SDLC Agile Model

The iterative waterfall model was popular in the olden days for completing a project. But, following this model, the software developer had to face various issues and difficulties especially when customers request for handling change or there is an update in the system requirement. So to mitigate such problems and drawbacks in older models, the new model was incorporated in the year 1990, and they proposed the model with a name Agile Software Development. In this tutorial, you will learn about the structure and working of the agile model.

What is the Agile Model?

The Agile software development model was mainly intended for helping developers build a project which can adapt to transforming requests quickly. So, the most important endeavour for developing the Agile model is to make easy and rapid project achievement. For attaining this task, developers need to preserve the agility during development. Agility can be achieved by correcting the progression to the project by eliminating activities which may not be crucial for that specific project.

For each iteration, engages a cross-functional group of developers functioning concurrently on various areas of product development such as:

1. Planning
2. Requirements Analysis
3. Design
4. Development
5. Unit Testing
6. Deployment

Once all the iterations are completed giving rise to a successful product, it is displayed to the customers and other stakeholders who will approve and accept the final product.

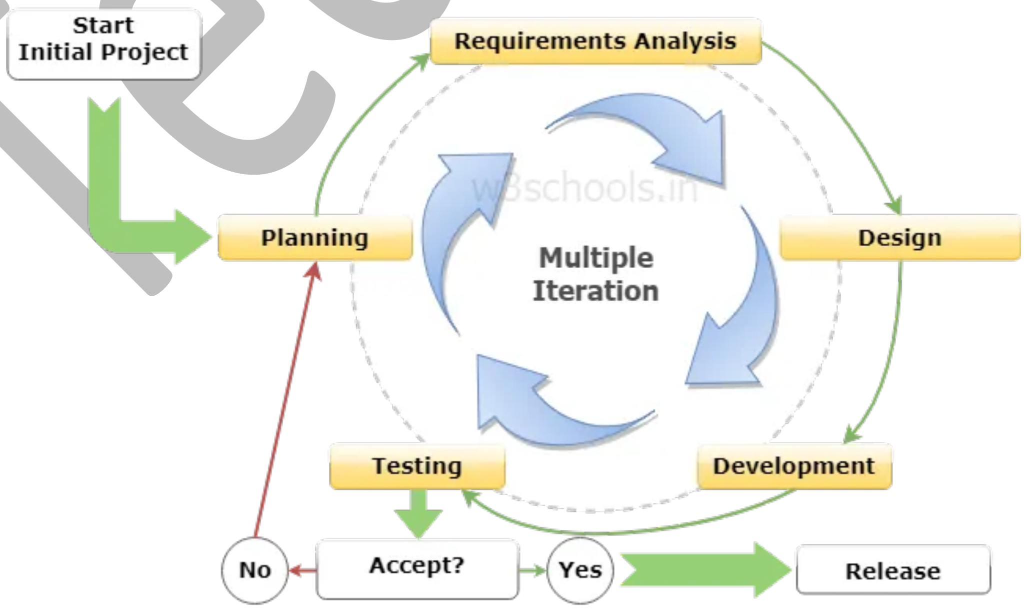


Fig: SDLC Agile Software Development Model

https://ankit11191.github.io/testadda/

More about Agile Methodology of Software Development

Essentially, the Agile model is a collective iteration grouped to do the development practice of a product. These procedures share some essential qualities but do have assured little differences between each of them. Here are lists of different kinds of Agile SDLC models that are used by software development companies. These are:

- Feature-driven development
- Crystal
- Atern
- Scrum
- Unified process
- Extreme programming (XP)
- Lean development

Amongst them, Scrum, Lean and Extreme programming are some of the most popular forms of Agile development methodologies.

Manifesto Principles of Agile Model

- For launching a close connection with the client throughout the development as well as to achieve a public perception of a variety of requirements; each Agile development process typically comprises of a customer representative within the development team.
- This model also relies on effective software deployment methodology rather than having comprehensive documentation.
- This model also provides regular release of incremental versions of the product to its customers and stakeholders in an interval of 2 to 3 weeks.
- Incorporation of change in customer's system requirement is always welcome entertained at any point of development.
- The agile model needs all its team members to be efficient, and there should have to be healthy communication among them for better development of the product. It is found that efficient and proper communication between team members is possible by the use of face-to-face discussion rather than exchanging emails or through online medium.

Benefits of Agile Methodologies

- Involves pair programming which reduces the number of errors in the development or coding phase and is better than a single programmer doing all the hard part.
- This model trims down the entire development time of any project.
- After each iteration, customers and stakeholders of the project can get a fair idea the updated software that is being developed by the agile model. So, any change in the system can be addressed at any iteration.

SDLC Waterfall Model

SDLC has different models designed which have their advantages and disadvantages. Waterfall model is a traditional SDLC model which will be discussed in this chapter. It is a straightforward and basic structure which can be easily understandable by software developers and testers. It is the first model of SDLC to be introduced for software development.

What is the Waterfall Model?

The classical waterfall model which is also known as the linear-sequential life cycle model is an essential software development model which can be understandable from the structure itself. The model is straightforward yet idealistic. When this model was first introduced, it used to be very popular, but time, the new model has come up with a change in features and requirements and hence it is used decidedly less but still a popular one which everyone must know. All the old software has been developed based on this model's life cycle. It is a sequential model which segregates software development into different phases. Each phase is designed with some unique functionality and use. The model was pioneered in the year 1970 by Winston Royce.

Stages of the Waterfall Model

The various phases of the Waterfall Model which are explained below:

1. Requirement Gathering Stage/Feasibility Study
2. Design Stage
3. Built Stage
4. Integration and Test Stage
5. Deployment Stage
6. Maintenance Stage

The different chronological phases of the waterfall model are shown below with the interconnection between them:

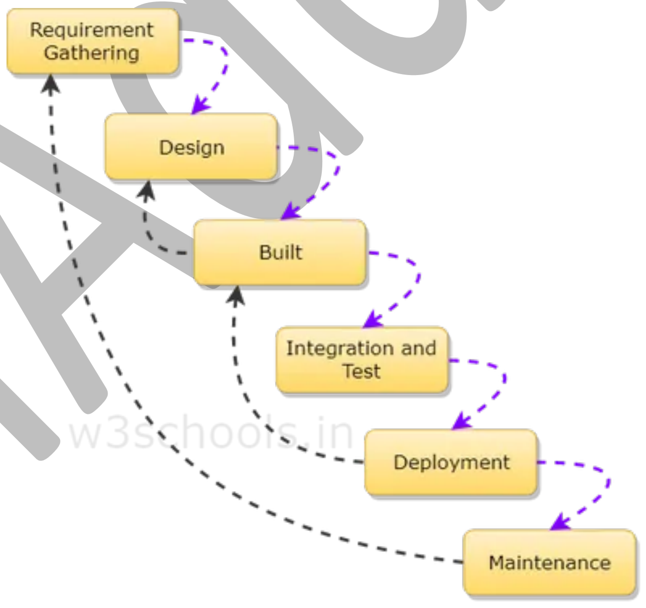


Fig: SDLC Waterfall Model

Requirement Gathering Stage/Feasibility Study

This phase has the purpose to establish whether it would be monetarily as well as technically practicable to expand the development of software. This has the achievability study with the understanding of the problem as well as determines the diverse potential strategies used for solving the problem.

Design Stage

There is a thorough study of the entire requirement specifications from the first phase, and then the system design is equipped. This phase helps developers to specify hardware as well as the system's requirement which ultimately helps in characterizing the system design as a whole.

Built Stage

This phase is also known as the coding phase of software development where the idea is converted into source code and UI plus UX design using programming language and tools. Hence, every designed module needs to be coded.

Integration and Test Stage

Once the coding of application is done, it is then integrated with all other modules with different functionality. During each step of integration, earlier planned modules are incorporated into the parts included the structure of the software and then the entire system is tested.

1. **α Testing**: In this testing, the software is tested by the development team, i.e., the developers.
2. **β Testing**: In this testing, the software is tested by friendly customers and other target users who will use the beta version of your product.
3. **Acceptance Testing**: Once the application has been distributed, the customer carries out the acceptance test for determining if the product should be accepted as delivered or rejects it for further modification.

Deployment Stage

As all the functional, as well as non-functional tests, are completed, the software is installed in the customer's end or the environment or gets released in the market.

Maintenance Stage

Another important phase of this model is the maintenance model. Updating the product, patching any bugs and errors and developing other essential components as per feedback to make this full software is done in this stage. It is of three types:

1. **Corrective Maintenance**: Corrective maintenance is where the maintenance is done to fix the errors.
2. **Perfective Maintenance**: Perfective maintenance is done where the maintenance is done to increase the efficiency of any system according to customer's requirement.
3. **Adaptive Maintenance**: Adaptive Maintenance is typically necessary for porting your application to a new work environment or porting from one type of OS to another.

SDLC Spiral Model

The spiral model is another important SDLC model that came into use when the iteration in product development came into the applied concept. The initial phase of the Spiral model is the early stages of Waterfall Life Cycle that are needed to develop a software product. This model supports risk handling, and the project is delivered in loops. Each loop in the Spiral model is the phases of the software development process.

What is Spiral model?

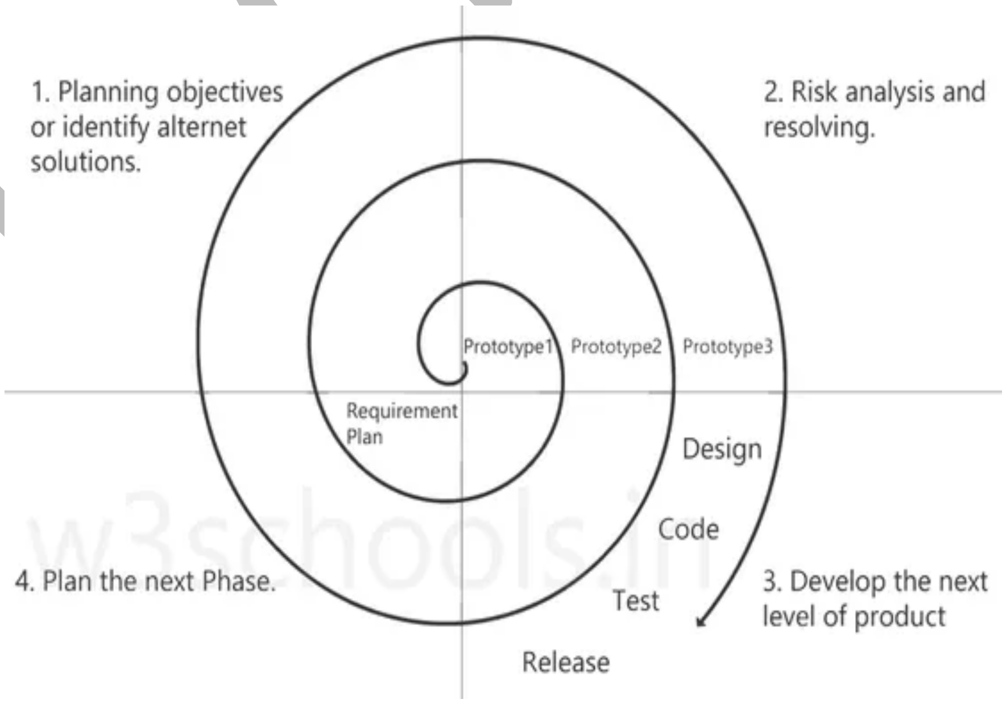
The popular spiral model is a blend of both iterative development method as well as sequential improvement model, i.e., the waterfall model that is having exceptionally high importance on risk analysis. In this model, the exact number of phases for developing a product varied based on some constraints and by project manager which calculates the project risks. Here the project manager dynamically decides the number of phases and hence play a significant role in the development of a product using the spiral model. The radius in spiral usually shows the expenses or cost needed for project development. The angular dimension shows the development done to date during the recent phase.

Different Phases of the Spiral model

The phase of the spiral model has four quadrants, and each of them represents some specific stage of software development. The functions of these four quadrants are listed below:

1. **Planning objectives or identify alternative solutions:** In this stage, requirements are collected from customers and then the aims are recognized, elaborated as well as analyzed at the beginning of developing the project. If the iterative round is more than one, then an alternative solution is proposed in the same quadrant.
2. **Risk analysis and resolving:** As the process goes to the second quadrant, all likely solutions are sketched, and then the best solution among them gets select. Then the different types of risks linked with the chosen solution are recognized and resolved through the best possible approach. As the spiral goes to the end of this quadrant, a project prototype is put up for the most excellent and likely solution.
3. **Develop the next level of product:** As the development progress goes to the third quadrant, the well-known and mostly required features are developed as well as verified with the testing methodologies. As this stage proceeds to the end of this third quadrant, new software or the next version of existing software is ready to deliver.
4. **Plan the next Phase:** As the development process proceeds in the fourth quadrant, the customers appraise the developed version of the project and reports if any further changes are required. At last, planning for the subsequent phase is initiated.

Graphical Presentation of the Spiral Model



1. Planning objectives or identify alternet solutions.
2. Risk analysis and resolving.
Prototype1
Prototype2
Prototype3
Requirement Plan
Design
Code
Test
Release
3. Develop the next level of product
4. Plan the next Phase.
w3schools.in

Advantages of the Spiral Model

The spiral model has some advantages compared to other SDLC models:

- **Suitable for large projects:** Spiral models are recommended when the project is large, bulky or complex to develop.
- **Risk Handling:** There are a lot of projects that have un-estimated risks involved with them. For such projects, the spiral model is the best SDLC model to pursue because it can analyze risk as well as handling risks at each phase of development.
- **Customer Satisfaction:** Customers can witness the development of product at every stage and thus, they can let themselves habituated with the system and throw feedbacks accordingly before the final product is made.
- **Requirements flexibility:** All the specific requirements needed at later stages can be included precisely if the development is done using this model.

SDLC V-Model

The software development life cycle has various models which follow a different approach to culminate a prototype to a successful product. In the previous chapter, you have learned about the Waterfall model - the V-model is an extension of the waterfall model. Usually, this model is pronounced as Vee model. This model is quite different from the Waterfall model because, in every phase of it, there is a related testing stage associated. In this chapter, you will learn about the V-Model and its approach to software development.

What is the V-Model?

The V-model of SDLC carries out its execution in a sequential manner. The structure it follows takes the shape of the letter V. This model is also popularly termed as a **Verification and Validation model**. Here, each phase has to be finished before beginning the next phase. A sequential design progression is followed like that of the waterfall model.

The Design of the SDLC V-Model

In parallel to the software development phase, a corresponding series of test phase also runs in this model. Each stage comprises a specific type of testing done, and once that testing is passed, only then the next phase starts.

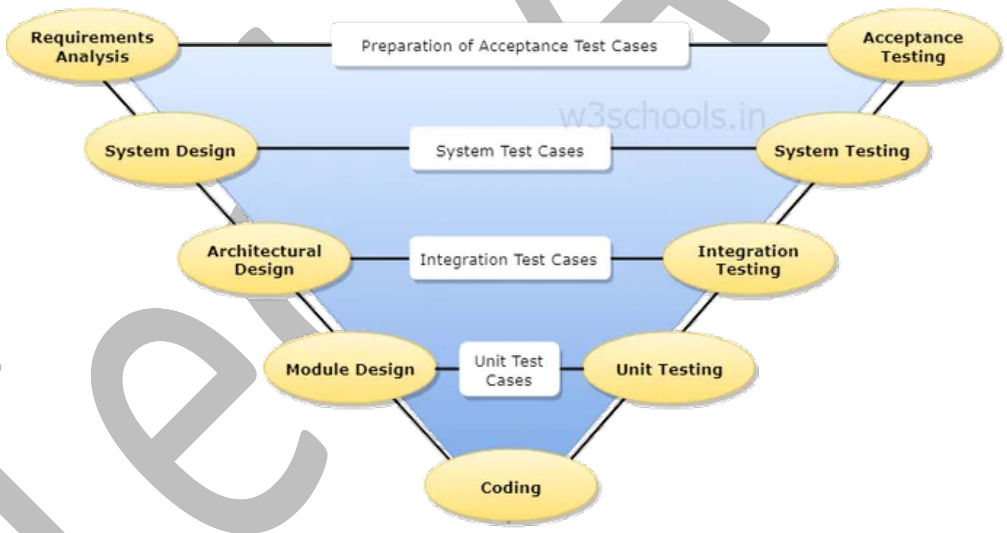


Fig: The Design of the SDLC V-Model

1. **Verification:** In the concept of verification in the V-Model, static analysis technique is carried out without executing the code. This evaluation procedure is carried out at the time of development to check whether specific requirements will meet or not.
2. **Validation:** This concept of V-Model comprises of dynamic analysis practice (both functional as well as non-functional), and testing is done by code execution. The validation of a product is done once the development is complete for determining if the software meets up the customer hope needs.

So both verification and validation are combined and work in parallel to make the V-Model fully functional.

Design Phase

1. **Requirement Analysis**: In this stage of SDLC, a detailed conversation with the customer is made to understand their requirements as well as anticipation. Requirement gathering is another name of this phase.
2. **System Design or High-level Design**: In this phase of SDLC, the system is designed with the entire hardware & the setup is constructed for product development.
3. **Architectural Design**: The breakdown of system design to a more detailed version, i.e., into modules which creates different functionalities. Transferring of data and connection between internal and external modules (i.e., the outside world) is evidently identified.
4. **Low-level design or Module Design**: This particular phase breaks down the entire product development into tiny modules where each intended module is specified. So it is also termed as Low-Level Design (LLD).

Testing Phase

1. **Unit Testing**: During the development of module design, unit testing is carried out. This plan is executed for eliminating bugs that are found in code at the development of your software.
2. **Integration Testing**: Once the unit testing is done, the integration testing is carried out where the integration of modules in the system is hardened. This testing is done in the architecture design phase.
3. **System Testing**: This ultimate test is done when the entire product is completed in conjunction with the functionality, internal dependency requirement and merging of different modules into a single unit.
4. **User Acceptance Testing**: This type of testing is carried out in front of the user or in a user environment where the product will ultimately set up. The UAT particularly test whether the product is capable enough to launch in the market or ready to work in the real world.