Unit-2

Process Models

Software Process

- Software Process: is the set of activities and associated results that produce a software product. There are four process activities:
  - Software specification where customer and engineer define the software to be produced and its constraints.
  - Software development where the software is designed And produced.
  - Software validation where the software is tested to ensure it meets customer requirements.
  - Software evaluation where software is modified to adapt to change in customer and market requirements.

Software Process

- Software Process can be categorized to:
  - Common process framework which are a small number of framework activities that are applicable to all software projects
  - Software Engineering Umbrella Activities which are used for quality assurance, configuration management and measurements.

Software Process

- Common process framework
  - Communication(customer collaboration and requirement gathering)
  - Planning(establishes engineering work plan, describes technical risks, list resources requirements and define work schedule)
  - Modeling(creation of model to help developers and customers understand the requirements and software design)
  - Construction(code generation and testing)
  - Deployment(software delivered for customer feedback and evaluation)

Software Process

- Software Engineering Umbrella Activities:
  - Software project tracking and control(allows team to assess progress and take corrective action)
  - Risk management(assess risks that may affect project outcome and quality)
  - Software Quality Assurance(activities required to maintain software quality)
  - Formal Technical Reviews(assess engineering work product to uncover and remove errors before they propagate to next activities.
  - Measurement(define and collect process, project and product measures)
  - Software Configuration management(manage effect of change)
  - Reusability management(define criteria for work product reuse and establish a mechanism to achieve components reuse)
  - Work product preparation and production(activities to create models, documents, logs, forms, lists, etc.)

Software Process

- Software Process characteristics
  1. Understandability
  2. Visibility
  3. Robustness
  4. Reliability
  5. Acceptability
  6. Maintainability
  7. Rapidity
  8. supportability

Software Process Model

Is a simplified description of software process that is presented from a particular perspective. Some Examples are:

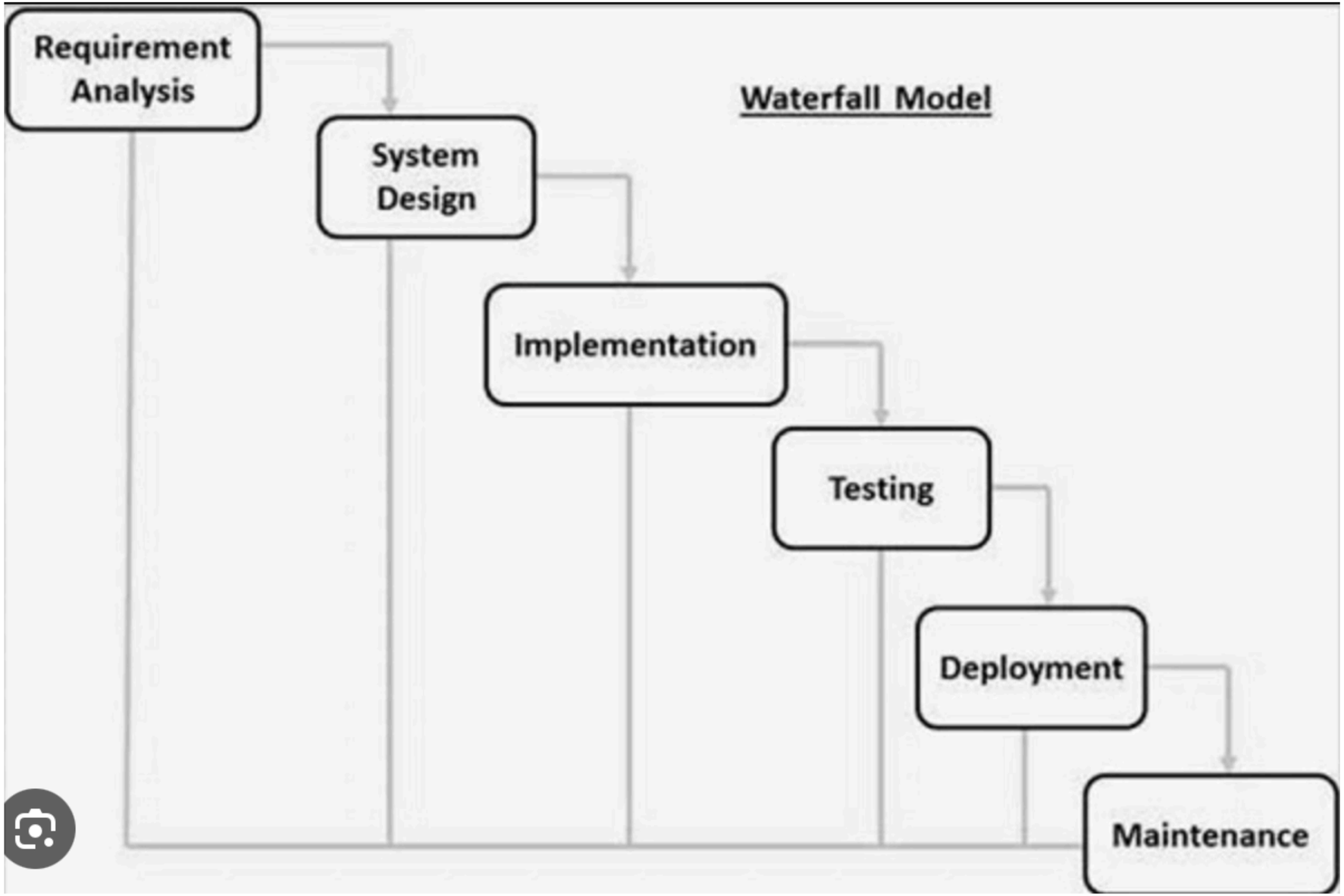
- A workflow Model: which show the sequence of activities in the process along with their inputs, outputs and dependencies, the activities represent human actions
- A dataflow or activity model: which represent the process as a set of activities each carries some data transformation.
- A role/action model: this represent peoples involve in software process and activities for which they are responsible.

Software Process Model

Most software process model are based on one of three general models:

1. The waterfall approach: represent activities as separate phases.
2. Iterative development: interleave the activities.
3. Component-based software engineering: assume part of the system exist, it focus on integration these parts.

Waterfall Model



- The Waterfall Model is a sequential software development process.
- It proceeds through defined phases, each building upon the previous one.
- The Waterfall Model offers a structured approach to software development but may not be suitable for all projects.

Phases of the Waterfall Model

- **Requirement Analysis:** Gathering and documenting requirements from stakeholders.
- **System Design:** Creating a blueprint of the system architecture.
- **Implementation:** Coding and building the system based on design specifications.
- **Testing:** Verifying that the system meets requirements and functions correctly.
- **Deployment:** Releasing the system to users or clients.
- **Maintenance:** Providing support, updates, and enhancements as needed.

Key Characteristics

- **Sequential:** Each phase follows the previous one in a linear manner.
- **Document-Driven:** Emphasis on comprehensive documentation at each stage.
- **Rigid:** Changes are difficult and costly to implement once a phase is complete.
- **Suitable for Stable Requirements:** Best suited for projects with well-defined and stable requirements.

Advantages

- **Clear Milestones:** Well-defined phases with distinct deliverables.
- **Structured Approach:** Provides a clear roadmap for development.
- **Ease of Management:** Easy to manage and track progress.
- **Early Detection of Issues:** Issues are typically detected early in the process.

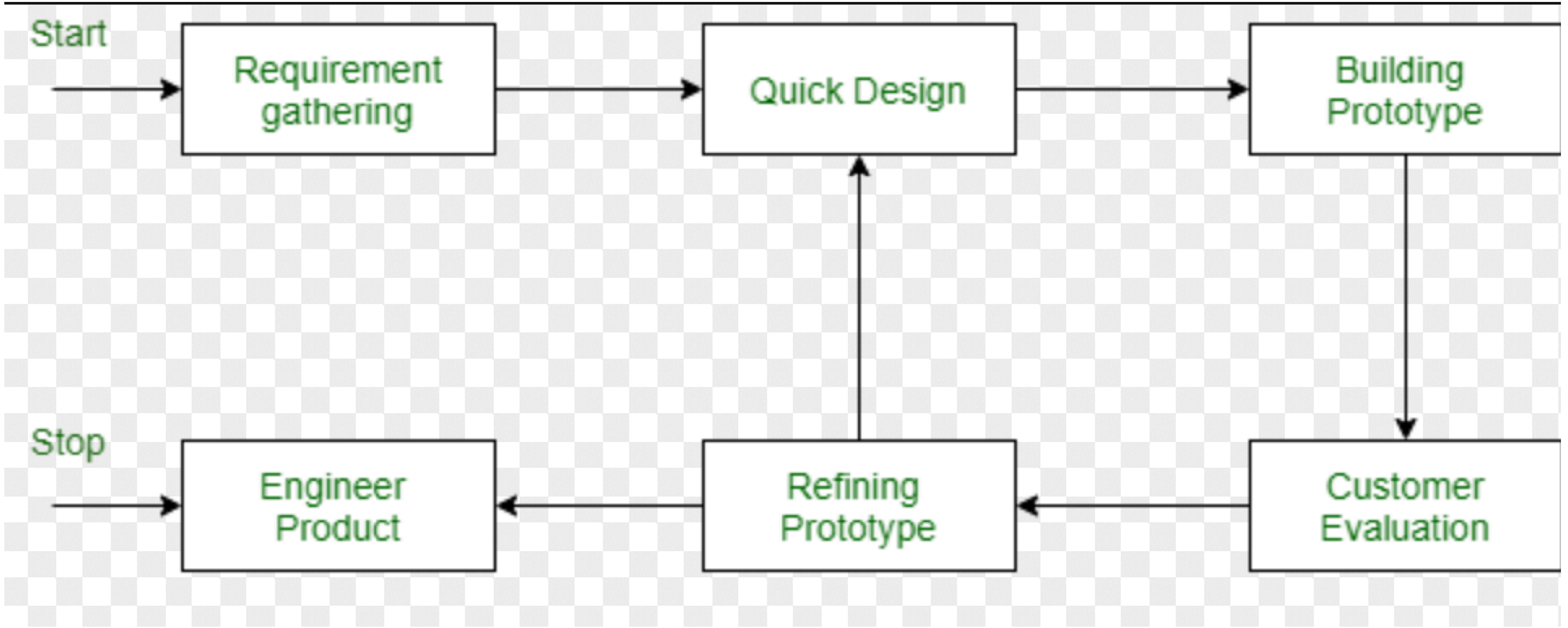
Disadvantages

- **Inflexible:** Limited flexibility for accommodating changes.
- **Late Feedback:** Stakeholders may not see the product until late in the process.
- **High Risk:** High risk of customer dissatisfaction if requirements change.
- **Not Ideal for Complex Projects:** May not be suitable for projects with evolving or unclear requirements.

Examples of Waterfall Model Use

- Traditional engineering projects such as construction.
- Projects with fixed requirements and a clear understanding of user needs.

Prototyping model



- The Prototyping Model is an iterative software development approach.
- It focuses on creating a preliminary version of the system to gather feedback and refine requirements.
- The Prototyping Model offers a flexible and iterative approach to software development, focusing on user feedback and rapid iterations.

Phases of the Prototyping Model

- **Requirements Gathering:** Initial requirements are collected from stakeholders.
- **Quick Design:** A basic prototype is developed to demonstrate key functionalities.
- **Prototyping:** Iterative development of prototypes based on feedback.
- **Feedback and Refinement:** Stakeholders review prototypes and provide feedback for further refinement.
- **Final Implementation:** Once requirements are well-understood, the final system is developed.

Key Characteristics

- **Iterative:** Development occurs in cycles, with prototypes refined based on feedback.
- **User-Centric:** Focus on user involvement and feedback throughout the process.
- **Rapid:** Allows for quick iterations and adjustments based on user input.
- **Flexible:** Can accommodate changes and evolving requirements more easily than the Waterfall Model.

Advantages

- **Early Feedback:** Stakeholders can provide feedback early in the development process.
- **Improved Understanding:** Helps in better understanding and refining requirements.
- **Reduced Risk:** Identifies potential issues early, reducing project risk.
- **Increased Stakeholder Satisfaction:** Users have more input, leading to a system better aligned with their needs.

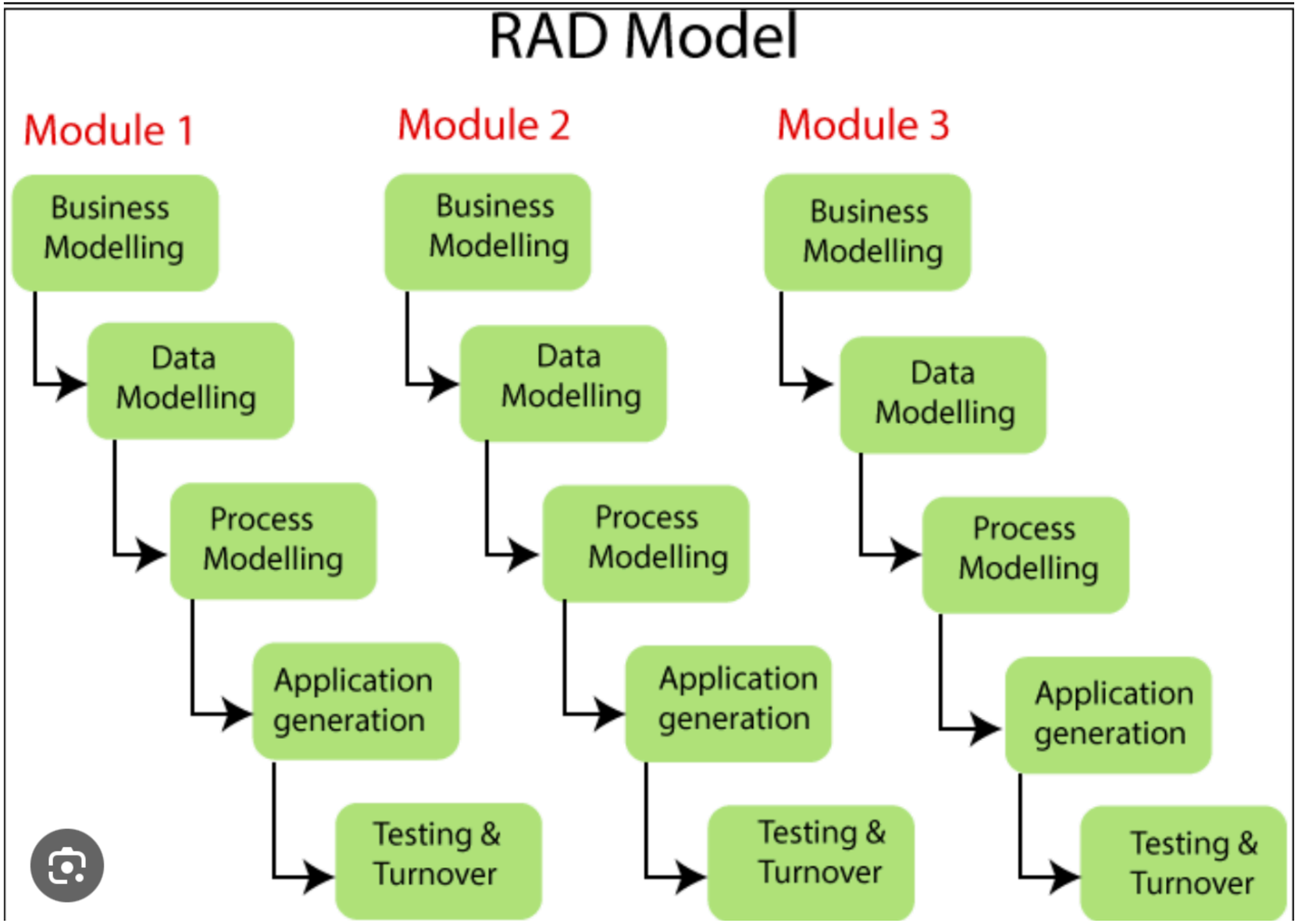
Disadvantages

- **Scope Creep:** Without proper control, the project can expand beyond the original scope.
- **Potential for Miscommunication:** Misunderstandings between stakeholders and developers may arise.
- **Resource Intensive:** Requires more involvement from stakeholders and development resources.
- **Difficulty in Finalizing Requirements:** Constant changes can make it challenging to finalize requirements.

Examples of Prototyping Model Use

- Software projects with evolving or unclear requirements.
- Projects where user involvement and feedback are critical.

RAD Model


RAD Model
Module 1
Business Modelling
Data Modelling
Process Modelling
Application generation
Testing & Turnover
Module 2
Business Modelling
Data Modelling
Process Modelling
Application generation
Testing & Turnover
Module 3
Business Modelling
Data Modelling
Process Modelling
Application generation
Testing & Turnover

- The RAD Model is an iterative software development process that prioritizes rapid prototyping and quick feedback.
- It emphasizes iterative development and the creation of prototypes over strict planning and sequential design.
- The RAD Model offers a rapid and iterative approach to software development, prioritizing quick feedback and rapid prototyping.

Phases of the RAD Model

Business Modeling:

- Identification of business requirements and objectives.
- Analysis of business processes and workflows.
- Understanding the needs and goals of end-users.

Data Modeling:

- Designing the structure and organization of the data required for the system.
- Defining data entities, relationships, and attributes.
- Creating data models such as entity-relationship diagrams (ERDs) or database schemas.

Phases of the RAD Model

Process Modeling:

- Modeling the flow of processes and interactions within the system.
- Defining workflows and sequences of activities.
- Identifying bottlenecks and areas for optimization.

Application Generation:

- Rapid development of prototypes or working models based on the requirements and models created in previous phases.
- Using tools and frameworks to quickly build user interfaces and functionality.
- Iterative development and refinement of application components.

Phases of the RAD Model

Testing:

- Conducting thorough testing of the developed application to ensure functionality, usability, and reliability.
- Identifying and resolving defects or issues through iterative testing cycles.
- Involving end-users in user acceptance testing (UAT) to validate the application against their needs and expectations.

Turnover:

- Deployment of the developed application to production environments.
- Providing user training and documentation to support the adoption of the new system.
- Handing over the system to operations and support teams for ongoing maintenance and management.

Key Characteristics

- **Iterative and Incremental:** Development occurs in cycles, with each iteration building upon the previous one.
- **User Involvement:** End-users play a crucial role throughout the development process, providing feedback and validation.
- **Rapid Prototyping:** Focus on quickly developing prototypes to gather feedback and refine requirements.
- **Flexible and Adaptive:** Can accommodate changes and evolving requirements more easily than traditional models.

Advantages

- **Quick Development:** Rapid prototyping allows for fast development and deployment.
- **Customer Satisfaction:** Continuous user involvement leads to a product that better meets user needs.
- **Reduced Development Time:** Parallel development of components shortens the overall development lifecycle.
- **Flexibility:** Can adapt to changing requirements and market conditions more effectively.

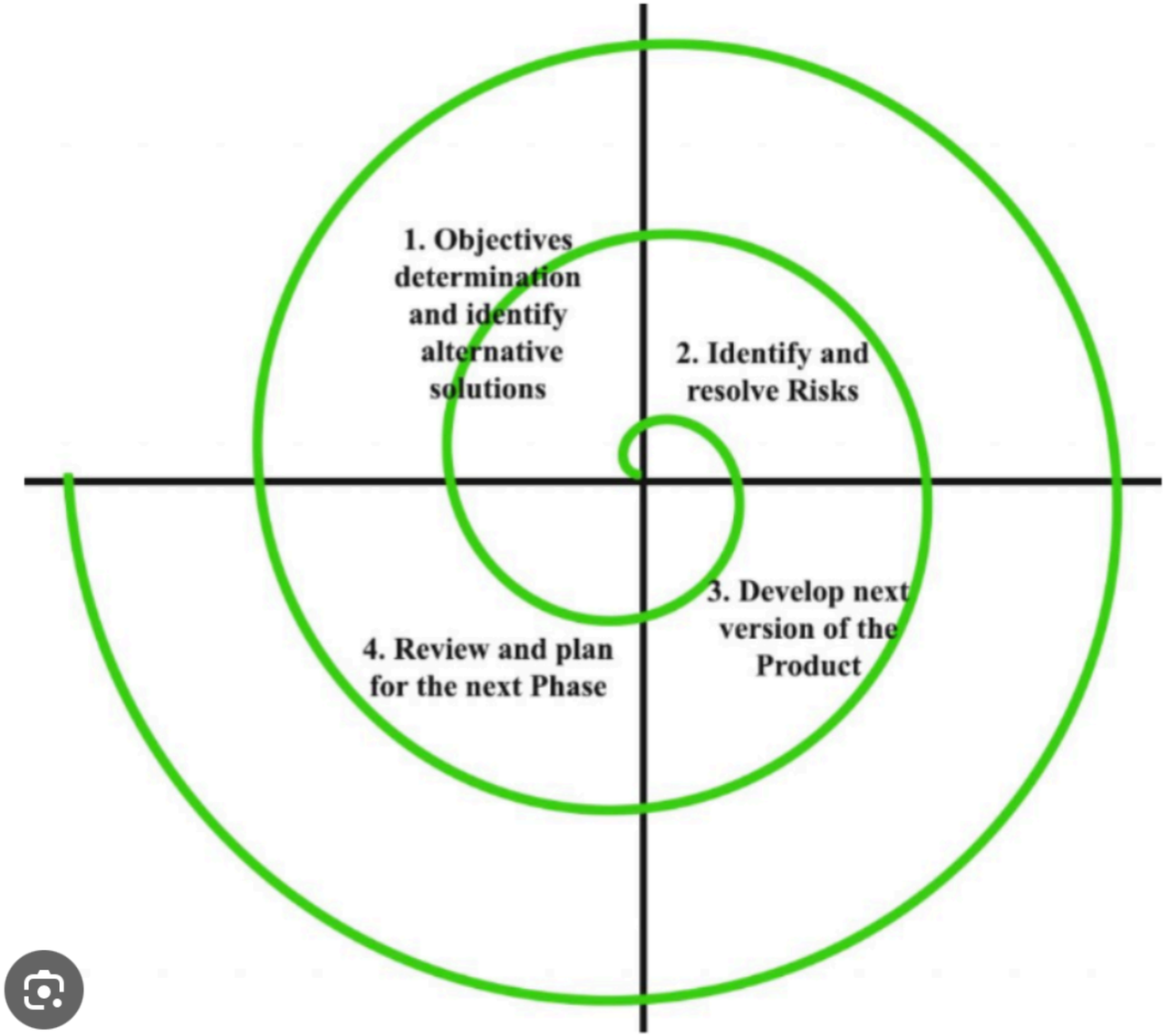
Disadvantages

- **Complexity:** Rapid development cycles can lead to increased complexity.
- **Resource Intensive:** Requires significant involvement from stakeholders and development resources.
- **Dependency on User Involvement:** Success relies heavily on active participation and feedback from end-users.
- **Potential for Scope Creep:** Without proper control, the project scope can expand beyond the original intent.

Examples of RAD Model Use

- Projects with tight deadlines and changing requirements.
- Applications where user involvement and quick iterations are crucial.

Spiral Model



- The Spiral Model is a risk-driven software development process model.
- It combines iterative development with elements of the Waterfall Model and prototyping.
- Developed by Barry Boehm in 1986, it emphasizes risk analysis, iteration, and flexibility.

Phases of the Spiral Model

- **Planning:** Identify objectives, constraints, and alternatives. Evaluate risks.
- **Risk Analysis:** Analyze and mitigate potential risks. Develop strategies to manage identified risks.
- **Engineering:** Develop, verify, and validate the product incrementally. Plan for next iterations.
- **Evaluation:** Review the results of the iteration. Decide whether to proceed to the next iteration or iterate further.

Key Characteristics

- **Iterative and Incremental:** Development occurs in iterations, with each iteration producing a deliverable.
- **Risk Management:** Focus on identifying and mitigating risks throughout the development process.
- **Flexibility:** Allows for changes and adjustments based on feedback and evolving requirements.
- **Emphasis on Verification and Validation:** Continuous testing and evaluation ensure product quality.

Advantages

- **Risk Mitigation:** Early identification and mitigation of risks reduce project failure.
- **Flexibility:** Can accommodate changes in requirements and technology.
- **Incremental Development:** Allows for early and frequent delivery of working software.
- **High Visibility:** Stakeholders are involved throughout the process, leading to increased transparency.

Disadvantages

- **Complexity:** Managing multiple iterations and risks can be complex.
- **Resource Intensive:** Requires significant effort for risk analysis and management.
- **Not Suitable for Small Projects:** Overhead may outweigh benefits for small projects with low risks.

Examples of Spiral Model Use

- Projects with high uncertainty and evolving requirements.
- Systems where risk management is critical, such as safety-critical systems.

Agile Model

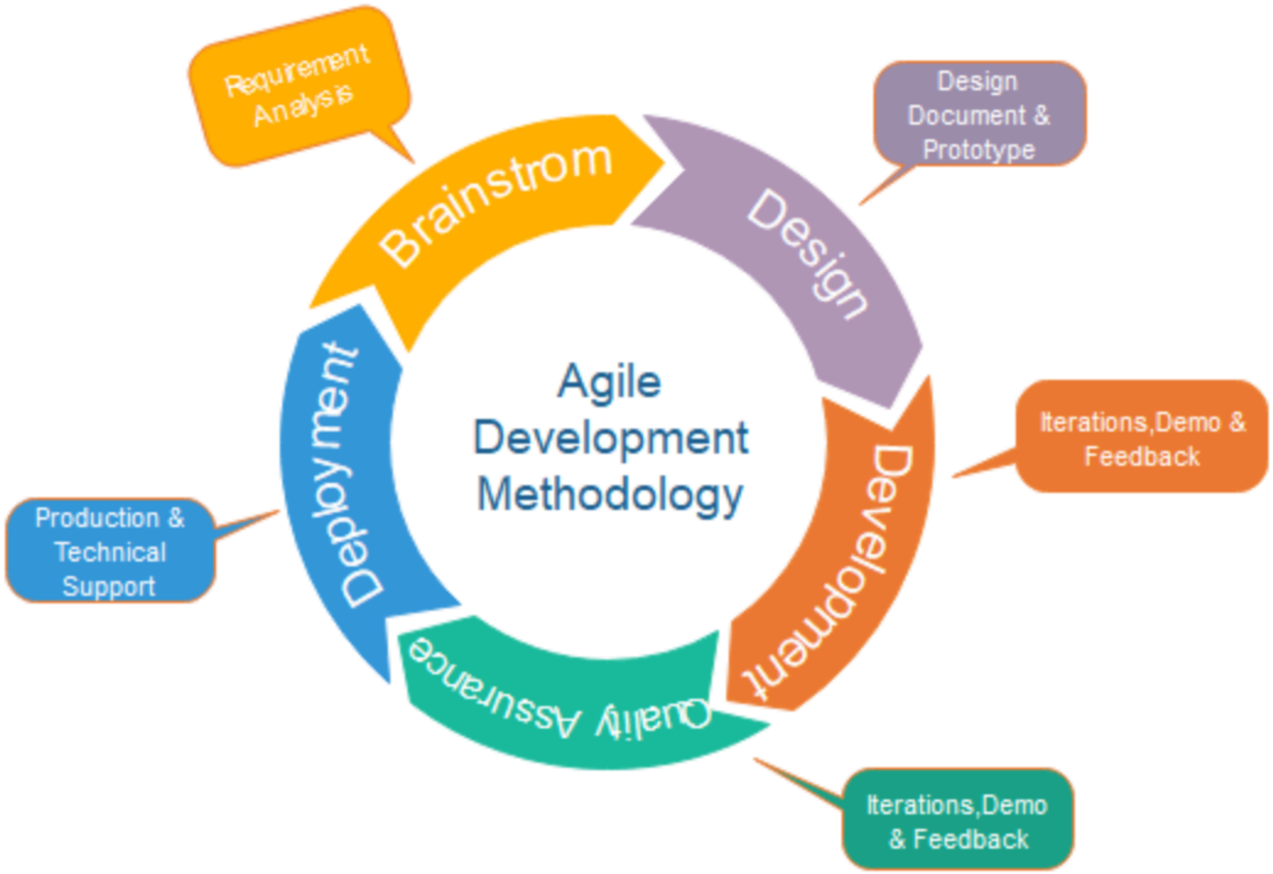


Fig. Agile Model

- The Agile Model is an iterative and incremental approach to software development.
- It emphasizes flexibility, collaboration, and customer feedback to deliver working software quickly and efficiently.
- Agile methodologies prioritize individuals and interactions over processes and tools, responding to change over following a plan.

Key Principles of Agile

- **Customer Satisfaction:** Delivering valuable software that meets customer needs and expectations.
- **Iterative Development:** Breaking down the project into small, manageable iterations or sprints.
- **Continuous Feedback:** Regularly gathering feedback from stakeholders and incorporating it into the development process.
- **Adaptability:** Embracing change and adjusting plans based on customer feedback and evolving requirements.

Core Values of Agile

- **Individuals and Interactions:** Valuing collaboration and communication among team members and stakeholders.
- **Working Software:** Prioritizing the delivery of functioning software in each iteration.
- **Customer Collaboration:** Involving customers and end-users throughout the development process.
- **Responding to Change:** Being responsive to changing requirements and priorities, even late in the development cycle.

Phases of the Agile Model

Planning:

- Identify project objectives, scope, and priorities.
- Define product backlog and high-level requirements.
- Evaluate risks and plan for iterative development.

Execution:

- Develop and deliver increments of the product.
- Conduct sprint planning and daily stand-up meetings.
- Code, test, and integrate features continuously.

Phases of the Agile Model

Review:

- Evaluate completed work against project objectives.
- Conduct sprint review meetings and gather feedback.
- Demonstrate completed features to stakeholders.

Iteration:

- Reflect on the previous iteration and plan improvements.
- Conduct sprint retrospective meetings to identify areas for improvement.
- Adapt processes and plans based on feedback and lessons learned.

Agile Methodologies

- **Scrum:** A framework for managing and organizing Agile projects into iterations called sprints, with defined roles, ceremonies, and artifacts.
- **Kanban:** A visual management tool for optimizing workflow and limiting work in progress, focusing on continuous delivery.
- **Extreme Programming (XP):** Emphasizes engineering practices such as pair programming, test-driven development (TDD), and continuous integration to ensure high-quality software.

Advantages

- **Faster Time to Market:** Delivering working software in short iterations allows for quicker delivery of value to customers.
- **Flexibility:** Adapting to changing requirements and priorities throughout the development process.
- **Increased Collaboration:** Encouraging close collaboration between development teams, stakeholders, and customers.
- **Improved Quality:** Continuous testing and feedback lead to higher-quality software.

Challenges

- **Scope Management:** Managing scope creep and ensuring that changes align with project goals.
- **Resource Allocation:** Balancing priorities and resources across multiple iterations or projects.
- **Cultural Shift:** Adopting Agile practices may require a cultural shift within organizations, including changes in mindset and practices.

Examples of Agile Model Use

- Software development projects in various domains, including web development, mobile app development, and enterprise software.
- Projects with evolving or unclear requirements, where flexibility and adaptability are crucial.