Chapter - 5

Acquisition, Development and Implementation of Information Systems

Business Process Design: Business Process Design means structuring or restructuring the tasks, functionalities and activities for improvising a business system. Business process design involves a sequence of steps, which are:

1. *Present Process Documentation:* In this step, the present business process is analyzed and documented. The key deliverable of this step includes the well-defined short-comings of the present processes and the overall business requirements.
2. *Proposed Process Documentation*: This step is to design the new process requirements for the system. The design is based on the new system requirements and the changes proposed.
3. *Implementation of New Process:* This step is to implement largely the new as well as modified processes at the entity.

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

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

b. **System Design**, which is the process of planning and structuring a new business system or one to replace or complement an existing system.

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

1. **User Related Issues:** It refers to those issues where user/customer is reckoned as the primary agent. Some of the aspects with regard to this problem are:
   a. *Shifting User Needs,*
   b. *Resistance to Change,*
   c. *Lack of User Participation,*
   d. *Inadequate Testing and*
   e. *User Training.*
2. **Developer Related Issues**: It refers to the issues and challenges with regard to the developers. Some of the critical bottlenecks are:
   a. *Lack of Standard Project Management and*
   b. *System Development Methodologies,*
   c. *Overworked or Under-Trained Development Staff.*
3. **Management Related Issues:** It refers to the bottlenecks with regard to organizational set up, administrative and overall management to accomplish the system development goals. Some of such bottlenecks are:
   a. *Lack of Senior Management Support and Involvement,*
   b. *Non-development of Strategic Systems*
4. **New Technologies:** When an organization tries to create a competitive advantage by applying advance technologies, it generally finds that attaining system development objectives is more difficult because personnel are not 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:

1. **Return on Investment (referred as RoI):** This defines the return, an entity shall earn on a particular investment i.e. capital expenditure. For this analysis, following data needs to be generated.
   a) **Cost:** This includes estimates for typical costs involved in the development, which are

a. *Development Costs,*
b. *Operating Costs, and*
c. *Intangible Costs*.

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

2. **Computing Cost of IT Implementation and Cost Benefit Analysis:** For analysis of RoI, accountants need the costs and returns from the system development efforts.
3. **Skills expected from an Accountant:** An accountant, being an expert in accounting field must possess skills to understand the system development efforts and nuances of the same.

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

1. Waterfall model
2. Prototyping
3. Incremental model
4. Spiral model
5. Rapid Application Development (RAD) model
6. Agile model

Waterfall Model: This is a traditional development approach in which each phase is carried out in sequence or linear fashion. These phases include

1) Requirements Analysis,
2) Specifications and Design Requirements,
3) Coding, Final Testing, and
4) Release.

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

a) Project is divided into sequential phases, with some overlap and splash back acceptable between phases.
b) Emphasis is on planning, time schedules, target dates, budgets and implementation of an entire system at one time.

(a) Strengths: Major strengths are given as follows:

1) It is ideal for supporting less experienced project teams and project managers or project teams, 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.

(b) Weaknesses:

I. It is criticized to be inflexible, slow, costly, and cumbersome due to significant structure and tight controls.
II. Project progresses forward, with only slight movement backward.
III. There is a little to iterate, which may be essential in situations.
IV. Problems are often not discovered until system testing..
V. Written specifications are often difficult for users to read and thoroughly appreciate.
VI. It promotes the gap between users and developers with clear vision of responsibility.

The Prototyping Model: The goal of prototyping approach is to develop a small or pilot version called a Prototype of a

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 modifying/replicating/expanding or even replacing it by a full-scale and fully operational system.

Prototyping can be viewed as a series of four steps:

1. Identify Information System Requirements
2. Develop the Initial Prototype
3. Test and Revise
4. Obtain User Signoff of the Approved Prototype

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

a) Potential exists for exploiting knowledge gained in an early iteration as later iterations are developed.
b) It helps to easily identify, confusing or difficult functions and missing functionality.
c) It enables to generate specifications for a production application.
d) It encourages innovation and flexible designs.
e) It provides for quick implementation of an incomplete

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

a) Approval process and control are not strict.
b) Requirements may frequently change significantly.
c) Identification of non-functional elements is difficult to document.
d) Prototype may not have sufficient checks and balances incorporated.

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

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

I. Potential exists for exploiting knowledge gained in an early increment as later increments are developed.
II. Stakeholders can be given concrete evidence of project status throughout the life cycle.
III. It is more flexible and less costly to change scope and requirements.
IV. It helps to mitigate integration and architectural risks earlier in the project

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

1. Each phase of an iteration is rigid and do not overlap each other.
2. It is difficult to demonstrate early success to management.
3. Since some modules will be completed much earlier than others, well-defined interfaces are required.

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.

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

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.

Weaknesses: Some of the weaknesses identified by the experts and practitioners include the following:
It may prove highly customized to each project, and thus is quite complex and limits reusability.

a) A skilled and experienced project manager is required to determine how to apply it to any given project.
b) No established controls exist for moving from one cycle to another cycle
c) There are no firm deadlines

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

Key features include the following:

1. Key objective is fast development and delivery of a high quality system at a relatively low investment cost,
2. Aims to produce high quality systems quickly, primarily through the use of iterative Prototyping.
3. Active user involvement is imperative.
4. Iteratively produces production software, as opposed to a throwaway prototype.
5. Produces documentation necessary to facilitate future development and maintenance.
6. Standard systems analysis and design techniques can be fitted into this framework.

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

1) It concentrates on essential system elements from user viewpoint.
2) It provides for the ability to rapidly change system design as demanded by users.
3) It leads to a tighter fit between user requirements and system specifications.

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

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

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

Agile Manifesto is based on following features:

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

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

1. 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.
2. Face to face communication and continuous inputs from customer representative leaves a little space for guesswork.
3. The documentation is crisp and to the point to save time.

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

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

1) Agile lacks the attention to outside integration.
2) There is lack of emphasis on necessary designing and documentation.
3) Agile requires more re-work
4) The project can easily get taken off track if the customer representative is not clear about the final outcome.

System Development Life Cycle (SDLC): SDLC provides system designers and developers to follow a sequence of activities. It consists of a generic sequence of steps or phases in which each phase of the SDLC uses the results of the previous one. **These phases are given as under:**

S.No.	Phase Name	Nature of Activity
1.	Preliminary Investigation	Determining and evaluating the strategic feasibility of the system and ensure that the solution fits the business strategy.
2.	Systems Requirements Analysis	Analyzing the typical system requirements, in view of its functionalities, deliverables etc.
3.	Systems Design	Designing the system in terms of user interface, data storage and data processing functions on the basis of the requirement phase by developing the system flowcharts, system and data flow diagrams, screens and reports.
4.	Systems Acquisition	This involves acquisition of operating infrastructure including hardware, software and services.
5.	Systems Development	Developing the system as per the system designed to fulfill of requirements to the satisfaction of all stakeholders.
6.	Systems Testing	Requisite testing to ensure valid and reliable implementation.
7.	Systems Implementation	Operationalization of the developed system for acceptance by management and user before migration of the system to live environment and data conversion from legacy system to the new system.
8.	Post Implementation Review and Maintenance	Continuous evaluation of the system as it functions in the live environment and its pupation / maintenance.

The advantages of this system are given as follows:

1. Better planning and control by project managers;
2. Compliance to prescribed standards ensuring better quality;
3. Documentation that SDLC stresses on is an important measure of communication and control.

Some of the shortcomings and anticipated risks associated with the SDLC are as follows:

1. The development team may find it cumbersome.
2. The users may find that the end product is not visible for a long time.

3. The rigidity of the approach may prolong the duration of many projects.
4. It may not be suitable for small and medium sized projects.

Preliminary Investigation: It is predominantly aimed to determine and analyze the strategic benefits in implementing the system through

1. Evaluation and quantification of -productivity gains;
2. Future cost avoidance;
3. Cost savings, and
4. Intangible benefits like improvement in morale of employees

The deliverable of the preliminary investigation includes a report including feasibility study observations. **Major activities are:**

(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.

(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.

(iii) Delineation of Scope: The scope of a solution defines its typical boundaries. It should be clear and comprehensible to the user management stating the extent and 'what will be addressed by the solution and what will not'. The typical scope determination may be performed on the following dimensions:

A. *Data to be Processed,*
B. *Control Requirements,*
C. *Performance Requirements,*
D. *Constraints, Interfaces, and*
E. *Reliability requirements.*
F. *Functionality Requirements*

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

A. *Reviewing Internal Documents and*
B. *Conducting Interviews*.

(iv) Feasibility Study: After possible solution options are identified, project feasibility i.e. the likelihood that these systems can be implemented in the organization is determined. The Feasibility Study of a system is evaluated under following dimensions:

i. *Technical,*
ii. *Financial,*
iii. *Economic,*
iv. *Schedule/Time,*
v. *Resources,*
vi. *Operational,*
vii. *Behavioral and*
viii. *Legal*

(v) Reporting Results to Management: After the analyst articulates the problem, defines it along with its scope, s/he provides one or more solution alternatives, estimates the cost and benefits of each alternative and reports these results to management.

System Requirements Analysis: This phase includes a thorough and detailed understanding of the current system, identification of the areas that need modification to solve the problem, determination of user/managerial requirements and to have fair idea about various systems development tools. Major deliverable is *Systems Requirements Specification (SRS).*

A generic set of process is described as follows:

(i) **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. Various fact-finding techniques/tools are:

a. *Documents,*
b. *Questionnaires,*
c. *Interviews and*
d. *Observation.*

(ii) **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. The following areas should be studied in depth:

1. *Reviewing Historical Aspects,*
2. *Analyzing Inputs,*
3. *Reviewing Data Files,*
4. *Reviewing Methods,*
5. *Procedures and Data Communications,*
6. *Analyzing Outputs,*
7. *Reviewing Internal Controls,*
8. *Modeling the Existing System,*
9. *Undertaking Overall Analysis of the Existing system*.

(iii) **System Analysis of Proposed Systems:** After a thorough analysis of each functional area of the present information system, the proposed system specifications must be clearly defined, which are determined from the desired objectives set forth at the first stage of the study.

(iv) **System Development Tools:** Many tools and techniques have been developed to improve current information systems and to develop new ones. Major tools used for system development specification or representations can be classified into four categories based on the systems features. These are:

i. *System Components and Flows,*
ii. *User Interface,*
iii. *Data Attributes and Relationships,*
iv. *Detailed System Processes*.

(v)
Some popular tools are: *Structured English, Flowcharts, Data Flow Diagrams, Decision Tree, Decision Table, CASE Tools, System Components Matrix, Data Dictionary, User Interface Layout and Forms*

(v) **Systems Specification:** At the end of the analysis phase, the systems analyst prepares a document called Systems Requirement Specifications (SRS).

Roles Involved in SDLC: A variety of tasks during the SDLC are performed by special teams/committees/individuals based on requisite expertise as well as skills. Some of the generic roles are: *Steering Committee,*

1. *Project Manager,*
2. *Project Leader, Systems Analyst / Business Analyst,*
3. *Module Leader/Team Leader,*
4. *Programmer/Developers,*
5. *Database Administrator,*
6. *Quality Assurance, Testers,*
7. *Domain Specialist,*
8. *IS Auditor*
9. *Steering Committee*

Systems Design: The key objective of this phase to design an Information System that best satisfies users/managerial requirements. Design phase documents/deliverables include a 'blueprint' for the design with the necessary specifications for hardware, software, people and data resources.

Major activities are as follows:

A. **Architectural Design:** Architectural design deals with the organization of applications in terms of hierarchy of modules and sub-modules.
B. **Design of Data/Information flow:** The design of the data and information flow is a major step in the conceptual design of the new system.
C. **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:

1. *Conceptual Modeling,*
2. *Data Modeling,*
3. *Storage Structure Design and*
4. *Physical Layout Design*

D. **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*.

E. **Physical Design:** For the physical design, the logical design is transformed into units, which in turn can be decomposed further into implementation units such as programs and modules. During physical design, the primary concern of the auditor is effectiveness and efficiency issues.

F. **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

G. **Internal Design Controls:** From internal control point of view, this phase is also an important phase as all internal controls are placed in system during this phase.

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

1. **Acquisition Standards:** Management should establish acquisition standards that address the security and reliability issues as per current state-of-the art development standards.
2. **Systems Components from Vendors:** At the end of the design phase, the organization gets a reasonable idea of the types of hardware, software and services; it needs for the system being developed. The following considerations are valid for both acquisition of hardware and software:
    a. *Vendor Selection,*
    b. *Geographical Location of Vendor,*
    c. *Presentation by Selected Vendors,*
    d. *Evaluation of Users Feedback.*
3. **Other Acquisition Aspects and Practices:** In addition to the above, there are several other acquisition aspects and practices also, which are:
    a. *Hardware Acquisition,*
    b. *Software Acquisition,*
    c. *Contracts,*
    d. *Software*
    e. *Licenses and Copyright Violations,*
    f. *Validation of Vendors'*
    g. *Proposals,*
    h. ***Methods of Validating the proposal like***
        i. *Checklists,*
        ii. *Point-Scoring Analysis,*
        iii. *Public Evaluation Reports,*
        iv. *Benchmarking Problems related to Vendor's Solutions,*
        v. *Testing Problems.*

System Development: This phase is supposed to convert the design specifications into a functional system under the planned operating system environment. Application programs are written, tested and documented, and system testing conducting. Finally, it results into a fully functional and documented system. A good coded application and programs should have the following **characteristics**:

1. *Reliability,*
2. *Robustness,*

3 . *Accuracy,*
4 . *Efficiency,*
5. Usability and
6. *Readability*

Other related aspects of this phase are as follows:

1. **Program Coding Standards:** The logic of the program outlined in the flowcharts is converted into program statements or instructions at this stage. For each language, there are specific rules concerning format and syntax.
2. **Programming Language:** Application programs are coded in the form of statements or instructions and the same is converted by the compiler to object code for the computer to understand and execute.
3. **Program Debugging:** Debugging refers to correcting programming language syntax and diagnostic errors so that the program compiles cleanly. A clean compilation means that the program can be successfully converted from the source code written by the programmer into machine language instructions. For example, bugs may also arise when the program processes data (e.g. invalid input) resulting in abrupt terminations or errors. These will not be identified when computing.
4. **Testing the Programs:** A careful and thorough testing of each program is imperative to successful installation of any system. The programmer should plan the testing to be performed, including testing of all possible exceptions.
5. **Program Documentation:** Writing of narrative procedures and instructions for people, who will use software, is done throughout the program life cycle. Managers and users should carefully review both internal and external documentation in order to ensure that the software and system behave as the documentation indicates.
6. **Program Maintenance:** The requirements of business data processing applications are subject to periodic change. This calls for modification of various programs. There are usually separate categories of programmers called maintenance programmers, who are entrusted with this task.

System Testing: Testing is a process used to identify the correctness, completeness and quality of developed computer software. Different levels/facets of Testing are given as under:

7.

1. **Unit Testing:** Unit testing is a software verification and validation method in which a programmer tests if individual units of source code are fit for use. A unit is the smallest testable part of an application, which may be an individual program, function, procedure, etc. or may belong to a base/super class, abstract class or derived/child class. There are **five categories of tests** that a programmer typically performs on a program unit, which are:
    a) **Functional Tests:** Functional Tests check 'whether programs do, what they are supposed to do or not'.
    b) **Performance Tests:** Performance Tests should be designed to verify the response time, the execution time, throughput, primary and secondary memory utilization traffic rates on data channel and communication links.
    c) **Stress Tests:** Stress testing is a form of testing that is used to determine the stability of a given system or entity. It involves testing beyond normal operational capacity, often to a breaking point, in order to observe the results.
    d) **Structural Tests:** Structural Tests are concerned with examining the internal processing logic of a software system. For example, if a function is responsible for tax calculation, verification of the logic is a structural test.
    e) **Parallel Tests:** In Parallel Tests, the same test data is used in the new and old system and the output results are then compared.

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

(a) **Static Testing:** Static Analysis Tests are conducted on source programs and do not normally require executions in operating conditions. Typical static analysis techniques include: Desk Check, Structured Walk Through and Code Inspection.

(b) **Dynamic Analysis Testing:** Such testing is normally conducted through execution of programs in operating conditions. Typical techniques for dynamic testing and analysis include**:**

1. **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 they are usually functional.
2. **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.
3. **Gray Box Testing:** It is a software testing technique that uses a combination of black box testing and white box testing.

2. **Integration Testing:** Integration testing is an activity of software testing in which individual software modules are combined and tested as a group. This is carried out in the following two manners:
   A. **Bottom-up Integration:** It is the traditional strategy used to integrate the components of a software system into a functioning whole. It consists of unit testing, followed by sub-system testing, and then testing of the entire system.
   B. **Top-down Integration:** It starts with the main routine, and stubs are substituted, for the modules directly subordinate to the main module.

Regression Testing: In the context of the integration testing, the regression tests ensure that changes or corrections have not introduced new faults. The data used for the regression tests should be the same as the data used in the original test.

3. **System Testing:** It is a process in which software and other system elements are tested as a whole. System testing begins either when the software as a whole is operational or when the well-defined subsets of the software's functionality have been implemented. The types of testing that might be carried out are:
   A. **Recovery Testing:** This is the activity of testing 'how well the application is able to recover from crashes, hardware failures and other similar problems'.
   B. **Security Testing:** This is the process to determine whether an Information System protects data and maintains functionality as intended or not.
   C. **Stress or Volume Testing:** Stress testing is a form of testing that is used to determine the stability of a given system or entity. It involves testing beyond normal operational capacity, often to a breaking point, in order to observe the results.
   D. **Performance Testing:** Performance testing is used to determine the speed or effectiveness of a computer, network, software program or device. This testing technique compares the new system's performance with that of similar systems using well defined benchmarks.

4. **Final Acceptance Testing:** 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 satisfies users. Thus, final acceptance testing has two major parts:
   1. **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.
   2. **User Acceptance Testing:** It ensures that functional aspects expected by users have been well addressed in the new system. There are two types of user acceptance testing: *Alpha Testing and Beta Testing*.

System Implementation: 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. Some of the generic activities involved in system implementation stage are:

3.

A. **Equipment Installation:** The hardware required to support the new system is selected prior to the implementation phase. Major tasks are: *Site Preparation, Installation of New Hardware / Software and Equipment Checkout*.

B. **Training Personnel:** 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.

C. **System Change-Over Strategies:** Conversion or changeover is the process of changing over or shifting from the old system (may be a manual system) to the new system. Four types of popular implementation strategies are:

   I. **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.

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

   III. **Pilot Changeover:** With this strategy, the new system replaces the old one in a single operation but only on a small scale e.g. in one division. Any errors can be rectified or further beneficial changes can be introduced and replicated throughout the whole system in good time with least disruption.

   IV. **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.

D. **System technical changeover or Conversion** activities includes
   a. *Procedure Conversion,*
   b. *File Conversion,*
   c. *System conversion and Scheduling*
   d. *Personnel and Equipment.*

Post Implementation Review: A Post Implementation Review answers the question "Did we achieve what we set out to do in business terms?" Typical evaluations include the following: *Development Evaluation, Operational Evaluation and Information Evaluation*.

System Maintenance: Maintaining the system is an important aspect of SDLC. Maintenance can be categorized in the following ways:

1. **Scheduled Maintenance:** Scheduled maintenance is anticipated and can be planned for insuring operational continuity and avoidance of anticipated risks.
2. **Rescue Maintenance:** Rescue maintenance refers to previously undetected malfunctions that were not anticipated but require immediate troubleshooting solution.
3. **Corrective Maintenance:** Corrective maintenance deals with fixing bugs in the code or defects found during execution.
4. **Adaptive Maintenance:** Adaptive maintenance consists of adapting software to changes in the environment, such as the hardware or the operating system.
5. **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.
6. **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.

Operation Manuals: It is typical user's guide, also commonly known as Operations Manual. Moreover, it may be a technical communication document intended to give assistance to people using a particular system.