# Software Engineering (303105253)

**Shubham Upadhyay**
Computer Science & Engineering

## Unit-1

## Introduction

## CONTENT

## Introduction:

### What Is 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.

- Software Engineering is mainly used for large projects based on software systems rather than single programs or applications.

## Software Characteristics

- It is intangible, meaning it cannot be seen or touched.
- It is easy to replicate, meaning it can be copied and distributed easily.
- It can be difficult to understand and modify, especially for large and complex systems.
- It can be affected by changing requirements, meaning it may need to be updated or modified as the needs of users change.

## Components:

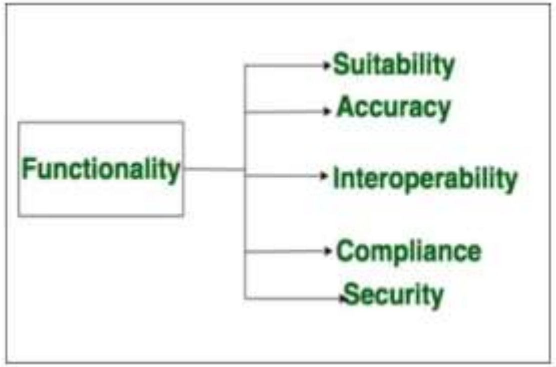
- **There are 6 components :**

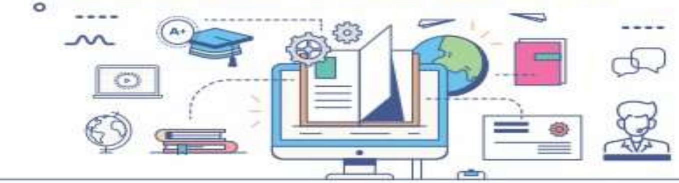
1. Functionality
2. Efficiency
3. Reliability
4. Maintainability
5. Portability
6. Usability

1. **Functionality**: Functionality refers to the set of features and capabilities that a software program or system provides to its users.

➢ Examples of functionality in software include:
-> Data storage and retrieval
-> Data processing and manipulation
-> User interface and navigation
-> Communication and networking
-> Security and access control

Required functions are:





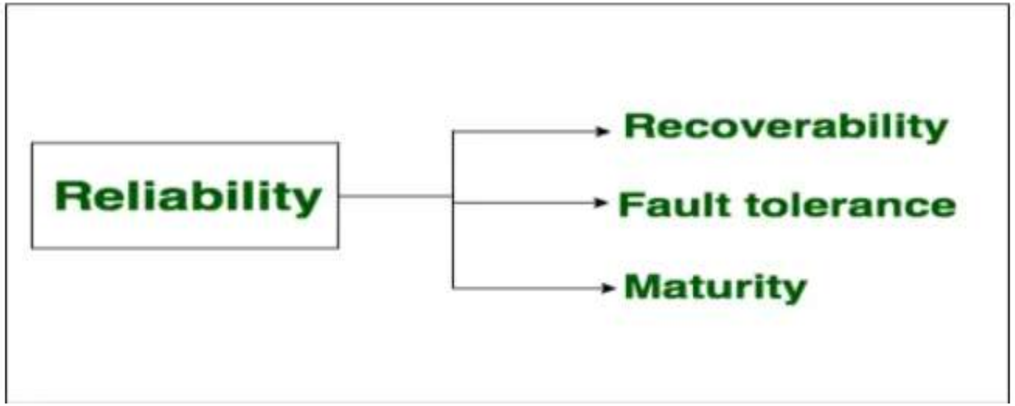
**2.Reliability:** Reliability is a characteristic of software that refers to its ability to perform its intended functions correctly and consistently over time.

- Reliability is an important aspect of software quality, as it helps ensure that the software will work correctly and not fail unexpectedly.

- Examples of factors that can affect the reliability of software include:
  -> Bugs and errors in the code
  ->Lack of testing and validation
  ->Poorly designed algorithms and data structures

- To improve the reliability of software, various techniques, and methodologies can be used, such as testing and validation, formal verification, and fault tolerance.

Required functions are:

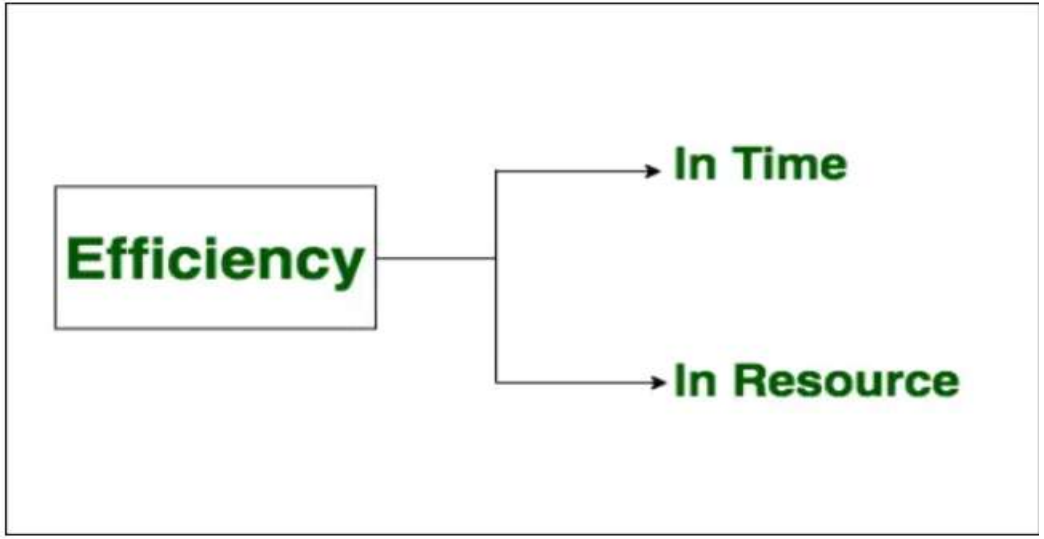


**3.Efficiency:** It refers to the ability of the software to use system resources in the most effective and efficient manner.

- Its ability to use resources such as memory, processing power, and network bandwidth in an optimal way.

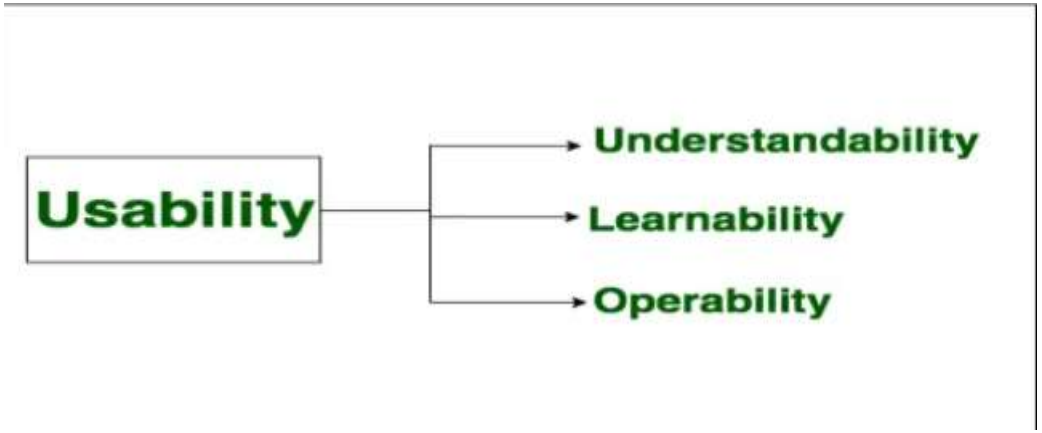
- Examples of factors that can affect the efficiency of the software include:
  -> Inefficient use of memory and processing power
  -> High network latency or bandwidth usage
  -> Unnecessary processing or computation

## Required Functions are:



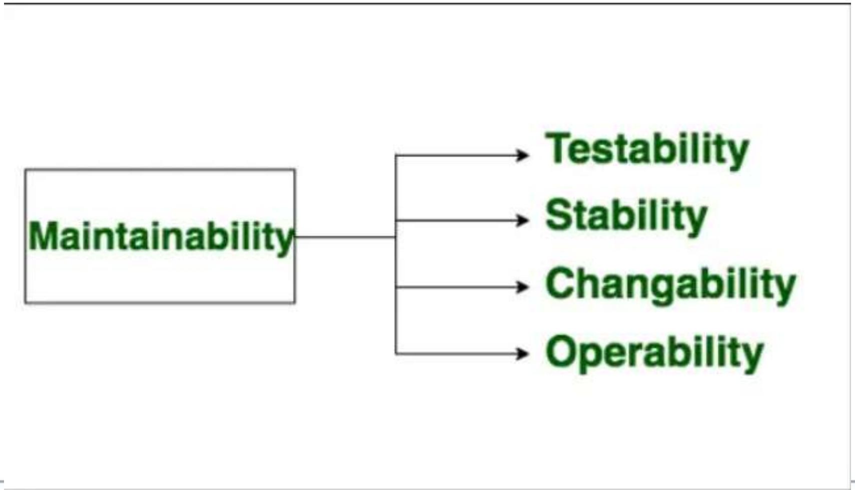
**4.Usability:** It refers to the extent to which the software can be used with ease. the amount of effort or time required to learn how to use the software.

**Required functions are:**



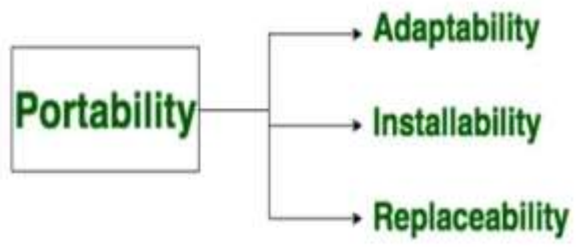
**5. Maintainability:** It refers to the ease with which modifications can be made in a software system to extend its functionality, improve its performance, or correct errors.

Required functions are:



**6. Portability:** A set of attributes that bears on the ability of software to be transferred from one environment to another, without minimum changes.

Required functions are:

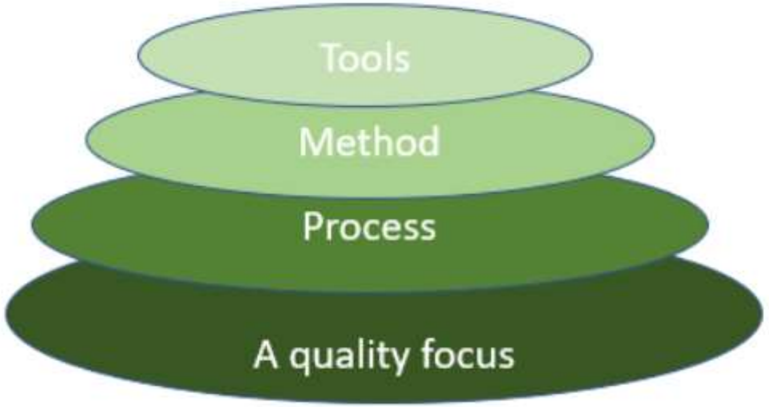


## Applications:

1. System Software
2. Real-time Software
3. Business Software
4. Engineering and Scientific Software
5. Embedded Software
6. Personal Computer Software
7. Web Based Software
8. Artificial Intelligence Software

## Layered Technology SE

➢ Software Engineering is a fully layered technology, to develop software we need to go from one layer to another. All the layers are connected and each layer demands the fulfillment of the previous layer.



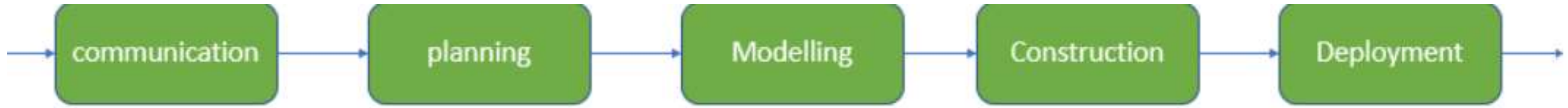
**Layered technology is divided into four parts:**

1. **A quality focus:** It defines the continuous process improvement principles of software. It provides integrity that means providing security to the software so that data can be accessed by only an authorized person, no outsider can access the data. It also focuses on maintainability and usability.

2. **Process:** It is the foundation or base layer of software engineering. It is key that binds all the layers together which enables the development of software before the deadline or on time. The software process covers all the activities, actions, and tasks required to be carried out for software development

## Process activities are listed below:-

I. **Communication:** It is the first and foremost thing for the development of software. Communication is necessary to know the actual demand of the client.
II. **Planning:** It basically means drawing a map for reduced the complication of development.
III. **Modeling:** In this process, a model is created according to the client for better understanding.
IV. **Construction:** It includes the coding and testing of the problem.
V. **Deployment:-** It includes the delivery of software to the client for evaluation and feedback.



3. **Method:** During the process of software development the answers to all “how-to-do” questions are given by method. It has the information of all the tasks which includes communication, requirement analysis, design modeling, program construction, testing, and support.

4. **Tools:** Software engineering tools provide a self-operating system for processes and methods. Tools are integrated which means information created by one tool can be used by another.

## Generic View Of SE

- The process of a software development has three Generic views which are:

**1.Definition Phase:** The definition phase involves understanding and defining the requirements of the software system. Key activities in this phase include:

I. Requirements Gathering and Analysis: Collecting detailed requirements from stakeholders to understand what the system should do.
II. Feasibility Study**:** Assessing the practicality and viability of the project, including technical, economic, and legal considerations.
III. Requirements Specification**:** Documenting the gathered requirements in a clear and concise manner, often in the form of a Software Requirements Specification (SRS) document.

**2. Development Phase:** The development phase is where the actual creation of the software takes place. This phase includes:

I. System Design: Creating the architecture of the system. This can be divided into high-level design (architectural design) and low-level design (detailed design).

II. Coding/Implementation: Translating the design into source code using programming languages. This involves writing the code, conducting unit tests, and integrating various components.

III. Testing: Verifying that the software works as intended. This includes various levels of testing such as unit testing, integration testing, system testing, and acceptance testing.

**3. Maintenance Phase:** The maintenance phase involves making changes to the software to correct defects, improve performance, or adapt to a changed environment. Key activities in this phase include:

I. **Corrective Maintenance:** Fixing bugs and defects found in the software after deployment.

II. **Adaptive Maintenance:** Modifying the software to accommodate changes in the environment, such as new operating systems or hardware.

III. **Perfective Maintenance:** Enhancing the software by adding new features or improving existing ones.

IV. **Preventive Maintenance:** Making changes to prevent potential future problems, improving the software's maintainability and reliability.

## Process Models

1. **Waterfall Model**
2. **Incremental Model**

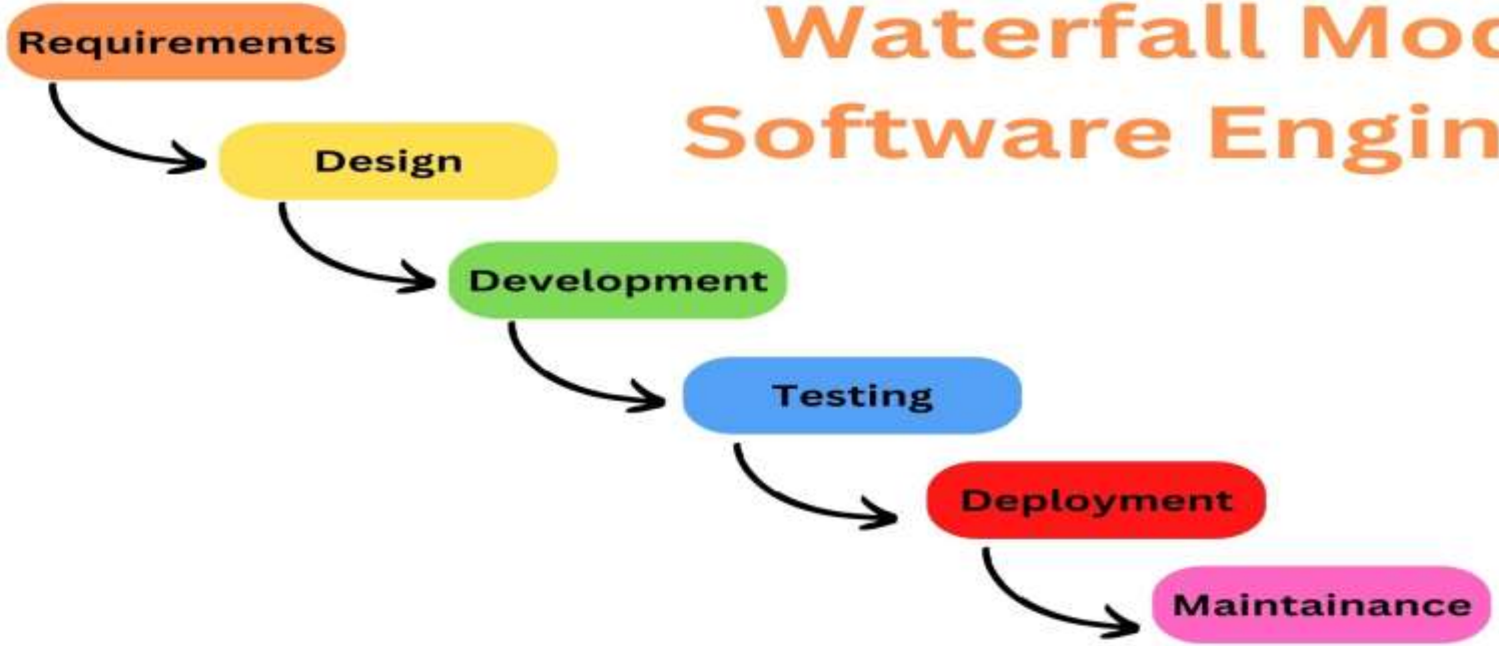
### 1. Waterfall Model:

- Waterfall model is a famous and good version of SDLC(Software Development Life Cycle) for software engineering.

- The waterfall model is a linear and sequential model, which means that a development phase cannot begin until the previous phase is completed.

- We cannot overlap phases in waterfall model.

# Waterfall Model in Software Engineering



**Note:**

- ✓ Similarly waterfall model also works, once one phase of development is completed then we move to the next phase but cannot go back to the previous phase.

**Phases of Waterfall model:**

1. **Requirements:** The first phase involves gathering requirements from stakeholders and analyzing them to understand the scope and objectives of the project.
2. **Design:** 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. **Development:** The Development phase include implementation 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. **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.

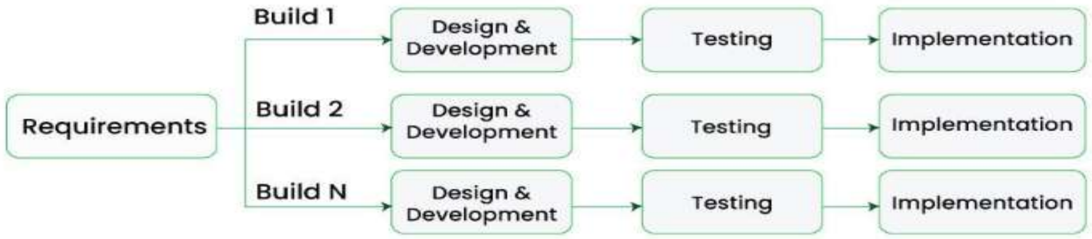
- **Advantages of the Waterfall Model**
- ✓ Easy to Understand: The Classical Waterfall Model is very simple and easy to understand.
- ✓ Individual Processing: Phases in the Classical Waterfall model are processed one at a time.
- ✓ Properly Defined: In the classical waterfall model, each stage in the model is clearly defined.
- ✓ Clear Milestones: The classical Waterfall model has very clear and well-understood milestones.
- ✓ Properly Documented: Processes, actions, and results are very well documented.

- **Disadvantages of the Waterfall Model**
- ✓ Inflexibility: Changes to requirements are difficult to accommodate once the process has started.
- ✓ Late Testing: Defects are found late in the development cycle, which can be costly to fix.
- ✓ High Risk: Not suitable for projects with high uncertainty or where requirements are not well understood from the beginning.

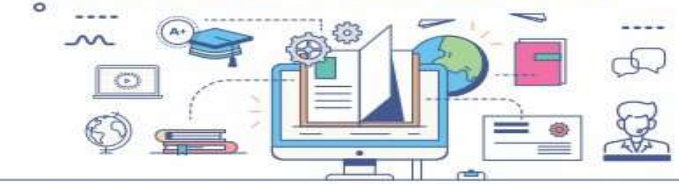
- **Characteristics of the Waterfall Model**
- ✓ Sequential Phases: Each phase must be completed before the next one begins.
- ✓ Documentation: Emphasizes documentation at each phase to maintain clarity and completeness.
- ✓ No Overlapping: Phases do not overlap; each has defined entry and exit criteria.
- ✓ Easy to Manage: Simple and easy to understand, making it straightforward to manage and control.

## 2. Incremental Model:

- The Incremental Model is a process model for software engineering that emphasizes building and delivering the system in small, manageable increments or iterations.
- In simple language, under this model a complex project is developed in many modules or builds.
- For example, we collect the customer's requirements, now instead of making the entire software at once, we first take some requirements and based on them create a module or function of the software and deliver it to the customer.



- **Following are the different phases of Incremental Model:-**
- **Communication:** In the first phase, we talk face to face with the customer and collect his mandatory requirements. Like what functionalities does the customer want in his software, etc.
- **Planning**: In this phase the requirements are divided into multiple modules and planning is done on their basis.
- **Modeling**: In this phase the design of each module is prepared. After the design is ready, we take a particular module among many modules and save it in DDS (Design Document Specification).
- **Construction**: Here we start construction based on the design of that particular module. That is, the design of the module is implemented in coding. Once the code is written, it is tested.
- **Deployment**: After the testing of the code is completed, if the module is working properly then it is given to the customer for use. After this, the next module is developed through the same phases and is combined with the previous module.



- **Advantages of Incremental Model**
  - ✓ This model is flexible and less expensive to change requirements and scope.
  - ✓ The customer can respond to each module and provide feedback if any changes are needed.
  - ✓ It is easier to test and debug during a short iteration.
  - ✓ Errors are easy to identify.

- **Disadvantages of Incremental Model**
  - ✓ Management is a continuous activity that must be handled.
  - ✓ The complete requirements of the software should be clear.
  - ✓ The total cost of this model is higher.

## Evolutionary Process Models

- It is designed to accommodate changes and refinements to the system throughout the development process.

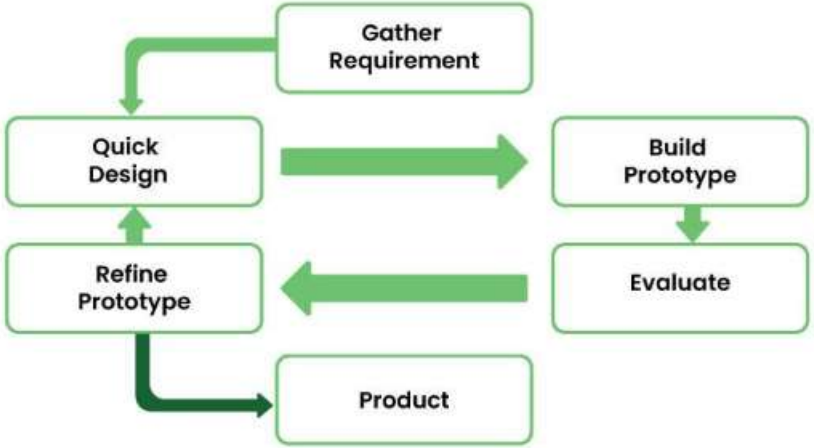
- These models are iterative, meaning they involve repeated cycles of development activities, allowing for continuous feedback and improvement.

- The key evolutionary process models include:

1. Prototyping Model
2. Spiral Model
3. Concurrent Development Model

1. **Prototype Model:**

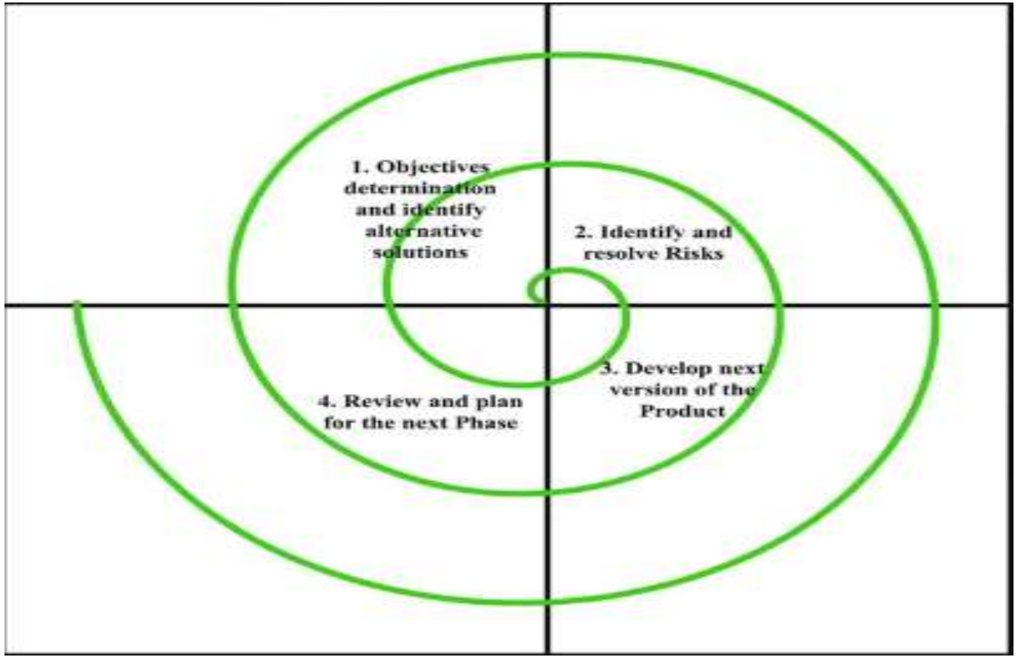
- Prototype model is an activity in which prototypes of software applications are created.
- First a prototype is created and then the final product is manufactured based on that prototype.



- **Phases of the Prototyping Model:**
- **Requirement gathering:** The first step of prototype model is to collect the requirements, although the customer does not know much about the requirements but the major requirements are defined in detail.
- **Build the initial prototype:** In this phase the initial prototype is built. In this some basic requirements are displayed and user interface is made available.
- **Quick Design:** Create a simple, quick design of the system focusing on key aspects.
- **Review the prototype:** When the construction of the prototype is completed, it is presented to the end users or customer and feedback is taken from them about this prototype. This feedback is used to further improve the system and possible changes are made to the prototype.
- **Revise and improve the prototype:** When feedback is taken from end users and customers, the prototype is improved on the basis of feedback. If the customer is not satisfied with the prototype, a new prototype is created and this process continues until the customer gets the prototype as per his

## 2. Spiral Model:

- The Spiral Model is a model that 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.
- Each loop of the spiral is called a **phase** of the software development process.
- 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.


1. Objectives determination and identify alternative solutions
2. Identify and resolve Risks
3. Develop next version of the Product
4. Review and plan for the next Phase

- Each phase of the Spiral Model is divided into four quadrants as shown in the above figure. The functions of these four quadrants are discussed below:

1. **Objectives determination and identify alternative solutions:** Requirements are gathered from the customers and the objectives are identified, elaborated, and analyzed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant.

2. **Identify and resolve Risks:** During the second quadrant, all the possible solutions are evaluated to select the best possible solution. Then the risks associated with that solution are identified and the risks are resolved using the best possible strategy. At the end of this quadrant, the Prototype is built for the best possible solution.

3. **Develop the next version of the Product:** During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available.

4. **Review and plan for the next Phase:** In the fourth quadrant, the Customers evaluate the so-far developed version of the software. In the end, planning for the next phase is started.

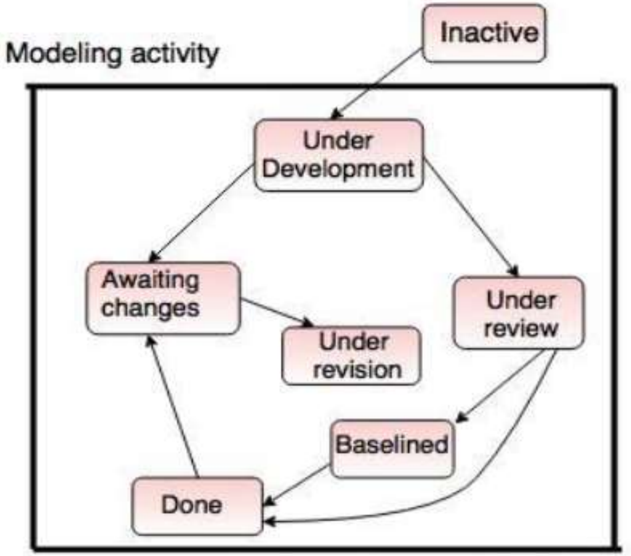
➢ **Advantages of the Spiral Model**

- ✓ Risk Handling
- ✓ Good for large Project
- ✓ Flexibility in Requirements
- ✓ Customer Satisfaction
- ✓ Improved Quality

- **Disadvantages of the Spiral Model**
  - ✓ Complex
  - ✓ Expensive
  - ✓ Difficulty in time Management

## 3. Concurrent Development Model:

- The Concurrent Development Model, also known as concurrent engineering, involves parallel activities across different phases of the software development process.
- Activities from different stages can overlap, allowing for simultaneous progress in multiple areas.
- 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.



In above figure each block represents the state of software engineering activity

Parul® University

## Agile Development

- ➢ Agile development is a group of methodologies that promote iterative development, collaboration, customer feedback, and flexibility in software engineering.
- ➢ Agile methodologies focus on delivering small, functional pieces of software frequently, with continuous improvement and adaptability to changing requirements.
- ➢ **Key Principles of Agile Development:**
- Customer Satisfaction: Deliver valuable software continuously to satisfy the customer.
- Frequent Delivery: Deliver working software frequently, from a couple of weeks to a couple of months.
- Collaboration: Business people and developers must work together daily throughout the project.
- Face-to-Face Communication: The most efficient and effective method of conveying information is through face-to-face conversation.
- Working Software: Working software is the primary measure of progress.

## Agility:

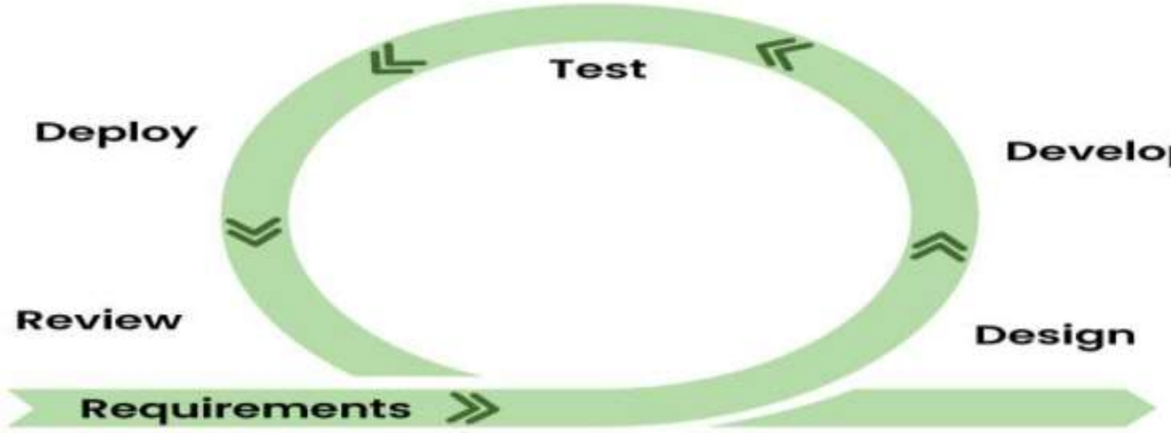
- Agility refers to the ability of a software development process to respond to changing requirements and environments quickly and efficiently. It emphasizes flexibility, collaboration, and customer satisfaction by delivering small, functional pieces of software frequently and iteratively.

- Key Characteristics of Agility:

1. Flexibility: Ability to adapt to changing requirements.
2. Speed: Rapid delivery of functional software.
3. Customer Focus: Continuous involvement of customers and stakeholders.
4. Collaboration: Strong emphasis on teamwork and communication.
5. Continuous Improvement: Regular reflection and adaptation of processes.

## Agile Process model

- Agile process model is a combination of iterative and incremental models, that is, it is made up of iterative and incremental models.
- The agile model was created mainly to make changes in the middle of software development so that the software project can be completed quickly.



## Extreme Programming

- Extreme Programming (XP) is an Agile software development methodology that emphasizes technical excellence, customer satisfaction, and rapid, iterative development.

- **Core Values of Extreme Programming:**

1. **Communication:** Clear, open, and continuous communication among all team members, customers, and stakeholders.
2. **Simplicity:** Focus on the simplest solution that works; avoid unnecessary complexity.
3. **Feedback:** Regular feedback from the system, team members, and customers to guide improvements and adjustments.
4. **Courage:** Encourage team members to make bold decisions and changes when necessary
5. **Respect:** Foster an environment of mutual respect among all team members.

Parul® University

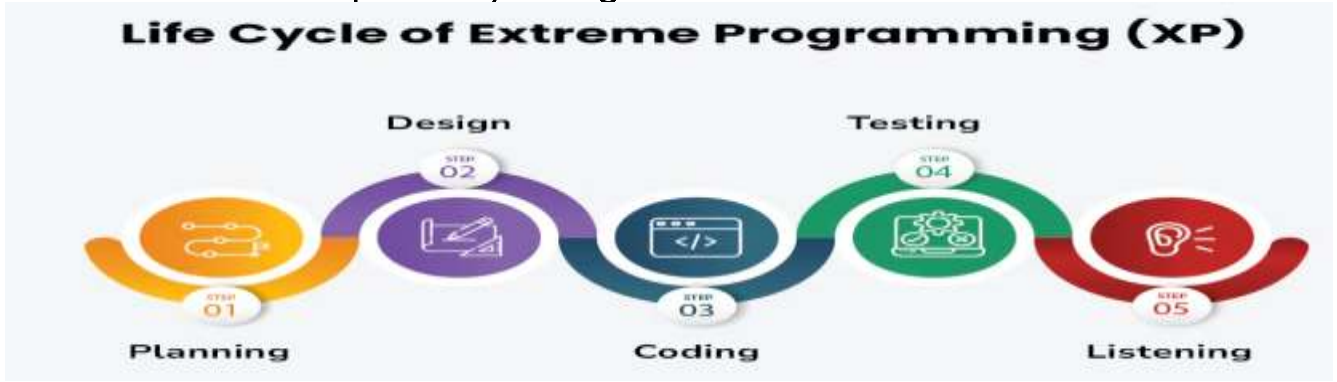
## Life Cycle of Extreme Programming (XP)

- The Extreme Programming Life Cycle consist of five phases:

1. **Planning:** The first stage of Extreme Programming is planning. During this phase, clients define their needs in concise descriptions known as user stories. The team calculates the effort required for each story and schedules releases according to priority and effort.
2. **Design:** The team creates only the essential design needed for current user stories, using a common analogy or story to help everyone understand the overall system architecture and keep the design straightforward and clear.
3. **Coding:** (XP) promotes pair programming i.e**.** developers work together at one workstation, enhancing code quality and knowledge sharing. They write tests before coding to ensure functionality from the start

**4.Testing:** Extreme Programming (XP) gives more importance to testing that consist of both unit tests and acceptance test**.** Unit tests, which are automated, check if specific features work correctly. Acceptance tests, conducted by customers, ensure that the overall system meets initial requirements.

**5. Listening:** In the listening phase regular feedback from customers to ensure the product meets their needs and to adapt to any changes.



- **Key Practices of Extreme Programming:**

i. **Pair Programming:**

- Two developers work together at one workstation.
- One writes code while the other reviews each line of code as it is written.
- Enhances code quality and knowledge sharing.

ii. **Test-Driven Development (TDD):**

- Write tests before writing the actual code.
- Ensures that the code is always tested and works as expected.

iii. **Refactoring:**

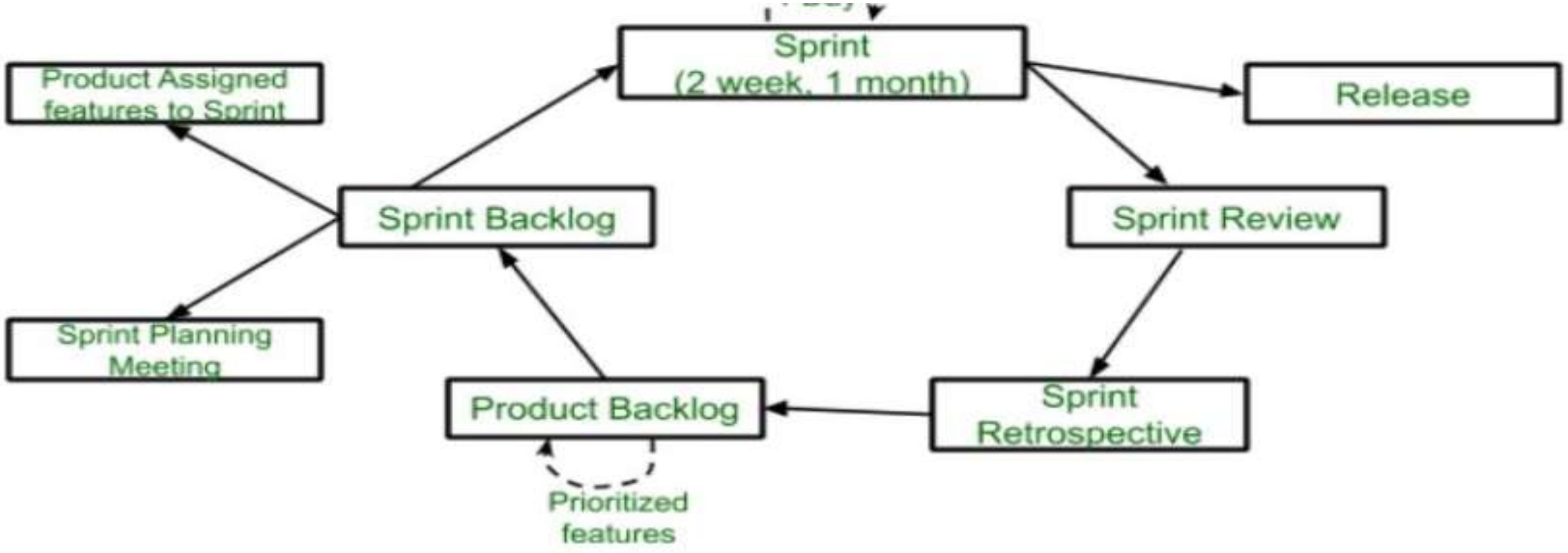
- Continuously improve the codebase without changing its functionality.
- Simplifies code, removes duplication, and enhances maintainability.

## Other process models of Agile Development and Tools.

1. **Scrum:**

- ➢ Scrum is a management framework that teams use to self-organize tasks and work towards a common goal.
- ➢ It is a framework within which people can address complex adaptive problems while the productivity and creativity of delivering products are at the highest possible value.
- ➢ **Silent features of Scrum:**

- Scrum is a light-weighted framework
- Scrum emphasizes self-organization
- Scrum is simple to understand
- Scrum framework helps the team to work together
- Lifecycle of Scrum


Sprint
(2 week, 1 month)
Release
Product Assigned features to Sprint
Sprint Backlog
Sprint Review
Sprint Planning Meeting
Product Backlog
Sprint Retrospective
Prioritized features

Parul® University

- **Sprint:** A Sprint is a time box of one month or less. A new Sprint starts immediately after the completion of the previous Sprint. **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.

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

- **Sprint Backlog**: Sprint Backlog is divided into two parts Product assigned features to sprint and Sprint planning meeting.

**2. Kanban:**

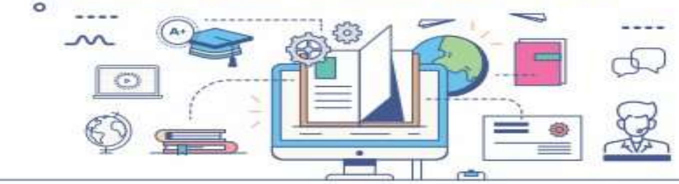
- Visualizes the workflow and limits work in progress to optimize the flow of work.
- Focuses on continuous delivery and releasing features as soon as they are ready.

**3. Lean Software Development:**

- Focuses on delivering value to the customer, eliminating waste, and continuous improvement.
- Emphasizes customer collaboration, shortening feedback loops, and empowering teams.

**4. Crystal:**

- Tailors the methodology to the specific characteristics of the project, team, and organization.
- Offers different colors (variants) of Crystal, such as Clear, Yellow, Orange, and others, each suited to different project types.



**5. Dynamic Systems Development Method (DSDM):**

- An Agile framework that focuses on delivering projects on time and within budget while maintaining quality.
- Defines roles such as Executive Sponsor, Project Manager, and Solution Developer.

**6. Feature-Driven Development (FDD):**

- An iterative and incremental software development process.
- Emphasizes designing features incrementally and focusing on domain object modeling.

## Tools for Agile Development

- Agile development is supported by various tools that facilitate collaboration, project management, and continuous integration. Some popular Agile tools include:

1. **JIRA:** A widely-used tool for Agile project management, including Scrum and Kanban boards, backlog management, and reporting.

2. **Trello:** A visual project management tool that uses boards, lists, and cards to organize tasks and projects.

3. **Asana:** A project management tool that helps teams organize, track, and manage their work, including task assignments and progress tracking.

4. **Azure DevOps:** A set of development tools and services by Microsoft, including version control, project tracking, and build automation.

5. **GitHub:** A platform for version control and collaboration, supporting Git repositories, pull requests, and code reviews.

**6. Slack:** A communication and collaboration tool that supports real-time messaging, file sharing, and integration with other tools.

## Agile Model

**What is Agile Model?**

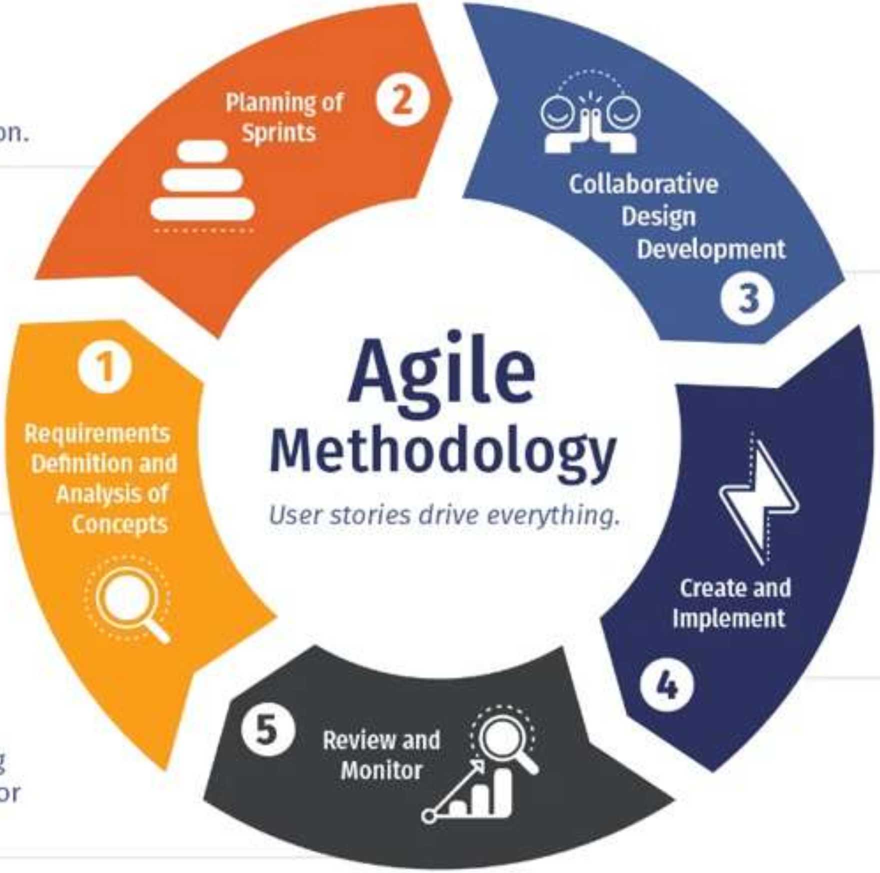
**The Agile Model** was primarily designed to help a project adapt quickly to change requests. So, the main aim of the Agile model is to facilitate quick project completion. To accomplish this task, it's important that agility is required. Agility is achieved by fitting the process to the project and removing activities that may not be essential for a specific project.

## Steps in the Agile Model

**Steps in the Agile Model**

The Agile Model is a combination of iterative and incremental process models. The phases involve in **Agile** (SDLC) **Model** are:

**Requirement Gathering**
**Design the Requirements**
**Construction / Iteration**
**Testing / Quality Assurance**
**Deployment**
**Feedback**


Agile
Methodology
User stories drive everything.
1
Requirements Definition and Analysis of Concepts
2
Planning of Sprints
3
Collaborative Design Development
4
Create and Implement
5
Review and Monitor

Arrange teams and tools needed to optimize production.

From the beginning of the process, the end users' involvement and feedback is critical.

Analysis of concepts and requirements definitions; Determine current state and your expectations.

Frequent development delivery through sprints. Feedback on testing & appropriate changes are imperative.

Ensure that you are reviewing and monitoring key metrics for success.

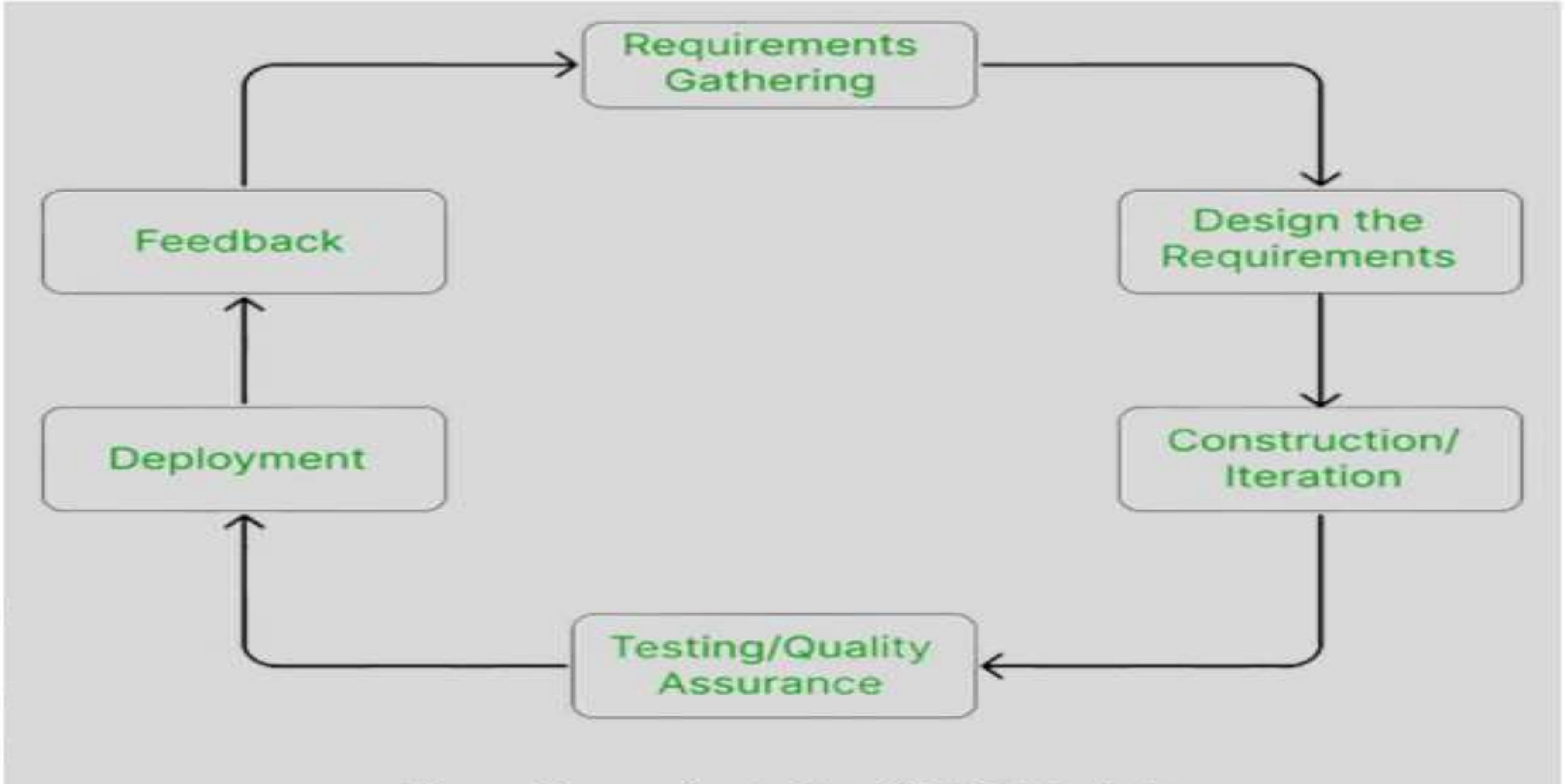


Fig:- Steps in Agile SDLC Model

## 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.

•Meet with the customer to really understand their needs and what they expect from the software.
•Identify the key requirements and business goals to make sure everyone is on the same page.
•Estimate how much time and effort it will take to develop the software.
•Assess if the project is technically possible and whether it's worth the investment from both a technical and economic standpoint.

## 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. Wireframing and designing user interfaces are done in this phase.

•**Designing the system**: Once the requirements are gathered, the next step is to design the system's overall architecture based on those needs. This helps to verify the software is structured in a way that meets the user's expectations.

•**Creating wireframes**: Next, wireframes for the user interface (UI) are created. These are simple blueprints that show how the software will look and how users will interact with it, ensuring it's user-friendly and easy to navigate.

•**High-level design with UML diagrams**: At this stage, high-level designs using UML (Unified Modeling Language) diagrams are created to visually represent the software's structure and how different parts will work together.

•**Prototyping for feedback**: Prototypes are made to give stakeholders an early look at the software. This helps gather feedback early in the process and allows for adjustments before the full development begins.

**Construction / Iteration**

In this step, development team members start working on their project, which aims to deploy a working product. Each cycle typically consist between **1-4 weeks**, and at the end, the team delivers a working version of the software.

**Development of Features**: The team works on the features identified during the requirement and design phases.

**Coding and Implementation**: New functionalities are coded and integrated into the software based on the goals for that specific iteration.

**Delivering a Working Product**: After each iteration, a usable version of the software is ready.

**Incremental Improvement**: With every cycle, the software is enhanced, adding more features and refining existing ones

## Testing / Quality Assurance

Testing involves Unit Testing, Integration Testing, and System Testing, Which help in the agile development models:

**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.

**Deployment**

In this step, the development team will deploy the working project to end users.

Once an iteration is finished and fully tested, the software is ready to be released to the end users. In Agile, deployment isn't a one-time event—it's an ongoing process. Updates and improvements are rolled out regularly, making sure the software keeps evolving and getting better with each release.

Deploy the software to a test or live environment so that it can be used by customers or end-users.

Make the software accessible to users, verifying they can start using it as expected.

Verify the deployment goes smoothly and fix any issues that come up quickly

**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.

**Take feedback** from customers, users, and stakeholders after each iteration.

Understand how well the product meets user needs and identify areas for improvement.

**Check for bugs** or issues that need fixing.

**Make adjustments** to the development plan based on feedback to improve the product further.

# DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in