project management and systems development methodologies**: Lack of formalized methodologies -> difficult to complete projects on time or within budget.

(- Mgmt vs. Staff issues + IT Systems Stds. issues -)

**System Development Team** - responsible for systems development

Several people responsible for systems development. In large systems,

- **Top management level steering committee** - group of key IS services users that acts as a review body. Decides project’s worth. Its aim - satisfy information requirements of managers and users.

- **A Project management team** = computer professionals and key users

- **System analysts** determine user requirements, design the system and assist in development and implementation activities

- **Systems designers** take a lead role during the design, development and implementation stages

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

**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/have

- **Combine knowledge of IT, business, accounting, and internal control, as well as behavior and communications**, to ensure that new systems meet user needs & has adequate internal controls.

- **Specialized skills** - such as accounting and auditing -> proposed system’s Cost-Benefit
SYSTEMS DEVELOPMENT “METHODOLOGY”

Formalized, standardized, documented set of activities used to manage a systems development project. Framework used to structure, plan and develop systems. The methodology is characterised by the following:

1. **Training** - Training plan must be developed for users.
2. **Processes** - Project divided into identifiable processes. Small, manageable steps helps planning and control. Each process has a starting point and an ending point and comprises several activities, deliverable(s), and management control point(s).
3. **Deliverables** - Specific reports / other documentation produced periodically during systems development to make developers accountable.
4. **Participation** - Users, managers, and auditors are required to participate in the project. Provides system of approvals/sign-offs at pre-established management control points.
5. **Testing** - Systems must be tested prior to implementation so that it meets users’ needs.
6. **Controls** - Formal program controls to prevent unauthorized changes.
7. **Review** - A post-implementation review must be performed to assess the effectiveness and efficiency of the new system and of the development process.

**Articleship in an Audit Firm**: Training provided by seniors on business processes of client cos. -> Identify deliverables for audit assignment -> Work is divided and all participate -> Testing Controls -> Review work done.

"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:

1. **Waterfall** : Linear framework type
2. **Prototyping** : Iterative framework type
3. **Incremental** : Combination of linear and iterative framework type
4. **Spiral** : Combination linear and iterative framework type
5. **Rapid Application Development (RAD)** : Iterative Framework Type
6. **Agile Methodologies**

(© S P I R A I Waterfall -)
The Traditional / Waterfall Approach / Sequential Approach

“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 as it eliminates testing to identity problems early in the process.

In the traditional approach of system development,
- Activities are performed in sequence.
- Activity starts only when prior step fully completed
- Examples of tasks performed during each phase of the traditional approach. (Refer diagram - Overview of WaterFall Model)

FRAMEWORK TYPE: LINEAR

Basic Principles
- Project is divided into sequential phases, with some overlap & splash back acceptable between phases
- Emphasis on Planning, time schedules, deadlines, budgets & implementation at one time
- Tight control is maintained via extensive written documentation, formal reviews, approval/signoff

Strengths
1) Ideal for supporting a) less experienced project teams/managers b) teams whose composition fluctuates
2) Orderly sequence/steps/reviews help ensure the quality, reliability, adequacy and maintainability of SW
3) Progress is measurable
4) Conserves resources

[Diagram of Waterfall Model]

(Refer Phases in SDLC -)
Weaknesses-
1) **Inflexible, slow, costly, and cumbersome** due to sequential structure and tight controls
2) **X iteration**

3) Requires *early* identification and specification of requirements
4) Produces excessive documentation and keeping it updated as the project progresses is time-consuming. Written specifications are often difficult for users to read and thoroughly appreciate.

5) System performance **can’t be tested** until system fully coded, & under-capacity difficult to correct
6) Difficult to **respond to Δs**. Δs that occur later in the life cycle are costly and are thus discouraged.
7) Requirement inconsistencies, missing system components and unexpected development needs are often discovered during design and coding while problems during system testing.

8) Increases **gap between users and developers** with clear vision of responsibility

---

**The Prototyping Model**

To avoid delays (as found in Traditional approach), use prototyping techniques to **develop smaller systems** such as DSS, MIS and Expert systems.

**Goal**- Develop a small/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.

...Users work - make suggestions - incorporated into another prototype - used & evaluated...
- When a prototype is developed that satisfies all user needs -> either refined/developed/implemented OR scrapped. If scrapped, the knowledge gained used to develop the real system

**FRAMEWORK TYPE: ITERATIVE.**

**Basic Principles**: Series of 4 steps. I&M phases after model is tested and found to meet user needs.
Step 1 - Identify Information System Requirements

- In traditional approach, system requirements have to be identified before development process starts.
- Here, ONLY fundamental system requirements needed to build initial prototype. Less formal and time-consuming process

Step 2: Develop the Initial Prototype

- Initial base model is created first - No consideration to internal controls, but to system characteristics such as "simplicity", "flexibility" and "ease of use".
- Enables users to interact with tentative versions of data entry display screens, menus, input prompts, and source documents.
- Also helps users to respond, make enquiries, study response time and issue commands.

Step 3: Test and Revise

- Model demonstrated to users for experimentation. Users submit feedback (likes, dislikes, recommendations) for modification.
- Design team modifies prototype and then resubmits the revised model to system users for re-evaluation. Such iterative process continues until the users are satisfied

Step 4: Obtain User Signoff of the Approved Prototype

- Users formally approve final version of the prototype
- Establishes users' commitment to current design and a contractual obligation about system capabilities

“Prototyping is not commonly used for developing traditional applications such as AR, AP, payroll, or inventory management, where the IPO are well known and clearly defined”

Strengths (comparison with Traditional approach)
1. Improves user participation & communication
2. Prototyping requires intensive user involvement. Results in better definition of users' needs
3. Helps resolving unclear objectives, validating requirements, experimenting, comparing design solutions
4. Helps identify confusing or difficult functions and missing functionality
5. Short time period required to develop and start testing
6. Users experiment; errors detected and eliminated early. System more reliable and less costly to develop
7. Encourages innovation and flexible designs
8. Using knowledge gained in an early iteration
9. Provides quick implementation of an incomplete, but functional application

Weaknesses
1. Approval process/controls - Not strict.
2. X sufficient checks and balances
3. Requirements change frequently
4. Incomplete / inadequate problem analysis -> obvious needs addressed -> inefficient practices built in
5. Identification of non-functional elements us difficult to document.
6. Designers may prototype too quickly, -> inflexible design that limits future system potential
7. Successful IF system users are devote time in experimenting and provide change suggestions
8. Interactive process -> extensive experimentation -> Developers tempted to j testing & documentation. Thus, approved system error-prone, and difficult to maintain.
9) Causes *behavioral problems* with system users. User dissatisfaction if ALL user demands for improvements not met + have to go through too many interactions

“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”

**The Incremental Model**

**Framework type:** Combination Linear (W/F) + Iterative

**Basic Principles:** Model is designed, implemented & tested *incrementally* (a little more is added each time) until product is finished (i.e. when it satisfies all of its requirements). Process-

- Product decomposed into components
- Each component designed & built separately (aka builds)
- Each component delivered to client when complete

+/-

- Partial utilization of product and avoids a *long development time*
- ↓ Traumatic effect of introducing new system “all at once”

**Steps**-
1. *Series of mini-w/f performed-* all phases of w/f development model completed for a small part of system
2. *Overall requirements* are defined before proceeding to evolutionary, mini-w/f development of individual increments of the system
3. The *initial* software concept, requirement analysis, and design of architecture and system core are defined using the W/F approach, followed by Iterative Prototyping, which concludes in installation of the *final* prototype (i.e. Working system)

**Incremental Model**-

**Strengths:**
1. *Using knowledge gained* in an early increment
2. *Moderate control* via- written documentation, formal review and approval/signoff, milestones
3. *Project status* throughout life cycle
4. *More flexible* - less costly to Δ scope / requirements.
5. ↓ integration & architectural *risks earlier*
6. Gradual implementation helps monitor *effect of incremental Δs*, isolated issues & make adjustments
7. Allows *delivery* of a series of implementations -> production more quickly with incremental releases
Weaknesses:
1. Lack of overall consideration of business problem & technical requirements for overall system
2. Each phase of an iteration is rigid and do not overlap
3. System architecture problems ↑ because not all requirements are gathered up-front
4. Some modules completed before others, hence, well-defined interfaces required
5. Difficult problems shifted to future to demonstrate early success to management

Spiral Model / Spiral Lifecycle

Framework Type: Combination Linear and Iterative.

Basic Principles: It is a Systems Development Method (SDM) which combines-
- Advantages of top-down and bottom-up concepts
- Features of prototyping model and w/f model

“The spiral model is intended for large, expensive and complicated projects. E.g. “Game development” - size and constantly shifting goals of those large projects”

1. System requirements are defined in detail. Interviewing users.
2. A preliminary design is created for the new system - For cost effective project, all possible alternatives analyzed & strategies decided. Identify and resolve all possible risks. If risks of uncertainty in requirements, prototyping used.
3. 1st prototype of new system is created - usually a scaled-down system, & represents an approximation of characteristics of final product
4. A 2nd prototype is evolved by a four-fold procedure:
   - evaluating SWOT of 1st prototype
   - defining requirements of the 2nd prototype
   - planning & designing 2nd prototype
   - creating & testing 2nd prototype

Strengths-
(i) Enhance risk avoidance
(ii) Helps select best methodology
(iii) for development of a given software iteration based on project risk
(iv) Can incorporate Waterfall, Prototyping, and Incremental methodologies & tell which combination best based upon type of project risk

“E.g., 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.”

Weaknesses-
(i) Challenges to determine exact composition of development methodologies
(ii) Highly customized to each project, & hence, complex, limiting reusability
(iii) Skilled & experienced project manager required
(iv) No established controls for moving from one cycle to another -> More work
(v) No firm deadlines - inherent risk of not meeting budget or schedule
Rapid Application Development (RAD)

Framework Type: Iterative

Basic Principles: Minimal planning for rapid prototyping. "Planning" of software is interleaved with writing the software itself. Lack of extensive pre-planning allows software to be written much faster, and change requirements easily. Objective/features:

- **Fast development** & delivery of a high quality system at a relatively low investment cost.
- **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.
- **Reduce project risk** by breaking a project into smaller segments and ease-of-change

Generally includes Joint Application Development (JAD) - users intensely involved in system design, via-consensus building in structured workshops, or through e-interaction. Active user involvement is vital

- **Iteratively produces production software**, as opposed to a "throwaway" prototype

- **Documentation** to facilitate future development and maintenance
- **Standard techniques** of systems analysis & design
- **Business needs before tech/engg excellence**
- **Use of iterative Prototyping** (at any stage of development), active user involvement, and computerized development tools. GUI builders, CASE tools, DBMS, 4th GL, Code generators & object-oriented techniques etc.

Strengths
1. **Operational version** available much earlier than with W/F, I, S -> Lower cost & Quick initial reviews
2. Generally produces a dramatic savings in time, money and human effort.
3. Constant integration isolates problems and encourages customer feedback
4. ↑ **Commitment** from stakeholders (users/developers) than with W/F, I, S.
5. Focus on system elements from user viewpoint -> Rapidly change system design as demanded by users.
6. Produces a tighter fit between user requirements and system specifications.

Weaknesses
1. Lower system quality (High speed-Low cost)
2. Misalignment of developed system with business due to missing information (no planning/need analysis)
3. May have more requirements than needed (Gold-plating) (Less vs More)
4. May have “feature creep" = rapidly adding features
5. May have inconsistent designs within and across systems
6. May violate programming standards, conventions and inconsistent documentation
7. **X module reuse** for future systems
8. **X scalability**
9. High cost of commitment on the part of key user personnel
10. **X Formal reviews and audits**
11. Some modules completed before others, hence, well-defined interfaces required
12. Difficult problems shifted to future to demonstrate early success to management
Agile Methodologies

All the above methodologies assume that any software development process is predictable & repeatable. They focus on following procedures, preparing documentation and on structure. Considered heavyweight or rigorous.

There is a movement called Agile Software Movement - a conceptual framework for undertaking software engineering projects.

- Most agile methods attempt to minimize risk by developing software in short time-boxes called iterations. Software development being essentially a human activity, will always have variations in processes and inputs and the model should be flexible enough to handle the variations.
- Each iteration = miniature software project, and includes all of the tasks necessary to release the mini-increment of new functionality: planning, requirements analysis, design, coding testing and documentation.
- While iteration may not add enough functionality to warrant releasing the product, an agile software project intends to be capable of releasing new software at the end of every iteration.

E.g. all software requirements cannot be known at the beginning of project nor do they remain static. If the model cannot handle this dynamism, then there can be lot of wastage of effort or the final product may not meet the customer's needs. Hence the agile methodologies advocate the principle “Build Short, Build Often”. That is, the given project is broken up into subprojects and each subproject is developed and integrated in to the already delivered system. This way the customer gets continuous delivery of useful and usable systems. The subprojects are chosen so that they have short delivery cycles, usually of the order of 3 to 4 weeks. The development team also gets continuous feedback.

Some of the Characteristics of Agile Methodology are as follows:
- Incremental and convergent approach that minimizes risks and facilitates functional additions
- Iterative with short cycles enabling fast verifications and corrections
- Time bound iterative cycles
- Modularity at development process level.

- People oriented
- Collaborative and communicative working style

- Some of the popular agile methodologies are - Scrum, FDD (Feature Driven Development), Crystal and XP (Extreme Programming).
SYSTEM DEVELOPMENT LIFE CYCLE (SDLC)

System development starts when management / personnel realize that business system needs improvement.

SDLC is a set of activities that analysts, designers and users carry out to develop and implement an information system.

- Activities are closely related and usually inseparable.
- Determining sequence difficult as different parts of a project can be in various phases at the same time - some components undergoing analysis while others are at advanced design stages.
- SDLC is a continuous iterative process that recycles through each stage for many applications. Each phase of SDLC uses results of previous one.

“SDLC is document driven” => at crucial stages during the process, documentation is produced.

- Phase incomplete until appropriate documentation is produced - referred 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.”

“SDLC can also be viewed from a more process oriented perspective.” => Parallel nature of 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
- Documentation for better communication and control
- Compliance of standards for better quality
- Phases are milestones for manager/user review & signoff

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

1) IS Auditor has clear understanding of the various phases if detailed documentation of SDLC
2) IS Auditor can report on compliance
3) IS Auditor, if has a technical knowledge in the area of SDLC, can guide SDLC phases
4) IS Auditor can evaluate of the methods & techniques used in SDLC phases

(-SDLC helps Auditors and Auditors help SDLC-)

Risks Associated with SDLC

Some of the shortcomings of the SDLC are as follows:

1. Development team - find it cumbersome
2. Users - end product is not visible for a long time
3. Rigidity of approach may prolong the duration of projects
4. Not suitable for small/medium sized projects

(-X Humans & Projects-)

Chap 2 | www.iscanotes.com
PHASES IN SYSTEM DEVELOPMENT LIFE CYCLE

<table>
<thead>
<tr>
<th>PHASE NAME</th>
<th>NATURE OF ACTIVITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Preliminary Investigation</td>
<td>• Evaluating strategic benefits and ensure solution fits business strategy&lt;br&gt;• Cost-benefit analysis of proposed system</td>
</tr>
<tr>
<td>2. Systems Requirements Analysis</td>
<td>• Analyzing type of system based on users needs</td>
</tr>
<tr>
<td>3. Systems Design</td>
<td>• In terms of UI, data storage and data processing functions on the basis of requirement phase by developing system flowcharts, diagrams, screens, reports</td>
</tr>
<tr>
<td>4. Systems Development / Programming</td>
<td>• Programming as designed&lt;br&gt;• Conduct continuous testing and debugging</td>
</tr>
<tr>
<td>5. Systems Testing</td>
<td>• Testing before implementation.&lt;br&gt;• Includes Unit Testing, Integration Testing and System Testing etc.</td>
</tr>
<tr>
<td>6. Systems Implementation</td>
<td>• Final Testing &amp; quality of controls audit -&gt; acceptance by mgmt &amp; user before: &lt;br&gt; o migration to live environment&lt;br&gt; o data conversion from legacy -&gt; new system</td>
</tr>
<tr>
<td>7. Post Implementation Review and Maintenance</td>
<td>• Continuous evaluation and updation</td>
</tr>
</tbody>
</table>

CID: Investigate crime -> Analyse Requirements (witness or evidence) to convict criminal -> Design & Develop a trap -> Test in court (Trial) -> Implement IPC to convict criminal -> Maintain law and order

STAGE I - 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

A preliminary investigation is normally initiated by some sort of system request. The steps involved in the preliminary investigation phase are as follows:
1) ID Problem
2) ID Objective
3) Delineation of scope
4) Feasibility Study
The following issues are typically addressed in the Feasibility Study:
1) Solution as per business strategy
2) Existing system can rectify the situation without major modification
3) Time frame for which the solution needed; Cost
4) Vendor product offers a solution

**Document / Deliverable:** A preliminary investigation report/ feasibility study for management

**Identification of Problem:**
- Several rounds of discussions with user group
- Impact on organization? (More impact = more attention/involvement)
- Shifting business requirements, changing organizational environments, and evolving IT systems inefficient
- Managers and users submit a request to IS department
- System analyst make preliminary investigation & submits proposals to steering committee

**Purpose:** To evaluate 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 project request
- Determine the size of the project
- Technical and operational feasibility of alternative approaches
- Assess costs-benefits of alternative approaches
- Report findings to the management with recommendation outlining acceptance or rejection

**Identification of Objective**

After ID of problem. E.g Railway’s inability to provide a convenient reservation system to public. Objective - “to introduce a system - book a ticket from source to destination, faster than in real-time.”

**Delineation of Scope**

Scope of a solution defines its boundaries. Should be clear, comprehensible. The following questions should be answered while stating the scope:
- **Functionality requirements:** What functionalities required?
- **Control requirements**
- **Constraints:** of input data?
- **Data** to be processed
- **Interfaces**
- **Performance requirements:** Level of response time, execution time & throughput?
- **Reliability requirements:** Remain uncorrupted despite misuse?

While eliciting information to delineate the scope, few aspects need to be kept in mind:
1) Different users will represent the problem and required solution in different ways. The system developer should bring out need from the initiator (aka champion / executive sponsor)
2) 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. Understanding their profile imp.
3) Understand the impact of the solution on the organization - its structure, roles and responsibilities. Wide impact -> greater resistance. E.g. ERP implementation
4) Quantify economic benefits to the user organization
5) Other factors besides economic benefit have to be given weight-age too

The two primary methods with the help of which the scope of the project can be analyzed are as follows (-Conducting the Investigation, analyst collect data-):

- **Reviewing internal documents**: Examine organisation charts, study written operating procedures. E.g. For inventory system proposal, analysts try to understand inventory department operations, who are managers/supervisors
- **Conducting Interviews**: Interviews reveal details, nature and reasons for requests by managers and supervisors. Gives further details and users’ views on current operations merits of proposed system

(- How do Police conduct investigation? - Through Saboot and Poochh-Tachch -)

---

**Feasibility Study**

- After possible solution options are identified, project feasibility - the “likelihood that these systems will be useful” is determined
- FS by **system analysts** - process of evaluating alternative systems through cost/benefit analysis so that most feasible system be selected

The Feasibility Study of a system is evaluated under following dimensions:
- **Legal**: Is the solution valid in legal terms?
- **Operational**: How will the solution work?
- **Schedule / Time**: Can the system be delivered on time?
- **Behavioral**: Is the solution going to bring any adverse effect on quality of work life?
- **Economic**: RoI?
- **Technical**: Is the technology needed available?
- **Financial**: Is the solution viable financially?
- **Resources**: Are HR reluctant for the solution?

(- LOST BET and thus, Financial Resources -)
**Technical Feasibility:** Related to hardware / software. Analyst ascertains technical issues during the feasibility stage of investigation:

- Application **be implemented with existing technology?**
- Equipment has the technical **capacity to hold data?**
- Necessary technology **exists or be acquired?**
- Guarantees of accuracy, reliability, ease of access, and data security?
- **Expanded?**
- Provide adequate responses to inquiries, regardless location / no. of users?

Some of the technical issues to be considered are given (Technical issues while Systems Design):

<table>
<thead>
<tr>
<th>Design Considerations</th>
<th>Design Alternatives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor (≥ 2 Ghz)</td>
<td>Micro, mini, or mainframe</td>
</tr>
<tr>
<td>Computer programs (≤ Win XP)</td>
<td>Independent vendor or in-house</td>
</tr>
<tr>
<td>Data storage medium (≥ 320GB HD)</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>Output medium (≤ LG Monitor)</td>
<td>CRT, hard copy, voice or turnaround document</td>
</tr>
<tr>
<td>Printed output (≤ HP deskjet)</td>
<td>Pre-printed forms or system-generated forms</td>
</tr>
<tr>
<td>Input medium (≤ Mouse/KB)</td>
<td>Keying, OCR, EDI or voice recognition</td>
</tr>
<tr>
<td>Transaction processing</td>
<td>Batch or online</td>
</tr>
<tr>
<td>Output scheduling</td>
<td>Pre-determined times or on demand</td>
</tr>
<tr>
<td>Output frequency</td>
<td>Instantaneous, hourly, daily, weekly, or monthly</td>
</tr>
<tr>
<td>Update frequency</td>
<td>Instantaneous, hourly, daily, weekly, or monthly</td>
</tr>
<tr>
<td>Communications network</td>
<td>Centralized or decentralized; local or distributed area</td>
</tr>
<tr>
<td>Communications channels</td>
<td>Telephone lines or fiber optics; microwave or satellite</td>
</tr>
<tr>
<td>Communications channel config.</td>
<td>Point to point, line sharing or multidrop</td>
</tr>
</tbody>
</table>

*Today’s Computers ◊ Process Output Frequently ◊ and Communicate via-NW/channel*
**Legal Feasibility:** Conflict between a newly proposed system and the organization’s legal obligations. E.g., financial reporting requirements

**Operational Feasibility:** Get views of workers, employees, customers, and suppliers about the use of computer facility. Technically feasible solutions can fail due to human problems. Questions which help in testing the operational feasibility of a project:
- Is there sufficient support from management, users?
- Are current business methods acceptable to users?
- Have the users been involved in planning and development of the project?
- Will proposed system enhance productivity?
- Will accessibility of information be lost?
This analysis - a subjective assessment of political + managerial environment. ↑ change = ↑ risk of implementation failure

**Schedule / Time Feasibility:** Design team’s estimate -> how long it will take new system to become operational -> communicating to steering committee (accept/reject)

**Behavioral Feasibility:** It refers to the systems which will be designed to process data and produce the desired outputs. No input = No success.

**Economic Feasibility/Cost-Benefit Analysis:** Evaluation of all incremental costs (time-money) and benefits. Financial / economic questions raised by analysts during preliminary investigation are for the purpose of estimating:
- Cost of conducting a full systems investigation
- Cost of hardware and software
- Benefits from reduced costs or fewer costly errors
- Opportunity cost if proposed system is not developed

**Financial Feasibility:** Proposed Solution may be prohibitively costly for the user organization.

**Resources Feasibility:** This focuses on HR. Implementing sophisticated software solutions difficult in non-metro locations because of reluctance of skilled personnel to relocate.
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.

1. **Development costs** include costs of the system development process such as:
   - Training and other start up costs
   - Salaries of analysts and programmers
   - Cost of testing and documenting the system
   - Cost of converting and preparing data files, manual etc.
   - Cost of preparing new or expanded computer facilities

   (- Cost of developing staff (training, salaries), systems (testing, data files, computer) -)

2. **Operating costs** include costs of a computer based information system such as:
   - Salaries of computer operators, others
   - Salaries of analysts and programmers
   - Overhead charges of the business firm
   - Hardware/software rental or depreciation charges
   - Cost of maintaining physical infrastructure
   - Cost of data input
   - Cost of data processing supplies

   (- Costs of operating staff (salaries, overheads), systems (hw-sw, physical infra, input supplies) -)

3. **Intangible costs** are costs that cannot be easily measured.
   - Loss of employee productivity or morale
   - Loss of Customer sales and goodwill due to errors
   - Disruptions caused by system change

   (- Costs of staff (productivity, morale), systems (disruption) -)

Benefits: resulting from developing new or improved information systems that utilize EDP are:
- Tangible benefits can be accurately measured and are directly related to the introduction of a new system. E.g. decrease in data processing cost.
- Intangible benefits such as improved business image are harder to measure and define.

Reporting Results to Management:
- Analyst reports alternative solutions, cost-benefits
- Report summarizes results with scope, recommendations
- Management uses report to decide on alternatives. Projects may not be accepted or may fail to pass feasibility test. -> Rework, resubmit, use workable part
- Investigation may suggest non-IS solutions. E.g. improvement in management and supervision
STAGE II - SYSTEM REQUIREMENT ANALYSIS

Objectives:
- Thorough & detailed understanding of current system
- Identify areas for modification to solve problem
- Determination of user/managerial requirements
- To have fair idea about various systems development tools

The following activities are performed in this phase:

1. To identify and consult stake owners to determine their expectations and resolve their conflicts
2. To analyze requirements to detect and correct conflicts and determine priorities
3. To verify requirements are complete, consistent, clear, verifiable, modifiable, testable, traceable
4. To gather data or find facts using tools like (refer Fact Finding)
5. To document activities such as (refer Fact Finding)
6. To model activities such as (- System Development Tools -)
7. Development of a system (data) dictionary to document the modeling activities

Document/Deliverable: Systems requirements report

Fact finding Techniques (How is analysis of Present System carried out? -)

Analysts interact with users to assess organisation and user needs. Various fact-finding techniques, which are used by the system analyst for determining these needs / requirements, are as follows:

1. Questionnaires: Under traditional system development approach, users and managers complete questionnaire. Large amt. of data collected from many users quickly. Responses analysed rapidly on PC.

2. Interviews: Provide complete in-depth picture of the problems and opportunities. Enables analysts to note user reaction first-hand and to probe further.

3. Observation:
   Under Prototyping approach- Necessary for successfully system development.
   Under Traditional approach- Not necessary but desirable.
   Surprise visits for real picture of users’ environment, needs

4. Documents: Documents are a very good source of information about user needs and current system. Easy to collect, convey a lot of information and provide objective data. Ensure that the collected documents are updated and accurate.
   Document means manuals, input forms, output forms, system operation diagrams, organisation charts, users’ job descriptions etc.

(-Fact finding techniques used in Job Interviews: Fill simple Questionnaire -> Interview call -> Candidate Observed during interviews, GDs -> Submit documents (mark sheets, birth certi. etc) if selected -)
Detailed investigation or Analysis of the PRESENT SYSTEM

To collect, evaluate facts about the system and the environment. Intensive survey of existing methods, procedures, data flow, outputs, files, input and internal controls to fully understand the present system and its related problems. The following areas should be studied in depth to “evaluate the effectiveness of the present system”:

1) **Review historical aspects:** Brief history of organisation - starting point for an analysis. Identify major turning points and milestones in growth. Review Annual Report, organisation charts. Identify management levels and functional areas, departments. Investigate $\text{system } A$ in the past.

2) **Analyse inputs:** Study source documents used to capture originating data because inputs are basic to manipulation of data. Outputs for one area may serve as an input for another area. Understand details of each form, content, layout, authorization procedure, origination etc. to determine how these inputs fit into the framework of the present system.

3) **Review data files maintained:** Analyst should note department, number, size, location, usage frequency, users. Information on common data files and their size is essential and is contained in the systems and procedures manuals. Must review all on-line and off-line files as they reveal information not reflected in any outputs. Important to study retrieving and processing costs of data.

4) **Review methods, procedures and data communications:** Methods and procedures transform input data into useful output. ($I \rightarrow O$)
   - 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. Aim is to eliminate unnecessary tasks or to identify improvement opportunities.

Review and understand the present data communications methods, network type, equipments.

5) **Analyse outputs:** Outputs or reports should be scrutinized carefully to assess relevance. Aim is to eliminate irrelevant reports. Analysts must understand what, why, who, when and where of information needs.

6) **Review internal controls:** Locating control points helps visualize system framework. Locate weaknesses and eliminate them. Advanced methods, procedures and equipments allow greater control over the data.

7) **Model the existing physical system and logical system:** The flow charting and diagramming of present IS helps organize facts, disclose gaps and duplication, comprehend details and problems. Proper documentation of logic of inputs, methods, procedures, data files, internal controls, data communications, reports etc.

The logical flow may be shown through system flow charts. The physical flow may be shown by employing data flow diagrams. The data flow diagrams are drawn after reviewing or developing system flow charts. Data dictionary should also be developed in tandem.

8) **Undertake overall analysis of present system:** Based upon the aforesaid investigation of the present IS, the final phase of the detailed investigation includes the analysis of present:
   - work volume
• personnel requirements
• benefits and costs

Additionally, in order to determine the user needs and to find weaknesses and strong points about the present system, analysts use various fact finding techniques (explained as above).

**Systems Analysis of Proposed Systems**

First analyse functional areas, strengths and short comings of the present system. Then clearly define specifications of the proposed system based on the desired objectives.

The *required system specifications* are as follows:

- **Input** data prepared directly from original source documents for processing by the system.
- **Methods and procedures** that relates inputs and outputs to the database, utilizes data communications.
- **Outputs** or timely managerial reports based on ‘management by exception’ principle

- **Database** with online processing capabilities
- **Work volumes and timings** considered for present and future periods including peak periods.

("Input -> Methods -> Outputs -> in Database @ right time/volume -")

**Output-to-input process**: Outputs are directly related to the objectives, hence, starting point for compiling these specifications. Thereafter, infer what inputs, database, methods, procedures and data communications must be used.

The future workload of the system must be defined for inputs, database and outputs in terms of average and peak loads, cycles and trends.
System Development Tools

System development tools and techniques help end users and systems analysts to improve current IS and to develop new ones. Its helps to:

- Conceptualise, clarify, document and communicate the activities and resources involved in the organisation and its IS
- Analyse business operations, decision making and information processing activities
- Propose and design new or improved IS to solve business problems or pursue business opportunities

The major tools used for system development can be grouped into four categories based on the systems features each document has.

User interface tools: Help design interface between end users and the computer system.
1. **Layout forms and screens**- are used to construct the formats and contents of input/output media and methods
2. **Dialogue flow diagrams**- analyse the flow of dialogues on screens generated by alternative user-responses

Data attributes and relationships tools: Define, catalogue and design data resources.
1. **Data dictionary** catalogues the description of the attributes (characteristics) of all data elements and their relationships to each other as well as to external systems.
2. **Entity-relationship diagrams**- help document the number and type of relationship among the entities in a system.
3. **File layout forms**- help document the type, size and names of the data elements in a system.
4. **Grid charts**- help identify the use of each type of data element in input/output or storage media of a system.

System components and flows tools:
Help document the data flow among the major resources and activities of an IS
1. **System component matrix**- provides a matrix framework to document the resources, activities and information output
2. **System flow charts**- show the flow of data media as they are processed by the hardware devices and manual activities
3. **Data flow diagram**- uses symbols to show the data flow among external entities (such as people or organisations, etc.), processing activities and data storage elements

Detailed system process tools: Help programmers develop detailed procedures and processes required in the design of a computer program.
1. **Decision trees and decision tables**- use a network or tabular form to document the complex conditional logic involved in choosing among the information processing alternatives in a system.
2. **Structure charts**- document the purpose, structure and hierarchical relationships of the modules in a program.

("System development tools should have User Interface like Windows XP so that Data attributes and their relationships can easily Flow into Components of System process :)"
Description of some of the major tools (documentation tools -)

- Structured English
- Flowcharts
- Data Flow Diagrams
- Decision Tree
- Decision Table
- CASE Tools
- System Components matrix
- Data Dictionary
- Layout form and Screen Generator, Menu Generator, Report generator, Code Generator

Structured English* (aka Program Design Language or Pseudo Code)

- Use of English language with the syntax of structured programming.
- Aim: Benefits of BOTH. Program logic that helps to attain precision & natural language that helps in getting the convenience of spoken languages. Its elements -
  - Operation statements written as English phrases executed from the top down.
  - Conditional blocks indicated by keywords such as IF, THEN, and ELSE.
  - Repetition blocks indicated by keywords such as DO, WHILE, and UNTIL

(Example Refer 2.29 - 2.30 = Customer has Bank A/C & no Dues -> Give Loan vs. Ask Mgmt & then give loan)

Flowcharts

- A graphic technique used by analysts to represent IPO of a business in a pictorial form.
- Represents an algorithm or process showing the steps as boxes of various kinds, and their order by connecting these with arrows.
- Used in analyzing, designing, documenting or managing a process or program in various fields
- Symbols used in Flow charts:
Types of Flow charts
Divided into four major categories:
1. Document flowchart, showing a document flow through systems
2. Data flowchart- data flows in a system
3. System flowchart- controls at a physical or resource level.
4. Program flowchart- controls in a program within a system

Benefits of Flowchart
1. Communication of system logic
2. Effective analysis
3. Proper documentation
4. Efficient Coding- Act as a guide or blueprint during systems analysis, program development phase.
5. Proper Debugging
6. Efficient Program Maintenance

Limitations of Using Flowcharts
1. Complex logic: makes flowchart complex & clumsy
2. Alterations and Modifications: Complete reqdrawing
3. Reproduction: FC Symbols cannot be typed, reproduction of flowchart becomes a problem
4. Flow lost amidst technical details

(Flowchart examples Refer Pg. 2.32-2.33)
- Draw a flowchart to find the sum of first 50 natural numbers
- Draw a flowchart to find the largest of three numbers A, B and C.
- Draw a flowchart for computing factorial N (N!) where N! = 1 * 2 * 3 * ... N

Data flow diagrams (DFD)
- 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. 4 symbols are combined to show how data are processed.

DFD Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Name</th>
<th>Explanation (Symbol represents ___ )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Data Sources and Data</td>
<td>People or organization that sends or</td>
</tr>
<tr>
<td></td>
<td>Destinations / Sinks</td>
<td>receives data within the system. [</td>
</tr>
<tr>
<td></td>
<td></td>
<td>= Square boxes called Data destinations or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Data Sinks]</td>
</tr>
<tr>
<td></td>
<td>Data flows</td>
<td>Flow of data into or out of a process. [ Curved or straight lines with arrows ]</td>
</tr>
<tr>
<td></td>
<td>Transformation process /</td>
<td>The processes that transform data from</td>
</tr>
<tr>
<td></td>
<td>Bubbles</td>
<td>inputs to Outputs. [ Circles/Bubbles ]</td>
</tr>
<tr>
<td></td>
<td>Data stores</td>
<td>Storage of data [ Two horizontal lines ]</td>
</tr>
</tbody>
</table>

Source & Destination of Data Flow Transformed into Data Stores
Decision Tree or tree diagram

- 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
- Used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal and to calculate conditional probabilities

(Flowchart examples Refer Pg. 2.34-2.35)

Decision Table

- A table which may accompany a flowchart, defining
  - possible contingencies in a program
  - appropriate course of action for each contingency
- WHY- Branches of flowchart multiply at each diamond (comparison symbol) and may easily run into hundreds. Programmer liable to miss some of the branches in FC.
- The four parts of the decision table are as follows-
  1. Condition Stub - lists conditions or comparisons
  2. Action Stub - lists actions to be taken along various program branches
  3. Condition entries - lists the possible permutations of answer to the questions in the conditions stub
  4. Action entries - lists corresponding to the condition entries, the actions contingent upon the set of answers to questions of that column

(Flowchart examples Refer Pg. 2.36-2.37)

CASE Tools

- DFD 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 - automation of anything that humans do to develop systems and support virtually all phases of traditional system development process. Ideal CASE system has 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.

System Components matrix

**System components matrix:** Useful in systems analysis and system design. Matrix showing resources used, activities performed and information produced by an IS. Highlights-
- IS Activities and processes- IPO, storage and controls
- How Resources- hardware, software and people can convert Data resources -> Information products

(Flowchart examples Refer Pg. 2.38)

Data Dictionary

Computer file containing descriptive information about the data items in the files of a business information system.

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

1. Codes: length, type (alphabet, numeric), range (0-99)
2. Source: name of source document used to create the data item
3. Files: names of the files storing the data item
4. Programs: names of programs that modify the data item
5. Access rights: Names of programs or users permitted to access/process the data item
6. Access denials: Names of programs or users not permitted to access/process the data item

Data dictionary is updated when 1) data items, 2) data fields, 3) programs are introduced/deleted

“Accountants and auditors can also make good use of a data dictionary.”

- To Auditors. E.g. establish audit trail (IPO)
- To Accountants. E.g. help plan data flow while designing new system
- For internal control procedures, documentation

**Layout form and Screen Generator, Menu Generator, Report generator, Code Generator**

- **Layout form and Screen Generator:** For printed report - easily formats or “paint” the desired layouts
- **Menu Generator:** Outlines functions which the system will 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. (e.g. MS Word, Excel)
- **Report Generator:** Performs 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:** Generate modular units of source code from the high level specifications provided by the system analyst

**System Specification**

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

- **Introduction:** Goals, objectives & description of software
- **Information Description:** Problem description; Information content, flow and structure. Hardware, software, human interfaces
- **Functional Description:** Diagrammatic representation of functions; Processing narrative for each function; Design constraints etc
- **Behavioral Description:** Response to external events AND internal controls
- **Validation Criteria:** Classes of tests to be performed to validate functions, performance and constraints
- **Appendix:** DFD, Object Diagrams; Tabular Data; Detailed description of algorithms charts, graphs etc
- **SRS Review:** It contains the following:
  - Development team makes a presentation & hands over SRS document for review by user or customer
  - Review reflects the team’s understanding of the existing processes
  - Only after ensuring that document represents processes accurately, should the user sign it. This is a technical requirement of the contract between users and development team / organization.
Roles Involved in SDLC

1. **Steering Committee**
   - Representation of affected parties
   - Direction -> Cost & timetables -> Review of progress -> corrective actions like rescheduling, re-staffing, redesign etc. (-1+4 pts. -)

2. **Project Manager**
   - Handles multiple projects -> Client communications
   - Review of progress -> Timely delivery within Budget

3. **Project Leader**
   - Dedicated to a project -> ensure completion
   - Reviews progress more frequently
   - Entire project team reports to him

4. **Systems Analyst / Business Analyst**
   - Conduct user interviews and understand requirements
   - Link between users and programmers
   - Requirements analysis & Design phase

5. **Module Leader / Team Leader**
   - Project divided into manageable modules -> development responsibility for each of them
   - Delivery of tested modules within stipulated time & cost

6. **Programmer / Coder / Developer**
   - Converts design into programs by coding; test & debug

7. **Database Administrator (DBA)**
   - Data has to be maintained by a specialist
   - DBA handles multiple projects; ensures integrity and security; helps in performance issues
   - Inclusion of new data elements only with his approval

8. **Quality Assurance**
   - Sets standards for development -> checks compliance regularly
   - X “independent” to carry out quality audits if participated in development process

9. **Tester**
   - Jr. level quality assurance personnel attached to a project
   - Tests as per plan & prepare test reports

10. **Domain Specialist**
    - Whenever a project team has to develop an application in a new field ...
    - Easier to anticipate or interpret user needs
    - A domain specialist need not have knowledge of software systems

11. **IS Auditor**
    - Ensures focus on control
    - Involved @ Design Phase and final Testing Phase to ensure ...
STAGE III - SYSTEMS DESIGN

AFTER the completion of requirements analysis for a system

Objective: Designs an IS that best satisfies user / managerial requirements.
• Describes parts of system and their interaction
• How system shall be implemented using hardware, software and network facilities
• Specifies program and DB specifications and security plan
• Specify Δ control mechanism

Activities:
• Key design phase activities include describing I & O, such as screen design and reports
• Determining processing steps & computation rules
• Determining data file or database system file design;
• Preparing the program specifications for various types of requirements or information criteria defined;
and Internal / external controls

Document / Deliverable: 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.
1. **Logical design**: The logical design is like an engineering blueprint of system’s features and shows how the elements are related to one another. (design specifications - )
2. **Physical construction**: It 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 :
   • Architectural Design;
   • Design of the Data / Information Flow;
   • Design of the Database;
   • Design of the User-interface;
   • Physical Design; and
   • Design and acquisition of the hardware/system software platform.

1) **Architectural Design**: Architectural design deals with the organization of applications in terms of hierarchy of modules and sub-modules. Here 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. 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.
2) Design of Data / Information flow

In designing the data / information flow for the proposed system, inputs that are required are: existing data / information flows; problems with the present system; and objective of the new system. All these have been identified in the analysis phase and documented in Software Requirements Specification (SRS).

3) Design of Database

Design of the database involves determining its scope ranging from local to global structure. The scope is decided on the basis of interdependence among organizational units. The greater the need for 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>Major activities in Database Designing</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Design Activity</strong></td>
</tr>
<tr>
<td>Conceptual Modeling</td>
</tr>
<tr>
<td>Data Modeling</td>
</tr>
</tbody>
</table>
| Storage Structure Design              | • Decisions must be made on how to linearize and partition the data structure so that it can be stored on some device.  
• E.g. tuples (row) in a relational data model must be assigned to records, and relationships among records might be established via symbolic pointer addresses. |
| Physical Layout Design                | • Decisions must be made on how to distribute the storage structure across specific storage media and locations. For example, the cylinders, tracks, and sectors on a disk and the computers in a LAN or WAN. |

4) Design of User Interface

Determining ways in which users will interact with a system:
• 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
• requirements for special input/output device
Designing System Outputs
Output generated imp. Output designed such that users should find the system easy to use effectively.

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 (issues) 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 (e.g.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>Data to be gathered to produce the required output for users</td>
<td>Types of data needed to generate the desired user outputs. New documents for collecting such information may be designed.</td>
<td>Weekly output report to sales manager-&gt; sales person's name, sales calls made, amount of each product sold by each</td>
</tr>
<tr>
<td>Timeliness</td>
<td>When users need outputs - regular, periodic basis (daily, weekly, monthly, quarterly or annually)</td>
<td>Data needs to be inputted to computer in time. Proper plan</td>
<td>Sales manager needs a weekly report. Airline agents, need both real-time information</td>
</tr>
<tr>
<td>Format</td>
<td>Input format refers to the manner in which data are physically arranged. Output format refers to arrangement of data output on a printed report / display screen.</td>
<td>Input formats are designed based on few constraints like - the type, length of each data field etc.</td>
<td>Should assists in decision-making, identifying and solving problems, planning and taking corrective action and searching.</td>
</tr>
<tr>
<td>Media</td>
<td>Input-output medium refers to the physical device used for input, storage or output.</td>
<td>Choice of input media and devices on which to enter the data. Alternatives - display workstations, magnetic tapes, magnetic disks, key-boards, OCR, etc.</td>
<td>Include paper, video display, microfilm, magnetic tape/disk, voice output.</td>
</tr>
<tr>
<td>Form</td>
<td>Way the information is inputted in the input form and the content is presented to users in various output forms - quantitative, text, graphics, video and audio.</td>
<td>Forms are pre-printed papers that require people to fill in responses in a standardized way. Forms extract and capture information that often will be input to the computer. Serve as source documents for the data entry personnel.</td>
<td>Output forms should be decided keeping in view user requirements.</td>
</tr>
</tbody>
</table>
**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 eff. & eff. 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**

- 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. E.g. 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.

**Design of the Hardware / System Software Platform**

Sometime, the new system requires hardware and system software not currently available in an organization. Will 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.
SYSTEM ACQUISITION - After a system is designed either partially or fully

Acquisition Standards should focus on:

- Review appropriate vendors, contract, and licensing
- Products compatible with existing systems
- Invitations-to-tender and request-for-proposals (RFP) for soliciting bids from vendors
- Operational requirements identified & detailed in RFPs.

Acquiring Systems Components from Vendors

1. Hardware Acquisition
2. Software Acquisition
3. Contracts, Software Licenses and Copyright Violations
4. Validation of Vendors’ proposals
5. Methods of Validating the proposal

1. Hardware Acquisition:
   - E.g. AC, machinery etc. - Purchased, leased from a 3rd party or to be rented?
   - Objective selection criteria can be delegated to technical specialist
   - After-sales: Support services, education and training etc.

2. Software Acquisition
   - Systems analyst determines requirement, nature of s/w based on IPO
   - Develop in-house or acquired

3. Contracts, Software Licenses and Copyright Violations
   - Contract: Rights and responsibilities of the parties - in writing with details
   - Software license: that grants permission to use/customize software
   - Copyright laws protect proprietary as well as open-source software. Using unlicensed software = possible litigation.

4. Validation of Vendors’ proposals
   - Evaluating and ranking proposals - difficult, expensive and time consuming esp. when variety of configurations
   - The following factors have to be considered towards rigorous evaluation:
     a. Costs - Benefits / Performance
     b. Maintainability
     c. Compatibility
     d. Vendor Support

5. 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:
     a. Test problems
     b. Checklists
     c. Point-Score Analysis
     d. Public Evaluation Reports
     e. Bench marking problem for vendor’s proposals

(- Test Cricket mey Points-Score hota hain aur Public Bench par baethe rehti hai -)
-----------------------------------------------------------------------
CHECKLISTS
- Most simple; subjective method
- Criteria/Parameters put in checklist as questions against which responses validated

POINT-SCORING ANALYSIS
- Objective method- no absolute rules, only guidelines for matching user needs with software capabilities

<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>

PUBLIC EVALUATION REPORTS
- Consultancy agencies compare performances, cost
- Useful where the buying staff has inadequate knowledge of facts

BENCH MARKING PROBLEM FOR VENDOR’S PROPOSALS
- Sample programs that represent a part of the buyer’s primary computer work load
- Testing whether vendor’s product meets buyer’s job requirements

TEST PROBLEMS
- Test problems disregard the actual job mix/work load
- Test the true capabilities (limits) of the hardware, software or system.
- Price-performance is compared for final judgment
- Various capabilities assigned relative weight-age
STAGE IV - 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.
- Program Coding Standards
- Programming Language & Choice of Programming Language
- Program Debugging → Test the program → Program Documentation → Program Maintenance


A “good coded program” should have the following characteristics (3R-3Vowels: RRR-AEU):
- **Reliability**: Consistent performance
- **Readability**: Ease of maintenance despite no program developer.
- **Robustness**: Considering all possible inputs and outputs
- **Accuracy**: What program is supposed to do + what it should not do
- **Efficiency**: Performance despite increase in input values.
- **Usability**: User-friendly interface and easy-to-understand document

Program Coding Standards

- Program logic outlined in flowcharts → program statements or instructions.
- 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 diligently.
- Different programmers may use different sets of instructions but each giving the same results. Hence, important to standardize
  a. Method of communication between teams, team members and hence, a good control tool
  b. Minimize the system development setbacks due to programmer turnover
  c. Provide **simplicity**, efficient utilization of **storage** and least processing **time**

Programming Language

Application programs are **coded** on the form of statements or instructions and the same is converted by the **compiler** to binary form 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- Criteria on the basis of which the language to be used should be decided:
- Application area
- Algorithmic complexity
- Data structure complexity
- Performance consideration
- Environment in which software has to be executed
- Knowledge of software development staff
- Capability of in-house staff for maintenance
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 4 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.

Test the program

A careful and thorough testing of each program is vital to the successful installation of any system.
• Programmer should plan the testing to be performed
• Test plan should include execution of all standard processing logic, testing all possible exceptions.
• Test plan should be discussed with project manager and/or system users.
• A log (record) of test results and all conditions successfully tested should be kept

Program Documentation

• Writing of narrative procedures & instructions for users
• Managers and users should review documentation for correctness → revised
• User documentation understandability i.e. user can clearly understand the instructions.

Program Maintenance

• Requirements of applications change → Modification.
• Usually separate categories of programmers called maintenance programmers for this
STAGE V - SYSTEM TESTING

Testing is a process used to

• Assess correctness, completeness, reliability & quality of developed computer software.
• Systematically uncover different classes of errors in a minimum time - effort
• However, testing can’t show absence of defect, it can only show that software defects are present.

Different LEVELS OF TESTING are as follows (UISF):

• Unit Testing
• Integration Testing
• System Testing
• Final Acceptance Testing

Unit Testing

Written and run by software developers, UT is a software verification and validation method in which a programmer tests if individual units of source code are fit for use.

• A unit = smallest testable part of an application - may be an individual program, function, procedure etc
• Goal - to isolate each part of the program and show that the individual parts are correct

There are 5 categories of tests that a programmer typically performs on a program unit:

1. Parallel Tests
2. Structural Tests
3. Performance Tests
4. Functional Tests
5. Stress Tests

<table>
<thead>
<tr>
<th>Parallel Tests</th>
<th>• Same test data is used in the new and old system → output results are then compared.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structural Tests</td>
<td>• Examining the internal processing logic of a software system</td>
</tr>
<tr>
<td>Performance Tests</td>
<td>• Verify response time, execution time, throughput, memory utilization and traffic rates</td>
</tr>
<tr>
<td>Functional Tests</td>
<td>• Check ‘whether programs do what they are supposed to do or not’.</td>
</tr>
<tr>
<td></td>
<td>• Specifies operating conditions, inputs, and expected results → input values → whether actual result match expected result match.</td>
</tr>
<tr>
<td>Stress Tests</td>
<td>• To determine stability → testing beyond normal operational capacity i.e. to a breaking pt.</td>
</tr>
<tr>
<td></td>
<td>• Overload to determine limitations &amp; situation handling</td>
</tr>
</tbody>
</table>

(- In Test cricket, Past Performance cannot tell how will you Function in a Stressful condition next time. -)

Types of Unit Testing

• Static Analysis Testing - Desk Check, Structured walk-through, Code inspection
• Dynamic Analysis Testing - Black Box Testing, White Box Testing, Gray Box Testing
Static Analysis Testing

- Walk karke har-ek Desk ko Closely Inspect karta hai

<table>
<thead>
<tr>
<th>Structured walk-through</th>
<th>Application developer leads other programmers through the text of program &amp; explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Desk Check</td>
<td>Programmer checks for logical syntax errors, and deviation from coding standards</td>
</tr>
<tr>
<td>Code inspection</td>
<td>Formal committee → Review is done with formal checklists</td>
</tr>
</tbody>
</table>

Dynamic Analysis Testing (3 Software testing techniques; Test design)

- Takes an external perspective of the test object to derive test cases. No knowledge of the test object's internal structure.
- Tests can be functional or non-functional
- The test designer selects valid and invalid inputs and determines the correct output.
- Applicable to all levels of software testing: unit, integration, functional testing, system and acceptance.
- Higher the level → bigger and more complex the box → 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.”

Black Box (BBT)

- Uses an internal perspective of the system to design test cases based on internal structure.
- Requires programming skills to identify all paths through the software.
- Tester chooses test case inputs to exercise paths through the code and determines the appropriate outputs.
- Applicable to: unit, integration and system levels of the testing process
- 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.
- Ensures that the internal operation of the product conforms to specifications and all the internal components are adequately exercised

Gray Box (GBT)

- Uses a combination of BBT & WBT
- Tester applies a limited number of test cases to internal workings of the s/w under test
- For remaining, one takes a black box approach in applying inputs to the software under test and observing the outputs.
Integration Testing

A software testing activity in which *individual software modules are combined* and *tested as a group.*

**Objective:** to evaluate connection of 2 or more components that pass information from one area to another. It takes as its input - modules that have been unit tested → groups them in larger aggregates → applies tests → delivers as its output the integrated system ready for system testing. This is carried out in the following manner:

<table>
<thead>
<tr>
<th>Bottom-up Integration</th>
<th>Top-down Integration</th>
<th>Regression Testing</th>
</tr>
</thead>
</table>
| • *Traditional* strategy that integrates components of a s/w system into a functioning whole | • Starts with the main routine, and stubs are substituted, for the modules directly subordinate to the main module.  
  o STUB *** = 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  
  o Lacks implementing details of the function or the program being executed | • Each time a new module is added → the software changes → New data flow paths are established & new I/O may occur & new control logic is invoked → These changes may cause problems with functions that previously worked flawlessly.  
  • Regression tests ensure that changes or corrections have not introduced new errors. |
| • First unit testing → sub-system testing → testing of the entire system | • 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.  
  • (-) High-level modules are tested, not with real outputs from subordinate modules, but from stubs. | |
| • (+) Easy to implement as tested subordinate modules are available during module testing | | |
| • (-) Testing of major decision / control points is deferred to a later period. | | |

(*** Illustration for understanding: While making a room’s furniture, carpenters mark the area where bookshelf has to be made. Even though, the bookshelf is INCOMPLETE, it helps them carry on making OTHER furniture pieces as space for bookshelf has already been declared. After rest of the furniture is made, they make and fix the bookshelf at its declared place)
System Testing

A process in which software and other system elements are tested as a whole.

- Software as a whole is operational or when well defined subsets of software's functionality have been implemented.
- Purpose: to ensure that the new or modified system functions properly.
- Test procedures performed in a non-production test environment. Types of testing that might be carried out:

<table>
<thead>
<tr>
<th>Testing Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recovery Testing</td>
<td>Testing ‘how well the application is able to recover from crashes, hardware failures and other similar problems’ (forced failure)</td>
</tr>
<tr>
<td>Security Testing</td>
<td>Determine that an Information System protects data and maintains functionality as intended or not. Security concepts include - CIA. Ensures the existence and proper execution of access controls in the new system.</td>
</tr>
<tr>
<td>Stress or Volume Testing</td>
<td>To determine the stability of a given system or entity → testing beyond normal operational capacity, often to a breaking point, in order to observe the results. E.g. large quantity of data during peak hours</td>
</tr>
<tr>
<td>Performance Testing</td>
<td>To determine the speed or effectiveness of a computer, network, software program etc. Compares new system's performance with that of similar systems using well defined benchmark</td>
</tr>
</tbody>
</table>

Final Acceptance Testing

It is conducted when the system is just ready for implementation. Ensured that the new system satisfies quality standards and user needs. Thus, the final acceptance testing has two major parts:

<table>
<thead>
<tr>
<th>Testing Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quality Assurance Testing</td>
<td>Ensures that new system satisfies the quality standards</td>
</tr>
<tr>
<td></td>
<td>Ensures development process is as per the organization’s quality assurance methodology.</td>
</tr>
<tr>
<td>User Acceptance Testing</td>
<td>Ensures that functional aspects expected by the users have been well addressed in the new system. There are two types of the user acceptance testing:</td>
</tr>
<tr>
<td></td>
<td>o Alpha Testing: This is the first stage, often performed by users within org</td>
</tr>
<tr>
<td></td>
<td>o 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.</td>
</tr>
</tbody>
</table>
STAGE VI - 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.


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.

Activities during Implementation Stage

A. Equipment Installation

Necessary hardware should be ordered in time to allow for installation/testing of equipment. Installation checklist is made. Adequate time should be allocated for the following activities:

1. 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.

2. Installation of new hardware / software- Equipment is physically installed by manufacturer. Connected to the power source and wired to communication lines.

3. Equipment check out- The equipment must be turned on for testing under normal operating conditions. Vendor runs routine 'diagnostic tests' while implementation team devises and run extensive tests. Ensure equipment is in proper working condition.

B. Training Personnel

A system can succeed or fail depending on the way it is operated and used. Quality of training important. 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.

C. System Implementation Conversion Strategies:

“Conversion or changeover is the process of changing from the old system (manual system) to the new system. 4 types of implementation strategies are as follows”: 
**Direct Implementation / Abrupt change-over:** Abrupt takeover - an “All or Nothing” approach. Changeover is done in one operation, completely replacing the old system in one go. Pre-set date.

Phased implementation: Implementation can be staged with conversion to the new system taking place by degrees. E.g. - 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 next phase is started, eventually leading to the final phase when the new system fully replaces the old one.

Pilot implementation: 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. E.g. - 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.

Parallel running implementation: 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.
Activities involved in conversion

Activities involved in conversion: Conversion includes all those activities which must be completed to successfully convert from the previous system (e.g. manual system) to the new information system (e.g. computer system). These activities can be classified as follows:

Procedure conversion: Operating procedures for the new system should be documented. Before any parallel or conversion activities can start, all users informed of Δs IPO, data files, methods, and internal control in understandable terms. Written operating procedures + oral communication during training sessions & brief meetings.

File conversion: Large files of information must be converted from one medium to another before programming and testing are completed. On-line files (common database) or off-line files.
- File conversion programs must be tested for accuracy
- Adequate control
- Existing files should be kept (backup) 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.

System conversion: After on-line and off-line files have been converted and the reliability of the new system has been confirmed, daily processing can be shifted from existing information system to new one.
- All transactions initiated after this time are processed on new system
- System development team members should be present to assist, answer queries
- Operating old system for some more time to permit checking & balancing total results of both

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.

(→ PFS Conversion with Scheduled Personnel/equipment ↓)
STAGE VII - POST IMPLEMENTATION REVIEW AND SYSTEMS MAINTENANCE

Objective: To ASSESS and REVIEW complete working solution

Activities: Some of the Systems maintenance activities are as follows:
- Adding new data elements
- Modifying reports or adding new
- 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?

Post Implementation Review

- PIR answers “Did we achieve what we set out to do in business terms?”
- PIR ascertains degree of success from the project, extent to which it met its objectives, delivered planned levels of benefit, and addressed the specific requirements as originally defined
- PIR examines if further improvements can be made
- PIR should be scheduled some time after the solution has been deployed. 6 weeks - 6 months
- 2 basic dimensions of Information system that should be evaluated.
  - whether new system is operating properly
  - whether user satisfied with information system reports

Evaluation of the information system in terms of...

Development evaluation: Whether system was developed on schedule and within budget?
- Schedules and budgets to be established in advance
- Records of actual performance and cost be maintained
- BUT, X clearly defined schedules or budgets. Due to uncertainty, they are not subjected to traditional management control procedures

Operation evaluation: Whether the hardware, software and personnel are capable to perform their duties?
- Evaluation straightforward if evaluation criteria are established in advance.

Information evaluation: To what extent information provided by system is supportive to decision making?
- Cannot be conducted in a quantitative manner, unlike earlier 2 cases.

System Maintenance

Systems usually require modification/updates after development. Causes: Failure to anticipate all requirements during system design; changing organizational and operational requirements.

Maintenance can be categorized in the following two ways:
- Scheduled
- Corrective
- Rescue
- Adaptive
- Perfective
- Preventive
1. **Scheduled maintenance:** Anticipated and can be planned for. E.g. new inventory coding scheme.

2. **Corrective maintenance:** Deals with fixing bugs in the code or defects found. A defect can result from design errors, logic errors, coding error etc. Starts after bug reports drawn up by the end users. E.g. correcting a failure to test for all possible conditions or a failure to process the last record in a file.

3. **Rescue maintenance:** Not anticipated - requires immediate solution. Previously undetected malfunctions. Occurs when system not properly developed and tested.

4. **Adaptive maintenance:** Adapting software to As in environment, such as the hardware or OS. Monitor environment. Environment = business rule, government policies, work patterns, software and hardware operating platforms.

5. **Perfective maintenance:** Deals with accommodating to new or changed user requirements and concerns functional enhancements, system’s performance, user interface.

6. **Preventive maintenance:** Deals with activities aimed at increasing system’s maintainability, such as updating documentation, adding comments, and improving the modular structure of the system.
   - The long-term effect of corrective, adaptive & 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.

**OPERATION MANUALS / User’s guide**

- **Technical communication** document to assist people using a particular system.  
- Usually written by a technical writer. In small companies, written by programmers, product or project managers, or other technical staff.  
- 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 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 + ways 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 (Refer Pg 2.63)*
ORGANIZATIONAL STRUCTURE OF IT DEPARTMENT

Management Structure:
1. Line management Structure
2. Project Management Structure

1) Line management Structure:

Responsibility:
- Provide a stable environment in which information systems are built and operated
- Ensure smooth day-to-day operation
- Development, implementation, operation and maintenance of systems as per plan

Organisation hierarchy (levels of management subsystems) and major functions performed within a data processing installation:

- Ensure data processing installation is well managed; long run policies decisions
- Planning and control of all computer activities; break up long term policies into short-term goals
- Design, implementation and maintenance of systems
- Programming new systems, maintaining old systems and providing general systems support software
- Control & use of database & library
- Physical security of the data processing and IS programs
- Data processing and preparation, hardware
- Authenticity, accuracy and completeness of IPO

(Top 8 InstituteS for MDP or Management Development Programs have very good Data & Security Administration. Ensures Optimal Quality of students - )
2) Project Management Structure

Steering committee:
- In Project management, Project requests are submitted to and prioritized by the steering committee.

Project team:
- Project manager - with complete operational control and appropriate resources
- IS auditors - as control advocates and experts to provide an independent, objective review

![Diagram of IS Manager roles]

Duties and Responsibilities

The structure of an IT Department is divided into two main areas of activity:
- Information processing (IP) - Concerned with the operational aspect of the information-processing environment and includes computer operations, systems programming, telecom & librarian functions.
- System development and enhancement - Concerned with the development, acquisition and maintenance of computer application systems and performs systems analysis and programming functions.

<table>
<thead>
<tr>
<th>ROLE</th>
<th>... ENSURES / ... are RESPONSIBLE for... / ... MANAGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data entry (supervisor)</td>
<td>• Data is authorized, accurate and complete when entered into the system</td>
</tr>
<tr>
<td></td>
<td>• Data input (Raw data or instructions) must be validated, errors detected, input resubmitted</td>
</tr>
<tr>
<td></td>
<td>• Info includes: 1) raw data to be processed 2) instructions to the system</td>
</tr>
<tr>
<td>File Library (Librarian)</td>
<td>• Recording, issuing, receiving, safeguarding &amp; managing all programs &amp; data files</td>
</tr>
<tr>
<td></td>
<td>1. Files must be used only for the purposes intended = Control</td>
</tr>
<tr>
<td></td>
<td>2. Storage media used for files must be maintained in correct working order</td>
</tr>
<tr>
<td></td>
<td>3. File backup strategy and file retention strategy must be implemented</td>
</tr>
<tr>
<td>Control Group</td>
<td>• Manages flow of data</td>
</tr>
<tr>
<td></td>
<td>• Collection, conversion and control of input</td>
</tr>
<tr>
<td></td>
<td>• Balancing distribution of output to the users</td>
</tr>
<tr>
<td>Operations</td>
<td>Daily running of hardware &amp; software facilities so that 1) production application system can accomplish their work and 2) development staff can design, implement and maintain systems.</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td>• 7 major functions:</td>
</tr>
<tr>
<td></td>
<td>o Computer operations</td>
</tr>
<tr>
<td></td>
<td>o Data preparation</td>
</tr>
<tr>
<td></td>
<td>o File library</td>
</tr>
<tr>
<td></td>
<td>o Documentation library</td>
</tr>
<tr>
<td></td>
<td>o Production work flow control</td>
</tr>
<tr>
<td></td>
<td>o Communication network control</td>
</tr>
<tr>
<td></td>
<td>o Performance monitoring</td>
</tr>
<tr>
<td>Security Administration</td>
<td>Matters of physical security</td>
</tr>
<tr>
<td></td>
<td>Physical facilities safe from threats that affect the continuity of operation</td>
</tr>
<tr>
<td>Physical Security</td>
<td>A complete reliable protection scheme for physical attacks on the database, storage devices</td>
</tr>
<tr>
<td>Data Security</td>
<td>Utility access and submission, as well as monitoring and performance tools, should be restricted to appropriate personnel.</td>
</tr>
<tr>
<td>Conducting a security program</td>
<td>Security program: A series of ongoing, regular, periodic evaluations conducted to ensure that the physical facilities of an IS are safeguarded adequately</td>
</tr>
<tr>
<td></td>
<td>• 1st evaluation: list of possible threats</td>
</tr>
<tr>
<td></td>
<td>• Subsequent evaluation: focus only on changes</td>
</tr>
<tr>
<td></td>
<td>Periodic</td>
</tr>
<tr>
<td>Production Work Flow Control</td>
<td>Responsibility of a control section. Manages flow of data between users and the information system, and between data preparation and the computer room.</td>
</tr>
<tr>
<td></td>
<td>Difficult for operators and data preparation personnel to collude and to perpetrate a fraud</td>
</tr>
<tr>
<td>Quality Assurance</td>
<td>Testing and verifying whether the programs, program changes and documentation adhere to standards and naming conventions</td>
</tr>
<tr>
<td>Systems Analysis</td>
<td>Responsible for interpreting the needs of the user, determining the programs and the programmers necessary to create application</td>
</tr>
<tr>
<td></td>
<td>Auditor key participant in the system development process</td>
</tr>
<tr>
<td></td>
<td>o System effectiveness: Design meets strategic requirements</td>
</tr>
<tr>
<td></td>
<td>o System efficiency: Resources available</td>
</tr>
<tr>
<td></td>
<td>o Safeguarding access and data integrity: System controls present</td>
</tr>
<tr>
<td>Applications Programming</td>
<td>Responsible for developing new systems and for monitoring systems in production</td>
</tr>
<tr>
<td></td>
<td>Work in a test-only environment- Should not move test versions into the production environment</td>
</tr>
<tr>
<td></td>
<td>Application programmers should not have access to system program libraries.</td>
</tr>
<tr>
<td>Systems programming</td>
<td>Maintaining the systems software, OS</td>
</tr>
</tbody>
</table>
### LAN Administration

- Technical and administrative control over the **LAN**
- E.g. proper functioning of transmission links, system backups, purchase processes and overall security

### Help Desk Administration

- Monitoring, improving and controlling system performance in (a) mainframe, (b) client/server hardware and software
- Useful: 1) if users are uncertain about the nature or format of the data to be entered into a particular field, 2) if inexperienced or novice users will submit data 3) validation rules