Lecture Notes: Minimum Viable Product

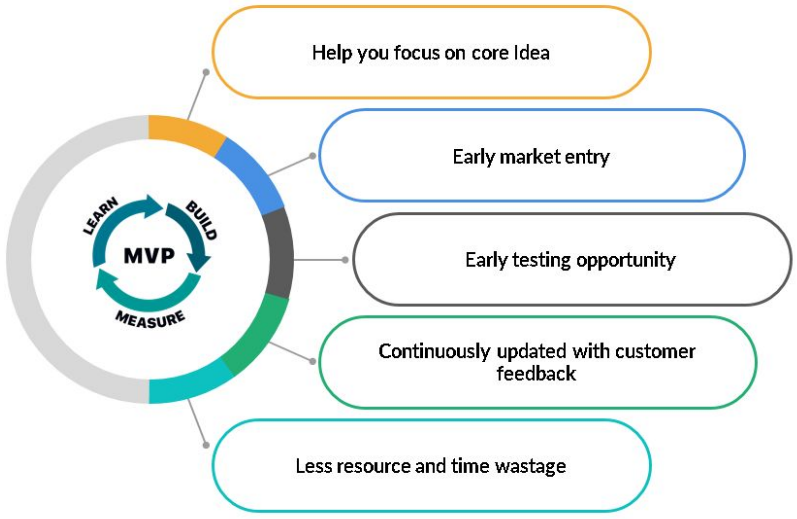
1.1 Introduction to Minimum Viable Product (MVP)

MVP is a product that is launched with features that are sufficient to implement the core idea. The aim of an MVP is to test your hypothesis in the actual market. It enables you to know whether the people would purchase your product or not.

An MVP helps you to reach the market early and learn continuously from the customers' feedback. It saves you from investing money in something that the users will not favour.

The MVPs are built iteratively according to the customers' requirement. Thus, incorporating customer feedback will help you develop a culture of product evangelists. The people who are fans of your product, eagerly wait for updates and promote the product without charging any fee.

Amazon, Facebook, Netflix and many other companies started from being quite basic websites and are now industry giants today.



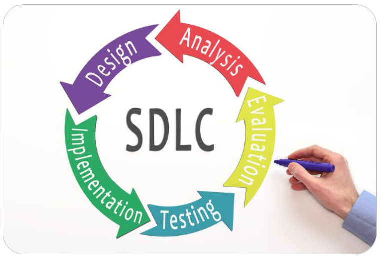
LEARN
BUILD
MVP
MEASURE
Help you focus on core Idea
Early market entry
Early testing opportunity
Continuously updated with customer feedback
Less resource and time wastage

1.2 Software Development Life Cycle (SDLC)

SDLC is a well-defined path for a software development process, that ensures timely delivery of software under budget constraints while collaborating with multiple teams.

The different phases of SDLC are as follows:

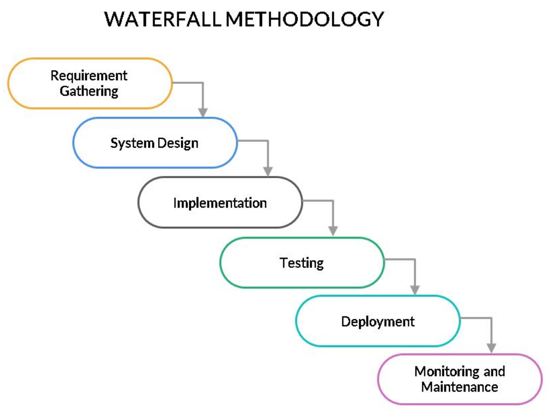
- Requirement gathering
- Design
- Implement
- Test
- Deploy and maintenance



1.3 Waterfall Methodology

The waterfall methodology is a linear sequential model of software development, where each phase starts only after the completion of the previous phase. Thus, a product is developed and delivered in one go.

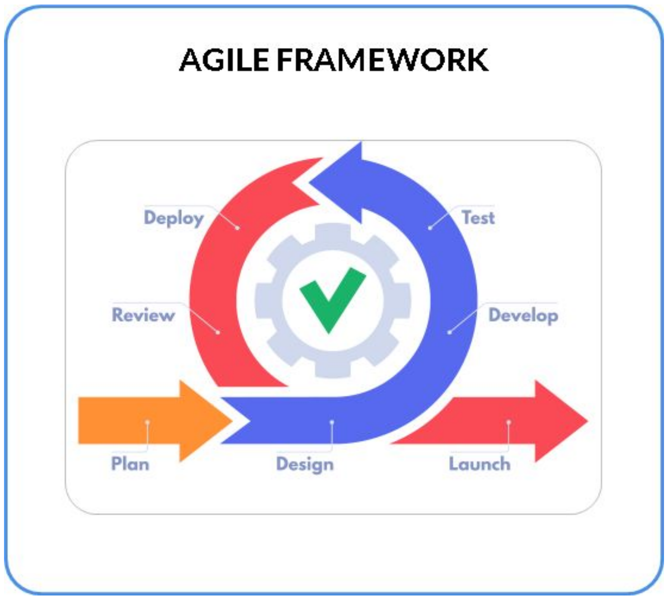
This methodology is inspired by early construction and manufacturing projects. Manufactured products are identical, that is, cars produced on the production line. But, given the fact that not every software produced by a company is the same, and even the requirements change very often in the software industry, a waterfall-like methodology is risky.

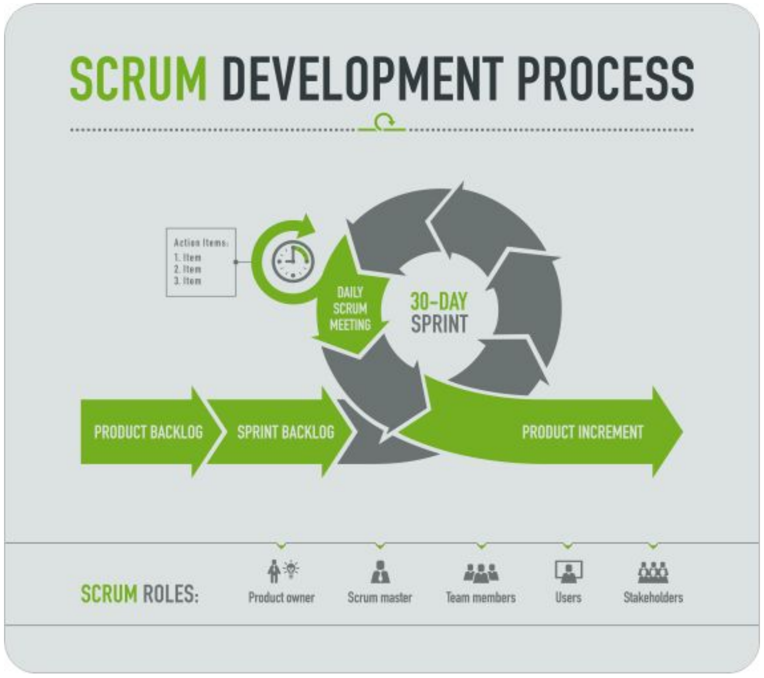


1.4 Agile Methodologies

'Agile' is an umbrella term for many modern methodologies focusing on iterative and incremental software development.

Scrum is the most popular agile framework.
In Scrum, members of a small cross-functional team work together to produce software in multiple short iterations of around 30 days, called a sprint. They start with a product backlog, a to-do list of items that need to be done to develop the product. Certain items are pulled from the product backlog for every sprint, known as a sprint backlog. At the end of every sprint, something deliverable is to be produced. This ensures that every sprint adds value to the customers.



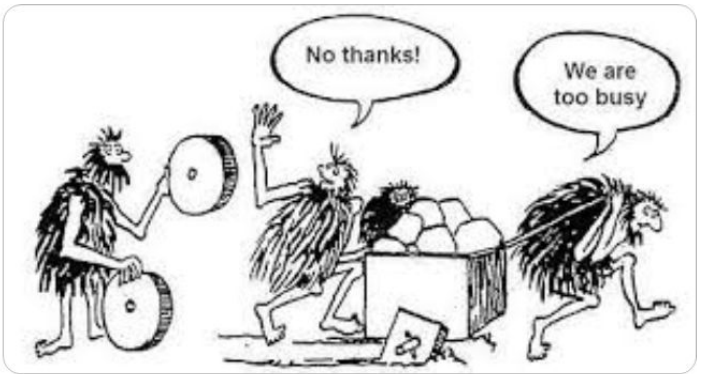


If you are interested to know more about project management, then watch the following video:
A brief overview of the Scrum framework.

For more detailed information on Scrum, refer to the Scrum guide at scrumguides.org.

1.5 Deployment Issues and Conflicts Between the Developer and Operation Teams

A crucial phase of SDLC is the deployment phase. Generally, a software works fine on the developers' computer but fails to work on the production servers. Many different reasons can cause this issue. Each part of the software has certain dependencies and a different environment that is required to run it. During the different SDLC phases, it is necessary to ensure consistency in the environment at all stages.
As organisations are becoming more agile, they leverage their creativity to satisfy their customers' needs. In their quest of agility, the developers push new code frequently, but the operations team finds it hard to cope with the frequent changes as it brings more risks for production systems to break. The operations team is responsible for the production environment's stability, and they have to keep things always up and working. They introduce many checklists to ensure that nothing wrong is deployed on the production environment, which slows down the process. Hence, this beats the motive of agility.



A few common deployment issues are as follows:

1. If the duration between deployments is very long, it is natural to forget some edge cases that you need to check before initiating the deployments. Hence, your application can fail on the production server.

2. Manually deploying on multiple servers is hectic and prone to error. While coding the deployment steps manually, making mistakes is a possibility.

3. Manual deployments are not easily trackable and hence, rolling back to the previous stable versions can be a difficult task.

4. The deployment process is a game of numbers in which a key performance index tells how efficiently your system works. Not being able to check the status of the impact of your configuration on the production system is also an issue.