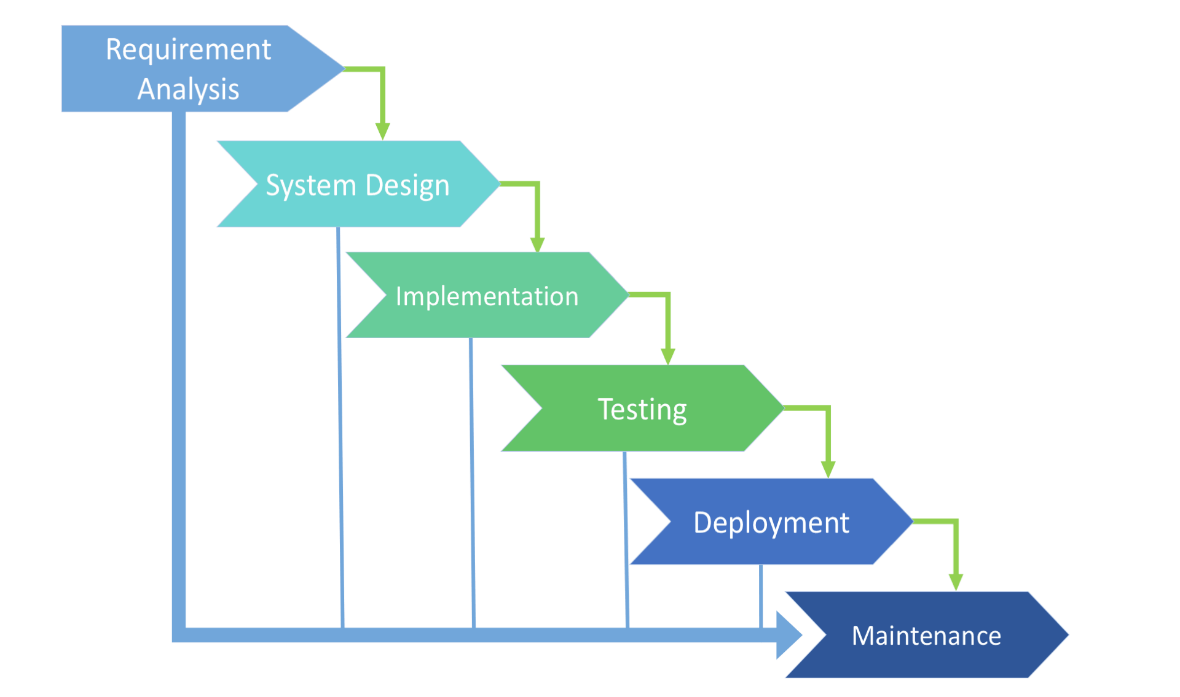
**SDLC (SOFTWARE DEVELOPEMENT LIFECYCLE)**



**🟢 What is SDLC?**

Software Development Life Cycle (SDLC) is the process of planning, developing, testing, and maintaining software. It ensures high-quality software is delivered efficiently.

→Used to Develops a product in systematic way And deliver a high quality product.

→is a process used by software developers to design, develop, test, and deliver high-quality software. It consists of several stages, each with specific tasks.

**✅ SDLC Phases:**

1. Requirement Gathering – Understanding customer needs.
2. Planning – Creating a project roadmap, timeline, and budget.
3. Design – Architectural & UI/UX design of the system.
4. Development – Writing and implementing the actual code.
5. Testing – Ensuring the software is bug-free and meets requirements.
6. Deployment – Releasing the software for users.
7. Maintenance – Fixing issues and updating features post-release.
8. **Requirement Gathering(What features do we need?)**
   * Business analyst collects the requirements from the customers /clients as per the client’s  business needs and document the requirements in the Software requirement specifications(SRS)
9. **Analysis And Planning (What do we need to build?)**
   * This is  to define and document the product requirements and get them approved by the customer .
   * •This is done through software requirements specification (SRS).
   * •Project manager,Business analysts and senior members of the team are the key people involved .
10. **Design (How will it look and work?)**
    * create a design to the software .

**High level design (HLD)**

•It gives the architecture  of the software product

•It is done by architects and senior developers

b.**Low level design (LLD)**

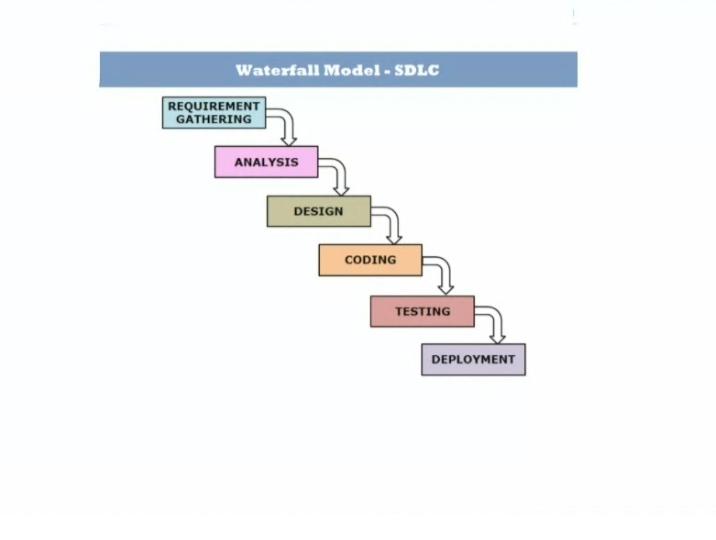
•It describes  how each and every feature in the product should work.

•It is done by senior developers .

1. **Development/Coding (Building the software)**
   * Developers write/construct the  code to  the application
   * Developers of all levels (seniors, juniors ,fresher's) are involved .
   * The outcome of this phase is source code document (SCD) and the product
2. **Testing (Checking for mistakes)**
   * It  is the phase where the developed software is evaluated either manually nor using Automation tools to ensure it meets the specified requirements and is defect free.
   * This phase involves identifying and fixing bugs, verifying functionality, performance, security, and overall quality before deployment.
   * Quality Analyst makes sure that the software is error free, then it goes to the next stage which is Deployment/Production.
3. **Deployment (Launching the software)**
   * It is the phase where the completed software is released to users or production environments.
   * This stage comes after the User Acceptance testing phase and involves making the software available for real-world use.
   * It is done by the deployment/ implementation engineers.
   * If the customer finds the application as expected then sign off is provided by the customer to go live.
4. **Maintenance (Fixing and improving after launch)**
   * It  is the final phase, After the deployment of a product on the production environment ,maintenance  of the  product.
   * where the deployed software is monitored, updated, and improved based on user feedback and changing requirements.
   * if any issue comes up & needs to be fixed or any enhancement is to be done is taken care by the developers.
   * Maintenance should be done as per SLA (service level agreement ).
   * This phase ensures the software remains functional, secure, and efficient over time.
   * Fixing bugs and issues.
   * Updating the software for new environments
   * Enhancing features and usability.
   * Preventing potential future issues.

**TYPES OF SDLC**

**1.Waterfall Model**



The Waterfall Model is a traditional methodology that follows a linear and sequential approach. Each phase must be completed before moving on to the next.

**Phases of the Waterfall Model:**

1. Requirement Gathering & Analysis – Collecting and documenting all software requirements.
2. Planning- As per the SRS ,the BA,Project manager ,developer etc are making plans like which platform we need,which language are used,discussions about the team efficiency and everything are planned in this section.
3. System Design – Creating architecture, UI/UX design, and database structure.
4. Coding & Development– Writing the actual code based on the design.
5. Testing – Identifying and fixing bugs, ensuring functionality meets requirements.
6. Deployment – Releasing the software for use.
7. Maintenance – Ongoing updates, bug fixes, and improvements.

**Characteristics:**

✔ Rigid & Structured – Each phase must be completed before the next begins.  
 ✔ Documentation-Driven – Extensive documentation is required at every stage.  
 ✔ Minimal Client Involvement – Clients review the project mostly at the beginning and end.  
 ✔ Easy to Manage – Best suited for projects with well-defined requirements.

**Advantages:**

✅ Simple and easy to understand.  
 ✅ Works well for small, well-defined projects.  
 ✅ Clear documentation helps with maintenance.

**Disadvantages:**

❌ Difficult to accommodate changes once development starts.  
 ❌ Late testing phase can lead to costly fixes.  
 ❌ Not ideal for complex, evolving projects.

**When to Use Waterfall?**

* When requirements are clear and stable.
* For small to medium-sized projects.
* When strict documentation is needed (e.g., government, healthcare).

**2.V-Model**

* The **V-Model (Validation and Verification Model)** is an **extension of the Waterfall Model** .
* where **each development phase is directly associated with a testing phase**.
* verification :checking if the product is being built correctly and
* validation: checking if the right product is being built at each stage.

**Phases of the V-Model**

**1. Verification Phase -Development**

|  |  |  |
| --- | --- | --- |
| **Requirement Analysis** | Business requirements are gathered and documented. |  |
| **System Design** | High-level design of the system, including architecture, components, and technologies. |  |
| **Software Design** | Major modules, data flow, and integration points are defined. |  |
| **Module Design** | Detailed design of individual modules, functions, and algorithms. |  |
| **Coding** | Actual software development happens. |  |

1. **Validation Phase  – Testing**

|  |  |  |
| --- | --- | --- |
| **Unit Testing** |  | **Tests individual modules and functions.** |
| **Integration Testing** |  | **Ensures modules work together correctly.** |
| **System Testing** |  | **Tests the complete system against requirements.** |
| **User Acceptance Testing (UAT)** |  | **Validates the system with real-world scenarios before deployment.** |

**Advantages of the V-Model**

✅ **Early defect detection** – Testing starts in parallel with development.  
 ✅ **Clear mapping** – Each development phase has a corresponding testing phase.  
 ✅ **Better documentation** – Clearly defined phases and deliverables.  
 ✅ **Improved quality** – Issues are caught early, leading to a more stable product.

**Disadvantages of the V-Model**

❌ **Rigid & inflexible** – Changes in requirements are difficult to accommodate.  
 ❌ **High cost** – More planning and testing can increase project costs.  
 ❌ **Not suitable for iterative development** – Not ideal for Agile or rapidly changing projects.

The **V-Model is best suited for projects where requirements are well-defined, stable, and unlikely to change**, such as medical, aerospace, or banking software.

**3-Incremental Model**



* The **Incremental Model** is a **software development approach** where the system is built **gradually** in small parts (increments).
* Each increment **adds functionality** to the previous version until the full system is developed.

**How It Works:**

1. The project is divided into **multiple smaller modules** (increments).
2. Each increment **undergoes SDLC phases** (requirements, design, coding, testing, and deployment).
3. The first increment **delivers a basic working version** of the software.
4. Subsequent increments **add new features** and improve previous versions.
5. The final increment results in a **fully functional product**.

**Phases of Incremental Model:**

1. **Requirement Analysis** – Identify core functionalities and break them into small modules.
2. **Design & Development** – Design and code each increment separately.
3. **Testing** – Every increment is tested before integration.
4. **Integration** – Each new increment is integrated into the existing system.
5. **Deployment & Feedback** – Users can access and provide feedback after each increment.

**Advantages of Incremental Model:**

✔ **Early Delivery:** A working version is available early in development.  
 ✔ **Flexibility:** Changes can be made after each increment.  
 ✔ **Risk Reduction:** Critical issues are identified early.  
 ✔ **Easier Testing & Debugging:** Each increment is tested separately.  
 ✔ **User Feedback Integration:** Users can provide input after each phase.

**Disadvantages of Incremental Model:**

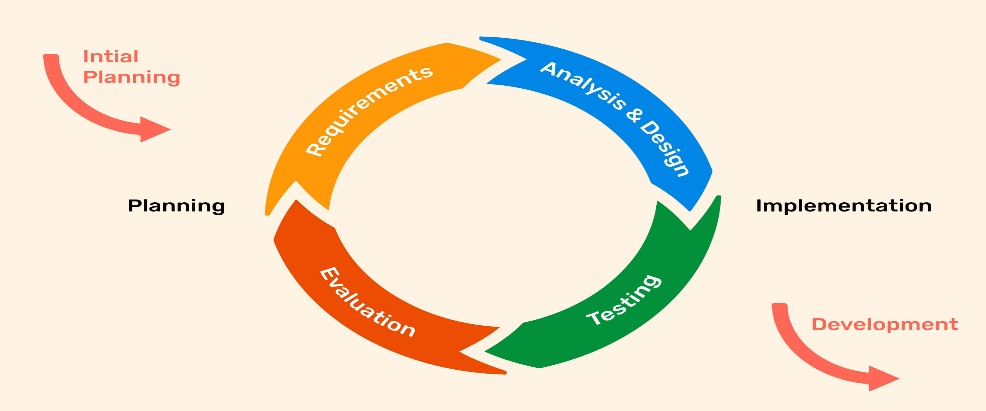
❌ **Requires Good Planning:** Poor planning can cause integration issues.  
 ❌ **Higher Cost:** Frequent testing and updates may increase costs.  
 ❌ **Complex Management:** Managing multiple increments can be challenging.

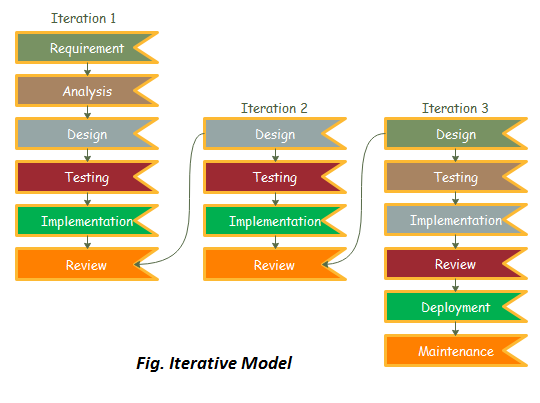
**Use Cases:**

* **Large and complex projects** where phased delivery is beneficial.
* **Projects with evolving requirements** that need continuous feedback.
* **Web applications and products** that require frequent updates.

This model is commonly used in **agile development**

**4.Iterative Model**





* the software is developed and improved **incrementally** through repeated cycles (iterations). Instead of delivering the entire system at once.
* the project is broken down into smaller parts, which are developed, tested, and refined in multiple iterations.

**Key Phases of the Iterative Model:**

1. **Planning & Requirements Gathering**
2. **Design & Development**
3. **Testing & Evaluation**
4. **Implementation & Deployment**
5. **Maintenance**

**Advantages of the Iterative Model:**

✅ **Early Feedback** – Users can see working versions early in development.  
 ✅ **Flexible Changes** – Modifications can be made in later iterations.  
 ✅ **Risk Management** – Issues are identified and resolved early.  
 ✅ **Continuous Improvement** – The software evolves with user needs.

**Disadvantages of the Iterative Model:**

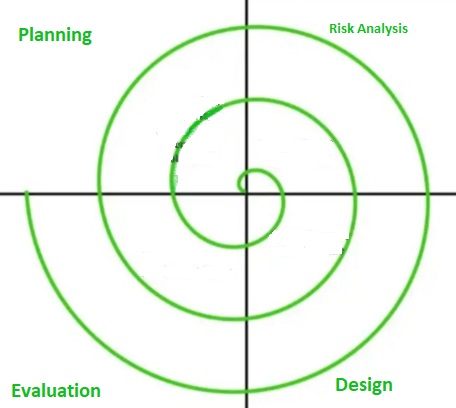
❌ **More Resource-Intensive** – Requires frequent testing and reviews.  
 ❌ **Scope Creep** – Constant changes can lead to uncontrolled expansion.  
 ❌ **High Dependency on Initial Requirements** – Poor initial planning can lead to major rework.

**When to Use :**

* Large and complex projects where requirements evolve over time.
* Systems requiring frequent updates and enhancements.
* Applications where user feedback is crucial (e.g., web apps, mobile apps).

The **Iterative Model** is commonly used in **Agile Development**

**5-Spiral Model**



* The **Spiral Model** is a **risk-driven** and **iterative** SDLC model that combines elements of both **Waterfall** and **Agile** methodologies.
* It is ideal for complex, high-risk projects where requirements are unclear at the start.

**Phases of the Spiral Model:**

Each iteration (spiral) in the model consists of four key phases:

1. **Planning Phase**
   * Gather requirements and define objectives.
   * Identify constraints and risks.
   * Plan the next iteration.
2. **Risk Analysis Phase**
   * Identify potential risks and uncertainties.
   * Perform feasibility studies and risk mitigation strategies.
   * Prototype development to test feasibility.
3. **Development & Testing Phase**
   * Design and develop a functional version of the software.
   * Test the prototype or module.
   * Implement changes based on feedback.
4. **Evaluation & Review Phase**
   * Collect feedback from users and stakeholders.
   * Assess the progress and decide on the next cycle.
   * Repeat the spiral for refinement and improvements.

**Advantages of the Spiral Model:**

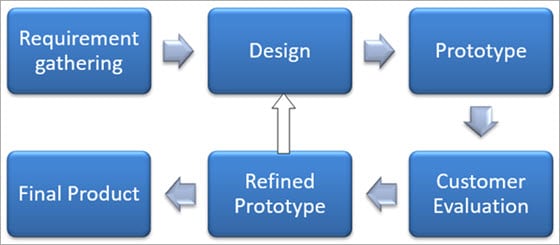
✔ **Risk Management** – Helps identify and mitigate risks early.  
 ✔ **Iterative Development** – Continuous refinement improves quality.  
 ✔ **Flexibility** – Suitable for evolving and unclear requirements.  
 ✔ **Customer Feedback Integration** – Ensures alignment with user needs.

**Disadvantages of the Spiral Model:**

✖ **Expensive & Time-Consuming** – Requires extensive risk analysis and iterations.  
 ✖ **Complex Process** – Needs experienced risk assessment.  
 ✖ **Not Ideal for Small Projects** – Best suited for large, complex systems.

**When to Use the Spiral Model?**

* When **project risks are high** and need continuous evaluation.
* For **large-scale, mission-critical software** (e.g., aerospace, defense, banking systems).
* When requirements are **unclear or frequently changing**.

**6-Prototype Model**[](https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2018/04/Prototype-Model.jpg)

* a **prototype (working model)** of the software is built, tested, and refined before the final system is developed.
* This model is useful when requirements are unclear, allowing stakeholders to visualize and interact with an early version of the software before committing to full development.

**Steps in the Prototype Model:**

1. **Requirement Gathering** – Initial requirements are collected, but they may not be fully defined.
2. **Quick Design** – A simple prototype with key functionalities is created.
3. **Prototype Development** – A basic working model of the software is built.
4. **User Evaluation** – Stakeholders review and provide feedback on the prototype.
5. **Refinement & Iteration** – The prototype is refined based on feedback, and the cycle repeats until the desired outcome is achieved.
6. **Final Product Development** – Once the prototype is approved, the actual system is developed and tested.

**Advantages of the Prototype Model:**

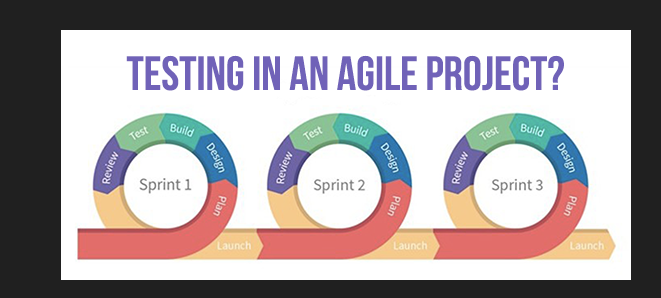
✔ **Better Requirement Understanding** – Helps clarify requirements through user feedback.  
 ✔ **Improved User Involvement** – Users can interact with an early version of the product.  
 ✔ **Reduced Development Risks** – Issues are identified early, reducing project failures.  
 ✔ **Faster Development** – Identifying necessary changes early saves time and cost.

**Disadvantages of the Prototype Model:**

❌ **Increased Development Cost** – Continuous modifications can lead to higher expenses.  
 ❌ **Time-Consuming** – Frequent iterations may extend the project timeline.  
 ❌ **Scope Creep** – Users may keep requesting changes, making it hard to finalize requirements.  
 ❌ **Lack of Documentation** – Rapid changes may result in poor documentation.

**When to Use the Prototype Model?**

* When **requirements are unclear or evolving**.
* For **highly interactive systems** like UI/UX-heavy applications.
* When user **feedback is crucial** before full development.
* For **new and innovative projects** where feasibility needs to be tested.

**7-Agile Model**

* The **Agile Model** is an iterative and incremental approach to software development.
* It Provides  **flexibility, collaboration, and customer feedback**.
* Agile breaks the project into **small, manageable cycles called iterations (or sprints)**. Each iteration results in a **working software increment** that can be tested and reviewed.

**Phases of Agile Model in SDLC**

1. **Requirement Analysis**– Understanding the business needs, project goals, and defining the requirements.
2. **Iteration / Incremental Development** –
   * The project is broken into small cycles called **sprints** (typically 1-4 weeks).
   * Teams prioritize features and develop a **working product increment** in each sprint.
   * Continuous feedback from users and stakeholders is corporated.
3. **Testing**
   * Testing happens in **parallel** with development
   * Bugs are fixed as they arise instead of waiting until the end.
4. **Deployment & Release** –
   * After multiple iterations, the product reaches a stable version for deployment.
   * Frequent releases ensure customers get updates faster.
5. **Maintenance & Continuous Improvement** –
   * Enhancements, bug fixes, and new features are continuously developed based on feedback.

**Popular Agile Frameworks:**

1. **Scrum** – Uses sprints, daily stand-up meetings, and defined roles (Scrum Master, Product Owner, Development Team).
2. **Kanban** – Visual workflow management using boards to track progress.

**Advantages of Agile Model:**

✅ **Faster Delivery** – Working software is delivered in short cycles.  
 ✅ **Flexibility** – Changes can be made anytime based on new requirements.  
 ✅ **Customer Involvement** – Regular feedback ensures the product meets expectations.  
 ✅ **Improved Quality** – Continuous testing ensures fewer defects.  
 ✅ **Better Team Collaboration** – Encourages teamwork and communication.

**Disadvantages of Agile Model:**

❌ **Less Predictability** – Harder to estimate time and cost upfront.  
 ❌ **Frequent Changes** – May cause scope creep if not managed well.  
 ❌ **High Team Commitment** – Requires active participation from developers, testers, and customers.

**When to Use Agile?**

✔ When requirements are **uncertain or frequently changing**.  
 ✔ When **quick delivery and customer feedback** are essential.  
 ✔ For **complex projects** that require iterative improvements.  
 ✔ When the **team is collaborative and adaptable**.

Agile has become the **industry standard** for modern software development.

**Agile Ceremonies (Scrum Events)**

Agile ceremonies are key meetings in **Scrum** that help teams collaborate efficiently.

1. **Sprint Planning**
   * Purpose: Plan the work for the upcoming sprint (1-4 weeks).
   * Participants: Scrum Team (Developers, Scrum Master, Product Owner).
     + **Scrum Master:** is the **coach and facilitator** of a Scrum team , Help the team work smoothly and solve problems that cause delays.
   * Activities:  
     ✅ Product Owner prioritizes backlog items.  
     ✅ Team selects tasks for the sprint.  
     ✅ Defines the **Sprint Goal**.
2. **Daily Stand-up (Daily Scrum)**
   * Purpose: A quick (15-minute) meeting to track progress.
   * Participants: Development Team, Scrum Master.
   * Activities: Each team member answers:  
     ✅ What did I do yesterday?  
     ✅ What will I do today?  
     ✅ Any blockers or issues?
3. **Sprint Review**
   * Purpose: Show the finished work to stakeholders
   * Participants: Scrum Team, Stakeholders.
   * Activities:   
     ✅ Team showcases finished work.  
     ✅ Stakeholders give feedback.  
     ✅ Product Owner updates the backlog if needed.
4. **Sprint Retrospective**
   * Purpose: Improve the team’s processes.
   * Participants: Scrum Team.
   * Activities:  
     ✅ Discuss what went well.  
     ✅ Identify challenges.  
     ✅ Plan improvements for the next sprint.