CH-5 Acquisition, Development and Implementation of Information Systems

This chapter is Divided in to 3 Divisions

Division-1 - SYSTEM DEVELOPMENT METHODOLOGY

Waterfall Model	Prototyping Model	Incremental Model	Spiral Model	RAD Model	Agile Model

Division -2 – SDLC – 8 Phases

Preliminary Investigation	Systems Requirements Analysis	Systems Design	Systems Acquisition	Systems Development	Systems Testing	Systems Implementation	Post Implementation Review and Maintenance

Division -3- Miscellaneous

Achieving System Development Objectives
Accountants' Involvement in Development Work

Division-1: SYSTEM DEVELOPMENT METHODOLOGY

SDM is a formalized, standardized, well-organized and documented set of activities used to manage a system development project.

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

- The project is divided into a number of identifiable processes, and each process has a starting point and an ending point.

Each process comprises several activities, one or more deliverables, and several management control points.

The division of the project into these small, manageable steps facilitates both project planning and project control.

-Specific reports and other documentation, called Deliverables must be produced periodically during system development.

-Users, managers, and auditors are required to participate in the project, which generally provide approvals, often called signoffs, at pre-established management control points.

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

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

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

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

MODEL-1 Waterfall Model

The waterfall approach is a traditional development approach in which each phase is carried in sequence or linear fashion.

In this traditional approach of system development, activities are performed in sequence.

These phases include

requirements analysis,
specifications and design requirements,
coding,
final testing, and
release.

3-key characteristics are the following

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

2. Emphasis is on

planning,
time schedules,
target dates,
budgets and implementation of an entire system at one time.

3. Tight control is maintained over the life of the project through the use of extensive written documentation, as well as through formal reviews and approval

by the user and IT management

occurring at the end of most phases before beginning the next phase.

4-Strengths

1.It is ideal for supporting less experienced project teams and project managers, whose composition fluctuates.

2.The orderly sequence of development steps and design reviews help to ensure the quality, reliability, adequacy and maintainability of the developed software.

3.Progress of system development is measurable.

4.It enables to conserve resources.

10- Weaknesses

1. It promotes the gap between users and developers with clear vision of responsibility.

2.Written specifications are often difficult for users to read and thoroughly appreciate.

3. It leads to excessive documentation and often time-consuming.

4. It is difficult to respond to changes, which may occur later in the life cycle, and if undertaken it proves costly and are thus discouraged.

5. System performance cannot be tested until the system is almost fully coded.

6. Problems are often not discovered until **system testing.**

7.Requirement inconsistencies, missing system components and unexpected development needs are often discovered during **design and coding.**

8. There is a little to iterate, which may be essential in situations.

9. Project progresses forward, with only slight movement backward.

10. It is criticized to be inflexible, slow, costly, and cumbersome due to significant structure and tight controls.

MODEL-2 Prototyping Model

The goal of prototyping approach is to develop a small or pilot version called a prototype of part or all of a system.

A prototype is a usable system component that is built quickly and at a lesser cost, and with the intention of modifying/replicating/expanding or even replacing it by a full-scale and fully operational system.

As users work with the prototype, they learn about the system criticalities and make suggestions about the ways to manage it.

These suggestions are then incorporated to improve the prototype, which is also used and evaluated.

Finally, when a prototype is developed that satisfies all user requirements, either it is refined and turned into the final system or it is scrapped.

If it is scrapped, the knowledge gained from building the prototype is used to develop the real system.

Prototyping can be viewed as a series of 4 steps:

The generic phases of this model are explained as follows: **(IDT-O)**

1. Identify Information System Requirements:

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

2. Develop the Initial Prototype:

The designers create an initial base model and give little or no consideration to internal controls, but instead emphasize system characteristics such as simplicity, flexibility, and ease of use.

These characteristics enable users to interact with tentative versions of data entry display screens, menus, input prompts, and source documents.

The users also need to be able to respond to system prompts, make inquiries of the information system, judge response times of the system, and issue commands.

3.Test and Revise:

After finishing the initial prototype, the designers **first demonstrate** the model to users and then give it to them to experiment and ask users to record their likes and dislikes about the system and recommend changes.

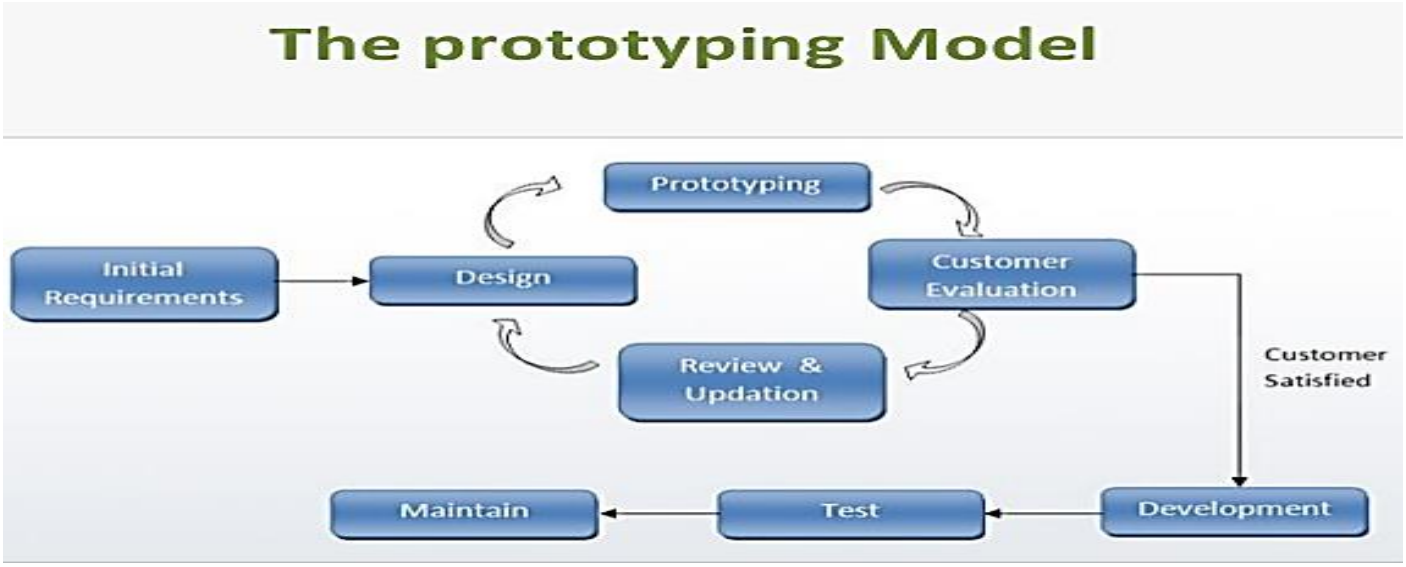
Using this feedback, the design **team modifies** the prototype as necessary and then resubmits the revised model to system users for reevaluation.

Thus iterative process of modification and reevaluation continues until the users are satisfied.

4.Obtain User Signoff of the Approved Prototype:

Users formally approve the final version of the prototype, which commits them to the current design and establishes a **contractual obligation** about what the system will, and will not, do or provide.

Prototyping is not commonly used for developing traditional applications such as accounts receivable, accounts payable, payroll, or inventory management, where the inputs, processing, and outputs are well known and clearly defined.



10-Strengths

1. It improves both user participation in system development and communication among project stakeholders.

2. It is especially useful for

resolving unclear objectives;
developing and validating user requirements;
experimenting with or comparing various design solutions, or
investigating both performance and the human computer interface.

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

4. It helps to easily identify, confusing or difficult functions and missing functionality.

5. It enables to generate specifications for a production application.

6. It encourages innovation and flexible designs.

7. It provides for quick implementation of an incomplete, but functional, application.

8. It typically results in a better definition of these users' needs and requirements than does the traditional systems development approach.

9. A very short time period is normally required to develop and start experimenting with a prototype.

10. Since system users experiment with each version of the prototype through an interactive process, errors are hopefully detected and eliminated early in the developmental process.

9-Weaknesses

1. Prototyping may cause behavioral problems, if, system developers are unable to meet all user demands.

2. The interactive process of prototyping causes the prototype to be experimented with quite extensively.

3. Inadequate testing can make the approved system error-prone, and inadequate documentation makes this system difficult to maintain.

4. Prototyping can only be successful if the system users are willing to devote **significant time** in experimenting with the prototype and provide the system developers with change suggestions.

5. Prototype may not have sufficient checks and balances incorporated.

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

7.Requirements may frequently change significantly.

8.Identification of non-functional elements is difficult to document.

9.Approval process and control are not strict.

MODEL-3 Incremental Model

The Incremental model is a method of software development where the model is designed, implemented and tested incrementally (a little more is added each time) until the product is finished.

This model combines the elements of the waterfall model with the iterative philosophy of prototyping.

The product is decomposed into a number of components, each of which are designed and built separately (termed as builds as shown in diagram).

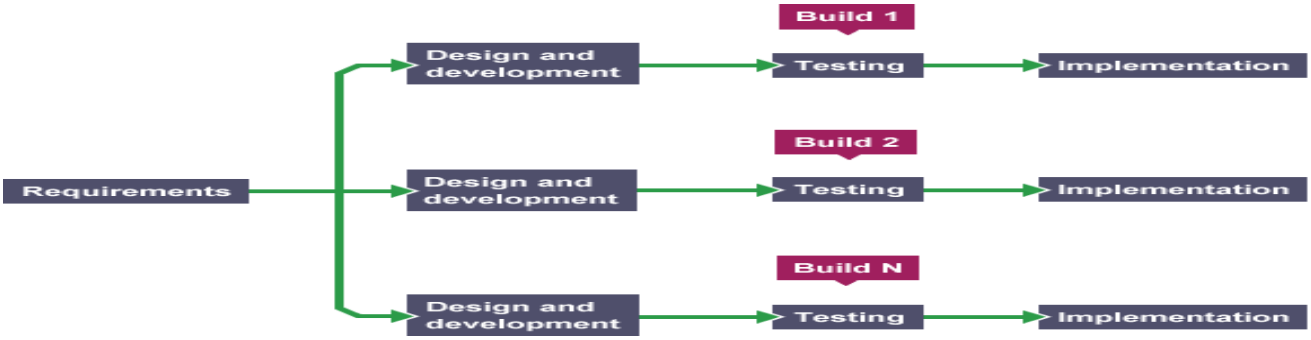
Each component is delivered to the client when it is complete.

3- few pertinent features are:

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

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

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



7-Strengths

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

2. Moderate control is maintained over the life of the project through the use of written documentation and the formal review and approval by the

user and information technology management at designated major milestones.

3. Stakeholders can be given concrete evidence of project status throughout the life cycle.

4. It is more flexible and less costly to change scope and requirements.

5. It helps to mitigate integration and risks earlier in the project.

6. It allows the delivery of a series of implementations that are gradually more complete and can go into production more quickly as incremental releases.

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

5-Weaknesses

1. lack of overall consideration of the business problem and technical requirements for the overall system, before moving onto the next increment.

2. Each phase of an iteration is rigid and do not overlap each other.

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

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

5. It is difficult to demonstrate early success to management.

Model-4 Spiral Model

The Spiral model is a software development process combining elements of both design and prototyping-in-stages.
It tries to combine advantages of top-down and bottom-up concepts.
It combines the features of the prototyping model and the waterfall model.
The spiral model is intended for large, expensive and complicated projects.

Game development is a main area where the spiral model is used and needed, that is because of the size and the constantly shifting goals of those large projects. A list of **pertinent characterizing features** includes the following:

1. The new system requirements are defined in as much detail as possible.

This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.

2. A preliminary design is created for the new system.

This phase is the most important part of "Spiral Model" in which all possible alternatives that can help in developing a cost effective project.

This phase has been added specially in order to identify and resolve all the possible risks in the project development.

If risks indicate any kind of uncertainty in requirements, prototyping may be used to proceed with the available data and find out possible solution in order to deal with the potential changes in the requirements.

3. A first prototype of the new system in constructed from the preliminary design.

This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.

4. A second prototype is evolved by a 4 times procedure by evaluating the

first prototype in terms of its strengths, weaknesses, and risks;
defining the requirements of the second prototype;
planning and designing the second prototype; and
constructing and testing the second prototype.

3- Strengths

1. It enhances the risk avoidance.

2. It is useful in helping for optimal development of a given software iteration based on project risk.

3. It can incorporate Waterfall, Prototyping, and Incremental methodologies as special cases in the framework, and provide guidance as to which combination of these models best fits a given software iteration, based upon the type of project risk.

5- Weaknesses

1. There are no firm deadlines, cycles continue with no clear termination condition leading to, inherent risk of not meeting budget or schedule.

2. No established controls exist for moving from one cycle to another cycle.

Without controls, each cycle may generate more work for the next cycle.

3. A skilled and experienced project manager is required to determine how to apply it to any given project.

4. It may prove highly customized to each project, and thus is quite complex and limits reusability.

5. It is challenging to determine the exact composition of development methodologies to use for each iteration around the Spiral.

Model-5 Rapid Application Development (RAD) Model

Rapid Application Development (RAD) refers to a type of software development methodology;

which uses minimal planning in favor of rapid prototyping.

The planning of software developed using RAD is inserted pages with writing the software itself.

The lack of extensive pre-planning generally allows software to be written much faster, and makes it easier to change requirements.

10-Key features include the following:

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

2. Attempts to reduce inherent project risk by breaking a project into smaller segments and

providing more ease-of-change during the development process.

3. Aims to produce high quality systems quickly, primarily through the use of iterative Prototyping (at any stage of development), active user involvement, and computerized development tools.

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

5. Project control involves prioritizing development and defining delivery deadlines or "time boxes."

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

7. Active user involvement is crucial.

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

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

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

8- Strengths

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

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

3. Quick initial reviews are possible.

4.Constant integration isolates problems and encourages customer feedback.

5.It holds a great level of commitment from stakeholders, both business and technical, than Waterfall, Incremental, or spiral frameworks.

6.It concentrates on essential system elements from user viewpoint.

7.It provides for the ability to rapidly change system design as demanded by users.

8.It leads to a tighter fit between user requirements and system specifications.

9- Weaknesses

1. Fast speed and lower cost may affect adversely the system quality.

2. The project may end up with more requirements than needed (gold-plating).

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

4. It may lead to inconsistent designs within and across systems.

5. It may call for violation of programming standards related to inconsistent naming conventions and inconsistent documentation.

6. It may call for lack of attention to later system administration needs built into system.

7. Formal reviews and audits are more difficult to implement than for a complete system.

8. Tendency for difficult problems to be pushed to the future to demonstrate early success to management.

9. Since some modules will be completed much earlier than others, well–defined interfaces are required.

Model-6 Agile Model

It is a conceptual framework that promotes foreseen interactions throughout the development life cycle.

Agile Manifesto is based on following 12 features:

1. Customer satisfaction by rapid delivery of useful software.
2. Welcome changing requirements, even late in development.
3. Working software is delivered frequently.
4. Working software is the principal measure of progress.
5. Sustainable development, able to maintain a constant pace.
6. Close, daily co-operation between business people and developers.
7. Face-to-face conversation is the best form of communication.
8. Projects are built around motivated individuals, who should be trusted.
9. Continuous attention to technical excellence and good design.
10. Simplicity.
11. Self-organizing teams.
12. Regular adaptation to changing circumstances.

5- Strengths

1. Agile methodology has the concept of an adaptive team, which enables to respond to the changing requirements.

2. The team does not have to invest time and efforts and finally find that by the time they delivered the product, the requirement of the customer has changed.

3. Face to face communication and continuous inputs from customer representative leaves a little space for guesswork.

4. The documentation is crisp and to the point to save time.

5. The end result is generally the high quality software in least possible time duration and satisfied customer.

6- Weaknesses

1. it is difficult to assess the efforts required at the beginning of the SDLC.

2. There is lack of emphasis on necessary designing and documentation.

3. By nature, Agile projects are extremely light on documentation.

4. Agile requires more re-work and due to the lack of long-term planning and the lightweight approach to architecture.

5. The project can easily get taken off track if the customer representative is not clear about the final outcome.

6. Agile lacks the attention to outside integration.

DIVISION-2: SDLC

Perspective of the IS Audit, the following are the possible advantages

IS auditor	-can provide an evaluation of the methods and techniques used through the various development phases of the SDLC. -can have clear understanding of various phases of the SDLC on the basis of the detailed documentation created during each phase of the SDLC. -if has a technical knowledge and ability of different areas of SDLC, can be a guide during the various phases of SDLC. -on the basis of his examination, can state in his report about the compliance by the IS management of the procedures, if any, set by the management.

PHASE-1 Preliminary Investigation

(i) Identification of Problem

The first step in an application development is to define the problem clearly and precisely, which is done only after the critical study of the existing system and several rounds of discussions with the user group.

Managers and users may feel compelled to submit a request for a new system to the IS department, irrespective of the reason.

A system analyst is assigned to perform preliminary investigation who submits all proposals to the steering committee for evaluation to identify those projects that are most beneficial to the organization.

The analyst working on the preliminary investigation should accomplish the following objectives:**(CAR-DD)**

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

(ii) Identification of objectives

After the identification of the problem, it is easy to work out and precisely specify the objectives of the proposed solution.

For instance, inability to provide a convenient reservation system, for a large number of intending passengers was the problem of the Railways.

(iii) Delineation(description) of Scope

The scope of a solution defines its typical boundaries.
It should be clear and comprehensible to the user management stating the extent and 'what will be addressed by the solution and what will not'.
Often, the scope becomes a controversial issue between development and user organizations.
Hence, drawing the scope in the beginning is essential and proves quite handy.

Moreover, while eliciting (evoking) information to delineate the scope, few aspects needs to be kept in mind:

1. Different users may represent the problem and required solution in different ways.

The system developer should evoke the need from the initiator of the project alternately called champion or executive sponsor of the project, addressing his concerns should be the basis of the scope.

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.

An understanding of their profile helps in designing appropriate user interface features.

3. While presenting the proposed solution for a problem, the development organization has to clearly quantify the economic benefits to the user organization.

The information required has to be gathered at this stage.

4. It is also necessary to understand the impact of the solution on the organization- its structure, roles and responsibilities.

Solutions, which have a wide impact, are likely to be met with greater resistance.

5. While economic benefit is a critical consideration when deciding on a solution, there are several other factors that have to be given weightage too.

These factors are to be considered from the perspective of the user management and resolved.

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

Reviewing Internal Documents: The **analysts** conducting the investigation first try to learn about the organization involved in, or affected by, the project. **For example,** to review an inventory system proposal, an analyst may try to know how does the inventory department operates and who are the managers and supervisors. **Analysts** can usually learn these details by examining organization charts and studying written operating procedures.	**Conducting Interviews:** Written documents tell the analyst how the systems should operate, but they may not include enough details to allow a decision to be made about the merits of a systems proposal, nor do they present users' views about current operations. To learn these details, analysts use interviews. Interviews allow analysts to know more about the nature of the project request and the reasons for submitting it.

(iv) Feasibility Study

After possible solution options are identified, project feasibility i.e. the likelihood that these systems will be useful for the organization is determined.

A feasibility study is carried out by the system analysts, which refers to a process of evaluating alternative systems through cost/benefit analysis so that the most feasible and desirable system can be selected for development.

The Feasibility Study of a system is evaluated under following dimensions described briefly as follows-**(BROS-LEFT)**

Behavior Feasibility

It refers to the systems, which is to be designed to process data and produce the desired outputs.

However, if the data input for the system is not readily available or collectable, then the system may not be successful.

Resource Feasibility: This focuses on human resources.

Implementing sophisticated software solutions becomes difficult at specific locations because of the reluctance of skilled personnel to move to such locations.

Operational Feasibility: It is concerned with ascertaining the views of workers, employees, customers and suppliers about the use of computer facility.

A system can be highly feasible in all respects except the operational and fails miserably because of human problems.

Some of the **questions, which help** in testing the operational feasibility of a project, may include the following:

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

Schedule Feasibility:

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

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

Legal Feasibility:

Legal feasibility is largely concerned with whether there will be any conflict between a newly proposed system and the organization's legal obligations.

Any system, which is liable to violate the local legal requirements, should also be rejected.

Economic Feasibility:

It includes an evaluation of all the incremental costs and benefits expected if the proposed system is implemented.

After problems or opportunities are identified, the analysts must determine the scale of response needed to meet the user's requests for a new system as well as the approximate amount of time and money that will be required in the effort.

The financial and economic **questions raised by analysts** during the preliminary investigation are for the purpose of estimating the following:

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

Financial Feasibility

The solution proposed may be prohibitively costly for the user organization.

For example, Monitoring the stock through VSAT network connecting multiple locations may be acceptable for an organization with high turnover. But this may not be a viable solution for smaller ones.

Technical Feasibility:

Essentially, an analyst ascertains whether the proposed system is feasible with existing or expected computer hardware and software technology.

The technical **issues usually raised** during the feasibility stage of investigation include the following:

Does	- the necessary technology exists to do what is suggested (and can it be acquired)? - the proposed equipment has the technical capacity to hold the data required to use the new system?
Can	- the proposed application be implemented with existing technology? - the system be expanded if developed?

- Will the proposed system provide adequate responses to inquiries, regardless of the number or location of users? - Are there technical guarantees of accuracy, reliability, ease of access, and data security?

PHASE-2 System Requirements Analysis -SRS

This phase includes a thorough and detailed understanding of the current system, identifies the areas that need modification to solve the problem, the determination of user/managerial requirements and to have fair idea about various systems development tools.

The following objectives are performed in this phase in order to generate the deliverable, Systems Requirements Specification (SRS):

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

In order to accomplish the aforementioned objectives, a series of steps are taken. Such steps result in process, assuring appropriate systems requirements analysis.

A generic set of process are described as follows:**(FASS-SRI)**

1.Fact Finding

Every system is built to meet some set of needs,

for example, the need of the organization for lower operational costs, better information for managers, smooth operations for users or better levels of services to customers.

To assess these needs, the analysts often interact extensively with people, who will be benefited from the system in order to determine 'what are their actual requirements'.

Various fact-finding techniques/tools are used by the system analyst for determining these needs/requirements are briefly discussed below: **(Q-ODI)**

Questionnaires: Users and managers are asked to complete questionnaire about the information systems when the traditional system development approach is chosen.

The main strength of questionnaires is that a large amount of data can be collected through a variety of users quickly.

Also, if the questionnaire is skillfully drafted, responses can be analyzed rapidly with the help of a computer.

Observation: In general, and particularly in prototyping approaches, observation plays a central role in requirement analysis.

Only by observing how users react to prototypes of a new system, the system can be successfully developed.

Documents: Document means manuals, input forms, output forms, diagrams of how the current system works, organization charts showing hierarchy of users and manager responsibilities, job descriptions for the people, who work with the current system, procedure manuals, program codes for the applications associated with the current system, etc.

Interviews: Users and managers may also be interviewed to extract information in depth.

The data gathered through interviews often provide system developers with a larger picture of the problems and opportunities.

Interviews also give analyst the opportunity to observe and record first-hand user reaction and to probe for further information.

2. Analysis of the Present System:

Detailed investigation of the present system involves collecting, organizing and evaluating facts about the system and the environment in which it operates.

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

The following areas should be studied in depth: **(RR-RAMU-RA)**

Reviewing Historical Aspects: A brief history of the organization is a logical starting point for an analysis of the present system.

The historical facts enable to identify the major turning points and milestones that have influenced its growth.

A review of annual reports and organization charts can identify the growth of management levels as well as the development of various functional areas and departments.

Reviewing Data Files: The analyst should investigate the data files maintained by each department, noting their number and size, where they are located, who uses them and the number of times per given time interval, these are used.

The system analyst should also review all on-line and off-line files, which are maintained in the organization as it will reveal information about data that are not contained in any outputs.

Reviewing Methods, Procedures and Data Communications: Methods and procedures transform input data into useful output.

A method is defined as a way of doing something; a procedure is a series of logical steps by which a job is accomplished.

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.

S/he must review the types of data communication equipment including data interface, data links, modems, dial-up and leased lines and multiplexers.

The system analyst must understand how the data- communications network is used in the present system so as to identify the need to revamp the network when the new system is installed.

Analyzing Inputs: A detailed analysis of present inputs is important since they are basic to the manipulation of data. Source documents are used to capture the originating data for any type of system.

The system analyst should be aware of various sources from where the data are initially captured, keeping in view the fact that outputs for one area may serve as an input for another area.

The system analyst must understand the nature of each form, ‘what is contained in it’, ‘who prepared it’, ‘from where the form is initiated’, ‘where it is completed’, the distribution of the form and other similar considerations.

Modeling the Existing System: As the logic of inputs, methods, procedures, data files, data communications, reports, internal controls and other important items are reviewed and analyzed in a top down manner; the processes must be properly documented.

The flow charting and diagramming of present information not only organizes the facts, but also helps to disclose gaps and duplication in the data gathered.

It allows a thorough comprehension of the numerous details and related problems in the present operation.

Undertaking Overall Analysis of the Existing system: Based upon the aforesaid investigation of the present information system, the final phase of the detailed investigation includes the analysis of the present work volume; the current personnel requirements;

the present costs-benefits of each of these must be investigated thoroughly.

Reviewing Internal Controls: A detailed investigation of the present information system is not complete until internal control mechanism is reviewed.

Locating the control points helps the analyst to visualize the essential parts and framework of a system.

An examination of the present system of internal controls may indicate weaknesses that should be removed in the new system.

Analyzing Outputs: The outputs or reports should be scrutinized carefully by the system analysts in order to determine 'how well they will meet the organization's needs.

The analysts must understand what information is needed and why, who needs it and when and where it is needed.

Often, many reports are a carry-over from earlier days and have little relevance to current operations. Attempts should be made to eliminate all such reports in the new system.

3. System Analysis of Proposed Systems

The required systems specifications should be in conformity with the project's objectives articulated and in accordance with the following: **(MODI-W)**

-**M**ethods and procedures that show the relationship of inputs and outputs to the database, utilize data communications as, when and where deemed appropriate.

-**O**utputs are produced with great emphasis on timely managerial reports that utilize the management by exception' principle.

-**D**atabases are maintained with great accent on online processing capabilities.

-**I**nput data is prepared directly from original source documents for processing by the computer system.

-**W**ork volumes and timings are carefully considered for present and future periods including peak periods.

4. System Development Tools

Major tools	Prominent tools	
System Components and Flows User Interface Data Attributes and Relationships Detailed System Processes	-Structured English -Flowcharts -Data Flow Diagrams -Decision Tree -Decision Table	-CASE Tools - System Components Matrix -Data Dictionary - User Interface Layout and Forms

System Components and Flows: These tools help the system analysts to document the data flow among the major resources and activities of an information system.

System flow charts are typically used to show the flow of data media as they are processed by the hardware devices and manual activities.

A data flow diagram uses a few simple symbols to illustrate the flow of data among external entities (such as people or organizations etc.), processing activities and data storage elements.

User Interface: Designing the interface between end users and the computer system is a major consideration of a system analyst while designing the new system.

Layout forms and screens are used to construct the formats and contents of input/output media and methods.

Dialogue flow diagrams analyze the flow of dialogue between computers and people.

Data Attributes and Relationships: The data resources in information system are defined, catalogued and designed by this category of tools.

A Data Dictionary catalogs the description of the attributes (characteristics) of all data elements and their relationships to each other as well as to external systems.

Entity-relationship diagrams are used to document the number and type of relationship among the entities in a system.

Detailed System Processes: These tools are used to help the programmer to develop detailed procedures and processes required in the design of a computer program.

Decision trees and decision tables use a network or tabular form to document the complex conditional logic involved in choosing among the information processing alternatives in a system.

Structured English: Structured English, also known as Program Design Language (PDL), is the use of the English language with the syntax of structured programming.

Thus, Structured English aims at getting the benefits of both the programming logic and natural language.

A better structured, universal and precise tool is referred to as pseudo code.

Flowcharts: Flowcharting is a pictorial representation technique that can be used by analysts to represent the inputs, outputs and processes of a business process.

It is a common type of chart that represents an algorithm or process showing the steps as boxes of various kinds, and their order by connecting these with arrows.

Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields.

Data Flow Diagrams: A Data Flow Diagram uses few simple symbols to illustrate the flow of data among external entities (such as people or organizations, etc.), processing activities and data storage elements.

A DFD is composed of four basic elements: Data Sources and Destinations, Data Flows, Transformation processes, and Data stores.

Decision Tree: A Decision Tree is a support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility.

Decision tree is commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal and to calculate conditional probabilities.

5. Systems Specification

A well-documented SRS may normally contain the following sections:

Introduction: Goals, Objectives, software context, Scope and Environment of the computer-based system.

Information Description: Problem description;
Information content, flow and structure;
Hardware, software, human interfaces for external system elements and internal software functions.

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

Behavioral Description: Response to external events and internal controls.

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

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

SRS Review: The development team makes a presentation and then hands over the SRS document to be reviewed by the user.

The review reflects the development team's understanding of the existing processes. Only, after ensuring that the document represents existing processes accurately, the user should sign the document.

6. Roles Involved in SDLC

Steering Committee: It is a special high power committee of experts to accord approvals for go-ahead and implementations.

Some of the functions of Steering Committee are given as follows:

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

Systems Analyst / Business Analyst: The systems analysts' main responsibility is to conduct interviews with users and understand their requirements.

he is a link between the users and the designers/programmers, who convert the users' requirements in the system requirements and plays a pivotal role in the Requirements analysis and Design phase.

Programmer/Developers: Programmers is a mason of the software industry, who converts design into programs by coding using programming language.

Apart from developing the application in a programming language, they also test the program for debugging activity to assure correctness and reliability

project manager: A project manager is normally responsible for more than one project and liaisonsing with the client or the affected functions.

he is responsible for delivery of the project deliverables within the time/budget and periodically reviews the progress of the project with the project leader and his team.

Phase-3 System Designing

The design phase activities includes
Architectural Design;
Physical Design;
Design of the Data Flow;
Design of the Database;
Design of the User-interface;
system software platform',

which are described briefly as follows:

Architectural design: deals with the organization of applications in terms of hierarchy of modules and sub-modules.

At this stage, we identify
major modules;
functions and scope of each module;
interface features of each module;
modules that each module can call directly or indirectly and Data received from / sent to / modified in other modules.

The architectural design is made with the help of a tool called Functional Decomposition, which can be used to represent hierarchies.

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.

physical design: the logical design is transformed into units, which in turn can be decomposed further into implementation units such as programs and modules.

During physical design, the primary concern of the auditor is effectiveness and efficiency issues.

The auditor should seek evidence that designers follow some type of structured approach like CASE tools to access their relative performance via simulations when they undertake physical design.

Some of the issues addressed here are

type of hardware for client application and server application,
Operating systems to be used,
type of networking,
processing – batch – online, real – time;
frequency of input, output; and
month-end cycles / periodical processing.

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

The design should	- be based on the analysis. - should follow standards laid down. - be modular, with high cohesion and low coupling.
- There is a tendency to develop merely one design and consider it the final product. -the recommended procedure is to design two or three alternatives and choose the best one on pre-specified criteria. - The software functions designed should be directly relevant to business activities.	

Modularity is measured by two parameters. Cohesion and Coupling.

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

Coupling is a measure of the interconnection between modules.

Design of Data flow: The design of the data flow is a major step in the conceptual design of the new system.

In designing the data flow for the proposed system, the inputs that are required are - existing data 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).

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

Conceptual Modeling	These describe the application domain via entities/objects, attributes of these entities/objects and static and dynamic constraints on these entities/objects, their attributes, and their relationships.
Data Modeling	Conceptual Models need to be translated into data models so that they can be accessed and manipulated by both high- level and low-level programming languages.
Storage Structure Design	Decisions must be made on how to linearize and partition the data structure so that it can be stored on some device.
Physical Layout Design	Decisions must be made on how to distribute the storage structure across specific storage media and locations.

User Interface Design: It involves determining the ways in which users will interact with a system.

The points that need to be considered while designing the user interface are - source documents to capture raw data, hard-copy output reports, screen layouts for dedicated source-document input, inquiry screens for database interrogation, graphic and color displays, and requirements for special input/output device.

System's Operating Platform: In some cases, the new system requires an operating platform including hardware, network and system software not currently available in an organization.

Auditors should be concerned about the extent to which modularity and generality are preserved in the design of the hardware/system software platform.

Phase-4 System Acquisition

Acquisition Standards: Management should establish acquisition standards that address the security and reliability issues as per current state-of-the art development standards.

Acquisition standards should focus on the following:

-Ensuring security, reliability, and functionality already built into a product;

-Ensuring manager's complete appropriate vendor, contract, and
licensing reviews and acquiring products compatible with existing systems;

-Establishing acquisition standards to ensure functional, security, and operational requirements to be accurately identified and clearly detailed in request -for-proposals.

-Invitations-to-tender soliciting bids from vendors when acquiring hardware or integrated systems of hardware and software;

-Request-for-proposals soliciting bids when acquiring off-the-shelf or third-party developed software;

Other Acquisition Aspects and Practices

(i) Hardware Acquisition: In case of procuring such machinery as machine tools, transportation equipment, air conditioning equipment, etc.,

the management can normally rely on the time tested selection techniques and the objective selection criteria can be delegated to the technical specialist.

The management depends upon the vendor for support services, systems design, education and training etc., and expansion of computer installation for almost an indefinite period; therefore, this is not just buying the machine and paying the vendor for it but it amounts to an enduring alliance with the supplier.

(ii) Software Acquisition: Once user output and input designs are finalized, the nature of the application software requirements must be assessed by the systems analyst.

This determination helps the systems development team to decide 'what type of application software products is needed' and consequently, the degree of processing that the system needs to handle.

This helps the system developers in deciding about the nature of the systems software and computer hardware that will be most suitable for generating the desired outputs, and also the functions and capabilities that the application software must possess.

At this stage, the system developers must determine whether the application software should be created in-house or acquired from a vendor.

Phase-5 System Development

A good coded application and programs should have the following characteristics: (U-R-REAR)

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

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

Reliability: It refers to the consistency with which a program operates over a period of time.

However, poor setting of parameters and hard coding of some data, subsequently could result in the failure of a program after some time.

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

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

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

Phase-6 System Testing

Different levels/facets of Testing are described as follows.

1.Unit Testing	2.Integration Testing	3.Regression Testing	4.System Testing	5.Final Acceptance Testing

1.Unit Testing

In computer programming, unit testing is a software verification and validation method in which a programmer tests if individual units of source code are fit for use. A unit is the smallest testable part of an application, which may be an individual program, function, procedure, etc

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

Functional Tests: It check

'whether programs do, what they are supposed to do or not'.

The test plan specifies operating conditions, input values, and expected results, and as per this plan, programmer checks by inputting the values to see whether the actual result and expected result match.

Stress Tests: Stress testing is a form of testing that is used to determine the stability of a given system or entity. It involves testing beyond normal operational capacity, often to a breaking point, in order to observe the results.

These tests are designed to overload a program in various ways.

The purpose of a stress test is to determine the limitations of the program.

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

Structural Tests: Structural Tests are concerned with examining the internal processing logic of a software system.

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

Unit Testing is classified as Static Testing and Dynamic Analysis Testing.

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

ii. Dynamic Analysis Testing: Such testing is normally conducted through execution of programs in operating conditions. Typical techniques for dynamic testing and analysis include the following:

Black Box Testing: Black Box Testing takes an external perspective of the test object, to derive test cases. These tests can be functional or non-functional, though usually functional.

The test designer selects typical inputs including simple, extreme, valid and invalid input-cases and executes to uncover errors.

There is no knowledge of the test object's internal structure.

This method of test design is applicable to all levels of software testing i.e. unit, integration, functional testing, system and acceptance.

The higher the level, hence the bigger and more complex the box, the more one is forced to use black box testing to simplify.

If a module performs a function, which is not supposed to, the black box test does not identify it.

White Box Testing: It uses an internal perspective of the system to design test cases based on internal structure. It requires programming skills to identify all paths through the software.

The tester chooses test case inputs to exercise paths through the code and determines the appropriate outputs.

Since the tests are based on the actual implementation, if the implementation changes, the tests probably will need to change, too.

It is applicable at the unit, integration and system levels of the testing process, it is typically applied to the unit.

While it normally tests paths within a unit, it can also test paths between units during integration, and between subsystems during a system level test.

After obtaining a clear picture of the internal workings of a product, tests can be conducted to ensure that the internal operation of the product conforms to specifications and all the internal components are adequately exercised.

Gray Box Testing: It is a software testing technique that uses a combination of black box testing and white box testing.

In gray box testing, the tester applies a limited number of test cases to the internal workings of the software under test.

In the remaining part of the gray box testing, one takes a black box approach in applying inputs to the software under test and observing the outputs.

2. Integration Testing:

It occurs after unit testing and before system testing with an objective to evaluate the validity of connection of two or more components that pass information from one area to another.

Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

This is carried out in the following two manners:

Bottom-up Integration: It is the traditional strategy used to integrate the components of a software system into a functioning whole. It consists of unit testing, followed by sub - system testing, and then testing of the entire system. Bottom-up testing is easy to implement as at the time of module testing, tested subordinate modules are available.	**Top-down Integration:** It starts with the main routine, and stubs are substituted, for the modules directly subordinate to the main module. An incomplete portion of a program code that is put under a function in order to allow the function and the program to be compiled and tested, is referred to as a stub. A stub does not go into the details of implementing details of the function or the program being executed. Once the main module testing is complete, stubs are substituted with real modules one by one, and these modules are tested with stubs. This process continues till the atomic modules are reached.

The disadvantage, however is that testing of major decision / control points is deferred to a later period.	The difficulty arises in the top-down method, because the high-level modules are tested, not with real outputs from subordinate modules, but from stubs.

3. Regression Testing:

Each time a new module is added or any modification made in the software, it changes.

New data flow paths are established, new I/O may occur and new control logic is invoked.

These changes may cause problems with functions that previously worked flawlessly.

the regression tests ensure that changes or corrections have not introduced new faults.

The data used for the regression tests should be the same as the data used in the original test.

4.System Testing:

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

System testing begins either when the software as a whole is operational or when the well-defined subsets of the software's functionality have been implemented.

The purpose of system testing is to ensure that the new or modified system functions properly.

These test procedures are often performed in a non-production test environment.

The types of testing that might be carried out are as follows:

Security Testing: This is the process to determine that an Information System protects data and maintains functionality as intended or not.

The six basic security concepts that need to be covered by security testing are – confidentiality, integrity, availability authentication, authorization, and non-repudiation.

This testing technique also ensures the existence and proper execution of access controls in the new system.

Stress Testing: It is a form of testing that is used to determine the stability of a given system or entity.

It involves testing beyond normal operational capacity, often to a breaking point, in order to observe the results.

It may be performed by testing the application with large quantity of data during peak hours to test its performance.

Recovery Testing: This is the activity of testing 'how well the application is able to recover from crashes, hardware failures and other similar problems'.

It is the forced failure of the software in a variety of ways to verify that recovery is liable to be properly performed, in actual failures.

Performance Testing: In the computer industry, software performance testing is used to determine the speed or effectiveness of a computer, network, software program or device.

This testing technique compares the new system's performance with that of similar systems using well defined benchmarks.

5.Final Acceptance Testing

It is conducted when the system is just ready for implementation.

During this testing, it is ensured that the new system satisfies the quality standards adopted by the business and the system satisfies the users.

Thus, the final acceptance testing has two major parts:

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

ii. User Acceptance Testing: It ensures that the functional aspects expected by the users have been well addressed in the new system.

There are two types of the user acceptance testing described as follows:

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

Beta Testing: This is the second stage, generally performed after the deployment of the system.

It is performed by the external users, during the real life execution of the project.

It normally involves sending the product outside the development environment for real world exposure and receives feedback for analysis and modifications, if any.

Phase-7 System Implementation

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

(i)Equipment Installation	**(ii) Training Personnel**	**(iii) System Change-Over Strategies (SCOS)**	**(iv) technical activities**
-Site Preparation -Installation of New Hardware / Software -Equipment Checkout		-Direct Implementation/Abrupt Change-Over -Phased Changeover -Pilot Changeover -Parallel Changeover	-Procedure Conversion -File Conversion -System conversion -Scheduling Personnel and Equipment

(i) Equipment Installation:

The necessary hardware should be ordered in time to allow for installation and testing of equipment during the implementation phase.

An installation checklist should be developed at this time with operating advice from the vendor and system development team.

In those installations, where people are experienced in the installation of the same or similar equipment, adequate time should be scheduled to allow completion of the following activities:

Site Preparation: An appropriate location as prescribed and the typical equipment must be availed to provide an operating environment for the equipment that will meet the

vendor's temperature, humidity and dust control specifications etc.

Installation of New Hardware / Software: The equipment must be physically installed by the manufacturer, connected-to the power source and wired to communication lines, if required.

If the new system interfaces with the other systems or is distributed across multiple software platforms, some final commissioning tests of the operating environment may be desirable to prove end to end connectivity.

Equipment Checkout: The equipment must be turned on for testing under normal operating conditions.

Though the routine 'diagnostic tests' should be run by the vendor, the implementation in-house team should devise and run extensive tests of its own to ensure that equipment functionalities in actual working conditions.

(ii) Training Personnel: A system can succeed or fail depending on the way it is operated and used.

Therefore, the quality of training received by the personnel involved with the system in various capacities helps the successful implementation of information system.

Thus, training is a major component of systems implementation.

When a new system is acquired, which often involves new hardware and software, both users and computer professionals generally need some type of training.

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

(iii) System Change-Over Strategies: The Four types of popular implementation strategies are described as follows:

Direct Implementation / Abrupt Change-Over: This is achieved through an abrupt takeover – an all or no approach.

With this strategy, the changeover is done in one operation, completely replacing the old system in one go.

Direct Implementation, which usually takes place on a set date, often after a break in production or a holiday period so that time can be used to get the hardware and software for the new system installed without causing too much disruption.

Phased Changeover: With this strategy, implementation can be staged with conversion to the new system taking place gradually.

For example, some new files may be converted and used by employees whilst other files continue to be used on the old system i.e. the new is brought in stages (phases).

If a phase is successful then the next phase is started, eventually leading to the final phase when the new system fully replaces the old one.

Pilot Changeover: With this strategy, the new system replaces the old one in one operation but only on a small scale.

Any errors can be rectified or further beneficial changes can be introduced and replicated throughout the whole system in good time with the least disruption.

For example - it might be tried out in one branch of the company or in one location.

If successful, then the pilot is extended until it eventually replaces the old system completely.

Parallel Changeover: This is considered the most secure method with both systems running in parallel over an introductory period.

The old system remains fully operational while the new systems come online.

With this strategy, the old and the new system are both used alongside each other, both being able to operate independently.

If all goes well, the old system is stopped and new system carries on as the only system.

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

Procedure Conversion: Operating procedures should be carefully completed with sufficient-enough documentation for the new system.

It applies to both computer- operations and functional area operations.

Before any parallel or conversion activities can start, operating procedures must be clearly spelled out for personnel in the functional areas undergoing changes.

Information on input, data files, methods, procedures, output, and internal control must be presented in clear, concise and understandable terms for the average reader.

File Conversion: Because large files of information must be converted from one medium to another, this phase should be started long before programming and testing are completed.

The cost and related problems of file conversion are significant whether they involve on-line files (common database) or off-line files.

In order for the conversion to be as accurate as possible, file conversion programs must be thoroughly tested.

Adequate control, such as record counts and control totals, should be required output of the conversion program.

The existing computer files should be kept for a period of time until sufficient files are accumulated for back up.

System conversion: After on-line and off-line files have been converted and the reliability of the new system has been confirmed for a functional area, daily processing can be shifted from the existing information system to the new one.

All transactions initiated after this time are processed on the new system.

System development team members should be present to assist and to answer any questions that might develop.

Consideration should be given to operating the old system for some more time to permit checking, matching and balancing the total results of both systems.

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, 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.

Phase-8 Post Implementation Review and Systems Maintenance

(i) Post Implementation Review: A Post Implementation Review answers the question "Did we achieve what we set out to do in business terms?"

Some of the purposes served a Post Implementation Review ascertains the degree of success from the project, in particular, the extent to which it met its objectives, delivered planned levels of benefit, and addressed the specific requirements as originally defined.

A Post-Implementation Review should be scheduled some time after the solution has been deployed.

Typical periods range from **6 weeks to 6 months**, depending on the type of solution and its environment.

There are two basic dimensions of Information system that should be evaluated. The first dimension is concerned with whether the newly developed system is operating properly.

The other dimension is concerned with whether the user is satisfied with the information system with regard to the reports supplied by it.

Typical evaluations include the following:

Development Evaluation: Evaluation of the development process is primarily concerned with whether the system was developed on schedule and within budget.

It requires schedules and budgets to be established in advance and that record of actual performance and cost be maintained.

However, it may be noted that very few information systems have been developed on schedule and within budget.

In fact, many information systems are developed without clearly defined schedules or budgets.

Operational Evaluation: The evaluation of the information system's operation pertains to whether the hardware, software and personnel are capable to perform their duties.

It tries to answer the questions related to functional aspects of the system.

Such an evaluation is relatively straightforward if evaluation criteria are established in advance.

Information Evaluation: This aspect of system evaluation is difficult and it cannot be conducted in a quantitative manner, as is the case with development and operational evaluations.

The objective of an information system is to provide information to a considerable extent to support the organizational decision system.

Therefore, the extent to which information provided by the system is supportive to decision making is the area of concern in evaluating the system.

(ii) System Maintenance:

Maintenance can be categorized in the following ways:

Preventive Maintenance: Preventive maintenance concerns with the activities aimed at increasing the system's maintainability, such as updating documentation, adding comments, and improving the modular structure of the system.

The long-term effect of corrective, adaptive and perfective changes increases the system's complexity.

As a large program is continuously changed, its complexity, which reflects deteriorating structure, increases unless work is done to maintain or reduce it.

Corrective Maintenance: Corrective maintenance deals with fixing bugs in the code or defects found during the executions.

A defect can result from design errors, logic errors coding errors, data processing and system performance errors.

The need for corrective maintenance is usually initiated by bug reports drawn up by the end users

Scheduled Maintenance: Scheduled maintenance is anticipated and can be planned for operational continuity and avoidance of anticipated risks.

Rescue Maintenance: Rescue maintenance refers to previously undetected malfunctions that were not anticipated but require immediate troubleshooting solution.

A system that is properly developed and tested should have few occasions of rescue maintenance.

Adaptive Maintenance: Adaptive maintenance consists of adapting software to changes in the environment, such as the hardware or the operating system.

The term environment in this context refers to the totality of all conditions and influences, which act from outside upon the system.

The need for adaptive maintenance can only be recognized by monitoring the environment.

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

Division-3: Miscellaneous

Achieving System Development Objectives

An analysis on 'why organizations fail to achieve their systems development objectives' reveals bottlenecks. Some of the most notable ones are described briefly as follows:

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

Shifting User Needs: User requirements for IT are constantly changing.

As these changes accelerate, there will be more requests for Information systems development and more development projects.

When these changes occur during a development process, the development team faces the challenge of developing systems whose very purpose might change since the development process began.

Resistance to Change: People have a natural tendency to resist change, and information systems development projects signal changes - often radical - in the workplace.

When personnel perceive that the project will result in personnel cutbacks, threatened personnel will dig in their heels, and the development project is doomed to failure.

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

Lack of Users' Participation: Users must participate in the development efforts to define their requirements, feel ownership for project success, and work to resolve development problems.

User participation also helps to reduce user resistance to change.

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

Overworked or Under-Trained Development Staff: In many cases, system developers often lack sufficient educational background and requisite state of the art skills.

Furthermore, many companies do a little to help their development personnel stay technically sound, and more so a training plan and training budget do not exist.

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

(iii) Management Related Issues: It refers to the bottlenecks with regard to organizational set up, administrative and overall management to accomplish the system development goals.

Some of such bottlenecks are mentioned as follows:

Lack of Senior Management Support and Involvement: Developers and users of information systems watch senior management to determine 'which systems development projects are important' and act accordingly by shifting their efforts away from any project, which is not receiving management attention.

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

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

Accountants' Involvement in Development Work

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

(i) Return on Investment (referred as ROI): This defines the return, an entity shall earn on a particular investment i.e. capital expenditure. This financial data is a prime consideration for any capital expenditure entity decides to incur.

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

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

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

Intangible Costs: Intangible Cost that cannot be easily measured.

(b) Benefits: The benefits, which result from developing new or improved information systems that can be subdivided into tangible and intangible benefits.

A post implementation analysis is also done to see how the system development effort has benefitted an organization.

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

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