



UNIVERSITY OF RAJSHAHI

Rajshahi, BANGLADESH.

Course Code:

ICE-3151

Course Title :

**Information System Analysis and
Software Engineering**

Chapter:

Software Process Model

Prerequisite Courses	:	CSE1291, ICE2231	
Course Objectives	:	The main objectives of this course are to provide students the basic concepts of information system analysis, design and implementation. This course also provides the details about how to use the knowledge of software engineering to develop a software project.	
Course Learning Outcomes (COs)	:	Students who successfully complete the course will be able to:	CO-PO Mapping
		CO-1: Identify the key aspects of the information system development and software engineering.	PO-1
		CO-2: Analyze and design an information systems and software products.	PO-2, PO-3
		CO-3: Apply the concept of information system and software engineering to solve real-world problems.	PO-3

Course Contents (ICE-3151)

RH

Course Contents

Section A

Introduction: System Concepts, characteristics of a system, Elements of a system, Types of systems, Types of information systems, Information systems environments, System Development Life Cycle, Role of system analyst.

Information Gathering: Categories of information, Information gathering tools: Review of literature, Procedures and forms, On-site observation, Interviews and questionnaires, Types of interviews and questionnaires.

Feasibility Study: System performance definition, Feasibility study, Feasibility Considerations, Steps in feasibility analysis, Feasibility report, Oral presentation, Data analysis, Cost/benefit analysis: Cost and benefit categories, Procedure for cost/benefit determination, Classification of costs and benefits, Cost/benefit evaluation methods.

System Design and Implementation: The process and stages of system design. Input/output and forms design, File organization and database design, System Testing and Quality Assurance, Implementation and software maintenance, Hardware/software selection, Project scheduling and Software, Security, Disaster/recovery and Ethics in System Development.

Course Contents (ICE-3151)

GR

Section B

Software Process Models: Software process, Software process models, Linear sequential model, Prototyping models, Waterfall model, RAD, Incremental model, Spiral model, Agility and agile process model.

Software Quality Management: Concept of quality, Quality factors, Achieving software quality, Elements of software quality assurance, SQA Tasks, Goals, and Metrics, Formal approaches to SQA, Statistical software quality assurance, Software reliability, The ISO 9000 quality standards, The SQA plan.

Software Testing and Maintenance: Different testing philosophy and methods, Software testing fundamentals, Internal and external views of testing, White-Box testing, Black-Box testing, Model-Based testing, Patterns for software testing, Testing object-oriented applications, Testing web applications.

Software Projects and Risk Management: Project management concepts: People, Product and process. Product metrics, Process and project metrics, Estimation for software project, Risk identification, Risk projection, Risk refinement, Risk mitigation, Monitoring, and management.

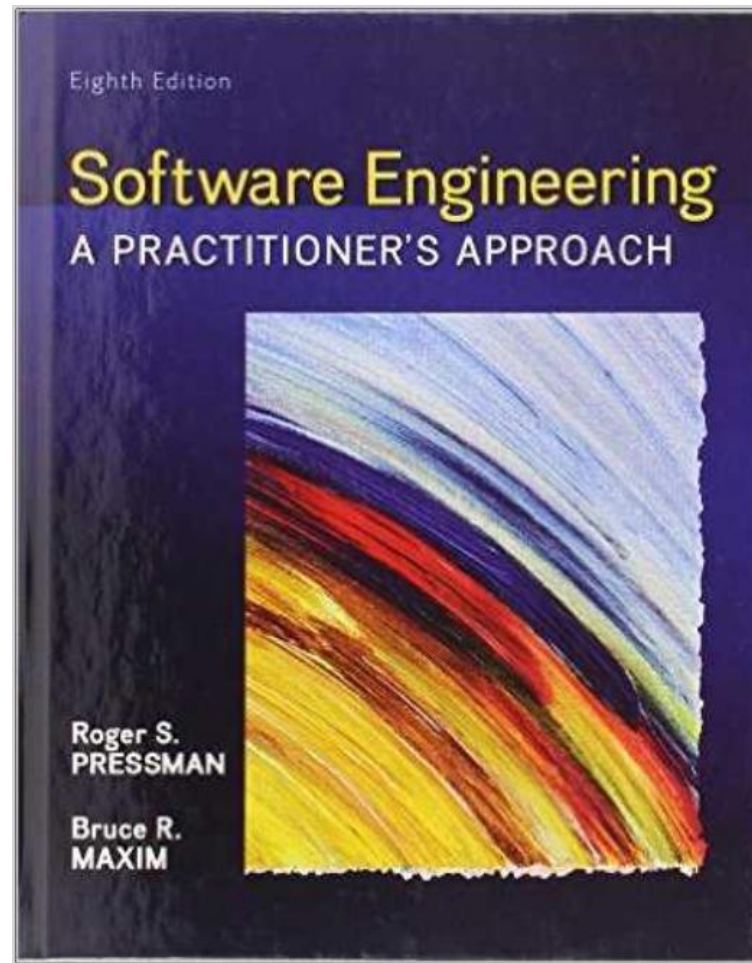
ICE-3151

Text Books:

1. R. S. Pressman : Software Engineering-A Practitioner's Approach

Reference Books:

2. K. K. Aggarwal & Yogesh Singh : Software Engineering
3. Rod Stephens : Beginning Software Engineering



Chapter:

Software Process Model

What is Process Model?

- Process models were originally proposed to bring order to the **chaos** of software development.
- A process model provides a specific roadmap for software engineering work.
- It defines the flow of all
 - activities,
 - actions and tasks,
 - the degree of iteration,
 - the work products, and
 - the organization of the work that must be done.

Who does it?

- Software engineers and their managers adapt a process model to their needs and then follow it.
- In addition, the people who have requested the software have a role to play in the process of defining, building, and testing it.

Software Process

- A set of activities and tasks that ensure software development is systematic, well-organized, and meets user requirements.
- Steps in a Software Process:
 1. **Requirements Analysis:** Understanding what the customer needs.
 2. **Design:** Creating architecture and design plans.
 3. **Implementation:** Writing and coding the software.
 4. **Testing:** Verifying the software's functionality.
 5. **Deployment:** Releasing the software to users.
 6. **Maintenance:** Fixing bugs and updating software.

Why is a Software process important?

- ✓ It ensures systematic development, reduces risks, maintains quality, and meets user needs efficiently.

What are the characteristics of a good software process?

- A good software process has the following characteristics:
 - ✓ Clearly defined objectives.
 - ✓ Adaptability to changing requirements.
 - ✓ Focus on quality assurance.
 - ✓ Efficient resource utilization.
 - ✓ Well-documented activities and results.

Software Process Model?

- A software process model is **an abstraction of the actual process**, which is being described.
- It can also be defined **as a simplified representation of a software process**.
- Each model represents a process from a specific perspective.

Need for Process Model

- The software development team **must decide the process model** that is to be used for software product development and then the entire team must adhere to it.
- This is necessary because the software product development can then be done systematically.
- Each team member will understand **what is the next activity** and **how to do it**.
- Thus, process model will **bring the definiteness and discipline** in overall development process.

Need for Process Model

- Every process model consists of **definite entry and exit criteria** for each phase.
- Hence the transition of the product through various phases is **definite**.
- If the process model is not followed for software development then any team member can perform any software development activity, this will ultimately cause **a chaos** and software project will definitely fail.
- Without using process model, it is **difficult to monitor the progress of software product**.
- Thus process model plays an important role in software engineering.

Waterfall Model

- It is a **classical software development methodology**.
- It was first introduced by **Winston W. Royce** in 1970.
- It is a **linear and sequential approach** to software development that **consists of several phases**.
- It must be completed in **a specific order**.
- This classical waterfall model is **simple and idealistic**.

Waterfall Model (Con't)

- It was once very popular. Today, it is not that popularly used. However, it is important because most other types of software development life cycle models are a **derivative of this**.
- It is characterized by a **structured, sequential approach** to project management and software development.
- The waterfall model is useful in situations where the **project requirements** are **well-defined** and **the project goals** are **clear**.

Features of Waterfall Model

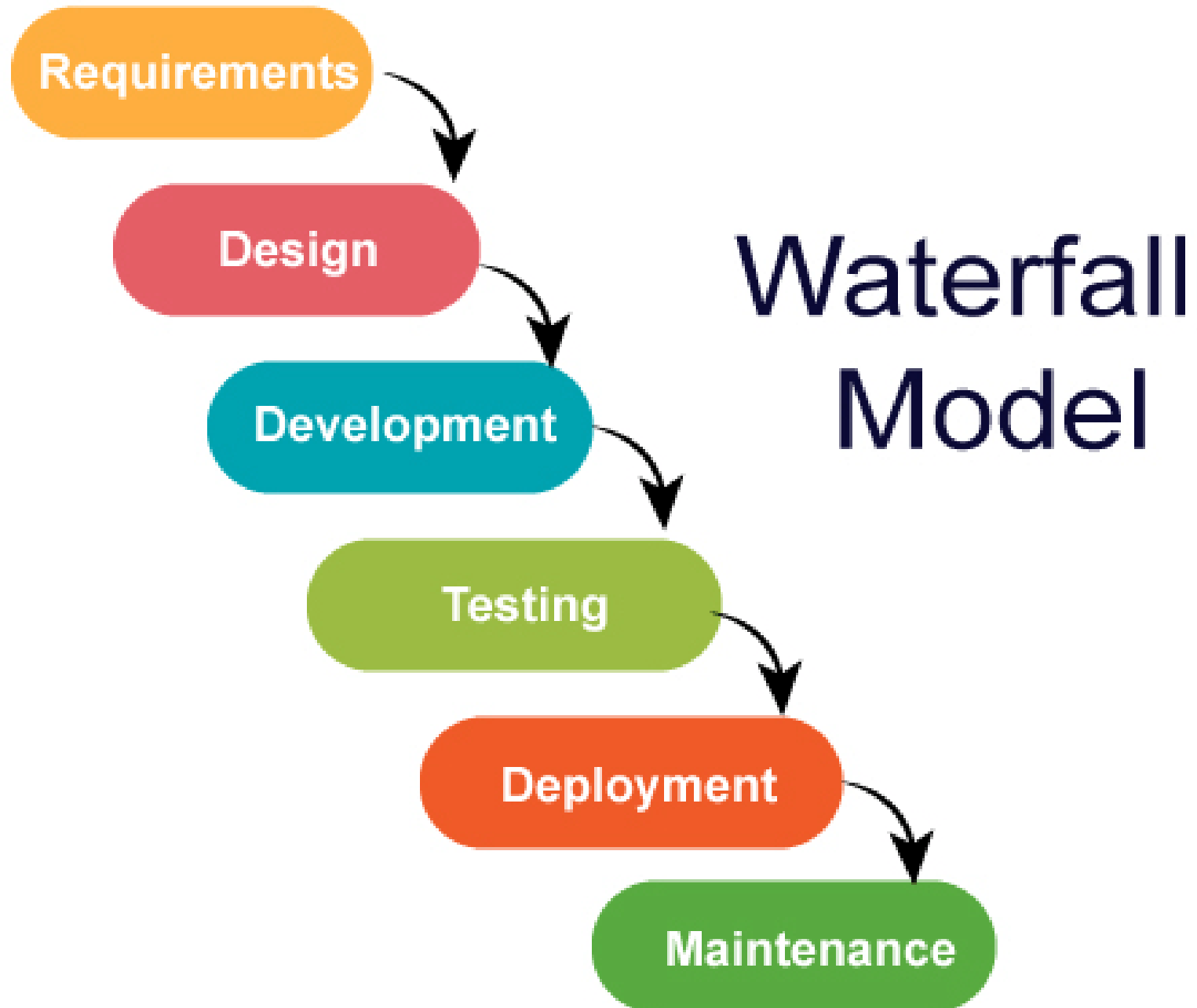
- 1. Sequential Approach:** Each phase of the project is completed before moving on to the next one.
- 2. Document-Driven:** The waterfall model depended on documentation to ensure that the project is well-defined and the project team is working towards a clear set of goals.
- 3. Quality Control:** The 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.

Features of Waterfall Model

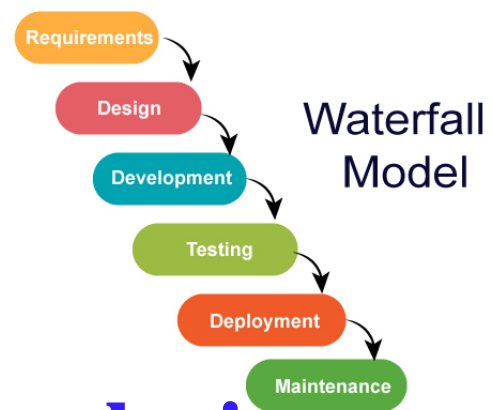
3. **Rigid:** Changes are difficult to implement once a phase is completed.

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.

Phases of Waterfall Model



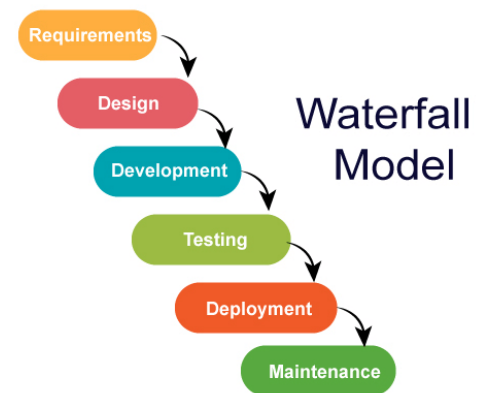
Phases of Waterfall Model



1. Requirements Gathering and Analysis:

- ✓ In this phase, **all the requirements** for the software are **collected** from **stakeholders** (clients, users, etc.).
- ✓ The goal is to understand what the software should do.
- ✓ **Output:** A Software Requirements Specification (SRS) document.

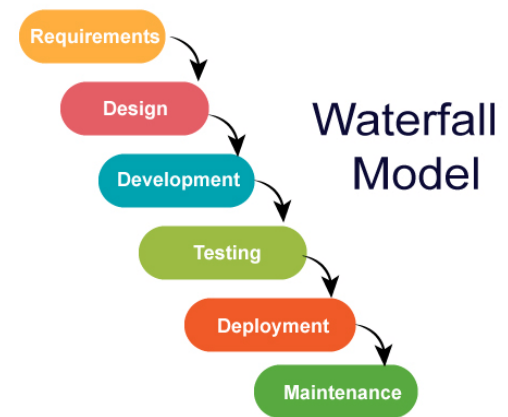
Phases of Waterfall Model



2. System Design:

- ✓ Once the requirements are understood, the design phase begins.
- ✓ Based on the SRS, **the system architecture and design** are **created**.
- ✓ This phase defines **how the software will be built**.
- ✓ **Output:** Design documents, including **system architecture, database design, and module specifications**.

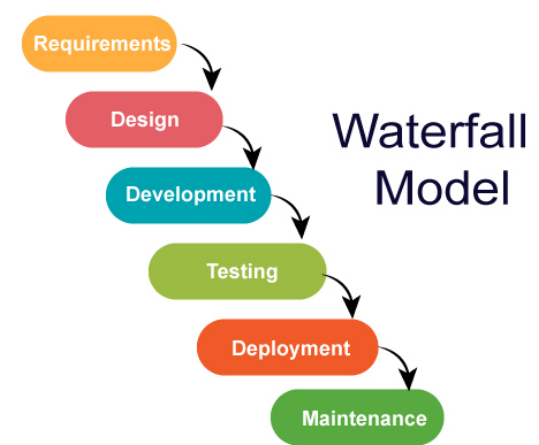
Phases of Waterfall Model



3. Implementation (Coding):

- ✓ Developers write the actual code based on the design documents.
- ✓ The software is divided into smaller modules, and each module is developed and tested individually.
- ✓ **Output:** A working software product.

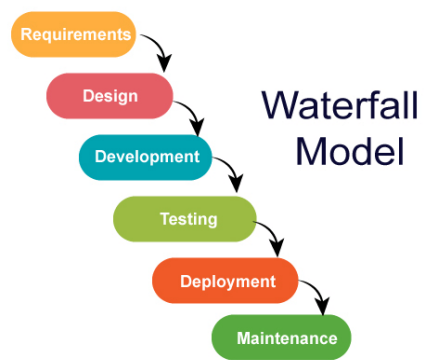
Phases of Waterfall Model



4. Testing:

- ✓ The software is tested **to ensure** it **meets the requirements and is free of bugs.**
- ✓ Testing includes unit testing, integration testing, and system testing.
- ✓ **Output:** A **tested** and **debugged** software product.

Phases of Waterfall Model



5. Deployment:

- ✓ The **software** is **deployed to the production environment for end-users**.
- ✓ **Output:** A live, operational system.

6. Maintenance:

- ✓ After deployment, the **software is maintained to fix any issues, improve performance, or add new features**.
- ✓ **Output:** Updates and patches to the software.

Advantages of Waterfall Model

- ✓ **Simple & Easy to Understand:** Since it **follows** a **step-by-step approach**, it is **easy to manage** and **execute**.
- ✓ **Clear Documentation:** **Each phase** has **well-defined** documentation, which helps in future references.
- ✓ **Well-Structured:** The **phases** are **well organized**, making it **easy to track project progress**.

Advantages of Waterfall Model

- ✓ **Early Detection of Errors:** **Errors** in the requirement phase can be **identified** early before development starts.
- ✓ **Ideal for Small & Well-Defined Projects:** **Works best when requirements are clear from the beginning.**
- ✓ **Better Control & Management:** Since each phase has a defined start and end, project managers can easily **control the timeline and budget.**

Disadvantages of Waterfall Model

- 1.Inflexible:** Once a phase is completed, it is difficult and costly to go back and make changes. This makes the model unsuitable for projects with evolving requirements.
- 2.Late Testing:** Testing is done only after the implementation phase, which means defects may not be detected until late in the process.
- 3.High Risk:** If the requirements are misunderstood or incorrect, the entire project may fail, as changes are hard to implement later.

Disadvantages of Waterfall Model

- 4. Not Suitable for Complex Projects:** For large or complex projects where requirements are likely to change, the Waterfall Model is not ideal.
- 5. Limited Client Feedback:** Clients only see the final product at the end, which may lead to dissatisfaction if the product does not meet their expectations.
- 6. Assumes Requirements are Clear:** The model assumes that all requirements can be clearly defined at the beginning, which is often not the case in real-world projects.

When NOT to use the waterfall

- Model**
- 1. Unclear or Changing Requirements:** If the requirements are not well-defined or are likely to evolve during the project.
 - 2. Large or Complex Projects:** For projects with many dependencies and uncertainties, Agile or Iterative models are better.
 - 3. Innovative Projects:** When the project involves new technologies or innovative solutions, where experimentation and flexibility are needed.
 - 4. Client Involvement is Required:** If the client wants to provide feedback and see progress throughout the project.

Real-Life Scenarios for Using the Waterfall Model

- 1. Building a House:** The requirements (e.g., number of rooms, layout) are clear upfront, and changes are costly once construction begins.
- 2. Developing a Payroll System:** The requirements for calculating salaries, taxes, and deductions are well-defined and unlikely to change.
- 3. Creating a Library Management System:** The features (e.g., book tracking, member management) are straightforward and stable.
- 4. Manufacturing Software:** In industries like automotive or electronics, where requirements are fixed and changes are expensive.
- 5. Government Projects:** Projects with strict regulations and documentation requirements, such as tax systems or voting systems.

In conclusion on Waterfall Model

- The Waterfall Model is best for structured and predictable projects but lacks flexibility for evolving requirements.
- If a project requires adaptability, Agile or Iterative models are better alternatives.

Agile Model

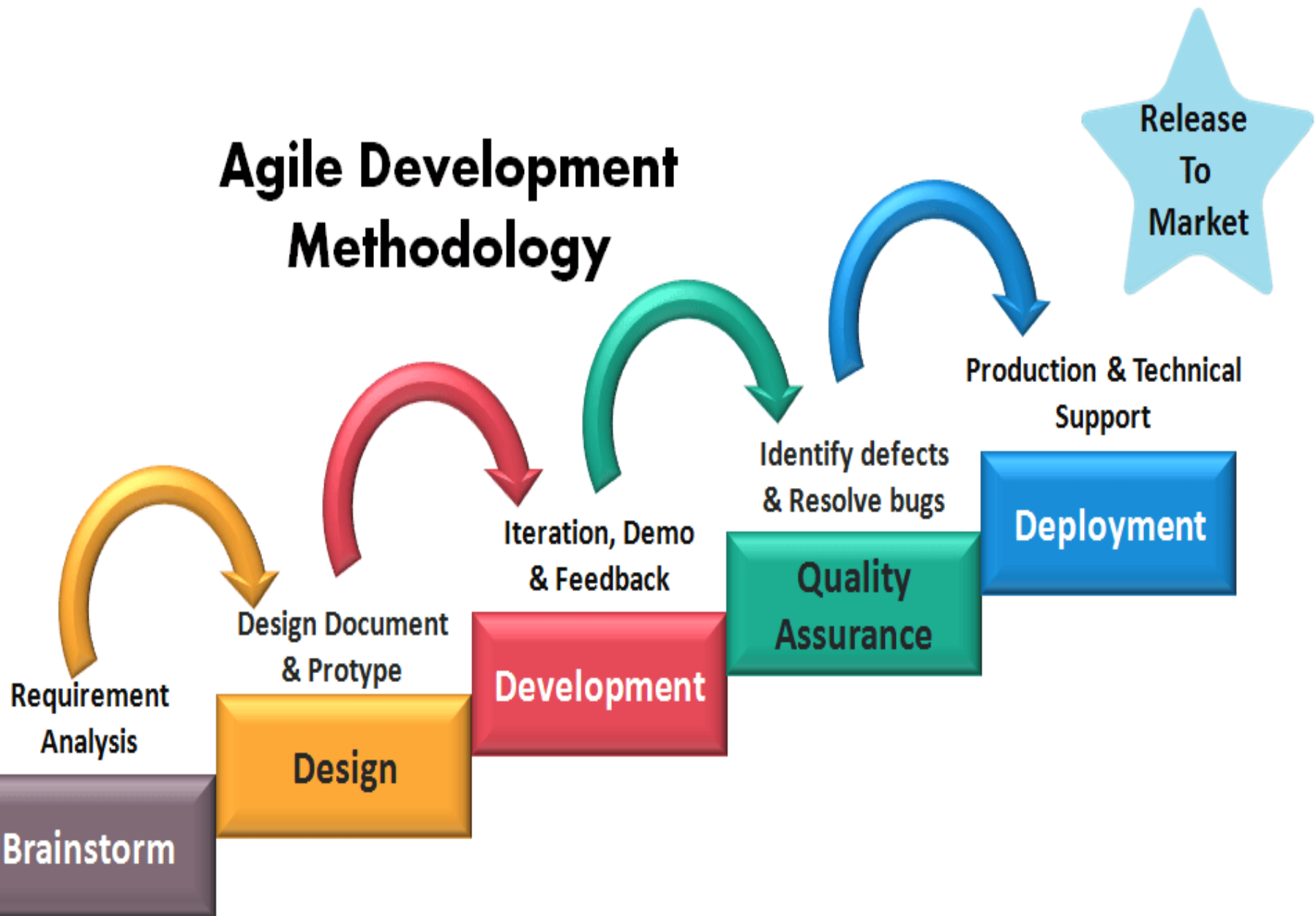
- ✓ In earlier days, the **Iterative Waterfall Model** was very popular for completing a project.
- ✓ But nowadays, developers face various problems while using it to develop software.
- ✓ The main difficulties included **handling customer change requests during project development** and **the high cost and time required to incorporate** these changes.
- ✓ To overcome these drawbacks of the Waterfall Model, in the mid-1990s the **Agile Software Development model** was proposed.

Agile Model

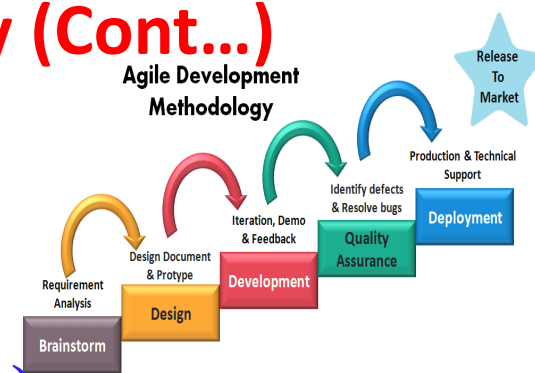
- The Agile Model is a **flexible, iterative, and incremental approach** to software development.
- Unlike the Waterfall Model, Agile **focuses on continuous collaboration, customer feedback, and rapid delivery** of working software.

Process of Agile Development Methodology

Agile Development Methodology



Process of Agile Development Methodology (Cont...)



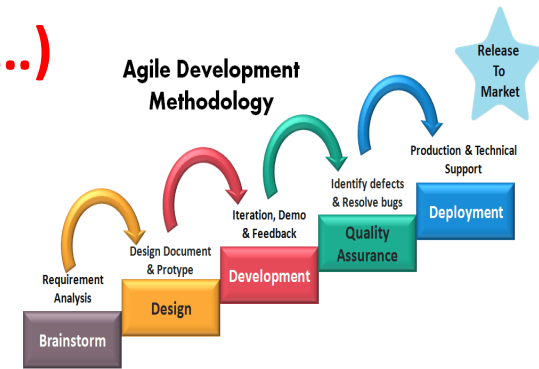
1. Brainstorm (Requirement Analysis):

- ✓ Teams and stakeholders gather and brainstorm requirements for the product.
- ✓ Objectives are clearly defined, ensuring alignment with customer needs.

2. Design (Design Document & Prototype):

- ✓ A design document or prototype is created based on the requirements.
- ✓ This phase involves planning the architecture, workflows, and user interface.

Process of Agile Development Methodology (Cont...)



3. Development (Iteration, Demo & Feedback):

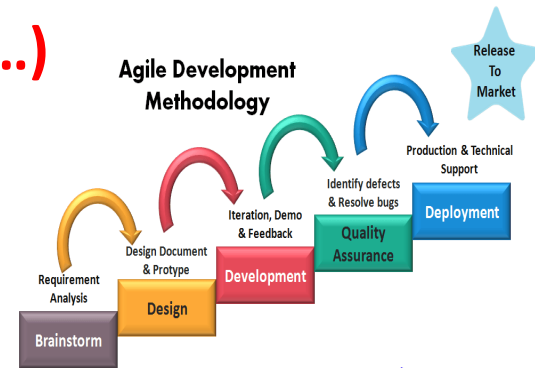
- ✓ Coding and implementation are carried out in iterations (small cycles called **sprints**).

AGILE SOFTWARE DEVELOPMENT LIFE CYCLE



- ✓ After every sprint, working software is delivered for feedback and adjustments.

Process of Agile Development Methodology (Cont...)



4. Quality Assurance (Identify Defects & Resolve Bugs):

- ✓ The developed features undergo testing to identify and fix bugs.
- ✓ Agile emphasizes continuous testing throughout the development cycle.

5. Deployment (Production & Technical Support):

- ✓ The software is deployed to production and made available to users.
- ✓ Ongoing technical support and maintenance are provided.

6. Release to Market:

- ✓ Once all iterations are complete, the product is officially launched to the market.
- ✓ Further updates or enhancements can follow in subsequent Agile cycles.

Advantages of Agile Development Methodology

- 1. Faster Delivery** – Working software is delivered in short iterations.
- 2. Adaptability** – Agile welcomes changing requirements at any stage.
- 3. Better Collaboration** – Continuous interaction between developers, testers, and customers.
- 4. Higher Quality** – Regular testing ensures early bug detection.
- 5. Customer-Centric** – Frequent releases allow customers to provide feedback.
- 6. Reduced Risk** – Small incremental updates reduce the risk of project failure.

Disadvantages of Agile Development Methodology

- 1. Lack of Fixed Scope** – Constant changes may lead to scope creep.
- 2. Requires Active Customer Involvement** – Not ideal if clients are unavailable. The agile model depends highly on customer interactions so if the customer is not clear, then the development team can be driven in the wrong direction.
- 3. Difficult to Predict Cost & Timeline** – Continuous iterations make budgeting challenging.
- 4. High Degree of Expertise Team Members-** as they need to be able to adapt to changing requirements and work in an iterative environment.
- 5. Less Documentation** – Agile prefers working software over detailed documents.
- 6. Not Ideal for Large Teams** – Scaling Agile for large enterprises can be complex.

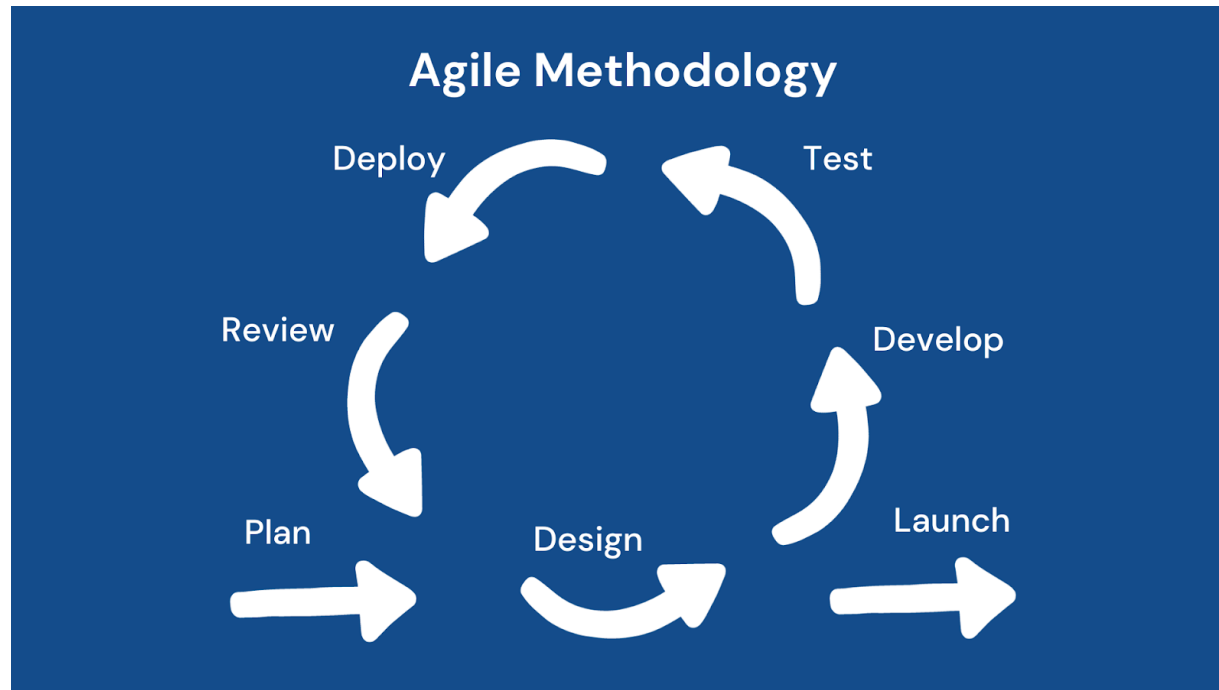
When to Use Agile Development Model

1. Agile is best when project requirements are not fixed.
2. When frequent modifications need to be made, this method is implemented.
3. When a highly qualified and experienced team is available.
4. When a customer is ready to have a meeting with the team all the time.
5. when the project needs to be delivered quickly.
6. Flexible project schedules and budgets.
7. For Customer-Focused Applications – Ideal for web apps, mobile apps, and SaaS products where continuous user feedback is needed.
8. For Fast-Paced Development – Agile is useful when speed and quick market release are priorities.
9. For Startups & Innovative Projects – Helps startups build MVPs (Minimum Viable Products) and test ideas quickly.
10. When Working in Small Teams – Agile is best for collaborative, cross-functional teams.

Real Life Scenario for Agile Model

- 1. E-commerce Platforms (Amazon, Shopify)** – Frequent updates and changing customer needs.
- 2. Mobile App Development (Facebook, WhatsApp)** – Continuous feature releases.
- 3. SaaS (Software as a Service) (Google Workspace, Microsoft 365)**
– Ongoing improvements.
- 4. Game Development (Fortnite, PUBG)** – Regular updates based on player feedback.
- 5. AI & Machine Learning Projects** – Agile allows iterative model training and testing.

Agile Development Model



- ✓ The Agile Model is ideal for fast-paced projects where customer feedback, flexibility, and incremental delivery are essential.

Key Differences: Waterfall and Agile Model

Waterfall Model Vs Agile Model

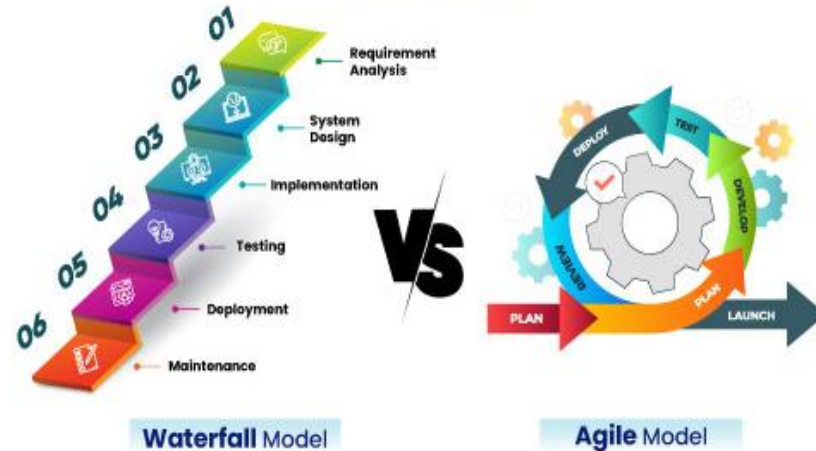
Choose whichever is suitable for your software project



Key Differences: Waterfall and Agile Model

Waterfall Model Vs Agile Model

Choose whichever is suitable for your software project



Feature	Waterfall	Agile
Approach	Sequential	Iterative & Incremental
Flexibility	Rigid, no changes allowed	Adaptive, welcomes changes
Customer Involvement	Low, feedback at the end	High, continuous feedback
Delivery	Single final product release	Frequent small releases
Testing Phase	At the end	Continuous testing
Documentation	Heavy documentation	Working software over documentation
Best For	Fixed-scope projects (e.g., Banking, Defense)	Dynamic projects (e.g., Mobile Apps, Startups)

Key Differences: Waterfall and Agile Model

PROS

Waterfall is **well-disciplined**



It has to start with **complete requirements**



Great for projects with **defined phases**



Focuses more on **customer collaboration**



Upholds individuals & **interactions**



It advances towards a **working software**



It puts emphasis on **Fast delivery**

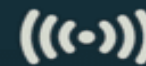


WATERFALL METHODOLOGY (PREDICTIVE MODEL)



AGILE METHODOLOGY (ADAPTIVE MODEL)

CONS



Early feedback is **absent**



Response to change is **slow**



Risk of **high cost** if a requirement is **missing**



Lack of **documentation**



Difficulty in building **great design**



Lack of emphasis in necessary **designing**



Can easily get off track if goals are **unclear**

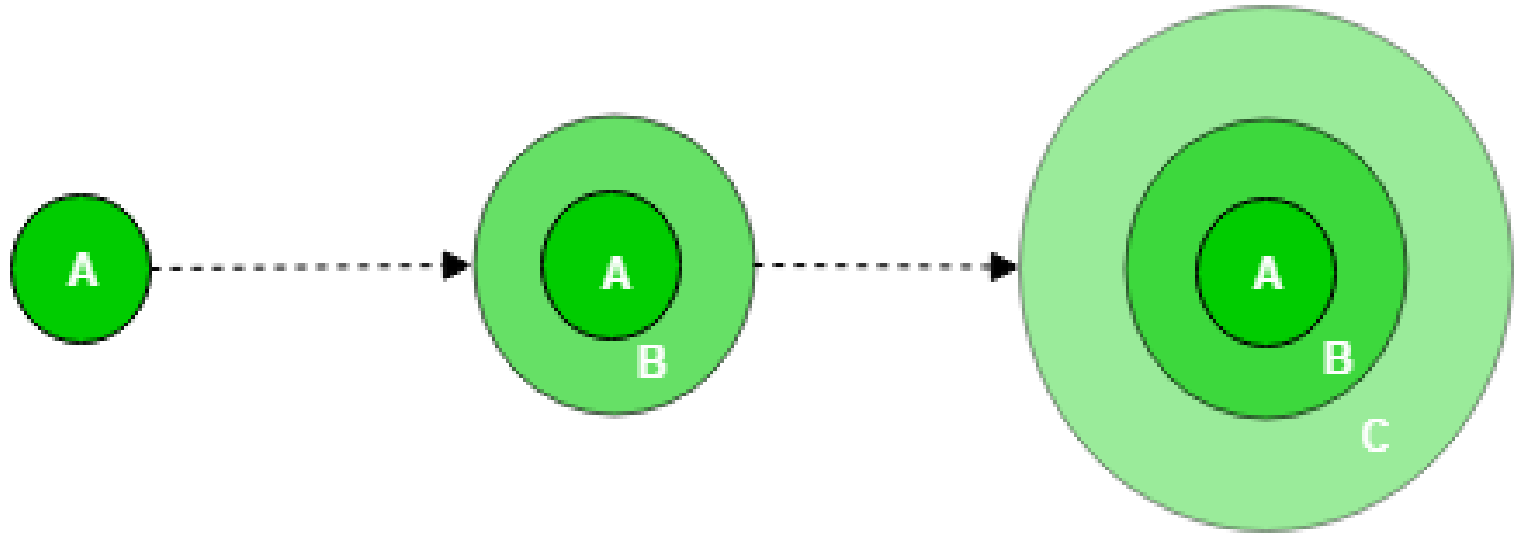


Difficulty in assessing efforts required at the **beginning**

Incremental Process Model

- The Incremental Process Model is also known as the Successive version model.
- First, a simple working system implementing only a few basic features is built and then that is delivered to the customer.
- Then thereafter many successive iterations/ versions are implemented and delivered to the customer until the desired system is released.
- The Incremental Model is widely used in real-life software development because it allows teams to deliver functional products early, incorporate feedback, and manage risks effectively.

Incremental Process Model



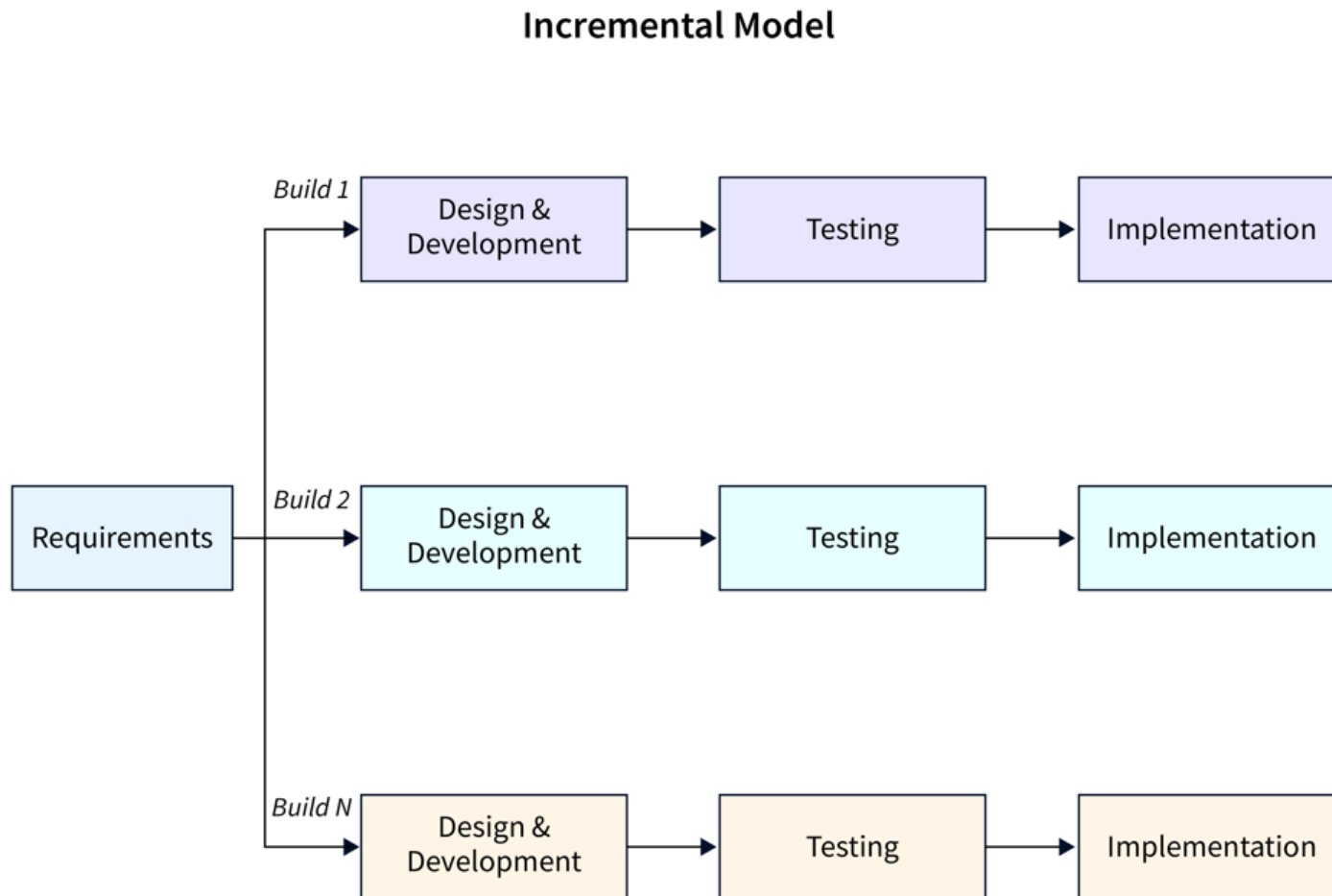
A, B, and C are modules of Software Products that are incrementally developed and delivered.

Key Characteristics of Incremental Model

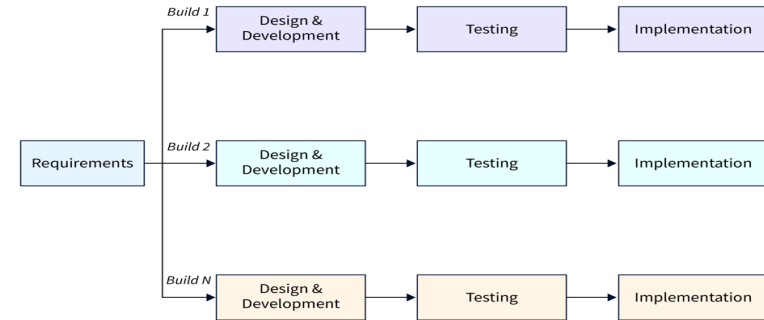
- 1. Divided into Increments:** The project is broken down into smaller, manageable modules or increments.
- 2. Iterative Development:** Each increment goes through the phases of requirements, design, implementation, and testing.
- 3. Early Delivery:** Partial functionality is delivered early, allowing users to benefit sooner.
- 4. Flexibility:** Changes can be incorporated in later increments.
- 5. Risk Management:** Risks are identified and addressed in early increments.

Phases of Incremental Process Model

- ✓ Requirements of the Software are first broken down into several modules that can be incrementally constructed and delivered



Phases of Incremental Process Model



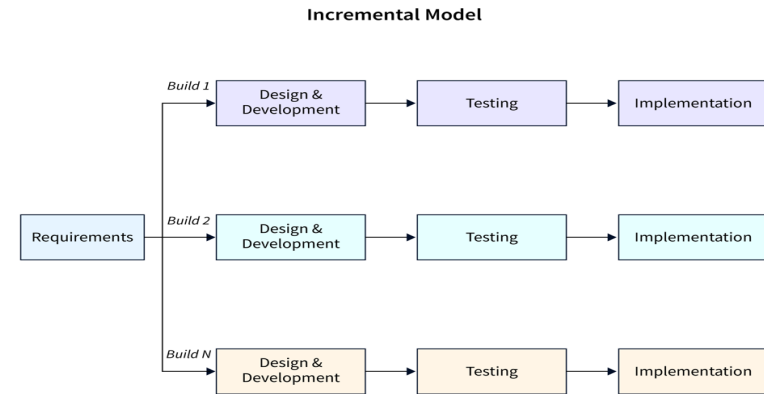
1. Requirement analysis:

- ✓ In Requirement Analysis **at any time**, the **plan is made just for the next increment** and **not for** any kind of **long-term plan**.
- ✓ Therefore, it is **easier to modify** the **version as per the needs of the customer**.

2. Design & Development:

- ✓ The Development Team **first undertakes** to develop **core features** (these do not need services from other features) of the system.
- ✓ Once the core features are fully developed, then these are refined to increase levels of capabilities by adding new functions in Successive versions.
- ✓ **Each incremental version** is usually developed **using an iterative waterfall** model of development.

Phases of Incremental Process Model



3. Deployment and Testing:

- ✓ After Requirements gathering and specification, **requirements** are then **split into several different versions** starting with version 1, in each successive increment, the next version is constructed and then deployed at the customer site.
- ✓ In development and Testing the **product is checked and tested** for the actual process of the model.

4. Implementation:

- ✓ In implementation After the last version (**version n**), it is now **deployed at the client site**.

Advantages of Incremental Process Model


1. **Early Delivery:** Users get to use parts of the system early.
2. **Flexibility:** Changes can be made in later increments.
3. **Risk Management:** Risks are identified and mitigated early.
4. **Easier Testing:** Smaller increments are easier to test and debug.
5. **Customer Feedback:** Users can provide feedback on early increments, improving the final product.

Disadvantages of Incremental Process Model

1. **Planning Required:** Requires careful planning to divide the system into increments.
2. **Integration Challenges:** Integrating increments can be complex.
3. **Cost:** May be more expensive due to repeated testing and deployment.
4. **Dependency:** Each increment depends on the previous one, so delays can affect the timeline.

Social Media App Development:

----Real Life Example of Incremental Model

- Imagine developing a social media app like Instagram:
 - ✓ Increment 1: Basic photo upload and sharing.
 - ✓ Increment 2: Add filters and editing tools.
 - ✓ Increment 3: Introduce likes, comments, and followers.
 - ✓ Increment 4: Add Stories and Reels.
 - ✓ Increment 5: Implement AI-based content recommendations.
- Each increment builds on the previous one, delivering value to users step by step.

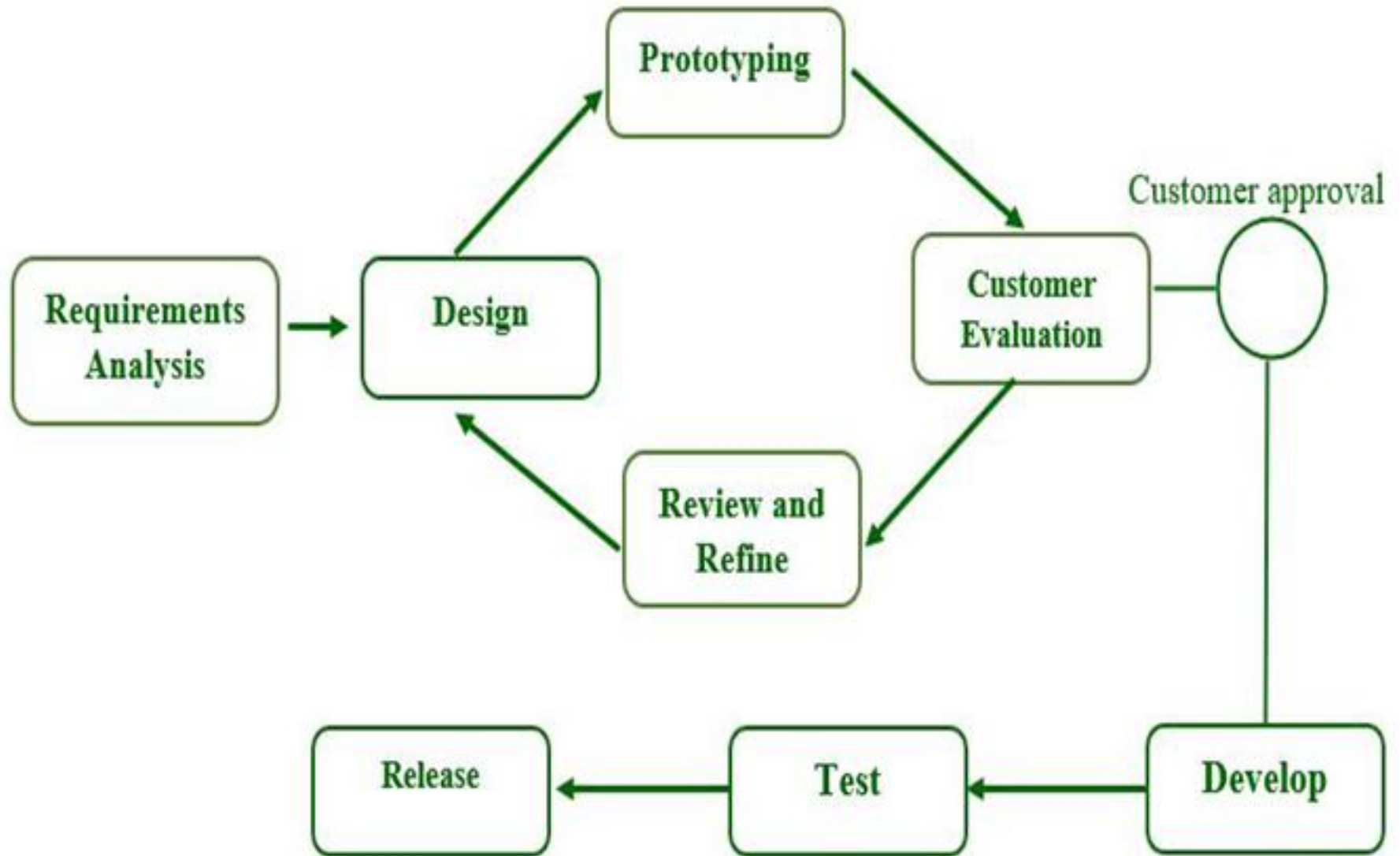
Prototyping Model

- The **Prototyping Model** is one of the most popularly used Software Development Life Cycle Models (SDLC models).
- This model is **used** when **the customers do not know the exact project requirements beforehand**.
- In this model, **a prototype of the end product** is **first developed, tested,** and **refined as per customer feedback** repeatedly till a final acceptable prototype is achieved which forms the basis for developing the final product.

Prototyping Model

- In this process model, the **system is partially implemented before or during the analysis phase** thereby allowing the customers **to see the product early** in the life cycle.

Steps of Prototyping Model



Six SDLC Phases of Prototyping Model

1. Requirement Gathering and Analysis:

- ✓ This is the **initial step** in designing a prototype model.
- ✓ In this phase, users are asked about **what they expect** or **what they want** from the system.

2. Quick Design:

- ✓ This is the **second step** in the Prototyping Model.
- ✓ This model covers the basic design of the requirement through which **a quick overview can be easily described**.

3. Build a Prototype:

- ✓ This step helps in building an actual prototype from the knowledge gained from prototype design.

Six SDLC Phases of Prototyping Model

4. Initial User Evaluation:

- ✓ This step describes the preliminary testing where the investigation of the performance model occurs, as the **customer will tell the strengths and weaknesses of the design**, which was sent to the developer.

5. Refining Prototype:

If any feedback is given by the user, then improving the client's response to feedback and suggestions, the final system is approved.

6. Implement Product and Maintain:

This is the **final step** in the phase of the Prototyping Model **where the final system is tested and distributed to production**, **here the program is run regularly to prevent failures.**

When to Use the Prototype Model?

1. When **requirements** are **unclear or incomplete**.
2. When developing user-centric applications.
3. For **complex systems** where **UI/UX is a priority**.
4. When **early validation** is **needed** before **full-scale development**.

Types of Prototypes Model

- 1. Throwaway/Rapid Prototype** – Built quickly to understand requirements and then discarded.
- 2. Evolutionary Prototype** – Continuously improved until it becomes the final product.
- 3. Incremental Prototype** – Divides the system into smaller parts, each prototyped separately.
- 4. Extreme Prototyping** – Used in web applications, with multiple layers (presentation, logic, services).

Advantages of Prototype Model

- ✓ The **customers** get to **see the partial product early** in the life cycle. This ensures a greater level of **customer satisfaction and comfort**.
- ✓ **New requirements can be easily accommodated** as there is **scope for refinement**.
- ✓ Missing functionalities can be easily figured out.
- ✓ Errors can be detected much earlier thereby saving a lot of effort and cost, besides enhancing the quality of the software.
- ✓ The developed prototype can be reused by the developer for more complicated projects in the future.
- ✓ Flexibility in design.
- ✓ Early feedback from customers and stakeholders.
- ✓ Prototyping can help reduce the risk of project failure by identifying potential issues and addressing them early in the process.
- ✓ Improves user involvement in development.
- ✓ Prototyping can help bridge the gap between technical and non-technical stakeholders by providing a tangible representation of the product.

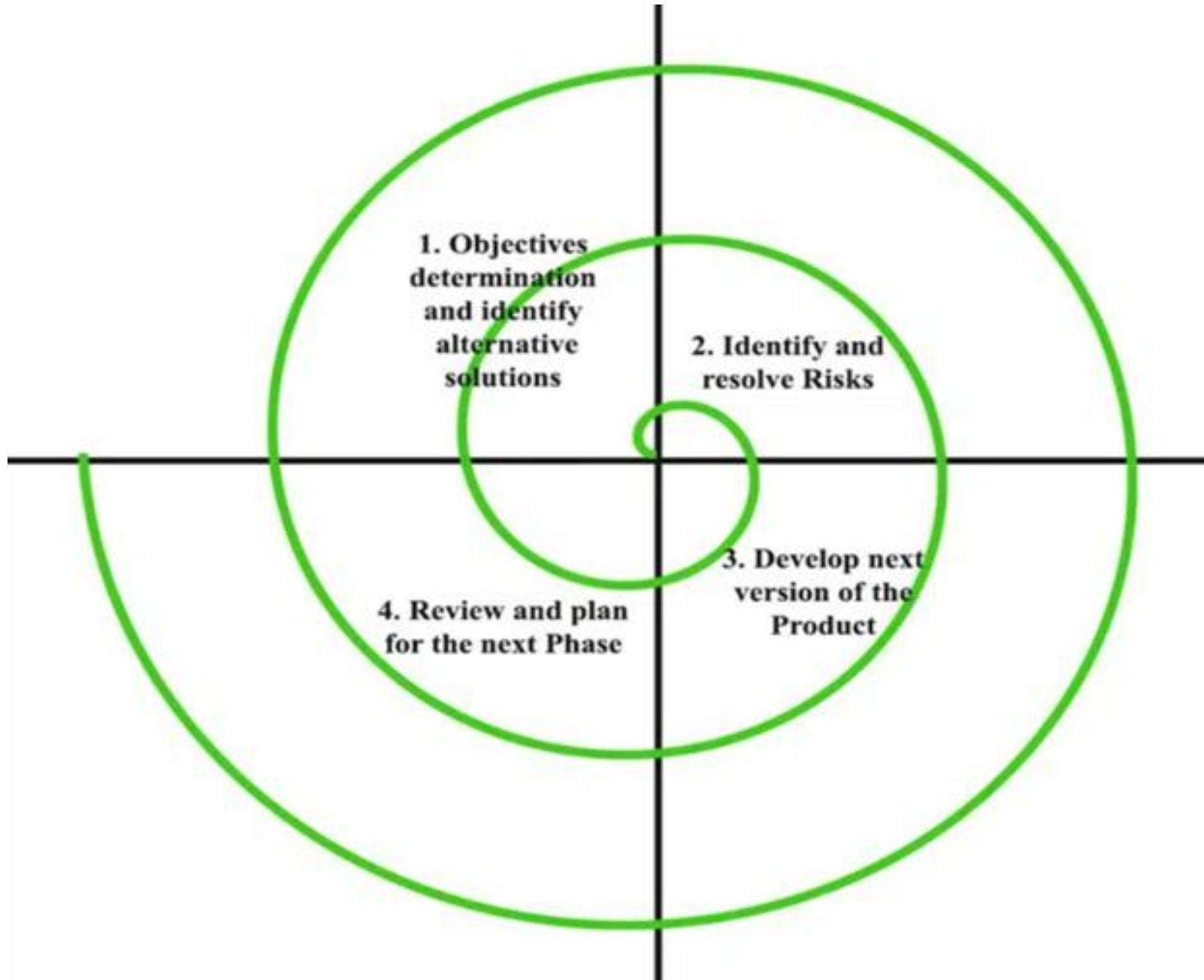
Disadvantages of Prototype Model

1. Can lead to scope creep due to continuous changes.
2. Requires extra development effort for prototypes.
3. May increase costs and development time.

Spiral Model

- ✓ The Spiral Model is one of the **most important** Software Development Life Cycle models.
- ✓ The Spiral Model is a **combination of the waterfall model and the iterative model**.
- ✓ It provides support for **Risk Handling**.
- ✓ The Spiral Model was first proposed by **Barry Boehm**
- ✓ The Spiral Model is **a risk-driven model**, meaning that the **focus** is on **managing risk through multiple iterations** of the software development process.

Phases in Spiral Model



Phases in Spiral Model

1. Objectives Defined:

- ✓ In first phase of the spiral model we clarify what the project aims to achieve, including functional and non-functional requirements.

2. Risk Analysis:

- ✓ In the risk analysis phase, the risks associated with the project are identified and evaluated.

3. Engineering:

- ✓ In the engineering phase, the software is developed based on the requirements gathered in the previous iteration.

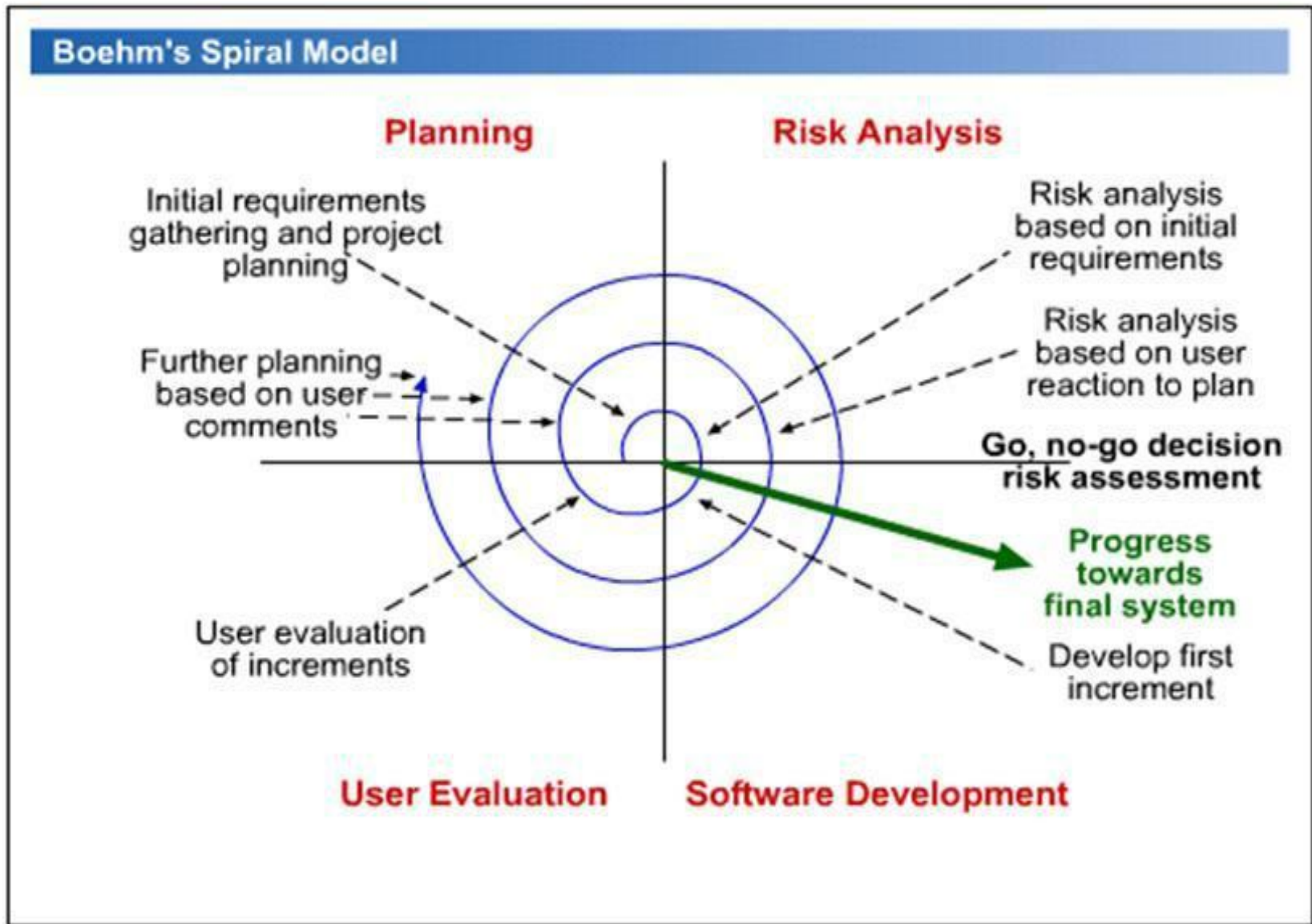
4. Evaluation:

- ✓ In the evaluation phase, the software is evaluated to determine if it meets the customer's requirements and if it is of high quality.

5. Planning:

- ✓ The next iteration of the spiral begins with a new planning phase, based on the results of the evaluation

Barry Boehm Spiral Model



✓ Each phase of the Spiral Model is divided into four quadrants as shown in the above figure.

The functions of the four quadrants Phases in Spiral Model

1. Objectives determination and identify alternative solutions:

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

2. Identify and resolve Risks:

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

The functions of the four quadrants Phases in Spiral Model

3. Develop the next version of the Product:

- ✓ During the third quadrant, the identified features are developed and verified through testing.
- ✓ At the end of the third quadrant, the **next version of the software is available.**

4. Review and plan for the next Phase:

- ✓ In the fourth quadrant, the Customers evaluate the so-far developed version of the software.
- ✓ In the end, **planning for the next phase is started.**

Why Spiral Model is called Meta Model?

- ✓ The Spiral model is called a **Meta-Model** because it **subsumes** all the other SDLC models.
- ✓ For example, a single loop spiral actually represents the Iterative Waterfall Model.
 - I. The spiral model incorporates the stepwise approach of the Classical Waterfall Model.
 - II. The spiral model uses the approach of the Prototyping Model by building a prototype at the start of each phase as a risk-handling technique.

Advantages of the Spiral Model

- ✓ Risk Handling
- ✓ Good for large projects
- ✓ Flexibility in Requirements
- ✓ Customer Satisfaction
- ✓ Iterative and Incremental Approach
- ✓ Emphasis on Risk Management
- ✓ Improved Communication
- ✓ Improved Quality

Disadvantages of the Spiral Model

- ✓ Complex
- ✓ Expensive
- ✓ Too much dependability on Risk Analysis
- ✓ Difficulty in time management
- ✓ Complexity
- ✓ Time-Consuming
- ✓ Resource Intensive

Real-Life Example of Spiral Model: Developing an E-Commerce Website

- ✓ **First Spiral – Planning and Requirements:** The initial phase involves gathering basic requirements for the e-commerce website, like product listing, shopping cart, and payment options.
- ✓ The team analyzes any risks, such as security or scalability, and creates a small prototype.

❖ **Example:**

- ❖ The team builds a simple homepage with a basic product catalog to see how users interact with it and identify any design flaws.

Real-Life Example of Spiral Model: Developing an E-Commerce Website

- **Second Spiral – Risk Analysis and Refining the Design:** After gathering feedback from the prototype, the next spiral focuses on adding more features and fixing early issues.
- The team addresses security risks, such as secure payment processing, and tests how well the site handles increasing user traffic.
 - **Example:**
 - A basic shopping cart and user registration system are added.
 - The payment system is also tested with dummy transactions to ensure security.

Real-Life Example of Spiral Model: Developing an E-Commerce Website

- **Third Spiral – Detailed Implementation:** With more feedback, the team further refines the design, adding advanced features like order tracking, customer reviews, and search functionality.
- Risks like scalability (handling many users) are re-evaluated, and more testing is conducted.

- **Example:**

- The website now supports user profiles, product reviews, and real-time inventory updates.
- The team tests how the system handles large volumes of orders during peak times.

Real-Life Example of Spiral Model: Developing an E-Commerce Website

- **Final Spiral – Full Deployment:** The final phase involves full implementation, thorough testing, and launching the e-commerce website to the public. Ongoing risks like system crashes or user feedback are monitored and addressed as needed.

- **Example:**

- The website goes live with all features, including secure payments, product listings, and order tracking, ready for users to shop online.

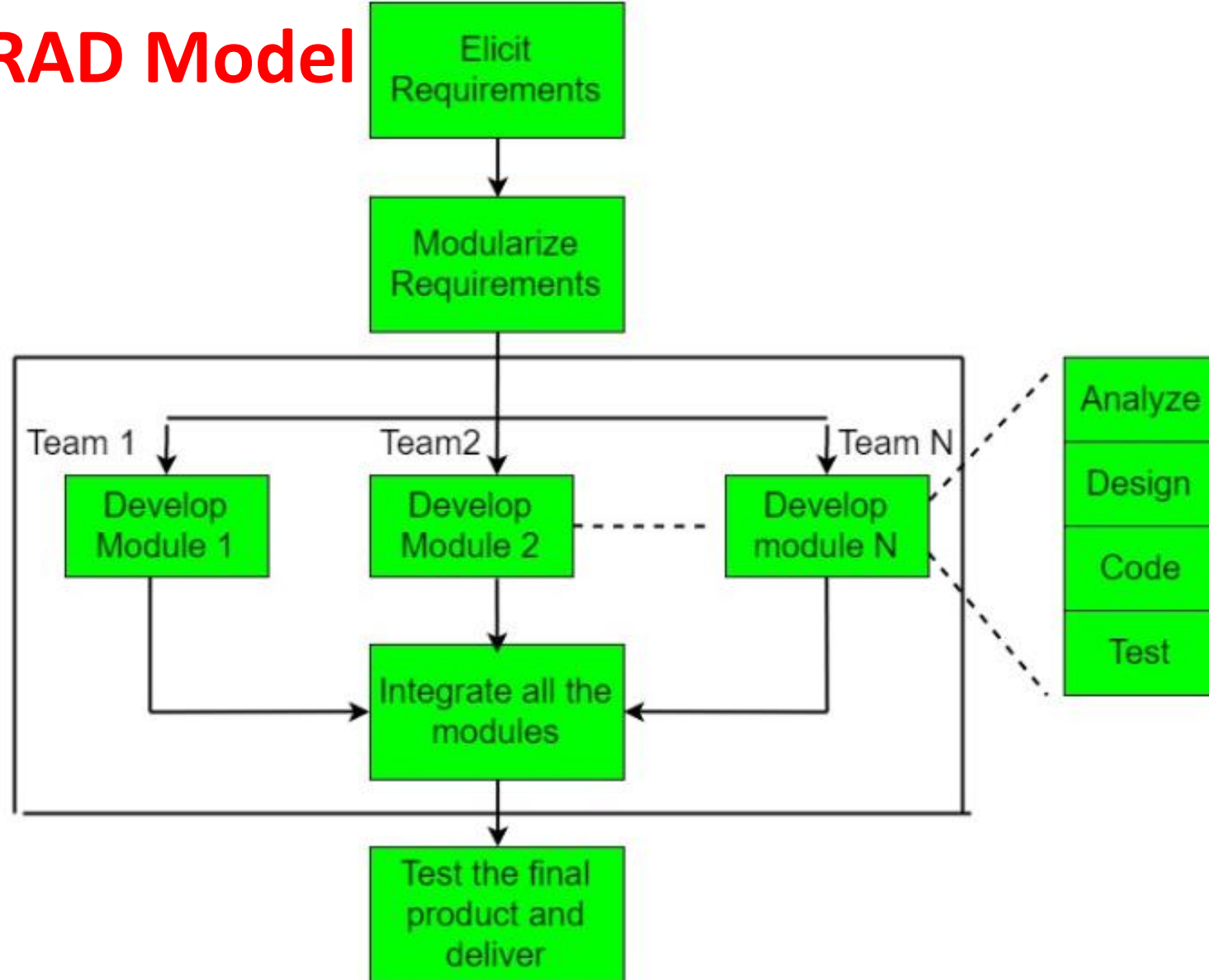
Rapid Application Development or RAD Model

- ✓ IBM first proposed the Rapid Application Development or RAD Model in the 1980s.
- ✓ The RAD model is a type of incremental process model in which there is a concise development cycle.
- ✓ The RAD model is used when the requirements are fully understood and the component-based construction approach is adopted.
- ✓ Various phases in RAD are **Requirements Gathering**, **Analysis** and **Planning**, **Design**, **Build or Construction**, and finally **Deployment**.
- ✓ A software project can be implemented using this model if the project can be broken down into small modules where is each module can be assigned independently to separate teams.
- ✓ These modules can finally be combined to form the final product.⁷⁷

Rapid Application Development or RAD Model

- ✓ Development of each module involves the various basic steps as in the waterfall model i.e. analyzing, designing, coding, and then testing, etc. as shown in the figure.
- ✓ Another striking feature of this model is a short period i.e. the time frame for delivery(time-box) is generally 60-90 days.
- ✓ Multiple teams work on developing the software system using the RAD model parallel.

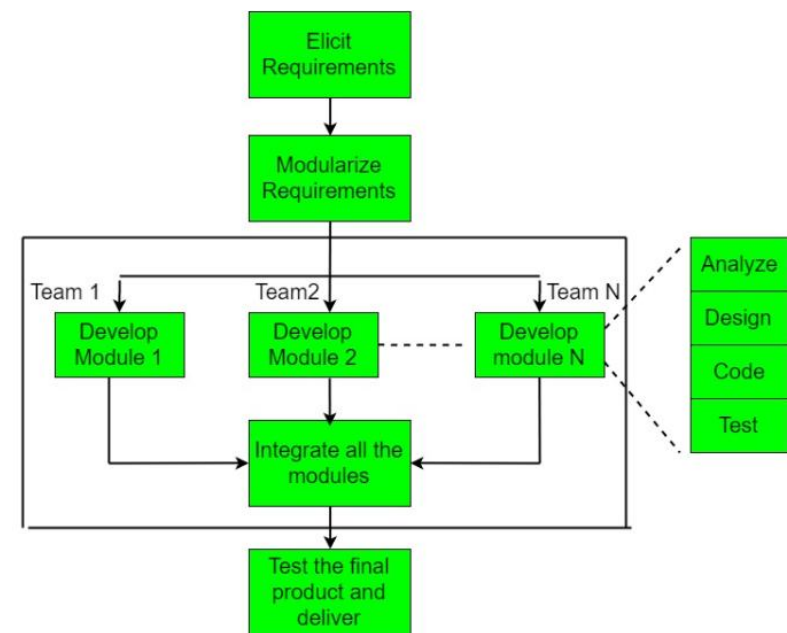
Phases of RAD Model



This model consists of 4 basic phases:

- Requirements Planning:
- User Description:
- Construction:
- Cutover:

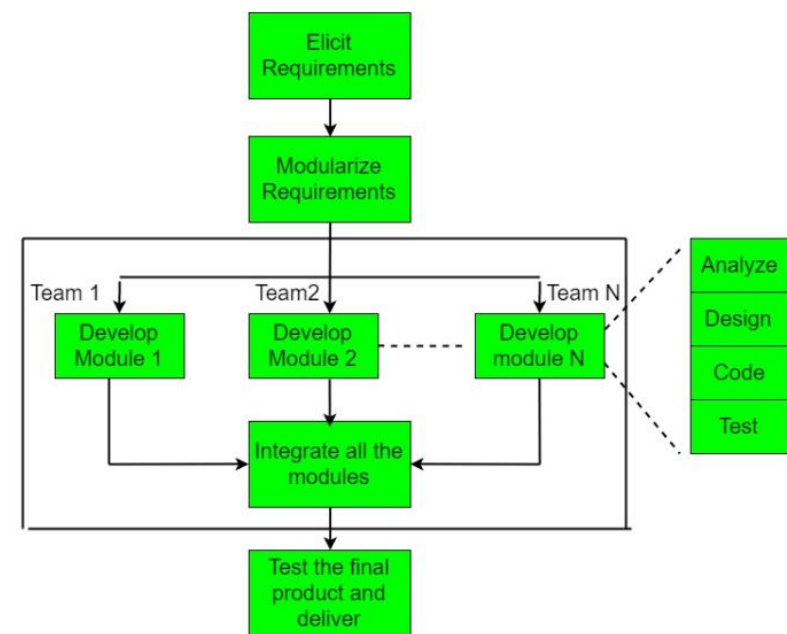
Phases of RAD Model



1. Requirements Planning:

- ✓ This involves the use of various techniques used in requirements elicitation like brainstorming, task analysis, form analysis, user scenarios, FAST (Facilitated Application Development Technique), etc.
- ✓ It also consists of the entire structured plan describing the critical data, methods to obtain it, and then processing it to form a final refined model.

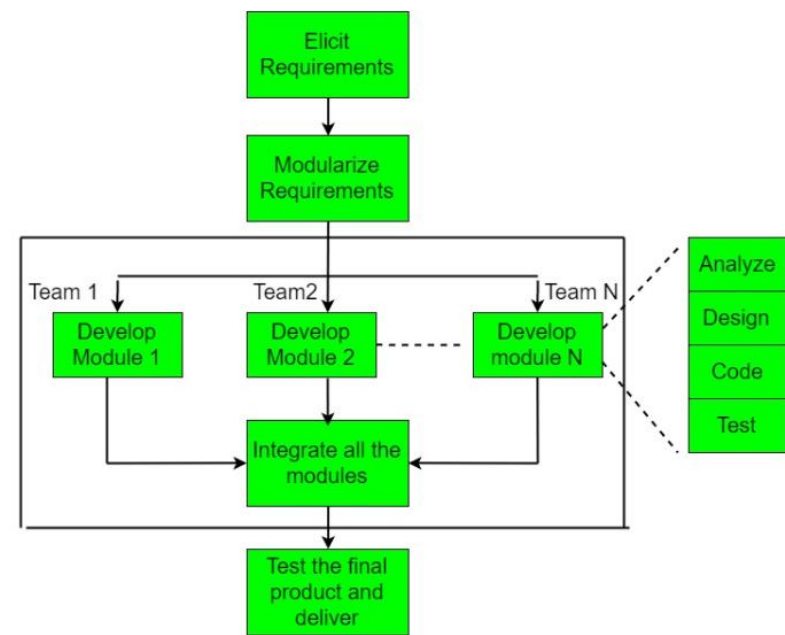
Phases of RAD Model



2. User Description:

- ✓ This phase consists of taking user feedback and building the prototype using developer tools.
- ✓ In other words, it includes re-examination and validation of the data collected in the first phase.
- ✓ The dataset attributes are also identified and elucidated in this phase.

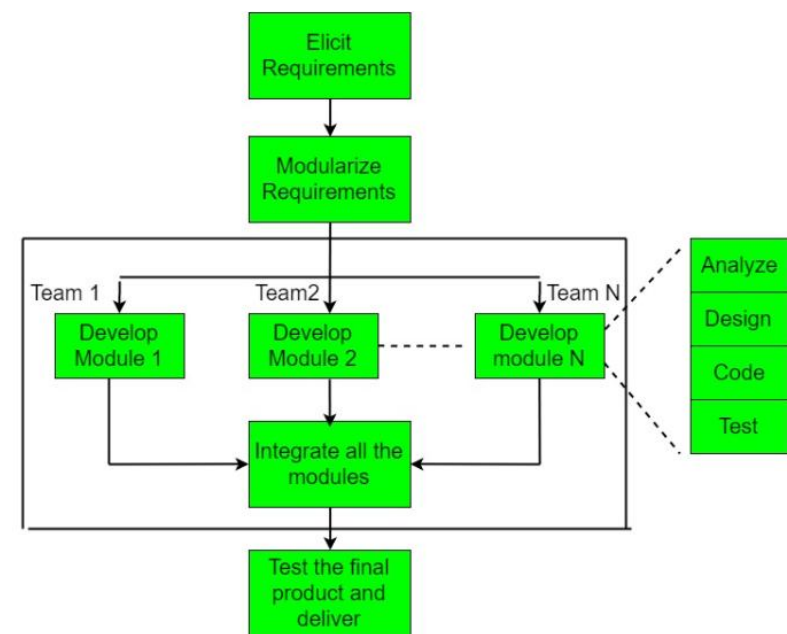
Phases of RAD Model



3. Construction:

- ✓ In this phase, refinement of the prototype and delivery takes place.
- ✓ It includes the actual use of powerful automated tools to transform processes and data models into the final working product.
- ✓ All the required modifications and enhancements are to be done in this phase.

Phases of RAD Model



4. Cutover:

- ✓ All the interfaces between the independent modules developed by separate teams have to be tested properly.
- ✓ The use of powerfully automated tools and subparts makes testing easier.
- ✓ This is followed by acceptance testing by the user.
- ✓ The process involves building a rapid prototype, delivering it to the customer, and taking feedback. After validation by the customer, the SRS document is developed and the design is finalized.

When to use the RAD Model?

- ✓ **Well-understood Requirements:** When project requirements are stable and transparent, RAD is appropriate.
- ✓ **Time-sensitive Projects:** Suitable for projects that need to be developed and delivered quickly due to tight deadlines.
- ✓ **Small to Medium-Sized Projects:** Better suited for smaller initiatives requiring a controllable number of team members.
- ✓ **High User Involvement:** Fits where ongoing input and interaction from users are essential.
- ✓ **Innovation and Creativity:** Helpful for tasks requiring creative inquiry and innovation.
- ✓ **Prototyping:** It is necessary when developing and improving prototypes is a key component of the development process.
- ✓ **Complexity:** Suitable for tasks using comparatively straightforward technological specifications.

Advantages of RAD Model

- ✓ Faster delivery of working software.
- ✓ High adaptability to changing requirements.
- ✓ Strong collaboration between developers and users.
- ✓ Feedback from the customer is available at the initial stages.
- ✓ Reduced costs as fewer developers are required.
- ✓ The use of powerful development tools results in better quality products in comparatively shorter periods.
- ✓ Productivity may be quickly boosted with a lower number of employees.

Disadvantages of RAD Model

- ✓ Requires highly skilled and motivated teams.
- ✓ Not suitable for large, complex projects with fixed requirements.
- ✓ Can lead to scope creep if not managed properly.
- ✓ The use of powerful and efficient tools requires highly skilled professionals.
- ✓ The absence of reusable components can lead to the failure of the project.
- ✓ The team leader must work closely with the developers and customers to close the project on time.
- ✓ The systems which cannot be modularized suitably cannot use this model.
- ✓ Customer involvement is required throughout the life cycle.