A software process model is an abstraction of the actual process which is being described and is a simplified representation of a software process.[[1]](#footnote-1)

Each model treats activities differently, and each model is suitable for different projects and for different teams.

It is important to realize that the activities outlined in the process models should be modified based on:

* The problem having to be solved
* The characteristics of the project
* The nature of the development team
* The organizational culture

Some of the different software process models along with their merits and demerits include;

**The waterfall life cycle model**

This model is also known as the traditional software life cycle as it was the first and for a time the only process model.

Features of the waterfall model are as follows:

* The system development process is broken into distinct stages
* Each stage involves a particular project activity such as communication or construction
* Each stage when completed, results in a deliverable or product
* The input to any particular stage is the deliverables from the previous stage
* The model presents the stages in a strict, one-way sequence — a project cannot go back to repeat a stage once that stage has been completed. Any required re-work (as a result, for example, of changing software requirements) is very limited.

Communication -> Planning -> Modeling -> Construction -> Deployment

Advantages of 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 or milestones to see how far a team has reached in the development process.
* The life cycle is broken into well-defined stages — so staff expertise can be used efficiently (e.g., a data modeler only needs to work on certain stages, a programmer only on other stages, and so on).
* At any one time the project team knows what should be happening and the deliverable they are to produce.

Limitations of Waterfall Model

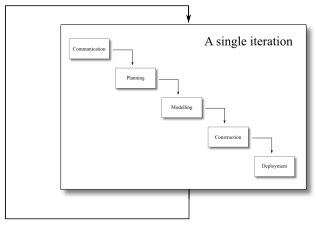
* It is rare that a software development project will follow the sequential process that the waterfall model uses.
* Although the requirements are specified early on, user understanding and feedback of the software will not occur until after the system is implemented, which is possibly too late or very 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 — some systems take years to develop, but once the early stages have been completed the model commits the project to a fixed specification of the system.
* Many projects based on the waterfall model stress the importance of certain products documents being delivered at certain times — it is possible for a project to become managed in a bureaucratic way, with documents being delivered on schedule, but the focus drifting away from developing a usable, effective system for the users.
* If a problem is identified at a later stage, the model does not make it easy to return to an earlier stage to rectify the mistake since all intermediate steps will need to be repeated, resulting in significant, unplanned, time and resource costs.

**Incremental process model**

Incremental process models provide limited functionality early in the software's lifecycle.

This functionality is then expanded on in later releases.

Incremental development software model: attempts to maintain some of the advantages of the pure waterfall model, but attempt to allow for greater change management and overall flexibility in the software process.

****

Software development occurs in small increments allowing this model to handle far better than waterfall method.

The incremental model allows the developers to quickly release a version of the software with limited functionality, and then at each development iteration to add additional, incremental functionality. The development in each iteration occurs in a linear method, as with the waterfall model. Ideally, the most important functions are implemented first and successive stages add new functionality in order of priority.

By doing this, the full development task is broken down into smaller, more manageable portions, allowing implementation problems to be highlighted before the full system is completed.

Incremental delivery is this process of releasing the product to the client at the end of each iteration. This allows the client to use regular, updated versions of the software, giving them the capability to judge the progress of the software development.

Advantages of Incremental development

* The process is more responsive to changing user requirements than a waterfall approach — later subsystems can be re-specified. Also, a modular approach can mean maintenance changes are simpler and less expensive.
* There is an opportunity for incremental delivery to users, so the users can benefit from parts of the system development without having to wait for the entire life cycle to run its course.
* Incremental delivery means that users have a portion of the software to examine in order to see how well the software meets their needs, and whether the software requirements have to be modified.
* Complete project failure is less likely, since users will have some working sub-systems even if time and money run out before the complete system is delivered.
* The project can begin with fewer workers, as only a subset of the final product is being worked on.
* The risk associated with the development of the software can be better managed.
* The time taken to develop previous iterations can be used as an estimate for the time needed to develop the remaining iterations, and hence improve project planning.

Disadvantages of Incremental Development

* This development model relies on close interaction with the users — if they are not easily available or slow in evaluating each iteration, the whole process can slow down.
* The reliance on user involvement can exacerbate the already difficult task of estimating the amount of time and budget required.
* High user involvement means that resources are drawn away from the customer's normal operation during system development.

**Rapid Application Development process model (RAD)**

Rapid Application Development is an incremental process model that has a focus on short development cycles hence the term "rapid".

This speed is obtained by using off-the-shelf components, and a component-based design and implementation approach.

Advantages of RAD

* Development cycles are rapid, typically between 60 to 90 days.

Disadvantages

* For large projects, RAD may require a large number of people to split the project into a sufficient number of teams.
* The developers and the customers must be committed to the necessary activities in order for the process to succeed.
* The project must be suitably modularized in order for RAD to be successful.
* RAD my not be appropriate where high-performance is necessary.
* RAD may also not be appropriate when technical risks are high.

**Prototyping life cycle model**

A prototype system is a smaller version of part 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 behavior. 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.

Advantages of Prototyping

* Users get an early idea of the final system features.
* The prototype provides an opportunity to identify problems early and to change the requirements appropriately.
* The prototype is a model that all users and customers should be able to understand and provide feedback on, thus the prototype can be an important tool to improve communication between users and developers.
* It may be possible to use aspects of the prototype specification and design in the final system specification and design, thus some of the prototype development resources can be recouped.

Disadvantage of Prototyping

* A major problem with developing disposable prototypes is that the customer may believe it to be the final product. Customers may not understand the need to re-engineer the software and restart development, and may ask that the prototype be cleaned up and released to them.

**Boehm’s Spiral model**

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.

This process model provides for the rapid development of progressively more complete versions of the software. Each iteration of the evolutionary development will have a release, which may merely be a paper model of the software, a prototype, or an early iteration of the software.

Each iteration of the spiral model contains all of the activities from the generic process framework outlined above: communication, planning, modelling, construction and deployment. One can consider an iteration to be an arc in a spiral: each arc contains the same breakdown of how the development is approached, but each arc will focus on something new

Advantages are:

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

Disadvantages:

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

**Agile method**

The overall goal of agile development is to satisfy the customer by early and continuous delivery of valuable software. Many customers have business needs that change over time, reflecting not only newly discovered needs but also the need to respond to changes in the marketplace[[2]](#footnote-2).

General idea about agile is that keeping it flexible in the development process ensures constant availability to change and adapt.

The key features of an agile process model can be as follows:

* The software itself is the important measure of the team's progress, rather than documentation.
* The development team has autonomy to determine how to structure itself, handle the development work, and apply the process model.
* Adaptability to change comes in large part through delivering software incrementally.
* Adaptability also comes from frequent delivery, so that customers can more easily examine the software and provide feedback.
* The process is tolerant: it is adapted to the development team's needs.
* Software is important, documentation less so: this means that design and construction are often heavily interleaved

Examples of agile processes in current literature and each is laid out on a set of principles that implement the core of the agile manifesto

Extreme programming (XP) – A set of techniques for leveraging the creativity of developers and minimizing the amount of administrative overhead. Extreme programming is an object-oriented development approach and provides four framework activities: planning, design, coding and testing.

Extreme programming has the following advantages:

* Extreme programming is an incremental process model, and so the customer will have working software very early. Process and Model 17
* The customer works closely with the developers, so the developers have a better understanding of the software requirements.
* Pair programming allows for quality checks of code as programming happens.
* Extreme programming has a strong focus on accepting changing project requirements.

Disadvantages

* Much time might be spent re-coding the software, rather than focusing initially on a better design

Scrum – It uses iterative development, constructed of sprints 2 week – 4-week duration, to implement the product’s backlog of prioritized requirements. Multiple self-organizing and autonomous teams implement product increments in parallel. Coordination is done at a brief daily status meeting called a “scrum”.[[3]](#footnote-3)

Lean - 7 principles: Eliminate waste, amplify learning, decide as late as possible, deliver as fast as possible, Empower the team, build integrity, See the whole.[[4]](#footnote-4)

Kanban - Work items are visualized to give participants a view of progress and process, from start to finish – usually via a Kanban board. The aim is to provide a visual process management system which helps decision-making about what, when and how much to produce. Manages workflow.[[5]](#footnote-5)

**HISTORY OF SOFTWARE CONSTRUCTION**

Software construction refers to the detailed creation of working software through a combination of coding, verification, unit testing, integration testing, and debugging.

The late 1950s is considered as an essential period of the era of computing. Large computers became available to research institutions and universities. Their presence was noticed mainly in engineering and natural sciences, but also in business they soon became indispensable. The time when they were accessible only to a few insiders in laboratories, when they broke down every time one wanted to use them, belonged to the past. Their emergence from the closed laboratory of electrical engineers into the public domain meant that their use, in particular their programming, became an activity of many. A new profession was born; but the large computers themselves became hidden within closely guarded cellars. Programmers brought their programs to the counter, where a dispatcher would pick them up, queue them, and where the results could be fetched hours or days later. There was no interactivity between man and computer.

In order to facilitate this coding, formal notations were created. We now call them programming languages. The primary idea was to replace sequences of special instruction code by mathematical formulas. The first widely known language, Fortran, was issued by IBM (Backus, 1957), soon followed by Algol (1958) and its official successor in 1960. As computers were then used for computing rather than storing and communicating, these languages catered mainly to numerical mathematics. In 1962 the language Cobol was issued by the US Department of Defense for business applications.[[6]](#footnote-6)

1. https://www.geeksforgeeks.org/software-processes-in-software-engineering/ [↑](#footnote-ref-1)
2. https://agilemanifesto.org/iso/en/principles.html [↑](#footnote-ref-2)
3. https://www.educative.io/blog/software-process-model-types [↑](#footnote-ref-3)
4. Poppendieck, M., Poppendieck, T., (2003): Lean Software Development: An Agile Toolkit. Addison-Wesley Professional [↑](#footnote-ref-4)
5. Gross, J. M., McInnis, K. R., (2003): Kanban Made Simple: Demystifying and Applying Toyota's Legendary Manufacturing Process. AMACOM [↑](#footnote-ref-5)
6. <https://people.inf.ethz.ch/wirth/Miscellaneous/IEEE-Annals.pdf> [↑](#footnote-ref-6)