WHAT ARE SOFTWARE DEVELOPMENT TECHNIQUES?

SDT also referred to as a system development, is the process of creating and maintaining information systems. Some of the popular SDT methods are:

- Traditional software development
  - The waterfall model.
  - The V model.
  - Prototyping model
  - Incremental model
  - Spiral model
- Agile software development
  - Extreme Programming
  - Scrum
  - Feature-Driven Development

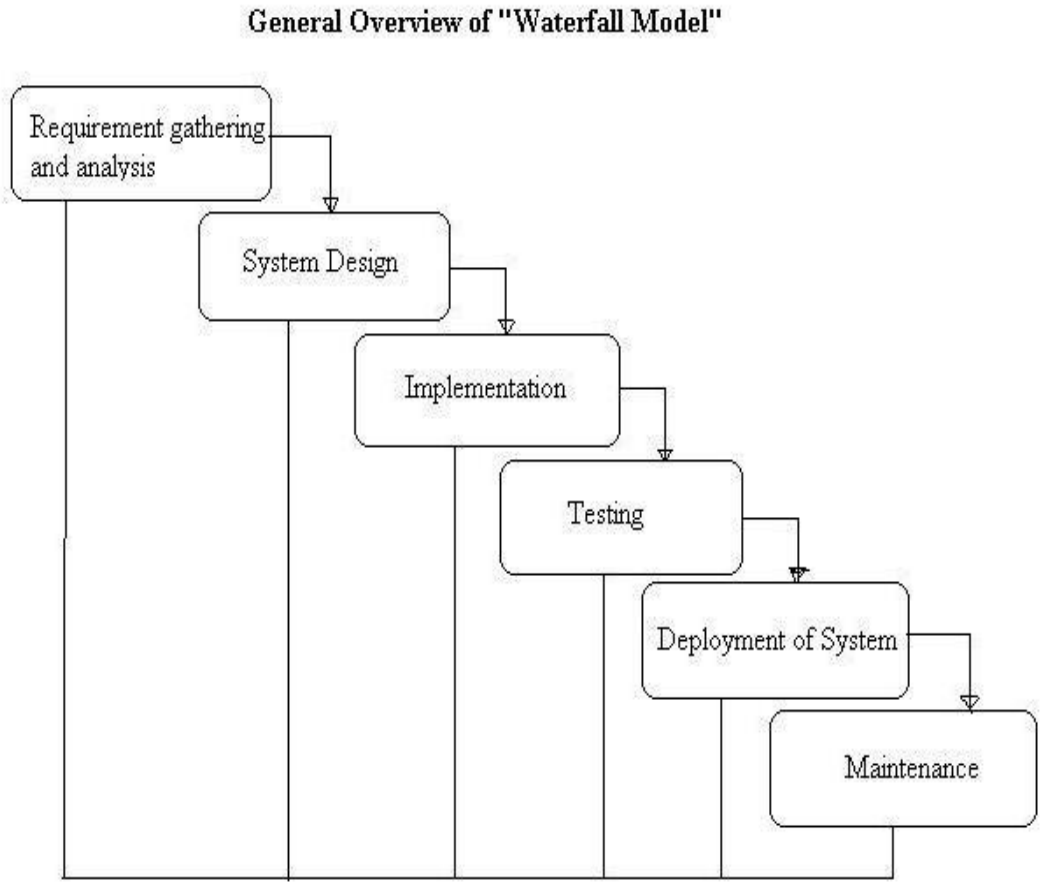
ADVANTAGES AND DISADVANTAGES OF SDT

- Advantages

  Building a software that specifically fits the needs of the customer.

- Disadvantages
  - Determining the requirements
  - Changing the requirements
  - Scheduling and budgeting difficulties
  - Changing technology

Waterfall Model



1) Requirement Analysis: All possible requirements of the system to be developed are captured in this phase. Requirements are set of functionalities and constraints that the end-user (who will be using the system) expects from the system. The requirements are gathered from the end-user by consultation, these requirements are analyzed for their validity and the possibility of incorporating the requirements in the system to be development is also studied. Finally, a Requirement Specification document is created which serves the purpose of guideline for the next phase of the model.

2) Software Design: Before a starting for actual coding, it is highly important to understand what we are going to create and what it should look like? The requirement specifications from first phase are studied in this phase and system design is prepared. Design step includes three major activates, data design, architecture design and procedural design.

3). Implementation & Unit Testing: On receiving system design documents, the work is divided in modules/units and actual coding is started. The system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality; this is referred to as Unit Testing. Unit testing mainly verifies if the modules/units meet their specifications.

4). Integration & System Testing: As specified above, the system is first divided in units which are developed and tested for their functionalities. These units are integrated into a complete system during Integration phase and tested to check if all modules/units work as required. After successfully testing the software, it is delivered to the customer.

5). Operations & Maintenance: This phase of "waterfall model" is virtually never ending phase (Very long). Generally, problems with the system developed (which are not found during the development life cycle) come up after its practical use starts, so the issues related to the system are solved after deployment of the system. Not all the problems come in picture directly but they arise time to time and needs to be solved; hence this process is referred as Maintenance.

The waterfall model can work reasonably well if all the following assumptions are satisfied:

➢The requirements are precisely known in advance.

➢The requirements include no unresolved high-risk items.

➢The requirements won't change much during development.

➢The team has previous experience with similar projects so that they know what's involved in building the application.

➢ There's enough time to do everything sequentially.

Advantages and Disadvantages

Advantages :

- Simple and easy to understand and use.
- Easy to manage due to the rigidity of the model each phase has well defined inputs and outputs.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.

Disadvantages :

- It is often difficult for the customer to state all requirements explicitly. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-throughout in the concept stage.
- The customer must have patience. No working software is produced until late during the life cycle.
- Not a good model for complex and object-oriented projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing.

Each of these problems is real. However, It provides a template into which methods for analysis, design, coding, testing, and support can be placed. The classic life cycle remains a widely used procedural model for software engineering. While it does have weaknesses, it is significantly better than a haphazard approach to software development.