Table of Contents
Ch 2-1

Introduction:

In this report, explore three fundamental software development models Waterfall,

Incremental, and Agile by breaking down their unique phases, advantages, and disadvantages. Each model offers different approaches to software development, ranging from the linear and structured nature of Waterfall to the flexible, iterative methods of Incremental and Agile. The Waterfall model follows a sequential process, which is straightforward but rigid. The Incremental model, on the other hand, divides the project into smaller modules that are built in stages, allowing for early delivery and regular feedback. Agile combines the best of both iterative and incremental models, emphasizing flexibility, collaboration, and adaptability to changing customer requirements. This report will assess each model, determine the most suitable process for software projects, and incorporate a reuse-oriented approach to enhance development efficiency.

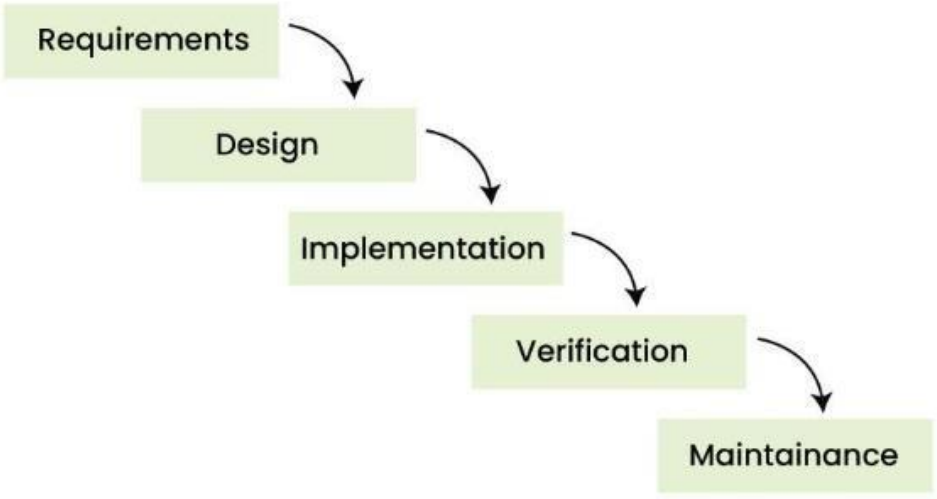
Primary Objective

My primary objective for this project is to analyze three software process models Waterfall, Incremental Development, and Agile by identifying their strengths and weaknesses. I will select the most suitable model for my project, explaining the reasons for my choice. Additionally, I will show how I can incorporate a reuse-oriented approach to improve the efficiency and scalability of my project. The process model I choose will be the one I use throughout the development of my project this semester.

Waterfall Model

The Waterfall model is a classic software development method that follows a linear, step-by-step process. Each phase of development must be completed before the next one begins, and there is no overlap between phases. The model is organized in a strict sequence, which means you cannot go back and make changes to earlier phases once they are completed.

Phases of the Waterfall Model:



- Requirements Phase:

  This is the initial phase where all the system's requirements are gathered and documented in detail. The success of the entire project hinges on how accurately the requirements are defined, as these will guide the rest of the development process.

- Design Phase:

  After the requirements are finalized, the design phase focuses on creating a detailed plan for how the system will be built. This involves creating system architectures, data models, and software specifications that will ensure the system is built to meet all requirements.

- Implementation Phase:

  This is the phase where the actual development happens. The system is coded and all components, such as hardware and software, are integrated. The system undergoes initial testing to ensure everything works according to the design.

- Verification Phase:

  Once the implementation is done, the software goes through extensive testing. This is where the system is evaluated to ensure it meets the requirements. The goal is to identify and fix any bugs or issues before the software is deployed.

- Maintenance Phase:

  After the software is delivered and deployed, the maintenance phase begins. This phase includes fixing any bugs discovered by users, making updates, and ensuring that the software continues to meet users' needs as time goes on.

Advantages of the Waterfall Model:

- It’s easy to follow and manage because of its structured approach.
- Well-suited for smaller projects where the requirements are clear and unlikely to change.
- Each phase is well-documented, making it easier to understand the progress at any given point.
- The clear separation between phases ensures that teams know exactly what needs to be done before moving on.

Disadvantages of the Waterfall Model:

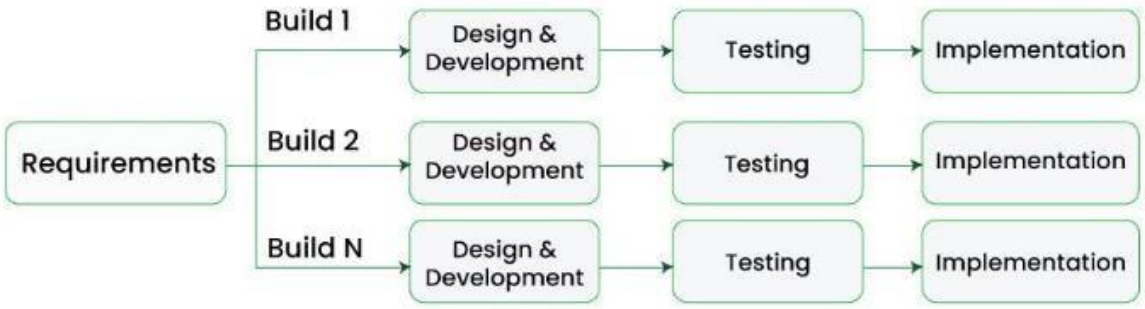
- It lacks flexibility; once a phase is complete, going back to make changes is very difficult, making it unsuitable for projects with evolving requirements.
- There is a long delay before any working software is available to test or use.
- Since testing happens after the implementation phase, problems found late in the process can be costly to fix.
- It’s not ideal for complex projects where constant changes and updates are needed during development.
- The sequential nature of this model does not reflect the more dynamic and iterative processes used in modern software development.

Incremental Model

The Incremental Model breaks down the software development process into smaller, manageable parts called increments. Instead of developing the entire system at once, the software is built in multiple stages. Each increment focuses on a specific set of requirements and builds a functional part of the software.

How the Incremental Model Works:

In this model, the customer’s requirements are first gathered, and then a portion of these requirements is used to create an initial module or functionality of the software. This module is delivered to the customer, who can begin using it right away. After that, more requirements are gathered, and another module is developed and added to the system. This process continues until the complete software is built. The early stages focus on developing the core functionalities, with additional features being added in each subsequent increment.
The key idea is that user feedback is incorporated after each increment, allowing for improvements and adjustments along the way. This helps ensure that the system being developed meets the user’s expectations.



Phases of the Incremental Model:

- Communication:

  The process starts by talking with the customer to collect the key requirements they want in the software. This phase is crucial to ensure that each increment delivers value to the customer.
- Planning:

  In this phase, the requirements are divided into multiple increments or modules. A plan is made on how each module will be built and delivered to the customer.
- Modeling:

  Each module is designed separately during the modeling phase. Detailed design documents, like ERDs (Entity-Relationship Diagrams) and DFDs (Data Flow Diagrams), are created to guide the development of each increment.
- Construction:

  This phase involves coding and testing the individual modules based on their designs.

  Each increment is developed, tested for functionality, and prepared for delivery.
- Deployment:

  Once a module has passed testing, it is delivered to the customer. The customer can start using the software and provide feedback. After deployment, the process repeats for the next module until the entire system is completed.

Advantages of the Incremental Model:

- Important functionalities are developed early, allowing the customer to use parts of the software sooner.
- Feedback can be collected from the customer after each increment, making it easier to adapt to changes in requirements.
- It is easier to manage and test smaller, individual increments, which makes debugging and error identification more straightforward.
- This model offers more flexibility in changing requirements compared to linear models like Waterfall, and the overall cost of making changes is lower.
- Progress is visible because working software is delivered at each stage of the development cycle.
-

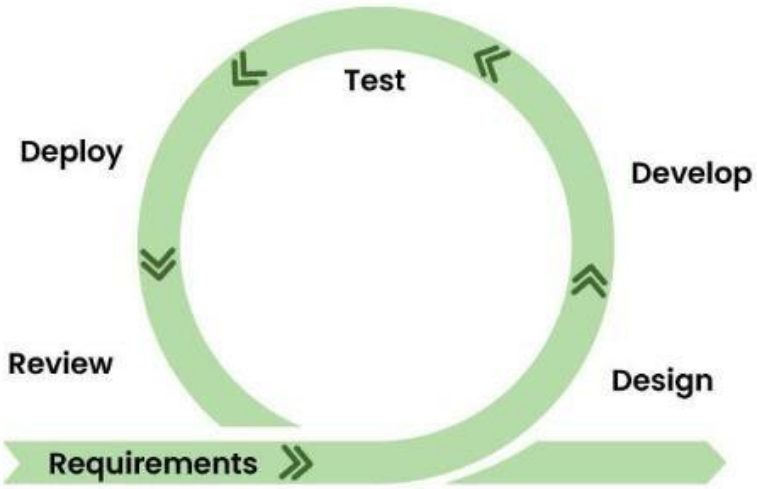
Disadvantages of the Incremental Model:

- Managing the project requires continuous oversight, as the development happens in stages.
- Before starting the incremental development, the complete software requirements must be clearly understood.
- This model requires thorough planning and design for each increment, which can be challenging and time-consuming.
- While the initial cost of development might be lower, the total cost of the project may be higher due to the repeated cycles of planning, building, and testing each increment.

Agile Model

The Agile model combines the principles of iterative and incremental models, focusing heavily on flexibility, adaptability, and customer satisfaction. It was developed to address the limitations of earlier models, like the Waterfall model, which struggled to accommodate changes during the software development process. Agile allows for continuous feedback and iteration, making it possible to adapt to new requirements or changes without disrupting the entire project.

How the Agile Model Works:



In Agile, the software is developed in small, manageable increments, with each increment being developed over multiple iterations. Each iteration lasts a few weeks and focuses on creating a working version of the software that can be reviewed by stakeholders. Agile emphasizes face-to face communication and collaboration, involving a customer representative who works directly with the development team to ensure that the product meets the customer's expectations. After each iteration, the team demonstrates the working software to the customer, gathers feedback, and adjusts as needed.

Principles of Agile Model:

- A customer representative works closely with the development team to provide ongoing feedback and clarify requirements during development.
- After every iteration, stakeholders and the customer representative review the progress and provide feedback.
- The focus is on delivering working software in small, incremental versions to the customer, typically every few weeks.
- Agile emphasizes small, highly communicative development teams, usually consisting of 5 to 9 people, to facilitate collaboration and adaptability.

- Agile promotes flexibility by encouraging rapid changes and updates whenever needed.
- Developers often work in pairs, where one person codes, and the other reviews the code.
  They frequently switch roles, ensuring continuous collaboration and error-free code.

Advantages of Agile Model:

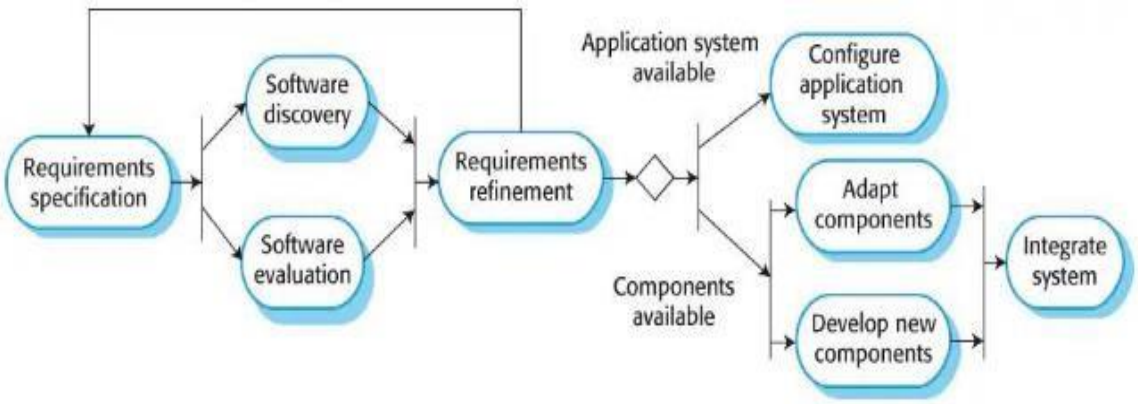
- The collaborative approach between programmers ensures fewer errors in the code as peer review is continuous.
- Software is delivered quickly, with incremental improvements provided regularly, shortening the development timeline.
- Since the customer is involved in every iteration, requirements can be adjusted easily, and the final product better aligns with customer expectations.
- Agile promotes teamwork and communication, ensuring all team members are aware of project goals and challenges.
- With minimal rules and documentation, Agile allows for flexibility and fast adaptation to changes.
- It is highly flexible and easy to manage because of its iterative nature.

Disadvantages of Agile Model:

- Agile can struggle to handle projects with complex interdependencies, as its focus is on smaller, incremental deliveries.
- The lack of formal documentation can lead to confusion among team members, especially in large projects.
- Agile heavily depends on the customer representative. If they provide incorrect information, it could lead to incorrect software development.
- Only experienced developers are empowered to make decisions, which can limit the participation of newer or less experienced team members.

Reuse-Oriented Model (ROM)

The Reuse-Oriented Model (ROM), also referred to as reuse-oriented development (ROD), is a method within software development that focuses on building software through a series of prototypes. Each prototype, or version, builds upon the previous one by following a structured set of guidelines, ensuring continuity and evolution of the system.
In practice, using the ROM approach can be challenging because there may not always be a complete set of reusable components available. When reusable parts are insufficient or incompatible, new components may need to be created from scratch. If this isn't done, the model may need to make compromises in meeting user requirements, leading to a final product that might not align precisely with initial expectations. The underlying concept of this model is that maintenance often involves reusing and adapting existing components.



The ROM approach follows these main steps:

1. Identify Reusable Components

   Recognize which parts of the existing system are suitable for reuse in the new project.

2. Understand the System

   Analyze all the components to understand how they fit within the current and new system requirements.

3. Adapt Components to New Requirements

   Modify existing components as needed to ensure they meet the new project requirements.

4. Integrate Modified Components

   Combine the adapted components to form the new, functional system.

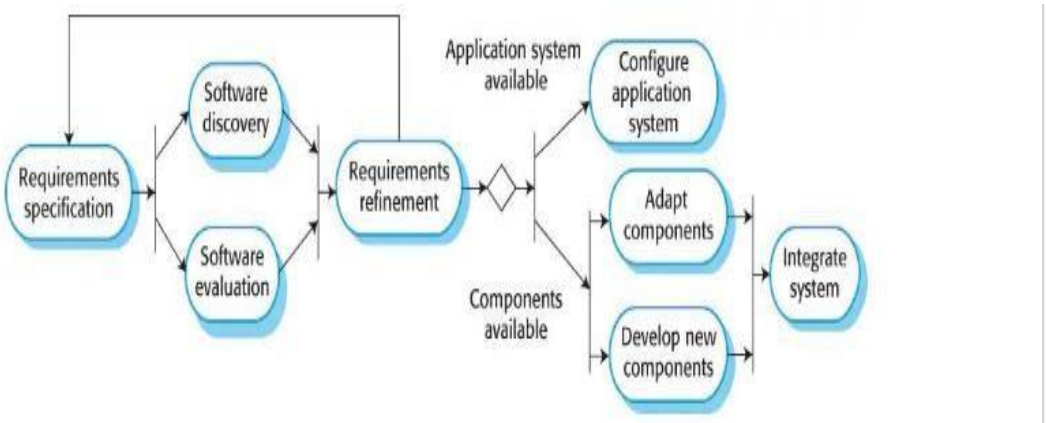
Advantages:

- ROM can significantly lower the overall cost of development.
- The model minimizes risks.
- It saves time and reduces the effort required for software creation.
- Efficient and adaptable by nature.

Disadvantages:

- ROM may not always work in its ideal form, as it's sometimes hard to find compatible components for reuse.
- Requirements may be compromised, which could result in a product that doesn't meet user needs completely.
- Using outdated components can create compatibility issues, potentially affecting future system updates.

Reuse-oriented approach



Requirements Specification

- This is the phase where I define the system's needs based on the project objectives. For example, the *Advanced Smart Inventory Stock Management System* will require:

- Real-time inventory tracking
- Automatic restocking features
- Integration with e-commerce platforms and mobile access

- Custom reporting and analytics for decision-making
- Barcode scanning for receiving goods and managing stock

Software Discovery

- During software discovery, I will identify existing tools or modules that can fulfill the system's needs. This step involves finding components that may already exist to handle the functionalities defined during the requirements specification. Some of the components I can reuse are:

- Barcode Scanning: Use pre-built barcode scanning APIs such as Zebra or Datalogic to streamline stock updates.
- POS Systems: Integrate with existing point-of-sale (POS) systems like Square or Shopify to handle sales and automatically update stock levels.
- E-commerce Integration: Explore APIs from platforms like Shopify or WooCommerce for online store synchronization.

Software Evaluation

- After identifying potential reusable components, I will evaluate them to ensure they meet the project's specific requirements. This might include:
- Testing different barcode scanning solutions to verify they meet the speed and accuracy needed.
- Evaluating POS systems for seamless integration and ensuring they support real-time stock updates.
- Checking whether existing mobile frameworks (e.g., React Native, Flutter) can be adapted for mobile access to manage inventory remotely.

Requirements Refinement

- Based on the results of the software evaluation, I may need to adjust or refine the system's requirements. For example, if a selected POS system has limitations in handling real-time updates, I may need to adjust how this feature is implemented or select a more suitable POS system that meets all requirements.

Application System Availability Check

- At this point, I will determine whether there is an entire application system available or whether individual components will need to be adapted. If there is a complete solution, like a configurable POS system that fulfills the needs of the project, I will proceed to configure it. If only individual components like barcode scanners or reporting tools are available, I will move to the adaptation phase.

Configure Application System

- If a complete application system, like a POS or inventory management platform, is available, I will configure it to meet the specific needs of the *Advanced Smart Inventory Stock Management System*. For example, configuring Shopify or Square to handle stock updates and online transactions can streamline this process.

Adapt Components

- If individual components need to be adapted to fit the project's requirements, I will modify these modules to meet specific needs. For example:
- Adapting barcode scanning APIs to interface with the inventory database.
- Customizing mobile frameworks (React Native, Flutter) to handle inventory management features like real-time updates and remote access.
- Adjusting e-commerce APIs to synchronize inventory levels with the online store.

Develop New Components

- If there are no existing solutions for a specific feature, I will develop new components. For example, if I cannot find an appropriate reporting module, I may need to create a custom reporting tool that allows users to generate and visualize detailed reports on inventory levels, sales, and order history.

Integrate System

- After configuring, adapting, or developing the necessary components, the next step is integrating all of them into a cohesive system. This integration includes:
- Linking barcode scanning, mobile access, and POS systems for seamless inventory management.
- Integrating automated restocking with inventory monitoring to ensure that purchase orders are generated when stock is low.
- Combining reporting tools, mobile access, and e-commerce platforms to create a comprehensive system that handles all aspects of inventory management.

Conclusion:

After careful consideration of various software development models, I have chosen to adopt the

Agile methodology for my project, with an incorporation of Incremental development principles. The decision to use Agile is based on its flexibility, adaptability, and strong focus on customer collaboration, all of which are crucial for the successful delivery of the *Advanced Smart Inventory Stock Management System*. Agile allows for ongoing changes and feedback, which is important in an inventory management project that must remain responsive to evolving business needs.

The Incremental model will complement Agile by enabling the project to be developed in smaller, manageable stages. This allows me to deliver core features—such as real-time stock updates, automatic restocking, and mobile access—early in the process. Each stage will be built upon, refined, and expanded in subsequent increments, ensuring that the system grows in functionality while maintaining a clear structure.

By combining Agile's iterative and flexible approach with Incremental's structured delivery of features, I will be able to manage the complexity of the project as it evolves, ensuring both customer satisfaction and timely delivery of a robust, scalable inventory management system.