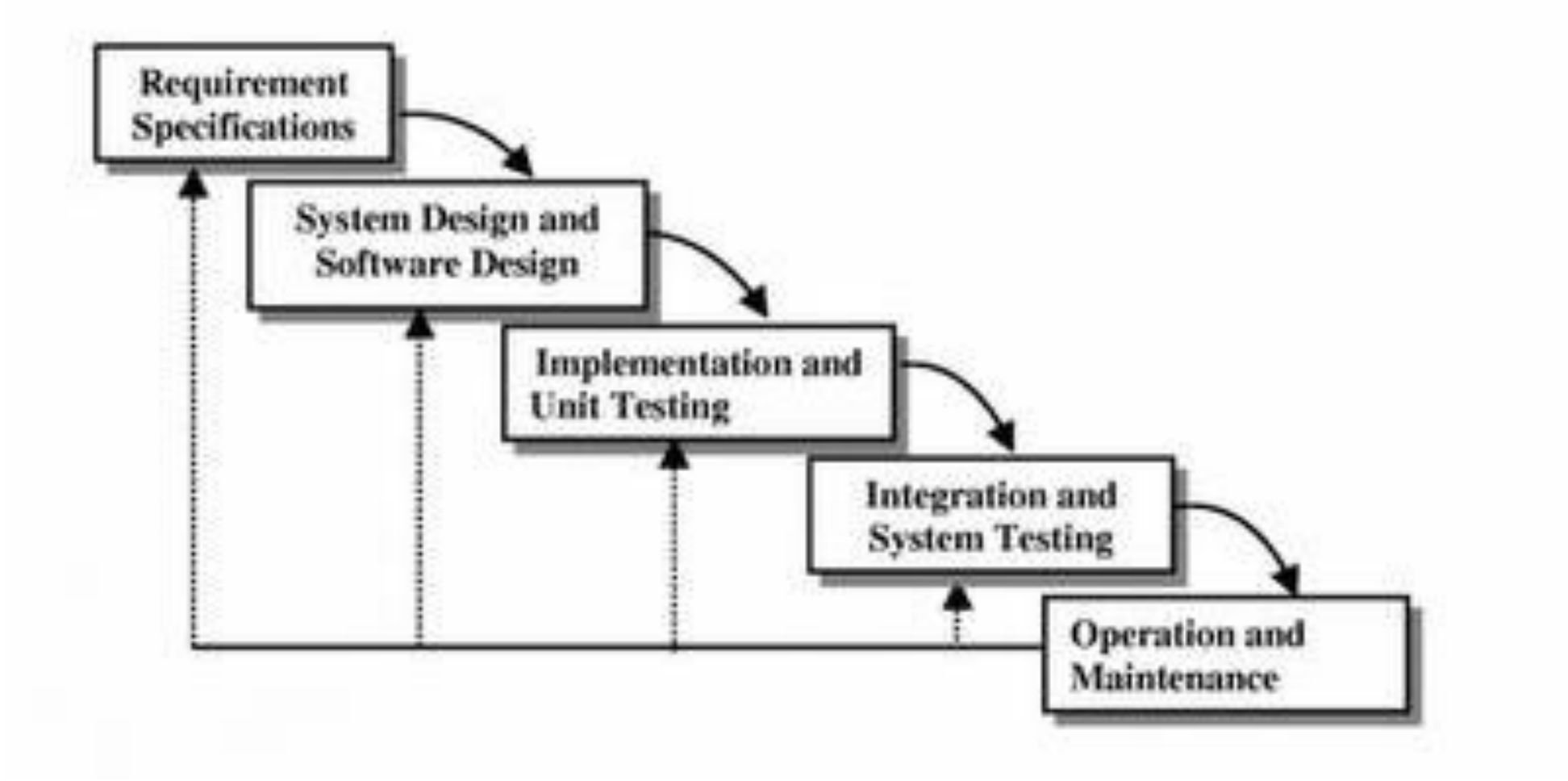
Software Development Life Cycle Model

Iterative Waterfall Model

To overcome the major shortcomings of the classical waterfall model, we come up with the iterative waterfall model.



Iterative Waterfall Model

- Feedback paths are provided for error correction as & when detected later in a phase. Though errors are inevitable, but it is desirable to detect them in the same phase in which they occur. If so, this can reduce the effort to correct the bug.
- The advantage of this model is that there is a working model of the system at a very early stage of development which makes it easier to find functional or design flaws. Finding issues at an early stage of development enables to take corrective measures in a limited budget.
- The disadvantage with this SDLC model is that it is applicable only to large and bulky software development projects. This is because it is hard to break a small software system into further small serviceable increments/modules.

Prototyping Model

- A prototype is a toy implementation of the system.
- A prototype usually exhibits limited functional capabilities, low reliability, and inefficient performance compared to the actual software.
- A prototype is usually built using several shortcuts. The shortcuts might involve using inefficient, inaccurate, or dummy functions. The shortcut implementation of a function. It may produce the desired results by using a table look-up instead of performing the actual computations.
- A prototype usually turns out to be a very crude version of the actual system.
- A prototype of the actual product is preferred in situations such as:
  - User requirements are not complete
  - Technical issues are not clear

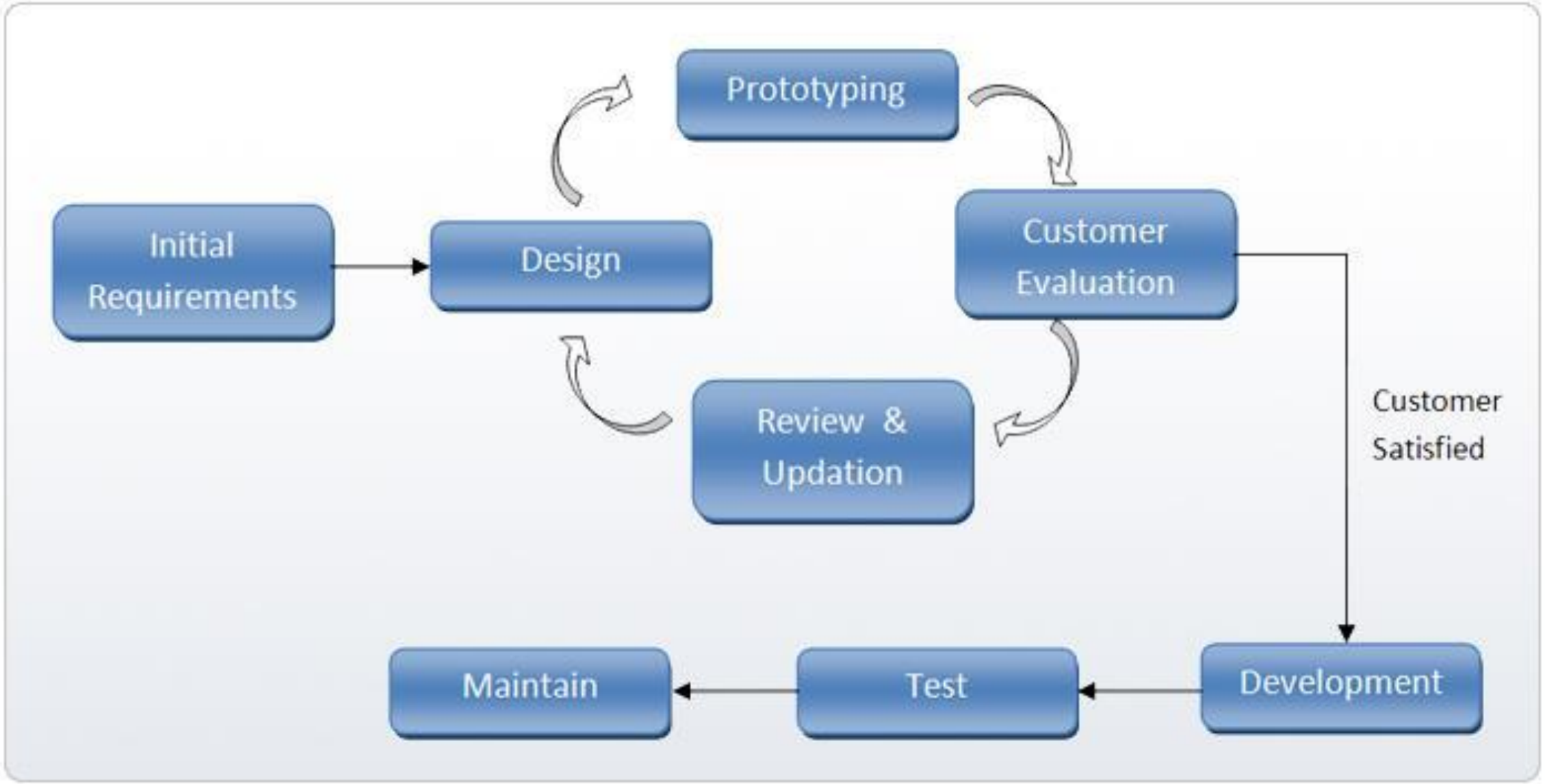
Prototyping Model

❑ Need for a prototype in software development

There are several uses of a prototype. An important purpose is to illustrate the input data formats, messages, reports, and the interactive dialogues to the customer. This is a valuable mechanism for gaining better understanding of the customer's needs:

- ➢How the screens might look like
- ➢How the user interface would behave
- ➢How the system would produce outputs

Prototyping Model



Prototyping Model

A reason for developing a prototype is that it is impossible to get the perfect product in the first attempt. Many researchers and engineers advocate that if anyone want to develop a good product he/she must plan to throw away the first version. The experience gained in developing the prototype can be used to develop the final product.

A prototyping model can be used when technical solutions are unclear to the development team. A developed prototype can help engineers to critically examine the technical issues associated with the product development. Often, major design decisions depend on issues like the response time of a hardware controller, or the efficiency of a sorting algorithm, etc. In such circumstances, a prototype may be the best or the only way to resolve the technical issues.

Evolutionary Model

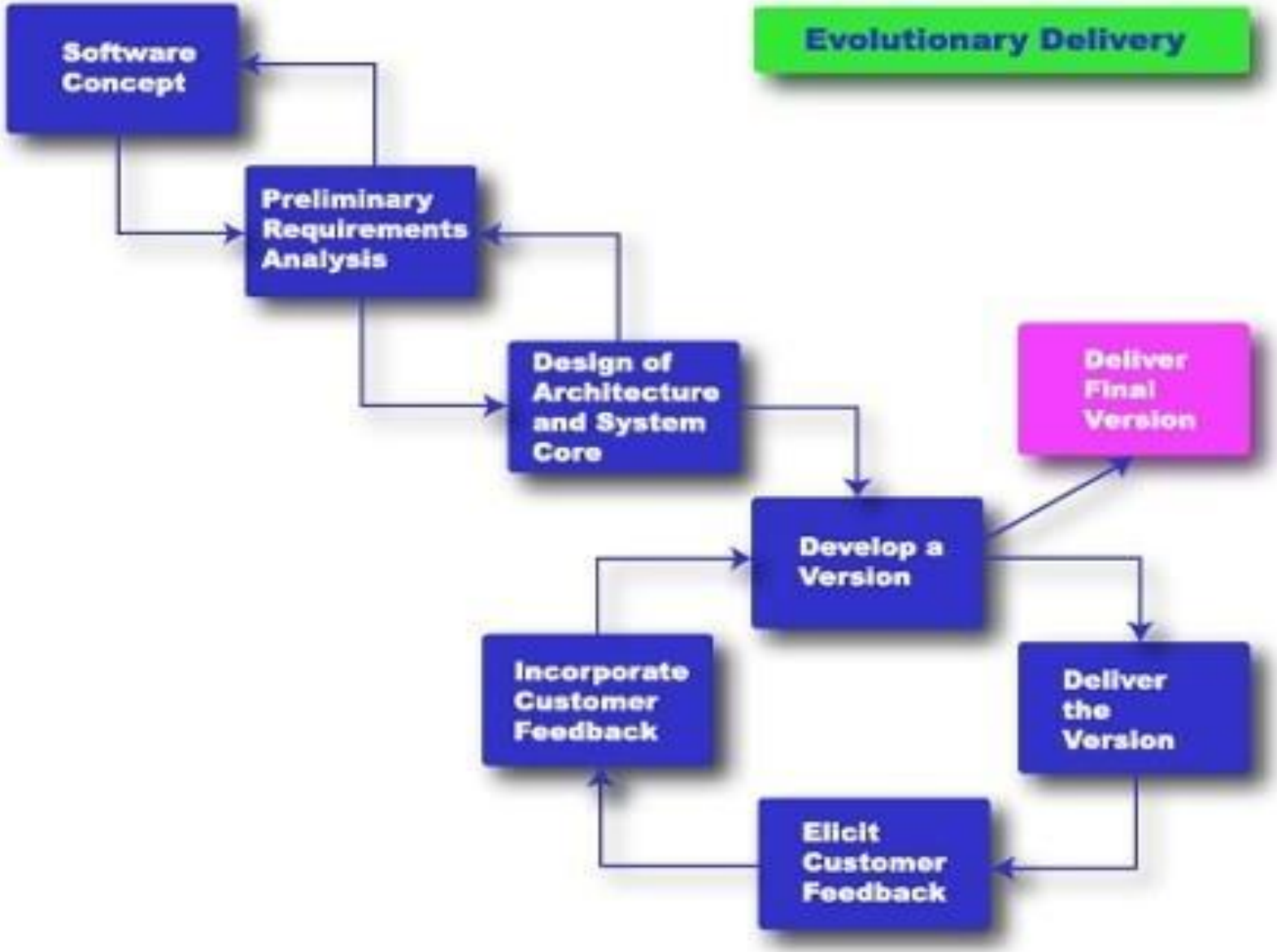
It is also called *successive versions model* or *incremental model*. At first, a simple working model is built. Subsequently it undergoes functional improvements & we keep on adding new functions till the desired system is built.

Applications:

Large projects where you can easily find modules for incremental implementation. Often used when the customer wants to start using the core features rather than waiting for the full software.

Also used in object oriented software development because the system can be easily portioned into units in terms of objects.

Evolutionary Model



Evolutionary Model

Advantages:

- User gets a chance to experiment partially developed system.
- Reduce the error because the core modules get tested thoroughly.

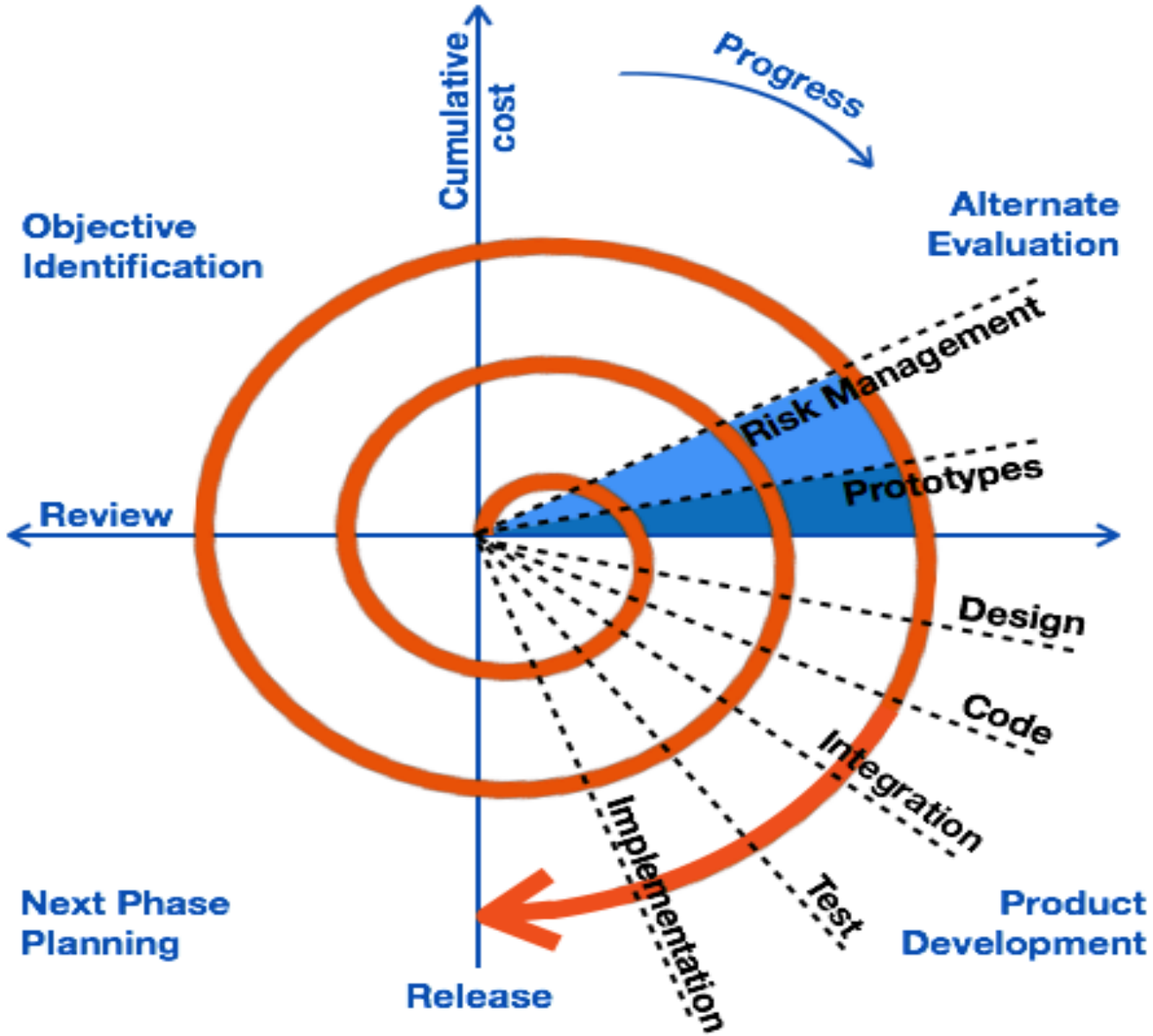
Disadvantages:

- It is difficult to divide the problem into several versions that would be acceptable to the customer which can be incrementally implemented & delivered.

Spiral Model

- The diagrammatic representation of the Spiral model appears like a spiral with many loops.
- The exact number of loops in the spiral is not fixed.
- Each loop of the spiral represents a phase of the software process. For example, the innermost loop might be concerned with feasibility study, the next loop with requirements specification, the next one with design, and so on.
- Each phase in this model is split into four sectors (or quadrants).

Spiral Model



Spiral Model

First quadrant (Objective Setting)

- During the first quadrant, it is needed to identify the objectives of the phase.
- Examine the risks associated with these objectives.

Second Quadrant (Risk Assessment and Reduction)

- A detailed analysis is carried out for each identified project risk.
- Steps are taken to reduce the risks. For example, if there is a risk that the requirements are inappropriate, a prototype system may be developed.

Spiral Model

Third Quadrant (Development and Validation)

- Develop and validate the next level of the product after resolving the identified risks.

Fourth Quadrant (Review and Planning)

- Review the results achieved so far with the customer and plan the next iteration around the spiral.
- Progressively more complete version of the software gets built with each iteration around the spiral.

Spiral Model

Circumstances to use spiral model

The spiral model is called a meta model since it encompasses all other life cycle models. Risk handling is inherently built into this model. The spiral model is suitable for development of technically challenging software products that are prone to several kinds of risks. However, this model is much more complex than the other models – this is probably a factor deterring its use in ordinary projects.

RAD Model

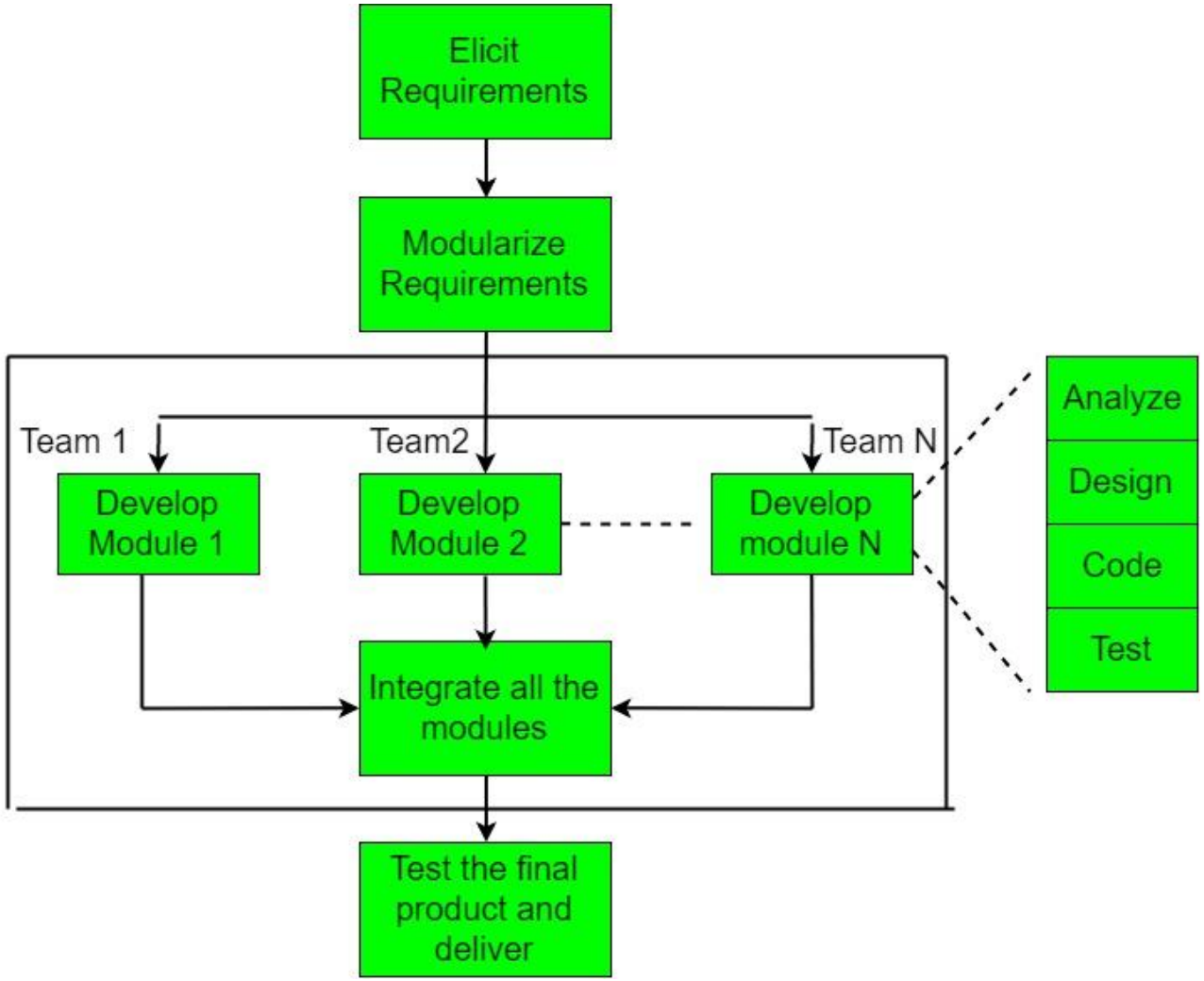
RAD Model

The Rapid Application Development Model was first proposed by IBM in the 1980s. The RAD model is a type of incremental process model in which there is an extremely short development cycle. When the requirements are fully understood and the component-based construction approach is adopted then the RAD model is used.

he critical feature of this model is the use of powerful development tools and techniques. A software project can be implemented using this model if the project can be broken down into small modules wherein each module can be assigned independently to separate teams. Multiple teams work on developing the software system using the RAD model parallelly.

Another striking feature of this model is a short period i.e. the time frame for delivery(time-box) is generally 60-90 days.

RAD Model


Elicit Requirements
Modularize Requirements
Team 1
Team2
Team N
Develop Module 1
Develop Module 2
Develop module N
Analyze
Design
Code
Test
Integrate all the modules
Test the final product and deliver

RAD Model

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.
- **Low technological Complexity:** Suitable for tasks using comparatively straightforward technological specifications.

RAD Model

Objectives of Rapid Application Development Model (RAD)

Speedy Development

Accelerating the software development process is RAD's main goal. RAD prioritizes rapid prototyping and iterations to produce a working system as soon as possible.

Adaptability and Flexibility

RAD places a strong emphasis on adapting quickly to changing needs. Due to the model's flexibility, stakeholders can modify and improve the system in response to changing requirements and user input.

Stakeholder Participation

Throughout the development cycle, RAD promotes end users and stakeholders' active participation.

RAD Model

Objectives of Rapid Application Development Model (RAD)

Improved Interaction

Development teams and stakeholders may collaborate and communicate more effectively. Frequent communication and feedback loops guarantee that all project participants are in agreement, which lowers the possibility of misunderstandings.

Improved Quality via Prototyping

Prototypes enable early system component testing and visualization in Rapid Application Development (RAD).

Customer Satisfaction

Through rapid delivery of functioning prototypes and user involvement throughout the development process, Rapid Application Development (RAD) enhances the probability of customer satisfaction with the final product.

RAD Model

Advantages of Rapid Application Development Model (RAD)

- The use of reusable components helps to reduce the cycle time of the project.
- 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.
- The progress and development of the project can be measured through the various stages.
- It is easier to accommodate changing requirements due to the short iteration time spans.
- Productivity may be quickly boosted with a lower number of employees.

RAD Model

Disadvantages of Rapid application development model (RAD)

- 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.
- It is not meant for small-scale projects as in such cases, the cost of using automated tools and techniques may exceed the entire budget of the project.
- Not every application can be used with RAD.

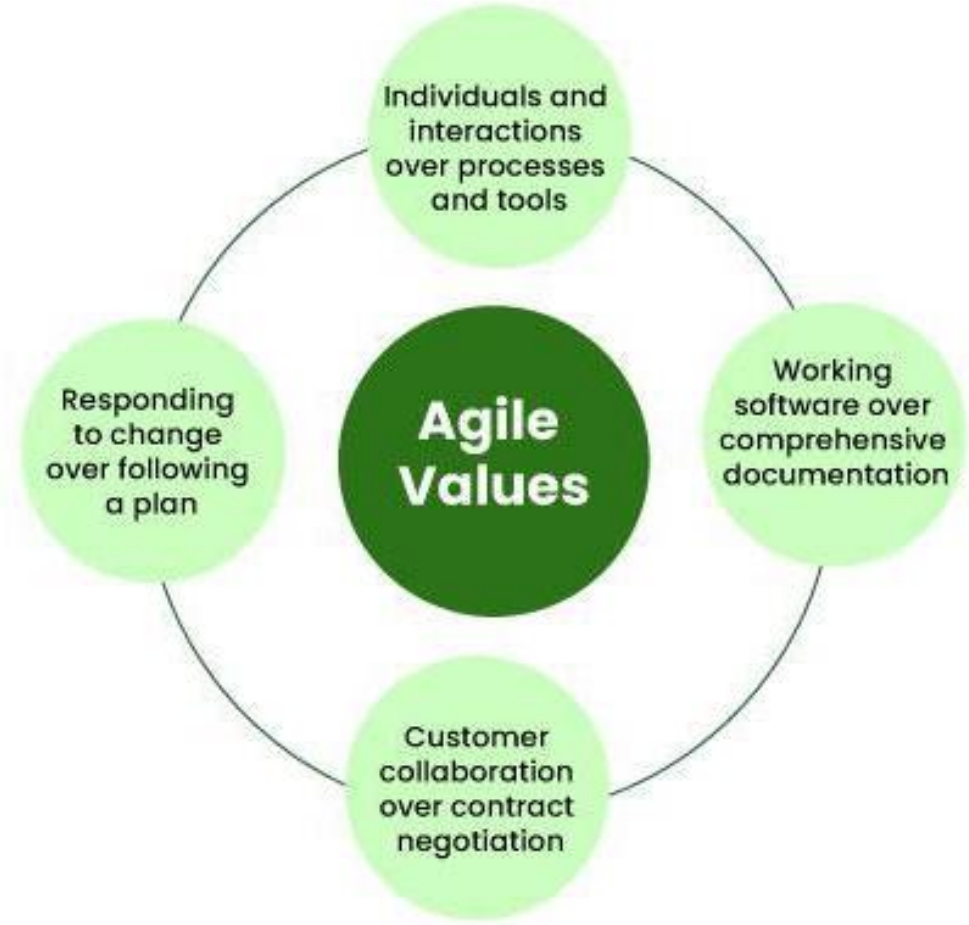
Agile Model

- Agile Software Development is a software development methodology that values flexibility, collaboration, and customer satisfaction.
- It is based on the Agile Manifesto, a set of principles for software development that prioritize individuals and interactions, working software, customer collaboration, and responding to change.
- Agile Software Development is an iterative and incremental approach to software development that emphasizes the importance of delivering a working product quickly and frequently.
- It involves close collaboration between the development team and the customer to ensure that the product meets their needs and expectations.

Agile Model

Four Core Values of Agile Software Development

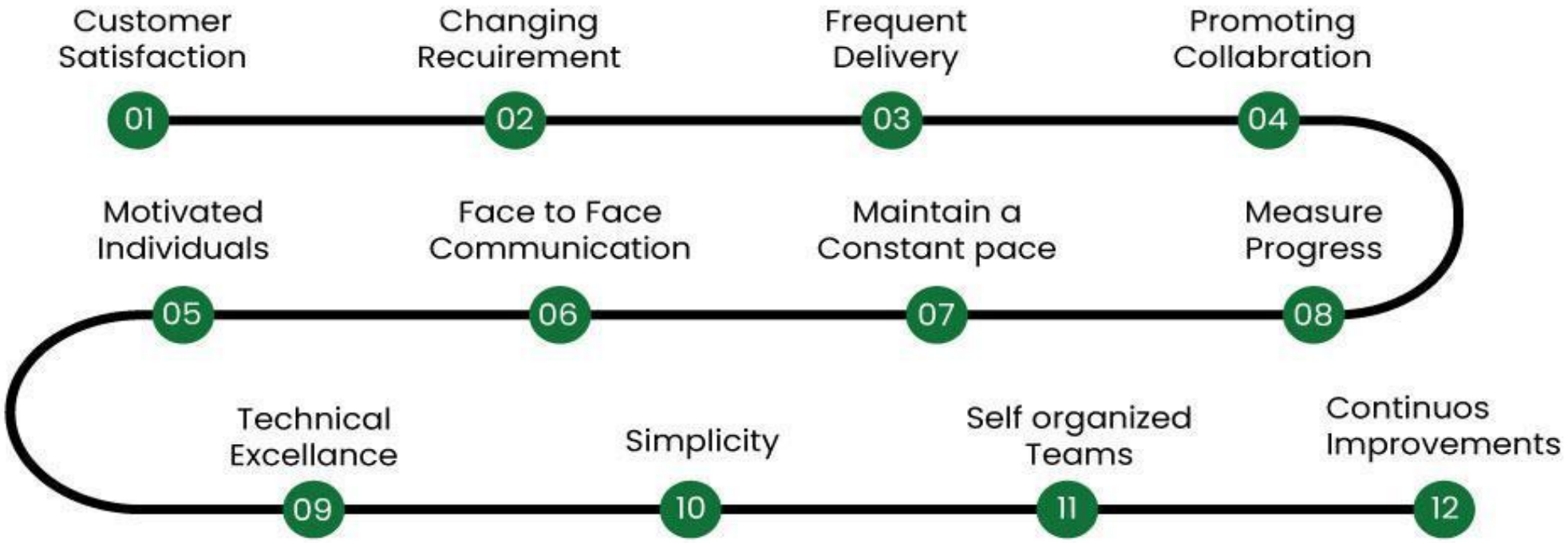
- Individuals and Interactions over Processes and Tools
- Working Software over Comprehensive Documentation
- Customer Collaboration over Contract Negotiation
- Responding to Change over Following a Plan



Agile Model

Twelve Principles of Agile Software Development

- The Agile Manifesto is based on four values and twelve principles that form the basis, for methodologies.



12 Principles of Agile Methodology

Agile Model

These principles include:

1. Ensuring customer satisfaction through the early delivery of software.
2. Being open to changing requirements in the stages of the development.
3. Frequently delivering working software with a main focus on preference for timeframes.
4. Promoting collaboration between business stakeholders and developers as an element.
5. Structuring the projects around individuals. Providing them with the necessary environment and support.
6. Prioritizing face to face communication whenever needed.
7. Considering working software as the measure of the progress.
8. Fostering development by allowing teams to maintain a pace indefinitely.
9. Placing attention on excellence and good design practices.
10. Recognizing the simplicity as crucial factor aiming to maximize productivity by minimizing the work.
11. Encouraging self organizing teams as the approach to design and build systems.
12. Regularly reflecting on how to enhance effectiveness and to make adjustments accordingly.

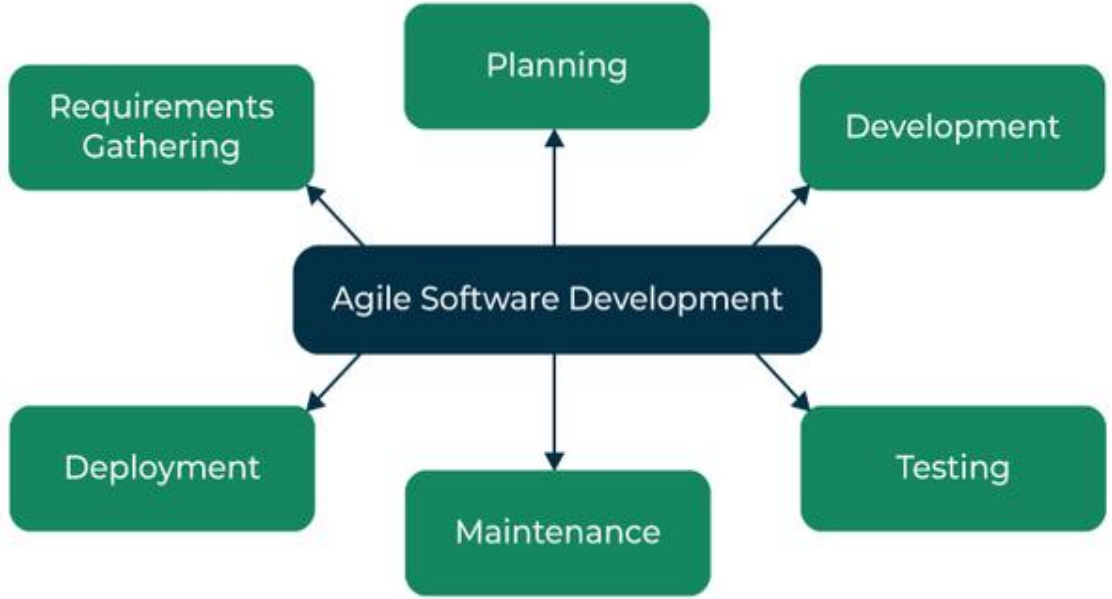
Agile Model

The Agile Software Development Process

Requirements Gathering: The customer's requirements for the software are gathered and prioritized.

Planning: The development team creates a plan for delivering the software, including the features that will be delivered in each iteration.

Development: The development team works to build the software, using frequent and rapid iterations.



Testing: The software is thoroughly tested to ensure that it meets the customer's requirements and is of high quality.

Deployment: The software is deployed and put into use.

Maintenance: The software is maintained to ensure that it continues to meet the customer's needs and expectations.

Agile Model

Advantages Agile Software Development

- Deployment of software is quicker and thus helps in increasing the trust of the customer.
- Can better adapt to rapidly changing requirements and respond faster.
- Helps in getting immediate feedback which can be used to improve the software in the next increment.
- People – Not Process. People and interactions are given a higher priority than processes and tools.
- Continuous attention to technical excellence and good design.
- Agile Software Development Methodology emphasize collaboration and communication among team members, stakeholders, and customers. This leads to improved understanding, better alignment, and increased buy-in from everyone involved.

Agile Model

Advantages Agile Software Development

- Agile methodologies are designed to be flexible and adaptable, making it easier to respond to changes in requirements, priorities, or market conditions. This allows teams to quickly adjust their approach and stay focused on delivering value.
- Agile methodologies place a strong emphasis on testing, quality assurance, and continuous improvement. This helps to ensure that software is delivered with high quality and reliability, reducing the risk of defects or issues that can impact the user experience.
- Agile methodologies prioritize customer satisfaction and focus on delivering value to the customer. By involving customers throughout the development process, teams can ensure that the software meets their needs and expectations.
- Agile methodologies promote a collaborative, supportive, and positive work environment. This can lead to increased team morale, motivation, and engagement, which can in turn lead to better productivity, higher quality work, and improved outcomes.