Evolution of Software Development

1. **Early Days (1940s-1960s)**

    - **What Happened?** In the beginning, programmers would just write code as they went along. There wasn't much planning or documentation.
    - **Problems:** This led to lots of mistakes and made the software hard to fix and improve.

2. **Waterfall Model (1960s-1990s)**

    - **What is it?** The Waterfall Model was one of the first formal methods to develop software. It's like a checklist where you finish one step before moving to the next.
    - **Steps:** Planning, Requirements Analysis, Design, Coding, Testing, Deployment, and Maintenance.
    - **Problems:** It was very rigid. If anything changed or needed fixing later, it was hard to go back and make changes.

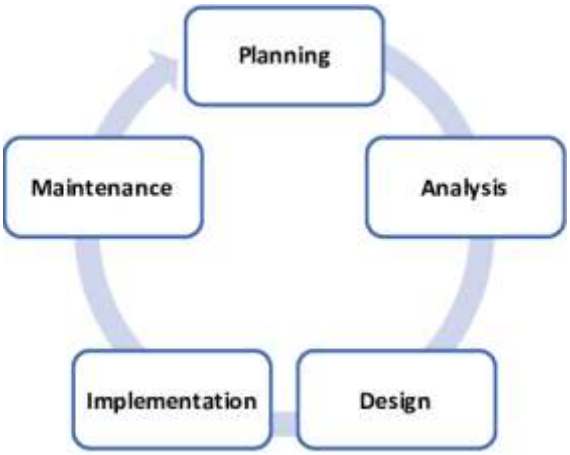
3. **Agile Development (1990s-Present)**

    - **What is it?** Agile is a more flexible way of developing software. Instead of doing everything step by step, you break the project into smaller parts and work on them in short bursts called sprints.
    - **Benefits:** This allows for quick adjustments and improvements based on feedback.

Software Development Life Cycle (SDLC) Phases

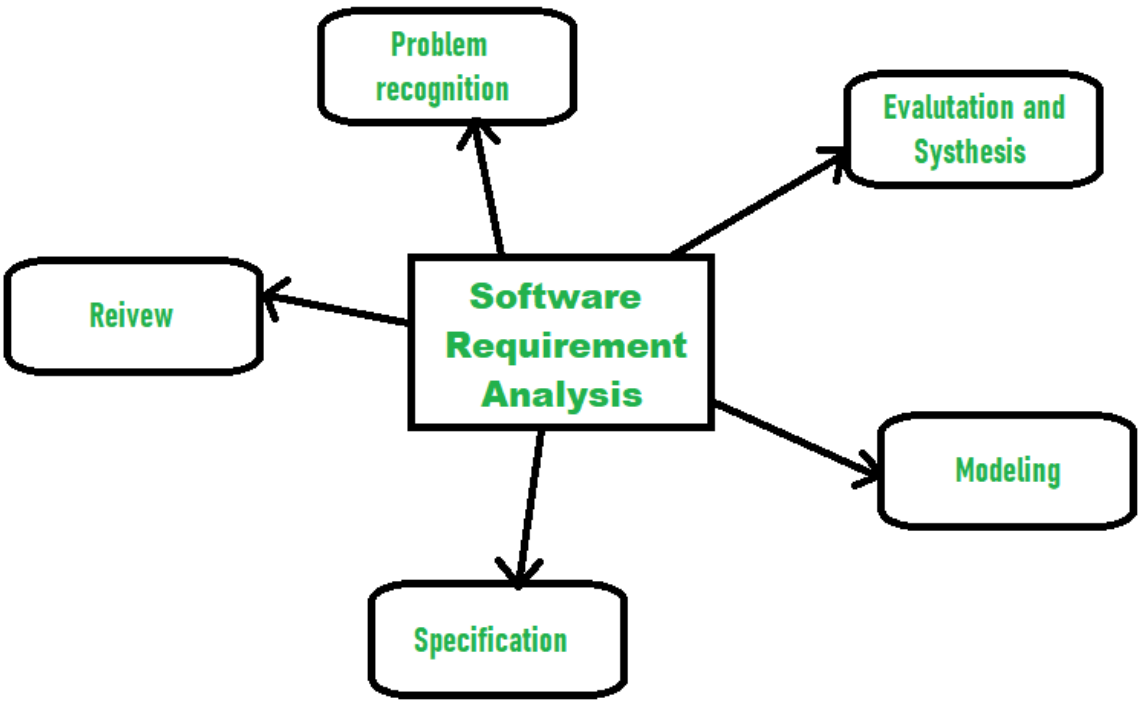
The SDLC is a roadmap that helps guide software development. It includes several phases, each with specific tasks to ensure the software is high quality.

1. **Planning and Analysis**



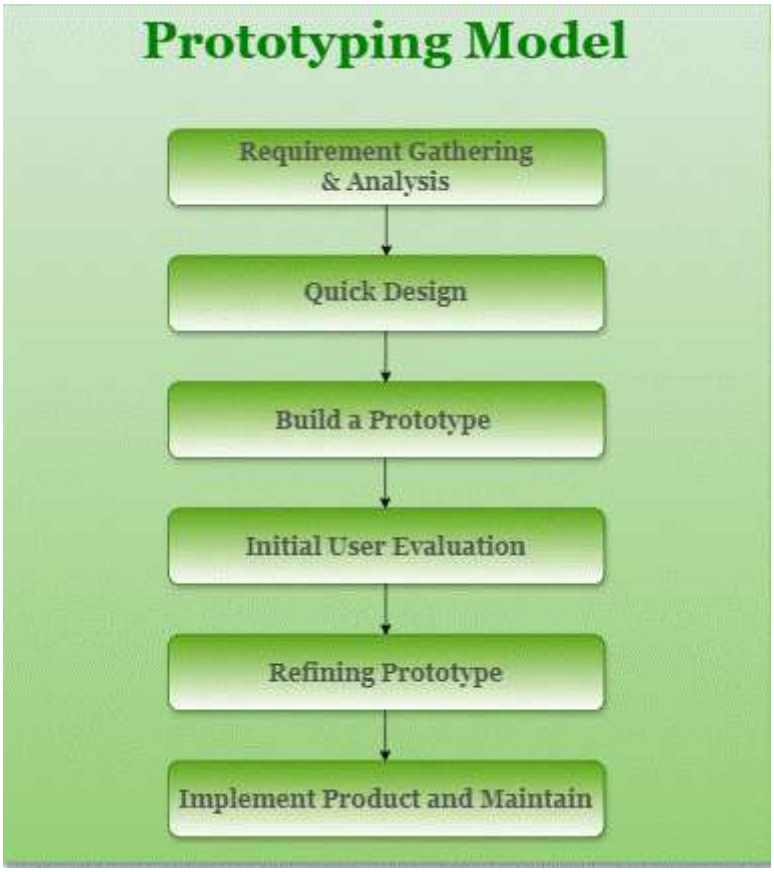
- **Planning:** Figure out the project's goals and what needs to be done.
- **Analysis:** Check if the project is practical and what resources (people, tools, money) are needed.
- **Risk Management:** Identify potential problems and plan how to handle them.
- **Scheduling:** Create a timeline with milestones and deadlines.

2. **Requirements Analysis**



- **Requirements Gathering:** Talk to stakeholders (users, clients) to understand what they need from the software.
- **Documentation:** Write down all the requirements in detail.
- **Validation:** Make sure the requirements are complete and realistic.

3. Design and Prototyping

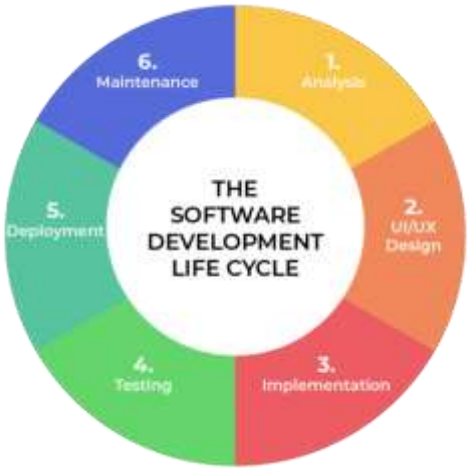


- **System Design:** Create diagrams and models to show how the software will work.
- **Prototyping:** Build a simple version of the software to test ideas and get feedback.
- **Review:** Check the design with stakeholders to ensure it meets their needs.

4. Development of the Application

- **Implementation:** Write the actual code for the software.
- **Integration:** Combine different parts of the software to work together.
- **Version Control:** Use tools like Git to manage changes to the code.

5. Testing and Deployment



- **Testing:** Test the software to find and fix bugs.

- **Quality Assurance:** Make sure the software meets quality standards.
- **Deployment:** Release the software to users.
- **Post-Deployment Support:** Provide ongoing help and fix any issues that come up after release.

6. Project Management

- **Project Planning:** Define what needs to be done and how it will be achieved.
- **Resource Management:** Make sure the right people and tools are available.
- **Time Management:** Keep the project on schedule.
- **Cost Management:** Stay within the budget.
- **Communication Management:** Ensure clear and effective communication among everyone involved.
- **Risk Management:** Identify and handle potential risks throughout the project.
- **Quality Management:** Regularly check that the project meets quality standards.

2. Conclusion

The evolution of software and its development methodologies has led to the creation of structured frameworks like the SDLC, which guide projects through a series of well-defined phases. These phases help ensure that software is developed in a systematic,

controlled, and efficient manner, leading to higher quality and more reliable products.