4) PROCESS MODEL

A software process (also knows as software methodology) is a set of related activities that leads to the production of the software. These activities may involve the development of the software from the scratch, or, modifying an existing system.

There are various software development life cycle models defined and designed which are followed during the software development process. These models are also referred as Software Development Process Models. Each process model follows a Series of steps unique to its type to ensure success in the process of software development.

Following are the most important and popular SDLC models followed in the industry −

- ➢ Waterfall Model
- ➢ Incremental Model
- ➢ Prototyping model
- ➢ V-Model

The Waterfall Model was the 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.

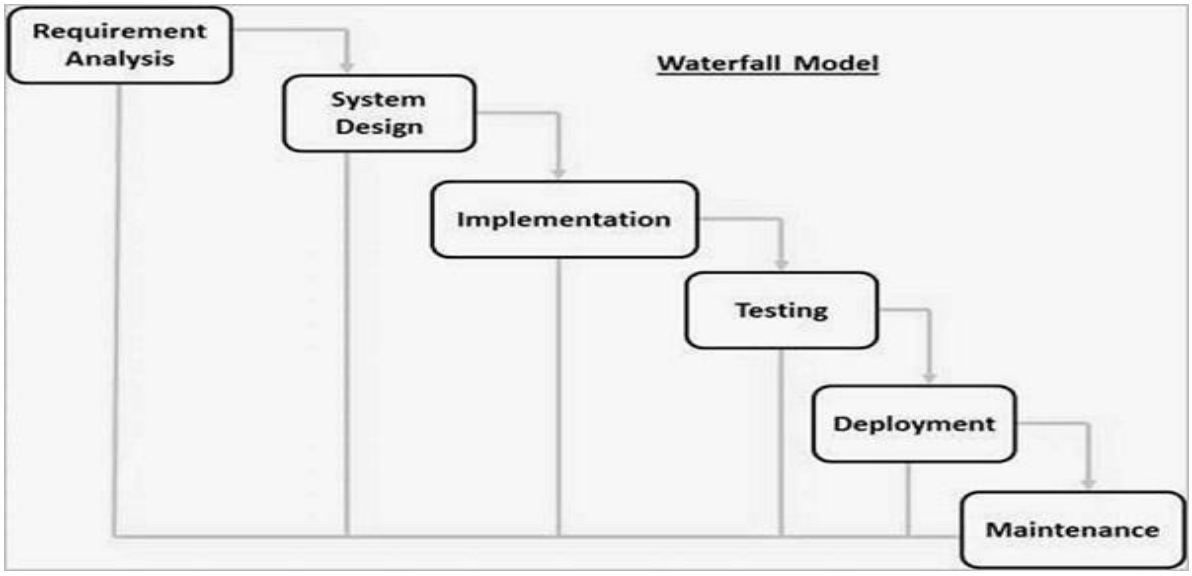
The Waterfall model is the earliest SDLC approach that was used for software development.

The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.

Design

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

The following illustration is a representation of the different phases of the Waterfall Model.



The sequential phases in Waterfall model are −

Requirement Gathering and analysis − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.

System Design − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.

Implementation − With inputs from the system design, 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, which is referred to as Unit Testing.

Integration and Testing − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

Deployment of system − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

Maintenance − There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model, phases do not overlap.

Application

Situation where Waterfall model is most appropriate are −

- ➢ Requirements are very well documented, clear and fixed.
- ➢ Product definition is stable.
- ➢ Technology is understood and is not dynamic.
- ➢ There are no ambiguous requirements.
- ➢ Ample resources with required expertise are available to support the product.

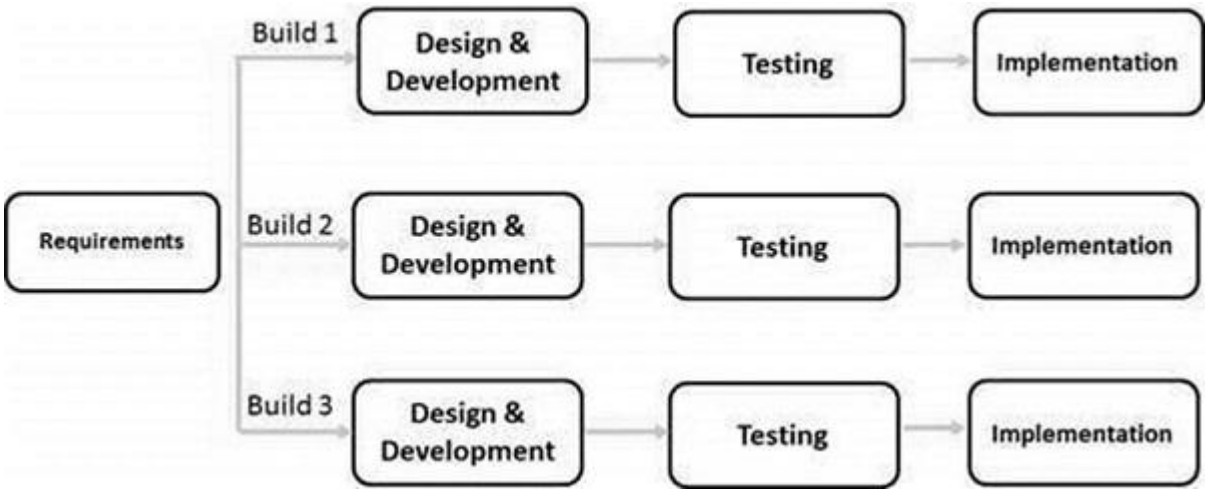
Disadvantages

- √ Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.
- √ No working software is produced until late during the life cycle.
- √ High amounts of risk and uncertainty.
- √ Not suitable for the projects where requirements are at a moderate to high risk of changing.So, risk and uncertainty is high with this process model.
- √ It is difficult to measure progress within stages.
- √ Cannot accommodate changing requirements.

In the **Iterative model**, iterative process starts with a simple implementation of a small set of the software requirements and iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed. An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which is then reviewed to identify further requirements. This process is then repeated, producing a new version of the software at the end of each iteration of the model.

Design

Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented. At each iteration, design modifications are made and new functional capabilities are added. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).
The following illustration is a representation of the Iterative and Incremental model −



Iterative and Incremental development is a combination of both iterative design or iterative method and incremental build model for development. "During software development, more than one iteration of the software development cycle may be in progress at the same time." This process may be described as an "evolutionary acquisition" or "incremental build" approach."

In this incremental model, the whole requirement is divided into various builds. During each iteration, the development module goes through the requirements, design, implementation and testing phases. Each subsequent release of the module adds function to the previous release. The process continues till the complete system is ready as per the requirement.

Application

This model is most often used in the following scenarios −

- √ Major requirements must be defined; however, some functionalities or requested enhancements may evolve with time.
- √ There is a time to the market constraint.
- √ Resources with needed skill sets are not available and are planned to be used on contract basis for specific iterations.
- √ There are some high-risk features and goals which may change in the future.

Disadvantage

It is applicable only to large and bulky software development projects. This is because it is hard to break a small software system into further small serviceable increments/modules.

The ***advantages*** of the Iterative and Incremental SDLC Model are as follows −

Some working functionality can be developed quickly and early in the life cycle.

- ➢ Results are obtained early and periodically.
- ➢ Parallel development can be planned.
- ➢ Progress can be measured.

- ➢ Less costly to change the scope/requirements.
- ➢ Testing and debugging during smaller iteration is easy.
- ➢ Risks are identified and resolved during iteration; and each iteration is an easily managed milestone.
- ➢ Easier to manage risk - High risk part is done first.
- ➢ With every increment, operational product is delivered.
- ➢ Issues, challenges and risks identified from each increment can be utilized/applied to the next increment.
- ➢ Risk analysis is better.
- ➢ It supports changing requirements.
- ➢ Initial Operating time is less.
- ➢ Better suited for large and mission-critical projects.

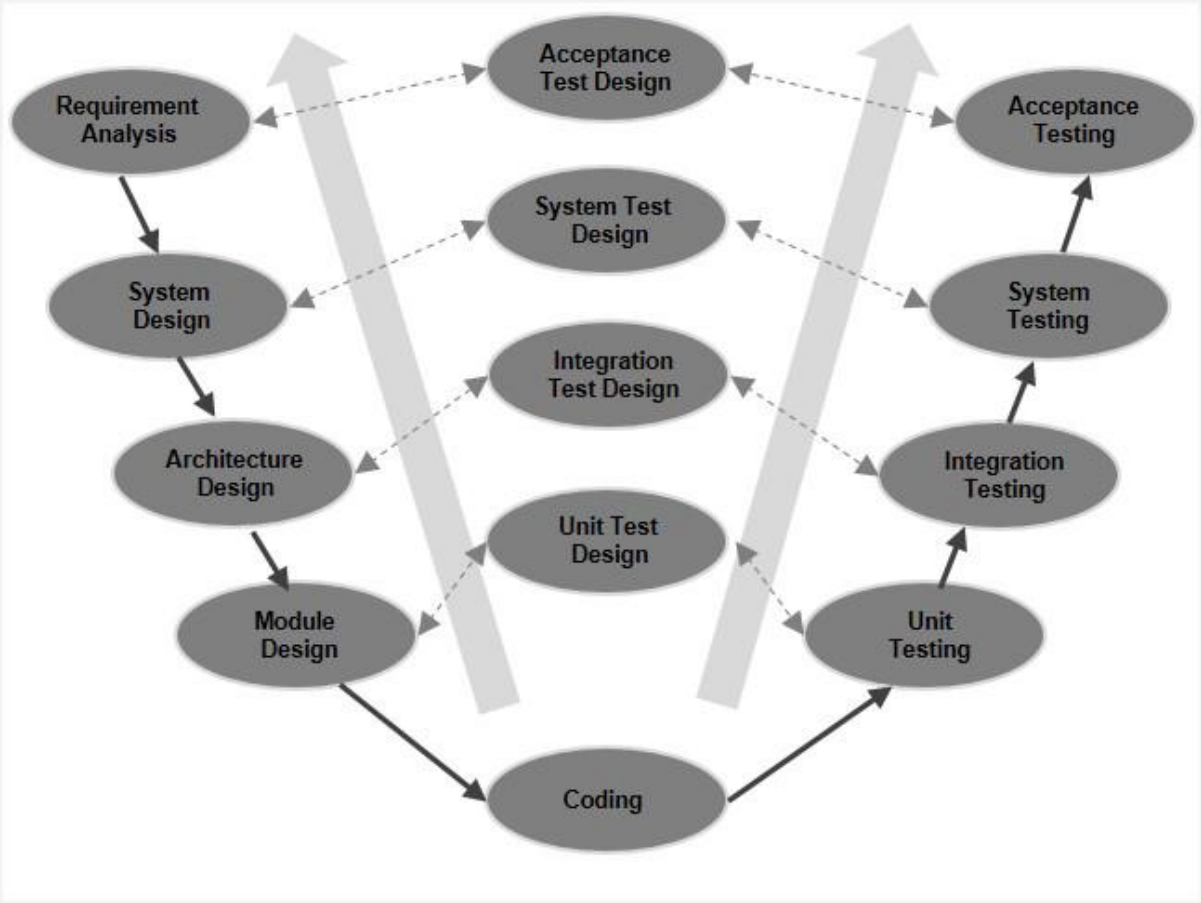
The V-model is an SDLC model where execution of processes happens in a sequential manner in a V-shape. It is also known as **Verification and Validation model**.

The V-Model is an extension of the waterfall model and is based on the 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. This is a highly-disciplined model and the next phase starts only after completion of the previous phase.

Design

Under the V-Model, the corresponding testing phase of the development phase is planned in parallel. So, there are Verification phases on one side of the 'V' and Validation phases on the other side. The Coding Phase joins the two sides of the V-Model.

The following illustration depicts the different phases in a V-Model of the SDLC.



V-Model - Verification Phases

There are several Verification phases in the V-Model, each of these are explained in detail below.

Business Requirement Analysis

This is the first phase in the development cycle where the product requirements are understood from the customer's perspective. This phase involves detailed communication with the customer to understand his expectations and exact requirement. This is a very important activity and needs to be managed well, as most of the customers are not sure about what exactly they need.

The **acceptance test design planning** is done at this stage as business requirements can be used as an input for acceptance testing.

System Design

Once you have the clear and detailed product requirements, it is time to design the complete system. The system design will have the understanding and detailing the complete hardware and communication setup for the product under development. The system test plan is developed based on the system design. Doing this at an earlier stage leaves more time for the actual test execution later.

Architectural Design

Architectural specifications are understood and designed in this phase. Usually more than one technical approach is proposed and based on the technical and financial feasibility the final decision is taken. The system design is broken

down further into modules taking up different functionality. This is also referred to as **High Level Design (HLD)**.
The data transfer and communication between the internal modules and with the outside world (other systems) is clearly understood and defined in this stage. With this information, integration tests can be designed and documented during this stage.

Module Design

In this phase, the detailed internal design for all the system modules is specified, referred to as **Low Level Design (LLD)**. It is important that the design is compatible with the other modules in the system architecture and the other external systems. The unit tests are an essential part of any development process and helps eliminate the maximum faults and errors at a very early stage. These unit tests can be designed at this stage based on the internal module designs.

Coding Phase

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. The code goes through numerous code reviews and is optimized for best performance before the final build is checked into the repository.

Validation Phases

The different Validation Phases in a V-Model are explained in detail below.

Unit Testing

Unit tests designed in the module design phase are executed on the code during this validation phase. Unit testing is the testing at code level and helps eliminate bugs at an early stage, though all defects cannot be uncovered by unit testing.

Integration Testing

Integration testing 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

System testing 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. Most of the software and hardware compatibility issues can be uncovered during this system test execution.

Acceptance Testing

Acceptance testing is associated with the business requirement analysis phase and involves testing the product in user environment. Acceptance tests uncover the compatibility issues with the other systems available in the user environment. It also discovers the non-functional issues such as load and performance defects in the actual user environment.

V- Model application is almost the same as the waterfall model, as both the models are of sequential type. Requirements have to be very clear before the project starts, because it is usually expensive to go back and make changes. This model is used in the medical development field, as it is strictly a disciplined domain.

The following pointers are some of the most suitable scenarios to use the V-Model application.

Requirements are well defined, clearly documented and fixed.

Product definition is stable.

Technology is not dynamic and is well understood by the project team.

There are no ambiguous or undefined requirements.

The project is short.

V-Model - Pros and Cons

The advantage of the V-Model method is that it is very easy to understand and apply. The simplicity of this model also makes it easier to manage. The disadvantage is that the model is not flexible to changes and just in case there is a requirement change, which is very common in today's dynamic world, it becomes very expensive to make the change.

The advantages of the V-Model method are as follows −

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 due to the rigidity of the model. Each phase has specific deliverables and a review process.

The disadvantages of the V-Model method are as follows −

High risk and uncertainty.

Not a good model for complex and object-oriented projects.

Poor model for long and ongoing projects.

Not suitable for the projects where requirements are at a moderate to high risk of changing.

Once an application is in the testing stage, it is difficult to go back and change a functionality.

No working software is produced until late during the life cycle.

The prototyping model is a systems development method in which a prototype is built, tested and then reworked as necessary until an acceptable outcome is achieved from which the complete system or product can be developed. This model works best in scenarios where not all of the project requirements are known in detail ahead of time. It is an iterative, trial-and-error process that takes place between the developers and the users.

Steps of the prototyping model

In most cases, the steps of the prototyping model are as follows:

The new system requirements are defined in as much detail as possible. This usually involves interviewing a number of users representing all the departments or aspects of the existing system.
A preliminary, simple design is created for the new system.
A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
The users thoroughly evaluate the first prototype and note its strengths and weaknesses, what needs to be added and what should to be removed. The developer collects and analyzes the remarks from the users.
The first prototype is modified, based on the comments supplied by the users, and a second prototype of the new system is constructed.
The second prototype is evaluated in the same manner as was the first prototype.
The preceding steps are iterated as many times as necessary, until the users are satisfied that the prototype represents the final product desired.
The final system is constructed, based on the final prototype.
The final system is thoroughly evaluated and tested. Routine maintenance is carried out on a continuing basis to prevent large-scale failures and to minimize downtime.

Types of prototype models

There are a few types of prototype models that can be implemented by development teams based on their needs:
Rapid throwaway- This method involves exploring ideas by quickly developing a prototype based on preliminary requirements that is then revised through customer feedback. The name rapid throwaway refers to the fact that each prototype is completely discarded and may not be a part of the final product.
Evolutionary- This approach uses a continuous, working prototype that is refined after each iteration of customer feedback. Because each prototype is not started from scratch, this method saves time and effort.
Incremental- This technique breaks the concept for the final product into smaller pieces, and prototypes are created for each one. In the end, these prototypes are merged into the final product.
Extreme- This prototype model is used specifically for web development. All web prototypes are built in an HTML format with a services layer and are then integrated into the final product.

Advantages of the prototyping model

Using a prototype model can bring multiple advantages, including:
Customers get a say in the product early on, increasing customer satisfaction.
Missing functionality and errors are detected easily.
Prototypes can be reused in future, more complicated projects.
It emphasizes team communication and flexible design practices.
Users have a better understanding of how the product works.
Quicker customer feedback provides a better idea of customer needs.

Disadvantages of the prototyping model

The main disadvantage of this methodology is that it is more costly in terms of time and money when compared to alternative development methods, such as the spiral or waterfall model. Since in most cases the prototype is discarded, some companies may not see the value in taking this approach.