Contrast and Compare different Software Development Model

The software development models are the various processes or methodologies that are being selected for the development of the software project depending on the project's aims and goals. There are many development life cycle models that have been developed in order to achieve different required objectives. The models specify the various stages of the process and the order in which they are carried out. The selection of model has very high impact on the outcome of the project. There are a number of models for such processes, each describing approaches to a range of tasks or activities that take place during the process. Therefore, it may be required to choose the right SDLC model according to the specific concerns and requirements of the project to ensure its success.

Below are the few popularly used software development models

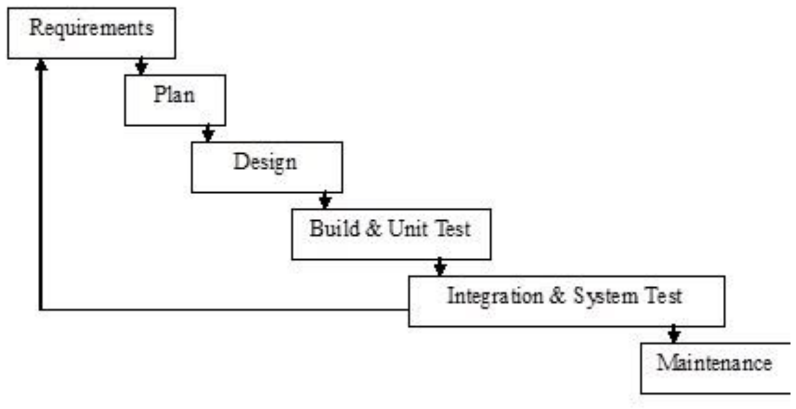
- Waterfall model
- Iterative model
- V- model
- Spiral model
- Extreme programming
- Incremental Method
- Rapid prototyping model/Rapid Application Development Model
- The Chaos Model
- Adaptive Software Development (ASD)
- The Agile Software Process (ASP)
- Dynamic System Development Method (DSDM)
- Feature Driven Development (FDD)
- Rational Unified Process (RUP)
- SCRUM
- Kanban

Waterfall Model

The waterfall model [1] is the classical model of software engineering. This model is one of the oldest models and is widely used in government projects and in many major companies. As this model emphasizes planning in early stages, it ensures design flaws before they develop. In addition, its intensive document and planning make it work well for projects in which quality control is a major concern. The model begins with establishing system requirements and software requirements and continues with architectural design, detailed design, coding, testing, and maintenance.

Iterative model

The Iterative model can be thought of as a "multi-waterfall" cycle. Cycles are divided into smaller and easily managed iterations. Each iteration passes through a series of phases, so after each cycle you will get working software.



V- Model

'V-model' is also used by many of the companies in their product. 'V-model' is nothing but 'Verification' and 'Validation' model. In 'V-model' the developer's life cycle and tester's life cycle are mapped to each other. In this model testing is done side by side of the development.

Spiral model

It is combining elements of both design and prototyping-in-stages, in an effort to combine advantages of top-down and bottom-up concepts. This model of development combines the features of the prototyping model and the waterfall model. The spiral model is favored for large, expensive, and complicated projects. This model uses many of the same phases as the waterfall model, in essentially the same order, separated by planning, risk assessment, and the building of prototypes and simulations.

Extreme programming

An approach to development based on the development and delivery of very small increments of functionality. It relies on constant code improvement, user involvement in the development team and pair wise programming. It can be difficult to keep the interest of customers who are involved in the process. Team members may be unsuited to the intense involvement that characterizes agile methods. Prioritizing changes can be difficult where there are multiple stakeholders. Maintaining simplicity requires extra work. Contracts may be a problem as with other approaches to iterative development

Incremental Method

In incremental model the whole requirement is divided into various builds. Multiple development cycles take place here, making the life cycle a "multi-waterfall" cycle. Cycles are divided up into smaller, more easily managed modules. New software modules are added in each iteration with no or little change in earlier added modules. The development process can go either sequentially or in parallel. Parallel development adds to the speed of delivery, while many repeated cycles of sequential development can make the project long and costly.

Rapid prototyping model/Rapid application development

A rapid prototype is a working model that is functionally equivalent to a subset of the product. Because the working prototype has been validated through interaction with the client, the resulting specification will be correct. Verification is needed in specification, planning, and design. In implementation and integration, testing is needed. An essential aspect of a rapid prototype is in the word rapid. We can combine waterfall and rapid prototyping, by using rapid prototyping to find out the client's requirements.

Chaos Model

The Chaos model [8] combines a linear problem solving loop with fractals to describe the complexity of software development. The linear problem-solving loop involves four different stages: problem definition, technical development, solution integration, and status quo. Fractals describe the structure between different parts of a project. The Chaos model differs from other models in that it imposes little organization on the development process; rather, it allows many organizations to evolve. This allows the Chaos model to apply in many complex situations. The structure of a simple problem is different from the structure of a more complex problem.

Adaptive Software Development (ASD)

Adaptive Software Development (ASD) as a framework from which to address the rapid pace of many software projects. ASD is grounded in the science of complex adaptive systems theory and has three interwoven components: the Adaptive Conceptual Model, the Adaptive Development Model, and the Adaptive Management Model. In contrast to the typical waterfall (plan, build, implement) or the iterative (plan, build, revise) life cycles, the adaptive development life cycle (speculate, collaborate, learn) acknowledges the existence of uncertainty, change and does not attempt to manage software development using precise prediction and rigid control strategies.

The Agile Software Process (ASP)

It is based on iterative and incremental development, where requirements and solutions evolve through collaboration between cross-functional teams. It can be used with any type of the project, but it needs more engagement from the customer and to be interactive. Also, we can use it when the customer needs to have some functional requirement ready in less than three weeks and the requirements are not clear enough. This will enable more valuable and workable piece for software early which also increase the customer satisfaction.

Dynamic System Development Method (DSDM)

The Dynamic Systems Development Method (DSDM) is a framework [12] used to control software development projects with short timelines. It was developed in 1994 by a consortium formed by a group companies in Great Britain. The methodology begins with a feasibility study and business study to determine if DSDM is appropriate. The rest of the process consists of three interwoven cycles. These are functional model iteration, design and build iteration, and implementation. The underlying principles of DSDM include frequent deliveries, active user communication, empowered development teams, and testing in all phases of a project. DSDM is different than traditional approaches in that requirements are not fixed. Project requirements are allowed to change based upon a fixed timeline and fixed project resources. This approach requires a clear prioritization of functional requirements. Emphasis is also put on high quality and adapting to changing requirements. It has the advantage of a solid infrastructure (similar to traditional methodologies), while following the principles of lightweight SDLC methods.

Feature Driven Development (FDD)

Feature Driven Development (FDD) is a model-driven short-iteration software development process. The FDD process starts by establishing an overall model shape. This is followed by a series of two-week “design by feature, build by feature” iterations. FDD consists of five processes: develop an overall model, build a features list, plan by feature, and design by feature, and build by feature. There are two types of developers on FDD projects: chief programmers and class owners. The chief programmers are the most experienced developers and act as coordinator, lead designer, and mentor. The class owners do the coding. One benefit of the simplicity of the FDD process is the easy introduction of new staff. FDD shortens learning curves and reduces the time it takes to become efficient. Finally, the FDD methodology produces frequent and tangible results. The method uses small blocks of user-valued functionality. In addition, FDD includes planning strategies and provides precision progress tracking.

Rational Unified Process (RUP)

The Rational Unified Process (RUP) is also a combination of linear and iterative frameworks. The model divides the software development process into 4 phases – inception, elaboration, construction, and transition. Each phase but Inception is usually done in several iterations. All basic activities (requirements, design, etc.) of the development process are done in parallel across these 4 RUP phases, though with different intensity. RUP helps to build stable and, at the same time, flexible solutions, but still, this model is not as quick and adaptable as the pure Agile group (Scrum, Kanban, XP, etc.). The degree of customer involvement, documentation intensity, and iteration length may vary depending on the project needs.

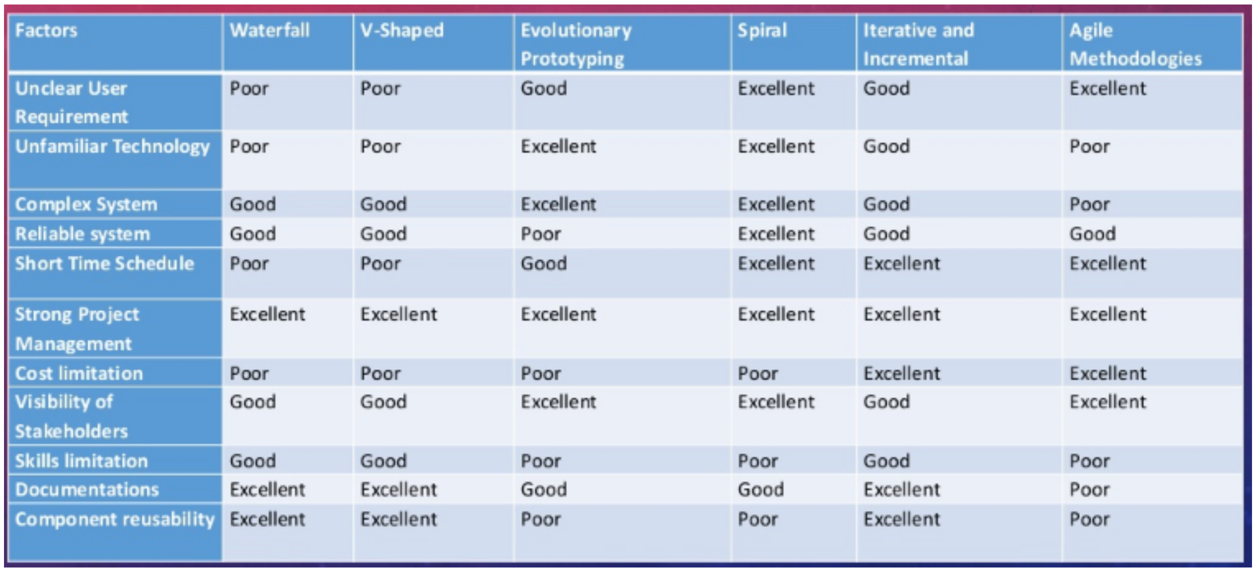
SCRUM

Scrum is probably the most popular Agile model. The iterations (‘sprints’) are usually 2-4 weeks long and they are preceded with thorough planning and previous sprint assessment. No changes are allowed after the sprint activities have been defined.

Kanban

As for Kanban, its key distinguishing feature is the absence of pronounced iterations. If used, they are kept extremely short (‘daily sprints’). Instead, the emphasis is placed on plan visualization. The team uses the Kanban Board tool that provides a clear representation of all project activities, their number, responsible persons, and progress. Such increased transparency helps to estimate the most urgent tasks more accurately. Also, the model has no separate planning stage, so a new change request can be introduced at any time. Communication with the customer is ongoing, they can check the work results whenever they like, and the meetings with the project team can happen even daily. Due to its nature, the model is frequently used in projects on software support and evolution.

Comparisons



Factors	Waterfall	V-Shaped	Evolutionary Prototyping	Spiral	Iterative and Incremental	Agile Methodologies
Unclear User Requirement	Poor	Poor	Good	Excellent	Good	Excellent
Unfamiliar Technology	Poor	Poor	Excellent	Excellent	Good	Poor
Complex System	Good	Good	Excellent	Excellent	Good	Poor
Reliable system	Good	Good	Poor	Excellent	Good	Good
Short Time Schedule	Poor	Poor	Good	Excellent	Excellent	Excellent
Strong Project Management	Excellent	Excellent	Excellent	Excellent	Excellent	Excellent
Cost limitation	Poor	Poor	Poor	Poor	Excellent	Excellent
Visibility of Stakeholders	Good	Good	Excellent	Excellent	Good	Excellent
Skills limitation	Good	Good	Poor	Poor	Good	Poor
Documentations	Excellent	Excellent	Good	Good	Excellent	Poor
Component reusability	Excellent	Excellent	Poor	Poor	Excellent	Poor