Unit 2: Software Process Models

• Waterfall Model

The Waterfall Model was the first Process Model to be introduced. It is also referred to as a liner-sequential life cycle model. The Waterfall model is the earliest SDLC approach that was used for software development.
The Waterfall Model is useful in situations where the project requirements are well-defined and the project goals are clear. It is often used for large-scale projects with long timelines, where there is little room for error and the project stakeholders need to have a high level of confidence in the outcome.

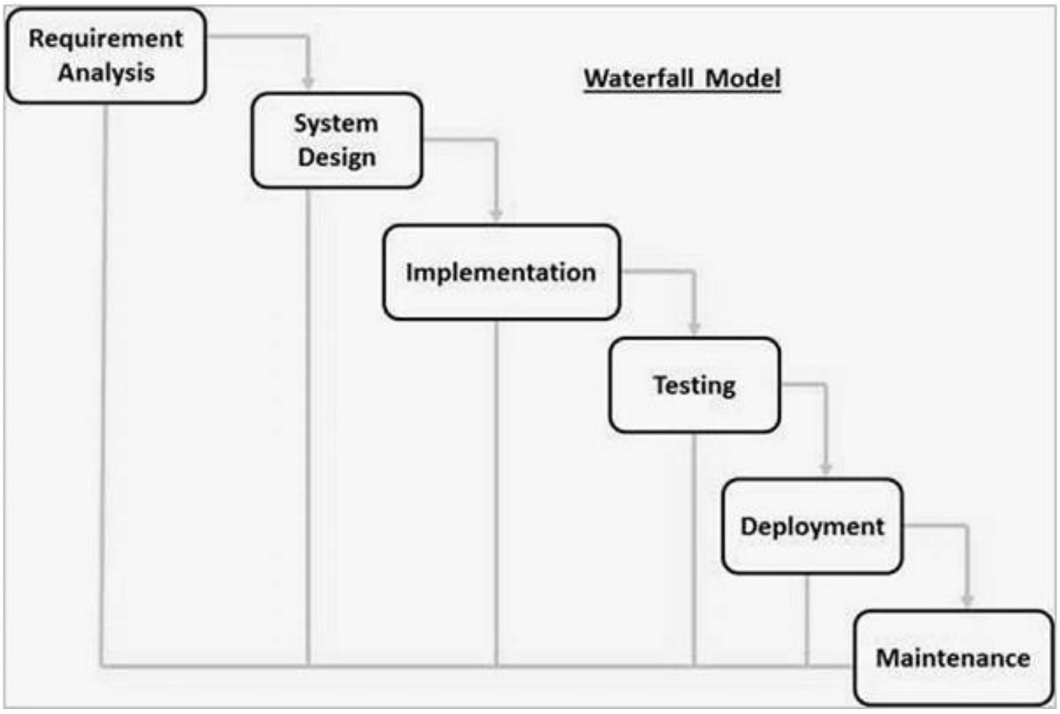
For example, in construction, these three general steps are usually followed:

1. A building's physical design is created before any construction begins.
2. The foundation is poured before the skeleton of a building is erected.
3. The skeleton of the building is completed before the walls are built.

Real Life Example: Developing an Online Banking System

Phases of Waterfall Model

The development process can be considered as a sequential flow in the waterfall. The different sequential phases of the classical waterfall model are follow:



1. **Requirements.** Potential project requirements, deadlines and guidelines for the project are analyzed and placed into a formal requirements document, also called a functional specification. This first phase of development defines and plans the

project without mentioning specific processes. It also defines the project scope, team members, stakeholders, process for requirements gathering, reporting of project progress, use of aids such as templates and workflow diagrams, and an overall roadmap of the project.

2. **System Design**: The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
3. **Implementation**: With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
4. **Integration and Testing**: All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
5. **Deployment of system**: Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
6. **Maintenance**: There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

Waterfall Model - Advantages

- Simple and easy to understand and use
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.

Waterfall Model - Disadvantages

- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.
- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.
- Adjusting scope during the life cycle can end a project.
- Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

Unit 2: Software Process Models

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 depended 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 careful planning process, where the project scope, timelines, and deliverables are carefully defined and monitored throughout the project lifecycle.

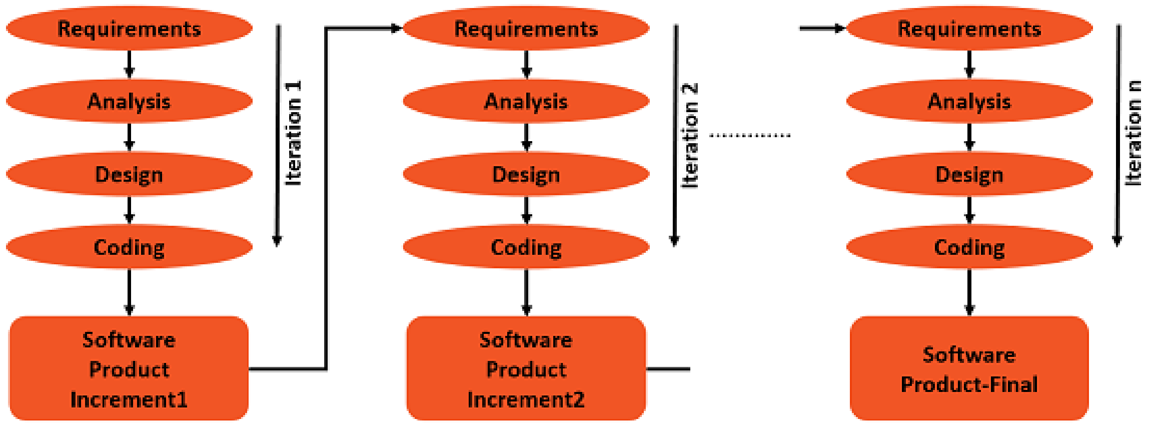
When to Use Waterfall Model?

- **Well-understood Requirements:** Before beginning development, there are precise, reliable, and thoroughly documented requirements available.
- **Very Little Changes Expected:** During development, very little adjustments or expansions to the project's scope are anticipated.
- **Small to Medium-Sized Projects**: Ideal for more manageable projects with a clear development path and little complexity.
- **Predictable:** Projects that are predictable, low-risk, and able to be addressed early in the development life cycle are those that have known, controllable risks.
- **Regulatory Compliance is Critical:** Circumstances in which paperwork is of utmost importance and stringent regulatory compliance is required.
- **Client Prefers a Linear and Sequential Approach**: This situation describes the client's preference for a linear and sequential approach to project development.
- **Limited Resources**: Projects with limited resources can benefit from a set-up strategy, which enables targeted resource allocation.

Unit 2: Software Process Models

➢ Iterative Incremental Models

Iterative Incremental model, initially, a partial implementation of a total system is constructed so that it will be in a deliverable state. Increased functionality is added. Defects, if any, from the prior delivery are fixed and the working product is delivered. The process is repeated until the entire product development is completed. The repetitions of these processes are called iterations. At the end of every iteration, a product increment is delivered.



Iterative Incremental Advantages

- You can develop prioritized requirements first.
- Initial product delivery is faster.
- Customers gets important functionality early.
- Lowers initial delivery cost.
- Each release is a product increment, so that the customer will have a working product at hand all the time.
- Customer can provide feedback to each product increment, thus avoiding surprises at the end of development.
- Requirements changes can be easily accommodated.

Iterative Incremental Model Disadvantages

- Requires effective planning of iterations.
- Requires efficient design to ensure inclusion of the required functionality and provision for changes later.
- Requires early definition of a complete and fully functional system to allow the definition of increments.

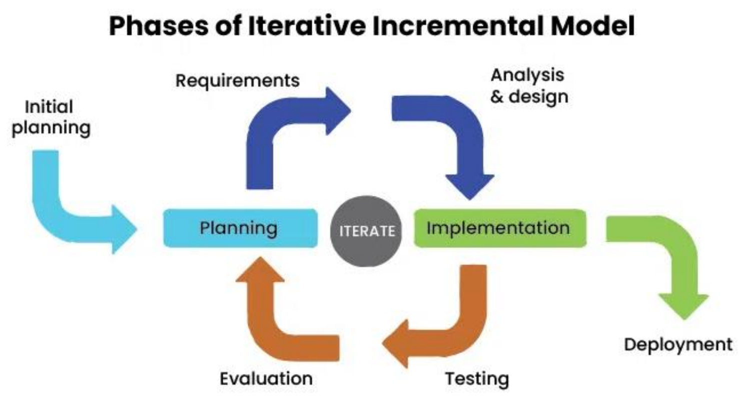
- Well-defined module interfaces are required, as some are developed long before others are developed.
- Total cost of the complete system is not lower.

When to Use the Iterative Approach

Numerous things can affect product delivery. The iterative approach is ideal for projects that involve regular deliveries. Such projects include repeated activities that need constant checking and improvement.
Iterative development allows your team to focus on collecting feedback and requirements. It involves a continuous cycle of:

- Planning
- Designing
- Implementing
- Testing
- Evaluating

<u>Phases of Iterative Incremental Model</u>



1. Planning Phase: In this phase, the team identifies the goals and objectives of the project, along with the project scope, requirements, and constraints on them. The team then identifies different iterations that would be needed to complete the project successfully.

2. **Requirements Analysis and Design Phase:** In this phase, the requirements met are then analyzed and the according system is designed based on these requirements. The projected design should be modular, which would allow easy modification and testing in subsequent iterations.

3. **Implementation Phase:** In this phase, the system is implemented based on the design created in the previous phase. The implementation should be done in small, manageable pieces or increments, which can then be tested in the next phase of the cycle.

4. **Testing Phase:** In this phase, the system is tested against the requirements identified in the planning phase. Testing is done for each iteration, and any defects or issues are identified and resolved, and this helps in each iteration.

5. **Evaluation Phase:** In this phase, the team evaluates the performance of the system based on the results of testing. Feedback is gathered from users and stakeholders, and changes are made to the system as needed which makes the system more scalable and flexible.

6. Incremental Release: In this phase, the completed iterations are released to users and stakeholders. Each release builds on the previous release, providing new functionality or improving existing functionality to a great extent.

Real-World Examples of the Iterative Incremental Model

- **Microsoft Windows Operating System:** Microsoft Windows is a prime example of a software system that has evolved through iterative and incremental development. With each new version, Microsoft introduces new features and improvements based on user feedback and changing market demands.
- **Amazon Web Services (AWS):** AWS, Amazon's cloud computing platform, has been developed using an iterative and incremental approach. AWS started with a few basic services and has gradually expanded its offerings based on customer feedback and market demand.

Unit 2: Software Process Models

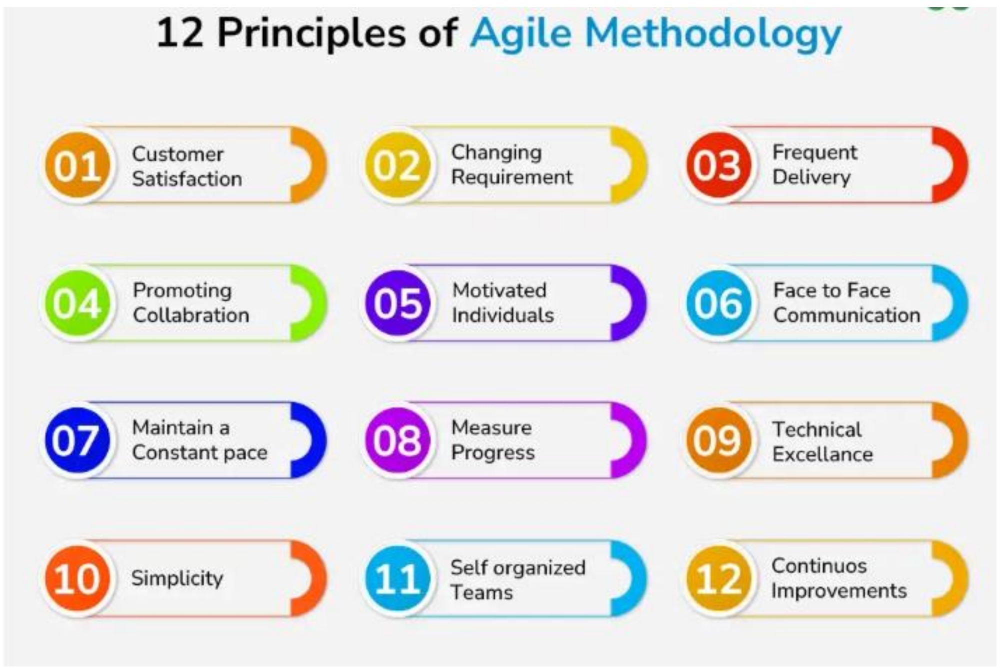
➢ Agile Methodology

What is the Agile Methodology?

Agile Methodology **is a** way to manage projects by breaking them into smaller parts**. It focuses on working together and making constant improvements. Teams plan, work on the project, and then review how things are going in a repeating cycle.**

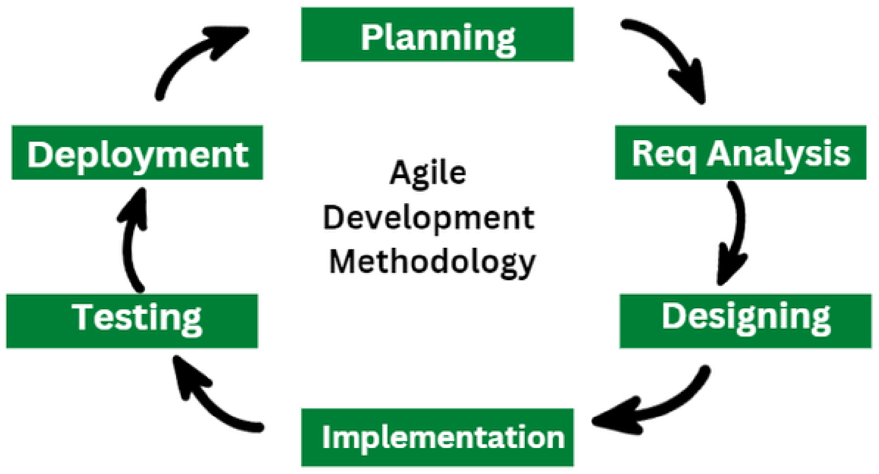
History of Agile

- In 1957, people started figuring out new ways to build computer programs. They wanted to make the process better over time, so they came up with iterative and incremental methods.
- In the 1970s, people started using adaptive software development and evolutionary project management. This means they were adjusting and evolving how they built software.
- In 1990s, there was a big change. Some people didn't like the strict and super-planned ways of doing things in software development. They called these old ways "waterfall." So, in response, lighter and more flexible methods showed up.



Unit 2: Software Process Models

Life cycle of Agile Methodology



1. Requirement Gathering

- In this stage, the project team identifies and documents the needs and expectations of various stakeholders, including clients, users, and subject matter experts.
- It involves defining the **Project's Scope**, objectives, and requirements.
- Establishing a budget and schedule.
- Creating a project plan and allocating resources.

2. Desiging

- Developing a high-level system architecture.
- Creating detailed specifications, which include data structures, algorithms, and interfaces.
- Planning for the software's user interface.

3. Development (Coding)

- Writing the actual code for the software.
- Conducting unit testing to verify the functionality of individual components.

4. Testing

This phase involves several types of testing:
Integration Testing: Ensuring that different components work together.
System Testing: Testing the entire system as a whole.
User Acceptance Testing: Confirming that the software meets user requirements.
Performance Testing: Assessing the system's speed, scalability, and stability.

5. Deployment

- Deploying the software to a production environment.
- Put the software into the real world where people can use it.
- Make sure it works smoothly in the real world.

Unit 2: Software Process Models

- Providing training and support for end-users.

6. Review (Maintenance)

- Addressing and resolving any issues that may arise after deployment.
- Releasing updates and patches to enhance the software and address problems.

<u>Benefits of Agile development methodology</u>

- **Flexibility and Adaptability**: Agile can quickly adapt to changes, allowing teams to respond to new customer needs and market conditions.
- **Improved Collaboration**: Agile encourages constant communication between developers and stakeholders, ensuring the product meets user expectations.
- **Faster Delivery**: Agile ensures quicker releases, keeping customers engaged and their feedback incorporated early.
- **Enhanced Quality and Customer Satisfaction**: Agile focuses on customer feedback, ensuring the product meets their needs and delivering high-quality results.
- **Iterative Development**: Work is done in small, manageable steps, allowing for regular improvements and quick adjustments.
- **Transparency**: Agile keeps stakeholders informed at every stage, ensuring clarity and alignment.
- **Quality Assurance**: Agile prioritizes quality, ensuring the product meets users' expectations through continuous improvements.
- **Continuous Improvement**: Regular feedback ensures the product keeps improving, preventing last-minute issues and maintaining high quality.

Limitations of Agile Methodology

- **Less Documentation:** Agile methodologies focus on less documentation; it prioritizes working on projects rather than paperwork.
- **Challenges in Large Organizations:** Busy schedule of clients can make daily meetup and face-to-face communication difficult.
- **Need for Senior Programmers:** It may require experienced programmers to make critical decisions during the development of software.
- **Limited Scope Control:** It has less rigid scope control, which may not be suitable in certain situations.
- **Predictability**: Compared to more structured project management methods, it may lack predictability.

When to use the Agile Methodology?

- **Unclear or Changing Requirements:** Agile is great for projects with requirements that aren't well-defined or might change.
- **Complex Projects:** It's good for big, complex projects by breaking them into smaller pieces.
- **Customer Focus:** Use Agile when making customers happy is a priority and you want to involve them throughout.
- **Quick Time-to-Market:** If you need to get your product out fast, Agile can help.

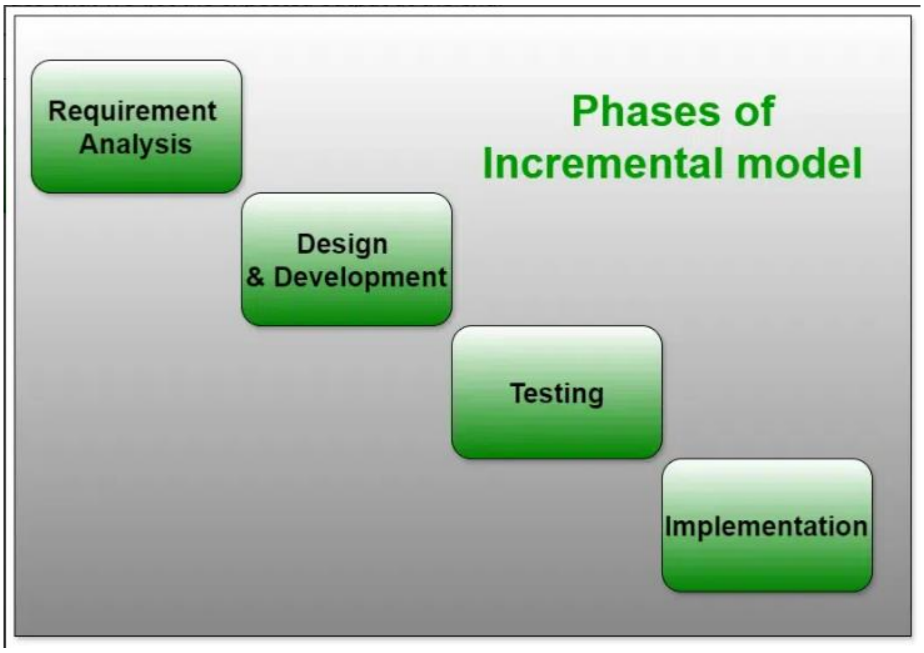
- **Small to Medium Teams:** Agile works well for teams of a few to a few dozen people.
- **Team Skills:** It's best when you have a mix of skills in your team, like development, testing, design, and more.
- **Collaboration:** Agile promotes working together and open communication.
- **Regular Updates:** If you want to check progress often and make changes as needed.
- **Transparency:** Agile emphasizes being open and clear with everyone involved in the project.
- **Risk Control:** It helps manage risks by tackling issues as they come up.
- **Innovation:** If you encourage trying new things and learning from experience, Agile supports that.
- **Continuous Improvement:** Agile fosters a culture of always getting better over time.

Popular Agile Tools for Software Development: Jira, Trello, Axosoft

➢ Incremental Process Model

The Incremental model is a software Development approach which is used to breakdown the project into smaller and easily manageable parts. In these, each part passes through Requirement, Design, Testing phases and Implementation phase. The overall process continue until we got the complete System.

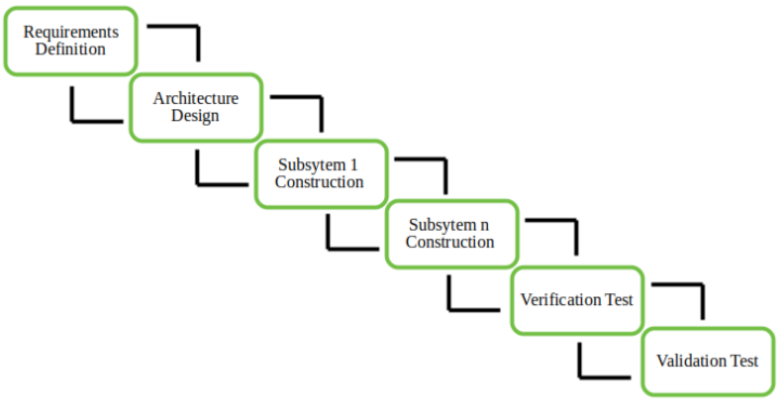
<u>Phases of the Incremental Model</u>



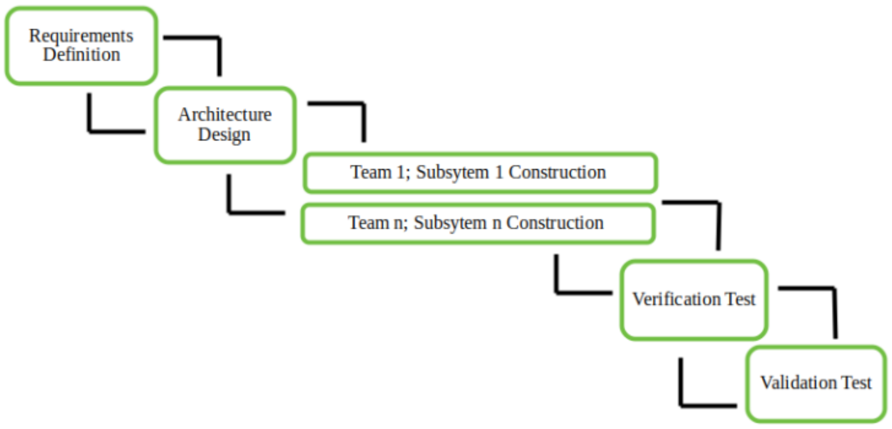
1. **Requirement Analysis**
   The first step in the Incremental Model is understanding what the software needs to do. The team gathers the requirements from the product experts and clearly defines the system's functional needs. This phase is important because it sets the foundation for everything else in the development process.
2. **Design & Development**
   Next, the team focuses on designing how the software will function and starts developing it. They work on adding new features and making sure the system works as expected. The design and development steps go hand-in-hand to build the functionality of the software.
3. **Testing**
   Once a feature is developed, it goes through testing. The testing phase checks how the software performs, including both new and existing features. The team uses different testing methods to make sure everything is working correctly.
4. **Implementation**
   This phase involves writing the final code based on the design and development steps. After testing the functionality, the team verify that everything is working as planned. By the end of this phase, the product is gradually improved and updated until it becomes the final working version.

Types of Incremental Model

1. **Staged Delivery Model:** The Staged Delivery Model develops software in a sequence of planned stages, where each stage delivers a functional part of the system. Each release brings the product closer to completion, allowing it to evolve gradually. Working versions are delivered at regular intervals, making progress visible and manageable throughout the development process.



2. Parallel Development Model: The Parallel Development Model divides the system into multiple modules that are developed simultaneously at the same time by different teams. By working on separate components in parallel, the development process becomes faster and more efficient. This approach reduces overall project time and allows teams to focus on specific functionalities concurrently.



Use Cases of Incremental Process Model

- **When the requirements are well-defined and clear**: Because increments can be planned and developed step-by-step with minimal requirement changes.
- **If the project has a long development timeline**: Incremental delivery helps manage complexity over time by breaking the project into smaller, manageable parts.
- **If the customer needs a quick product release**: You can deliver the most critical features early in the first increment, allowing the customer to start using the software sooner.
- **When you want to develop the most important features first**: This allows early feedback on key functionalities and better prioritization for subsequent increments.

Advantages of Incremental Process Model

- **Faster Software Delivery:** Initial working versions of the software can be delivered quickly. Early delivery increases customer satisfaction and feedback opportunities.
- **Clear Understanding for Clients:** Clients get to see parts of the system at each stage. This visibility ensures that the final product meets their expectations.

- **Easy to Implement Changes:** Requirements can evolve, and changes can be incorporated in subsequent increments. It supports flexibility without heavily disrupting earlier stages.
- **Effective Risk Management:** Risks can be identified and handled early due to the staged approach. Each increment allows for testing and validation, reducing the impact of unforeseen issues.
- **Flexible Criteria and Scope:** Requirements can be adjusted without a major cost increase. Better scope management helps keep the project aligned with business goals.
- **Cost-Effective:** Compared to models like the Waterfall, the incremental model is generally more cost-efficient. Budget is spread across stages, making it easier to manage finances.
- **Simpler Error Identification:** Since development is done in parts, it's easier to pinpoint and fix errors within a specific increment. Testing each module separately enhances quality and reliability.

Disadvantages of Incremental Process Model

- **Requires a Skilled Team and Proper Planning:** Successful implementation demands an experienced team. Poor planning or coordination can lead to confusion between increments.
- **Cost Can Increase Over Time:** Due to repeated testing, redesign, and integration in every cycle, the overall project cost may rise. Continuous iteration involves added overhead
- **Incomplete Requirement Gathering Can Cause Design Issues:** If all requirements are not identified early, the system architecture may not support future needs. Inadequate upfront design can lead to rework and architectural mismatches.
- **Lack of Smooth Flow between Increments:** Each iteration may function independently, which can create inconsistencies. There might be integration challenges when combining all the increments into a unified product.
- **High Effort to Fix Repeated Issues:** A defect in one increment may exist in others. Fixing the same issue across multiple units can be time-consuming and resource-intensive.

➢ Comparison of Models:

Simple Takeaway:

- **Waterfall**: One straight path, no turning back.
- **Iterative Incremental**: Build, test, improve and repeat.
- **Agile**: Fast, flexible, and team-focused.

- **Incremental**: Build piece by piece, adding more each time.

Aspect	Waterfall Model	Iterative Incremental Models	Agile Methodology	Incremental Process Model
Approach	Linear, sequential; each phase completes before the next starts.	Cyclic; builds software through repeated iterations, adding features incrementally.	Iterative and adaptive; delivers small, working software in short cycles (sprints).	Builds software in increments, each adding a functional part of the system.
Flexibility	Low; changes are difficult after a phase is complete.	High; allows changes through iterations.	Very high; embraces changes even late in development.	Medium; changes possible in later increments.
Customer Involvement	Low; mainly at start (requirements) and end (delivery).	Medium; feedback after each iteration.	High; continuous feedback and collaboration.	Medium; feedback after each increment.
Delivery	Single delivery at the end of the project.	Partial working product after each iteration.	Frequent delivery of working software (every 2-4 weeks).	Partial product after each increment.
Risk	High; issues found late	Medium; risks addressed in iterations.	Low; risks managed through frequent feedback.	Medium; risks reduced by early increments.
Best For	Fixed, well-defined requirements	Evolving requirements; large projects needing feedback.	Dynamic projects with frequent changes	Large projects divisible into functional modules.