Software Development Cycle

Design and Planning to build a large software is very important.

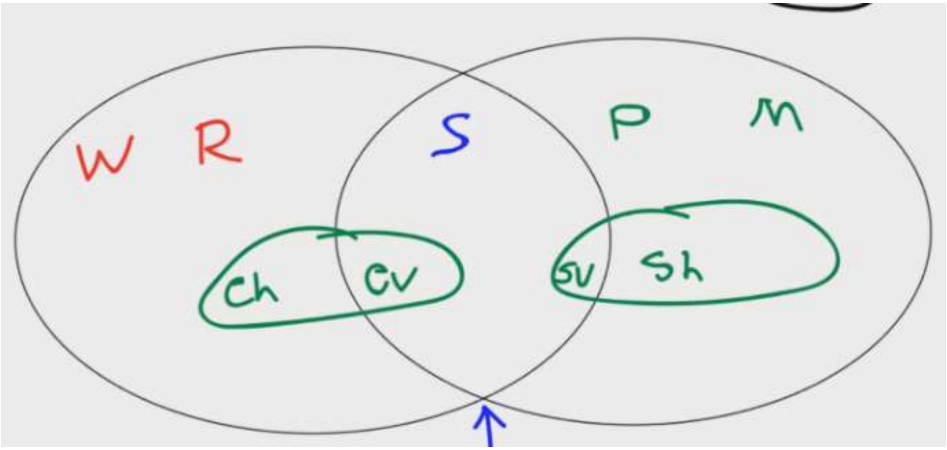
- Typical software development lifecycle steps:
  - Requirements (WHAT): A way of figuring out exact specifications of what the software should do (or) finding goals of a system.
    - Functional and Non-Functional Requirements.
  - Design(HOW): To come up with the exact plan on how we can create a product that satisfies the requirements.
  - Implementation: This is step where we actually begin creating the solution
  - Verification : Verifying whether the solutions is solving the problem.

WRSPM Model

(World, Requirements, Specifications, Program, Machine)

Referential Model to understand connection between requirements, specifications and the real world.

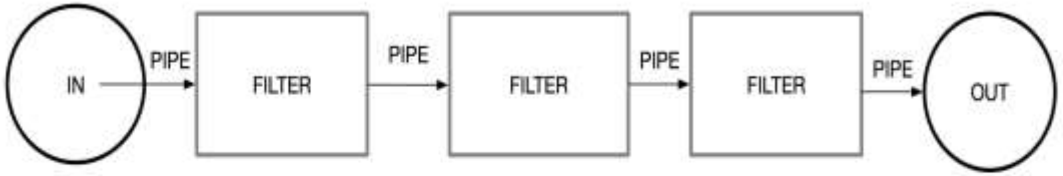
- World: Making worldly assumptions like assuming availability of electricity when we plan to install an ATM machine.
- Requirements: Defining the problem at hand in terms of user of the system.
- Specifications: Linking together the requirement of the environment to system
- Program: Program or the development of the code.
- Machine : Hardware specifications for the needed software.



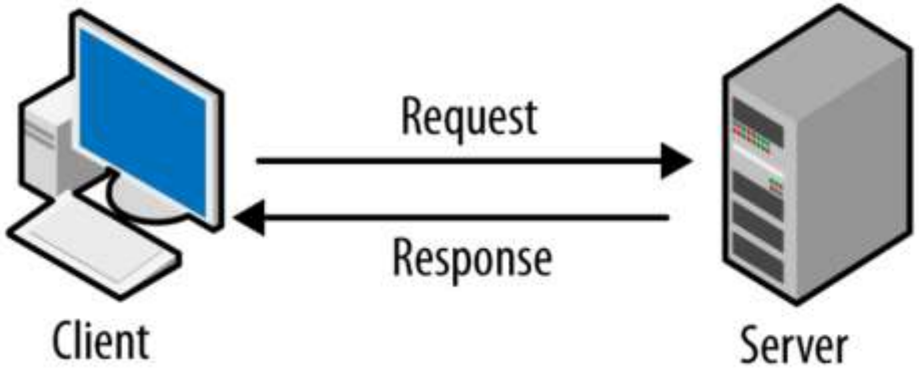
Software Architecture

- Breaking up larger systems and ideas, into smaller focused systems.
- Taking this broad set of ideas and guidelines, and have to organize it into functioning areas.
- The blueprint of the entire system is designed by putting through the same process of breaking each of these areas into smaller and smaller pieces.
- **“Bad architecture CAN'T be fixed with good programming”.**

Architecture Patterns



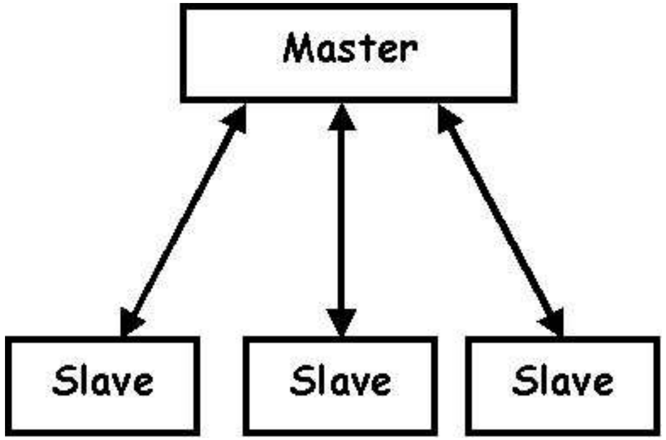
Data is passed through the filters, the connection which connects those filters are pipes. Mix and match of the logic in any order is allowed.



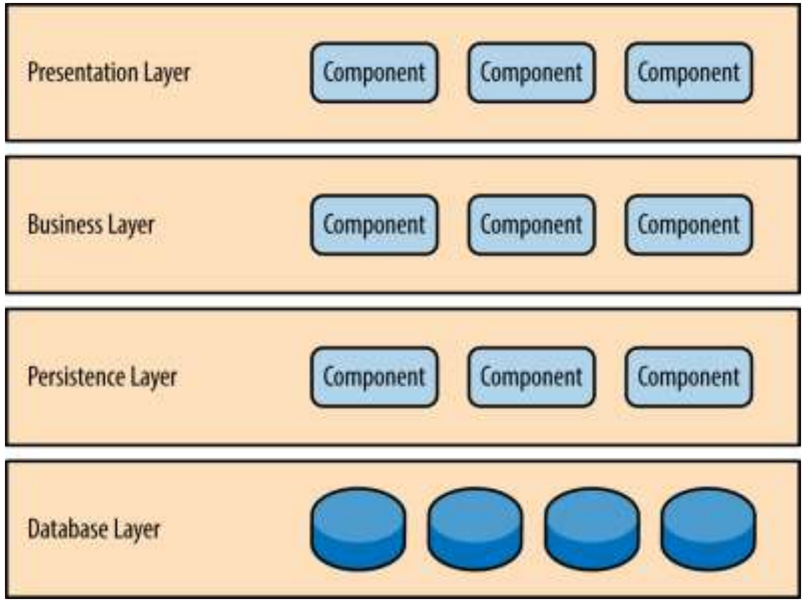
Architecture of a computer network in which many clients request and receive service from a centralized server.

- Client Requests
- server distributes the requested information back to clients.

Architecture Patterns



The master slave pattern consists of two elements, the master, and the slave. The master is in full control of all slaves associated with it.
Ex: Multithreading, Backup-servers



Divvying up program into layers of technology. These layers only communicate with adjacent layers.

Design and Modularity

Once we have the architecture of the system set up, we can begin working on the design. The design is where we really plan out our system.

- **Subsystem** - Independent system which holds independent value.
- **Module** - Component of a subsystem which cannot function as a standalone.

Information Hiding and Data Encapsulation

- **Information Hiding** - Hiding the complexity of the program inside of a "black box".
- **Data Encapsulation** - Hiding the implementation details from the user, providing only an interface.

Coupling

Measuring the strength of connections between modules or subsystems.

❖ Tight Coupling: Strong Dependence between modules.

- Content coupling : When one module modifies or relies on the inner workings of another module
- Common coupling: Several Modules have access to same global data.
- External Coupling: Direct access to external I/O.

❖ Medium Coupling : Dependency is reduced to some extent.

- Control Coupling: when data is passed that influences the internal logic of another module.
- Data Structure Coupling - This is when multiple modules share the same data-structure.

❖ Loose Coupling: Ability to swap our modules easily, with little code that must be updated. The Program becomes more flexible, and easier to maintain.

- Data Coupling: Modules are processing the data, and making their decisions independently. Due to this, our dependency overall is extremely low.
- Message Coupling - This coupling is when messages or commands are passed between modules.
- No Coupling - No communication between modules whatsoever.

Cohesion

A modules ability to work towards a set and well defined purpose.

- ❖Weak Cohesion
  - Co-incidential Cohesion: Tasks within the modules are linked because they are in the same module.
  - Temporal Cohesion: The tasks within the module are only linked because events happen around the same time.
  - Logical Cohesion: The tasks within the module are linked due to being in the same general category.
- Medium Cohesion:
  - Procedural Cohesion - The order of execution passes from one command to the next.
  - Communicational Cohesion - When all tasks support the same input and output data.
  - Sequential Cohesion - When all tasks work in which the output data for one, is the input data for the next.
- ❖Strong Cohesion:This is when all tasks within a module support activities needed for only one problem related task.

IDEAL - STRONG COHESION AND LOOSE COUPLING

Types Of Testing

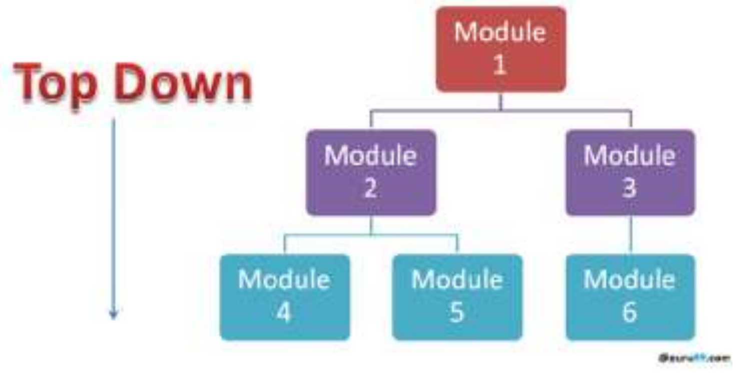
Testing is the process of finding errors and bug is a deviation from the expected behaviour.

Types Of Testing	Explanation
Unit Testing	Unit testing focus on the smallest unit of a software. Different areas are repeatedly isolated and each module is given a set of test cases and results are checked against the oracle.
Integration Testing	Integration testing is testing the architecture and the communication as a whole. In Integration testing, we will come up with test cases for groups of modules
Back- to back Testing	A test which compares a known good to a new version. Step 1: Version-1 output is compared with oracle. Step2 : Output with additional functionalities(Version-2) is tested with output of version-1.
Black Box vs White Box Testing	
White Box: A tester knows the internal workings of the system. Black Box: A tester does not know the internal workings of the system.	

Types Of Testing

Incremental Testing: In incremental testing, we slowly add one module after the next into the testing environment.

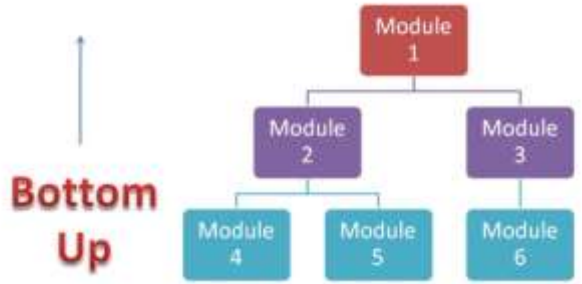
Top-Down Testing



Stub – Returns Hard-coded values.

- Testing begins at the highest possible level, and then work ways down. To have this work, we need to have a set dummy modules(stub) that we slowly replace with regular modules.

Bottom-up Testing

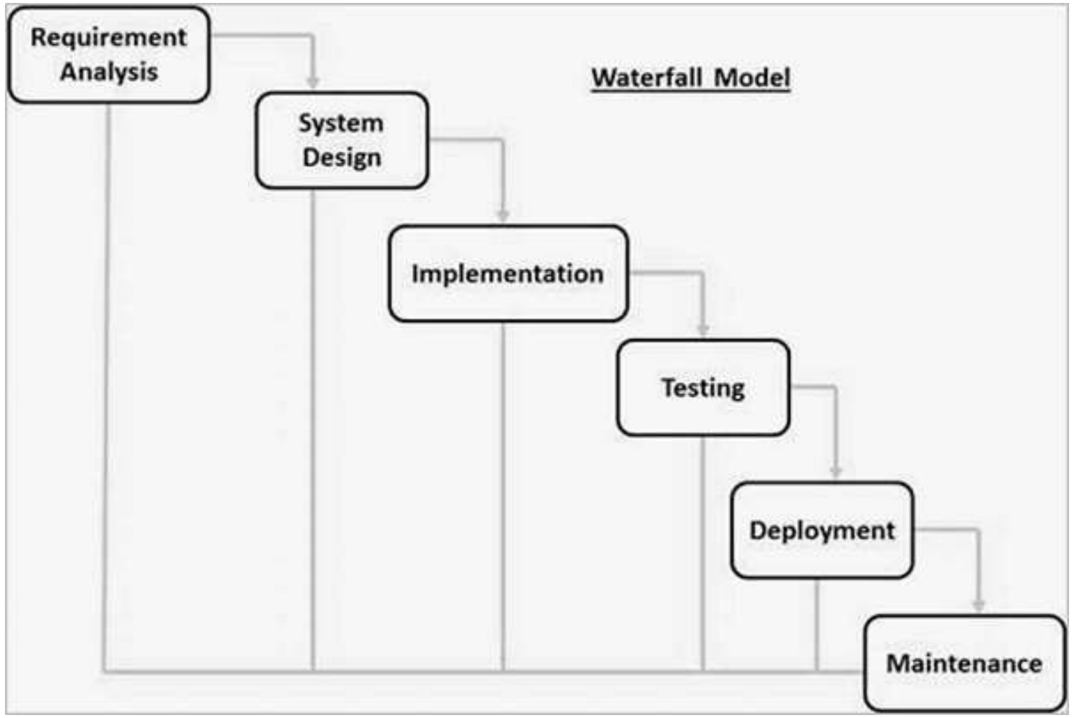


Driver-Templates which execute commands and initialize the needed variables.

- Bottom-Up Testing is the opposite of Top-Down. Here work starts from the bottom, and use things called drivers to make our way upwards.

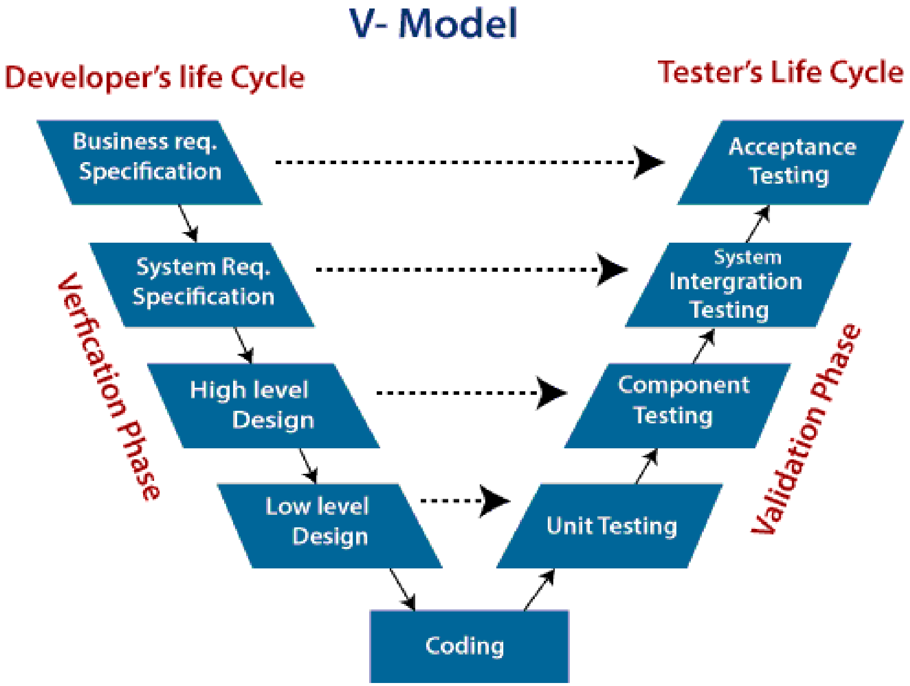
Software development Models

Waterfall Model



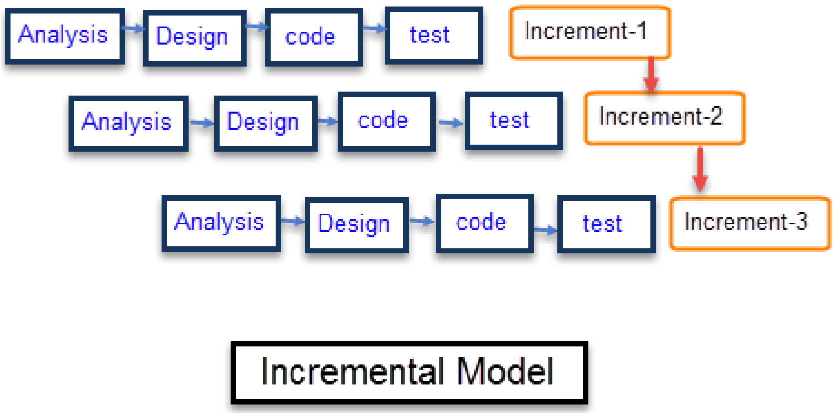
In this Waterfall model the outcome of one phase acts as the input for the next phase sequentially.

V-Model



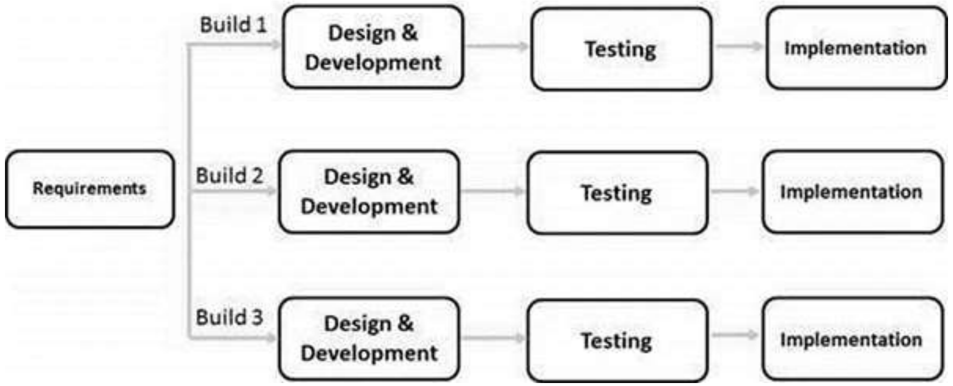
The V-Model is an extension of the waterfall model and is based on the association of a testing phase for each corresponding development stage.

Incremental



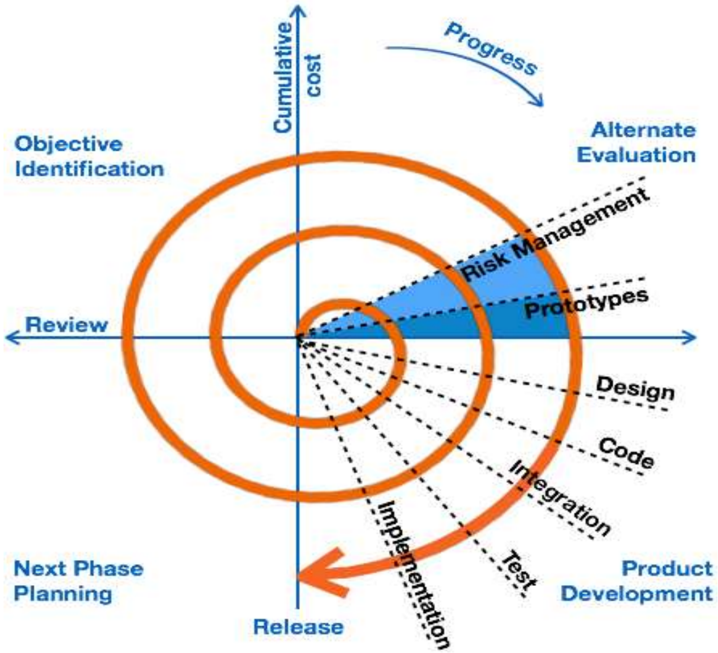
- Incremental Model is a process of software development where requirements are broken down into multiple standalone modules of software development cycle.
- Each iteration passes through the requirements, design, coding and testing phases.

Iterative



- Iterative process starts with a simple implementation of a small set of the software requirements and iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed.

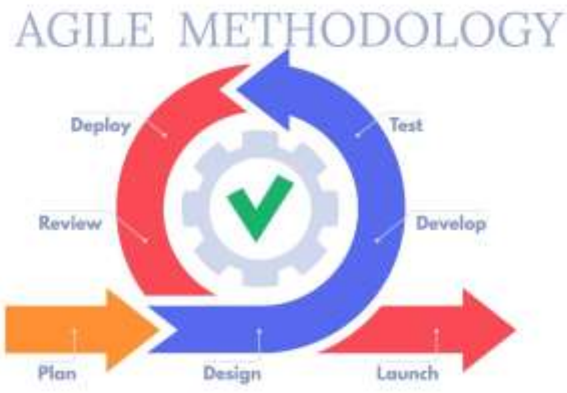
Spiral Model



Spiral Model involves repeatedly passes through these phases in iterations called Spirals.

- Identification : This phase starts with gathering the business requirements in the baseline spiral.
- Design: The Design phase starts with the conceptual design in the baseline spiral
- Construct phase refers to production of the actual software product at every spiral
- Risk Analysis includes identifying, estimating and monitoring the technical feasibility and management risks

Agile



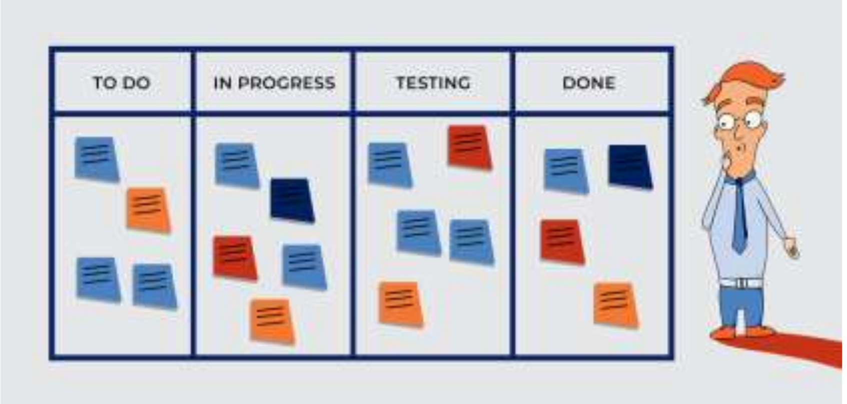
Agile is a set of ideas to develop better, quicker and more agile software

Agile Manifesto:

- ***Individuals and interactions*** over process and tools.
- ***Working software*** over comprehensive documentation.
- ***Customer collaboration*** over contract negotiation.
- ***Responding to change*** over following a plan.

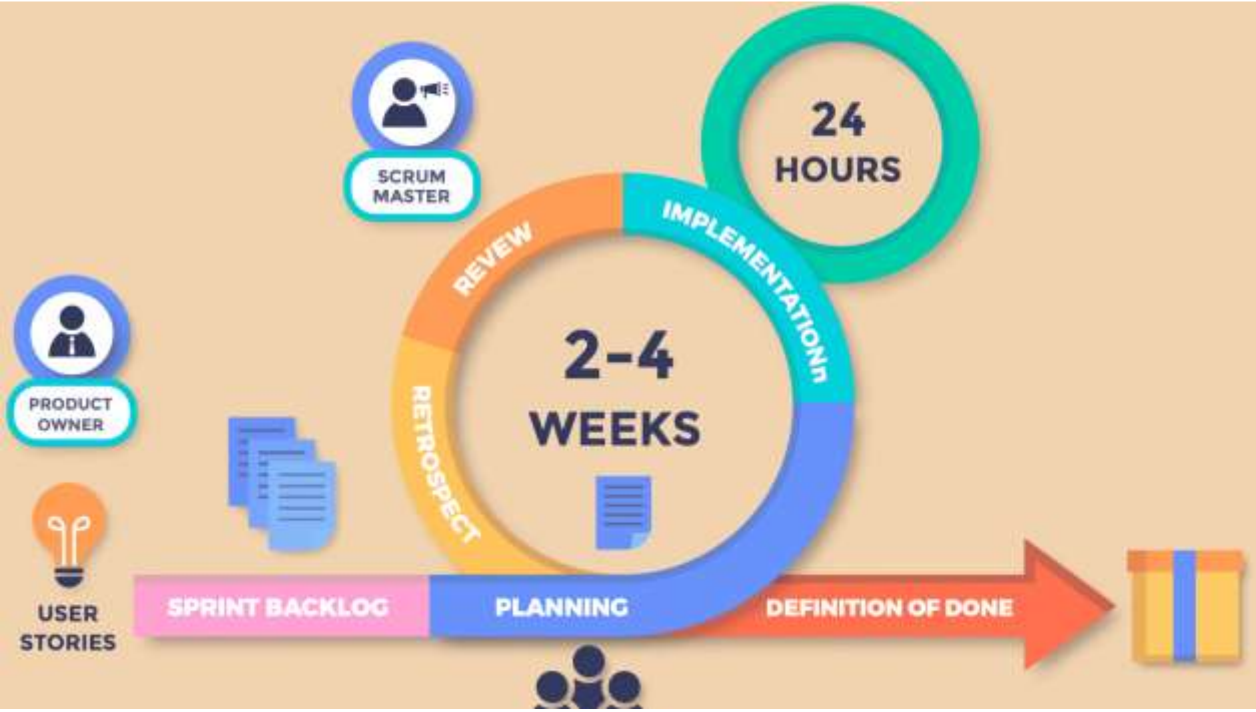
Agile Models:

Kanban - The kanban system is one of optimization. With kanban, we are trying to analyze the flow of production and figure out the slowdowns.



Lean Start-up : Making experiments to actually see whether the product is making money in market.

SCRUM



SCRUM Roles: Product Owner, Scrum Master, Development Team

SCRUM Events: Sprint Planning, Daily standup, The Sprint, Sprint Review, Sprint Retrospective**.**

SCRUM Artifacts: Sprint Backlog, Product Increment, Product Backlog

SCRUM Values: Focus,openness, Respect,Commitment, Courage.

SCRUM Pillars: Transparency, Inspection, Adaption.