**SOFTWARE PROCESS MODELS**

A software process model is an abstraction of the software development process.

The goal of a software process model is to provide guidance for controlling and coordinating the tasks to achieve the end product and objectives as effectively as possible.

A model will define the following:

* The tasks to be performed
* The input and output of each task
* The pre and post conditions for each task
* The flow and sequence of each task

The Software Development Life Cycle is a process utilized by the Software Development Industry for designing, developing, and testing software solutions. The objective of this process is to develop high-quality software solutions that exceed or meet user expectations within stipulated time frames.

There are many kinds of process models for meeting different requirements. We refer to these as SDLC models (Software Development Life Cycle models). The most popular and important SDLC models are as follows:

**WATERFALL MODEL**

The Waterfall model represents a breakdown of software project activities in a linear sequential phase

The Waterfall model is the first software process model that was introduced. This model is extremely easy to understand by the developers.

Each phase of this model is completed before the next phase to avoid the overlapping among the multiple phases. The Waterfall model illustrates the software process in a linear sequential flow, which means that any face of the whole process begins only if the previous phase is completed.

It has the following phases:

1. Requirements
2. Design
3. Implementation
4. Testing
5. Deployment
6. Maintenance

**Advantages of the waterfall model:**

* The stages consist of well-defined tasks which promotes good scheduling and cost estimation (if all stages occur in the expected sequence once only).
* The deliverables provide targets to see how far a team has reached in the development process.
* The life cycle is broken into well-defined stages
* At any one time the project team knows what should be happening and the deliverable(s) they are to produce.

**However, there are also a number of major limitations of the waterfall model, which occur frequently in software development:**

* It is rare that a software development project will follow the sequential process that the waterfall model uses.
* Although the requirements are specified early on, user understanding and feedback of the software will not occur until after the system is implemented, which is possibly too late (or very costly) to change.
* The user may not be able to describe the requirements of the desired system in any detail early on.
* The model does not easily allow for the anticipation of change
* If a problem is identified at a later stage, the model does not make it easy (or cheap) to return to an earlier stage to rectify the mistake

**AGILE MODEL**

The agile process model encourages continuous iterations of development and testing. Each incremental part is developed over an iteration, and each iteration is designed to be small and manageable so it can be completed within a few weeks.

Each iteration focuses on implementing a small set of features completely. It involves customers in the development process and minimizes documentation by using informal communication.

**Agile development considers the following:**

* Requirements are assumed to change
* The system evolves over a series of short iterations
* Customers are involved during each iteration
* Documentation is done only when needed

Though agile provides a very realistic approach to software development, it isn’t great for complex projects. It can also present challenges during transfers as there is very little documentation. Agile is great for projects with changing requirements.

**Some commonly used agile methodologies include:**

**Scrum:** One of the most popular agile models, Scrum consists of iterations called sprints. Each sprint is between 2 to 4 weeks long and is preceded by planning. You cannot make changes after the sprint activities have been defined.

**Extreme Programming (XP):** With Extreme Programming, an iteration can last between 1 to 2 weeks. XP uses pair programming, continuous integration, test-driven development and test automation, small releases, and simple software design.

**Kanban:** Kanban focuses on visualizations, and if any iterations are used, they are kept very short. You use the Kanban Board that has a clear representation of all project activities and their numbers, responsible people, and progress.

**Advantages of Agile Method:**

1. Frequent Delivery
2. Face-to-Face Communication with clients.
3. Efficient design and fulfils the business requirement.
4. Anytime changes are acceptable.
5. It reduces total development time.

**Disadvantages of Agile Model:**

1. Due to the shortage of formal documents, it creates confusion and crucial decisions taken throughout various phases can be misinterpreted at any time by different team members.
2. Due to the lack of proper documentation, once the project completes and the developers allotted to another project, maintenance of the finished project can become a difficulty.

**V MODEL**

The V model for software process represents a development methodology that can be considered as an extension of the Waterfall software model. In this process, instead of moving down in a unique and linear manner, the steps of the process are bent upwards soon after the coding phase, to develop a typical V shape.

This model represents the relationship between each phase of the software development life cycle, along with the associated phases of software testing. The horizontal and vertical axis of this model represents time/project completeness and level of abstraction, respectively.

**Advantages:**

The V model is highly disciplined.

Easy to understand, and makes project management easier.

**Disadvantage:**

It isn’t good for complex projects or projects that have unclear or changing requirements.

**RAD MODEL**

The Rapid Application Development (RAD model) is based on iterative development and prototyping with little planning involved. You develop functional modules in parallel for faster product delivery. It involves the following phases:

* Business modeling
* Data modeling
* Process modeling
* Application generation
* Testing and turnover

The RAD concept focuses on gathering requirements using focus groups and workshops, reusing software components, and informal communication.

**Advantage of RAD Model**

* This model is flexible for change.
* In this model, changes are adoptable.
* Each phase in RAD brings highest priority functionality to the customer.
* It reduced development time.
* It increases the reusability of features.

**Disadvantage of RAD Model**

* It required highly skilled designers.
* All application is not compatible with RAD.
* For smaller projects, we cannot use the RAD model.
* On the high technical risk, it's not suitable.
* Required user involvement.
* RAD may also not be appropriate when technical risks are high.

**Boehm's spiral model**

The spiral model was published by Barry Boehm in 1986. It provides an iterative, evolutionary approach to software development combined with the step-by-step aspects of the waterfall process model and the requirements analysis abilities of prototyping. It is intended for development of large, complicated software projects.

The Spiral model refers to a test-driven software development model that was introduced for superimposing the shortcomings present in a conventional Waterfall model. The Spiral model looks exactly like a spiral having multiple loops. The exact number of spiral loops is unknown and they can differ from project to project. The Spiral model facilitates risk handling management, and the final software project is delivered in the form of loops.

Each loop of the Spiral model is known as the phase of the whole software development process. The initial phase of the Spiral model in its early development stages of the Waterfall life cycle is required to develop the final software product. The total number of faces required to develop the software can differ from project managers and depends on associated risks.

The advantages of this model are:

* The spiral model considers the entire software life-cycle.
* Because of its iterative approach, it is adaptable, and appropriate for large-scale projects.

However, the model does have disadvantages:

* It requires expertise at assessing and managing risk.
* It may be difficult to convince customers that such an evolutionary approach is necessary.

**Prototyping life cycle model**

A prototype system is a smaller version of part(s) of the final system that gives the user a sense of the finished system's functionality. It has some of the core features of the final system and, where features and functions are omitted, it pretends to behave like the final system. Prototypes are typically developed quickly, may lack unnecessary features, may be buggy, and have poor usability. However, prototypes can fill an important role in understanding software which does not have clear requirements.

Where the system to be developed is a truly new system, there may be no clear requirements defining the software's behaviors. By building a prototype, both the developers and users have some real, visible working system model on which to focus their ideas. An analysis of this prototype forms the basis for the requirements specification, and perhaps even some of the design. If there is still uncertainty of the new system and questions still remain, further prototypes can be developed (or an existing prototype extended). In this way, prototyping allows developers and customers to better understand incomplete and fuzzy software requirements.

To prototype quickly and effectively, fourth generation languages (4GLs), graphical user-interface (GUI) tools (like those that come with Visual Studio, QT and GTK), and off-the-shelf components are commonly used. The quality of the prototype is only of concern where it would hinder the prototype's use in understanding the final software being developed. If the prototype is usable enough to meet the objectives put forward for its development, the prototype has been successful.

**Advantage of Prototype Model**

1. Reduce the risk of incorrect user requirement
2. Good where requirements are changing/uncommitted
3. Regular visible process aids management
4. Support early product marketing
5. Reduce Maintenance cost.
6. Errors can be detected much earlier as the system is made side by side.

**Disadvantage of Prototype Model**

1. An unstable/badly implemented prototype often becomes the final product.
2. Difficult to know how long the project will last.
3. Easy to fall back into the code and fix without proper requirement analysis, design, customer evaluation, and feedback.
4. Prototyping tools are expensive.
5. Special tools & techniques are required to build a prototype.
6. It is a time-consuming process.