**SDLC Models**

1. Waterfall Model:

Waterfall model is a linear-sequential life cycle model. It flows from higher end of stages to lower end of stages. It involves stages named as requirement analysis, design, development, testing, deployment and maintenance.

Advantages:

It is simple to understand and easy to use for small projects. Milestones of the project can be well understood. The phases are processed and completed once at a time.

Disadvantages:

We cannot go back to the previous stage once move forward. Testing comes at the very last stage. The progress of a project is difficult to measure within stages. The changes in project cannot be done afterwards.

1. V Model:

V model has a V like structure. It is also known as Verification and Validation Model. It has SDLC phases named as requirement analysis, high level design, low level design and coding. The STLC (Software Testing Life Cycle) phases named as unit testing, integration testing, system testing and user acceptance testing.

Advantages:

It is easy to use and simple to understand. Testing activities like planning, test designing happens well before coding. It is time saving and quick. Hence higher chance of success over the waterfall model. It has proactive defect tracking with which defects are found at early stage. It avoids the downward flow of the defects. It works well for small projects where requirements are easily understood.

Disadvantages:

It is very rigid and least flexible. Software is developed during the implementation phase, so no early prototypes of the software are produced. If any changes happen in midway, then the test documents along with requirement documents has to be updated hence it is risky.

1. Prototype Model:

In this model, a prototype of the end product is first developed, tested and refined as per customer feedback repeatedly till a final acceptable prototype is achieved which forms the basis for developing the final product.

Advantages:

This model is flexible in design. It is easy to detect errors. We can find missing functionality easily. There is scope of refinement, it means new requirements can be easily accommodated. It can be reused by the developer for more complicated projects in the future. It ensures a greater level of customer satisfaction and comfort. It is ideal for online system. It helps developers and users both understand the system better. Integration requirements are very well understood and deployment channels are decided at a very early stage. It can actively involve users in the development phase.

Disadvantages:

This model is costly. It has poor documentation because of continuously changing customer requirements. There may be too much variation in requirements. Customers sometimes demand the actual product to be delivered soon after seeing an early prototype. There may be sub-optimal solutions because of developers in a hurry to build prototypes. Customers may not be satisfied or interested in the product after seeing the initial prototype. There is certainty in determining the number of iterations. There may be incomplete or inadequate problem analysis. There may increase the complexity of the system.

1. Spiral Model:

It is also known as Meta Model because it subsumes all the other SDLC models. In its diagrammatic representation, it looks like a spiral with many loops, that’s the reason it’s called as Spiral. Each loop of the spiral is called a Phase of the software development process. This model has capability to handle risks.

Advantages:

Software is produced early in the software life cycle. Risk handling is one of important advantages of the Spiral model, it is best development model to follow due to the risk analysis and risk handling at every phase. Flexibility in requirements. In this model, we can easily change requirements at later phases and can be incorporated accurately. Also, additional Functionality can be added at a later date. It is good for large and complex projects. It is good for customer satisfaction. We can involve customers in the development of products at early phase of the software development. Also, software is produced early in the software life cycle. Strong approval and documentation control. It is suitable for high risk projects, where business needs may be unstable. A highly customized product can be developed using this.

Disadvantages:

It is not suitable for small projects as it is expensive. It is much more complex than other SDLC models. Process is complex. Too much dependable on Risk Analysis and requires highly specific expertise. Difficulty in time management. As the number of phases is unknown at the start of the project, so time estimation is very difficult. Spiral may go on indefinitely. End of the project may not be known early. It is not suitable for low risk projects. May be hard to define objective, verifiable milestones. Large numbers of intermediate stages require excessive documentation.

1. Agile Model:

The meaning of Agile is swift or versatile. It refers to a software development approach based on iterative development. Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning. The project scope and requirements are laid down at the beginning of the development process. Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance.

Advantages:

Customer satisfaction by rapid, continuous delivery of useful software. People and interactions are emphasized rather than process and tools. Customers, developers and testers constantly interact with each other. Working software is delivered frequently (weeks rather than months). Face-to-face conversation is the best form of communication. Close, daily cooperation between business people and developers. Continuous attention to technical excellence and good design. Regular adaptation to changing circumstances. Even late changes in requirements are welcomed

Disadvantages:

In case of some software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the software development life cycle. There is lack of emphasis on necessary designing and documentation. The project can easily get taken off track if the customer representative is not clear what final outcome that they want. Only senior programmers are capable of taking the kind of decisions required during the development process. Hence it has no place for newbie programmers, unless combined with experienced resources.