System Development Life Cycle

Waterfall Model

Mr. GD Makkar
HOD, Dept. of CA & IT
SGRRITS
Dehradun

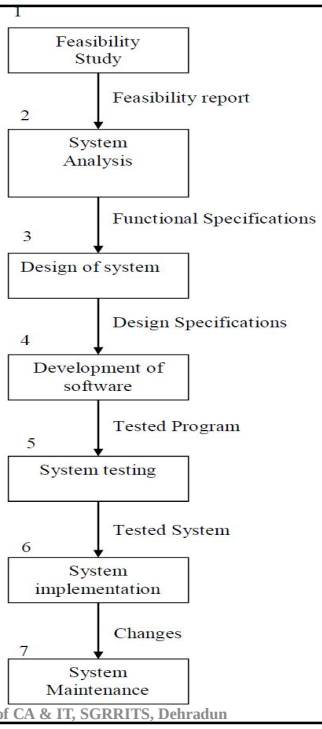
The need for a software life cycle model

- Without using of a particular life cycle model the development of a software product would not be in a systematic and disciplined manner.
- When a software product is being developed by a team there must be a clear understanding among team members about when and what to do. Otherwise it would lead to chaos and project failure.
- A software life cycle model defines entry and exit criteria for every phase.
- A phase can start only if its phase-entry criteria have been satisfied.
- Without software life cycle model the entry and exit criteria for a phase cannot be recognized.
- Without software life cycle models it is difficult for software project managers to monitor the progress of the project.

Different software life cycle models

- Classical Waterfall Model
- Iterative Waterfall Model
- Prototyping Model
- Evolutionary Model
- Spiral Model

Classical Waterfall Model



Shortcomings of the classical waterfall model

The classical waterfall model is an idealistic one since it assumes that no development error is ever committed by the engineers during any of the life cycle phases. However, in practical development environments, the engineers do commit a large number of errors in almost every phase of the life cycle. The source of the defects can be many:

- oversight,
- wrong assumptions,
- use of inappropriate technology,
- communication gap among the project engineers, etc.

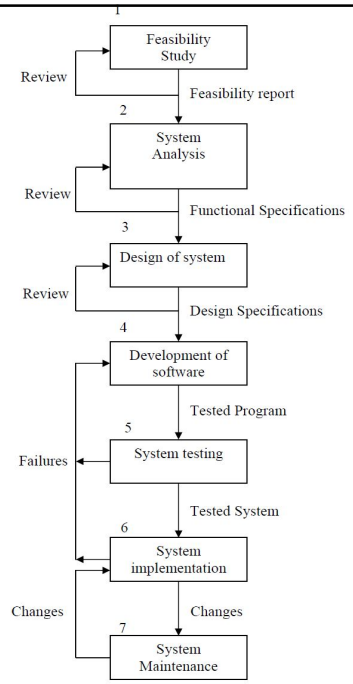
These defects usually get detected much later in the life cycle.

For example, a design defect might go unnoticed till we reach the coding or testing phase. Once a defect is detected, the engineers need to go back to the phase where the defect had occurred and redo some of the work done during that phase and the subsequent phases to correct the defect and its effect on the later phases. Therefore, in any practical software development work, it is not possible to strictly follow the classical waterfall model.

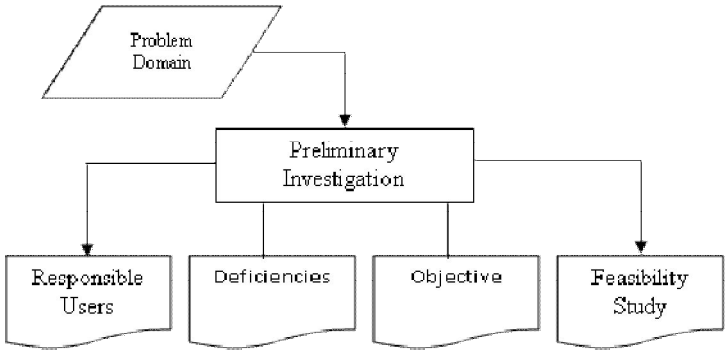
System Development Life Cycle (SDLC) is a sequence of events carried by analysts, designers and users to develop and implement an information system.

The phases of SDLC:

1. Preliminary Investigation
2. System Analysis
3. Design of the System
4. Development of Software
5. System Testing
6. System Implementation
7. System Maintenance.



Preliminary Investigation (Feasibility Study)



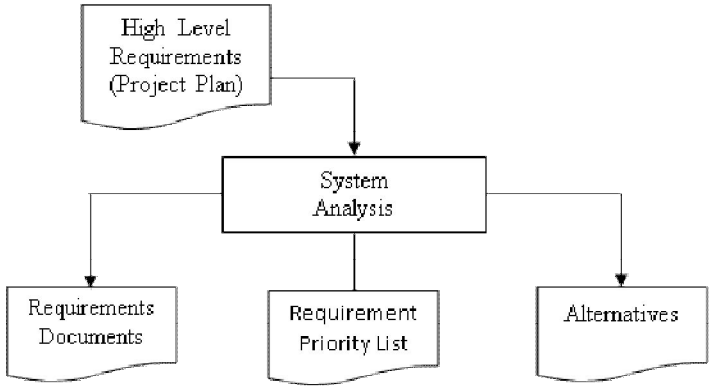
- Identify the responsible users and develop an initial scope of the system
- Identify current deficiencies in the system
- Determine objectives for the new system
- Determine whether it is feasible to automate the system
  - Technical feasibility
  - Economic Feasibility
  - Operational Feasibility

System Analysis – Determination of Requirements

The objective of the analysis phase is to determine system requirements, which involves studying the current business system in detail, to find out how it works and where improvements have to be made.

There are five activities are performed in the analysis phase:

- Gather information
- Defining system requirements
- Prioritize requirements
- Generate and evaluate alternatives
- Review analysis with users



Analysts work closely with the users of the system to get the answers of the following key questions: -

- What is being done by the current system?
- How it is being done?
- How big is the volume of transactions?
- How well is the task being performed?
- Does a problem exist?
- If a problem exists, how serious it is?
- If a problem exists, what is the underlying cause?

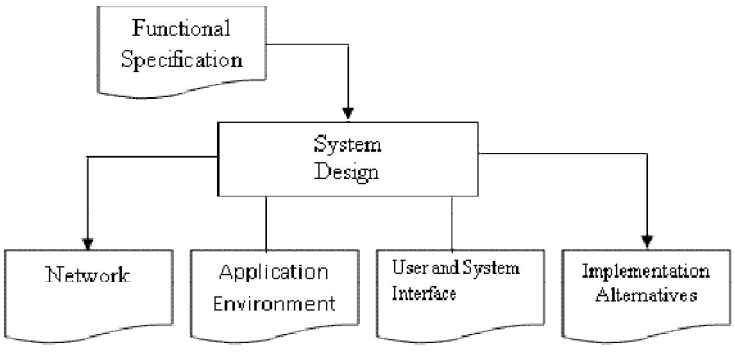
A detailed document is prepared by the system analyst containing the following: -

- Input that must be received by the system.
- The output to be produced by the system.
- The data to be retained.
- The procedure to get the output from the given input.
- Audit and control requirement - This specify the features, functions and procedures that are required for the user to monitor and ensure that the new system is working properly or not.
- System acceptance criteria – this would list the tests that the user would actually perform to check if the system is acceptable or not.

System analyst creates a detail document of analysis performed by him and this document is called **Functional Specification**.

Design of the system

Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudocode, and a complete entity-relationship diagram with a full data dictionary.

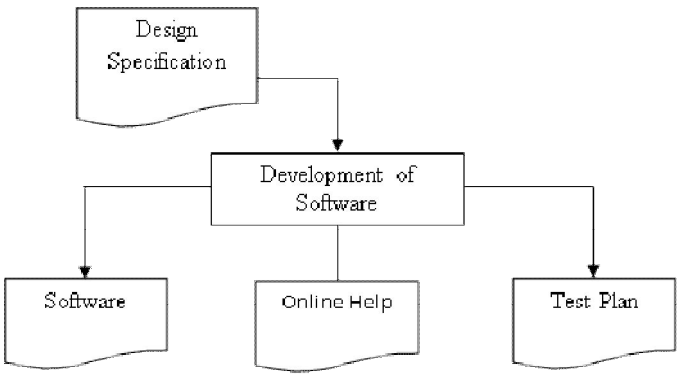


The design process should take care of the following:

- Design and integrate the network
- Design the application environment
- Design the user interfaces
- Design the system interfaces
- Design and integrate the database
- Prototype for design details
- Design and integrate the system control

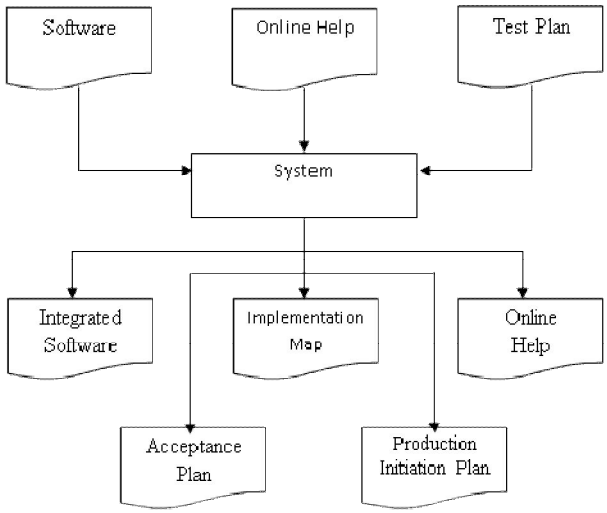
A document is produced at the end of this activity, called "**Design Specification**". This document should have charts, tables and special symbols to portray the design.

Development of Software



This activity of system development life cycle produces tested programs

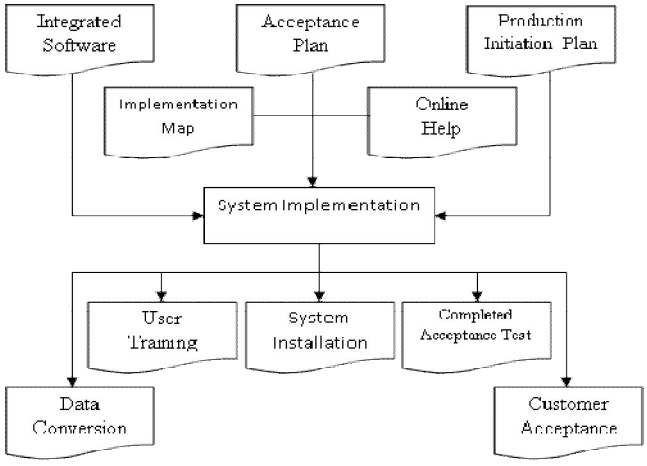
System Testing



System Implementation

Major activities of implementation phase are:

- Convert data
- Train users and document the system
- Install the system



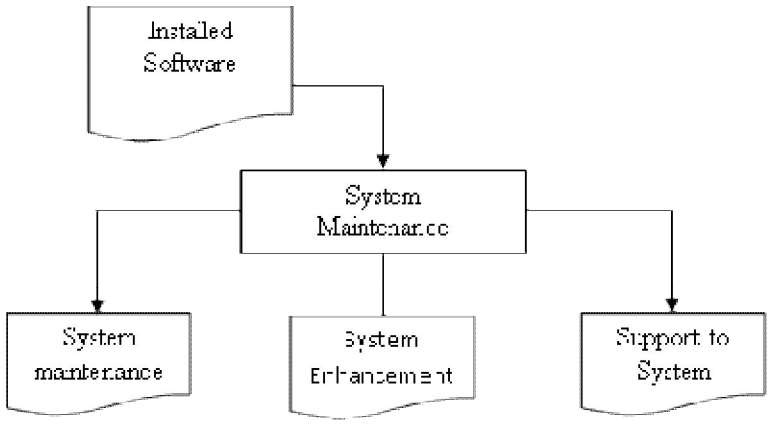
System Maintenance

Maintenance of a typical software product requires much more than the effort necessary to develop the product itself.

Many studies carried out in the past confirm this and indicate that the relative effort of development of a typical software product to its maintenance effort is roughly in the 40:60 ratio.

There are three major activities occur during this phase are:

- Maintain the system
- Enhance the system
- Support the system



Mr. GD Makkar, HOD, Dept. of CA & IT, SGRRITS, Dehradun

Types of System Maintenance

Correcting errors: Correcting errors that were not discovered during the product development phase. This is called corrective maintenance.

Perfective maintenance: Improving the implementation of the system, and enhancing the functionalities of the system according to the customer's requirements. This is called perfective maintenance.

Adaptive maintenance: Porting the software to work in a new environment. For example, porting may be required to get the software to work on a new computer platform or with a new operating system. This is called adaptive maintenance.

The classical waterfall model can be considered as the basic model. The classical waterfall model can not be used in practical development projects, since this model supports no mechanism to handle the errors committed during any of the phases.

This problem is overcome in the iterative waterfall model. The iterative waterfall model is probably the most widely used software development model evolved so far. This model is simple to understand and use. However, this model is suitable only for well-understood problems; it is not suitable for very large projects and for projects that are subject to many risks.