Agile SDLC model

What is Agile Methodology?

- AGILE methodology is a practice that promotes **continuous iteration** of development and testing throughout the software development lifecycle of the project. In the Agile model, both development and testing activities are concurrent, unlike the Waterfall model.
- Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product. Agile Methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. Every iteration involves cross functional teams working simultaneously on various areas like −

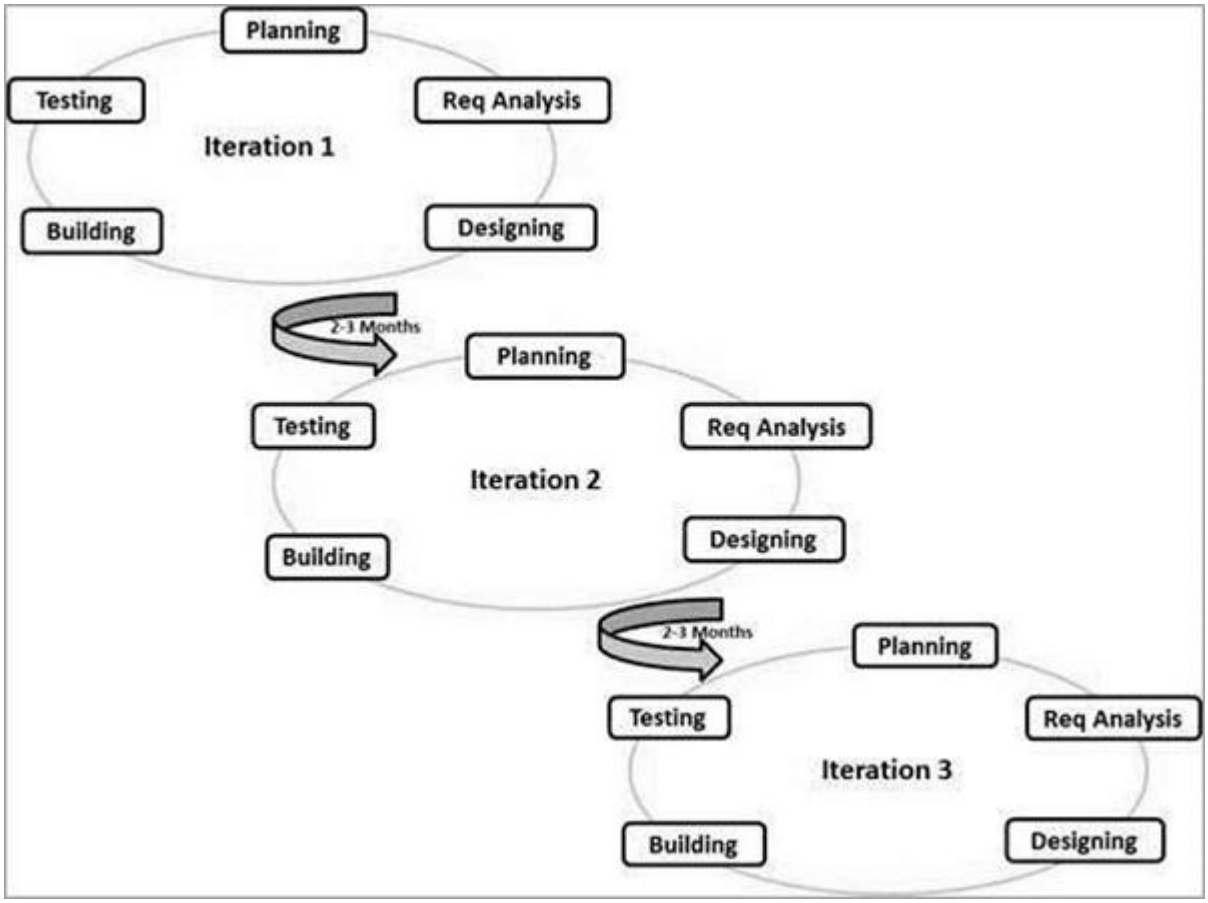
- ✓ Planning
- ✓ Requirements Analysis
- ✓ Design
- ✓ Coding
- ✓ Unit Testing and
- ✓ Acceptance Testing.

At the end of the iteration, a working product is displayed to the customer and important stakeholders.

What is Agile?

- Agile model believes that every project needs to be handled differently and the existing methods need to be tailored to best suit the project requirements. In Agile, the tasks are divided to time boxes (small time frames) to deliver specific features for a release.
- Iterative approach is taken and working software build is delivered after each iteration. Each build is incremental in terms of features; the final build holds all the features required by the customer.

Here is a graphical illustration of the Agile Model −



The Agile thought process had started early in the software development and started becoming popular with time due to its flexibility and adaptability.

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 Vs Traditional SDLC Models

Agile is based on the **adaptive software development methods**, whereas the traditional SDLC models like the waterfall model is based on a predictive approach. Predictive teams in the traditional SDLC models usually work with detailed planning and have a complete forecast of the exact tasks and features to be delivered in the next few months or during the product life cycle.

Predictive methods entirely depend on the **requirement analysis and planning** done in the beginning of cycle. Any changes to be incorporated go through a strict change control management and prioritization.

Agile uses an **adaptive approach** where there is no detailed planning and there is clarity on future tasks only in respect of what features need to be developed. There is feature driven development and the team adapts to the changing product requirements dynamically. The product is tested very frequently, through the release iterations, minimizing the risk of any major failures in future.

Agile Model	**Waterfall Model**
• Agile method proposes incremental and iterative approach to software design	• Development of the software flows sequentially from start point to end point.
• The **agile process** is broken into individual models that designers work on	• The design process is not broken into an individual models
• The customer has early and frequent opportunities to look at the product and make decision and changes to the project	• The customer can only see the product at the end of the project
• Agile model is considered unstructured compared to the waterfall model	• Waterfall model are more secure because they are so plan oriented
• Small projects can be implemented very quickly. For large projects, it is difficult to estimate the development time.	• All sorts of project can be estimated and completed.
• Error can be fixed in the middle of the project.	• Only at the end, the whole product is tested. If the requirement error is found

	or any changes have to be made, the project has to start from the beginning
• Development process is iterative, and the project is executed in short (2-4) weeks iterations. Planning is very less.	• The development process is phased, and the phase is much bigger than iteration. Every phase ends with the detailed description of the next phase.
• Documentation attends less priority than software development	• Documentation is a top priority and can even use for training staff and upgrade the software with another team
• Every iteration has its own testing phase. It allows implementing regression testing every time new functions or logic are released.	• Only after the development phase, the testing phase is executed because separate parts are not fully functional.
• In agile testing when an iteration end, shippable features of the product is delivered to the customer. New features are usable right after shipment. It is useful when you have good contact with customers.	• All features developed are delivered at once after the long implementation phase.
• Testers and developers work together	• Testers work separately from developers
• At the end of every sprint, user acceptance is performed	• User acceptance is **performed** at the end of the project.
• It requires close communication with developers and together analyze requirements and planning	• Developer does not involve in requirement and planning process. Usually, time delays between tests and coding

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.