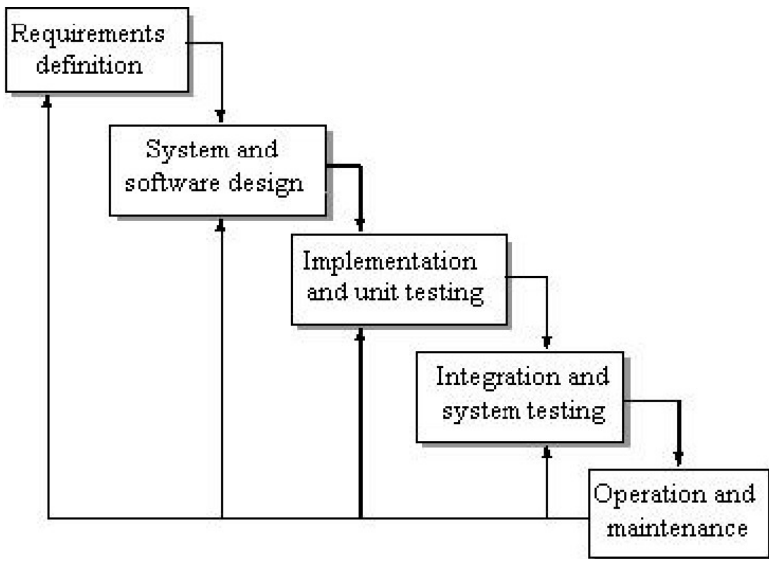
# Slide 03 : Software Development Life Cycle

***Made by Md. Mehedi Hasan Rafy***

## 2. *Iterative Waterfall Model*

An enhancement of the classical waterfall model where feedback paths are provided between phases for error correction. Errors detected later can be corrected by revisiting earlier phases. This can reduce the effort to correct the bug.


Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

***Advantages:***

- A working model is available at a very early stage of the development process, making it easier to identify functional or design flaws.
- Early issue detection enables to take corrective measures in a limited budget.

***Disadvantages:***

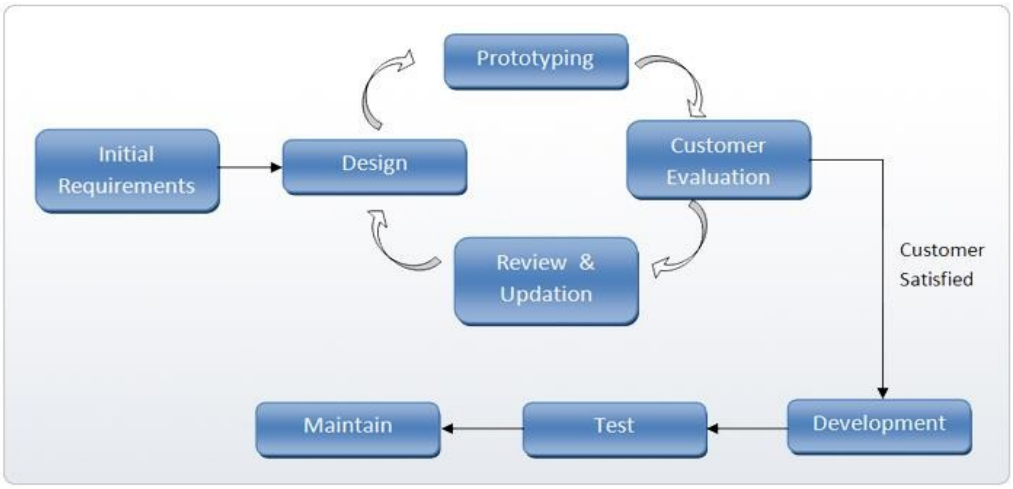
- Effective mostly for large, complex software projects.
- For small projects, breaking the system into incremental deliverables is impractical.

## 2. ***Prototyping Model***

A simplified or “toy” version of the final product is built to clarify requirements and explore technical uncertainties. Often created using shortcuts, dummy functions, or less efficient approaches.

***When to Use:***

- User requirements are incomplete or unclear.
- Technical issues (e.g., performance, integration) need exploration.
- UI/UX design needs user feedback early.



***Need for a Prototype***

The prototype illustrates the input data formats, messages, reports and the interactive dialogues to the customer. This is a valuable mechanism to gain better understanding of the customer’s needs.

**Advantages:**

- Helps customers visualize the final product.
- Improves understanding of inputs, outputs, and workflows.
- Early feedback reduces misunderstandings.

**Disadvantages:**

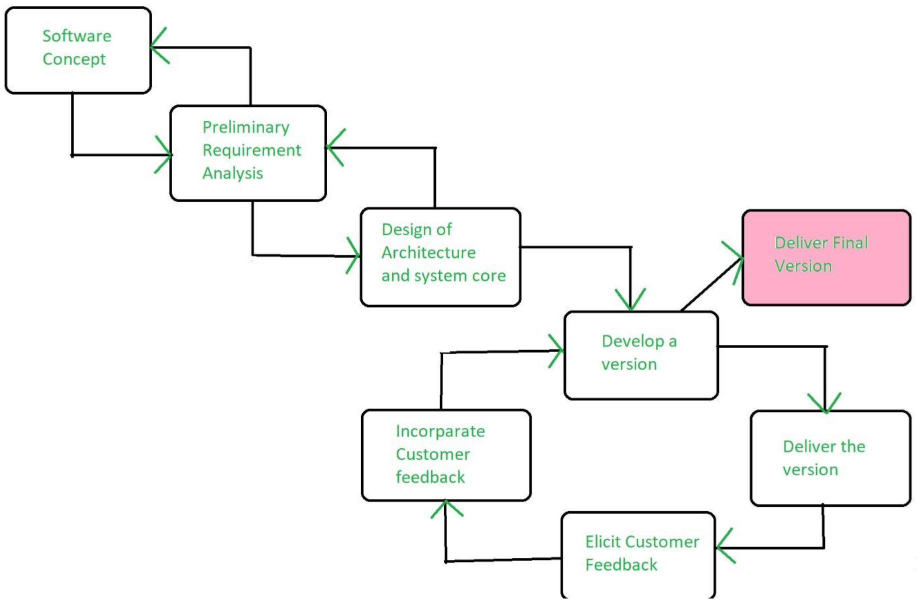
- Prototype may give false impression of final system readiness.
- Building and discarding prototypes adds extra cost/time.

# 3. *Evolutionary Model*

In the Evolutionary Model, development begins with a basic working version, and features are added in increments until the full system is complete. It is also called 'Successive Versions Model' or 'Incremental Model'.

## *Applications:*

- Large projects that can be divided into modules.
- Situations where customers want core functionality early.
- Works well with object-oriented systems as the system is easy to partition into units in terms of objects.


Software Concept
Preliminary Requirement Analysis
Design of Architecture and system core
Deliver Final Version
Develop a version
Incorparate Customer feedback
Deliver the version
Elicit Customer Feedback

## *Advantages:*

- Users get to experiment with partial systems early.
- Core modules are tested thoroughly before adding new feature.

## *Disadvantages:*

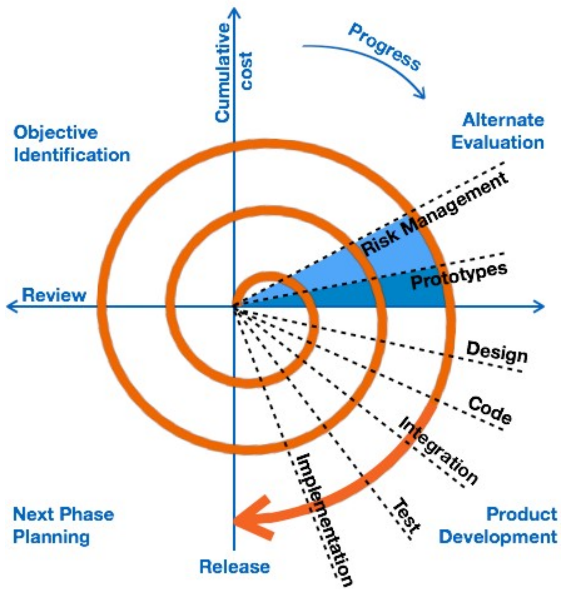
- Dividing the project into acceptable increments may be challenging.
- Each increment must be valuable to the customer.

## 4. ***Spiral Model***

Spiral Model is a model that appears like a outward spiral with many loops where each loops represents a phase of the software process. For example, the innermost loop might be concerned with feasibility study, the next loop with requirements specification, and so on.

It is also called as ‘Meta Model’ since it encompasses all other life cycle models.

Each loop consists of ***four*** quadrants.



1. ***Objective Setting*** – Identify objectives of the phase and examine potential risks associated with them.
2. ***Risk Assessment & Reduction*** – Analyze and identify risks, take steps to reduce risks.
3. ***Development & Validation*** – Develop and validate the next level of the product after resolving the identified risks.
4. ***Review & Planning*** – Review with customer and plan the next loop of the spiral.

**Advantages:**

- Strong risk management capability.
- Combines benefits of other models (meta-model).
- Allows incremental releases with customer feedback.

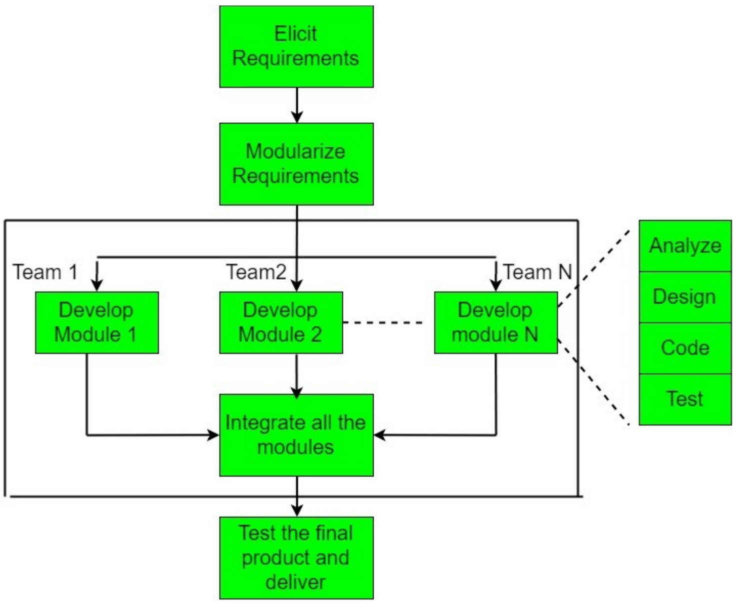
**Disadvantages:**

- Complex to manage and document.
- Not suitable for low-risk, small-scale projects.

## 5. ***RAD Model***

The RAD Model or Rapid Application Development Model is a high-speed, incremental model emphasizing quick delivery usually 60–90 days per cycle using powerful development tools and reusable components.

A software project can be implemented using this model if the project can be broken down into small modules where each module can be assigned independently to separate teams. Multiple teams work on developing the software system using the RAD model in parallel.



***Objectives:***

1. ***Speedy Development*** **–** Use rapid prototyping and short iterations to deliver working systems quickly.
2. ***Adaptability*** **–** Easily adjust to changing requirements and user feedback.
3. ***Stakeholder Participation*** – Ensure active involvement of users and stakeholders throughout.
4. ***Improved Interaction*** – Promote frequent communication and feedback to reduce misunderstandings.
5. ***Better Quality via Prototyping*** – Test and visualize components early to improve quality.
6. ***Customer Satisfaction*** **–** Achieve higher satisfaction through fast delivery and continuous user engagement.

***When to Use:***

- Project requirements are well understood and stable.
- Projects that need to be developed and delivered quickly due to tight deadline.
- Project can be divided into independent, parallel modules.
- Project size is small to medium.
- High user involvement meaning ongoing input and interactions from users is possible.

***Advantages:***

- Faster delivery of functional modules.
- Reuse of components reduces time and cost.
- Continuous user feedback improves satisfaction.
- Easier to adapt to requirement changes.

**Disadvantages:**

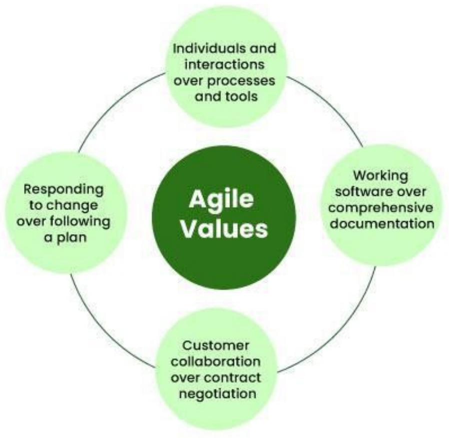
- Needs skilled developers and powerful tools.
- High cost for tools may not suit small projects.
- Not suitable for non-modular systems.
- Customer involvement is mandatory throughout the life cycle.

## 6. ***Agile Model***

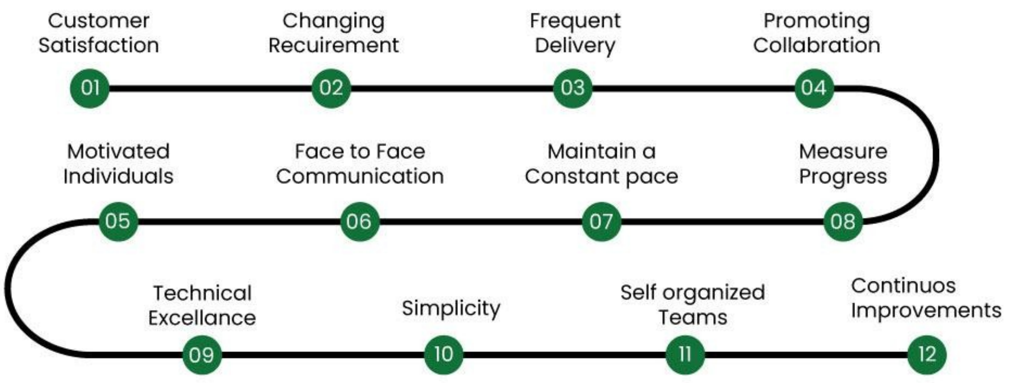
Agile Model is a flexible, iterative, and incremental model based on the Agile Manifesto which focuses on the ***four core values*** and ***twelve principles*** of agile software development.

***Four Core Values:***

1. Individuals and interactions over processes and tools.
2. Working software over comprehensive documentation.
3. Customer collaboration over contract negotiation.
4. Responding to change over following a plan.

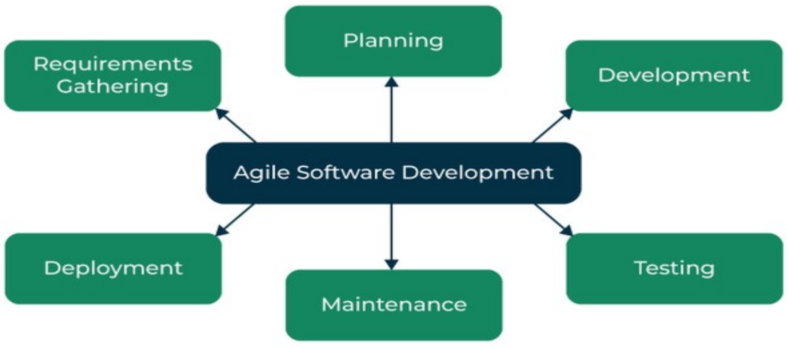

Individuals and interactions over processes and tools
Responding to change over following a plan
Agile Values
Working software over comprehensive documentation
Customer collaboration over contract negotiation

***<u>Twelve Principles:</u>***



1. ***Customer Satisfaction*** – Ensuring customer satisfaction through the early delivery of software.
2. ***Changing Requirement*** – Being open to changing requirements in the stages of the development.
3. ***Frequent delivery*** – Frequently delivering working software with a main focus on preference for time frames.
4. ***Promoting Collaboration*** – Promoting collaboration between business stakeholders and developers as an element.
5. ***Motivated Individuals*** – Structuring the projects around individuals. Providing them with the necessary environment and support.
6. ***Face to Face Communication*** – Prioritizing face to face communication whenever needed.
7. ***Maintain a Constant Pace*** – Fostering development by allowing teams to maintain a pace indefinitely.
8. ***Measure Progress*** – Considering working software as the measure of the progress.
9. ***Technical Excellence*** – Placing attention on excellence and good design practices.
10. ***Simplicity*** – Recognizing the simplicity as crucial factor aiming to maximize productivity by minimizing the work.
11. ***Self-organized Teams*** – Encouraging self organizing teams as the approach to design and build systems.
12. ***Continuous Improvements*** – Regularly reflecting on how to enhance effectiveness and to make adjustments accordingly.

## ***Agile Software Development Process***



1. ***Requirements Gathering*** – The customer's requirements for the software are gathered and prioritized.
2. ***Planning*** – The development team creates a plan for delivering the software, including the features that will be delivered in each iteration.
3. ***Development*** – The development team works to build the software using frequent and rapid iterations.
4. ***Testing*** – The software is thoroughly tested to ensure that it meets the customer's requirements and is of high quality.
5. ***Deployment*** – The software is deployed and put into use.
6. ***Maintenance*** – The software is maintained to ensure that it continues to meet customer's needs and expectations.

### ***Advantages:***

- ***Fast Delivery*** – Quicker deployment builds customer trust.
- ***High Adaptability*** – Easily responds to changing requirements.
- ***Continuous Feedback*** – Improves each increment based on user input.
- ***High Quality*** – Regular testing and focus on good design reduce defects.
- ***Customer Satisfaction*** – Ongoing involvement ensures the product meets needs.

### ***Disadvantages***

- Requires active user involvement.
- May lack predictability in cost/timeline.