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.

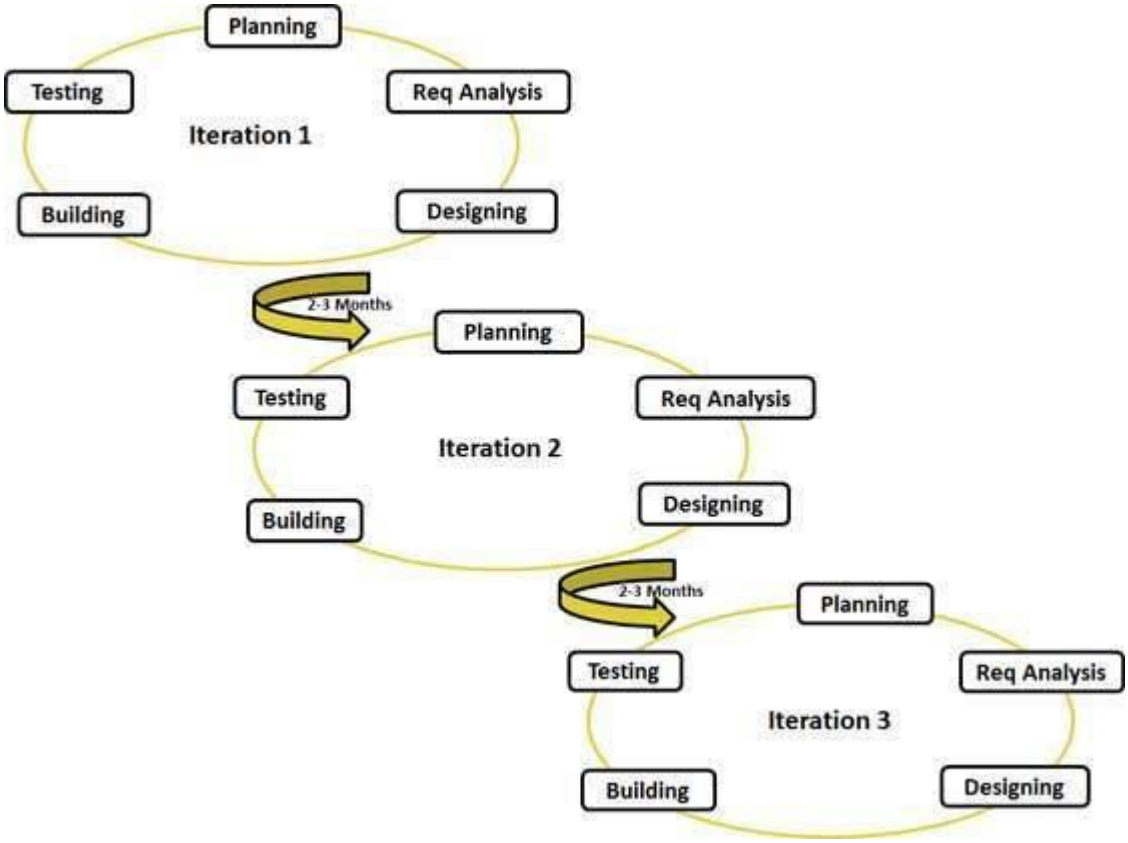
At the end of the iteration a working product is displayed to the customer and important stakeholders.

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:



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 customer to understand their requirement, 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 where as the traditional SDLC models like waterfall model is based on 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 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.

Customer interaction is the backbone of Agile methodology, and open communication with minimum documentation are the typical features of Agile development environment. The agile teams work in close collaboration with each other and are most often located in the same geographical location.

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.

Following table lists out the pros and cons of Agile Model:

Pros	Cons
• Is a very realistic approach to	• Not suitable for handling complex

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	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 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.