# Software Development Lifecycle

# Agenda

In this session, we will discuss:

- Software Engineering
- Key Aspects of Software Engineering
- Importance of Software Engineering
- Introduction to Software Development Lifecycle (SDLC)
- Importance of SDLC
- Different Types of Software Development Models
- Real-Time Applications
- Difference Between Software Models

# Software Engineering

# Software Engineering

- Software engineering is a discipline that involves the application of engineering principles
  - to the creation,
  - development,
  - maintenance,
  - testing, and
  - evaluation of software and systems

that make computers or anything containing software work.

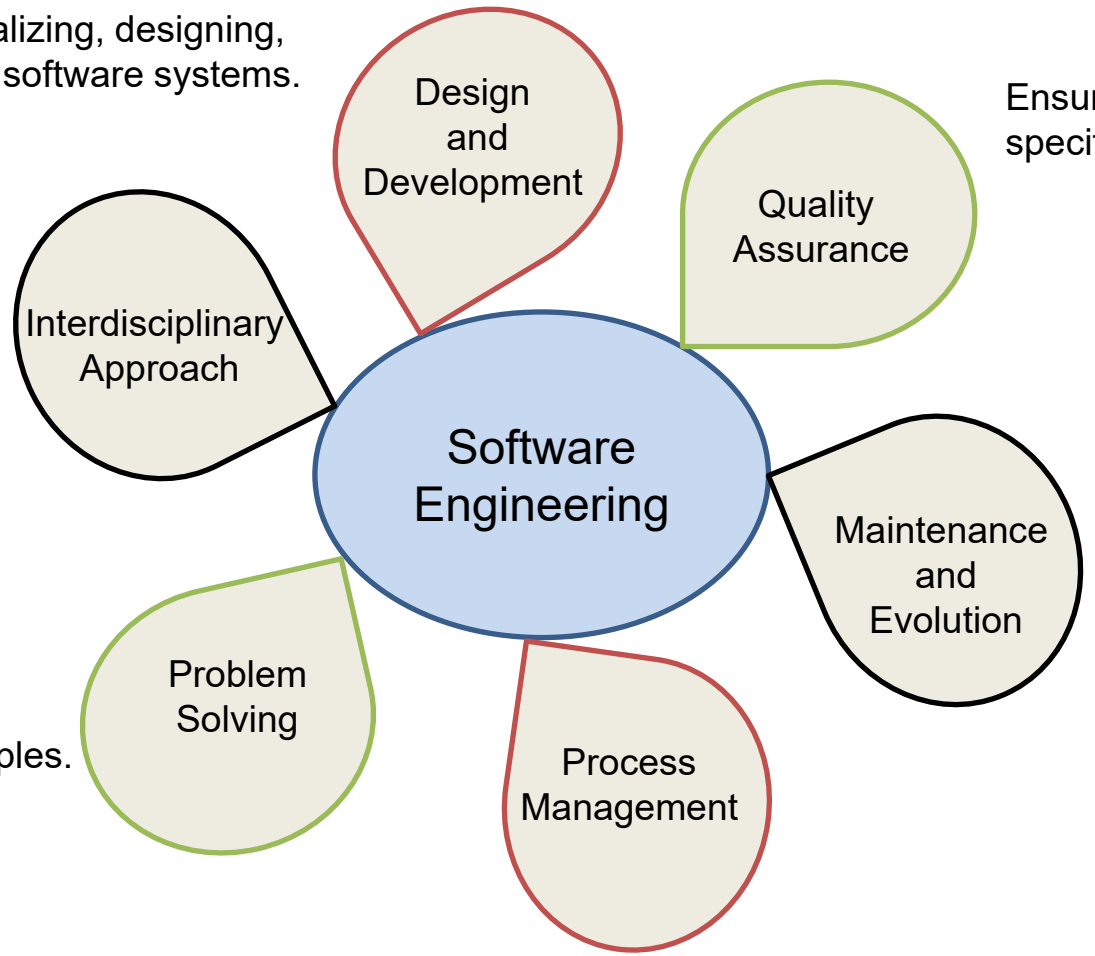

SOFTWARE

- It is a systematic, disciplined, and quantifiable approach to the development, operation, and maintenance of software.

# Key Aspects of Software Engineering

# Key Aspects of Software Engineering

It involves conceptualizing, designing, and building reliable software systems.




Ensuring that the software meets specified requirements and is free of bugs.

It combines elements from computer science, mathematics, and engineering, as well as domain-specific knowledge.

Software needs to be updated, upgraded, and sometimes completely redesigned to adapt to new requirements or technology environments.

Software engineering is about solving problems efficiently and effectively through the application of technology and scientific principles.

Software engineering focuses on improving and managing the process of software development itself.

# Importance of Software Engineering

# Importance of Software Engineering

- **Economic and Social Impact:** Software is integral to the functioning of modern society, impacting all sectors from healthcare to finance and entertainment.
- **Complexity Management:** As software systems become increasingly complex, the role of software engineering in managing this complexity is crucial.
- **Safety and Reliability:** Especially in critical applications ensuring the reliability and safety of software is paramount.
- **Innovation and Adaptation:** Software engineering drives technological innovation and adapts to the rapidly changing technology landscape.

# Software Development Lifecycle

# Software Development Lifecycle

- The Software Development Lifecycle (SDLC) is a systematic process used by software engineers to design, develop, and test high-quality software.
- The SDLC provides a structured framework that encompasses the entire life of a software product, from its initial conception to its eventual retirement.
- It provides a structured sequence of stages in software development that helps organizations to produce high-quality software in a cost-effective way.

# Importance of SDLC

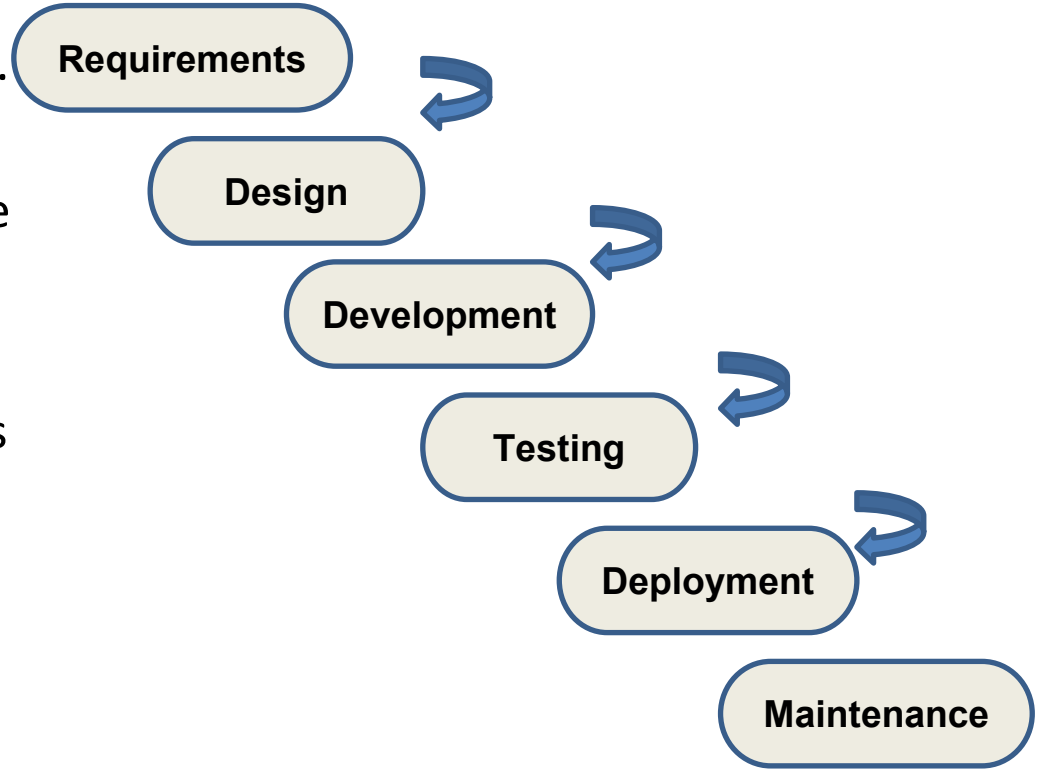
# Importance of SDLC

- **Predictability and Control:** Provides a structured approach, making it easier to control and manage the development process.
- **Quality Assurance:** Through predefined standards and guidelines, the SDLC ensures higher software quality.
- **Risk Management:** Early identification and mitigation of risks due to a systematic approach.
- **Resource Management:** Efficient allocation and utilization of resources due to planned stages.
- **Documentation and Standardization:** Facilitates better documentation and standardization of the development process, aiding in scalability and future maintenance.

# Waterfall Model

# Waterfall Model

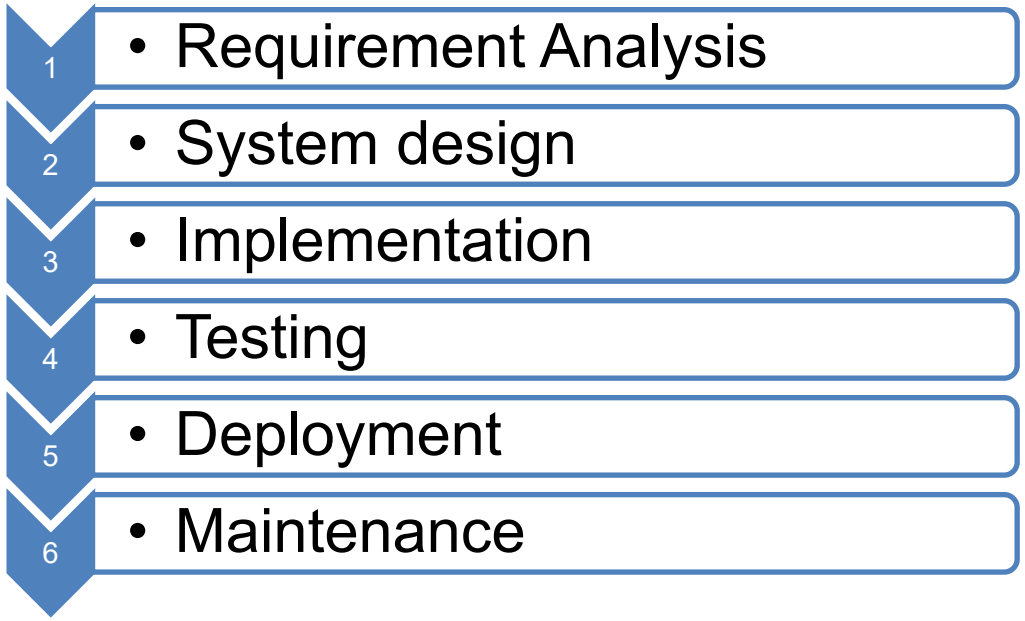
- The Waterfall model is one of the earliest SDLC models.
- It is a linear and sequential approach where each phase must be completed before the next phase begins.
- Best suited for projects with well-defined requirements and where changes are not expected during the development process.
- Phases: Requirements → Design → Development → Testing → Deployment → Maintenance .



# Phases in Waterfall Model

**Requirement Analysis and Specification:**

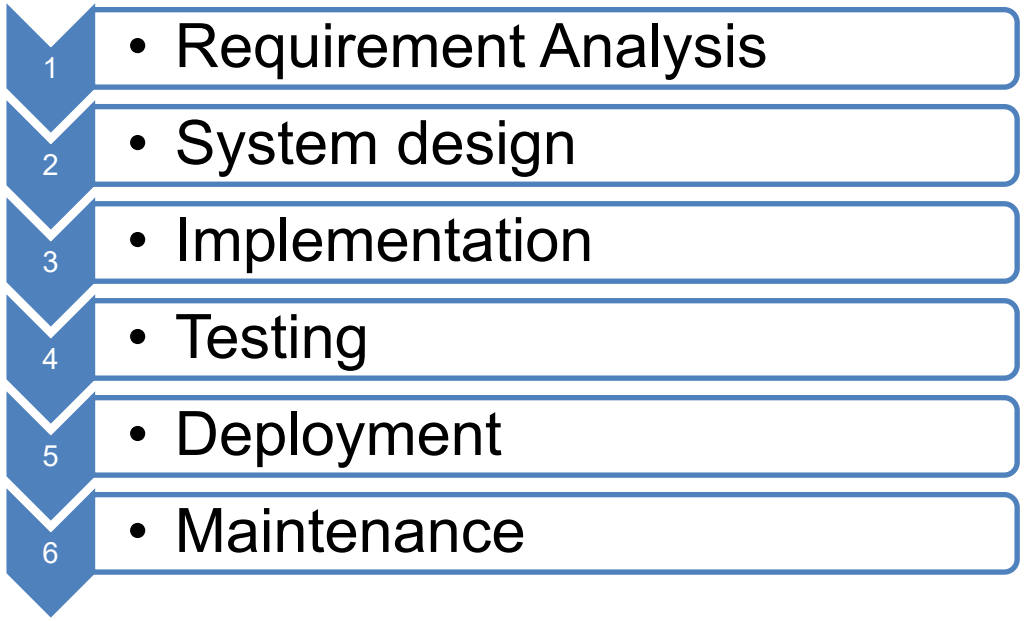
- The primary goal is to comprehensively grasp and accurately document the customer's specific needs.
- This collaborative process involves both the customer and the software developer, who work in tandem to capture all necessary details about the software's functionality, performance, and interface requirements.
- The focus of this phase is on detailing "what" the system is expected to achieve, rather than delving into the "how" of the system's operation.
- As a result of this phase, an extensive document known as the Software Requirement Specification (SRS) is produced.



# Phases in Waterfall Model

**Design Phase:**

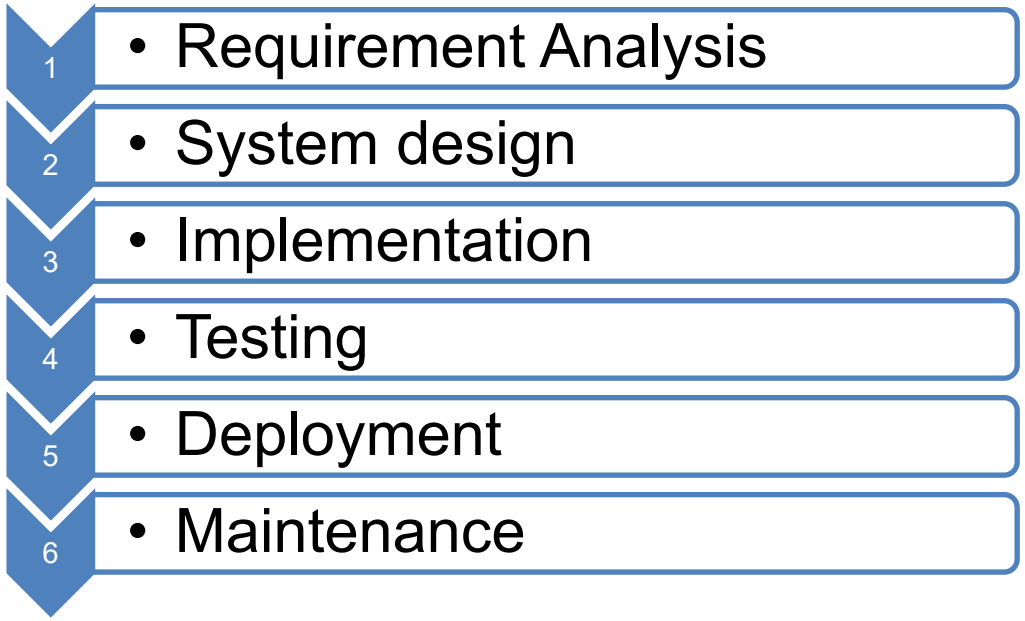
- The objective is to convert the requirements detailed in the Software Requirement Specification (SRS) into a format that facilitates subsequent coding in a programming language.
- This phase involves defining the overall architecture of the software as well as its high-level and detailed design aspects.
- The outcomes of this process are systematically recorded in a document known as the Software Design Document (SDD).



# Phases in Waterfall Model

**Implementation and Unit Testing:**

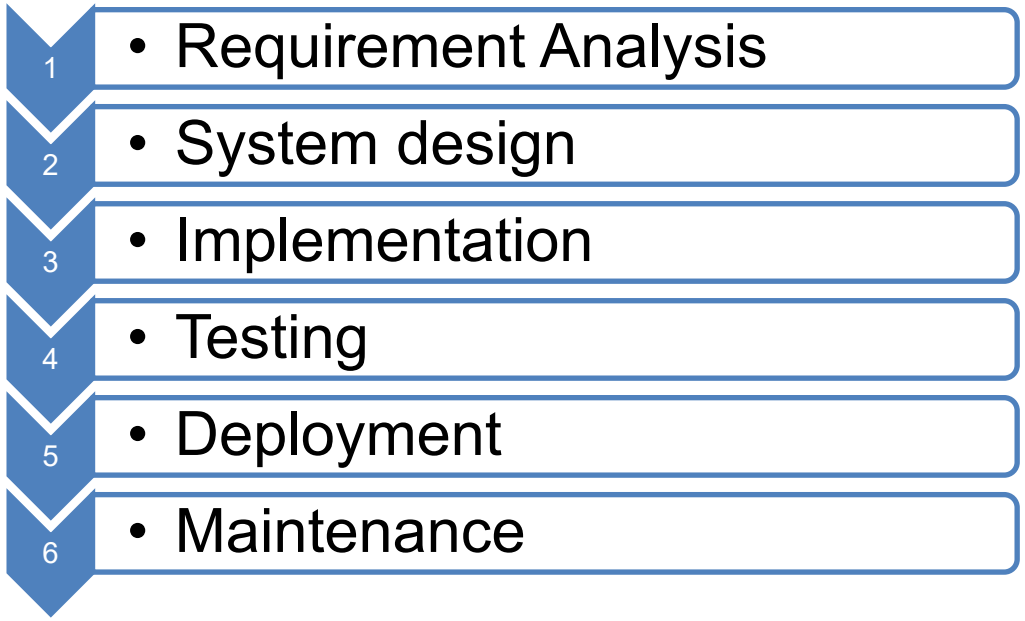
- In the Implementation and Unit Testing phase, the designs laid out in the Software Design Document (SDD) are brought to life through coding.
- The testing part of this phase involves an extensive examination and refinement of the code.
- Initially, individual small modules are tested in isolation to ensure their functionality. Subsequently, these modules undergo further testing, where additional code is written to assess the interaction between the modules and to monitor the flow of intermediate outputs.
- This process helps in ensuring that all parts of the software work together as intended.



# Phases in Waterfall Model

**Integration and System Testing:**

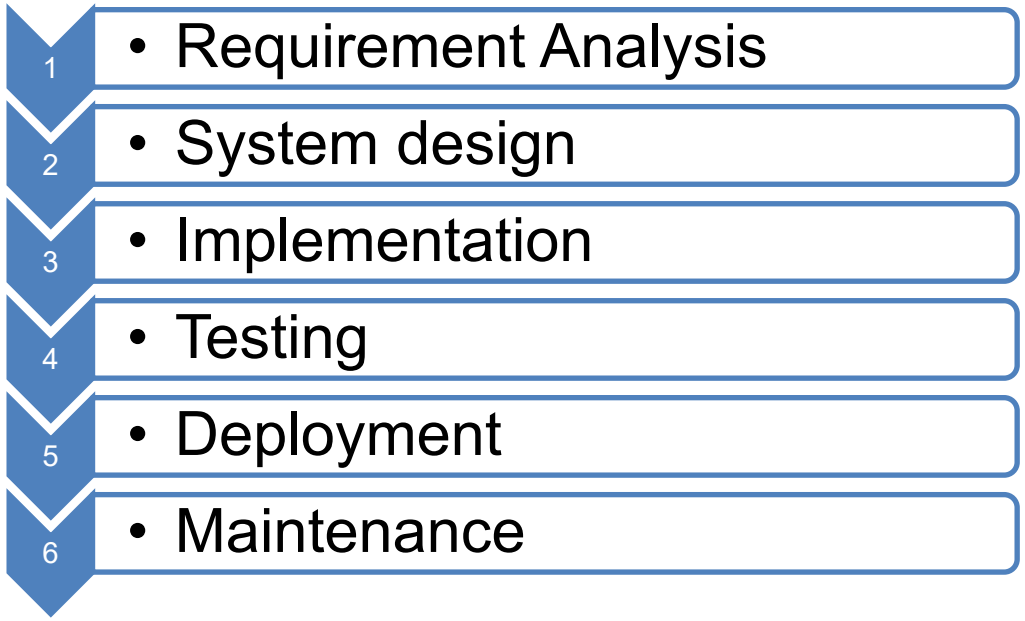
- The Integration and System Testing phase plays a pivotal role in determining the quality of the final software product.
- The effectiveness of the testing conducted during this phase directly impacts customer satisfaction, maintenance costs, and the accuracy of the software.
- While unit testing assesses the performance of individual modules, this phase focuses on evaluating how these modules interact with each other and with the overall system.
- Successful testing in this stage is key to ensuring the software functions correctly as a unified whole.



# Phases in Waterfall Model

**Maintenance:**

- It is a critical stage that commences once the software has been delivered, installed, and is operational at the customer's end. In this phase, the software is actively used and maintained.
- Maintenance involves ongoing activities undertaken by users to ensure the software continues to function effectively and efficiently.
- This phase includes updating the software to adapt to new environments or requirements, fixing any issues or bugs that arise, and improving performance as needed.



# Banking Software System Development

- A bank decides to develop a new software system to handle customer transactions, account management, and back-office banking operations.
- The requirements for such a system are typically well-defined and stable.
- Banking regulations, transaction protocols, and account management processes are established and not subject to frequent changes.
- The project involves multiple departments, including finance, IT, and compliance, each with a clear understanding of their requirements.

# Application of the Waterfall Model

**System Design**
Designing the overall system architecture.
Creating database schemas for customer data, transaction records, and account information.

**Requirements Analysis**
Gathering detailed requirements from all stakeholders, including compliance officers, bank tellers, and IT staff.

**Implementation**
Developing separate modules for transactions, account management, and reporting.

**Verification**
Conducting rigorous testing, including system testing, integration testing, and user acceptance testing.

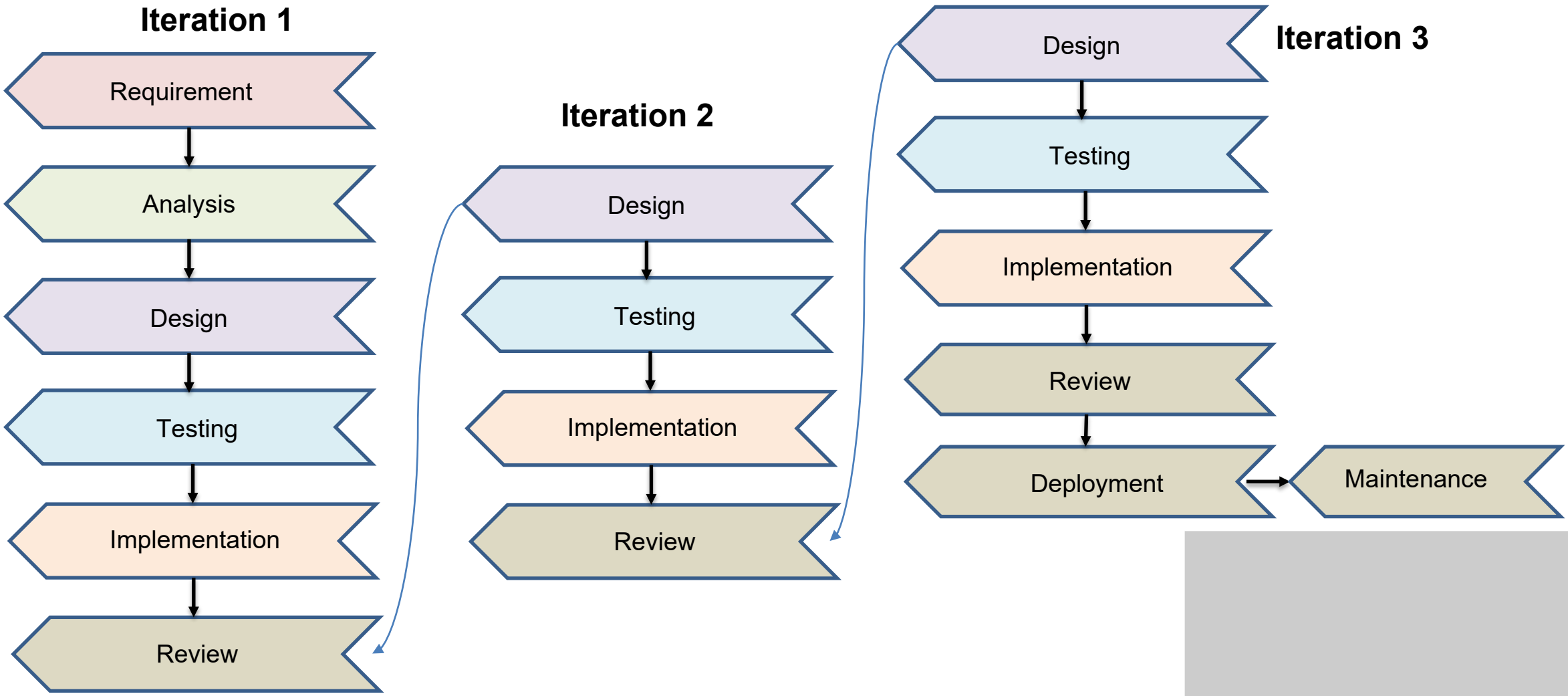
**Maintenance**
Once deployed, the system enters the maintenance phase where bugs are fixed, and minor updates are made.

# Iterative Model

# Iterative Model

- The Iterative Model is a software development approach that emphasizes the repetition of a cycle of software development activities.
- In this model, the software is developed and refined through repeated cycles (iterations), with each iteration typically producing a new version of the software.
- It allows for partial use of the software and its refinement based on feedback until the final complete system is implemented.

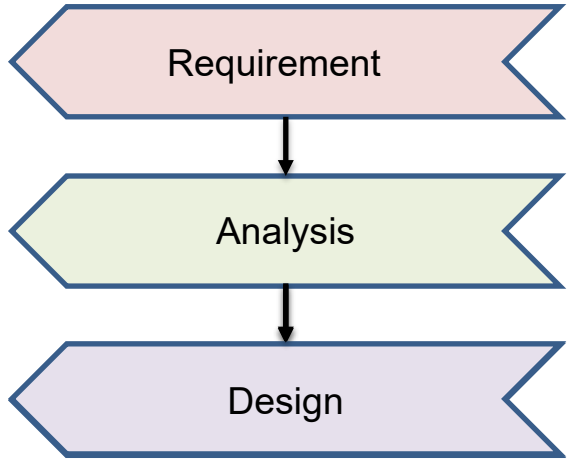
# Iterative Model



# Iterative Model

**Requirement Collection & Analysis:**

- This initial phase involves gathering customer requirements and assessing them for feasibility.
- An analyst evaluates whether these requirements can be met within the budget and scope.
- Once this assessment is complete, the team moves on to the next phase.




**Design Phase:**

- Here, the team develops the software's design using various diagrams like Data Flow diagrams, activity diagrams, class diagrams, and state transition diagrams.

# Iterative Model

**Implementation:**

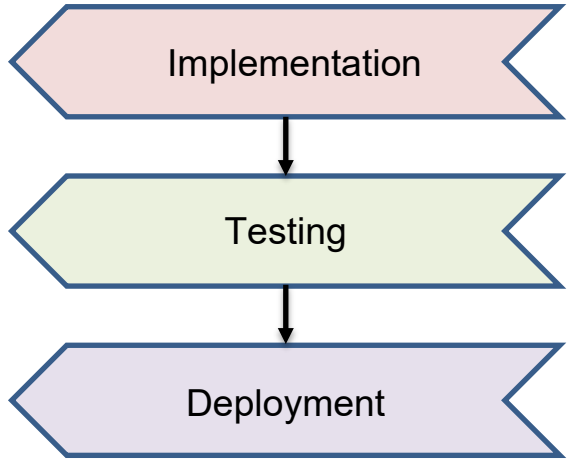
- During this stage, the gathered requirements are translated into code, effectively turning them into computer programs, which constitute the software.

**Testing:**

- Once coding is completed, the software undergoes rigorous testing through different methodologies, including white box, black box, and grey box testing techniques.

**Deployment:**

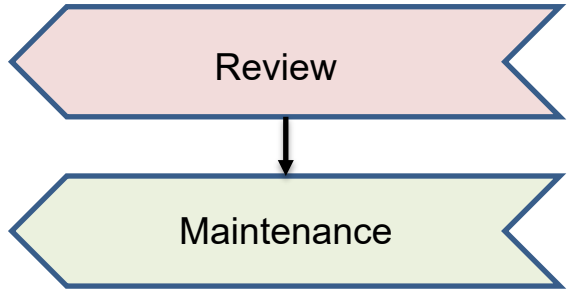
- After all previous phases are successfully completed, the software is deployed into its operational environment.



# Iterative Model

**Review:**

- Post-deployment, a review is conducted to evaluate the software's performance and accuracy.
- Should any issues be identified, the process may cycle back to the requirement gathering phase.




**Maintenance:**

- The final phase involves maintaining the software in its operational environment.
- This includes fixing bugs, resolving errors, and making updates or additions as necessary.

# Development of a Custom Content Management System (CMS)

- A media company requires a CMS to manage its vast array of content, including articles, videos, and images.

- The requirements are complex and expected to evolve as the system is developed and user feedback is integrated.

# Application of Iterative Model

**Initial Iteration – Basic CMS Structure**
Initial requirements gathered focus on core functionalities like article creation, editing, and publishing.

**Second Iteration – Advanced Features**
Based on feedback, the second iteration includes the development of advanced features like multimedia support, SEO tools, and user analytics.

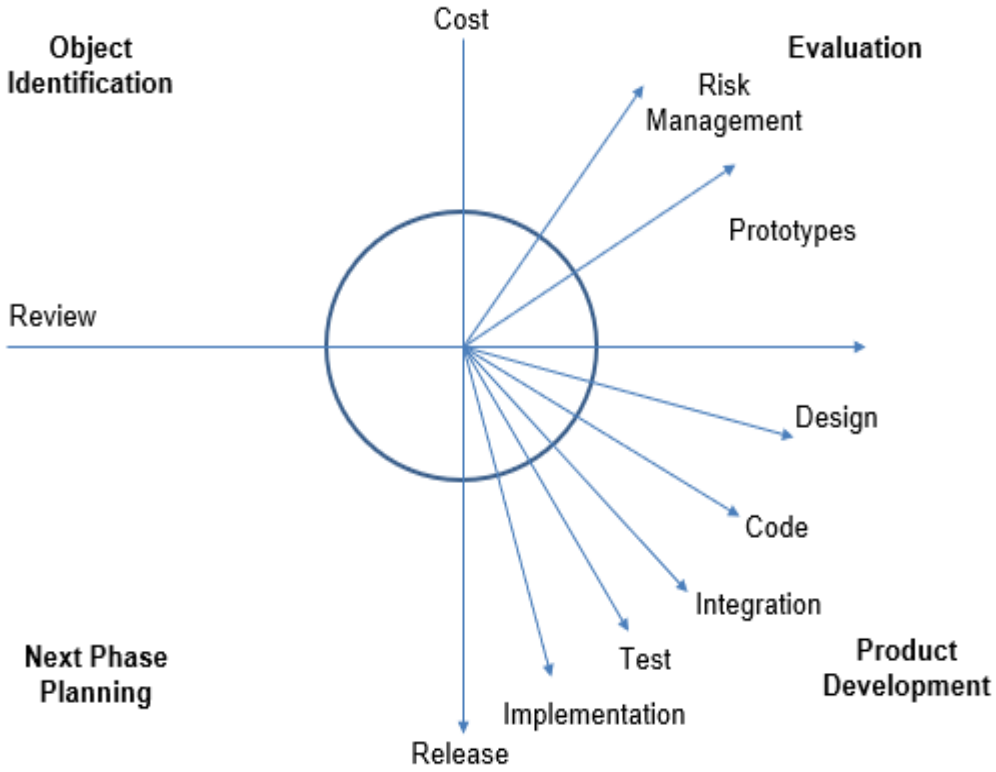
**Ongoing Iterations for New Technologies and Trends**
Regular updates ensure the CMS stays relevant and efficient in the dynamically changing media landscape.

**Subsequent Iterations – Continuous Improvement**
Further iterations involve adding more features such as social media integration, advanced search capabilities, and mobile responsiveness.

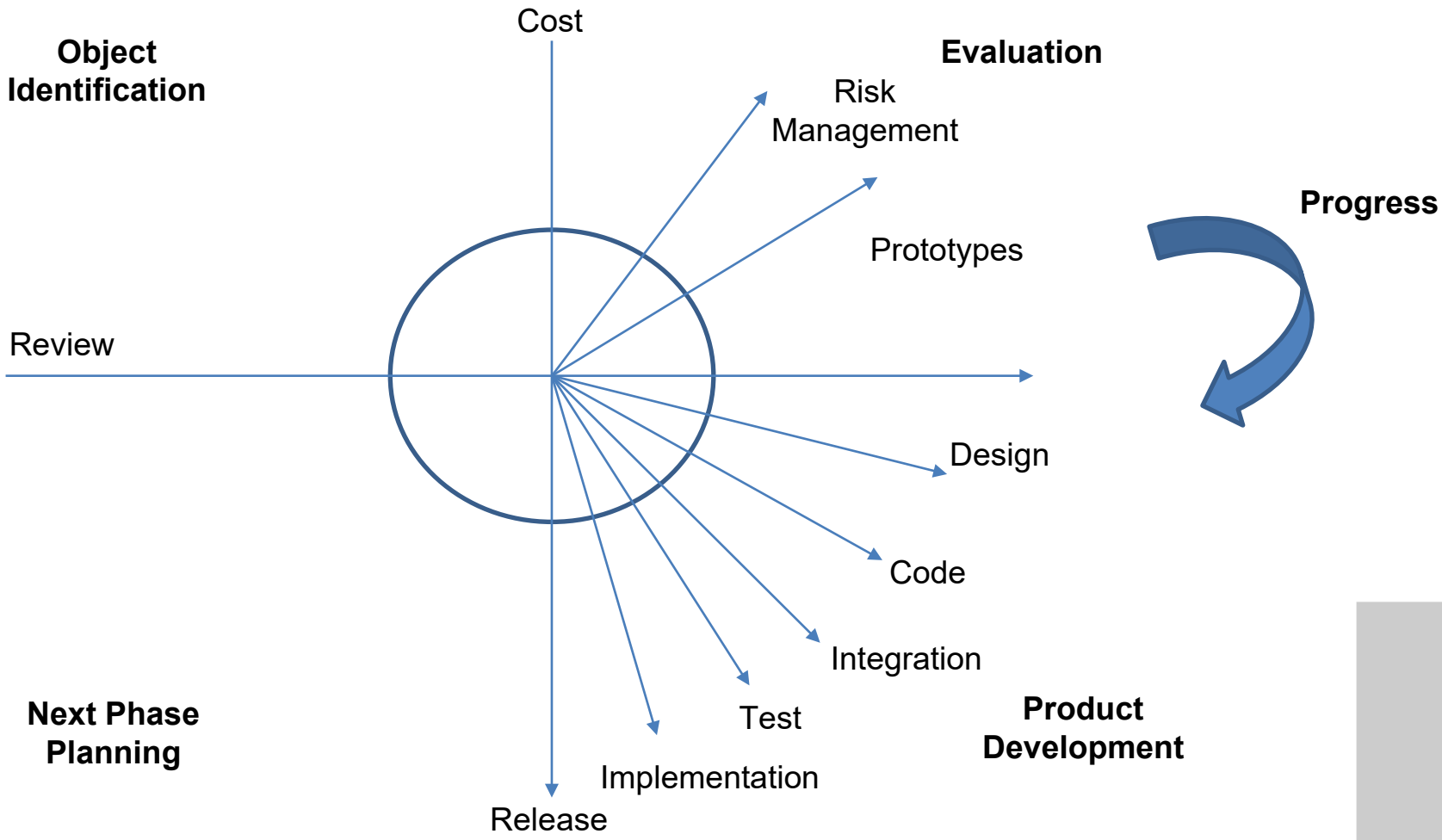
# Spiral Model

# Spiral Model

- It is a risk-driven software development process model that combines elements of both iterative development and the Waterfall model.
- It is particularly noted for its focus on risk analysis and is often used for large, complex, and high-risk projects.
- The model emphasizes iterative refinement through a series of cycles, allowing for incremental releases of the product.



# Spiral Model



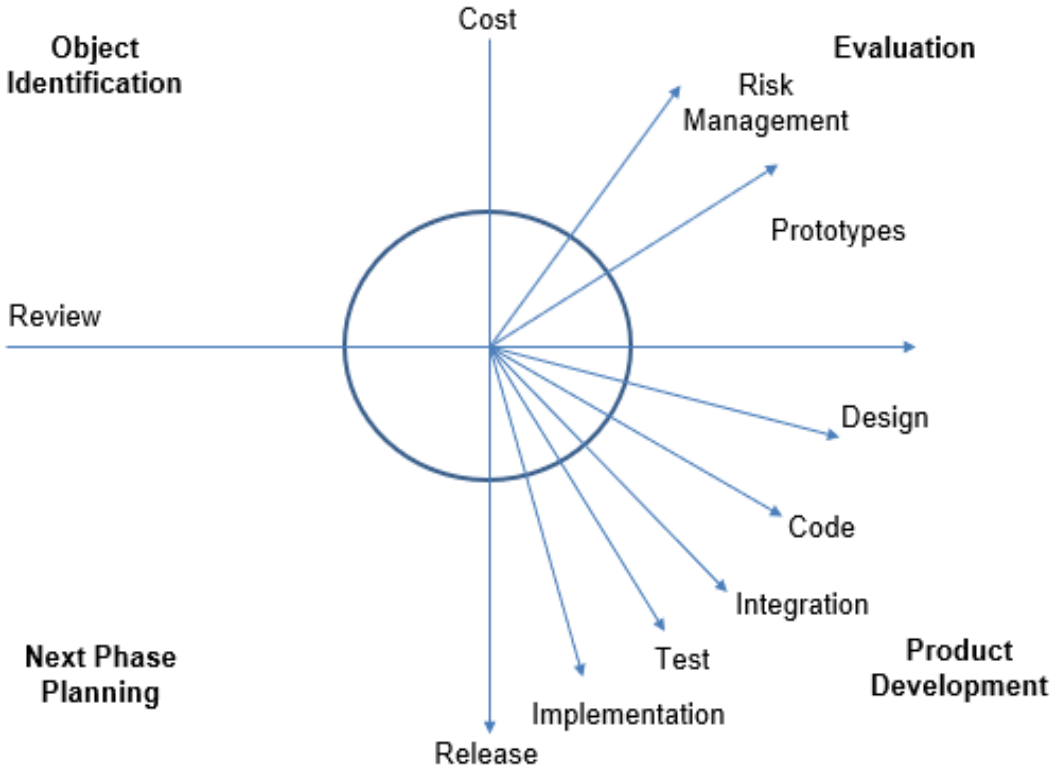
# Spiral Model

**Object Identification:**

- Every iteration of the spiral begins by establishing the goals for that cycle, exploring different possible approaches to achieve these objectives, and recognizing any existing limitations or constraints.

**Risk Management:**

- In this stage, the identified alternatives are evaluated in the context of the set objectives and constraints. The primary focus here is on understanding and addressing the potential risks associated with the project.



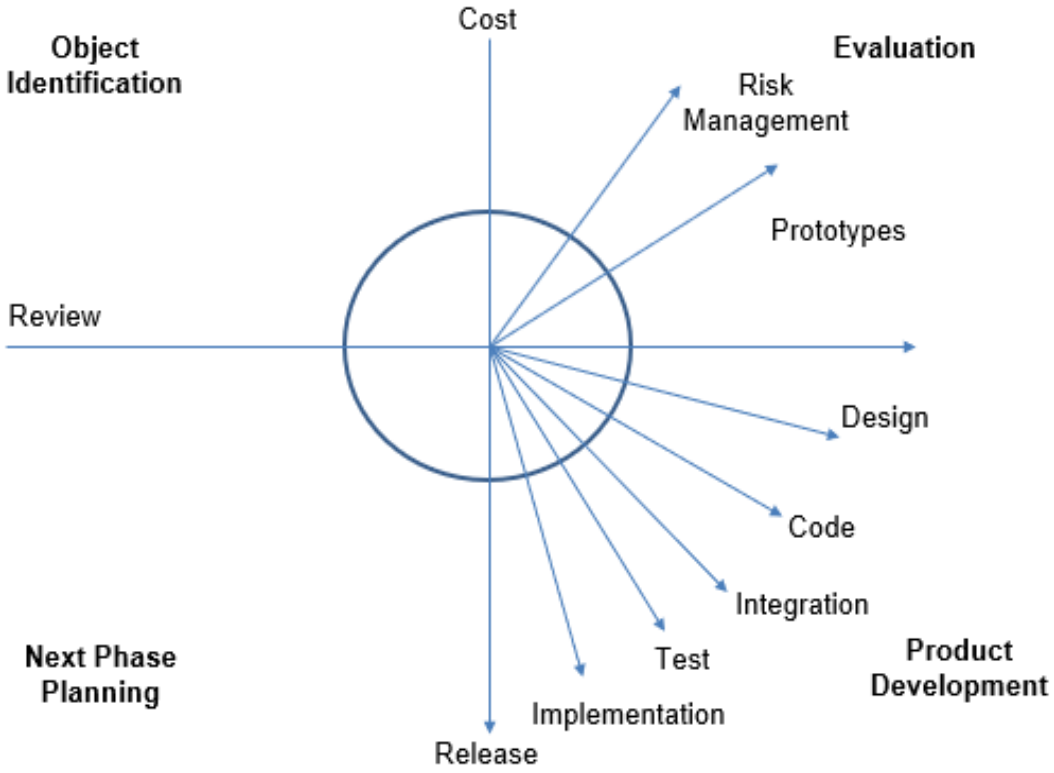
# Spiral Model

**Product Development:**

- This phase involves formulating and implementing strategies to alleviate the uncertainties and risks identified earlier.
- Activities in this stage may include benchmarking, simulations, and creating prototypes to validate the approaches.

**Project Planning:**

- The final stage involves a thorough review of the project and deciding whether to proceed to the next cycle of the spiral. If the decision is to continue, detailed plans for the next phase of the project are formulated and set in motion.



# Enterprise Resource Planning (ERP) System Development

- An organization needs a customized ERP system to integrate various business processes like finance, HR, supply chain, and customer relationship management.

- Given the complexity and the high level of integration required, the project involves significant risks, including integration challenges, compliance issues, and user acceptance.

# Application of Spiral Model

**Initial Planning and Objective Setting**
The first spiral begins with identifying the core objectives of the ERP system, such as improving process efficiency, ensuring data consistency, and integrating different business functions.

**Risk Assessment and Reduction**
Early identification of risks such as potential integration issues with existing systems, data migration challenges, or user resistance.

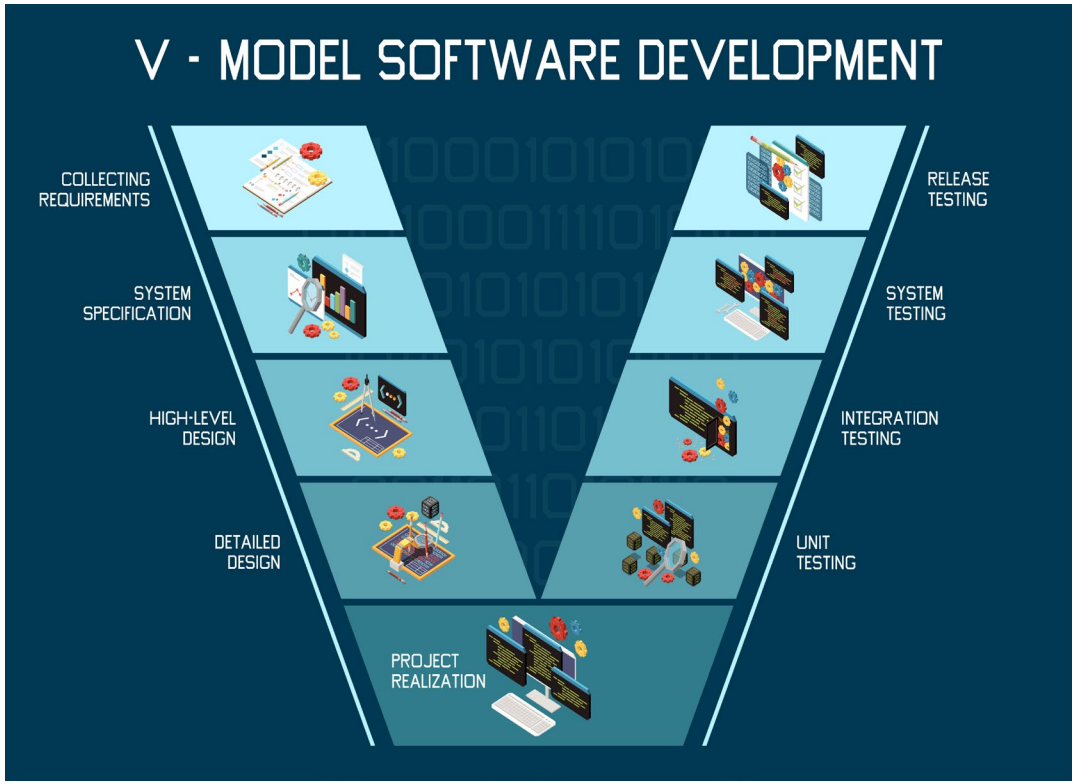
**Planning for Next Phase**
Reviewing the results of the first iteration: What worked, what didn't, and what risks are now apparent and then planning the next iteration.

**Development and Validation**
Developing a prototype or a module of the ERP system, focusing on high-risk areas. Validating this development with stakeholders, including testing for functionality and user feedback.

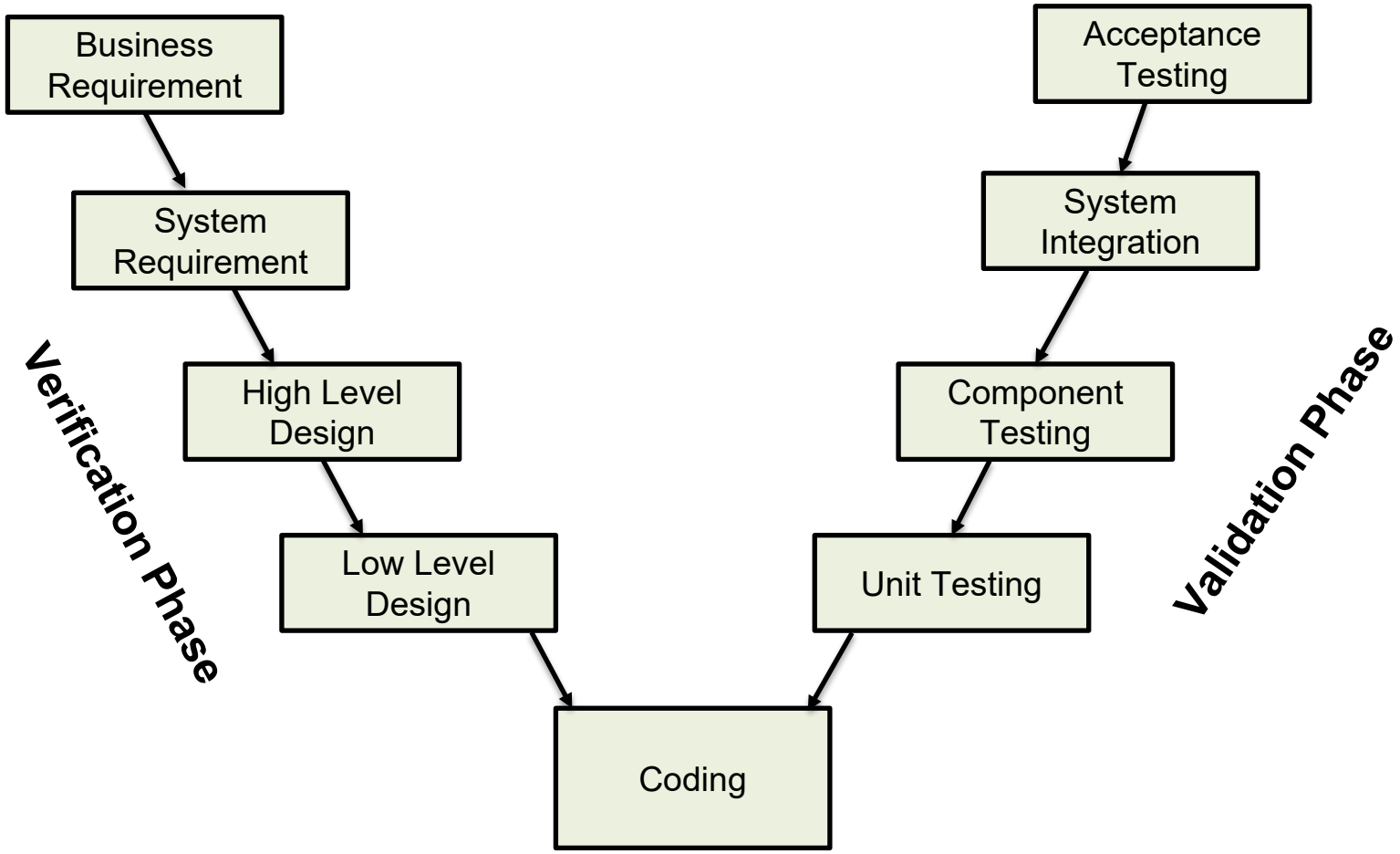
# V-Model

# V-Model

- The V-Model, also known as the Verification and Validation model, is a software development process that is an extension of the Waterfall Model.
- It is characterized by its linear flow and sequential steps, but with a strong emphasis on corresponding testing phases for each development stage.
- The "V" shape of the model demonstrates the relationships between each phase of development and its associated phase of testing.



# V-Model



# V-Model

**Verification:**

- This step encompasses static analysis methods, such as reviews, that are conducted without executing any code.
- It is a process aimed at evaluating the product's development process to ensure that it aligns with the specified requirements.

**Validation:**

- This involves dynamic analysis methods, which include both functional and non-functional testing, and are performed by executing code.
- Validation is the process undertaken after the development phase is complete, to assess if the software fulfills customer expectations and requirements.

# V-Model

**Requirements Analysis:**

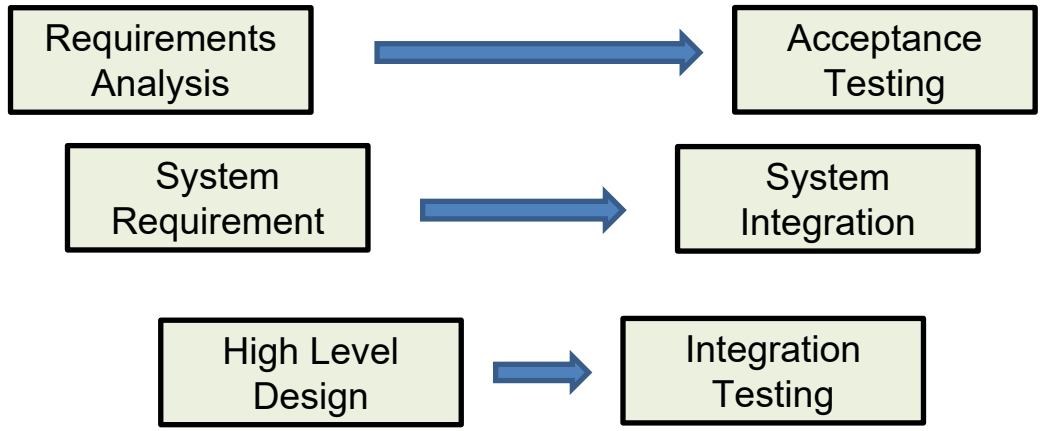
- Understanding and documenting the software requirements.
- Corresponding Test Phase: Acceptance Testing

**System Design:**

- Outlining the overall system architecture.
- Corresponding Test Phase: System Testing Preparation.

**High-Level Design:**

- Defining high-level software architectures and identifying major components.
- Corresponding Test Phase: Integration Testing Preparation.



# V-Model

**Low-Level Design/Module Design:**

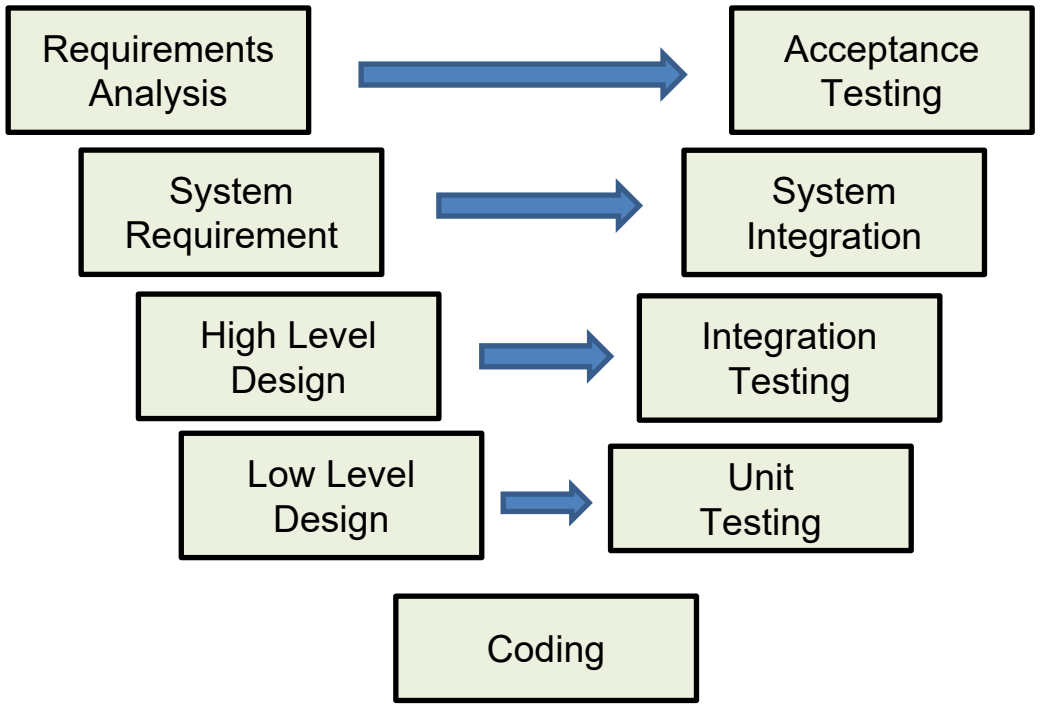
- Detailed design of each component.
- Corresponding Test Phase: Unit Testing Preparation.

**Implementation/Coding:**

- Actual coding of the software components.
- Corresponding Test Phase: Unit Testing.

**Unit Testing:**

- Testing individual units or components of the software.
- Corresponding Design Phase: Low-Level Design.



# V-Model

**Integration Testing:**

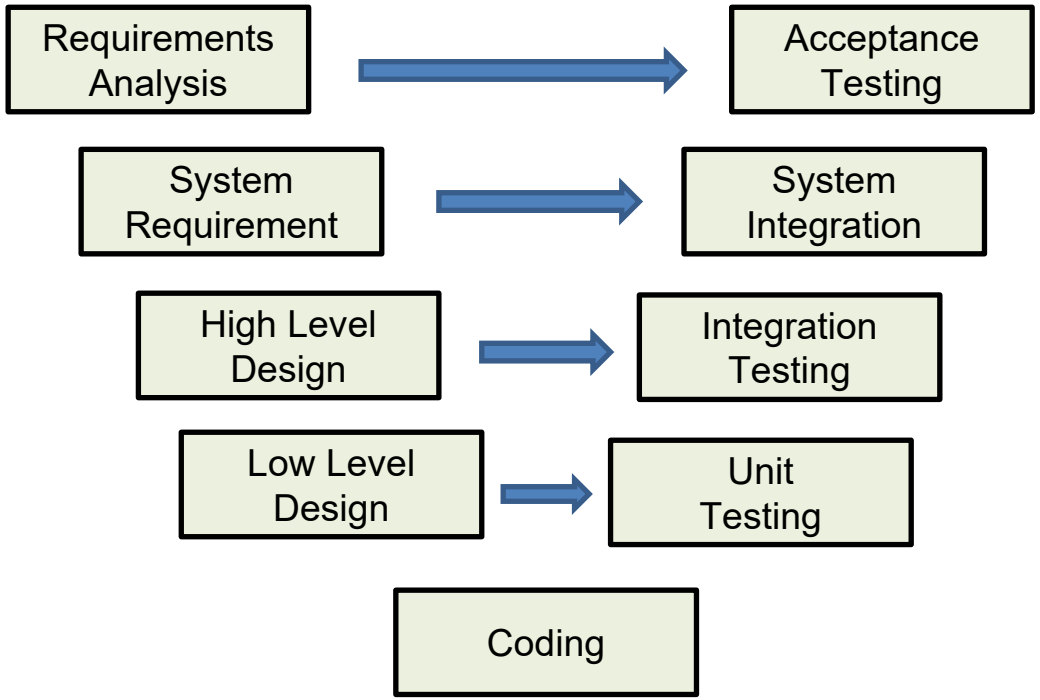
- Testing the integration of different software units.
- Corresponding Design Phase: High-Level Design.

**System Testing:**

- Testing the complete integrated system.
- Corresponding Design Phase: System Design.

**Acceptance Testing:**

- Conducted to ensure that the system meets the requirements and is ready for operational use.
- Corresponding Design Phase: Requirements Analysis.



# Banking Software System for Account Management

- A financial institution requires a new software system to manage customer accounts, handle transactions, and ensure compliance with banking regulations.

- The requirements are clear, well-defined, and unlikely to change significantly during the development process.

# Application of V-Model

**Requirements Analysis**
Detailed gathering of requirements such as account handling, transaction processing, security measures, and compliance needs.

**Corresponding Test Phase**
Preparation of acceptance testing criteria to ensure all requirements are met.

**System Design**
Outlining the overall architecture, including database design, user interfaces, and integration with existing banking systems.

**Corresponding Test Phase**
System testing plans are developed to ensure the architecture is sound and functional.

# Application of V-Model

**Coding**
Actual coding of the software based on the design documents.

**High-Level Design**
Designing major software components and modules.

**Corresponding Test Phase**
Integration testing strategies are devised to ensure these components work together seamlessly.

**Low-Level Design**
Detailed design of individual components, including specific algorithms and database tables.

**Corresponding Test Phase**
Unit testing plans are created for each component.

# Incremental Model

# Incremental Model

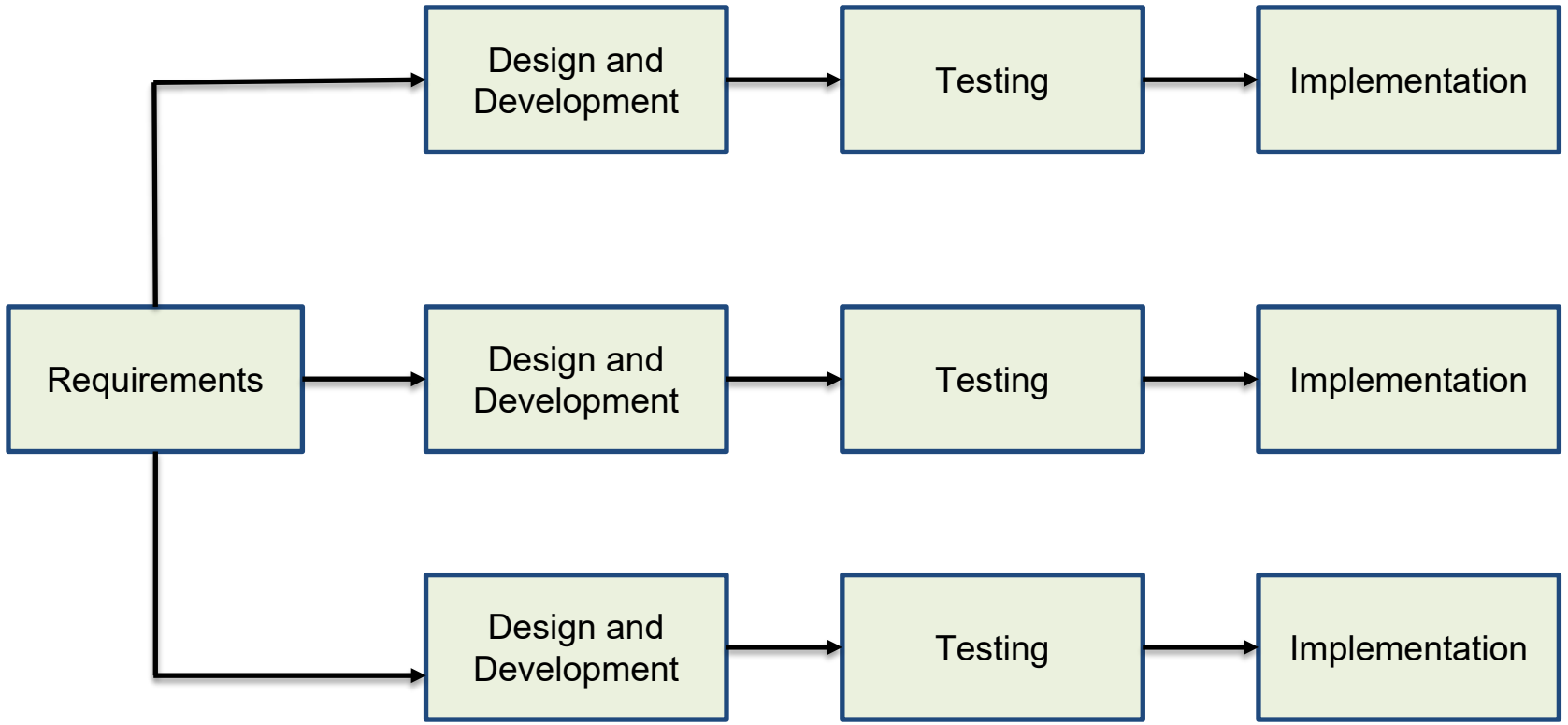
- The Incremental Model is a method of software development where the product is designed, implemented, and tested incrementally (a little more is added each time) until the product is finished.

- It involves both development and maintenance.

- The product is defined as finished when it satisfies all of its requirements.

- This model combines elements of the waterfall model and the iterative philosophy of prototyping.

# Incremental Model



# Incremental Model

**Planning and Requirements Analysis:**

- Initial planning, understanding, and documenting the software requirements.

**Design:**

- Based on the requirements, the first increment is designed.

**Implementation and Testing:**

- The first increment is developed and tested.

**Deployment:**

- After successful testing, the increment is deployed for use.

**Review and Feedback:**

- Gathering feedback on the system from users or other stakeholders.

**Next Increment:**

- Planning and development of the next increment, incorporating feedback from the previous release.

# Development of a Mobile Health (mHealth) Application

- A healthcare technology company plans to develop a mobile app to offer health monitoring, appointment scheduling, and medical consultation services.

- The market for mHealth apps is competitive and rapidly evolving, requiring quick release cycles and responsiveness to user feedback.

# Application of Incremental Model

**Initial Increment – Core Features**
The first increment includes basic features such as user registration, profile creation, and basic health monitoring (like step count and heart rate monitoring).

**Second Increment – Enhanced Features**
Based on user feedback, the second increment adds features like appointment scheduling with healthcare providers and prescription management.

**Subsequent Increments – Advanced Services**
Further increments introduce advanced features like telemedicine consultations, integration with wearable health devices, and AI-driven health insights.

**Ongoing Development**
The app continues to evolve with additional features such as community forums, health blogs, and personalized health plans.

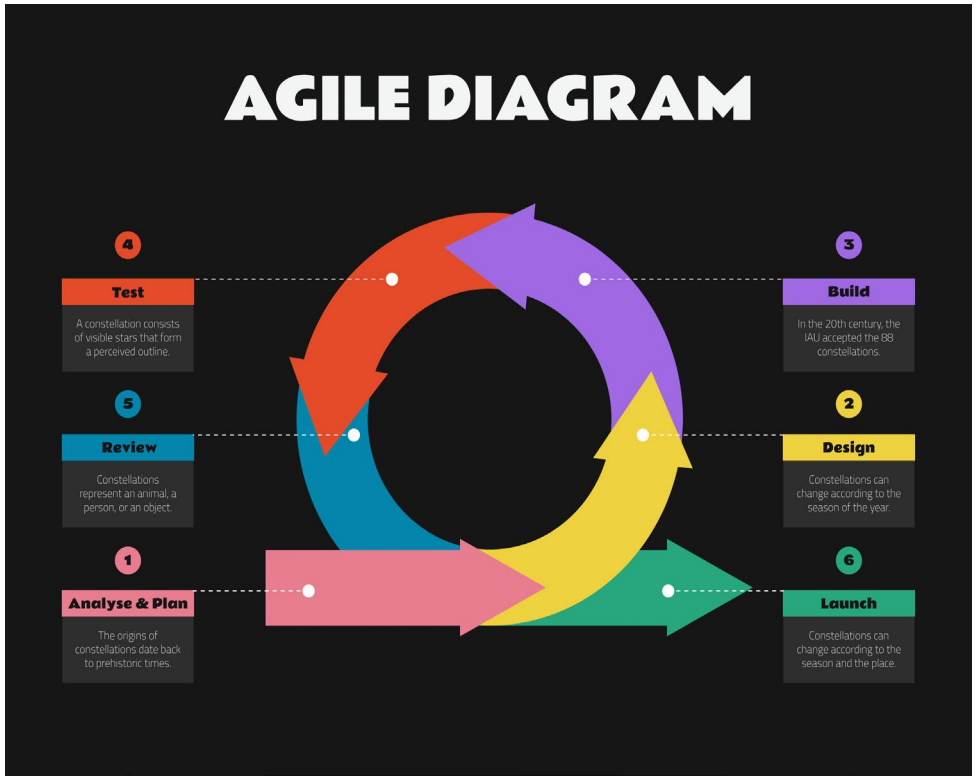
# Agile Model

# Agile Model

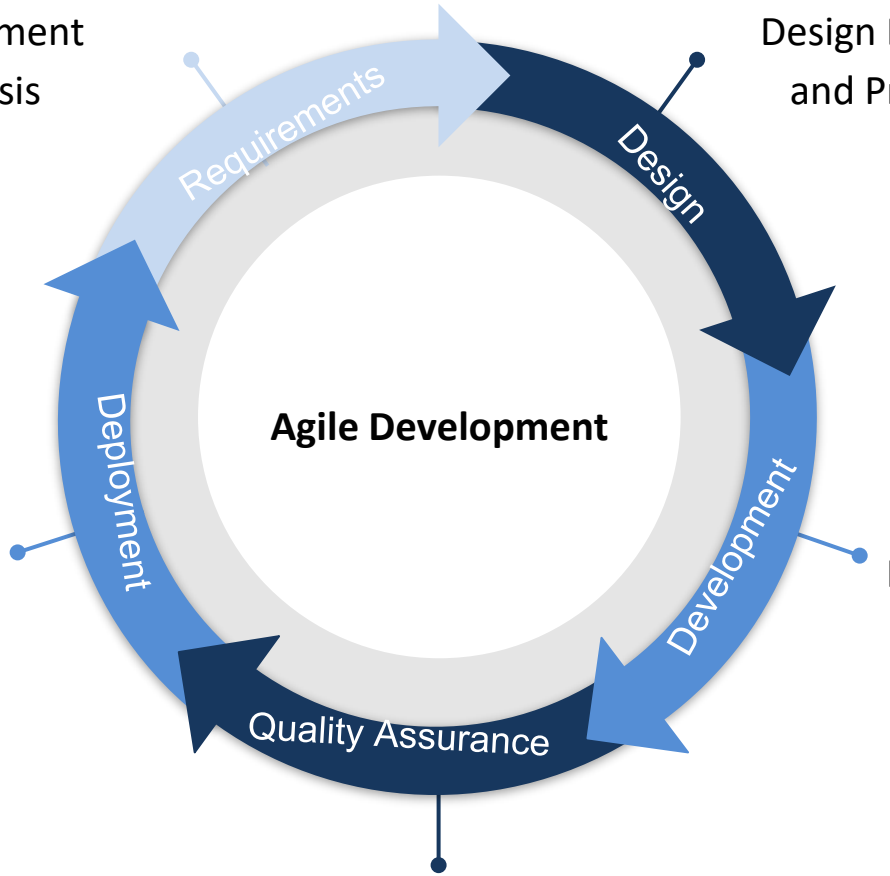
- The Agile Model is a highly iterative and incremental approach to software development, distinguished by its focus on flexibility, customer satisfaction, and rapid delivery of functional software.

- Agile methodologies break down the product into small, incremental builds and involve frequent reassessment and adaptation of plans.

- This approach is fundamentally collaborative, involving continuous input from stakeholders and constant improvement at every stage.



# Agile Model




Requirement Analysis

Design Document and Prototype

Iterations, Demo and Feedback

Iterations, Demo and Feedback

Production and Technical Support

# Agile Model

**Requirements Design:**

- Once the project is identified, collaboration with stakeholders is essential to detail the requirements.
- Tools like user flow diagrams or high-level UML diagrams are used to illustrate the functionality of new features and their integration into the existing system.

**Construction/Iteration:**

- After defining the requirements, the actual development begins.
- Designers and developers work together to create a functional product.
- This initial version is typically basic, offering minimal functionality, and is intended for further refinement.

# Agile Model

**Testing:**

The Quality Assurance (QA) team evaluates the product during this phase, focusing on performance and identifying any bugs or issues.

**Deployment:**

In this stage, the developed product is released into the user's operational environment.

**Feedback Collection:**

- Post-deployment, the final step involves gathering feedback about the product.
- This feedback is crucial for identifying areas for improvement and planning future iterations of the product.

# Development of a Food Delivery Mobile Application

- A company aims to develop a mobile application for food delivery, offering features like restaurant browsing, menu selection, order placement, and real-time delivery tracking.

- The market for food delivery apps is fast-paced and customer preferences can change quickly.

- Additionally, the app needs to integrate with various restaurants' systems and handle a high volume of user data.

# Application of Agile Model

**Initial Development - Core Features**
The first development cycle focuses on creating basic but essential features such as user registration, restaurant listing, and a basic ordering system.

**Enhancing Features**
New features like advanced search filters, favorite restaurant saving, and personalized recommendations.

**Continuous Testing and Feedback Integration**
Each iteration includes thorough testing of new features. Feedback from users is continuously gathered and analyzed.

**Adapting to Market Changes**
The development team remains agile, ready to adapt to new market trends, such as integrating new payment options or offering contactless delivery options during health crises.

**Regular Deployment and Updates**
The app is regularly updated with improvements, bug fixes, and new features, keeping it competitive and aligned with user expectations.

# Difference Between Software Models

# Difference Between Software Models

| Waterfall Model | Agile Model |
| --- | --- |
| **Best for:** Projects with well-defined, stable requirements where changes are unlikely during development.<br>**Why:** It is a linear approach simple and easy to understand, making it suitable for smaller projects or projects with a clear end goal. | **Best for:** Projects that require flexibility and have evolving requirements.<br>**Why:** Agile allows for frequent reassessment and adaptation, making it ideal for projects where the end-user's needs are expected to change over time. |

# Difference Between Software Models

| V-Model | Incremental Model |
| --- | --- |
| **Best for:** Projects where quality assurance and testing are paramount.<br>**Why:** Its rigorous testing phases for each development stage ensure a high-quality product, making it suitable for projects where errors can have serious consequences. | **Best for:** Large projects and those requiring functionality to be delivered in stages.<br>**Why:** It allows portions of the project to be delivered and tested incrementally, which can provide early partial working solutions to stakeholders. |

# Difference Between Software Models

| Spiral Model | RAD (Rapid Application Development) |
| --- | --- |
| **Best for:** High-risk projects and those requiring constant refinement.<br>**Why:** It places a strong emphasis on risk analysis and allows for extensive refinement at each stage, making it suitable for complex projects. | **Best for:** Projects that need to be developed in a short timeframe.<br>**Why:** Its focus on rapid prototyping is ideal for getting a functional product to market quickly. |

# Summary

Here's a brief recap:

- Software engineering involves a systematic approach to developing software, from understanding requirements to designing, building, and testing the software.
- Software engineering encompasses several key aspects that collectively ensure the effective and efficient development, maintenance, and evolution of software systems and its importance.
- The Software Development Lifecycle (SDLC) is a process used in the software industry to design, develop, test, and deliver high-quality software.
- The SDLC is fundamental for guiding the development process in a way that maximizes quality, minimizes risks, and ensures alignment with customer and business needs.
- Different software models has its strengths and is chosen based on project requirements, team dynamics, customer needs, and various other factors.