Unit-2
Software Development Life Cycle Models

(Marks: 12)

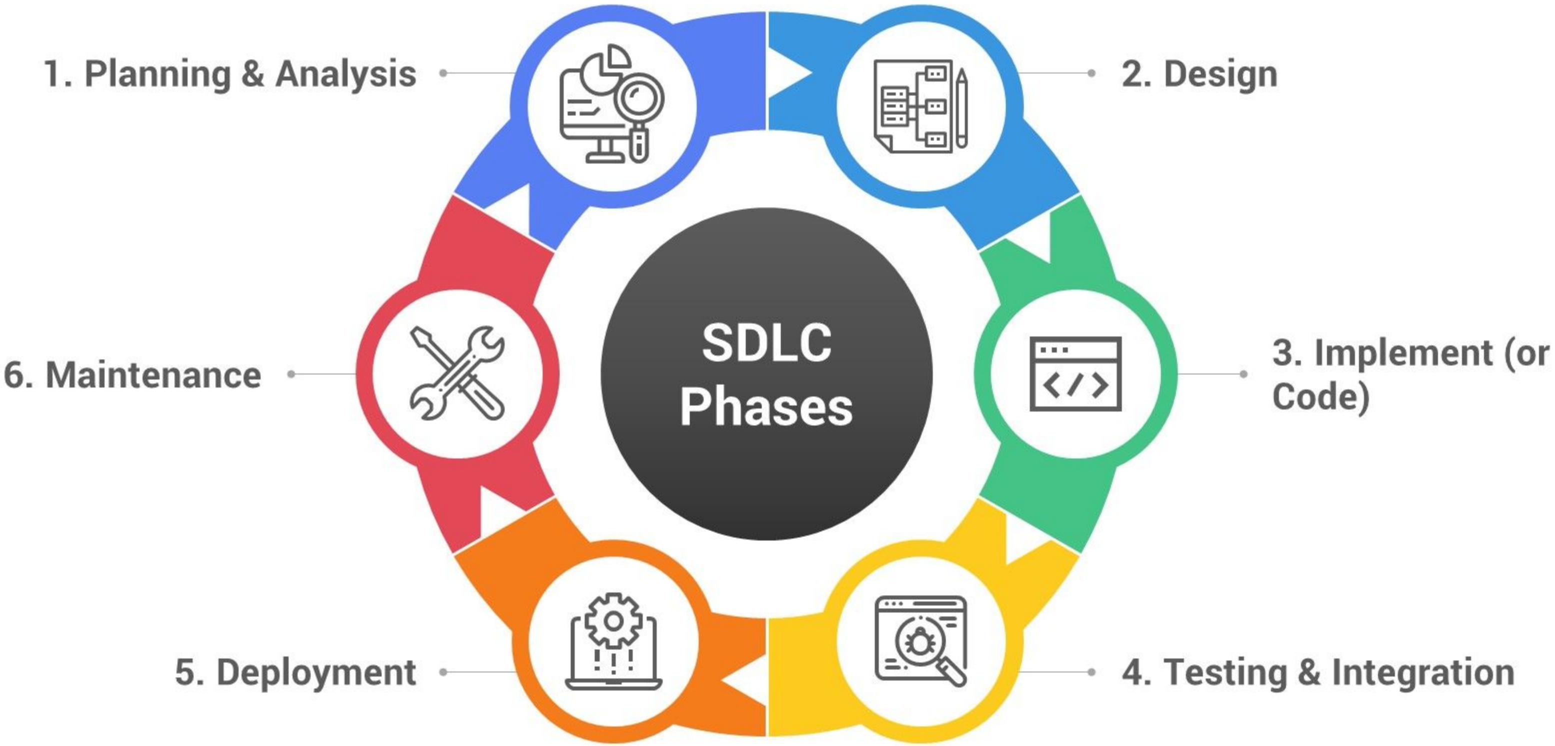
CTEVT Diploma in Computer Engineering
Subject : Software Engineering
Shree Sarwajanik Secondary Technical School
Prepared By: Er. Abinash Adhikari

Definition

The Software Development Life Cycle (SDLC) is a structured process used to design, develop, test, and deploy software applications efficiently and systematically. It defines the stages and steps that developers follow to create high-quality software that meets user requirements and is delivered on time and within budget.

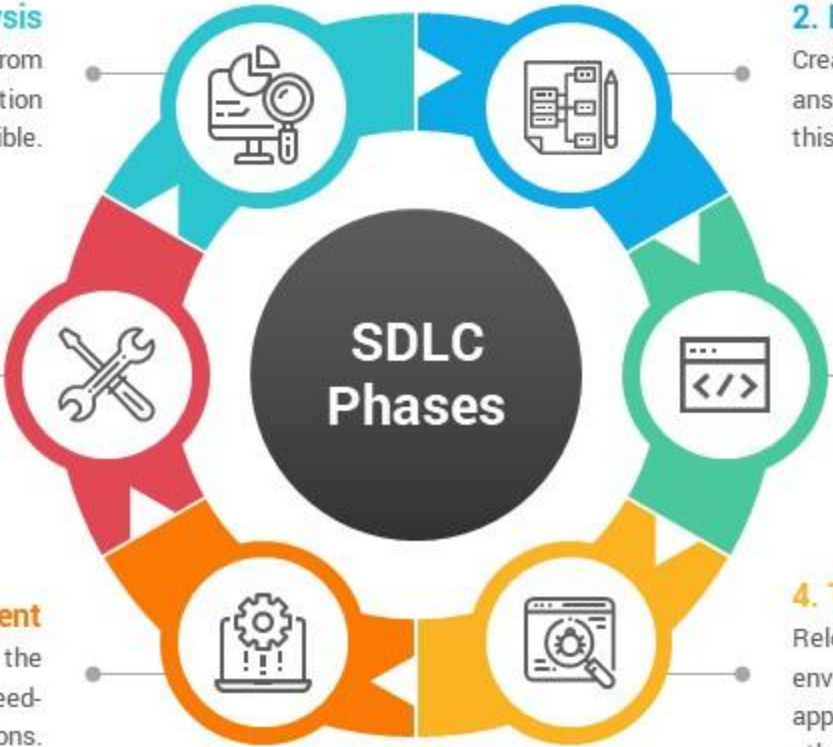
Software Development Life Cycle (SDLC) Phases

Software Development Life Cycle (SDLC) Phases



Software Development Life Cycle (SDLC) Phases

Software Development Life Cycle (SDLC) Phases



1. Planning & Analysis
Gather all relevant information from stakeholders and analyze this information to determine what will be feasible.

2. Design
Create a design specification. This should answer the question, "How will we build this?".

3. Implement (or Code)
Building the software using a programming language by the development team.

4. Testing & Integration
Release the application to a testing environment, test all areas of the application to spot any defects, bugs, or other problems.

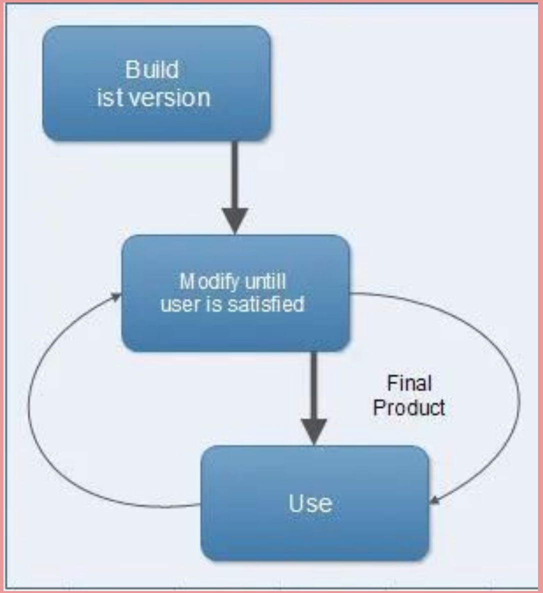
5. Deployment
All defects have been fixed and the software has been built to the agreed-upon goals and specifications.

6. Maintenance
Updating and supporting the software after it has been delivered to the market.

Software Development Life Cycle (SDLC) Models

1. Build and fix model
2. The waterfall model
3. Prototyping model
4. Iterative enhancement model
5. Spiral model
6. Rapid application development model (RAD)

Build and Fix Model



- ❖ The Build and Fix model (also called the Ad Hoc model) is an unstructured software development approach where code is created and continually fixed until it meets user expectations.
- ❖ No formal design or planning is involved, with modifications made based on user feedback.
- ❖ Although this model allows quick changes, it often results in reduced software quality and acceptance due to lack of structure. Consequently, it is discouraged in professional software engineering.

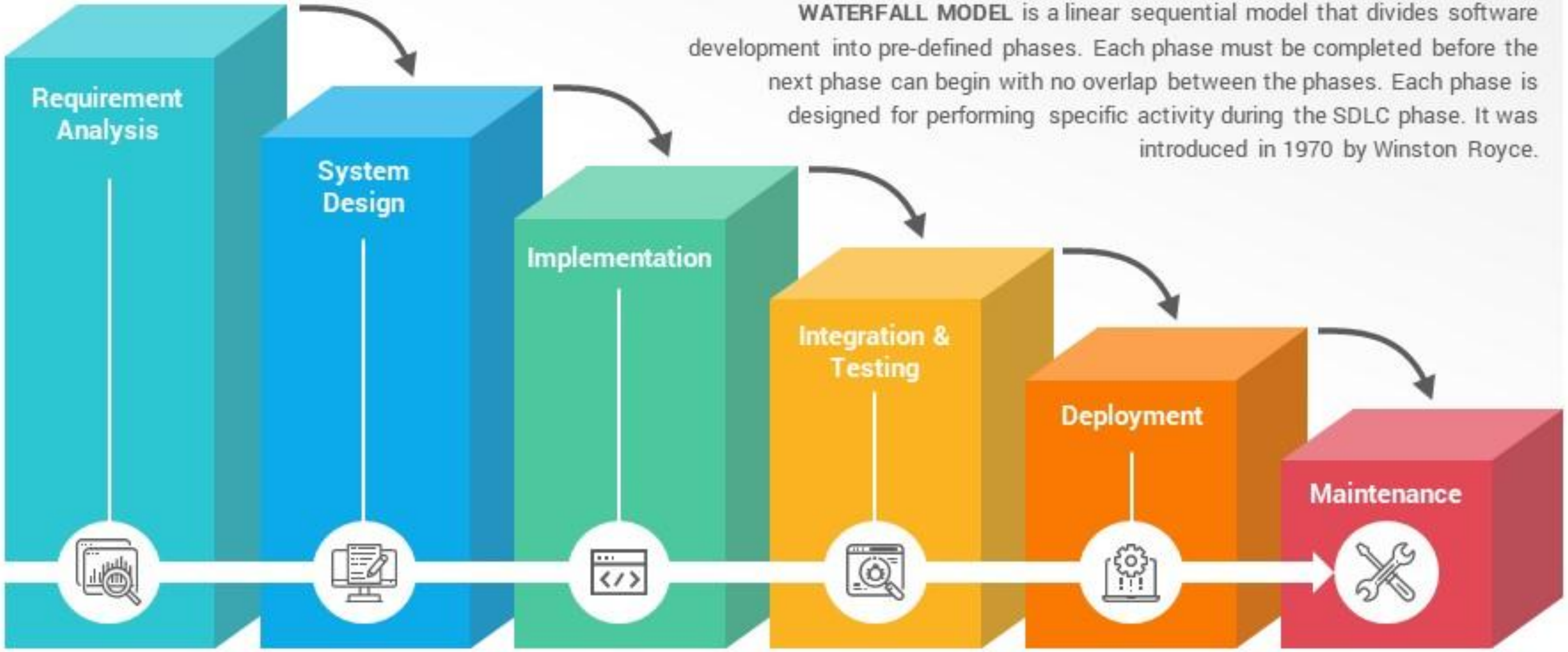
Advantages	Disadvantages
● Fast to start with minimal planning—suitable for very small or individual projects. ● Can produce a working product quickly when requirements are clear and simple.	● Limited scalability; challenging to maintain as project complexity increases. ● Often results in poor-quality code with many bugs. ● High risk of errors due to lack of planning, testing, and documentation.

When to Use:
Rarely recommended; mainly useful for small, personal projects or quick fixes.

What is The Waterfall Model?

Waterfall Model - SDLC

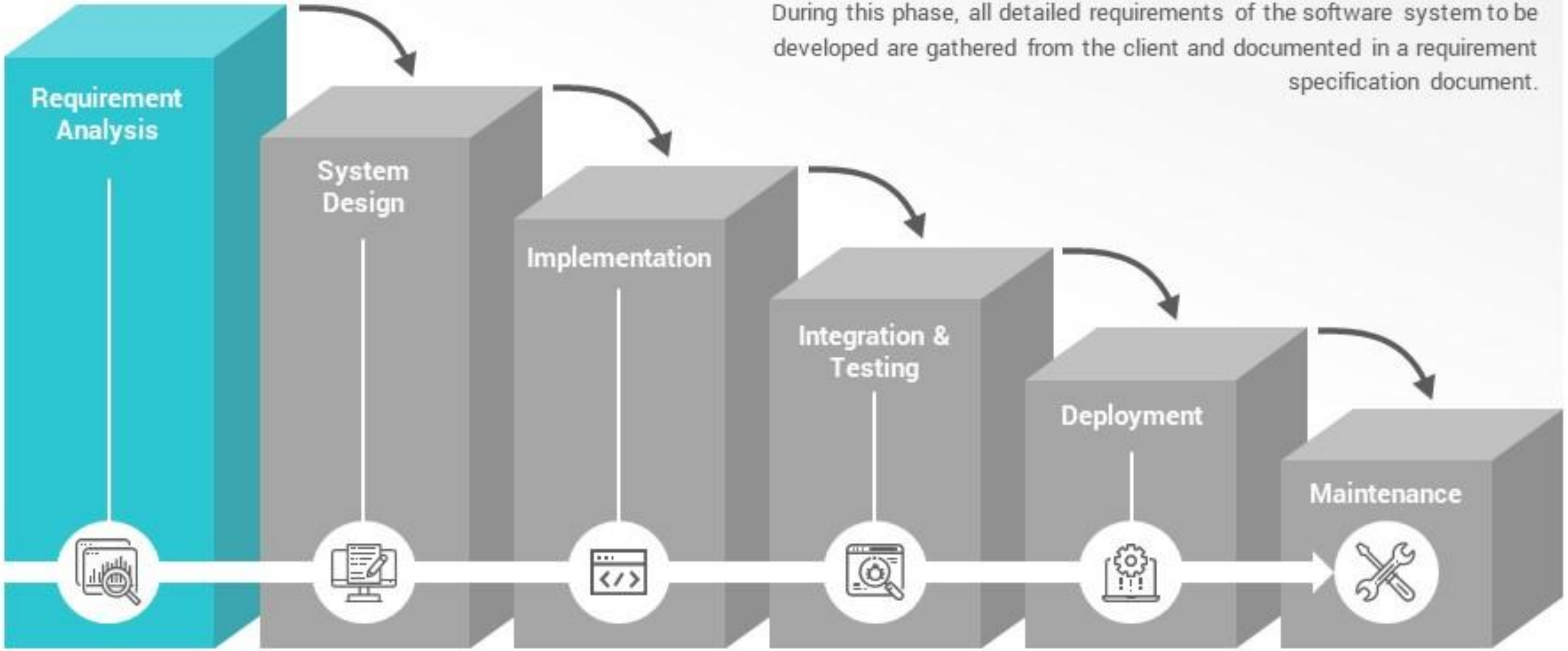
WATERFALL MODEL is a linear sequential model that divides software development into pre-defined phases. Each phase must be completed before the next phase can begin with no overlap between the phases. Each phase is designed for performing specific activity during the SDLC phase. It was introduced in 1970 by Winston Royce.



Waterfall Model – Requirement Analysis Stage

Waterfall Model - SDLC

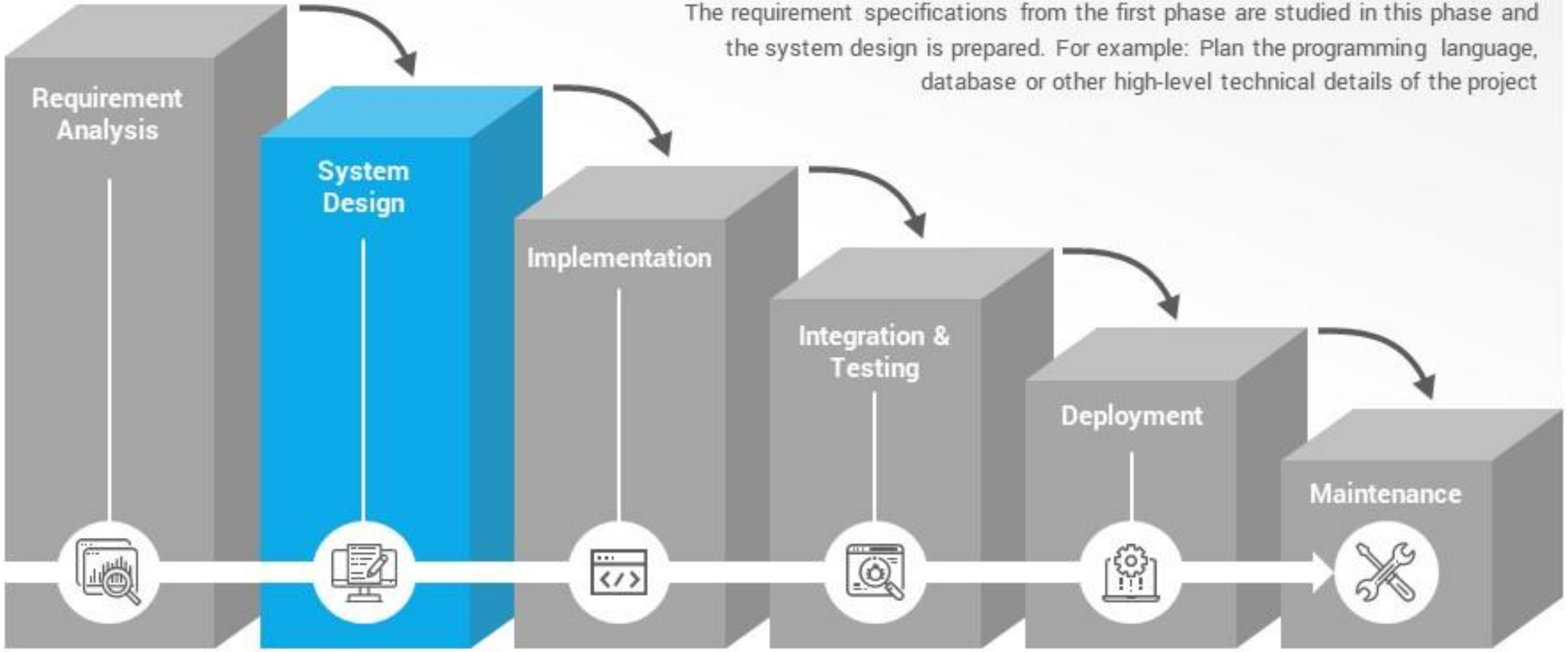
During this phase, all detailed requirements of the software system to be developed are gathered from the client and documented in a requirement specification document.



Waterfall Model – System Design Stage

Waterfall Model - SDLC

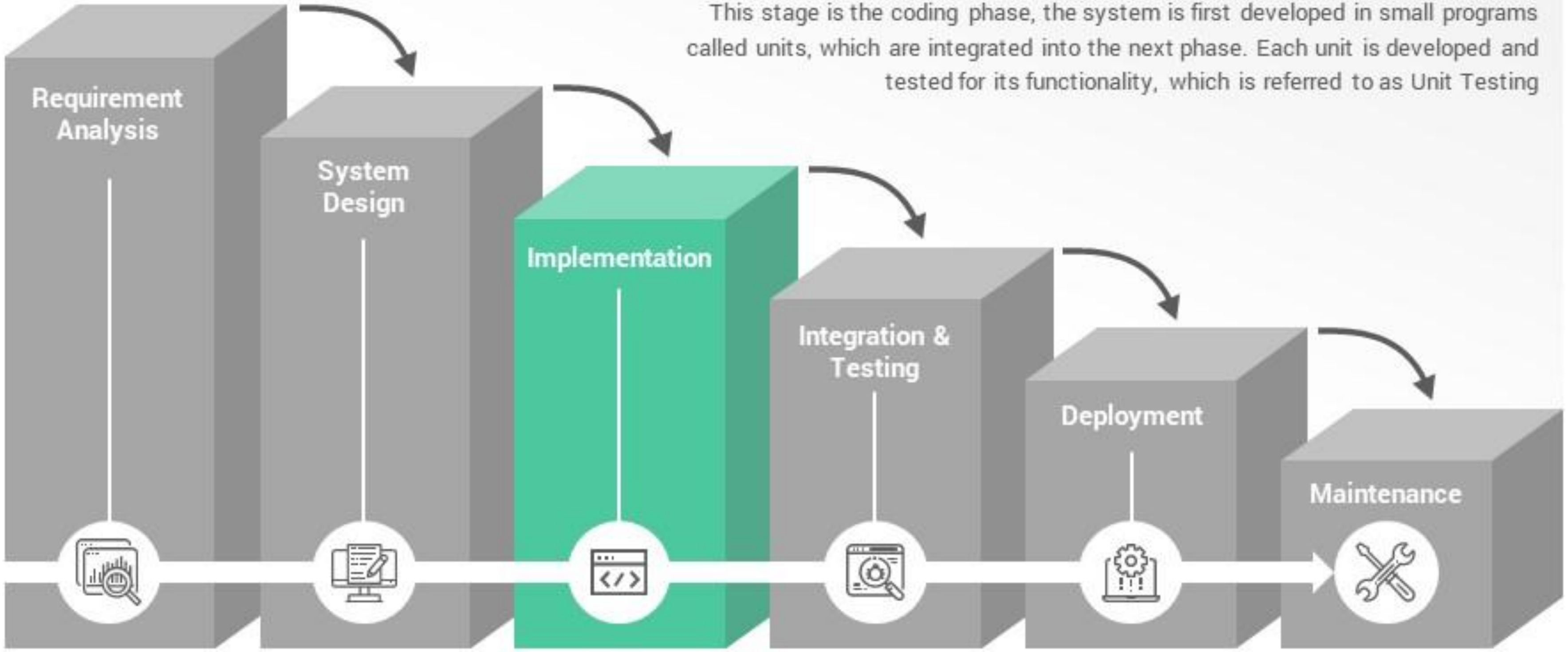
The requirement specifications from the first phase are studied in this phase and the system design is prepared. For example: Plan the programming language, database or other high-level technical details of the project



Waterfall Model – Implementation Stage

Waterfall Model - SDLC

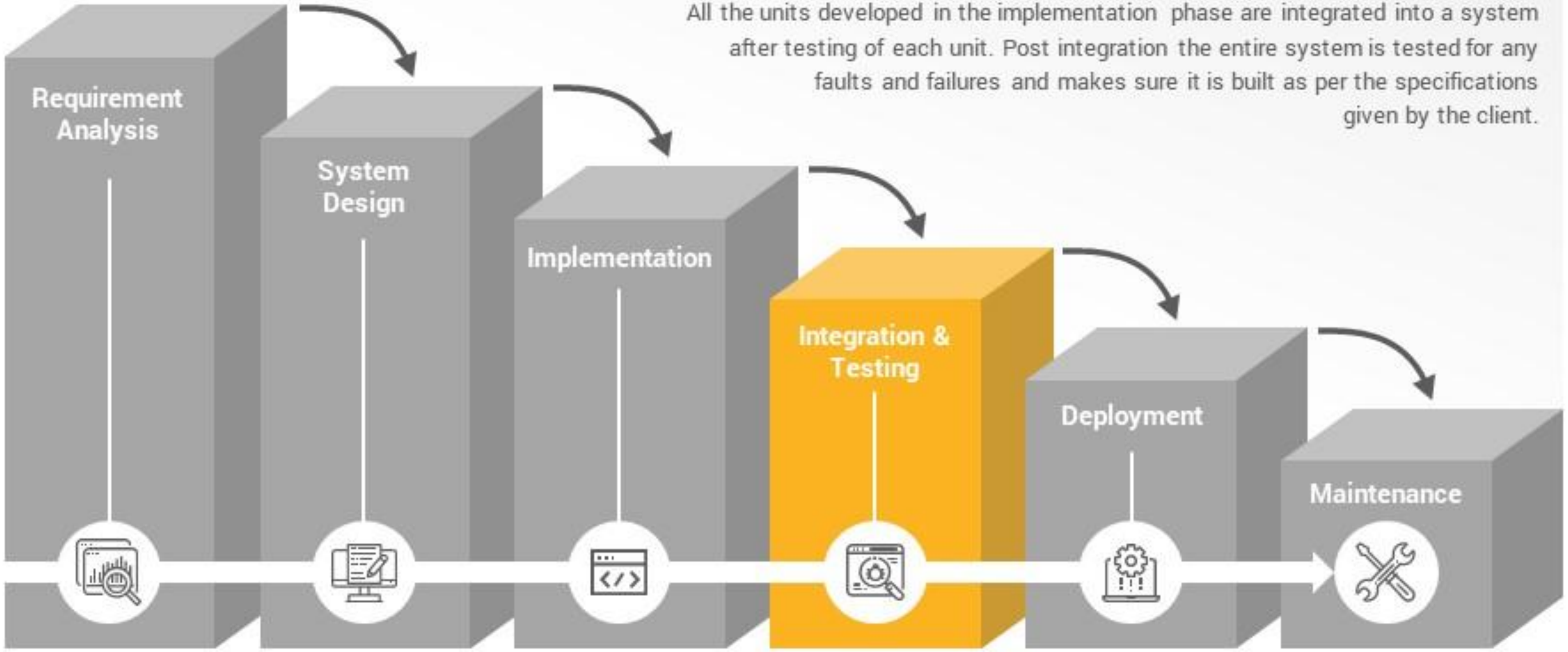
This stage is the coding phase, the system is first developed in small programs called units, which are integrated into the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing



Waterfall Model – Testing Stage

Waterfall Model - SDLC

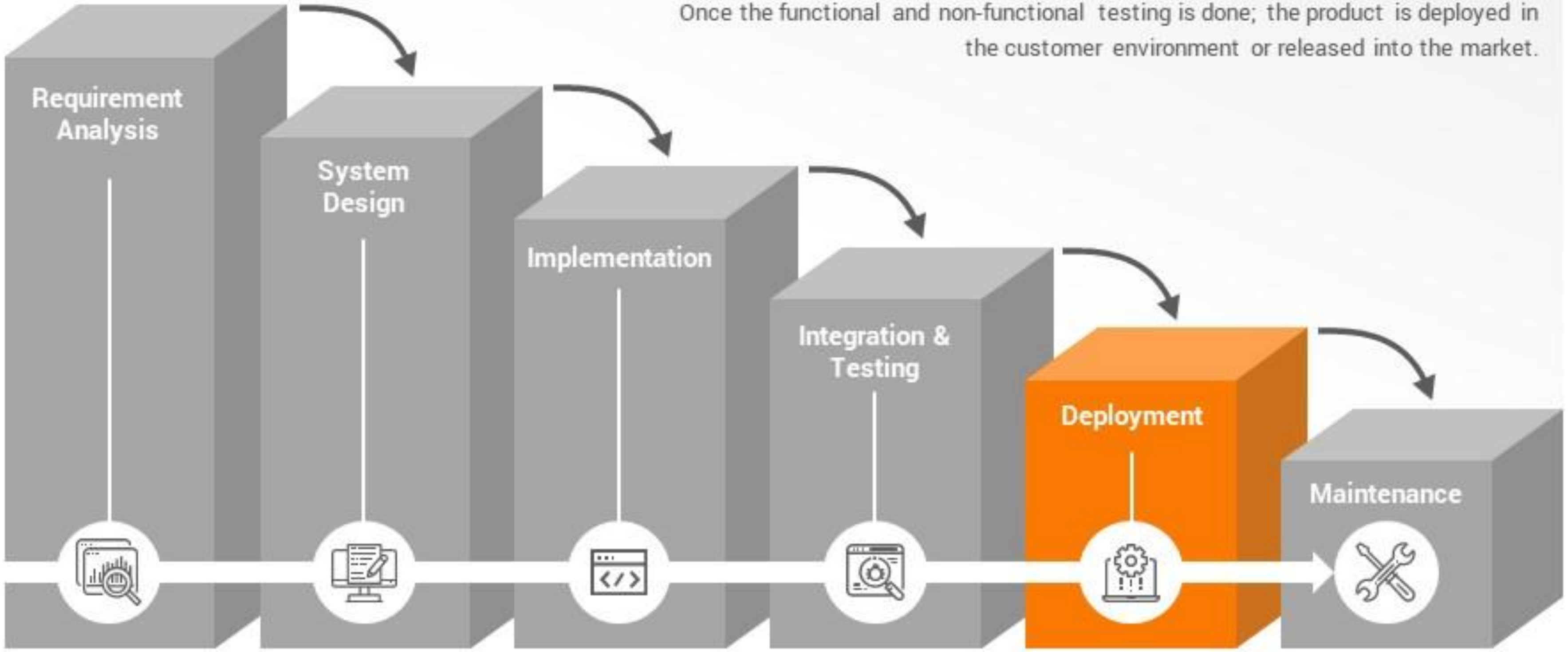
All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures and makes sure it is built as per the specifications given by the client.



Waterfall Model – Deployment Stage

Waterfall Model - SDLC

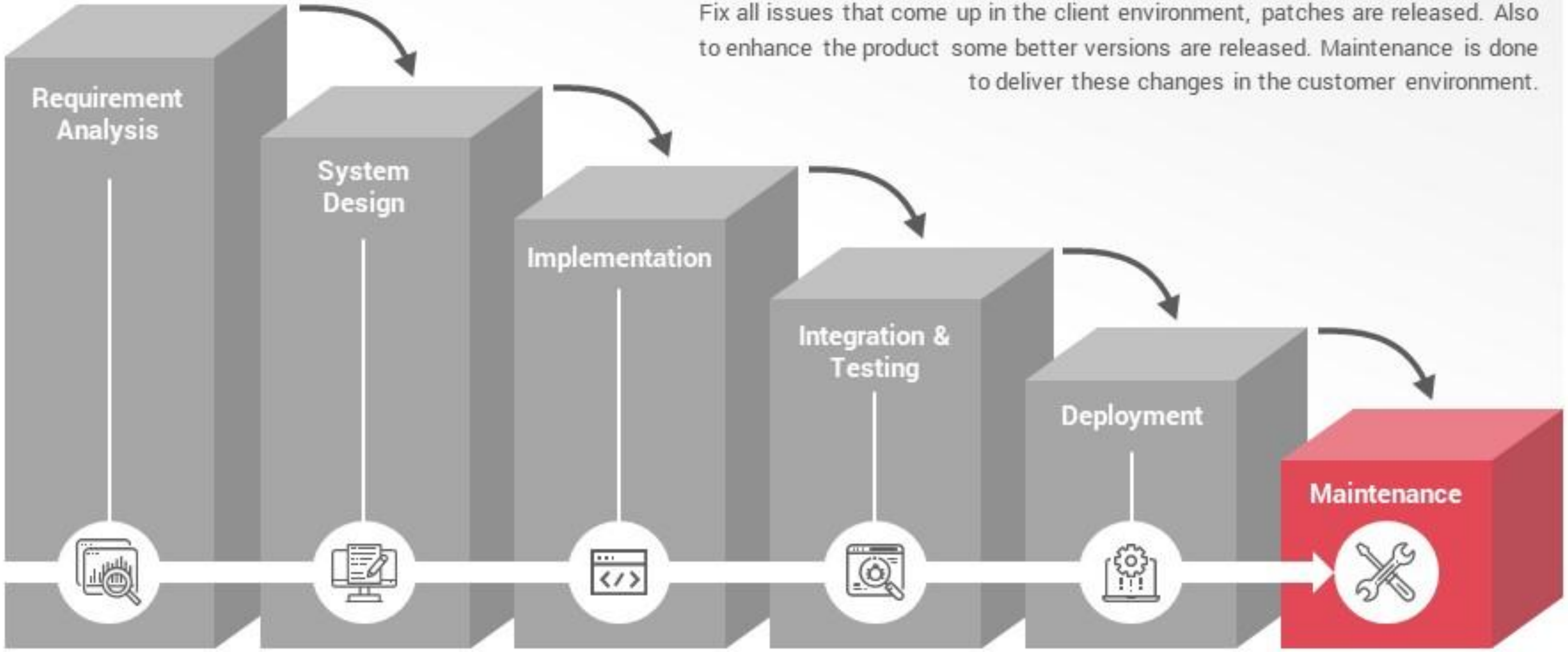
Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.



Waterfall Model – Maintenance Stage

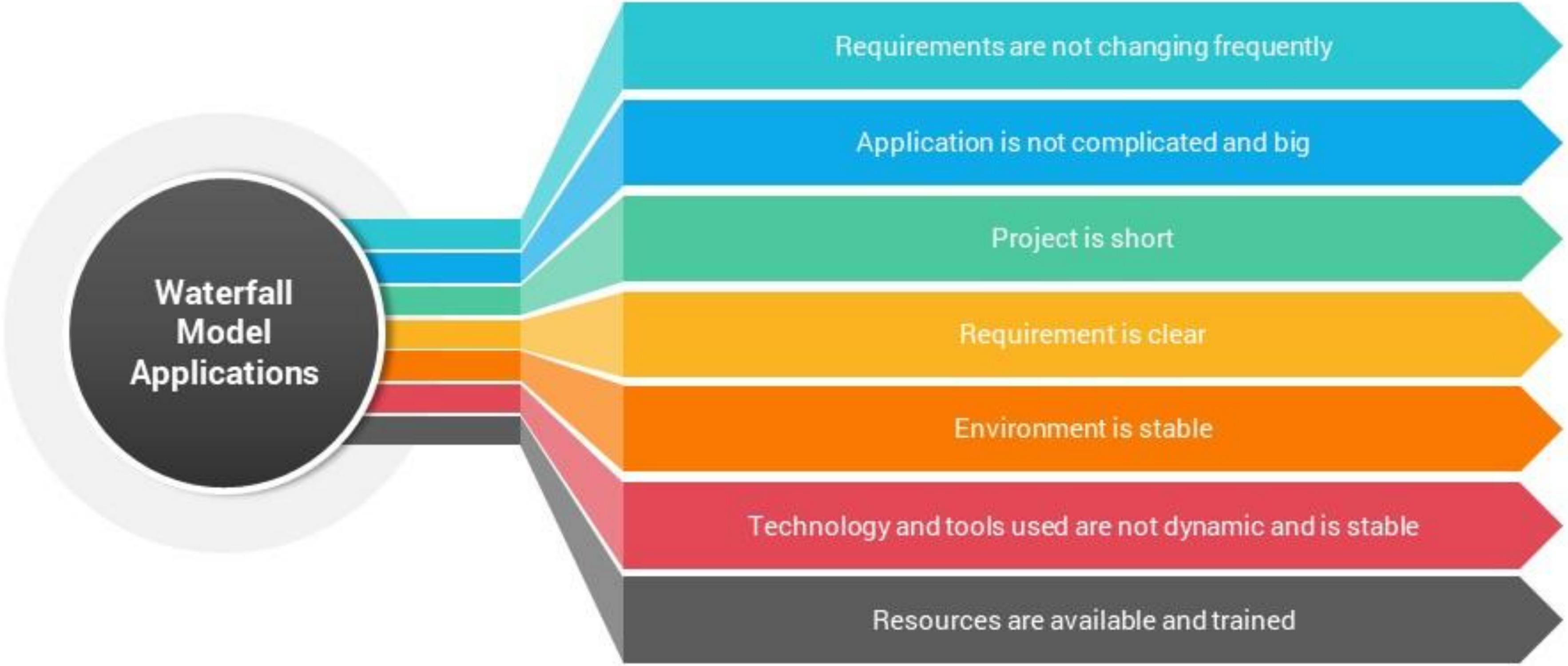
Waterfall Model - SDLC

Fix all issues that come up in the client environment, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.



Waterfall Model - Application

Some situations where the use of the Waterfall model is most appropriate are:

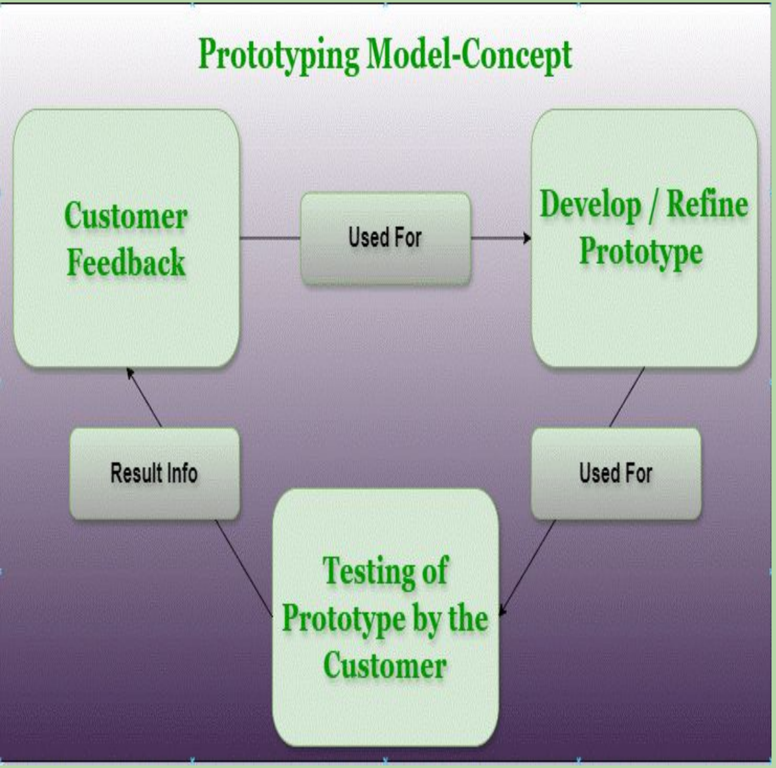

Waterfall Model Applications
Requirements are not changing frequently
Application is not complicated and big
Project is short
Requirement is clear
Environment is stable
Technology and tools used are not dynamic and is stable
Resources are available and trained

Advantages & Disadvantages of Waterfall Model

Advantages & Disadvantages of Waterfall Model

Advantages of Waterfall Model	Disadvantages of Waterfall Model
Simple and easy to understand and use	No working software is produced until late during the life cycle.
Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.	High amounts of risk and uncertainty.
Phases are processed and completed one at a time.	Not a desirable model for complex and object-oriented projects.
Works well for smaller projects where requirements are very well understood.	Poor model for long and ongoing projects.
Clearly defined stages.	It is difficult to measure progress within stages.
Well understood milestones.	Clients feedback cannot be included with the ongoing development phase.
Easy to arrange tasks.	It cannot accommodate changing requirements.
Process and results are well documented.	Small changes or errors that arise in the completed software may cause a lot of problems

Prototyping Model


Prototyping Model-Concept
Customer Feedback
Used For
Develop / Refine Prototype
Used For
Testing of Prototype by the Customer
Result Info

- ❖ The Prototyping Model is an SDLC approach used when project requirements are unclear.
- ❖ An initial, partial prototype is developed and refined through customer feedback until it meets expectations, providing early visibility into the product.
- ❖ This iterative process continues until a final, approved prototype forms the basis of the end product.

Advantages	Disadvantages
● Reduces risk by validating requirements early and allowing for feedback. ● Helps to clarify ambiguous requirements, ensuring the final product better meets user needs. ● Useful for projects with high user interaction or unclear initial requirements.	● Can be time-consuming and costly due to the extra development of the prototype. ● Risk of “prototype creep,” where users continuously request changes, delaying the project. ● Potentially increases project cost and complexity if overused.

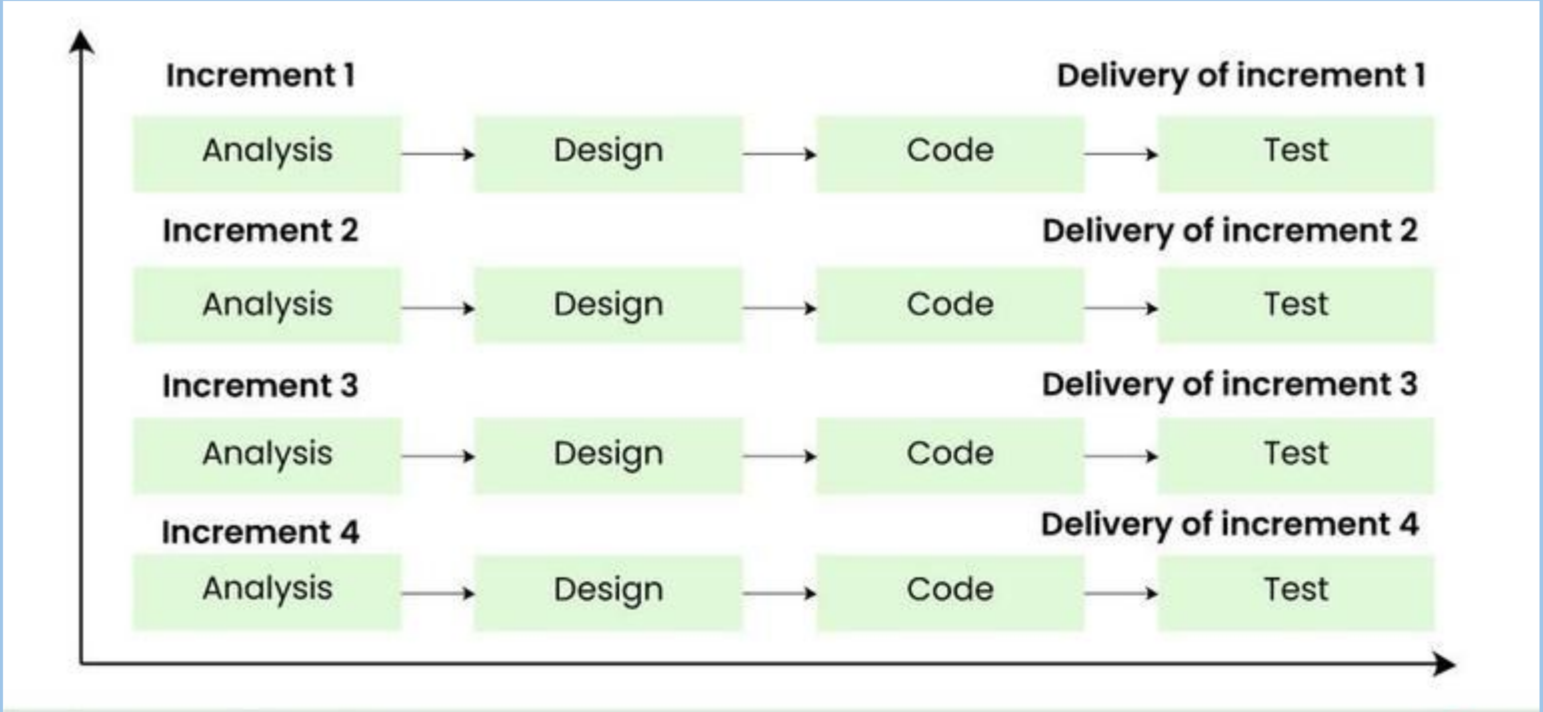
When to Use:

Suitable for projects with unclear requirements or where user feedback is essential to the development process (e.g., UI/UX-intensive projects).

Iterative Enhancement Model

- The Iterative Enhancement Model is a flexible software development approach that emphasizes continuous evolution through ongoing improvement.
- It segments development into manageable parts, allowing for easy adaptation to changing requirements, user needs, and market demands.
- Each iteration builds upon the previous one, adding new features and resolving issues, with ongoing collaboration between developers, stakeholders, and end-users to ensure the product meets evolving expectations.

Iterative Enhancement Model



Life-Cycle of Iterative Enhancement Model

Advantages

- Allows changes to be made after each iteration, making it more flexible.
- Reduces risk by delivering partially functional versions to users for feedback.
- Good for projects where requirements evolve over time.

Disadvantages

- Can increase costs and timeline if too many iterations are required.
- Requires good documentation to keep track of changes.
- There’s a risk of scope creep if the requirements continue to change without limits.

When to Use:

Ideal for medium-to-large projects where requirements are not fully known initially or likely to evolve over time (e.g., web applications).

Spiral Model

The Spiral Model is a software development methodology that combines iterative development with elements of risk management. It is particularly suited for complex and large-scale projects where risk assessment is critical.

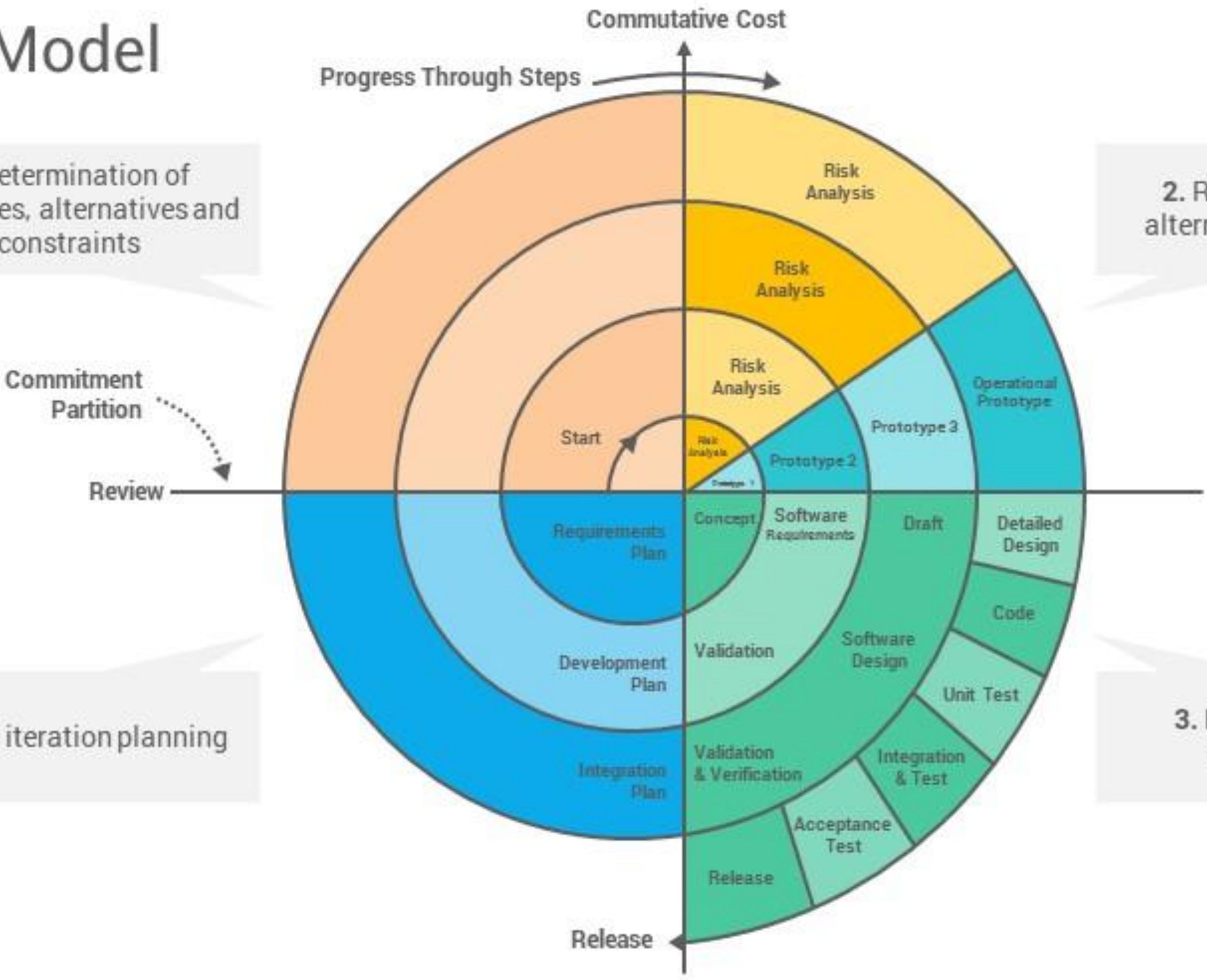
Spiral Model

1. Determination of objectives, alternatives and constraints

2. Risk analysis and alternatives evaluation

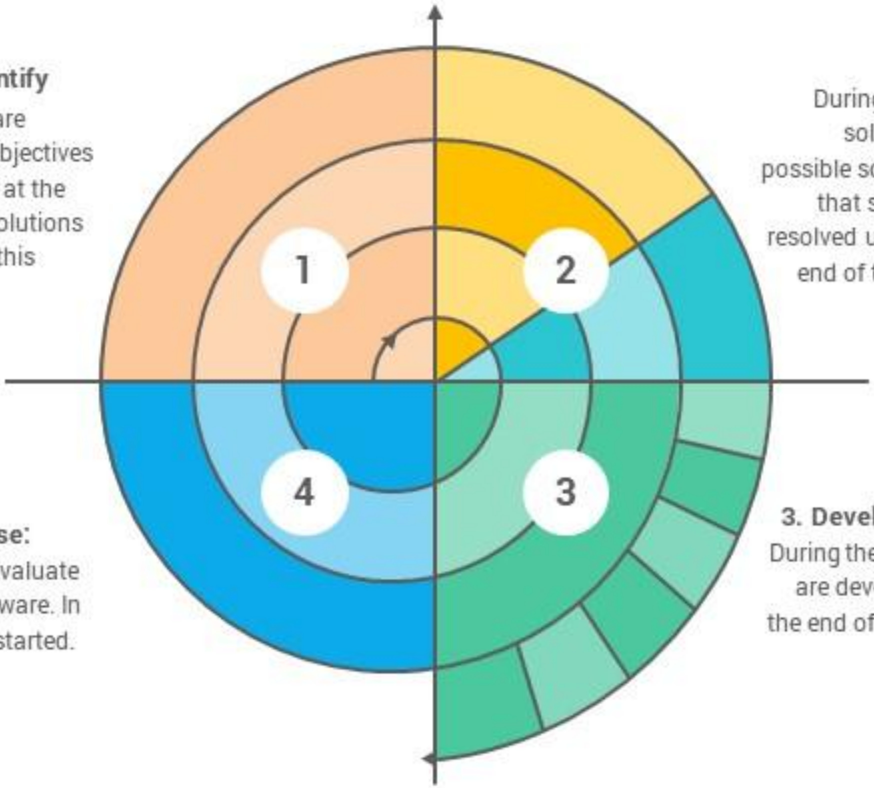
3. Development of deliverables

4. Next iteration planning



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



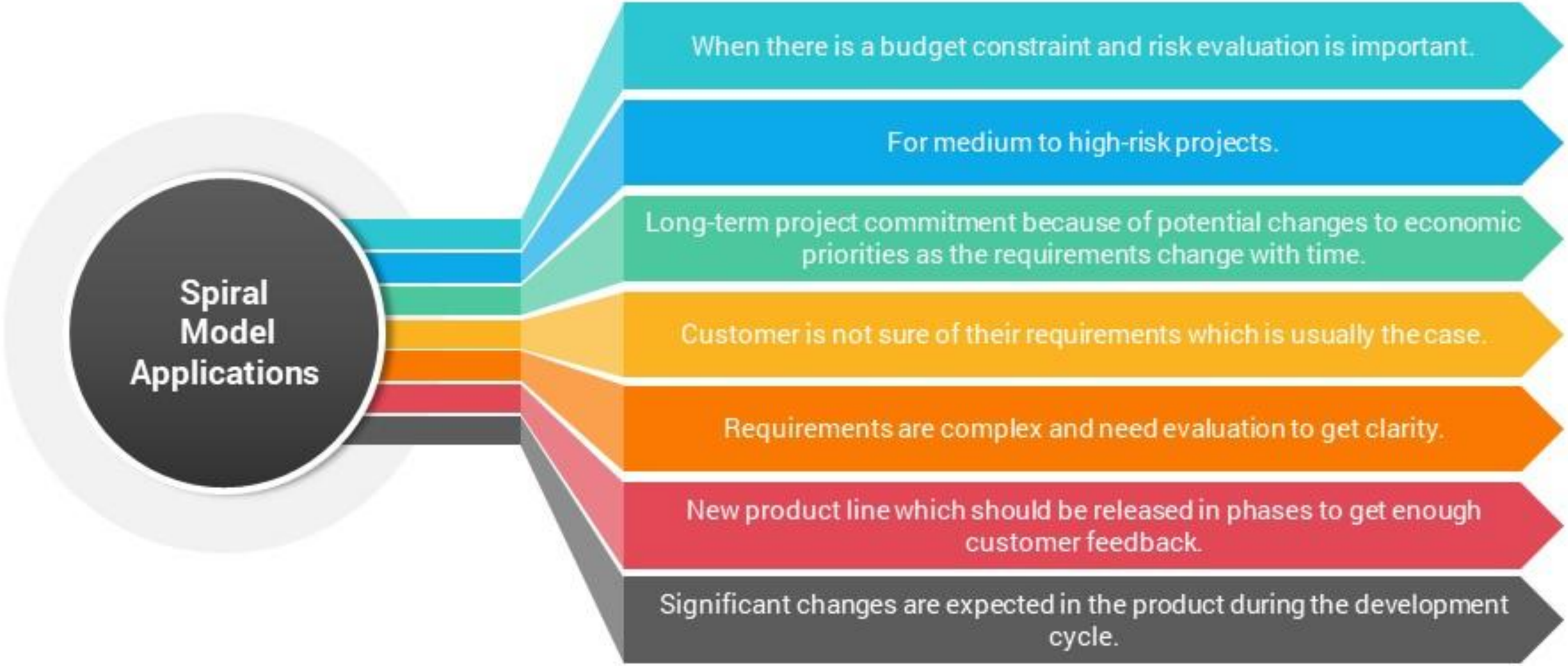
2. Identify and resolve Risks: During the second quadrant, all the possible solutions are evaluated to select the best possible solution. Then the risks associated with that solution are identified and the risks are resolved using the best possible strategy. At the end of this quadrant, Prototype is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

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.

Spiral Model Application

The following pointers explain the typical uses of a Spiral Model:



- When there is a budget constraint and risk evaluation is important.
- For medium to high-risk projects.
- Long-term project commitment because of potential changes to economic priorities as the requirements change with time.
- Customer is not sure of their requirements which is usually the case.
- Requirements are complex and need evaluation to get clarity.
- New product line which should be released in phases to get enough customer feedback.
- Significant changes are expected in the product during the development cycle.

Advantages & Disadvantages of Spiral Model

Advantages & Disadvantages of Spiral Model

Advantages of Spiral Model

- Changing requirements can be accommodated.
- Allows extensive use of prototypes.
- Requirements can be captured more accurately.
- Users see the system early.
- Development can be divided into smaller parts and the risky parts can be developed earlier which helps in better risk management.

Disadvantages of Spiral Model

- Management is more complex.
- End of the project may not be known early.
- Not suitable for small or low-risk projects and could be expensive for small projects.
- Process is complex
- Spiral may go on indefinitely.
- Large number of intermediate stages require excessive documentation.

Rapid Application Development (RAD)

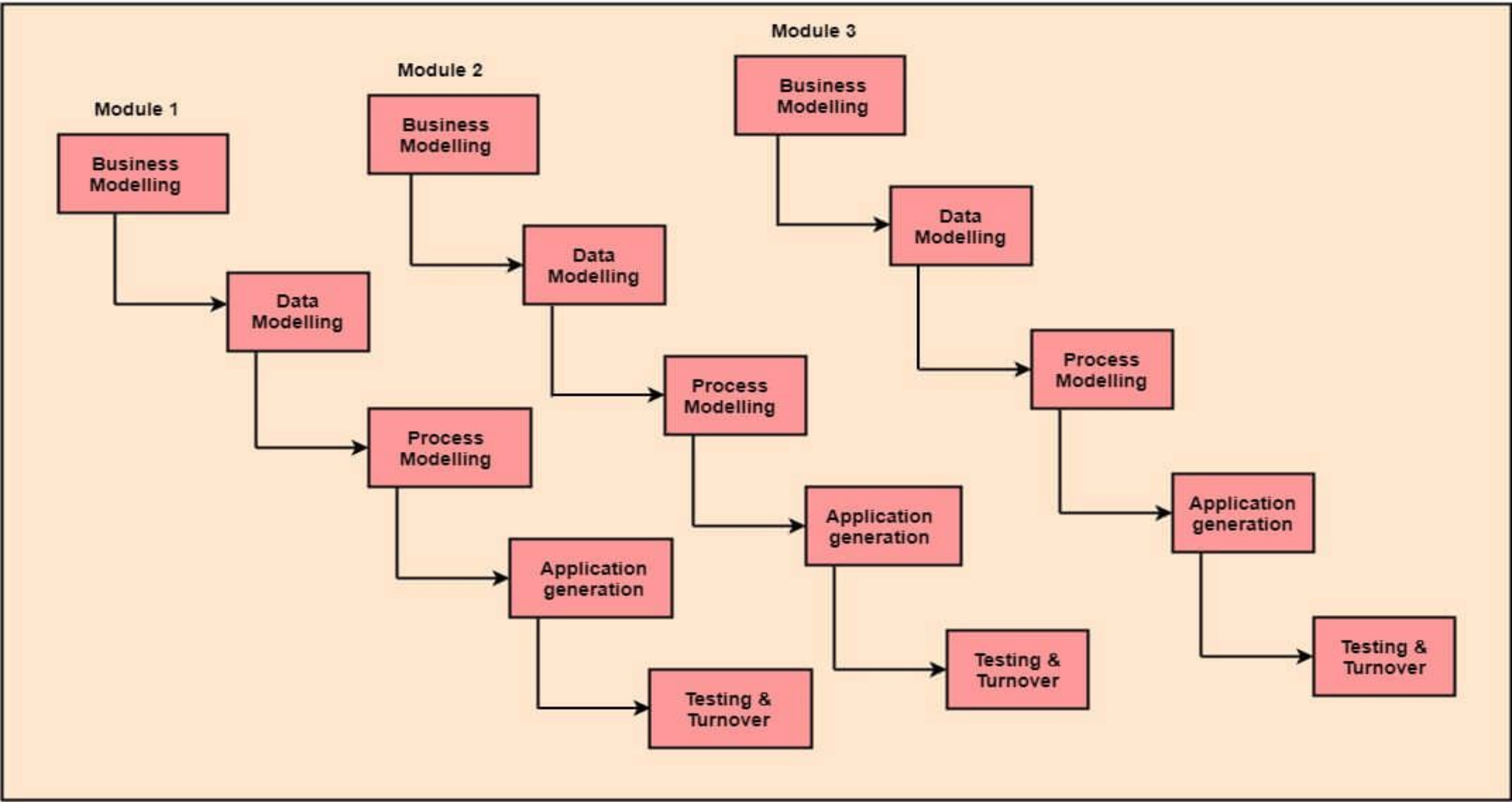
The **RAD (Rapid Application Development)** model is based on prototyping and iterative development with no specific planning involved. The process of writing the software itself involves the planning required for developing the product.

Rapid Application Development focuses on gathering customer requirements through workshops or focus groups, early testing of the prototypes by the customer using the iterative concept, reuse of the existing prototypes (components), continuous integration, and rapid delivery.

The RAD Model phases are:

- **Business Modeling**
- **Data Modeling**
- **Process Modeling**
- **Application Generation**
- **Testing and Turnover**

Fig: RAD Model



RAD Model Application

The following points describe the typical scenarios where RAD can be used:

RAD Model Applications

- RAD should be used only when a system can be modularized to be delivered in an incremental manner.
- It should be used if there is a high availability of designers for modeling.
- It should be used only if the budget permits the use of automated code generating tools.
- RAD SDLC model should be chosen only if domain experts are available with relevant business knowledge.
- Should be used where the requirements change during the project and working prototypes are to be presented to the customer in small iterations of 2-3 months.

Advantages & Disadvantages of RAD

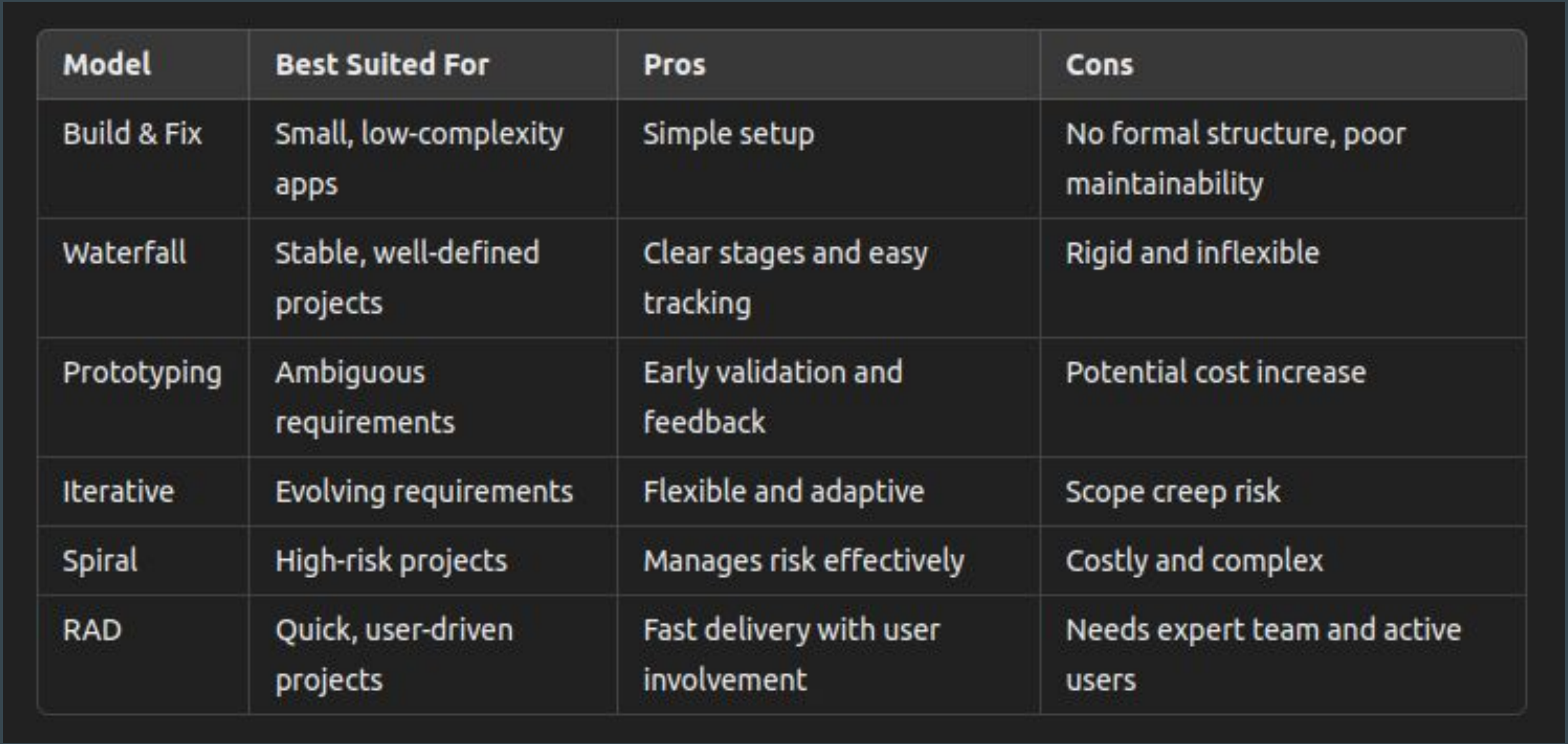
Advantages & Disadvantages of RAD

Advantages of RAD	Disadvantages of RAD
Changing requirements can be accommodated.	Dependency on technically strong team members for identifying business requirements.
Progress can be measured.	Only system that can be modularized can be built using RAD.
Iteration time can be short with the use of powerful RAD tools.	Requires highly skilled developers/designers.
Productivity with fewer people in a short time.	High dependency on modeling skills.
Reduced development time.	Inapplicable to cheaper projects as the cost of modeling and automated code generation is very high.
Increases the reusability of components.	Management complexity is more.
Quick initial reviews occur.	Suitable for systems that are component-based and scalable.
Encourages customer feedback.	Requires user involvement throughout the life cycle.
Integration from the very beginning solves a lot of integration issues.	Suitable for projects requiring shorter development times.

Selection Criteria of a Lifecycle Model

- ❖ **Factors to Consider:**
  - ➢ Project Requirements: Clarity, complexity, and flexibility of requirements.
  - ➢ Project Size and Scope: Is it a small, single-feature project or a large, multi-component project?
  - ➢ Cost and Resources: Budget and resource availability, including skilled team members.
  - ➢ Team Expertise: Availability of experienced developers or project managers.
  - ➢ Time Constraints: Tight deadlines may favor models with faster delivery times.
- ❖ **Decision-Making Tips**
  - ➢ Waterfall: Stable, well-defined projects.
  - ➢ Prototyping: Projects needing detailed user feedback.
  - ➢ Iterative/Spiral: Complex or evolving projects.
  - ➢ RAD: Quick, flexible projects.

Comparing SDLC Models



Model	Best Suited For	Pros	Cons
Build & Fix	Small, low-complexity apps	Simple setup	No formal structure, poor maintainability
Waterfall	Stable, well-defined projects	Clear stages and easy tracking	Rigid and inflexible
Prototyping	Ambiguous requirements	Early validation and feedback	Potential cost increase
Iterative	Evolving requirements	Flexible and adaptive	Scope creep risk
Spiral	High-risk projects	Manages risk effectively	Costly and complex
RAD	Quick, user-driven projects	Fast delivery with user involvement	Needs expert team and active users