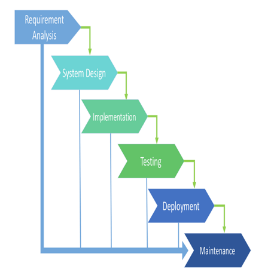
*Software Development Life Cycle Models*

1. *Waterfall Model - Probably the oldest and most well-known model. This model is built like a waterfall, as the name would imply. This model's flow can be compared to a waterfall, which moves from higher to lower points before arriving at its destination. The other step's input is the output of the first step. Up until the software is developed, this process continues.*



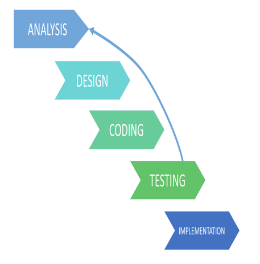
*Advantages:*

* *This paradigm is straightforward and easy to understand. Because of this, teamwork is incredibly simple, and everyone is on the same page.*
* *With this model, deadlines are met with ease. This is because the team has previously been given precise instructions and no time is lost trying to figure out how to proceed.*
* *Since this type of architecture involves a lot of paperwork, it is better suited for larger projects with greater teams.*

*Disadvantages:*

* *Adjusting the product at a later stage in the project is practically difficult. This is because the steps are tied to one another in a sequential process.*
* *The inability of the stakeholders and customers to utilise or view the product right away is another drawback of this approach. The customer will have to wait a few months before they can view the goods, which occasionally causes them to feel uncomfortable or disappointed.*
* *As the product cannot be altered at a later stage of the development process, there is also no space for error. Therefore, extensive investigation is required.*
* *A paradigm like this is not appropriate for small teams because it necessitates excessive research, which raises the cost.*

1. *Iterative Model - This type of model doesn't require a lot of planning. This concept necessitates a method where the entire project is broken down into smaller components. Iterations are each individual little piece. After breaking the project down into iterations, each iteration goes through all 7 phases of the software development life cycle. The nicest thing about this kind of model is that after one iteration, a tiny portion of the project is finished and may be released to the market. Customers and other stakeholders benefit greatly from this since they may test the product on the market and are involved throughout the entire product development process.*

**

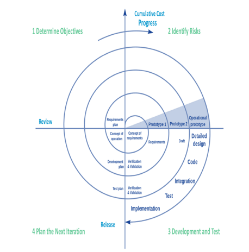
*Advantages:*

* *It is feasible to make minor adjustments to the project.*
* *There is not much planning and preparation necessary for this process.*
* *It is simpler to deploy and analyse software in the market when little pieces of it are released.*
* *The team can see the results of their work and the project's development can be followed in real time, which will keep them motivated.*
* *It is appropriate for larger projects with huge crews.*

*Disadvantages:*

* *Making adjustments to a project that is already underway is not recommended because it is a very sensitive procedure that involves a lot of management. Better planning is suggested.*
* *Lack of preparation will result in less-than-specific requirements, which may lead to coding errors.*
* *For this project, highly qualified personnel are needed.*

1. *Spiral Model - This model deviates greatly from the norm. It is a hybrid model that combines elements of the waterfall approach and the iterative model. The intriguing feature of this model is that it enables us to evaluate the product at each stage, ensuring the creation of a top-notch final result. This SDLC differs slightly from the cycles for the other models.*

**

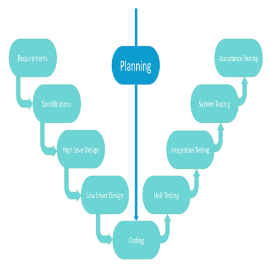
*Advantages:*

* *It enables the developers to easily add new functionality to the software that was before impossible to do.*
* *We can receive feedback following each spiral, that allows us to continuously enhance our offering.*
* *This methodology enables us to launch the software sooner, which benefits both your consumers and your team.*
* *Tasks can be ranked in order of importance based on market demands.*

*Disadvantages:*

* *In order to manage these spirals, highly qualified managerial personnel are needed. If management is poor, there is a chance that the spiral will continue forever.*
* *For small projects, this model is not appropriate.*
* *Due to the complexity of this process, there is an increased risk of error.*

1. *V-shaped SDLC Model - A variation on the traditional waterfall approach, the V-shaped SDLC model is based on a separate test stage for each development stage. This is an extremely rigid model where the subsequent step is only begun after the preceding one. This model is also referred to as "validation and verification."*

**

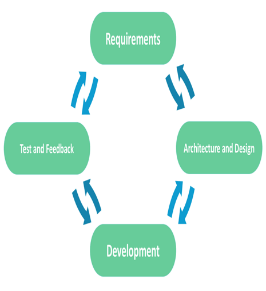
*Advantages:*

* *The V-shaped model's stages each produce exact outcomes, making it simple to regulate.*
* *The early steps involve testing and verification.*
* *Excellent for small projects with dependable and well-defined criteria.*

*Disadvantages:*

* *Unwillingness to bend.*
* *For little jobs, this is a bad choice.*
* *Large hazards, relatively.*

1. *Agile SDLC Model - The customer can view the outcome of each development iteration in the agile process and determine whether he is satisfied with it or not. It is challenging to predict the resources and development costs in the absence of well-specified needs.*



*Advantages:*

* *Changes to functional requirements are incorporated into the development process.*
* *Each iteration of the project is brief and open.*
* *The flexible transition process reduces risks.*
* *Rapid release of the initial product version.*

*Disadvantages:*

* *Permanent alterations make it difficult to calculate the total cost.*
* *The team should be extremely professional and client-focused.*
* *There could be conflicts between new requirements and the current architecture.*
* *There is a chance that the project will take longer than anticipated given all the repairs and revisions.*