**SDLC Models :**

**Waterfall 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.

 In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

Fig. Waterfall Model.



**Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured 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.

**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 different 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.

**Advantages :**

* Phases are processed and completed one at a time.
* Clearly defined stages.
* Easy to arrange tasks.
* Process and results are well documented.

**Disadvantages :**

* Not a good model for complex and object-oriented projects.
* Poor model for long and ongoing projects.
* High amounts of risk and uncertainty.

**Agile Model :**

Agile SDLC model is a combination of iterative and incremental process models. Agile Methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. Every iteration involves cross functional teams working simultaneously on various areas like –Planning, Requirements Analysis, Design, Coding, Unit Testing and Acceptance Testing.

Agile model believes that every project needs to be handled differently. In Agile, the tasks are divided to time boxes (small time frames) to deliver specific features for a release.

Iterative approach is taken and working software build is delivered after each iteration. Each build is incremental in terms of features.

Fig. Agile Model −



**Advantages :**

* Promotes teamwork and cross training.
* Is a very realistic approach to software development.
* Good model for environments that change steadily.
* Little or no planning required.
* Minimal rules, documentation easily employed.

**Disadvantages :**

* Not suitable for handling complex dependencies.
* Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction.
* More risk of maintainability and extensibility.

**V Model :**

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.

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.

fig V-Model.



## 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.

**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.

**Disadvantages**

* 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.

**Prototype model**

The Software Prototyping refers to building software application prototypes which displays the functionality of the product under development, but may not actually hold the exact logic of the original software.

Software prototyping is becoming very popular as a software development model, as it enables to understand customer requirements at an early stage of development. It helps get valuable feedback from the customer and helps software designers and developers understand about what exactly is expected from the product under development.

Following is a stepwise approach explained to design a software prototype.

### Basic Requirement Identification

This step involves understanding the very basics product requirements especially in terms of user interface. The more intricate details of the internal design and external aspects like performance and security can be ignored at this stage.

### Developing the initial Prototype

The initial Prototype is developed in this stage, where the very basic requirements are showcased and user interfaces are provided. These features may not exactly work in the same manner internally in the actual software developed. While, the workarounds are used to give the same look and feel to the customer in the prototype developed.

### Review of the Prototype

The prototype developed is then presented to the customer and the other important stakeholders in the project. The feedback is collected in an organized manner and used for further enhancements in the product under development.

### Revise and Enhance the Prototype

The feedback and the review comments are discussed during this stage and some negotiations happen with the customer based on factors like – time and budget constraints and technical feasibility of the actual implementation. The changes accepted are again incorporated in the new Prototype developed and the cycle repeats until the customer expectations are met.

Prototypes can have horizontal or vertical dimensions. A Horizontal prototype displays the user interface for the product and gives a broader view of the entire system, without concentrating on internal functions. A Vertical prototype on the other side is a detailed elaboration of a specific function or a sub system in the product.

The purpose of both horizontal and vertical prototype is different. Horizontal prototypes are used to get more information on the user interface level and the business requirements. It can even be presented in the sales demos to get business in the market. Vertical prototypes are technical in nature and are used to get details of the exact functioning of the sub systems. For example, database requirements, interaction and data processing loads in a given sub system.

## Software Prototyping - Types

There are different types of software prototypes used in the industry. Following are the major software prototyping types used widely −

### Throwaway/Rapid Prototyping

Throwaway prototyping is also called as rapid or close ended prototyping. This type of prototyping uses very little efforts with minimum requirement analysis to build a prototype. Once the actual requirements are understood, the prototype is discarded and the actual system is developed with a much clear understanding of user requirements.

### Evolutionary Prototyping

Evolutionary prototyping also called as breadboard prototyping is based on building actual functional prototypes with minimal functionality in the beginning. The prototype developed forms the heart of the future prototypes on top of which the entire system is built. By using evolutionary prototyping, the well-understood requirements are included in the prototype and the requirements are added as and when they are understood.

### Incremental Prototyping

Incremental prototyping refers to building multiple functional prototypes of the various sub-systems and then integrating all the available prototypes to form a complete system.

### Extreme Prototyping

Extreme prototyping is used in the web development domain. It consists of three sequential phases. First, a basic prototype with all the existing pages is presented in the HTML format. Then the data processing is simulated using a prototype services layer. Finally, the services are implemented and integrated to the final prototype. This process is called Extreme Prototyping used to draw attention to the second phase of the process, where a fully functional UI is developed with very little regard to the actual services.

**Advantages :**

* Increased user involvement in the product even before its implementation.
* Since a working model of the system is displayed, the users get a better understanding of the system being developed.
* Reduces time and cost as the defects can be detected much earlier.
* Quicker user feedback is available leading to better solutions.
* Missing functionality can be identified easily.

**Disadvantages**

* Risk of insufficient requirement analysis owing to too much dependency on the prototype.
* Users may get confused in the prototypes and actual systems.
* Practically, this methodology may increase the complexity of the system as scope of the system may expand beyond original plans.
* Developers may try to reuse the existing prototypes to build the actual system, even when it is not technically feasible.

**Spiral model**

The spiral model combines the idea of iterative development with the systematic, controlled aspects of the waterfall model. This Spiral model is a combination of iterative development process model and sequential linear development model i.e. the waterfall model with a very high emphasis on risk analysis. It allows incremental releases of the product or incremental refinement through each iteration around the spiral.

## Spiral Model - Design

The spiral model has four phases. A software project repeatedly passes through these phases in iterations called Spirals.

### Identification

This phase starts with gathering the business requirements in the baseline spiral. In the subsequent spirals as the product matures, identification of system requirements, subsystem requirements and unit requirements are all done in this phase.

This phase also includes understanding the system requirements by continuous communication between the customer and the system analyst. At the end of the spiral, the product is deployed in the identified market.

### Design

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

### Construct or Build

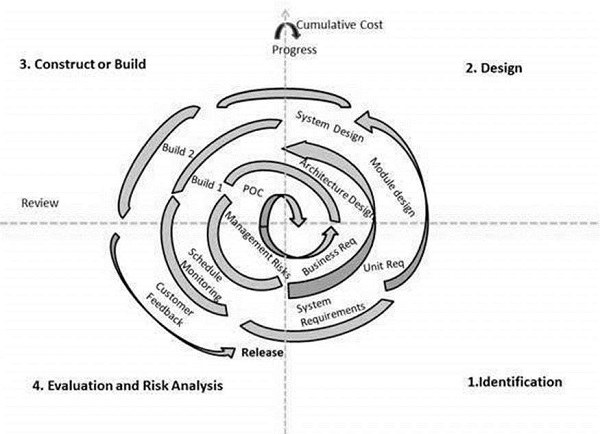
The Construct phase refers to production of the actual software product at every spiral. In the baseline spiral, when the product is just thought of and the design is being developed a POC (Proof of Concept) is developed in this phase to get customer feedback.

Then in the subsequent spirals with higher clarity on requirements and design details a working model of the software called build is produced with a version number. These builds are sent to the customer for feedback.

### Evaluation and Risk Analysis

Risk Analysis includes identifying, estimating and monitoring the technical feasibility and management risks, such as schedule slippage and cost overrun. After testing the build, at the end of first iteration, the customer evaluates the software and provides feedback.

The following illustration is a representation of the Spiral Model, listing the activities in each phase.



Based on the customer evaluation, the software development process enters the next iteration and subsequently follows the linear approach to implement the feedback suggested by the customer. The process of iterations along the spiral continues throughout the life of the software.

**Advantages**

* Changing requirements can be accommodated.
* Allows extensive use of prototypes.
* Requirements can be captured more accurately.
* Users see the system early.
* Development can be divided into smaller parts and the risky parts can be developed earlier which helps in better risk management.

**Disadvantages**

* Management is more complex.
* End of the project may not be known early.
* Not suitable for small or low risk projects and could be expensive for small projects.
* Process is complex
* Spiral may go on indefinitely.
* Large number of intermediate stages requires excessive documentation