**POPULAR SDLC MODEL: AGILE**

**Agile Model**

Agile methodology is a practice which promotes continue interaction of development and testing during the SDLC process of any project. In the Agile method, the entire project is divided into small incremental builds. All of these builds are provided in iterations, and each iteration lasts from one to three weeks. The Manifesto for Agile Software Development was drafted and signed by a group of software developers in 2001. Reading the manifesto, you can see clearly the contrast between Waterfall, then the de-facto standard for development methods, and Agile, the newer method.

The Manifesto addresses key problems with Waterfall that led to challenges in software delivery. Where Waterfall tends to be a “one-way road,” Agile is a more flexible framework that allows for uncertainty. Agile emphasizes teamwork, prototyping, and feedback loops that can change the direction of the development effort in response to changing requirements. Several variants of Agile have emerged since the signing of the Manifesto. Scrum defines specific roles and events, known as ceremonies, as part of its practice. Kanban is simpler, with fewer prescriptions and more flexibility. Agile teams often combine these together to adapt a bespoke process that fits them best.



The most popular Agile methods include Rational Unified Process (1994), Scrum (1995), Crystal Clear, Extreme Programming (1996), Adaptive Software Development, Feature Driven Development, and Dynamic Systems Development Method (DSDM) (1995). These are now collectively referred to as **Agile Methodologies**, after the Agile Manifesto was published in 2001.

**Following are the Agile Manifesto principles**

**Individuals and interactions** − In Agile development, self-organization and motivation are important, as are interactions like co-location and pair programming.

**Working software** − Demo working software is considered the best means of communication with the customers to understand their requirements, instead of just depending on documentation.

**Customer collaboration** − As the requirements cannot be gathered completely in the beginning of the project due to various factors, continuous customer interaction is very important to get proper product requirements.

**Responding to change** − Agile Development is focused on quick responses to change and continuous development.

**Agile Model - Pros and Cons**

Agile methods are being widely accepted in the software world recently. However, this method may not always be suitable for all products. Here are some pros and cons of the Agile model.

**The advantages of the Agile Model are as follows**

* Is a very realistic approach to software development.
* Promotes teamwork and cross training.
* Functionality can be developed rapidly and demonstrated.
* Resource requirements are minimum.
* Suitable for fixed or changing requirements
* Delivers early partial working solutions.
* Good model for environments that change steadily.
* Minimal rules, documentation easily employed.
* Enables concurrent development and delivery within an overall planned context.
* Little or no planning required.
* Easy to manage.
* Gives flexibility to developers.

**The disadvantages of the Agile Model are as follows**

* Not suitable for handling complex dependencies.
* More risk of sustainability, maintainability and extensibility.
* An overall plan, an agile leader and agile PM practice is a must without which it will not work.
* Strict delivery management dictates the scope, functionality to be delivered, and adjustments to meet the deadlines.
* Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction.
* There is a very high individual dependency, since there is minimum documentation generated.
* Transfer of technology to new team members may be quite challenging due to lack of documentation.

Comparison of Various SDLC Models

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Properties of Model** | **Water-Fall Model** | **Incremental Model** | **Spiral Model** | **Rad Model** |
| Planning in early stage | Yes | Yes | Yes | No |
| Returning to an earlier phase | No | Yes | Yes | Yes |
| Handle Large-Project | Not Appropriate | Not Appropriate | Appropriate | Not Appropriate |
| Detailed Documentation | Necessary | Yes, but not much | Yes | Limited |
| Cost | Low | Low | Expensive | Low |
| Requirement Specifications | Beginning | Beginning | Beginning | Time boxed release |
| Flexibility to change | Difficult | Easy | Easy | Easy |
| User Involvement | Only at beginning | Intermediate | High | Only at the beginning |
| Maintenance | Least | Promotes Maintainability | Typical | Easily Maintained |
| Duration | Long | Very long | Long | Short |
| Risk Involvement | High | Low | Medium to high risk | Low |
| Framework Type | Linear | Linear + Iterative | Linear + Iterative | Linear |
| Testing | After completion of coding phase | After every iteration | At the end of the engineering phase | After completion of coding |
| Overlapping Phases | No | Yes (As parallel development is there) | No | Yes |
| Maintenance | Least Maintainable | Maintainable | Yes | Easily Maintainable |
| Re-usability | Least possible | To some extent | To some extent | Yes |
| Time-Frame | Very Long | Long | Long | Short |
| Working software availability | At the end of the life-cycle | At the end of every iteration | At the end of every iteration | At the end of the life cycle |
| Objective | High Assurance | Rapid Development | High Assurance | Rapid development |
| Team size | Large Team | Not Large Team | Large Team | Small Team |
| Customer control over administrator | Very Low | Yes | Yes | Yes |

**A quick summary on SDLC**

* The SDLC is a systematic process for building software that ensures the quality and correctness of the software built.
* The full form SDLC is Software Development Lifecycle.
* SDLC process provides a framework for a standard set of activities and deliverables.
* Seven different SDLC stages are 1) Requirement collection and analysis 2) Feasibility study: 3) Design 4) Coding 5) Testing: 6) Installation/Deployment and 7) Maintenance.
* The senior team members conduct the requirement analysis phase
* Feasibility Study stage includes everything which should be designed and developed during the project life cycle
* In the Design phase, the system and software design documents are prepared as per the requirement specification document.
* In the coding phase, developers start build the entire system by writing code using the chosen programming language.
* Testing is the next phase which is conducted to verify that the entire application works according to the customer requirement.
* Installation and deployment face begin when the software testing phase is over, and no bugs or errors left in the system.
* Bug fixing, upgrade, and engagement actions covered in the maintenance face.
* Waterfall, Incremental, Agile, V model, Spiral, Big Bang are some of the popular SDLC models.
* SDLC consists of a detailed plan which explains how to plan, build, and maintain specific software.