**Practical - 1**

**Aim:** Study the complete Software Development Life Cycle (SDLC) and analyze various activities conducted as a part of various phases and explain any four SDLC models.

**What is SDLC?**

SDLC, Software Development Life Cycle is a process used by software industry to design, develop and test high quality software. The SDLC aims to produce high quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

The life cycle defines a methodology for improving the quality of software and the overall development process.

**Process Models:**

There are various software development life cycle models defined and designed which are followed during software development process. These models are referred as *Software Development Process Models.*

Each process model follows a series of steps unique to its type, in order to ensure success in process of software development.

Following is the list of various SDLC models:

1. Waterfall Model
2. Iterative Model
3. RAD Model
4. V-Model
5. Spiral Process Model
6. Component Based Process Model

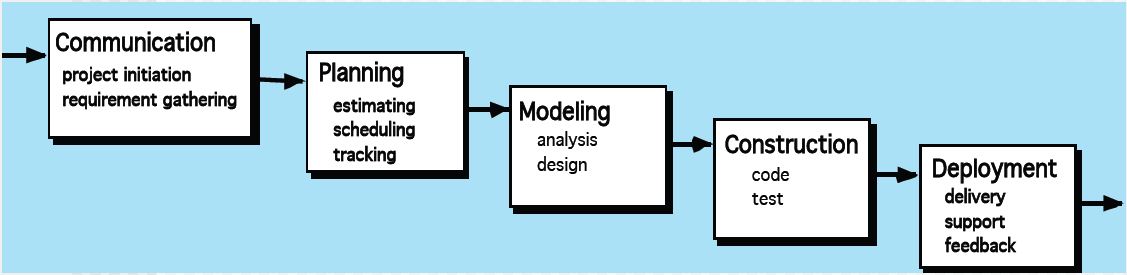
All of these models are described below.

**Waterfall Model:**

The waterfall Model was first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

In the *Waterfall* approach, the whole process of software development is divided into separate phases. In this model, the outcome of one phase acts as the input for the next phase sequentially.

The following diagram shows the various phases of software development in waterfall model:



**Communication:** As the figure suggests, the software project is initialized at this stage. All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.

**Planning:** Planning for the quality assurance requirements and identification of the risks associated with the project is done in this stage. The outcome of the technical feasibility study is to define the various technical approached that can be followed to implement the project successfully with minimum tasks.

**Modeling:** The requirement specifications from the first two phases are studies in this phase and system design is prepared. System design helps in specifying hardware and system requirements and also helps in analyzing overall system architecture.

**Construction:** With inputs from system design, the system is first developed in small programs called units, which are integrated in this phase as well. Each unit is developed and tested for its functionality.

**Deployment:** Once the functional and non functional testing is done, the product is deployed in the customer environment or released into the market. The final product is delivered at this phase, as well as customer support is provided. To provide better maintenance services, customer feedbacks are also accepted.

**Application:**

* Requirements are very well documented, clear and fixed
* Product definition is stable
* There are no ambiguous requirements
* The project is short

**Advantages:**

* Simple and easy to understand and use.
* Each phase has specific deliverables and a review process, therefore easy to manage.
* Phases are processed and managed one at a time.
* Clearly defined stages.
* Easy to arrange tasks.

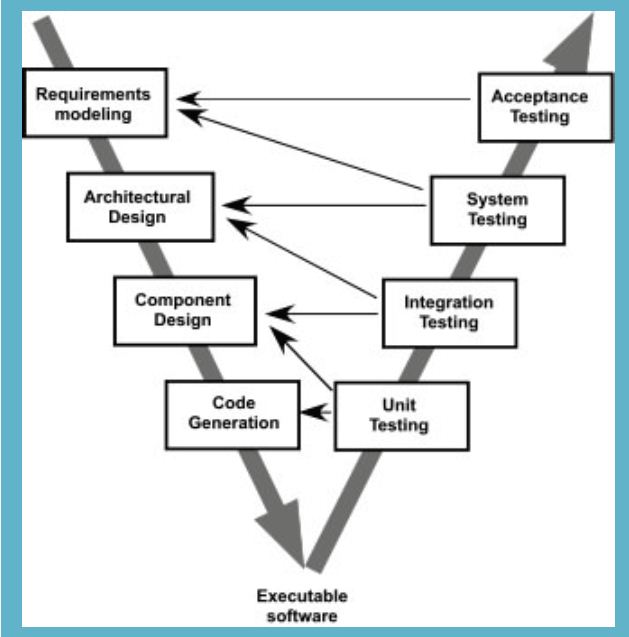
**Disadvantages:**

* No working software is produced until late during the life cycle.
* High amount of risk and uncertainty.
* Poor model for long and ongoing projects.
* Cannot accommodate changing requirements.
* It is difficult to measure progress within stages.

**V-Model:**

The V-model is one of the SDLC models where execution of process happens in a sequential manner in V shape. It is also known as Verification and Validation model. This model is an extension of the waterfall model and is based on association of a testing phase for each corresponding development stage. This means that for every single phase in the development cycle there is a directly associated testing phase.

Below figure shows various phases of the V-model:



**Requirements modeling:** this is the first phase where the product requirements are understood from the customer perspective. This phase involves detailed communication with the customer to understand their expectations and exact requirements.

**Architectural design:** Once you have the clear and detailed product requirements, it’s time to design the complete system. System design would comprise of understanding and detailing the complete hardware and communication setup for the product under development.

**Component design:** In this phase the detailed internal design for all the system modules is specified, referred to as Low Level Design. It is important that the design is compatible with the other modules in the system architecture and the other external systems.

**Code generation:** The actual coding of the system modules designed in the design phase is taken up in the coding phase. The best suitable programming language is decided based on the system and architectural requirements. The coding is performed based on the coding guidelines and standards.

**Unit testing:** This is the testing technique at code level and helps eliminate bugs at an early stage, though all defects cannot be uncovered by unit testing.

**Integration testing:** It is associated with the *Architectural design* phase. Integration tests are performed to test the coexistence and communication of the internal modules within the system.

**System testing:** This phase is directly associated with the *System design* phase. System tests check the entire system functionality and the communication of the system under development with external systems.

**Acceptance testing:** It is associated with the *Requirement analysis* phase and involves testing the product in user environment. These tests uncover the compatibility issues with the other systems available in the user environment.

**Applications:**

* Requirements are well defined, clearly documented and fixed.
* Product definition is stable.
* The project is short.
* There are no ambiguous or undefined requirements.

**Advantages:**

* This is a highly disciplined model and phases are completed one at a time.
* Works well for smaller projects where requirements are very well understood.
* Simple and easy to understand and use.
* Easy to manage.

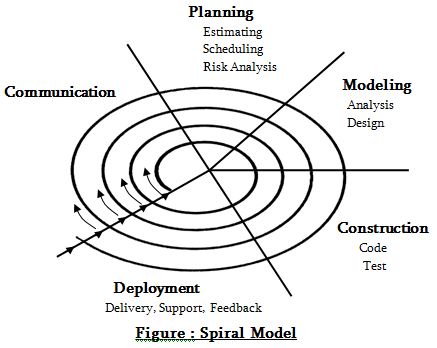
**Disadvantages:**

* High risks and uncertainty.
* Not a good model for complex and object-oriented projects.
* Poor model form long an ongoing projects.
* Not suitable for the projects where requirements are at a moderate to high risk of changing.
* Once application is in the testing stage, it is difficult to go back and change functionality.

**Spiral Model:**

The spiral model combines the idea of iterative development with the systematic, controlled aspects of the waterfall model. It is similar to the incremental model, with more emphasis place on *risk analysis.*

This model has five phases: Communication, Planning, Modeling, Construction and Deployment. A software project repeatedly passes through these phases in iterations.



**Communication:** All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.

**Planning:** Requirements are gathered during the planning phase. The requirements include SRS (Software Requirements Specifications), and BRS (Business Requirement Specification). In the risk analysis phase, a process is undertaken to identify risk and alternate solutions.

**Modeling:** Modeling phase starts with the conceptual design in the baseline spiral and involves architectural design, logical design of modules, physical product design and final design in the subsequent spirals.

**Construction:** In this phase software is developed, along with testing at the end of the phase. Hence in this phase the development and testing is done.

**Deployment:** This phase allows the customer to evaluate the output of the project to date before the project continues to the next spiral.

**Application:**

* When costs and risk evaluation is important
* For medium to high-risk projects
* Long-term project commitment unwise because of potential changes to economic priorities
* Requirements are complex

**Advantages:**

* High amount of risk analysis hence, avoidance of Risk is enhanced.
* Good for large and mission-critical projects
* Strong approval and documentation control
* Software is produced early in the *Software Life Cycle*.

**Disadvantages:**

* Can be a costly model to use.
* Risk analysis requires highly specific expertise
* Doesn’t work well, for smaller projects.
* Project’s success is highly dependent on the risk analysis phase.

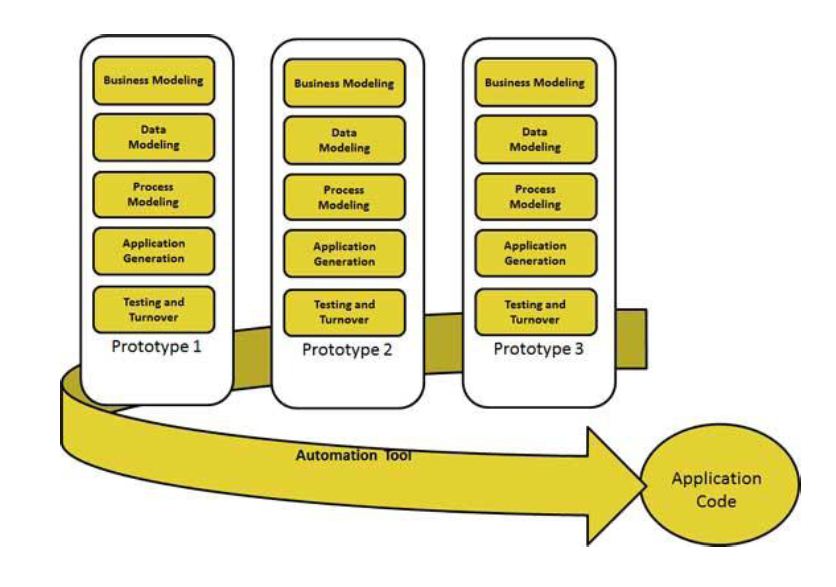
**RAD Model:**

The RAD (Rapid Application Development) model is based on prototyping and interactive development with no specific planning involved. The process of writing the software itself involves the planning required for developing the product.

RAD is a software development methodology that uses minimal planning in favor of rapid prototyping. A prototype is a working model that is functionally equivalent to a component of the product.

In RAD model, the functional modules are developed in parallel as prototypes and are integrated to make the complete product for faster product delivery.

The following image illustrates the RAD model:



**Business Modeling:** The business model for the product under development is designed in terms of flow of information and the distribution of information between various business channels.

**Data Modeling:** The information gathered in the Business modeling phase is reviewed and analyzed to form sets of data objects vital for the business.

**Process Modeling:** The data object sets defined in the Data Modeling phase are connected to establish the business information flow needed to achieve specific business objectives as per the business model.

**Application Generation:** The actual system is built and coding is done by using automation tools to convert process and data models into actual prototypes.

**Testing:** The overall testing time is reduced in RAD model as the prototypes are independently tested during every iteration.

**Application:**

* Used when a system can be modularized to be delivered in incremental manner.
* Used if there’s high availability of designers for modeling.
* Used only if the budget permits use of automated code generating tools.
* Used when the requirements change during the course of the project.

**Advantages:**

* Changing requirements can be accommodated.
* Progress can be measured.
* Iteration time can e short with use of powerful RAD tools.
* Productivity with fewer people in short time.
* Reduced development time.

**Disadvantages:**

* Dependency on technically strong team members for identifying business requirement.
* Only system that can be modularized can be built using RAD.
* Requires highly skilled developers.
* High dependency on modeling skills
* Inapplicable to cheaper projects as cost of modeling and automated code generation is very high.

**Summary:**

By going through all the process models described here and others (i.e. agile model, Component-based model etc.) that incorporates with software engineering process; it is clear that depending on the requirements and time limits, the Project Manager chooses an appropriate process model for the Software project. If the requirements do not change dynamically, then Waterfall and V-models are appropriate choices. In case of complex products, the Spiral model is helpful. Whereas process models like RAD model can be used in situations when requirement are to be better understood and product improvements are made upon the feedbacks.