Introduction to Software Engineering

Content

- Nature of Software
- Software Engineering, Software Process
- Capability Maturity Model (CMM) Generic Process Model
- Prescriptive Process Models: The Waterfall Model, V-model, Incremental Process Models
- Evolutionary Process Models
- Concurrent Models
- Agile process, Agility Principles
- Extreme Programming (XP), Scrum, Kanban model

Software

Computer Software is the product that software professionals build and then support over the long term.

Software encompasses: (1) **instructions** (computer programs) that when executed provide desired features, function, and performance; (2) **data structures** that enable the programs to adequately store and manipulate information and (3) **documentation** that describes the operation and use of the programs.

Software Engineering encompasses a process, a collection of methods (practice) and an array of tools that allow professionals to build high-quality computer software.

Nature of Software

Changing Nature of Software:

Nowadays, seven broad categories of computer software present continuing challenges for software engineers which is given below:

1. **System Software**
2. **Application Software**
3. **Engineering and Scientific Software**
4. **Embedded Software**
5. **Product-line Software**
6. **Web Application**
7. **Artificial Intelligence Software**

Nature of Software

1. **System Software**

- System software is a collection of programs which are written to service other programs.
- Some system software processes complex
- The system software area is characterized by the heavy interaction with computer hardware that requires scheduling, resource sharing, and sophisticated process management.
- E.g OS, device drivers, BIOS, Spreadsheets, web browsers, Programming languages.

Nature of Software

2. Application Software

- Application software is defined as programs that solve a specific business need.
- Application in this area process business or technical data in a way that facilitates business operation or management technical decision making.
- In addition to convention data processing application, application software is used to control business function in real time.
- E.g. web browser, spreadsheets, ERP, simulation software etc.

Nature of Software

3. Engineering and Scientific Software

- This software is used to facilitate the engineering function and task.
- however modern application within the engineering and scientific area are moving away from the conventional numerical algorithms.
- Computer-aided design, system simulation, and other interactive applications have begun to take a real-time and even system software characteristic.
- E.g. MATLAB, AUTOCAD etc

Nature of Software

4. Embedded Software

- Embedded software resides within the system or product
- is used to implement and control feature and function for the end-user and for the system itself.
- Embedded software can perform the limited and esoteric function or provided significant function and control capability.
- E.g. runtime services, activity tracker, embedded system, GPS System etc

Nature of Software

5. Product-line Software

- Designed to provide a specific capability for use by many different customers
- product line software can focus on the limited and esoteric marketplace or address the mass consumer market.
- The focus of product line software may be on a restricted marketplace like inventory control applications or can focus on mass consumer markets such as word processing, spreadsheets, graphics based applications, multimedia, database management and personal as well as business financial products.

Nature of Software

6. Web Application

- It is a client-server computer program which the client runs on the web browser.
- n their simplest form, Web apps can be little more than a set of linked hypertext files that present information using text and limited graphics.
- However, as e-commerce and B2B application grow in importance.
- Web apps are evolving into a sophisticate computing environment that not only provides a standalone feature, computing function, and content to the end user.
- E.g. google, amazon, facebook, netflix

Nature of Software

7. Artificial Intelligence Software

- Artificial intelligence software makes use of a non numerical algorithm to solve a complex problem that is not amenable to computation or straightforward analysis.
- Application within this area includes robotics, expert system, pattern recognition, artificial neural network, theorem proving and game playing.
- E.g. chatbot

Software Engineering

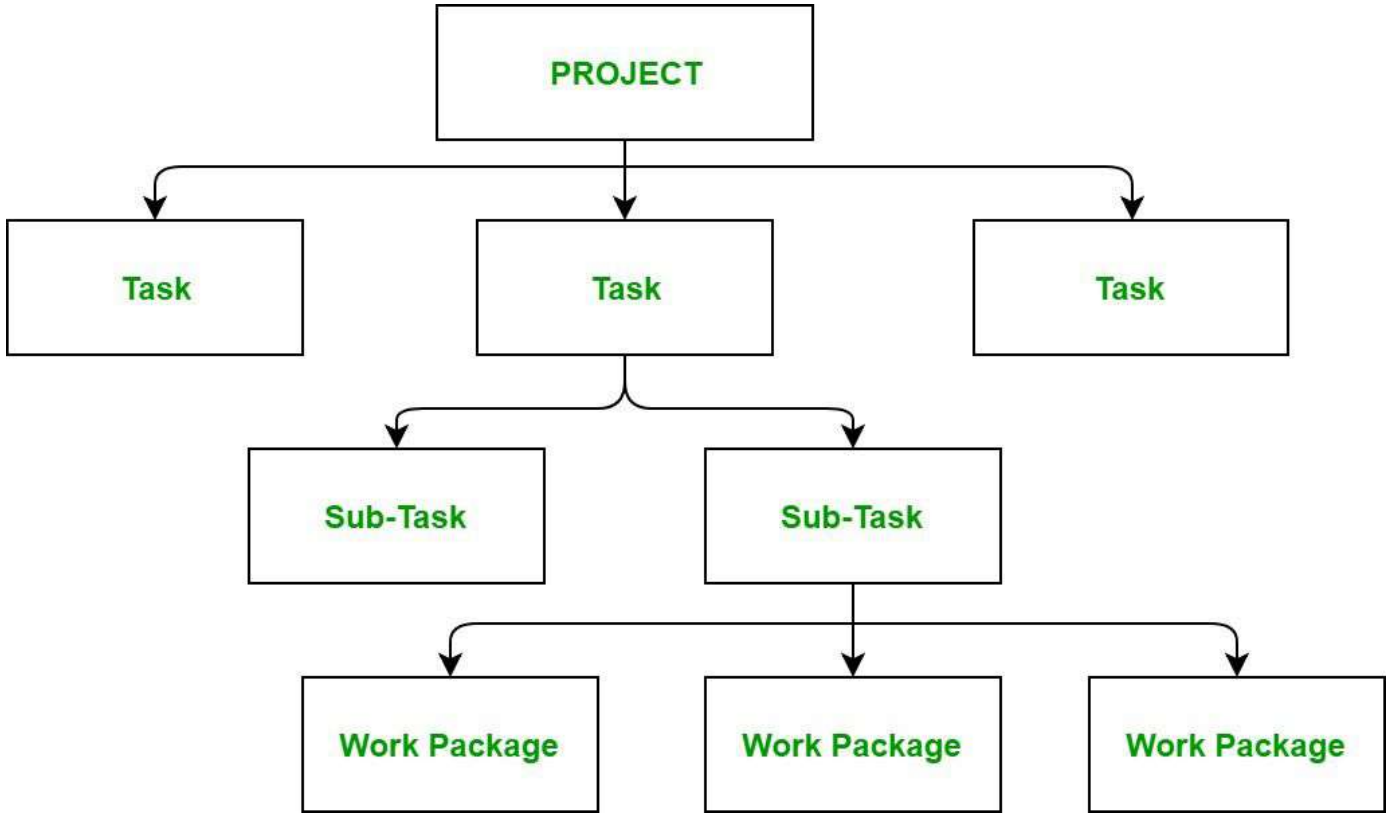
Software Engineering is the process of designing, developing, testing, and maintaining software. It is a systematic and disciplined approach to software development that aims to create high-quality, reliable, and maintainable software.

1. Software engineering includes a variety of techniques, tools, and methodologies, including requirements analysis, design, testing, and maintenance.
2. It is a rapidly evolving field, and new tools and technologies are constantly being developed to improve the software development process.
3. By following the principles of software engineering and using the appropriate tools and methodologies, software developers can create high-quality, reliable, and maintainable software that meets the needs of its users.
4. Software Engineering is mainly used for large projects based on software systems rather than single programs or applications.
5. The main goal of Software Engineering is to develop software applications for improving quality, budget, and time efficiency.
6. Software Engineering ensures that the software that has to be built should be consistent, correct, also on budget, on time, and within the required requirements.

Key Principles of Software Engineering

1. **Modularity**: Breaking the software into smaller, reusable components that can be developed and tested independently.
2. **Abstraction**: Hiding the implementation details of a component and exposing only the necessary functionality to other parts of the software.
3. **Encapsulation**: Wrapping up the data and functions of an object into a single unit, and protecting the internal state of an object from external modifications.
4. **Reusability**: Creating components that can be used in multiple projects, which can save time and resources.
5. **Maintenance**: Regularly updating and improving the software to fix bugs, add new features, and address security vulnerabilities.
6. **Testing**: Verifying that the software meets its requirements and is free of bugs.
7. **Design Patterns**: Solving recurring problems in software design by providing templates for solving them.

Work Breakdown Structure



Construction of work breakdown structure

1. Firstly, the project managers and top level management identifies the main deliverables of the project.
2. After this important step, these main deliverables are broke down into smaller higher-level tasks and this complete process is done recursively to produce much smaller independent tasks.
3. It depends on the project manager and team that upto which level of detail they want to break down their project.
4. Generally the lowest level tasks are the most simplest and independent tasks and takes less than two weeks worth of work.
5. Hence, there is no rule for upto which level we may build the work breakdown structure of the project as it totally depends upon the type of project we are working on and the management of the company.
6. The efficiency and success of the whole project majorly depends on the quality of the Work Breakdown Structure of the project and hence, it implies its importance.

Uses of work breakdown structure

1. **Cost estimation:** It allows doing a precise cost estimation of each activity.
2. **Time estimation:** It allows estimating the time that each activity will take more precisely.
3. **Easy project management:** It allows easy management of the project.
4. **Helps in project organization:** It helps in proper organization of the project by the top management.

Software Process

There are four basic key process activities:

1. **Software Specifications –**
   In this process, detailed description of a software system to be developed with its functional and non-functional requirements.

2. **Software Development –**
   In this process, designing, programming, documenting, testing, and bug fixing is **done**.

3. **Software Validation –**
   In this process, evaluation software product is done to ensure that the software meets the business requirements as well as the end users needs.

4. **Software Evolution –**
   It is a process of developing software initially, then timely updating it for various reasons.

Software Process

Software processes in software engineering refer to the methods and techniques used to develop and maintain software. Some examples of software processes include:

- **Waterfall**: a linear, sequential approach to software development, with distinct phases such as requirements gathering, design, implementation, testing, and maintenance.
- **Agile**: a flexible, iterative approach to software development, with an emphasis on rapid prototyping and continuous delivery.
- **Scrum**: a popular Agile methodology that emphasizes teamwork, iterative development, and a flexible, adaptive approach to planning and management.
- **DevOps**: a set of practices that aims to improve collaboration and communication between development and operations teams, with an emphasis on automating the software delivery process.

Each process has its own set of advantages and disadvantages, and the choice of which one to use depends on the specific project and organization.

Need of software process

- The software development team must decide the process model that is to be used for software product development and then the entire team must adhere to it.
- This is necessary because the software product development can then be done systematically.
- Each team member will understand what is the next activity and how to do it.
- Thus process model will bring the definiteness and discipline in overall development process.
- Every process model consists of definite entry and exit criteria for each phase.
- Hence the transition of the product through various phases is definite.
- If the process model is not followed for software development then any team member can perform any software development activity, this will ultimately cause a chaos and software project will definitely fail
- without using process model, it is difficult to monitor the progress of software product.
- Thus process model plays an important rule in software engineering.

Software Process

Waterfall:

Advantages:

- Clear and defined phases of development make it easy to plan and manage the project.
- It is well-suited for projects with well-defined and unchanging requirements.

Disadvantages:

- Changes made to the requirements during the development phase can be costly and time-consuming.
- It can be difficult to know how long each phase will take, making it difficult to estimate the overall time and cost of the project.
- It does not have much room for iteration and feedback throughout the development process.

Software Process

Agile:

Advantages:

1. Flexible and adaptable to changing requirements.
2. Emphasizes rapid prototyping and continuous delivery, which can help to identify and fix problems early on.
3. Encourages collaboration and communication between development teams and stakeholders.

Disadvantages:

1. It may be difficult to plan and manage a project using Agile methodologies, as requirements and deliverables are not always well-defined in advance.
2. It can be difficult to estimate the overall time and cost of a project, as the process is iterative and changes are made throughout the development.

Software Process

Scrum:

Advantages:

1. Encourages teamwork and collaboration.
2. Provides a flexible and adaptive framework for planning and managing software development projects.
3. Helps to identify and fix problems early on by using frequent testing and inspection.

Disadvantages:

1. A lack of understanding of Scrum methodologies can lead to confusion and inefficiency.
2. It can be difficult to estimate the overall time and cost of a project, as the process is iterative and changes are made throughout the development.

Software Process

DevOps:

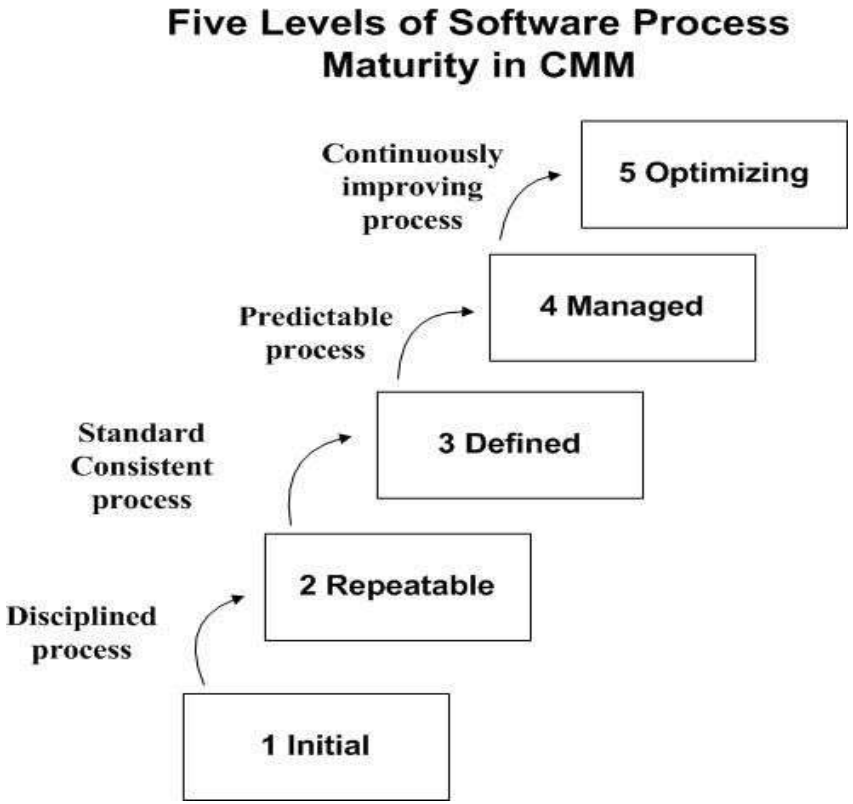
Advantages:

1. Improves collaboration and communication between development and operations teams.
2. Automates software delivery process, making it faster and more efficient.
3. Enables faster recovery and response time in case of issues.

Disadvantages:

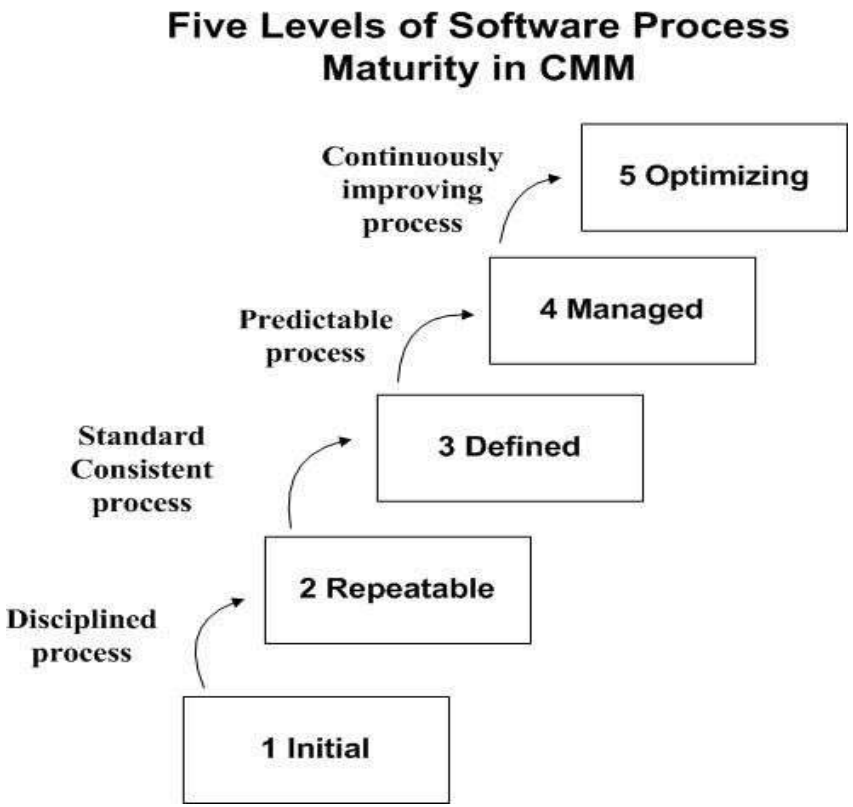
1. Requires a significant investment in tools and technologies.
2. Lack of culture of collaboration.
3. Need to have a skilled workforce to effectively implement the devops practices.

Capability Maturity Model (CMM)



- Capability Maturity Model (CMM) specifies an increasing series of levels of a software development organization.
- The higher the level, the better the software development process
- reaching each level is an expensive and time-consuming process.

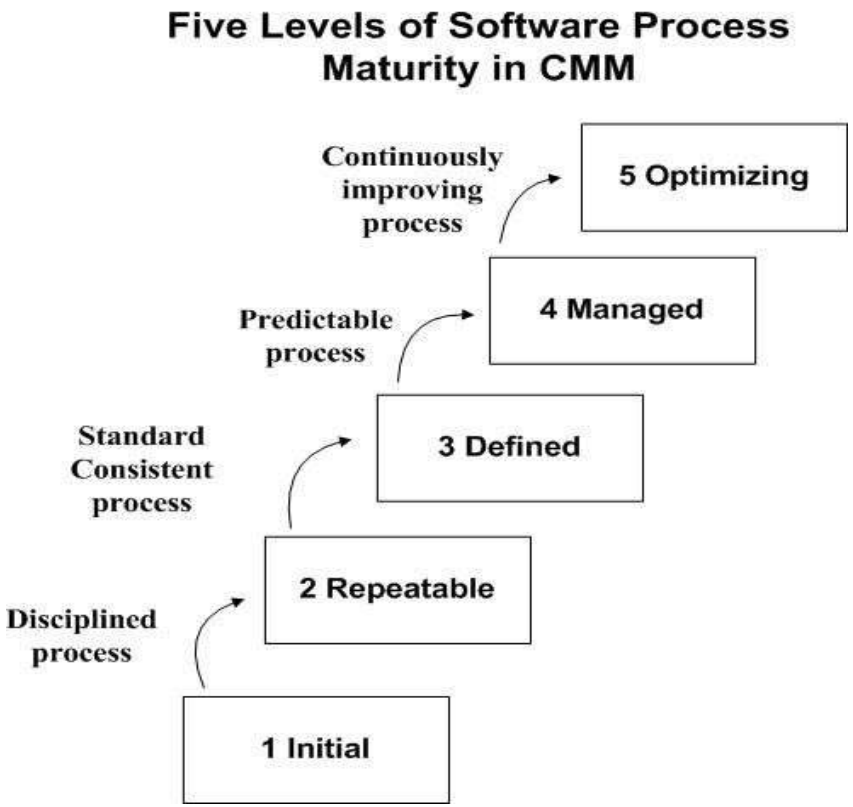
Capability Maturity Model (CMM)



Level One :Initial

- No KPA defined.
- Processes followed are Ad Hoc and immature and are not well defined.
- Unstable environment for software development.
- No basis for predicting product quality, time for completion, etc.
- Limited project management capabilities, such as no systematic tracking of schedules, budgets, or progress.
- We have limited communication and coordination among team members and stakeholders.
- No formal training or orientation for new team members.
- Little or no use of software development tools or automation.
- Highly dependent on individual skills and knowledge rather **than** standardized processes.
- High risk of project failure or delays due to a lack of process control and stability.

Capability Maturity Model (CMM)



Level Two: Repeatable

Focuses on establishing basic project management policies.

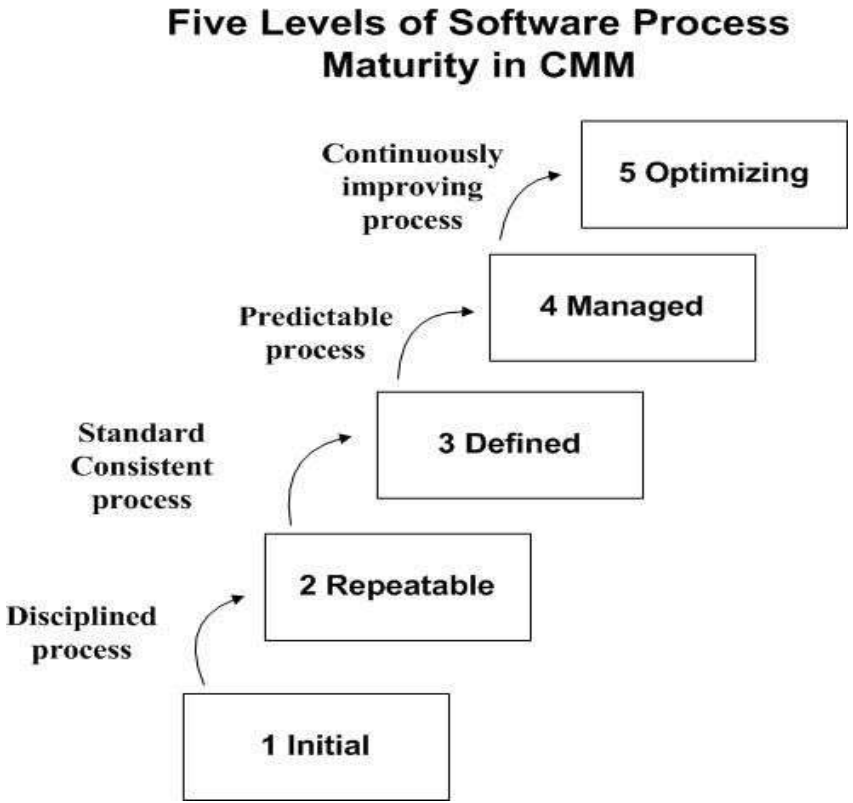
Experience with earlier projects is used for managing new similar-natured projects.

Project Planning- It includes defining resources required, goals, constraints, etc. for the project. It presents a detailed plan to be followed systematically for the successful completion of good-quality software.

Configuration Management- The focus is on maintaining the performance of the software product, including all its components, for the entire lifecycle.

Requirements Management- It includes the management of customer reviews and feedback which result in some changes in the requirement set. It also consists of accommodation of those modified requirements.

Capability Maturity Model (CMM)

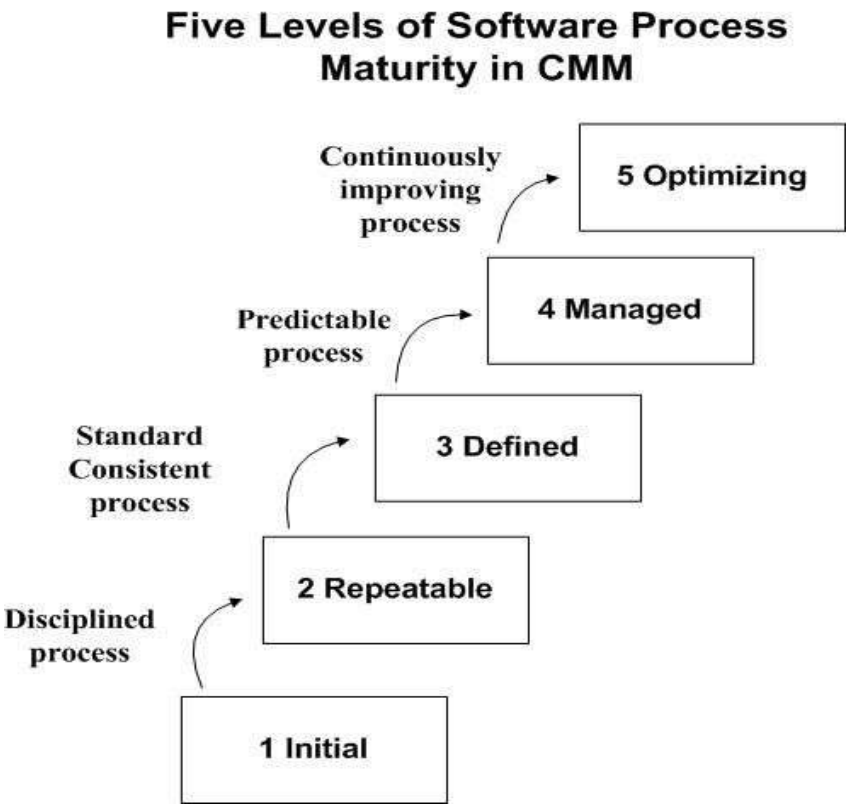


Level Two: Repeatable

Subcontract Management- It focuses on the effective management of qualified software contractors i.e. it manages the parts of the software developed by third parties.

Software Quality Assurance- It guarantees a good quality software product by following certain rules and quality standard guidelines while developing.

Capability Maturity Model (CMM)

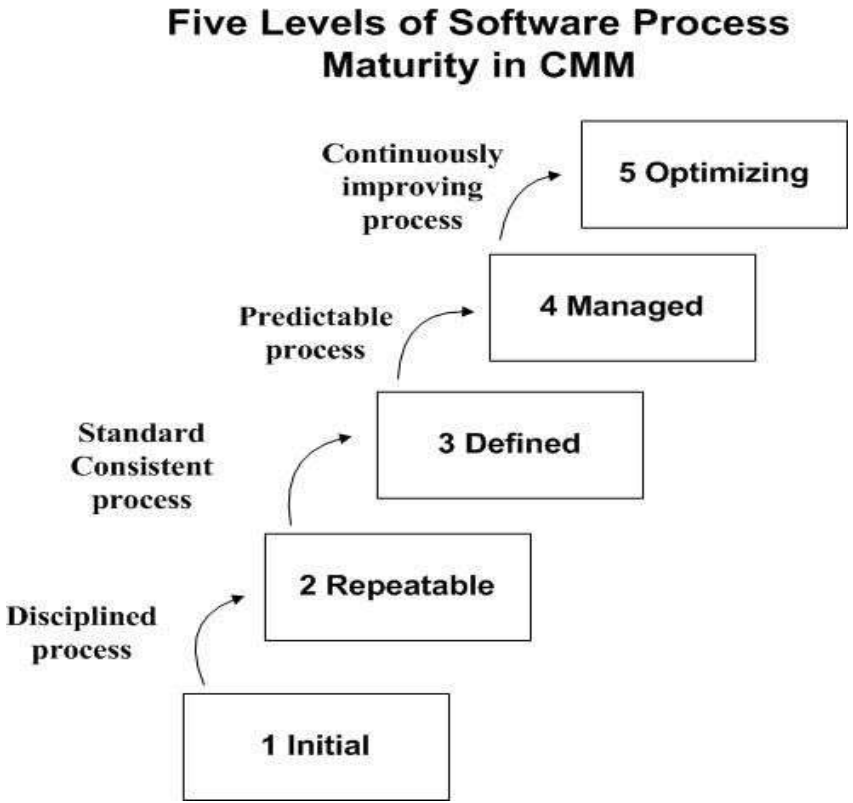


Level Three: Defined

At this level, documentation of the standard guidelines and procedures takes place.

- It is a well-defined integrated set of project-specific software engineering and management processes.
- **Peer Reviews:** In this method, defects are removed by using a number of review methods like walkthroughs, inspections, buddy checks, etc.
- **Intergroup Coordination:** It consists of planned interactions between different development teams to ensure efficient and proper fulfillment of customer needs.
- **Organization Process Definition:** Its key focus is on the development and maintenance of standard development processes.
- **Organization Process Focus:** It includes activities and practices that should be followed to improve the process capabilities of an organization.
- **Training Programs:** It focuses on the enhancement of knowledge and skills of the team members including the developers and ensuring an increase in

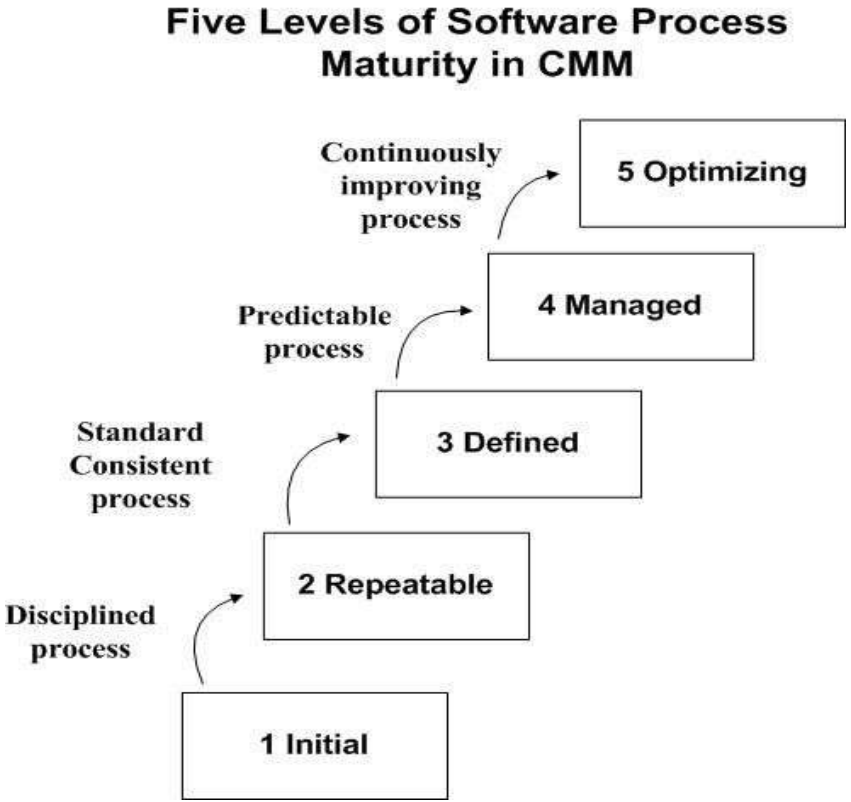
Capability Maturity Model (CMM)



Level Four: Managed

- At this stage, quantitative quality goals are set for the organization for software products as well as software processes.
- The measurements made help the organization to predict the product and process quality within some limits defined quantitatively.
- **Software Quality Management:** It includes the establishment of plans and strategies to develop quantitative analysis and understanding of the product's quality.
- **Quantitative Management:** It focuses on controlling the project performance in a quantitative manner.

Capability Maturity Model (CMM)



Level Five: Optimizing

This is the highest level of process maturity in CMM and focuses on continuous process improvement in the organization using quantitative feedback.

The use of new tools, techniques, and evaluation of software processes is done to prevent the recurrence of known defects.

- **Process Change Management:** Its focus is on the continuous improvement of the organization's software processes to improve productivity, quality, and cycle time for the software product.
- **Technology Change Management:** It consists of the identification and use of new technologies to improve product quality and decrease product development time.
- **Defect Prevention** It focuses on the identification of causes of defects and prevents them from recurring in future projects by improving project-defined processes.

Capability Maturity Model (CMM)

CMM Level	Focus	Key Process Areas
1. Initial	Competent People	NO KPA'S
2. Repeatable	Project Management	Software Project Planning software Configuration Management
3. Defined	Definition of Processes	Process definition Training Program Peer reviews
4. Managed	Product and Process quality	Quantitiative Process Metrics Software Quality Management
5. Optimizing	Continuous Process improvement	Defect Prevention Process change management Technology change management

The focus of each SEI CMM level and the Corresponding Key process areas.

Waterfall Model

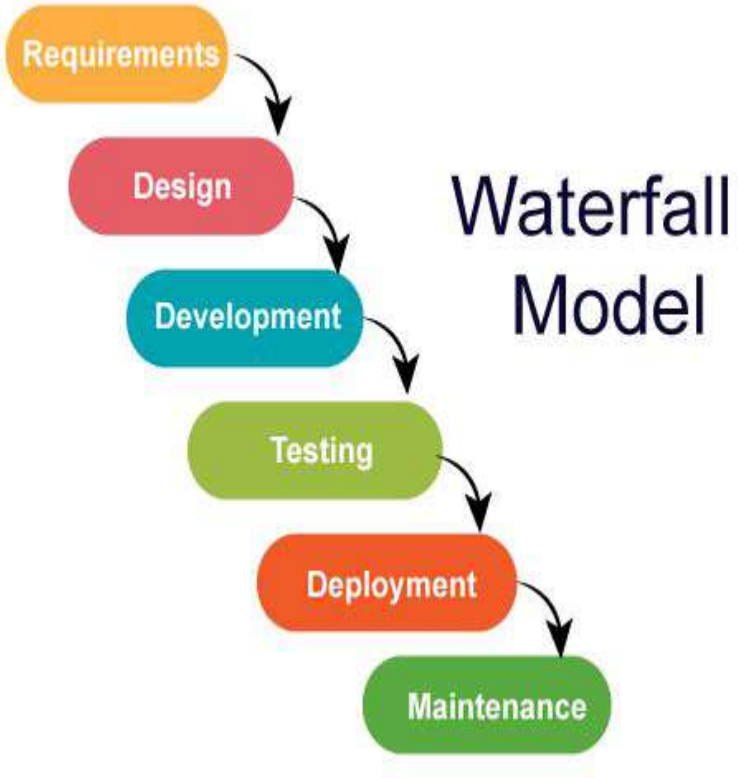
- basic software development life cycle model.
- It is very simple but idealistic.
- Important because all the other software development life cycle models are based on the classical waterfall model.
- It is characterized by a structured, sequential approach to project management and software development.
- The waterfall model is useful in situations where the project requirements are well-defined and the project goals are clear.

Waterfall Model

Features of Waterfall Model

1. **Sequential Approach**: The waterfall model involves a sequential approach to software development, where each phase of the project is completed before moving on to the next one.
2. **Document-Driven:** The waterfall model relies heavily on documentation to ensure that the project is well-defined and the project team is working towards a clear set of goals.
3. **Quality Control:** The waterfall model places a high emphasis on quality control and testing at each phase of the project, to ensure that the final product meets the requirements and expectations of the stakeholders.
4. **Rigorous Planning**: The waterfall model involves a rigorous planning process, where the project scope, timelines, and deliverables are carefully defined and monitored throughout the project lifecycle.

Waterfall Model



The Waterfall Model has six phases:

1. Requirements Gathering and Analysis: The first phase involves gathering requirements from stakeholders and analyzing them to understand the scope and objectives of the project.

2. Design Phase: Once the requirements are understood, the design phase begins. This involves creating a detailed design document that outlines the software architecture, user interface, and system components.

3. Implementation and Unit Testing: The implementation phase involves coding the software based on the design specifications. This phase also includes unit testing to ensure that each component of the software is working as expected.

4. Integration and System Testing: In the testing phase, the software is tested as a whole to ensure that it meets the requirements and is free from defects.

5. Deployment: Once the software has been tested and approved, it is deployed to the production environment.

6. Maintenance: The final phase of the Waterfall Model is maintenance, which involves fixing any issues that arise after the software has been deployed and ensuring that it continues to meet the requirements over time.

Waterfall Model

Advantages

- **Easy to Understand:** Classical Waterfall Model is very simple and easy to understand.
- **Individual Processing:** Phases in the Classical Waterfall model are processed one at a time.
- **Properly Defined:** In the classical waterfall model, each stage in the model is clearly defined.
- **Clear Milestones:** Classical Waterfall model has very clear and well-understood milestones.
- **Properly Documented:** Processes, actions, and results are very well documented.
- **Reinforces Good Habits:** Classical Waterfall Model reinforces good habits like define-before-design and design-before-code.
- **Working:** Classical Waterfall Model works well for smaller projects and projects where requirements are well understood.

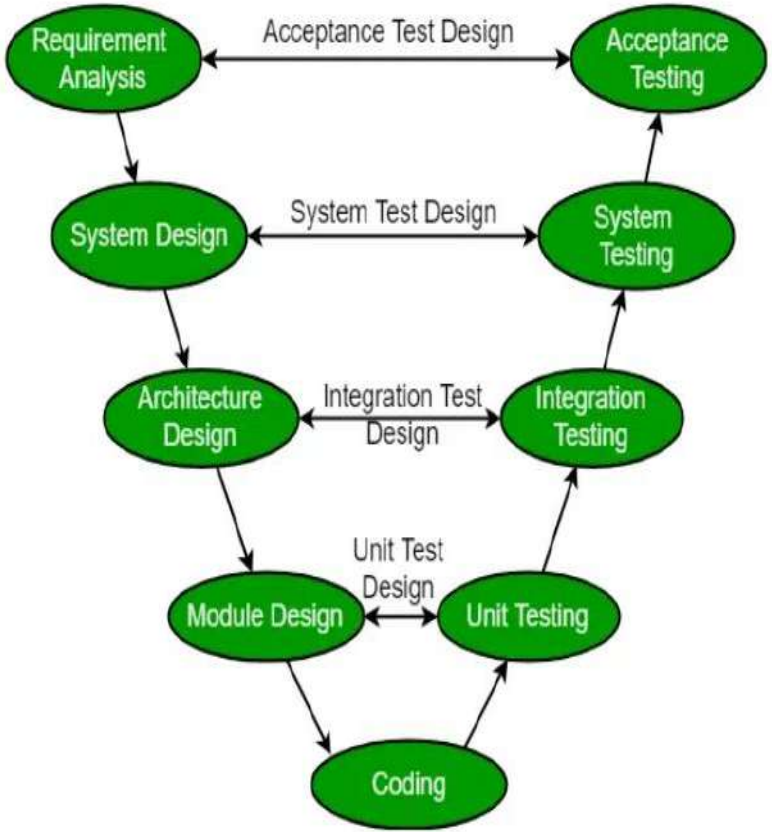
Waterfall Model

Disadvantages:

- No Feedback Path
- Difficult to accommodate Change Requests
- No Overlapping of Phases
- Limited Flexibility
- Limited Stakeholder Involvement
- Late Defect Detection
- Lengthy Development Cycle
- Not Suitable for Complex Projects

V Model

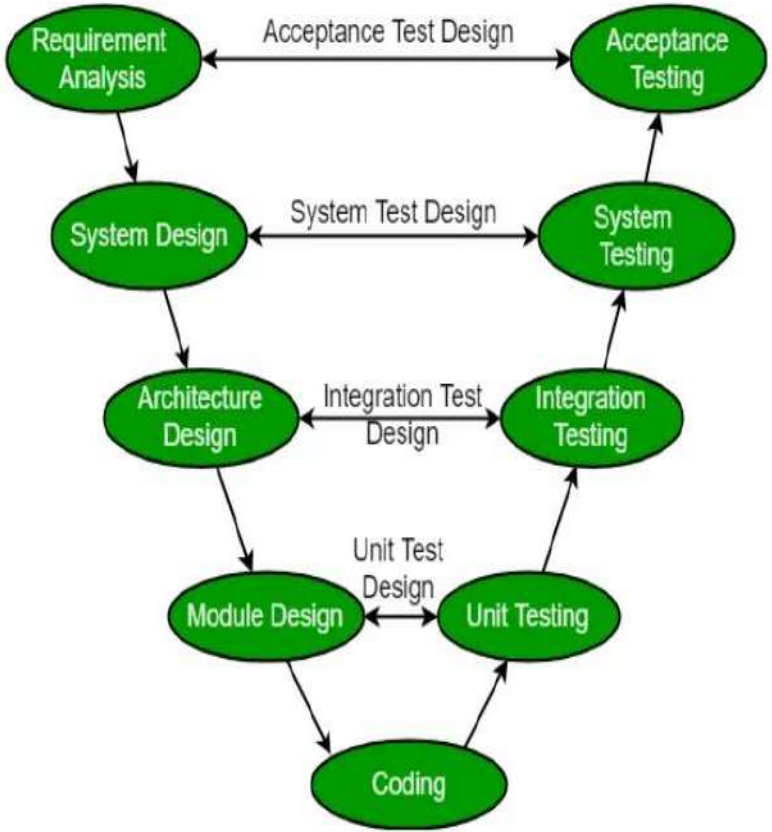
- referred to as the Verification and Validation Model
- each phase of SDLC must complete before the next phase starts
- The structure it follows takes the shape of the letter V.
- Verification
  - Requirement analysis
  - System Design
  - Architectural design
  - Module design
  - coding
- Validation
  - Unit testing
  - Integration testing
  - System testing
  - Acceptance testing



V Model- Verification Phase

Business Requirement Analysis:

- first step where product requirement needs to be cure with the customer perspectives
- include the proper communication with the customer to understand the requirement of the customers.
- it will be used as an input for acceptance testing.



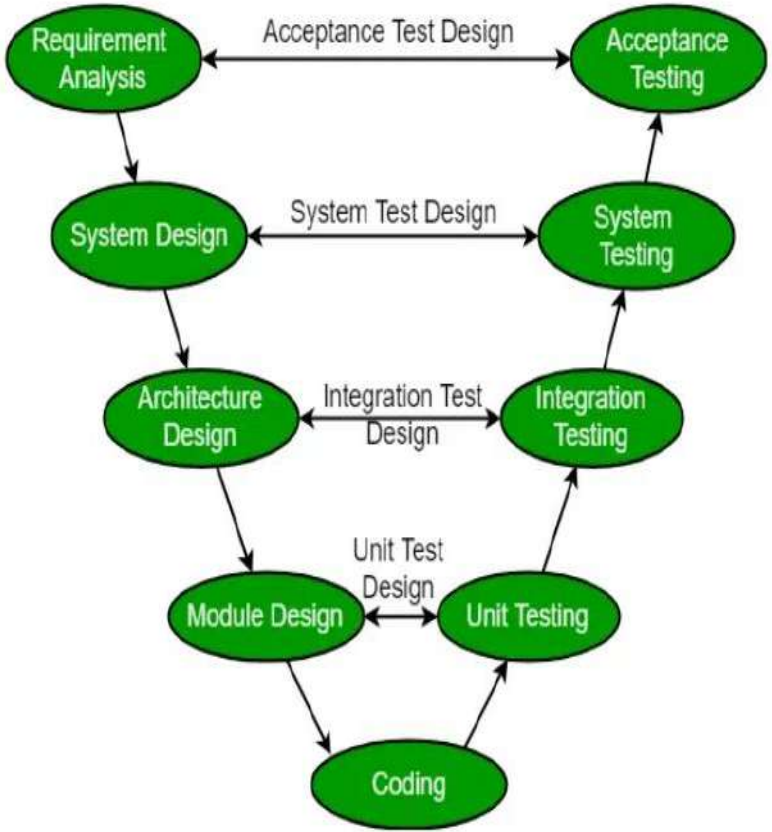
V Model- Verification Phase

System Design:

- Design of system will start when the overall we clear with the product requirements, then need to design the system completely.
- the system is designed with the entire hardware & the setup is constructed for product development.

Architectural Design:

- The breakdown of system design to a more detailed version, i.e., into modules which creates different functionalities.
- Transferring of data and connection between internal and external modules (i.e., the outside world) is evidently identified.



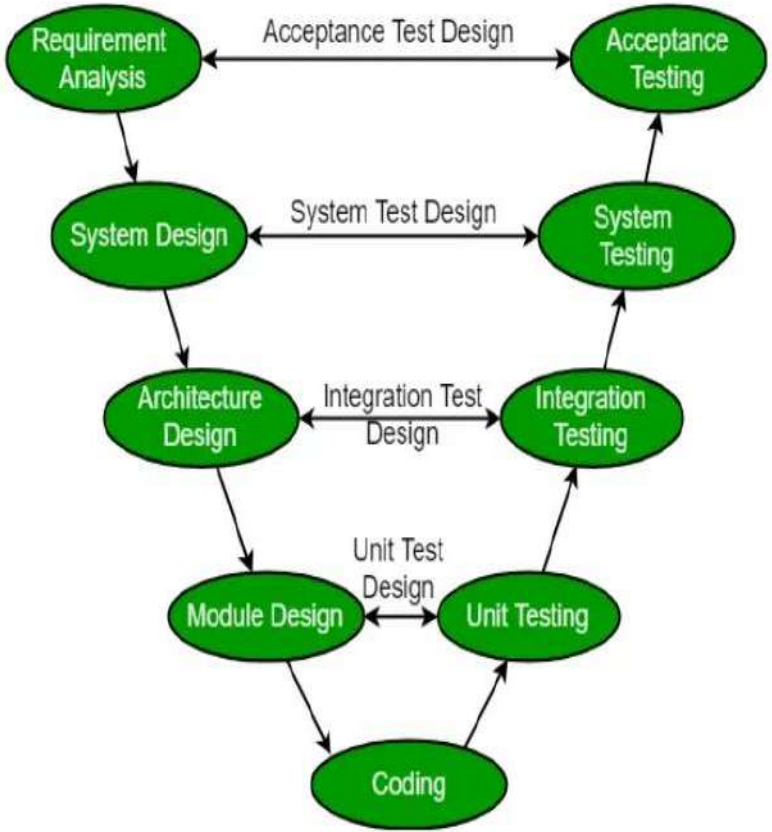
V Model- Verification Phase

Module Design

- In the module design phase, the system breaks down into small modules.
- The detailed design of the modules is specified, which is known as Low-Level Design

Coding Phase

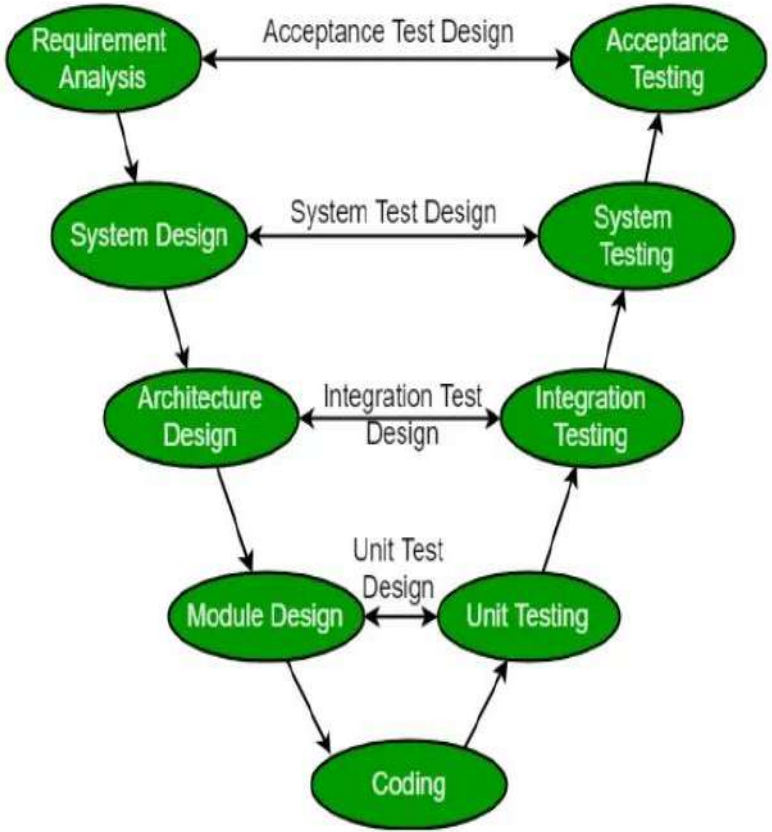
- After designing, the coding phase is started.
- Based on the requirements, a suitable programming language is decided.
- There are some guidelines and standards for coding.
- the code goes through many code reviews to check the performance.



V Model- Validation Phase

Unit Testing

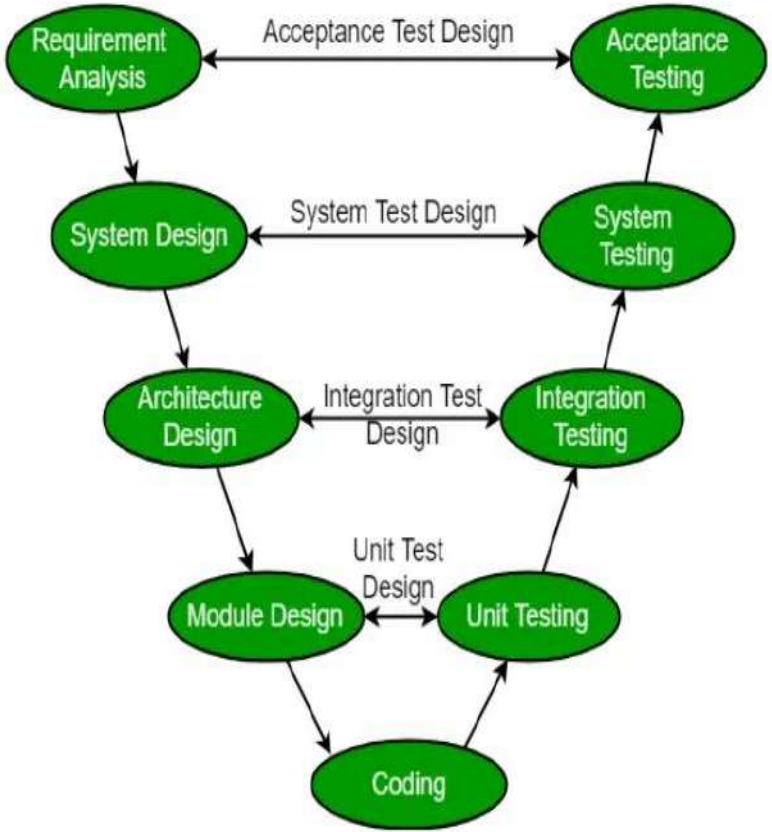
- Unit Test Plans (UTPs) are developed during the module design phase.
- These UTPs are executed to eliminate errors at code level or unit level.
- A unit is the smallest entity which can independently exist, e.g., a program module.
- Unit testing verifies that the smallest entity can function correctly when isolated from the rest of the codes/ units.



V Model- Validation Phase

Unit Testing

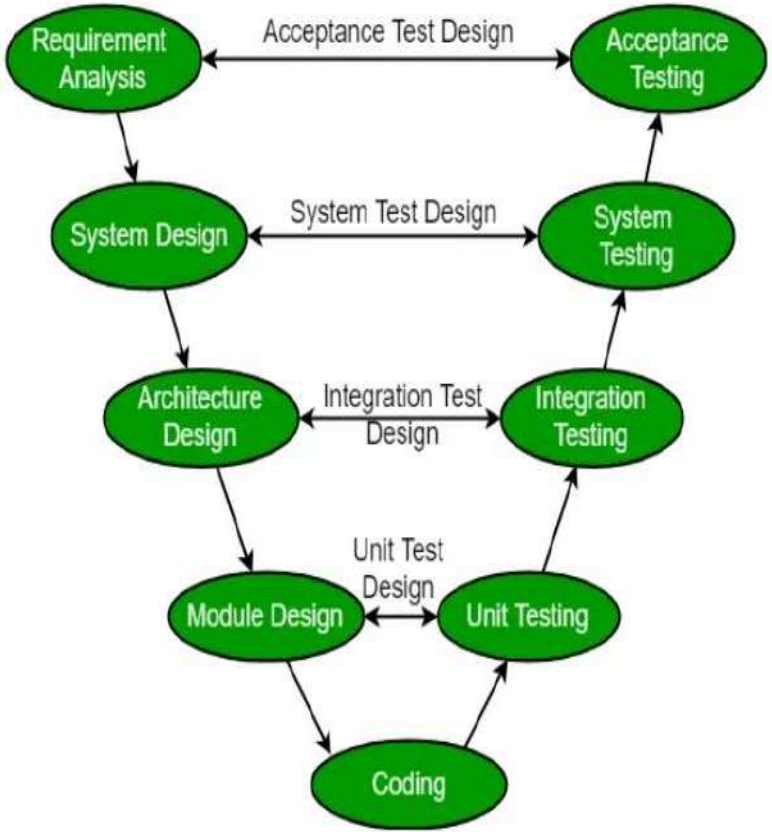
- Unit Test Plans (UTPs) are developed during the module design phase.
- These UTPs are executed to eliminate errors at code level or unit level.
- A unit is the smallest entity which can independently exist, e.g., a program module.
- Unit testing verifies that the smallest entity can function correctly when isolated from the rest of the codes/ units.



V Model- Validation Phase

Integration Testing

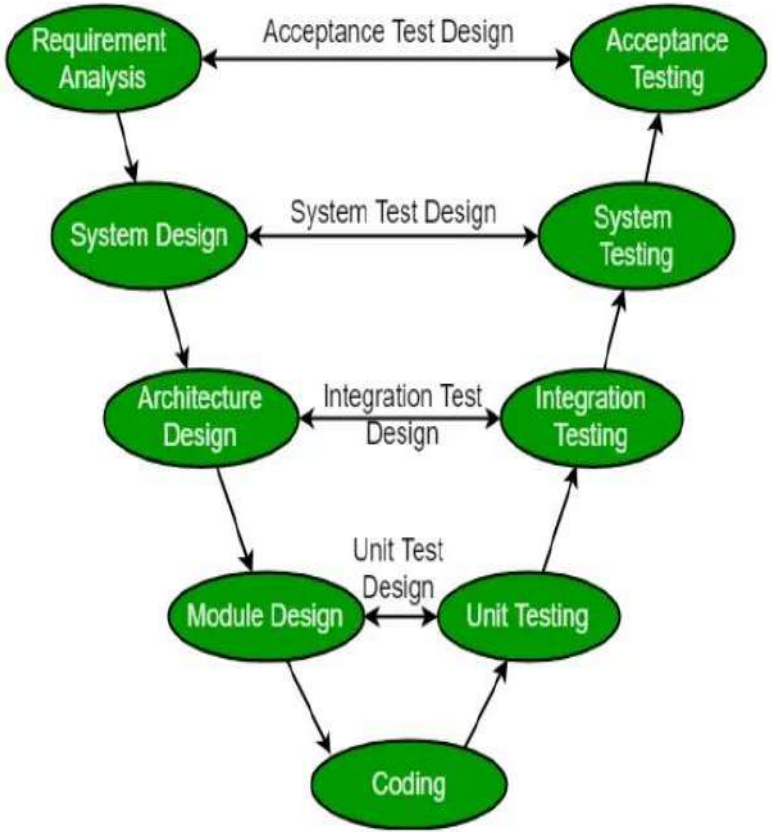
- Integration Test Plans are developed during the Architectural Design Phase.
- These tests verify that groups created and tested independently can coexist and communicate among themselves.



V Model- Validation Phase

System Testing

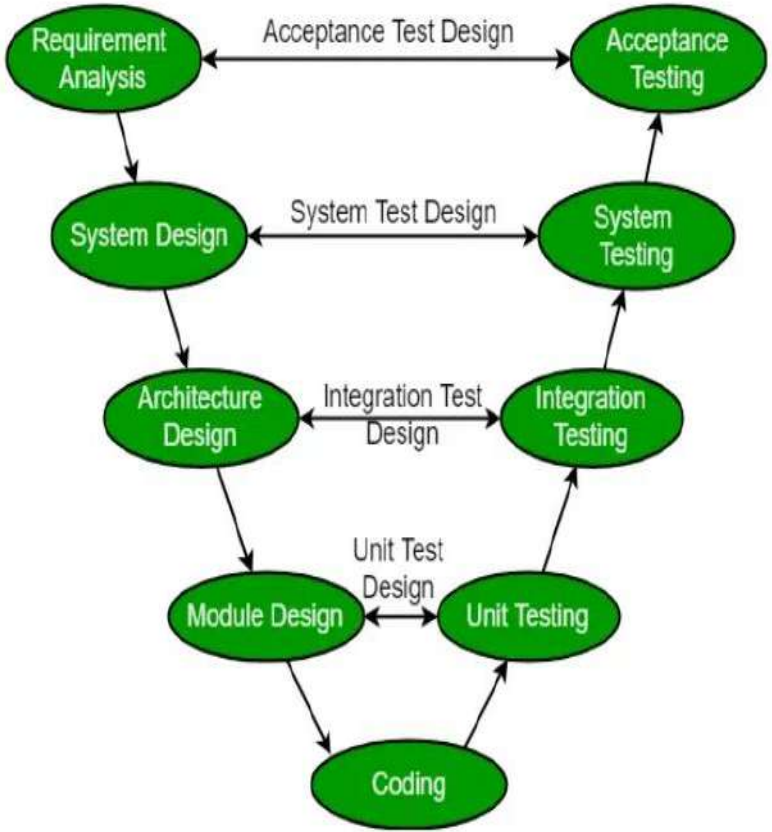
- System Tests Plans are developed during System Design Phase.
- System Test ensures that expectations from an application developer are met.



V Model- Validation Phase

Acceptance Testing

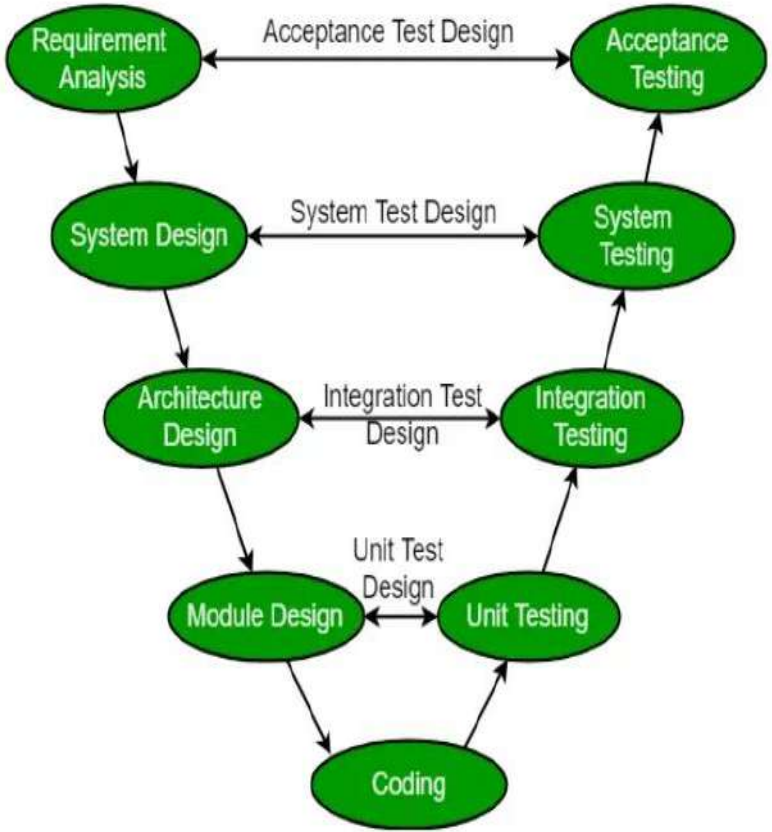
- Acceptance testing is related to the business requirement analysis part.
- It includes testing the software product in user atmosphere.
- Acceptance tests reveal the compatibility problems with the different systems, which is available within the user atmosphere.
- It conjointly discovers the non-functional problems like load and performance defects within the real user atmosphere.



V Model

When to use?

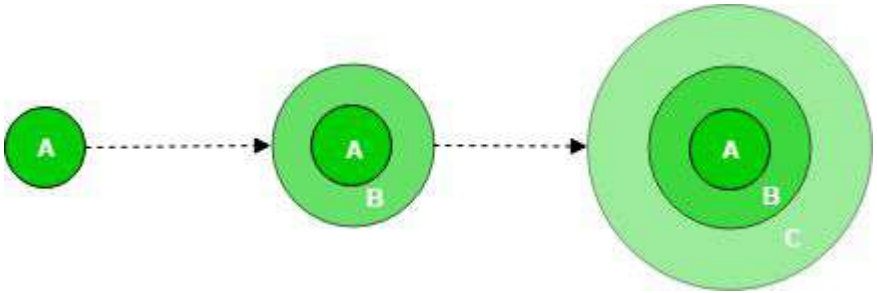
- When the requirement is well defined and not ambiguous.
- The V-shaped model should be used for small to medium-sized projects where requirements are clearly defined and fixed.
- The V-shaped model should be chosen when sample technical resources are available with essential technical expertise.



Incremental Process Model

First, a simple working system implementing only a few basic features is built and then that is delivered to the customer.

Then thereafter many successive iterations/ versions are implemented and delivered to the customer until the desired system is released.



A, B, and C are modules of Software Products that are incrementally developed and delivered.

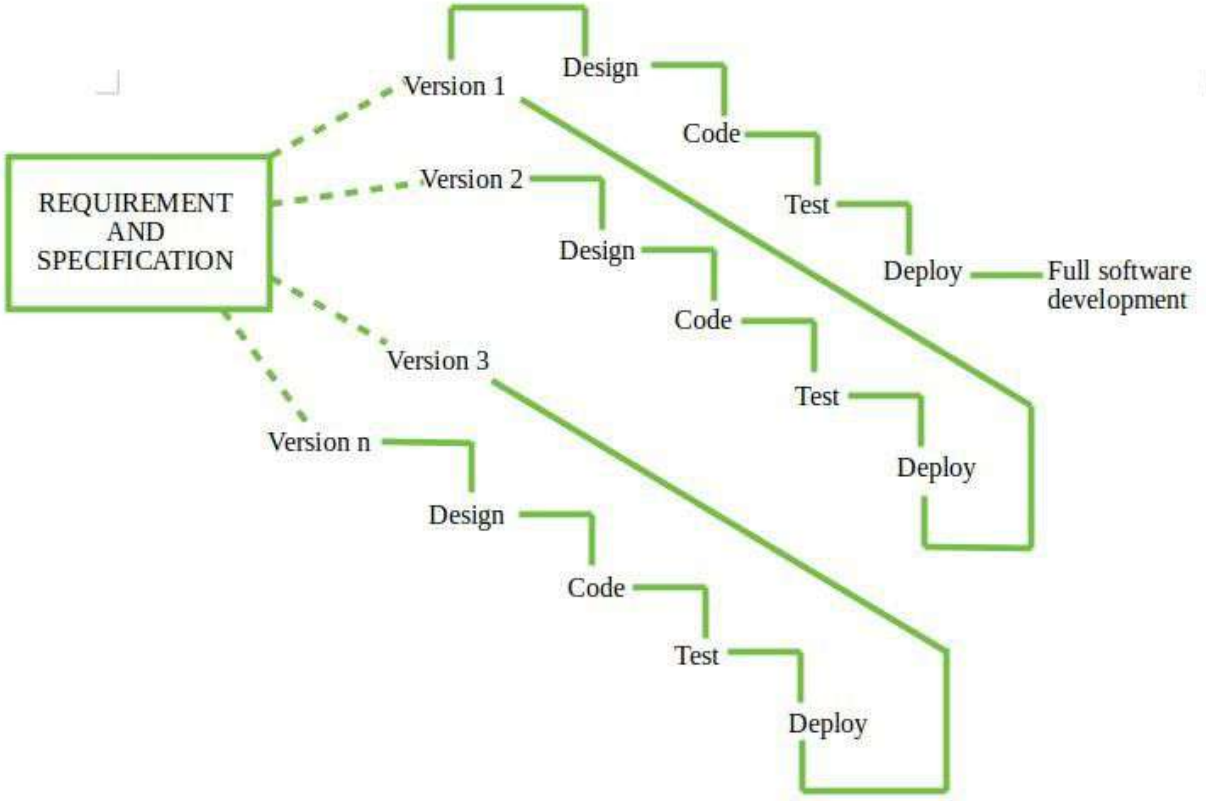
Incremental Process Model: Life Cycle Activities

- In the incremental model, the software is divided into different modules that can be developed and delivered incrementally.
- Since the plan just considers the next increment, there aren't really any long-term plans throughout this period.
- This makes it possible to alter a version in order to meet users' demands.
- During the development life cycle, the development team creates the system's fundamental components first before adding new functionalities in successive versions.
- Once these features are successfully implemented, they're then improved by including new functionality in later editions to increase the functionalities of the product.
- This process is iterative and follows a waterfall model, with each version building upon the previous one.

Incremental Process Model: Life Cycle Activities

- As the software is developed, each new version is delivered to the customer site for testing and feedback, which is taken into account to improve the version.
- Once the collection and specification of the requirements are completed, the development process is then separated into different versions.
- Starting with version 1, each version is developed and then deployed to the client site after each successful increment.

Incremental Process Model



Incremental Process Model

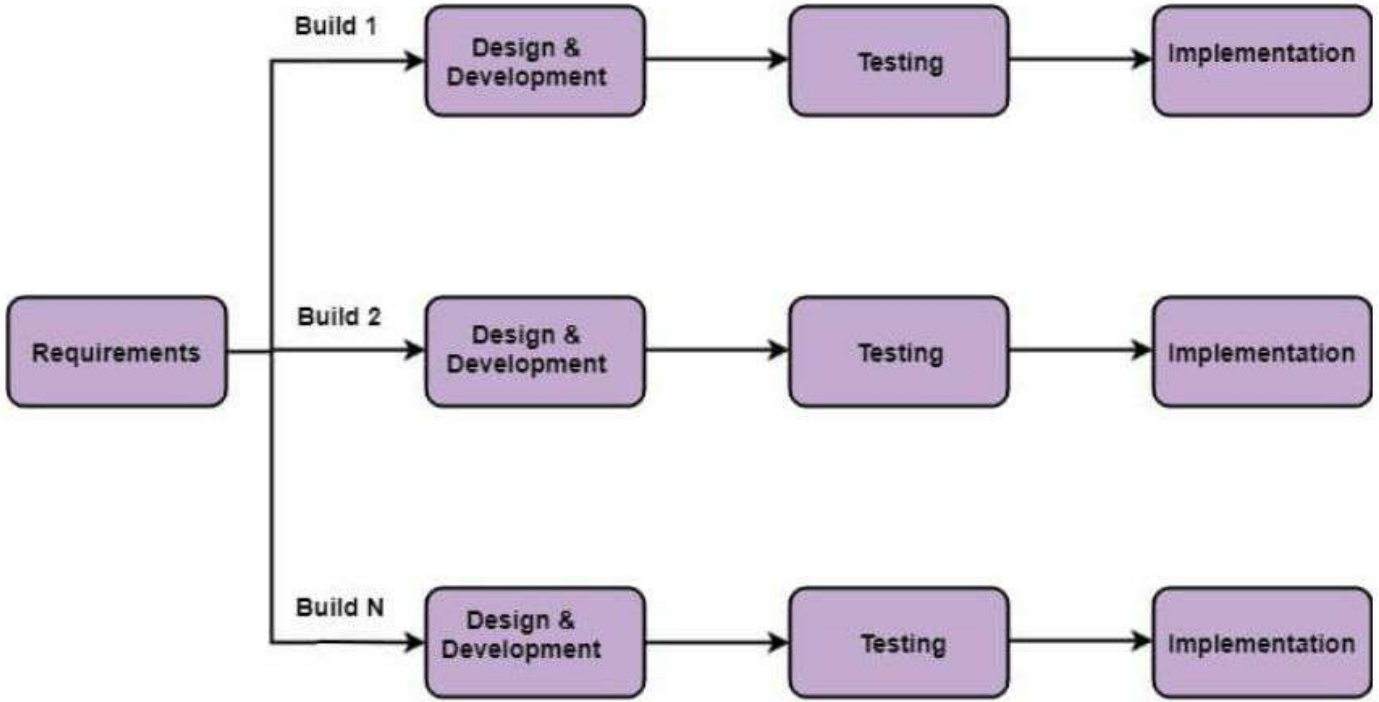


Fig: Incremental Model

Incremental Process Model: Phases

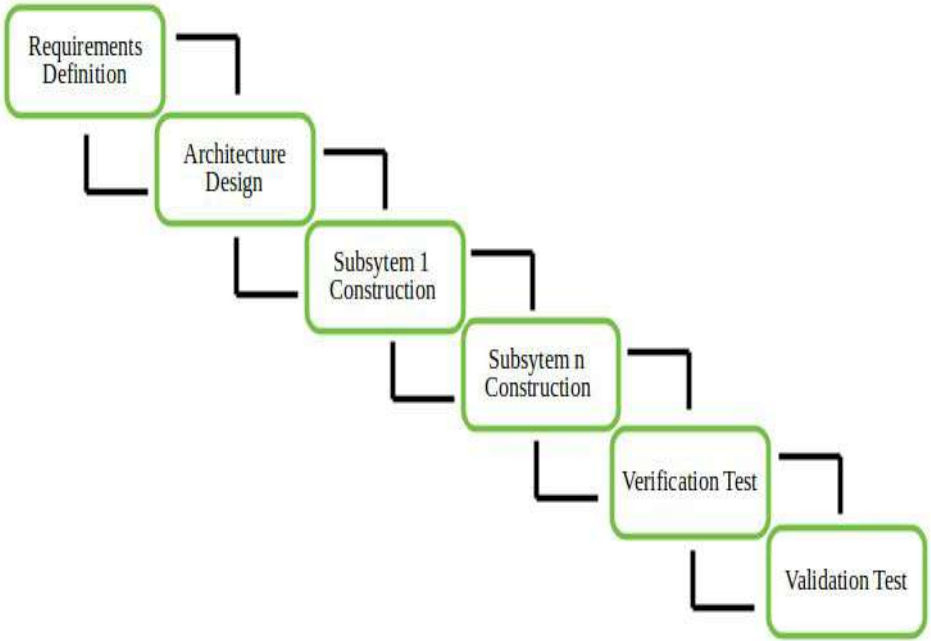
- Requirements Gathering: In this initial phase, the high-level requirements for the software are gathered and analyzed. These requirements serve as a foundation for the subsequent phases.
- Design: Based on the gathered requirements, the software's architecture, design, and user interfaces are planned and developed. The design is often divided into smaller segments to ensure a focused and organized development process.
- Implementation: Each increment involves implementing a portion of the software's functionality. Developers work on adding features, modules, or components that were planned in the design phase. This incremental approach allows for quicker delivery of usable software.
- Testing: As each increment is completed, testing is carried out to ensure that the new features work as expected and do not negatively impact the existing functionality. This ongoing testing helps catch and address issues early in the development process.

Types of Incremental Model

1. Staged Delivery Model
2. Parallel Development Model

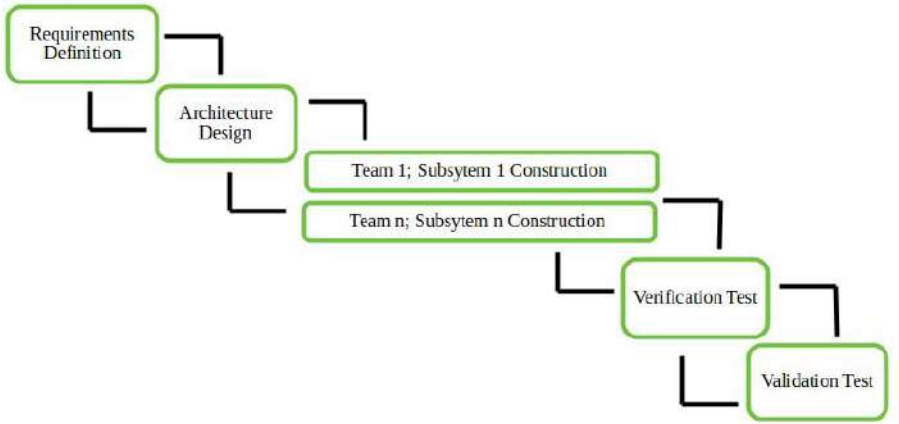
Staged Delivery Model

Construction of only one part of the project at a time.



Parallel Development Model

Different subsystems are developed at the same time. It can decrease the calendar time needed for the development if enough resources are available.



When to Use an Incremental Model

The incremental model can be used when:

- the objectives of the entire system are clearly stated and recognized, though some details can evolve at each increment over time.
- there's a pressing need to get a product to market as soon as possible.
- new technology is being deployed.
- the software team is inexperienced or unskilled.
- a project's development timeline is prolonged.
- there are some high-risk features and goals.
- there are no resources with the necessary skills.

Evolutionary Process Models

- Evolutionary models are iterative type models.
- They allow to develop more complete versions of the software.
- Following are the evolutionary process models.

  1. The prototyping model
  2. The spiral model

The Prototyping model

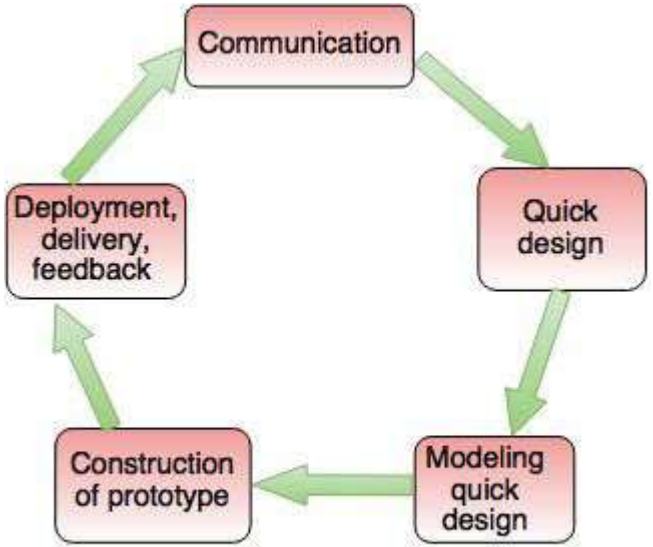


Fig. - The Prototyping Model

- Prototype is defined as first or preliminary form using which other forms are copied or derived.
- Prototype model is a set of general objectives for software.
- It does not identify the requirements like detailed input, output.
- It is software working model of limited functionality.

The Prototyping model

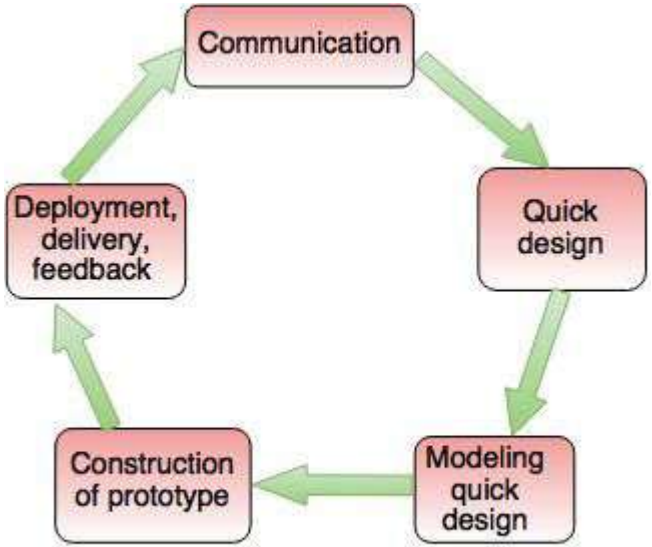


Fig. - The Prototyping Model

The different phases of Prototyping model are:

1. Communication

- In this phase, developer and customer meet and discuss the overall objectives of the software.

2. Quick design

- Quick design is implemented when requirements are known.
- It includes only the important aspects like input and output format of the software.
- It focuses on those aspects which are visible to the user rather than the detailed plan.
- It helps to construct a prototype.

The Prototyping model

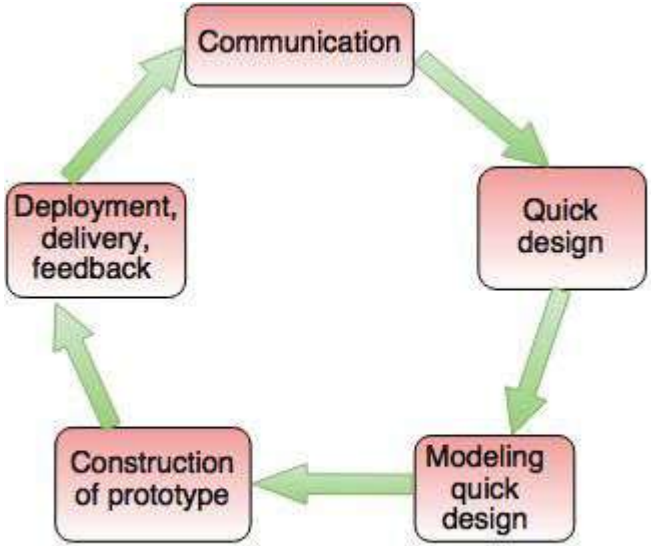


Fig. - The Prototyping Model

The different phases of Prototyping model are:

3. Modeling quick design

- This phase gives the clear idea about the development of software
- It allows the developer to better understand the exact requirements.

4. Construction of prototype

- The prototype is evaluated by the customer itself.

The Prototyping model

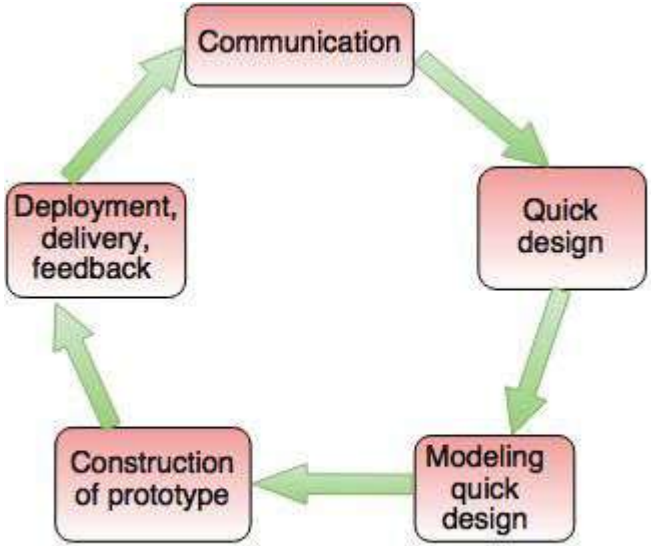


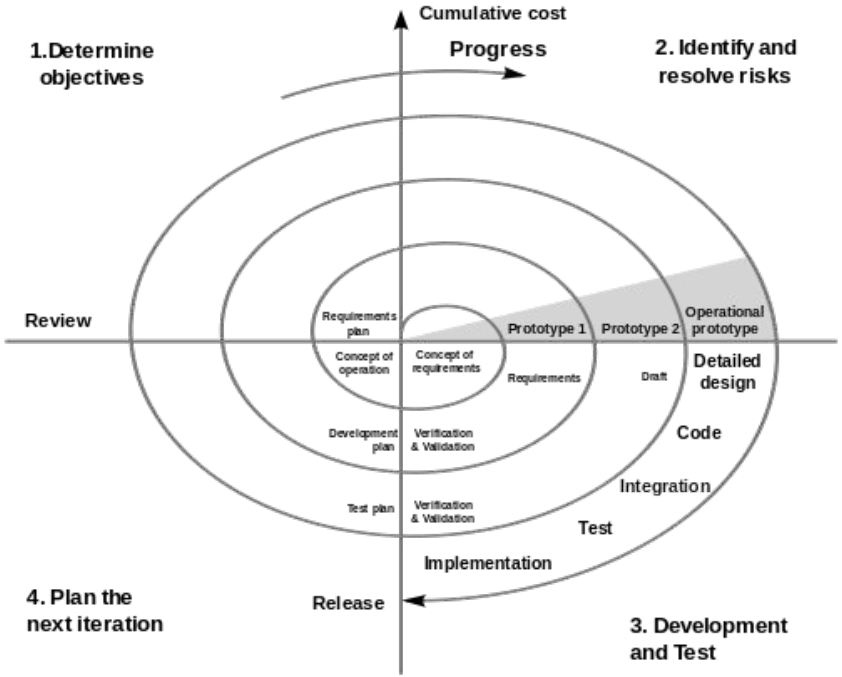
Fig. - The Prototyping Model

The different phases of Prototyping model are:

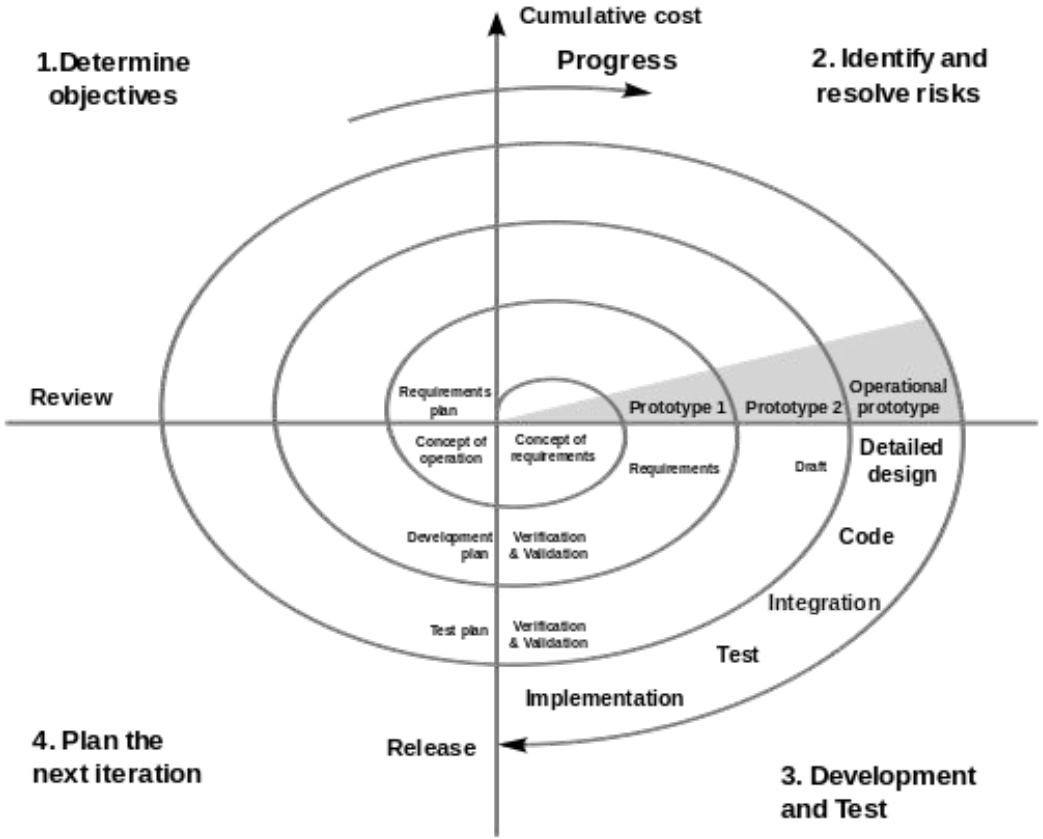
5. Deployment, delivery, feedback

- If the user is not satisfied with current prototype then it refines according to the requirements of the user.
- The process of refining the prototype is repeated until all the requirements of users are met.
- When the users are satisfied with the developed prototype then the system is developed on the basis of final prototype.

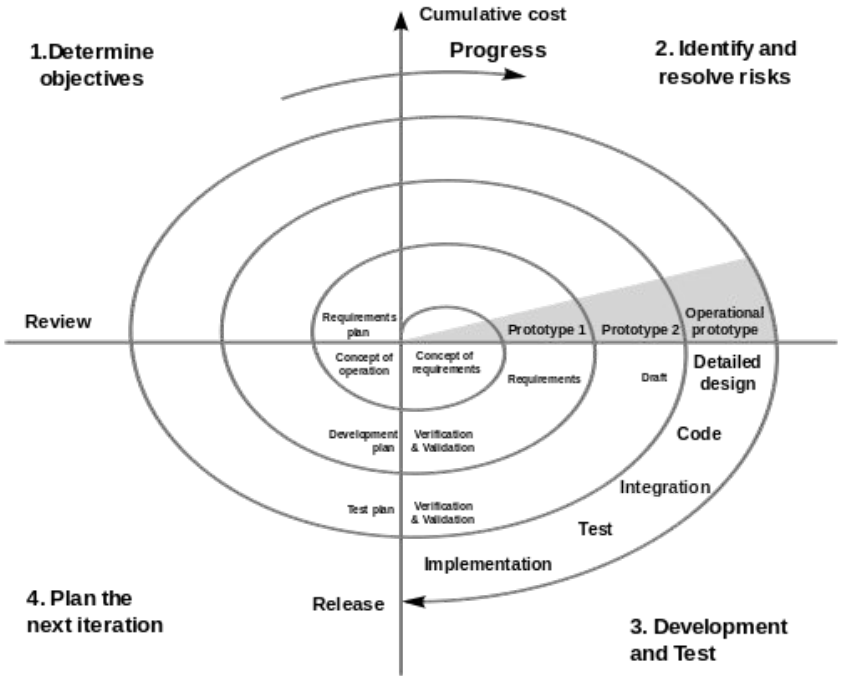
Spiral Model



- provides a systematic and iterative approach to software development. In its diagrammatic representation,
- looks like a spiral with many loops.
- The exact number of loops of the spiral is unknown and can vary from project to project.
- It is based on the idea of a spiral, with each iteration of the spiral representing a complete software development cycle, from requirements gathering and analysis to design, implementation, testing, and maintenance.


Cumulative cost
1.Determine objectives
Progress
2. Identify and resolve risks
Review
Requirements plan
Prototype 1
Prototype 2
Operational prototype
Concept of operation
Concept of requirements
Requirements
Draft
Detailed design
Development plan
Verification & Validation
Code
Integration
Test plan
Verification & Validation
Test
Implementation
4. Plan the next iteration
Release
3. Development and Test

Spiral Model



Phases:

1. Determine Objective

The first phase of the Spiral Model where the scope of the project is determined

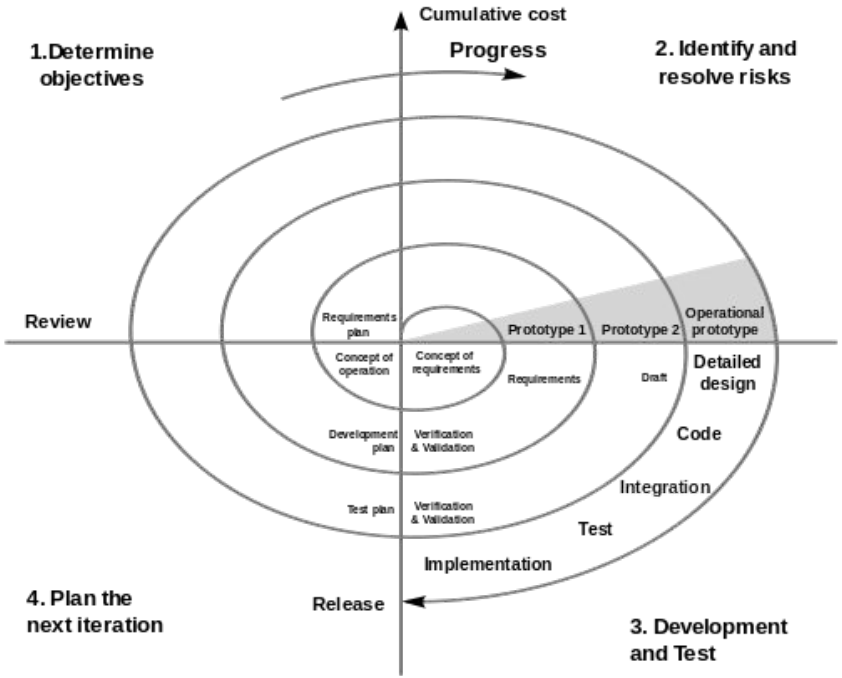
2. Risk Analysis

In the risk analysis phase, the risks associated with the project are identified and evaluated.

3. Engineering

In the engineering phase, the software is developed based on the requirements gathered in the previous iteration.

Spiral Model



Phases:

4. Evaluation

In the evaluation phase, the software is evaluated to determine if it meets the customer's requirements and if it is of high quality.

5. Planning

The next iteration of the spiral begins with a new planning phase, based on the results of the evaluation.

Spiral Model

- The Spiral Model is often used for complex and large software development projects, as it allows for a more flexible and adaptable approach to software development.
- It is also well-suited to projects with significant uncertainty or high levels of risk.
- The Radius of the spiral at any point represents the expenses(cost) of the project so far
- the angular dimension represents the progress made so far in the current phase.

Concurrent Model

- Type of evolutionary model
- Also known as concurrent engineering
- Concurrent means done at the same time
- Used in all SDLC
- Transition from state to state for each of the software engineering activities

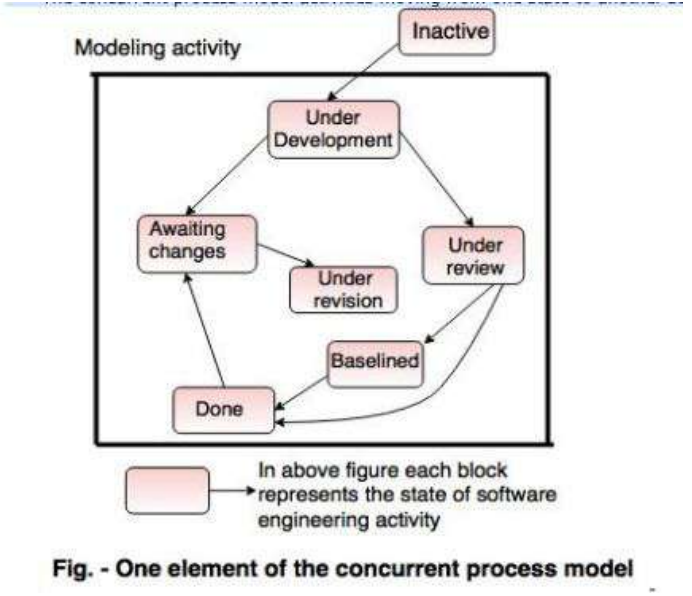


Fig. - One element of the concurrent process model

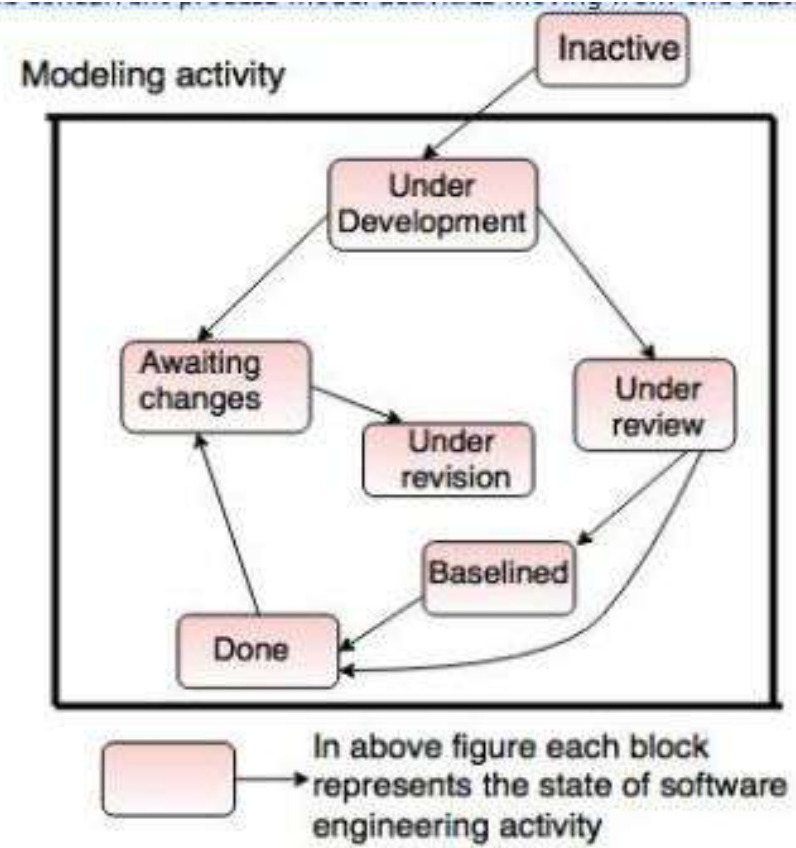


Fig. - One element of the concurrent process model

Concurrent Model

- SDLC Activities: Communication, Design, Coding, Testing, Deployment
- Inactive: no any activity/state performed
- Under development: any activity performed
- Awaiting changes: if customer want any changes
- Under review: do all required changes
- Baselined: as per the SRS document
- Done : project completed and deployed

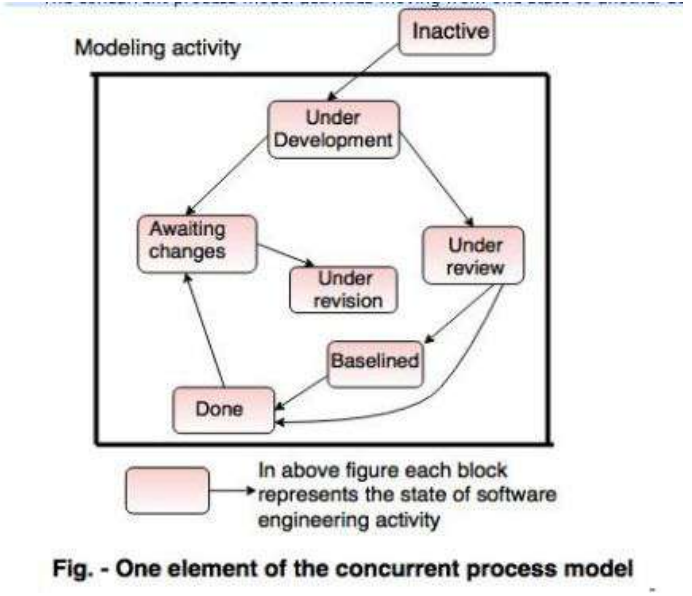


Fig. - One element of the concurrent process model

Concurrent Model

- The concurrent development model is called as concurrent model.
- The communication activity has completed in the first iteration and exits in the awaiting changes state.
- The modeling activity completed its initial communication and then go to the underdevelopment state.
- If the customer specifies the change in the requirement, then the modeling activity moves from the under development state into the awaiting change state.
- The concurrent process model activities moving from one state to another state

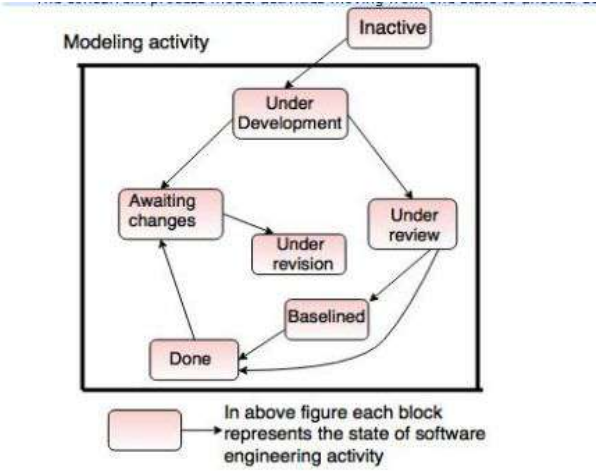


Fig. - One element of the concurrent process model

Concurrent Model

Advantages:

- Applicable to all types of software development processes.
- Easy for understanding and use.
- Immediate feedback from testing
- Provides an accurate picture of the current state of the project

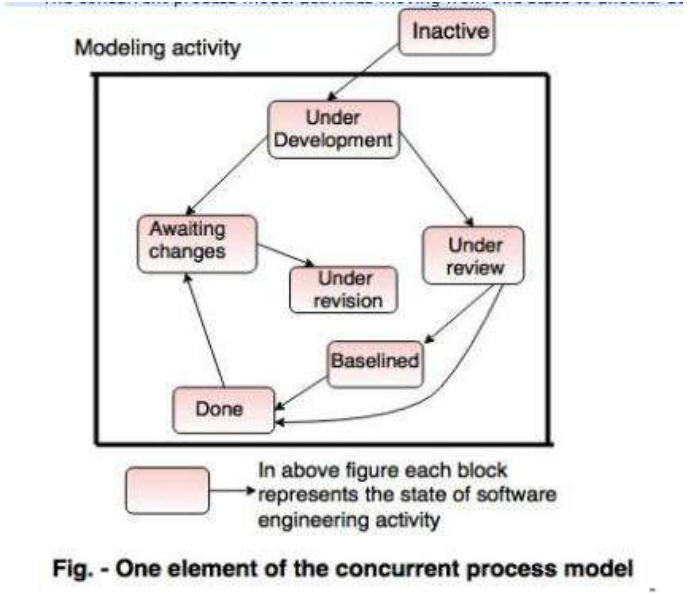


Fig. - One element of the concurrent process model

Concurrent Model

Disadvantages:

- It needs better communication between the team members. This may not be achieved all the time.
- It requires to remember the status of the different activities.

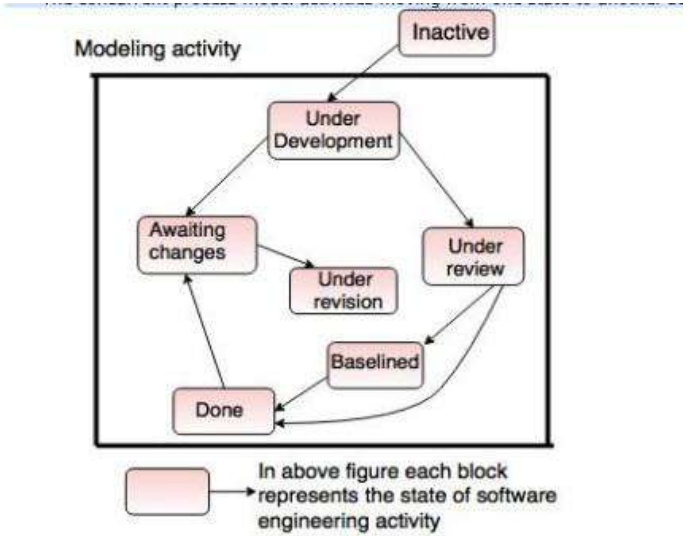


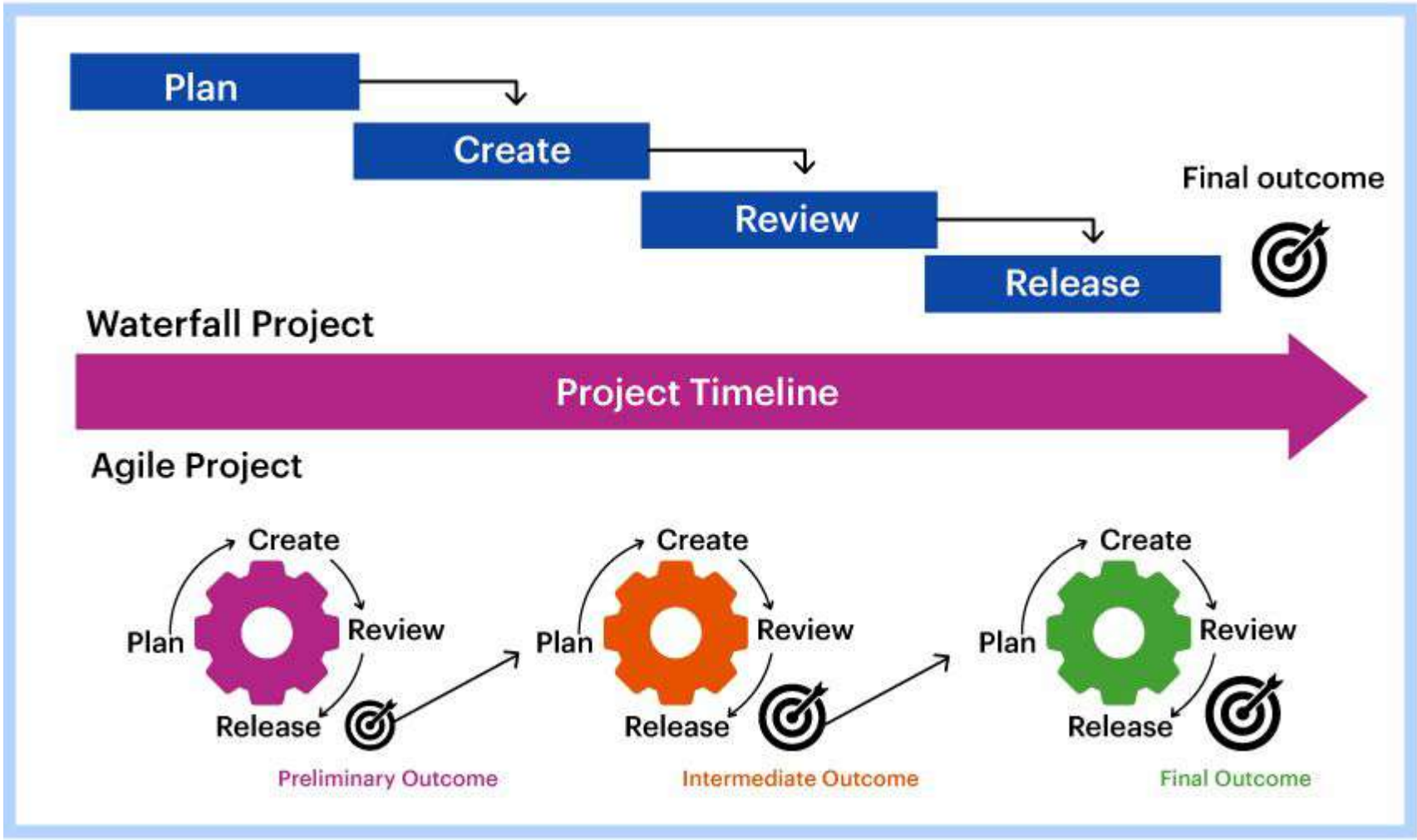
Fig. - One element of the concurrent process model

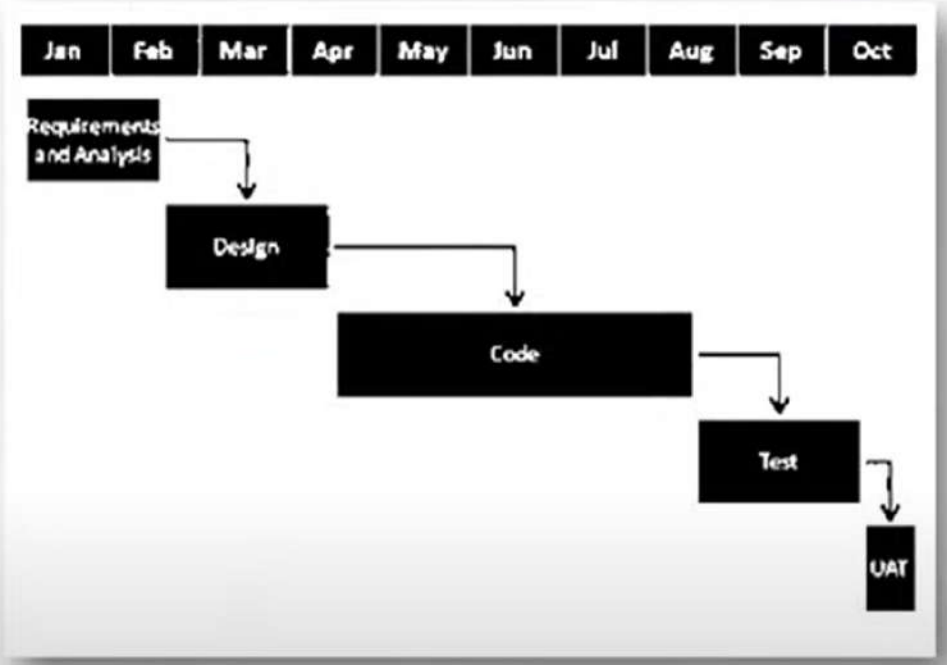
Agile Process Model

- In earlier days, the Iterative Waterfall Model was very popular for completing a project.
- But nowadays, developers face various problems while using it to develop software.
- The main difficulties included handling changes in requirements during project development and the high cost and time required to incorporate these changes.
- To overcome these drawbacks of the Waterfall Model, in the mid-1990s the Agile Software Development model was proposed.

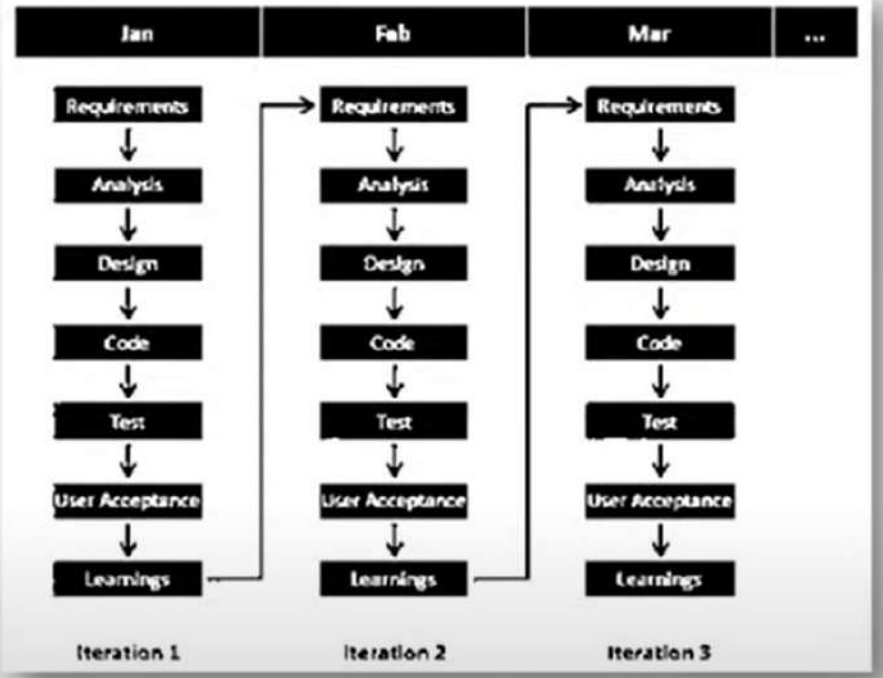
Agile Process Model

- Agile Software Development is widely used by software development teams and is considered to be a flexible and adaptable approach to software development that is well-suited to changing requirements and the fast pace of software development.
- Agile is a time-bound, iterative approach to software delivery that builds software incrementally from the start of the project, instead of trying to deliver all at once.


Plan
Create
Review
Release
Final outcome
Waterfall Project
Project Timeline
Agile Project
Create
Plan
Review
Release
Preliminary Outcome
Create
Plan
Review
Release
Intermediate Outcome
Create
Plan
Review
Release
Final Outcome



Waterfall Model
Time Span



Agile Model
Time Span

Agile Process Model

Principles of Agile:

- The highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- It welcomes changing requirements, even late in development.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference for the shortest timescale.
- Build projects around motivated individuals. Give them the environment and the support they need and trust them to get the job done.
- Working software is the primary measure of progress.
- Simplicity the art of maximizing the amount of work not done is essential.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- By the amount of work that has been finished, gauge your progress.
- Never give up on excellence.
- Take advantage of change to gain a competitive edge.

Agile Process Model

The Agile Software Development Process

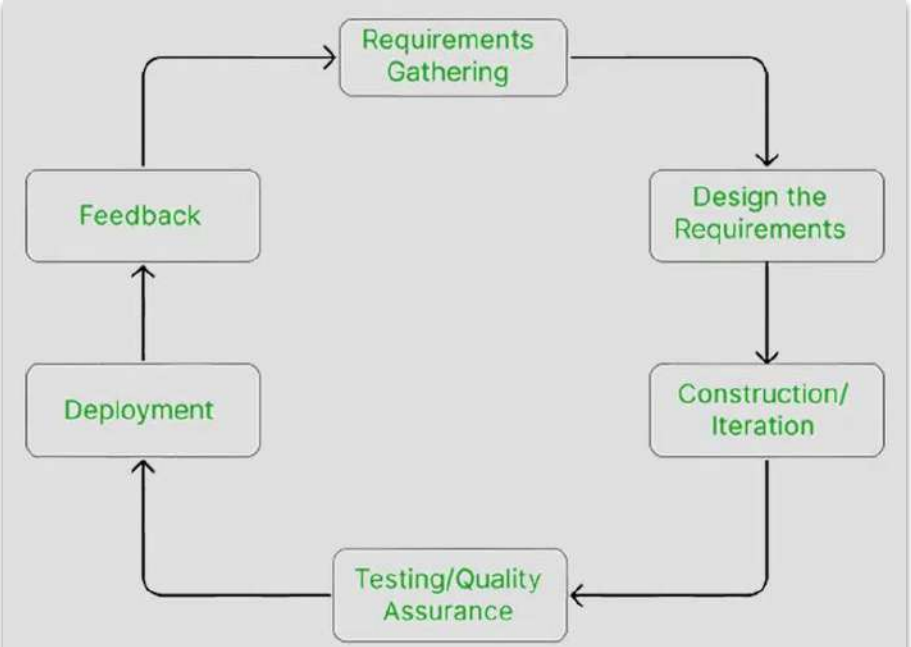


Fig:- Steps in Agile SDLC Model

Agile Process Model

The Agile Software Development Process:

1.Requirement Gathering:- In this step, the development team must gather the requirements, by interaction with the customer. development team should plan the time and effort needed to build the project. Based on this information you can evaluate technical and economical feasibility.

2.Design the Requirements:- In this step, the development team will use user-flow-diagram or high-level UML diagrams to show the working of the new features and show how they will apply to the existing software.

Agile Process Model

The Agile Software Development Process:

3. Construction / Iteration:- In this step, development team members start working on their project, which aims to deploy a working product.

4. Testing / Quality Assurance:- Testing involves Unit Testing, Integration Testing, and System Testing. A brief introduction of these three tests is as follows:

- Unit Testing:- Unit testing is the process of checking small pieces of code to ensure that the individual parts of a program work properly on their own. Unit testing is used to test individual blocks (units) of code.
- Integration Testing:- Integration testing is used to identify and resolve any issues that may arise when different units of the software are combined.
- System Testing:- Goal is to ensure that the software meets the requirements of the users and that it works correctly in all possible scenarios.

Agile Process Model

The Agile Software Development Process:

5. Deployment:- In this step, the development team will deploy the working project to end users.

6. Feedback:- This is the last step of the Agile Model. In this, the team receives feedback about the product and works on correcting bugs based on feedback provided by the customer.

Agile Process Model

Advantages:

- It reduces the total development time of the whole project.
- Agile development emphasizes face-to-face communication among team members, leading to better collaboration and understanding of project goals.
- Customer representatives get the idea of updated software products after each iteration. So, it is easy for him to change any requirement if needed.
- Agile development puts the customer at the center of the development process, ensuring that the end product meets their needs.

Agile Process Model

Disadvantages:

- The agile model depends highly on customer interactions so if the customer is not clear, then the development team can be driven in the wrong direction.
- Agile development models require a high degree of expertise from team members, as they need to be able to adapt to changing requirements and work in an iterative environment. This can be challenging for teams that are not experienced in agile development practices and can lead to delays and difficulties in the project.
- Due to the absence of proper documentation, when the project completes and the developers are assigned to another project, maintenance of the developed project can become a problem.

Extreme Programming (XP)

- Extreme programming (XP) is one of the most important software development frameworks of Agile models.
- It is used to improve software quality and responsiveness to customer requirements.
- The extreme programming model recommends taking the best practices that have worked well in the past in program development projects to extreme levels.
- A developer focuses on the framework activities like planning, design, coding and testing.
- XP has a set of rules and practices.

Extreme Programming (XP)

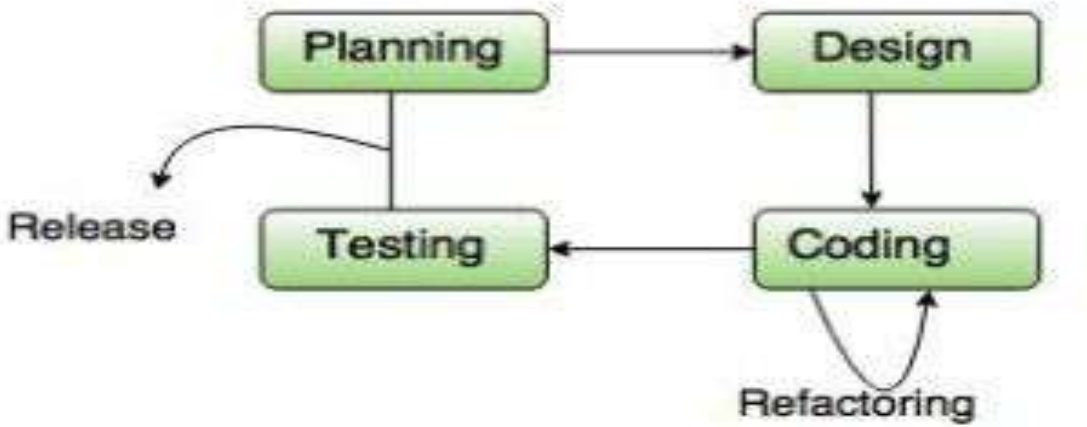


Fig. - The Extreme Programming Process

Extreme Programming (XP)

XP Values: Values offer teams purpose, guiding them in making high-level decisions

- Communication: You can’t work together effectively without sharing knowledge, and you can’t share knowledge without communication.
- Simplicity: Developers can save time and effort by writing simple, effective code that works properly. Ultimately, the less complex code enhances product value.
- Feedback: Early, constant feedback is ideal for team members who release frequent software deliveries, helping them to adjust as the project evolves and changes. The sooner programmers know that the product requires changes, the easier it is to create those changes (and less painful).
- Respect: Every team member cares about their work, and everyone contributes.
- Courage: It takes guts to admit you're mistaken and that your idea didn't work and must be changed. Being honest takes courage, and you need honesty in providing realistic estimates, even if stakeholders don't like the truth.

Extreme Programming (XP)

XP Process: The XP process comprises four framework activities

- Planning
  - Begins with the listening, leads to creation of "user stories" that describes required output, features, and functionality.
  - Customer assigns a value(i.e., a priority) to each story.
  - Agile team assesses each story and assigns a cost (development weeks. If more than 3 weeks, customer asked to split into smaller stories)

Extreme Programming (XP)

XP Process: The XP process comprises four framework activities

- Design
  - begin with the simplest possible design, knowing that later iterations will make them more complex
- Coding
  - Recommends the construction of a unit test for a story before coding commences. So implementer can focus on what must be implemented to pass the test.
  - Encourages “pair programming”. Two people work together at one workstation.

Extreme Programming (XP)

XP Process: The XP process comprises four framework activities

- Testing
  - Validation testing of the system occurs on a daily basis. It gives the XP team a regular indication of the progress.
  - 'XP acceptance tests' are known as the customer test

Scrum

Scrum is the type of **Agile framework**. It is a framework within which people can address complex adaptive problem while productivity and creativity of delivering product is at highest possible values. Scrum uses **Iterative process**. **Salient features of Scrum are:**

- Scrum emphasizes self-organization
- Scrum is simple to understand
- Scrum framework help the team to work together

What is SCRUM?

Scrum is a **framework** within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value.

What is SCRUM?

These iterations are called **Sprints** and at the end of each sprint you have the **launch** of a **potentially deliverable software**.

Product Backlog is ordered list of tasks and requirements the final product needs

Product Backlog

Sprint Backlog

List of all items from the Product Backlog that need to be worked on during a Sprint.

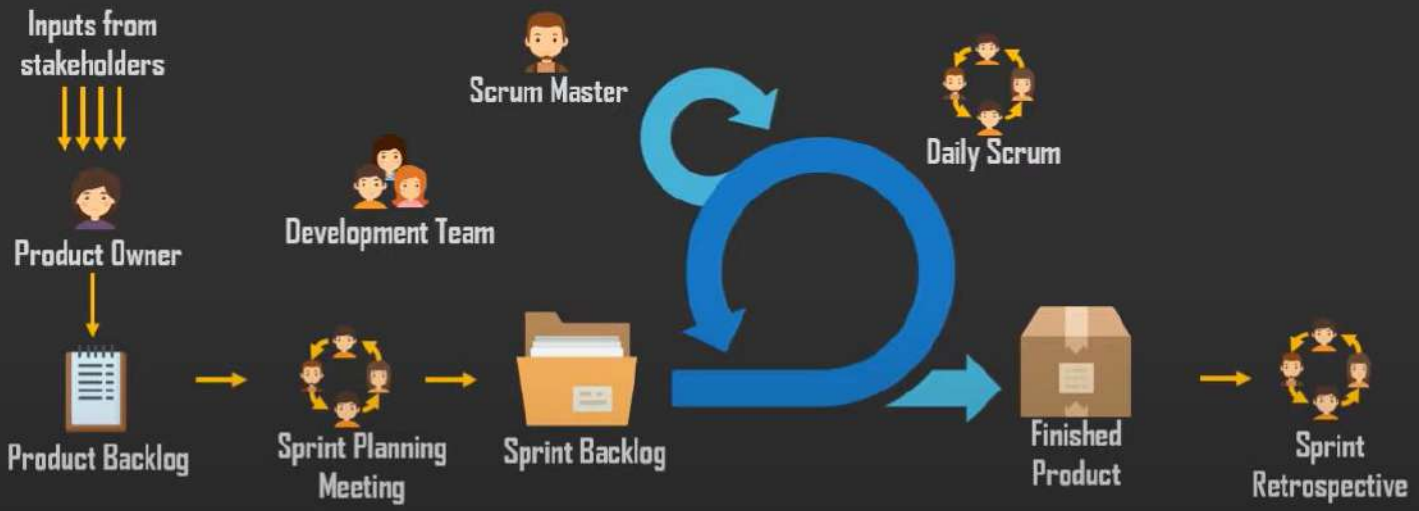
A burndown chart is a graphical representation of the amount of estimated remaining work.

Burndown Charts

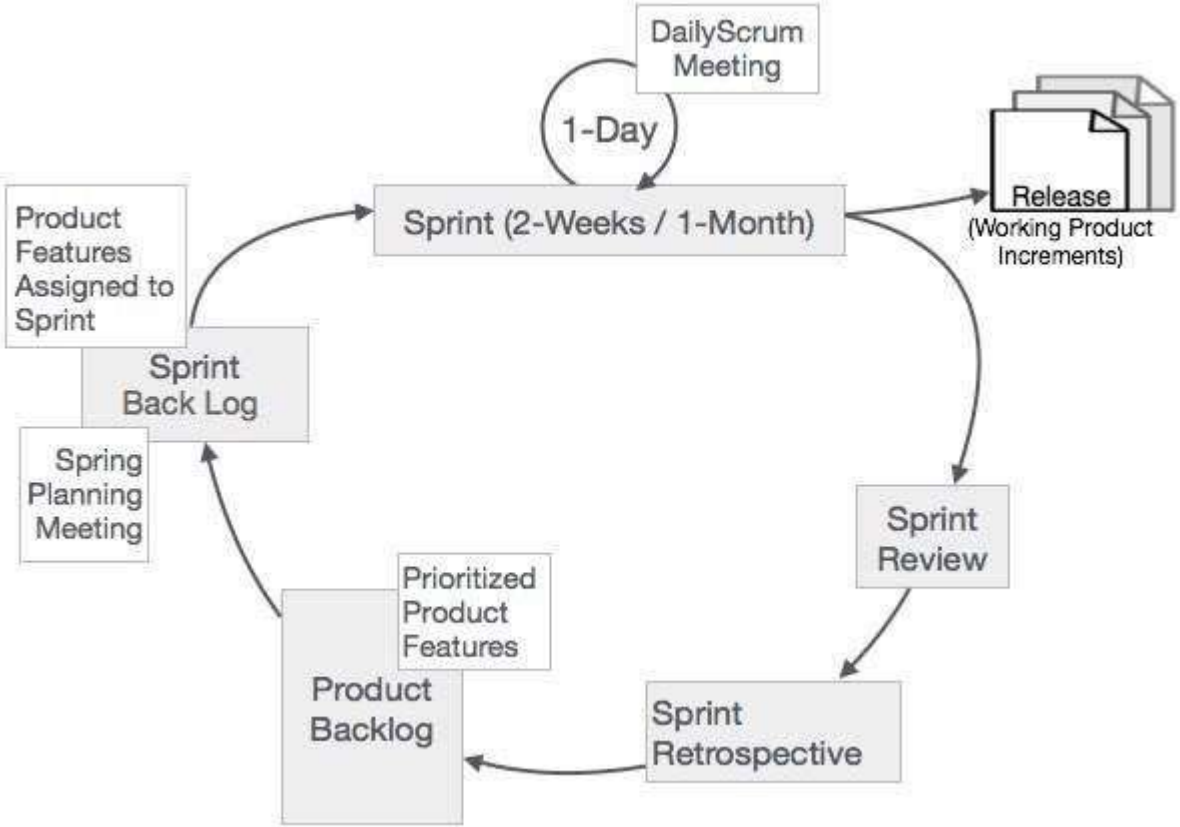
Product Increments

Sum of work completed in a sprint, combined with all work completed during previous sprints

What is the SCRUM Framework?



Scrum Process Framework



Scrum

Sprint:

- It breaks down big complex projects into bite-size pieces.
- It makes projects more manageable, allows teams to ship high quality, work faster, and more frequently.
- The sprints give them more flexibility to adapt to the changes.
- Sprints are a short, time-boxed period for Scrum team that works to complete a set amount of work.
- Sprints are the core component of Scrum and agile methodology.
- The right sprints will help our agile team to ship better software.
- **A Sprint is a time box of one month or less. A new Sprint starts immediately after the completion of the previous Sprint.**

Scrum

Release: When the product is completed, it goes to the Release stage.

Sprint Review: If the product still has some non-achievable features, it will be checked in this stage and then passed to the Sprint Retrospective stage.

It is held at the end of the Sprint to inspect the Increment and make changes to the Product Backlog, if needed.

Sprint Retrospective: In this stage quality or status of the product is checked.

It occurs after the Sprint Review and prior to the next Sprint Planning. In this meeting, the Scrum Team is to inspect itself and create a plan for improvements to be enacted during the subsequent Sprint.

Scrum

Product backlog: In scrum features of the product are written from the perspective of the end-user. Features are known as user stories. The collection of user stories is called a Product backlog.

It is the wish list of features of the product. After creating a product backlog.

According to the prioritize features the product is organized.

Scrum

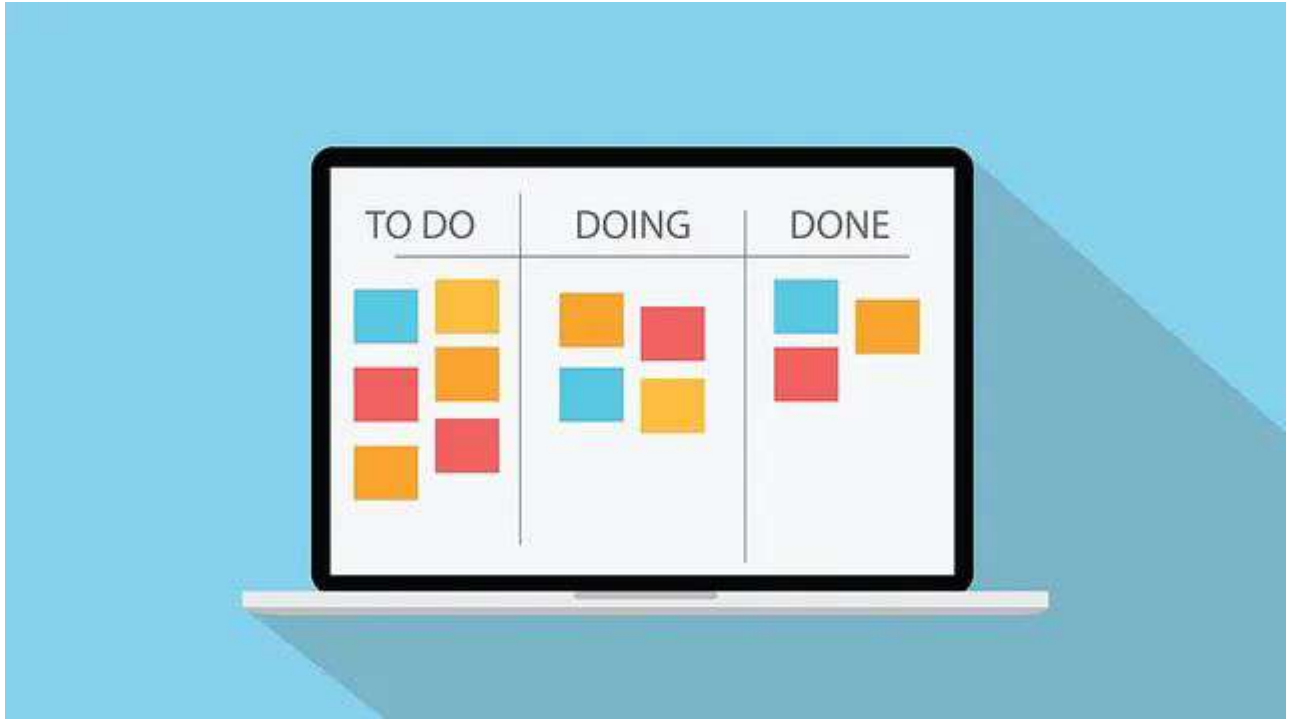
Sprint Backlog: Sprint Backlog is divided into two parts

1. Product features assigned to sprint
2. Sprint planning meeting.

Kanban

- Kanban is a visual management method that originated from lean manufacturing in Japan.
- In software engineering, it’s been adapted as an agile methodology to manage and improve workflow.
- The main visualization tool is a board (either physical or digital) where tasks or work items are represented by cards that move through columns representing stages of the process.

Kanban

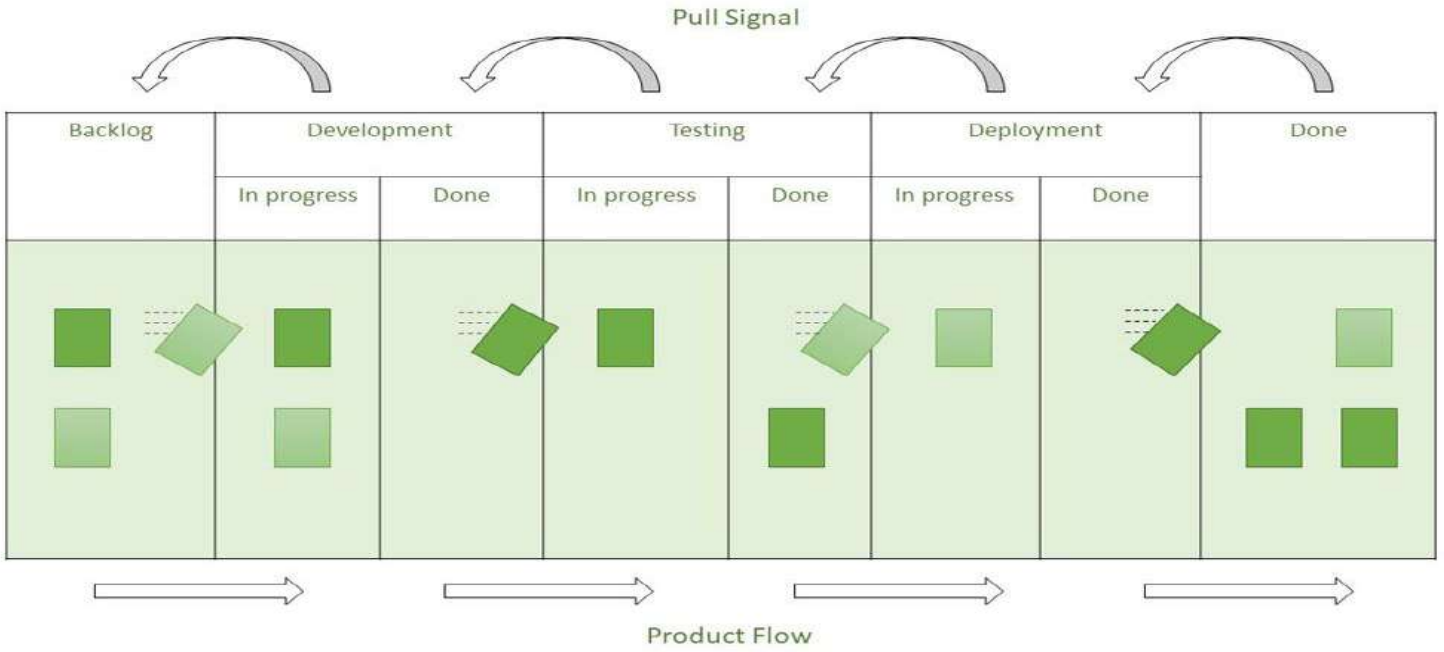


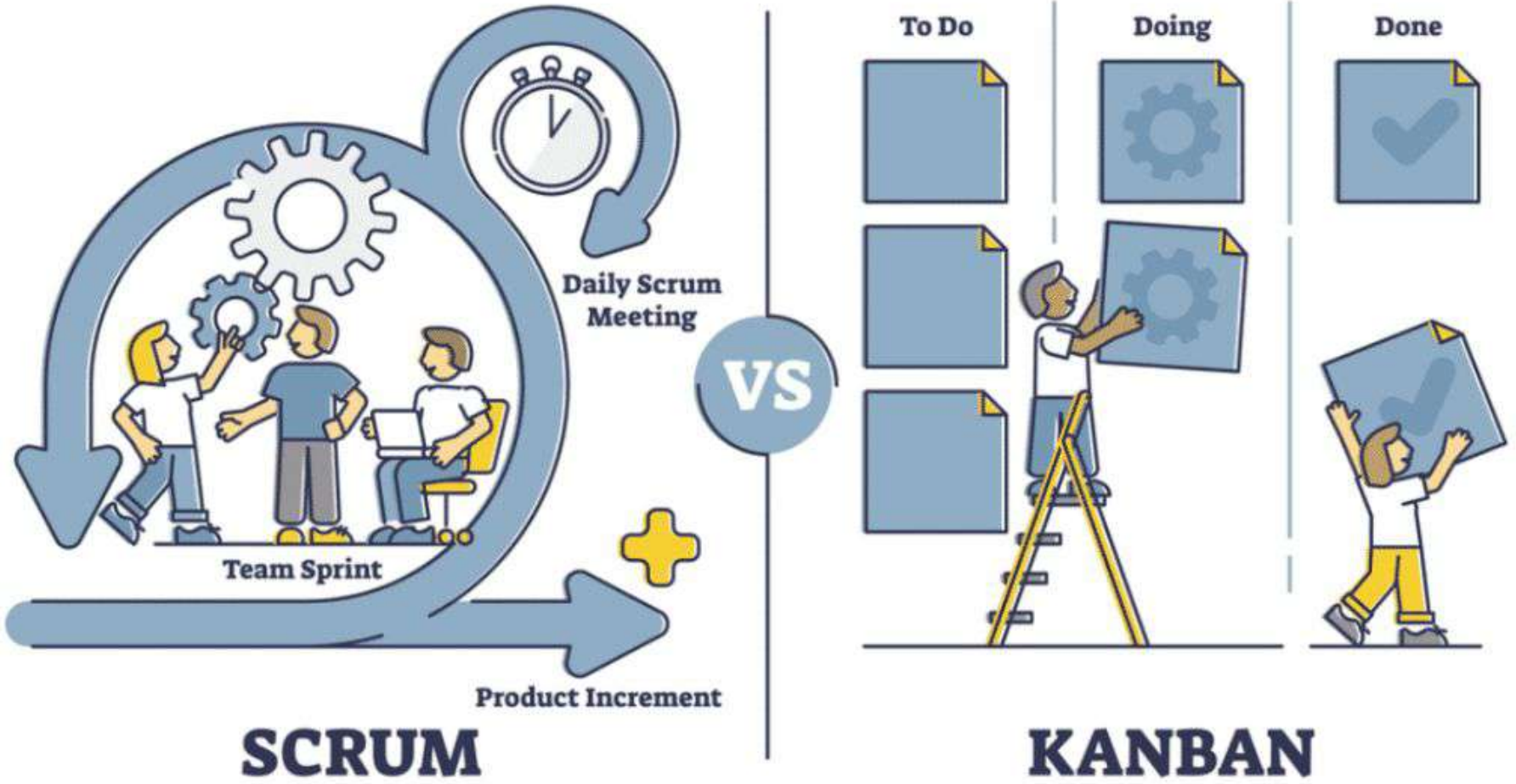
Kanban

Kanban is used for several reasons:

- Visualization of Work: By visualizing the flow of work, teams can easily identify bottlenecks, blockers, and areas of inefficiency.
- Limiting Work in Progress (WIP): This encourages teams to finish what they started before taking on new tasks.
- Continuous Improvement: It fosters a culture of continuous improvement by allowing teams to adapt their workflow as they identify opportunities for optimization.

Kanban


Pull Signal
Backlog
Development
Testing
Deployment
Done
In progress
Done
In progress
Done
In progress
Done
Product Flow


Daily Scrum Meeting
Team Sprint
Product Increment
SCRUM
VS
To Do
Doing
Done
KANBAN

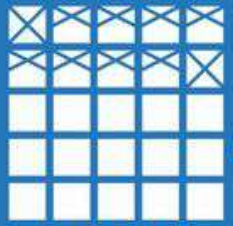
SCRUM

ITERATIVE

INCREMENTAL

FIXED TIMEBOX
FIXED SCOPE

ROLES



CEREMONIAL

XP = EXTREME PROGRAMMING

ITERATIVE

INCREMENTAL

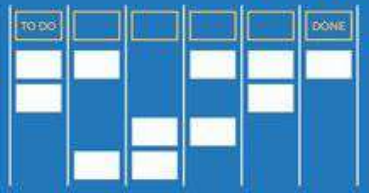
FIXED TIMEBOX
FIXED SCOPE

OFTEN USED AS A COMPLEMENT TO SCRUM

PRACTICES

PAIR PROGRAMMING
TEST DRIVEN DEVELOPMENT (TDD)
CONTINUOUS INTEGRATION
...

KANBAN



VISUAL MANAGEMENT

LIMIT WIP TO DECREASE TTM
(WIP = WORK IN PROGRESS)

PULL FLOW / CONTINUOUS FLOW

OFTEN USED FOR SUPPORT / CORRECTIVE & EVOLUTIVE MAINTENANCE

SCRUMBAN

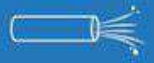
MIX SCRUM + KANBAN

TIMEBOX

CEREMONIAL

VISUAL MANAGEMENT

ROLES



LIMIT WIP