Software Development LifeCycle

.NET CORE

*The **Software Development Life Cycle** is the process of dividing software development work into distinct phases to improve design and product management.*

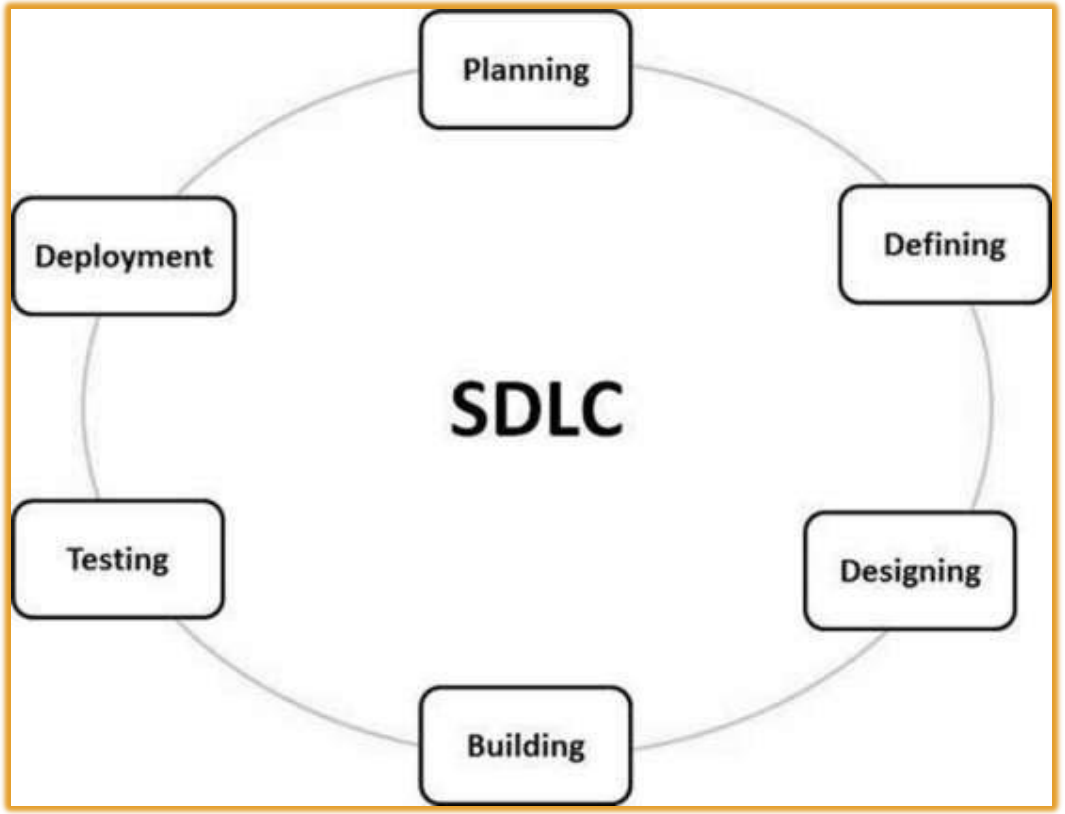
https://en.wikipedia.org/wiki/Software_development_process

SDLC – Software Development Lifecycle

https://www.tutorialspoint.com/sdlc/sdlc_overview.htm

The Software Development Life Cycle (SDLC) is a process used to design, develop and test software. The phases of the SDLC are:

1. Planning and Requirement Analysis.
2. Define the project requirements
3. Design the Product Architecture
4. Building the product,
5. Testing
6. Deployment



SDLC – Stages

https://www.tutorialspoint.com/sdlc/sdlc_overview.htm

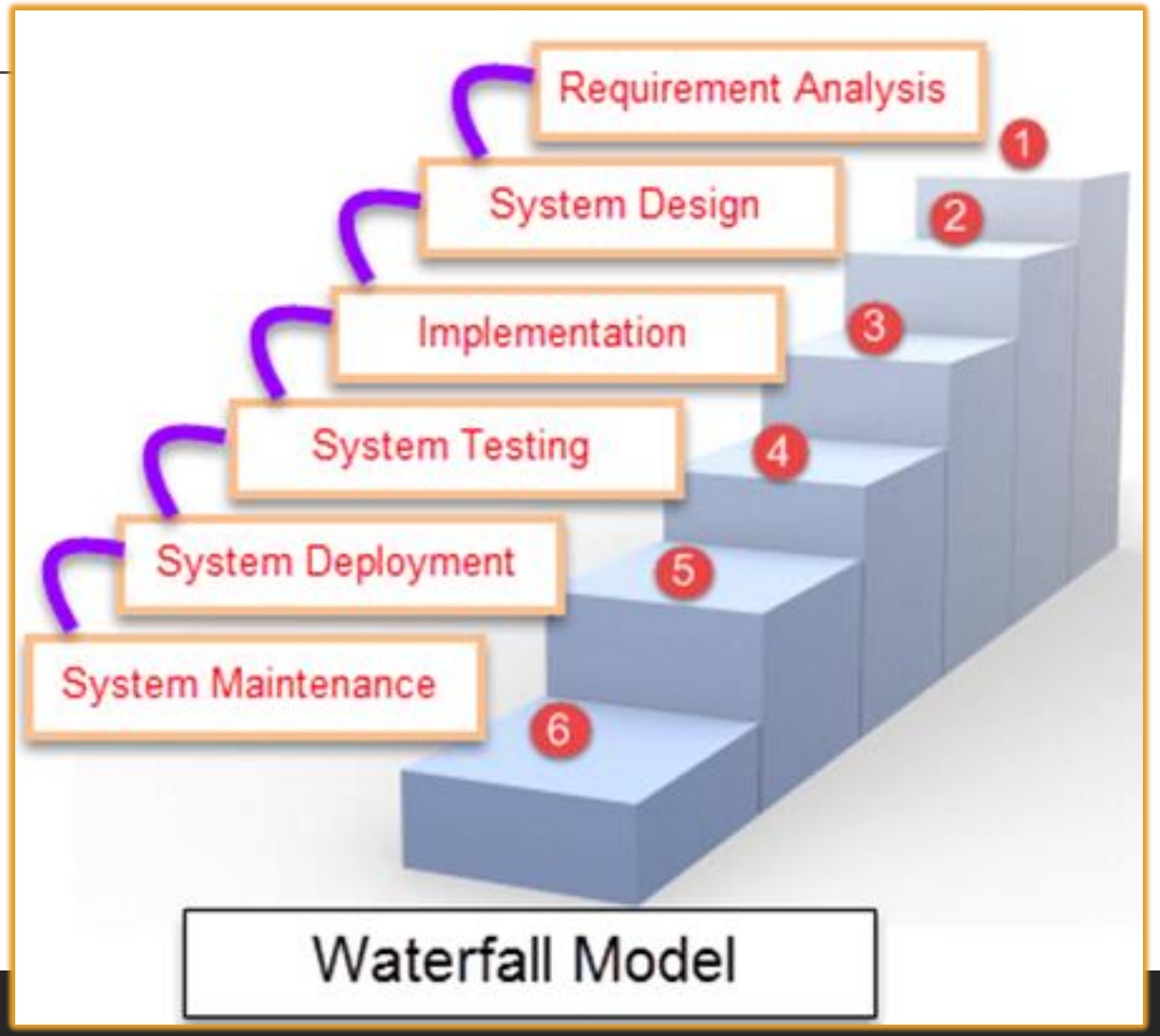
SDLC – Waterfall Model

https://en.wikipedia.org/wiki/Waterfall_model

The ***Waterfall Model*** breaks down project activities into phases which each depend on the deliverables of the previous phase.

Waterfall is less iterative and less flexible than other Models. ***Waterfall*** has high accountability and documentation. Progress flows downward like a waterfall through each phase only when the preceding phase is verified to be complete.

This model is too rigid for most situations.



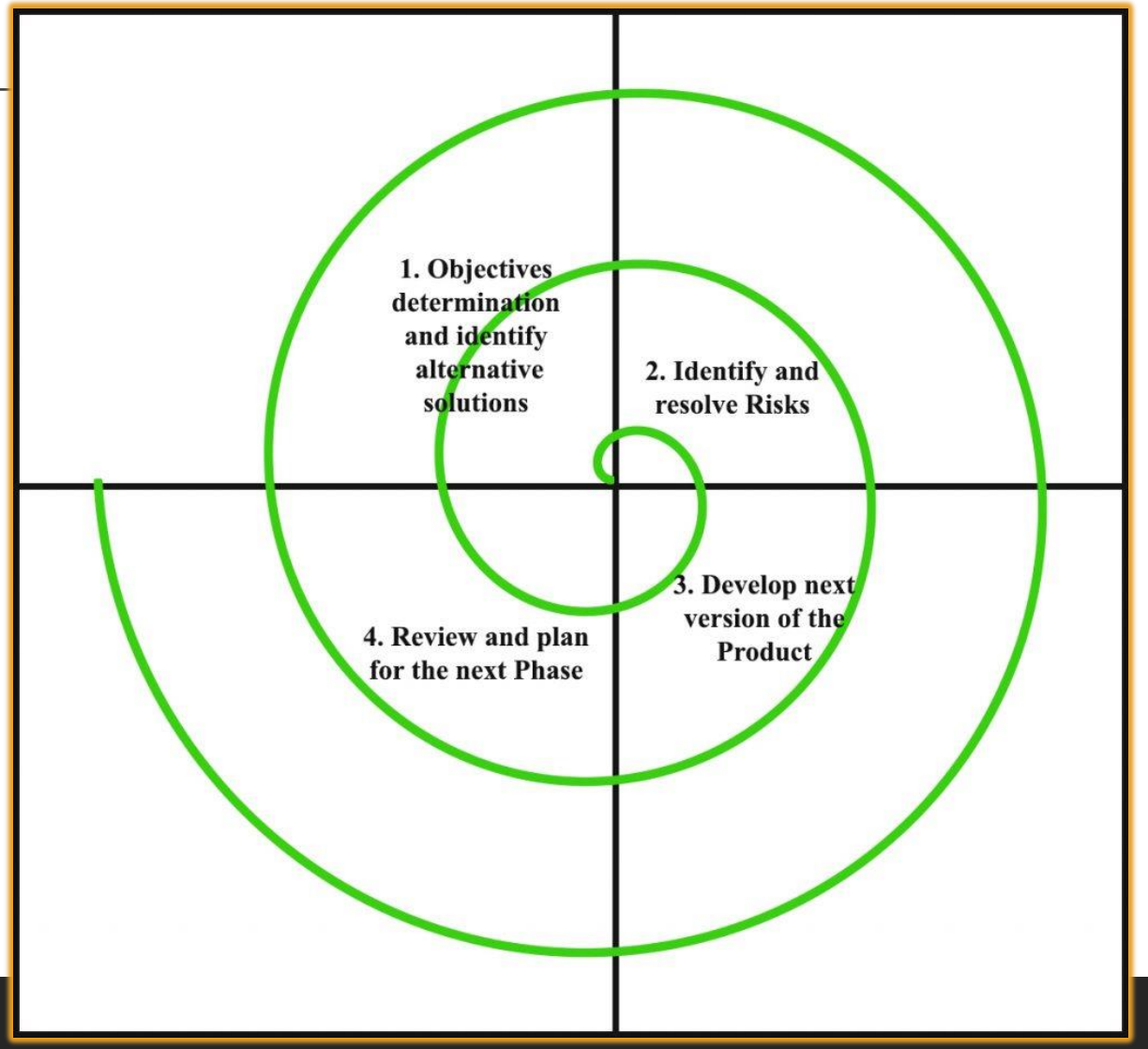
SDLC – Spiral Model

https://en.wikipedia.org/wiki/Spiral_model#The_six_invariants_of_spiral_model
https://www.geeksforgeeks.org/software-engineering-spiral-model/

The ***Spiral Model*** bases its processes on the unique risk patterns of a given project. This guides a team to adopt elements of other process models to implement the ***Spiral Model*** design pattern.

Spiral often results in a series of mini-Waterfalls. A single loop ***Spiral*** represents the Iterative ***Waterfall Model***.

The exact number of loops of the ***Spiral*** is unknown and can vary from project to project. Each loop of the ***Spiral*** is called a “phase” of the software development process. Each phase is divided into four quadrants.



SDLC – Spiral Model – Pros and Cons

https://en.wikipedia.org/wiki/Spiral_model#The_six_invariants_of_spiral_model
https://www.geeksforgeeks.org/software-engineering-spiral-model/

Advantages of Spiral Model	Disadvantage of Spiral Model
Risk Handling – deal with previously unknown risks in the next cycle.	Complexity
Good for Large, complex projects.	Expensive – Spiral is too extensive for small, simple projects.
Requirement flexibility – each cycle can adjust requirements.	This model requires expertise to determine the risk factors before each cycle.
Customer Satisfaction – Clients can see the results of each cycle as they happen.	Time management difficulty due to the unknown number of cycles in the whole project.

SDLC – Iterative Model

https://en.wikipedia.org/wiki/Iterative_and_incremental_development

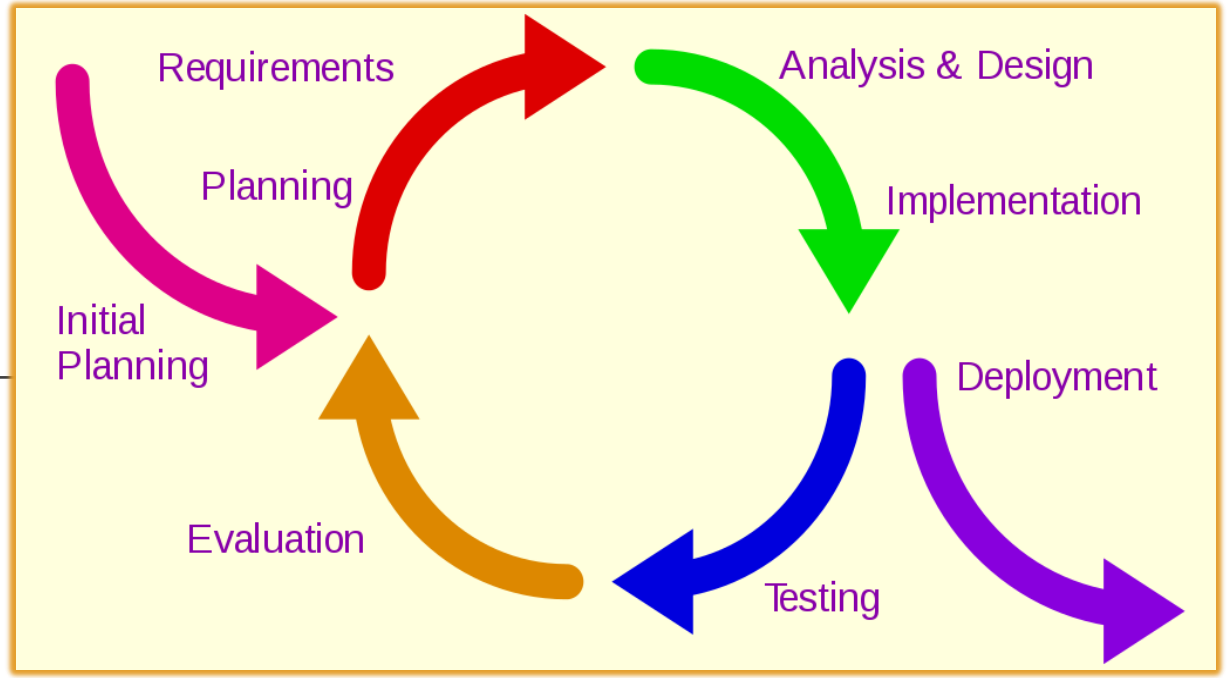
The ***Iterative Model*** is to develops a system through repeated, whole project development cycles (iterative) and in smaller portions at a time (incremental), This allows SE's to take advantage of what was learned during development of earlier parts or versions of the system.

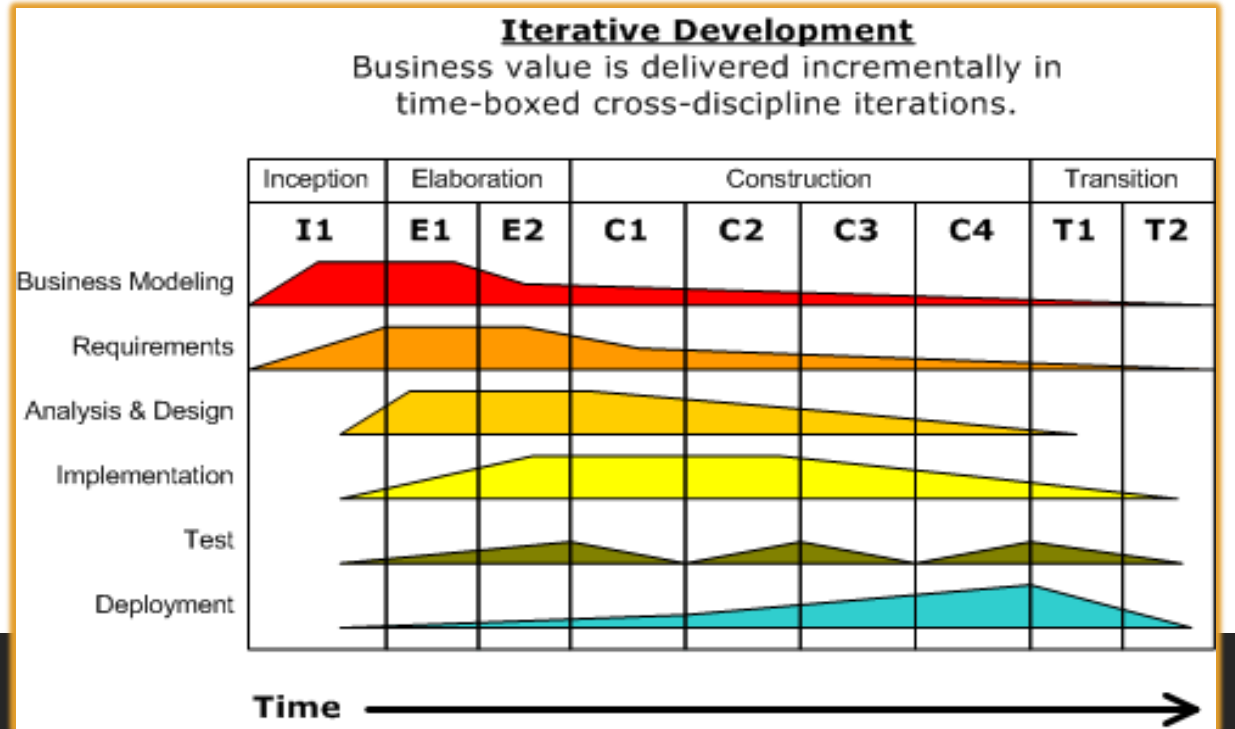
Each iteration of in the development process can add features until the full system is implemented.

A ***Project Control List*** is updated after each iteration to focus developers on specific functionality to implement.

Analysis of each iteration is based on Client/User feedback. Desired changes are implemented in the next iteration.

The ***Iterative Model*** is distinguished from ***Waterfall*** in that backtracking is possible in ***Iterative*** while in ***Waterfall*** each step in the process in completed before moving on to the next step.



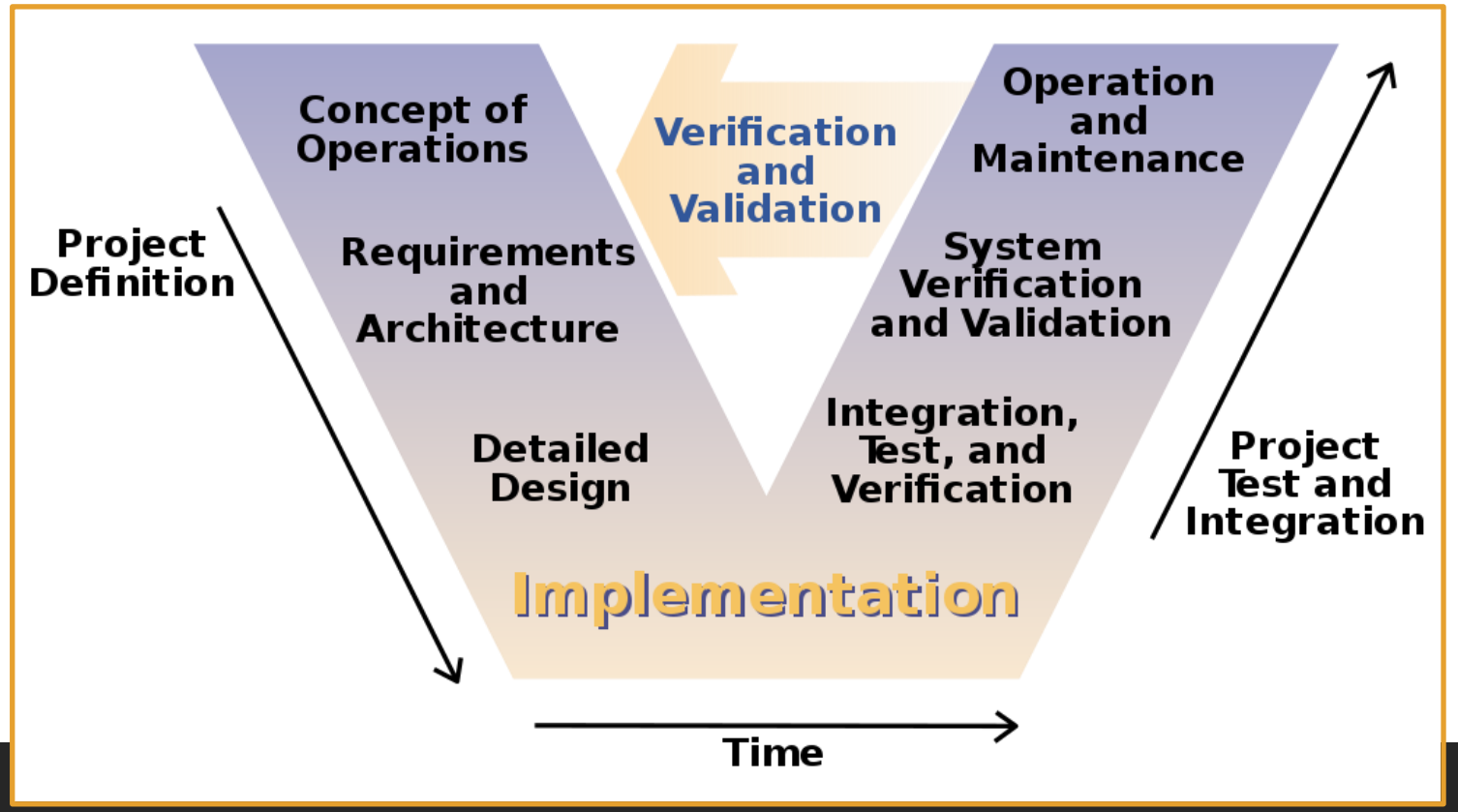


SDLC - V-Model

https://en.wikipedia.org/wiki/V-Model_(software_development)

The ***V-Model*** can be considered an extension of the ***Waterfall Model.***
Instead of moving down linearly, the process steps bend back upwards after the coding phase to form a V shape.

The ***V-Model*** demonstrates the relationships between each phase of the development life cycle and its associated phase of testing.
The horizontal and vertical axes represents time or project completeness (left-to-right) and level of abstraction (top-to-bottom).



SDLC - Big Bang Model

https://www.tutorialspoint.com/sdlc/sdlc_bigbang_model.htm

The ***Big Bang Model*** does not follow any specific process. Even the client may not be sure about what exactly he wants. The requirements are implemented on the fly without much analysis.

Development starts with the required money and efforts as the only input. The output is the software developed, which may not be as per customer requirement.

Usually the ***Big Bang Model*** is followed for small projects where the development teams are also very small. This model is higher risk than other models and misunderstandings can even lead to scrapping the whole project.

For small projects, the Big Bang Model offers easy management, rapid prototyping, flexibility, and requires few resources.

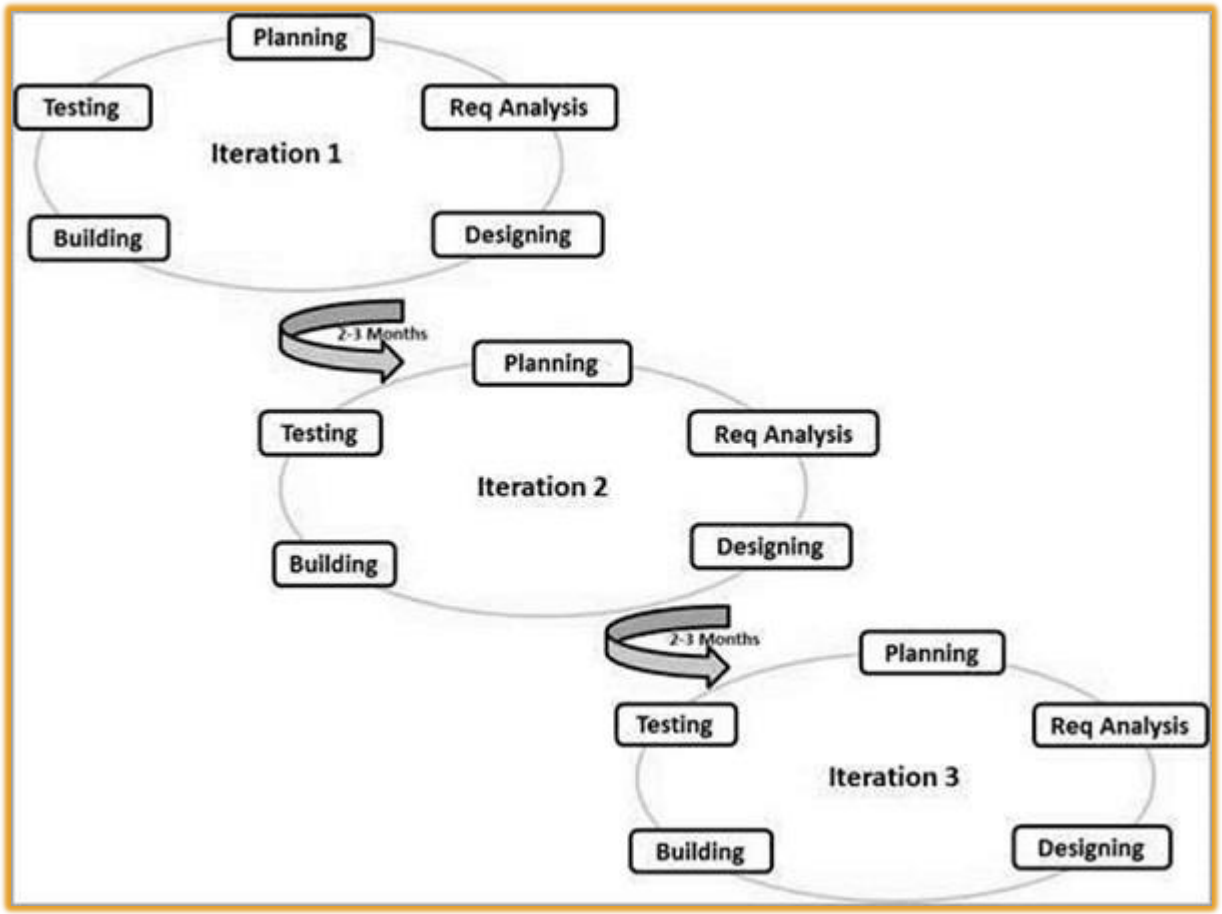
Agile – Overview

https://agilemanifesto.org/principles.html

Agile is a software development method in which interactions and are prioritized over processes, working software is more important than extensive documentation, collaboration with the client is prioritized above negotiation, and quick response to new requirements is the most important aspect of the whole process. There are multiple Agile styles. Together, they are referred to as ***Agile Methodologies***.

Agile Priorities:

- Early and continuous delivery (CI/CD).
- Quick acceptance of, and adjustment to, changing requirements.
- Quick delivery of a working product.
- Sustainable development process.
- Regularly scheduled sessions of introspection (Stand-up's)
- Constant adjustment to a dynamic development environment.



Agile - Characteristics

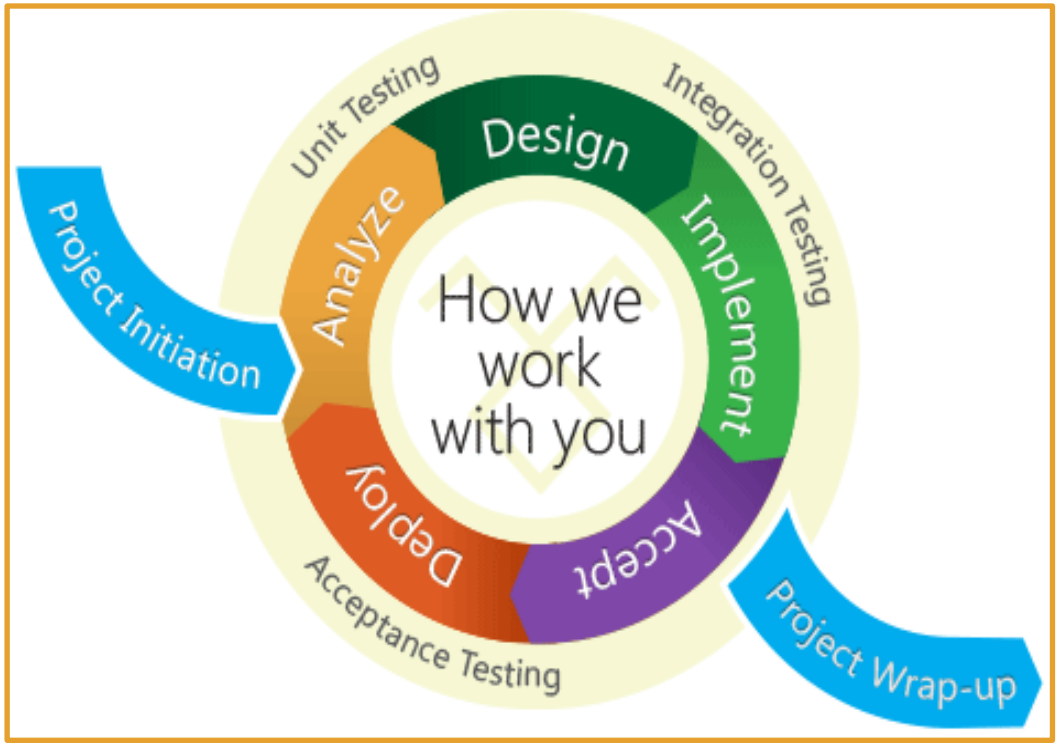
https://en.wikipedia.org/wiki/Agile_software_development
https://www.agilealliance.org/glossary/daily-meeting
https://www.tutorialspoint.com/sdlc/sdlc_agile_model.htm

The ***Agile*** Process is a series of ***Sprints*** where each team has a specific ***User Story*** to guide their efforts. A ***Sprint*** has a fixed start and stop date and typically lasts 2-4 weeks. At the end of each ***Sprint*** teams demonstrate what they've accomplished to each other and to the client. Feedback is taken to inform the next ***Sprint***.

Daily Standup – Members report to all involved:

1. What they did yesterday,.
2. What they intend to do today.
3. What "blockers" they have.

Information Radiator - A board presenting an up-to-date summary of the status of the product.
Continuous Integration – The practice of merging all developers' working copies to a shared master several times a day.
Test Driven Development – Relies on the repetition of a very short development cycle: requirements are turned into very specific test cases, then the code is improved so that the tests pass.



Agile – Sprint

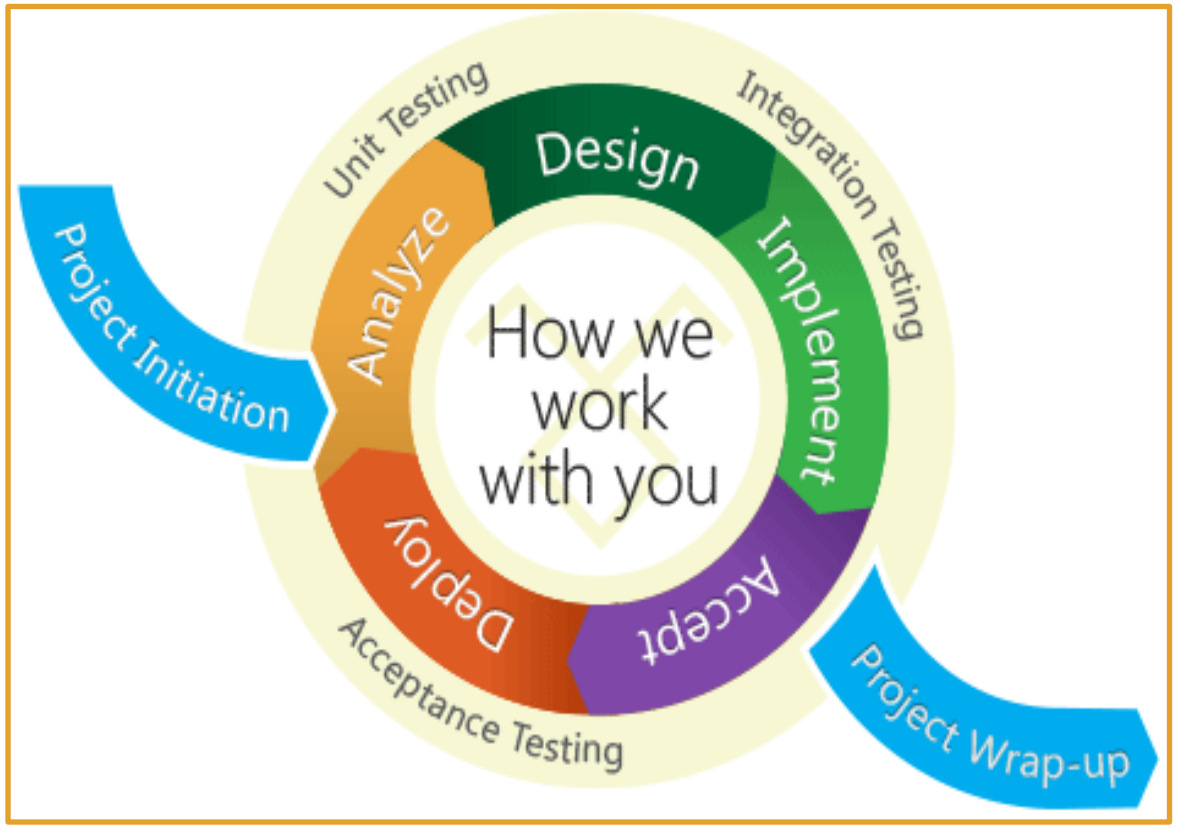
https://www.agilealliance.org/glossary/user-story-template/

A ***Sprint*** in ***Agile*** is a period of time in which teams work to complete a ***User Story***. All ***User Stories*** for a project form the ***Backlog***.

Each ***User Story*** is assigned a number of ***points*** representing the expected effort needed to complete it.

After the ***Sprint***, Team members add up effort ***points*** for ***User Stories*** completed during the ***Sprint***. Points completed represent the ***Velocity***.

Knowing ***Velocity***, the team can estimate how many ***Sprints*** will be required to complete the project. Teams can use their ***Velocity*** to modulate how much work they take on in each ***Sprint***.



Agile – Scrum

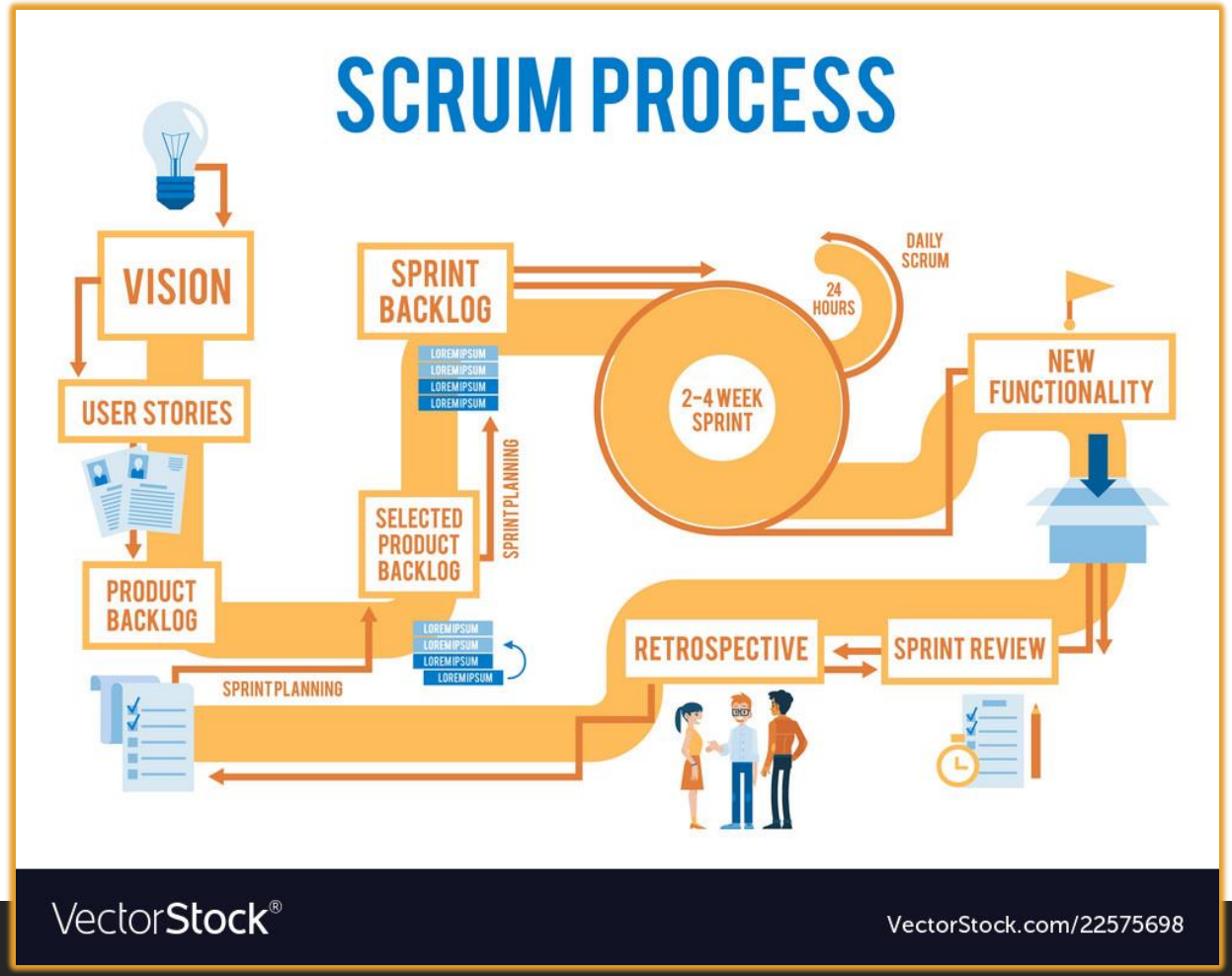
https://www.agilealliance.org/glossary/scrum

Scrum is an ***Agile*** framework in which teams work iteratively to:

1. hypothesize about how the final project should work,
2. try to implement the idea,
3. reflect on the experience,
4. make adjustments,
5. then start the cycle again.

Scrum is best used when the amount of work required can be split in to more than one ***Sprint***.

Scrum requires transparency between members. Daily ***Stand-ups*** allow each person to speak to the difficulties (***Blockers***) they are having. Adjustments are made based on the Stand-up to focus the following days efforts.



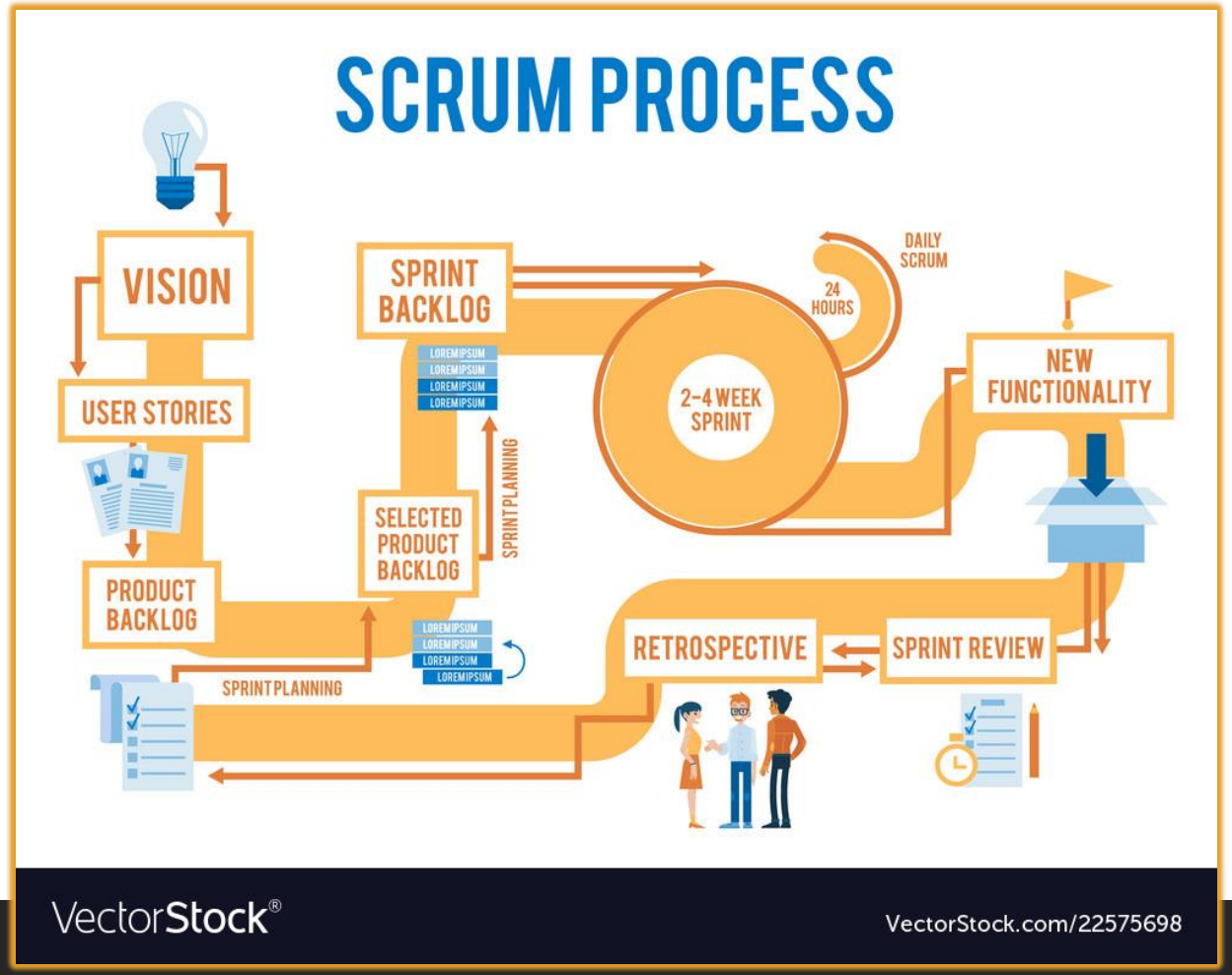
Agile – Scrum

https://www.agilealliance.org/glossary/scrum

A ***Scrum*** starts with a ***Sprint*** Planning meeting with the client to form ***User Stories*** that form the ***Sprint Backlog***. The ***Sprint Backlog*** items are added to the ***Scrum Board***.

Then then teams decide how they will successfully deliver a working ***Iteration*** at the end of the first ***Sprint***. The ***Sprint Backlog*** is then locked and no more ***User Stories*** can be added.

For the duration of the ***Sprint***, the team ***Scrum Master*** conducts a 15-minute ***Daily Scrum*** in which members discuss ***blockers*** and coordinate efforts for the next day.

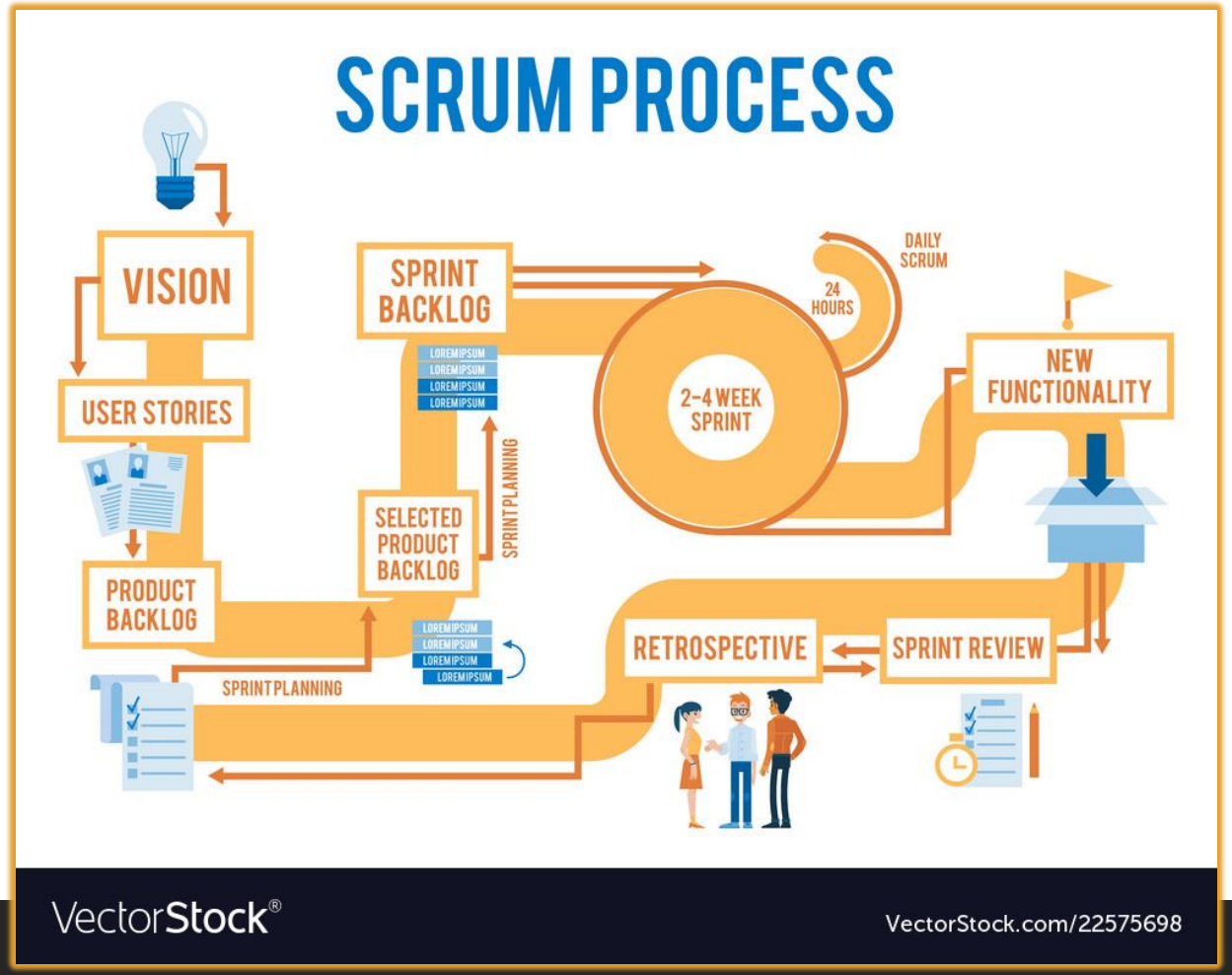


Agile – Scrum

https://www.agilealliance.org/glossary/scrum

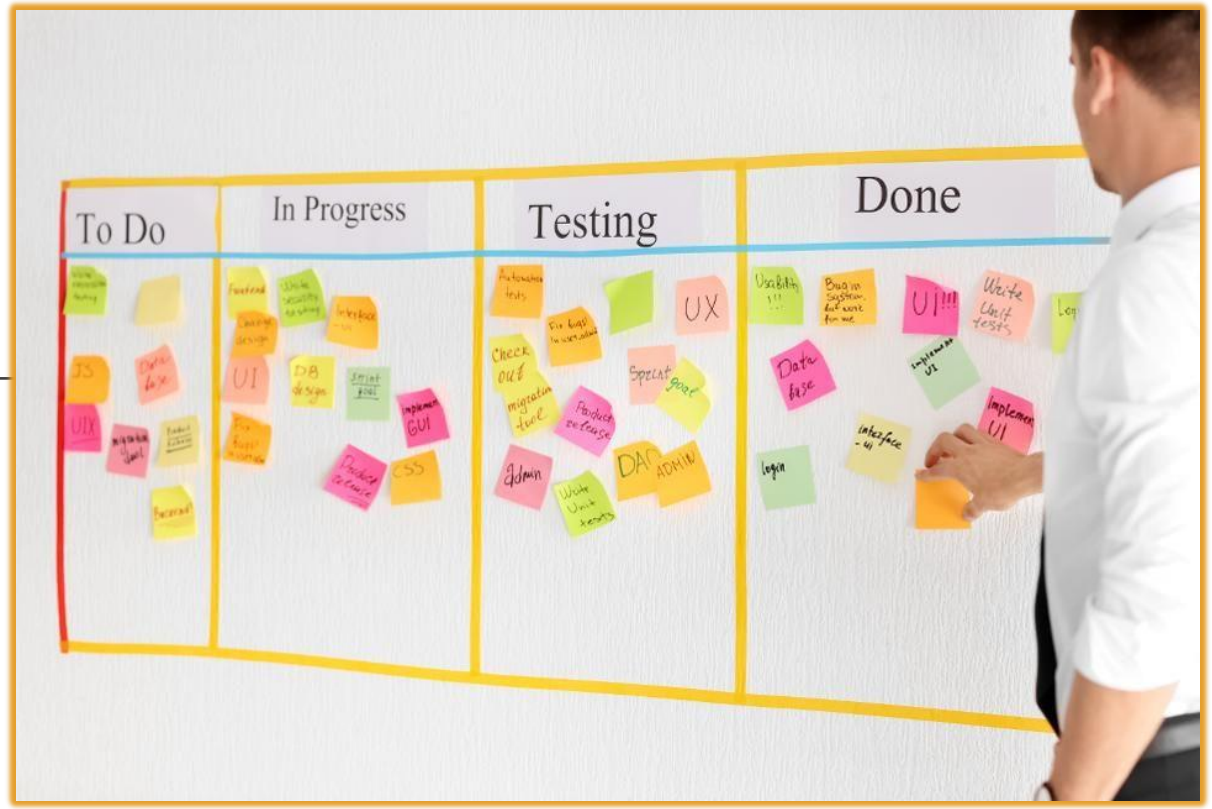
Sprint Review - After the ***Sprint***, teams, client and ***Stakeholders*** meet to demonstrate the ***Increment*** and review the results of the ***Sprint***. Feedback from this meeting is placed in the ***Product Backlog*** for consideration in future iterations of the product.

Sprint Retrospective – After the ***Sprint Review***, the team meets to make adjustments going forward and decide on which ***User Stories*** will be included in the next ***Sprint***.



Agile – Kanban

https://www.atlassian.com/agile/kanban/wip-limits
https://www.agilealliance.org/glossary/kanban



The ***Kanban Method*** gets its name from the use of "***Kanban***", visual signaling mechanisms, to control work in progress for intangible work products. A general term for systems using the ***Kanban Method*** is "***flow***". Flow reflects that work ***flows*** continuously through the system instead of being organized into distinct timeboxes.

Kanban systems use mechanisms such as a ***Kanban Board*** to visualize work and the process it goes through.
Benefits of the ***Kanban*** system are that workers are aided in maintaining a narrow focus on a smaller amount of tasks.
Bottlenecks and blockers are identified and addressed so that ***flow*** is maximized.
Feedback loops used to iterate over the process and make small changes to increase efficiency. Feedback Loops are a repeating series of reviews of each part of the workflow. Balance of effort across services and response to risks and inefficiencies is emphasized
Kanban starts with the process as it currently exists and applies continuous and incremental improvement.

Agile – CMMI (Capability Maturity Model Integration)

https://en.wikipedia.org/wiki/Capability_Maturity_Model_Integration
http://dthomas-software.co.uk/resources/frequently-asked-questions/what-is-cmmi-2/

CMMI is a process level improvement training and appraisal program. Its main sponsors included the Office of the Secretary of Defense (OSD) and the National Defense Industrial Association.

CMMI was developed by Carnegie Mellon Univ. and the *CMMI* project. They aimed to improve existing software development processes.

The *CMMI* principal is that "the quality of a system or product is highly influenced by the process used to develop and maintain it". *CMMI* can be used to guide process improvement across a project, a division, or an entire organization.