**Software Devlopment**

<https://www.geeksforgeeks.org/what-is-software-development/>

**Software Development Life Cycles**

**Software development life cycle (SDLC) is a structured process that is used to design, develop, and test good-quality software.** SDLC, or software development life cycle, is a methodology that defines the entire procedure of software development step-by-step.

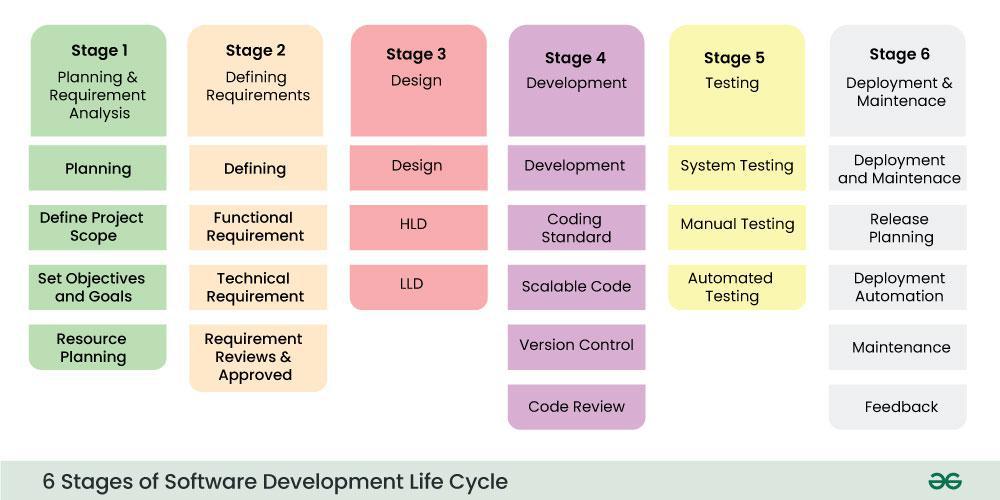


*Software Development Life Cycle (SDLC)*

The goal of the SDLC life cycle model is to deliver high-quality, maintainable software that meets the user’s requirements. SDLC in software engineering models outlines the plan for each stage so that each stage of the software development model can perform its task efficiently to deliver the software at a low cost within a given time frame that meets users’ requirements

## Stages of the Software Development Life Cycle

SDLC specifies the task(s) to be performed at various stages by a software engineer or developer. It ensures that the end product is able to meet the customer’s expectations and fits within the overall budget. Hence, it’s vital for a software developer to have prior knowledge of this software development process.



**Waterfall model :**

<https://www.geeksforgeeks.org/waterfall-model/>

**V-Model model :**

<https://www.geeksforgeeks.org/software-engineering-sdlc-v-model/>

**Prototype Model:**

<https://www.geeksforgeeks.org/software-engineering-prototyping-model/>

**Incremental model :**

<https://www.geeksforgeeks.org/software-engineering-incremental-process-model/>

**Evolutionary model :**

<https://www.geeksforgeeks.org/what-are-evolutionary-process-models/>

**Rapid application development model (RAD) – Software Engineering**

<https://www.geeksforgeeks.org/software-engineering-rapid-application-development-model-rad/>

# **Spiral Model :**

<https://www.geeksforgeeks.org/software-engineering-spiral-model/>

**Agile processes model;**

<https://www.geeksforgeeks.org/software-engineering-agile-development-models/>

**Project size Estimation Techniques**

<https://www.geeksforgeeks.org/software-engineering-project-size-estimation-techniques/>

**Software Process customization and improvement**

<https://www.geeksforgeeks.org/software-process-customization-and-improvement/>

**Product and Process Metrics**

<https://www.javatpoint.com/software-engineering-software-metrics>

**Functional and Non-functional requirements**

<https://www.geeksforgeeks.org/functional-vs-non-functional-requirements/#functional-requirements>

**Function-oriented and Object-oriented software development**

<https://www.geeksforgeeks.org/difference-between-function-oriented-design-and-object-oriented-design/>

**Requirements Gathering and Analysis**

<https://www.geeksforgeeks.org/requirements-gathering-introduction-processes-benefits-and-tools/>

**Software Requirements Specification (SRS)**

<https://www.geeksforgeeks.org/software-requirement-specification-srs-document-checklist/>

**Software Product and Process Characteristics**

<geeksforgeeks.org/difference-between-product-and-process/>

**Project Planning :**

Planning is an important process in[project management](https://www.geeksforgeeks.org/project-management-tutorial/) that requires the objectives, scope, schedule, and budget among other things to be clearly outlined. Planning entails establishing a path for the project team that will direct them through every stage of their life expectancy to get past together efficiently and effectively down the derived objectives of this venture.

1. It involves creating a comprehensive roadmap that guides the team throughout the [project lifecycle](https://www.geeksforgeeks.org/project-management-life-cycle/), from initiation to completion.
2. Effective planning sets the foundation for successful project execution by ensuring clarity, alignment with objectives, and efficient resource utilization.

## Key Components of Project Planning

Here are the following key components of Project Planning:

1. **Scope Definition:**Clearly articulate [project goals](https://www.geeksforgeeks.org/whats-the-significance-of-setting-project-goals-and-objectives/), deliverables, and constraints with acceptance criteria to prevent scope creep.
2. **Time Planning (Scheduling):**Develop a timeline outlining activity sequences, task dependencies, and estimated durations, often represented using Gantt charts.
3. **Resource Planning:**Identify and allocate human resources, equipment, and other necessary resources for efficient project implementation.
4. **Cost Estimation and Budgeting:**Estimate costs for each project activity and create a budget to control and manage expenses effectively.
5. **Risk Management:**Identify potential project risks and develop response or mitigation plans to ensure successful project outcomes.
6. **Communication Planning:**Determine communication methods, frequencies, formats, and channels both within the project team and with stakeholders.
7. **Quality Planning:**Establish a clear understanding of quality requirements and implement procedures to ensure project deliverables meet specified standards.
8. **Procurement Planning:**Identify external goods and services required for the project, and map out the procurement process.
9. **Stakeholder Management:**Identify stakeholders, understand their needs and expectations, and develop engagement strategies for effective communication.
10. **Monitoring and Control Planning:**Establish mechanisms for tracking project progress, and performance, and implement corrective actions if necessary to stay on course.

**Q-1. Differentiate between agile methodology and waterfall model of software development.**

| [**Agile Project Management**](https://www.geeksforgeeks.org/what-is-agile-project-management-apm/) | [**Waterfall Project Management**](https://www.geeksforgeeks.org/waterfall-model/) |
| --- | --- |
| Client input is required throughout the product development. | Client input is required only after completing each phase. |
| Changes can be made at any stage. | Changes cannot be made after the completion of a phase. |
| Coordination among [project teams](https://www.geeksforgeeks.org/software-project-team-organization/) is required to ensure correctness. | Coordination is not needed as one team starts the work after the finish of another team. |
| It is really useful in large and complex projects. | It is mainly used for small [project development](https://www.geeksforgeeks.org/phases-project-management-processes/). |
| The testing part can be started before the development of the entire product. | Testing can only be performed when the complete product is ready. |
| A Small team is sufficient for Agile project management. | It requires a large team. |
| The cost of development is less. | The cost of development is high. |
| It completes the project in comparatively less time. | It takes more time compared to Agile. |
| The Agile Method is known for its flexibility. | The waterfall Method is a structured software development methodology so it is quite rigid. |
| After each sprint/cycle test plan is discussed. | Hardly any test plan is discussed during a cycle. |
|  |  |
|  |  |

(a) **What is a prototype?**

A prototype is a preliminary version or an early representation of a software product, typically developed to demonstrate concepts, test functionality, and gather feedback from stakeholders. Prototypes can range from simple mock-ups or wireframes to more functional and interactive versions of the software.

**(b) Is it necessary to develop a prototype for all types of projects?**

No, it is not necessary to develop a prototype for all types of projects. The decision to develop a prototype depends on various factors such as project goals, complexity, budget, timeline, and stakeholder preferences. Some projects may have well-defined requirements and minimal uncertainty, making prototyping unnecessary.

**(c) Under what circumstances is it beneficial to construct a prototype?**

Prototyping is beneficial in the following circumstances:

**Uncertain or evolving requirements**: When requirements are not well-defined or are likely to change, prototyping allows for exploration and experimentation to better understand user needs and refine requirements.

**Complex or innovative projects**: For projects involving complex technologies or innovative solutions, prototyping helps validate technical feasibility and identify potential challenges early in the development process.

**User involvement:** Prototyping facilitates early user involvement and feedback, enabling stakeholders to visualize and interact with the software, provide input, and suggest improvements.

**Risk mitigation:** Prototyping can help mitigate project risks by identifying issues and addressing them before full-scale development begins, reducing the likelihood of costly rework later in the project lifecycle.

**(d) Does construction of a prototype always increase the overall cost of software development?**

No, construction of a prototype does not always increase the overall cost of software development. While prototyping may incur additional upfront costs, it can ultimately lead to cost savings by:

Identifying and addressing issues early in the development process, reducing the need for costly rework later.

Improving communication and alignment between stakeholders, leading to fewer misunderstandings and changes.

Accelerating development by providing a clear vision of the final product and guiding subsequent development efforts.

Enhancing the quality and usability of the software, resulting in fewer defects and support requests post-deployment.

**Q13. What are the different categories of software development projects according to the COCOMO estimation model? Give an example of software product development projects belonging to each of these categories.**

The Cocomo Model is a procedural cost estimate model for software projects and is often used as a process of reliably predicting the various parameters associated with making a project such as size, effort, cost, time, and quality.

**Effort:**Amount of labor that will be required to complete a task. It is measured in person-months units.

**Schedule:** This simply means the amount of time required for the completion of the job, which is, of course, proportional to the effort put in. It is measured in the units of time such as weeks, and months.

**1. Organic**

A software project is said to be an organic type if the team size required is adequately small, the problem is well understood and has been solved in the past and also the team members have a nominal experience regarding the problem.

**2. Semi-detached**

A software project is said to be a Semi-detached type if the vital characteristics such as team size, experience, and knowledge of the various programming environments lie in between organic and embedded. The projects classified as Semi-Detached are comparatively less familiar and difficult to develop compared to the organic ones and require more experience better guidance and creativity. Eg: Compilers or different Embedded Systems can be considered Semi-Detached types.

**3. Embedded**

A software project requiring the highest level of complexity, creativity, and experience requirement falls under this category.

**Q14. What do you mean by project size? What are the popular metrics to measure project size? How can the size of a project be estimated during the project planning stage?**

Project size refers to the scale or magnitude of a software development project, typically measured in terms of the amount of work, effort, or functionality required to complete the project successfully. Project size is a crucial factor in project planning, estimation, and management, as it influences resource allocation, scheduling, and budgeting decisions.

Popular metrics to measure project size include:

**Lines of Code (LOC)**: LOC measures the size of a project by counting the number of lines of code in the software source files..

**Function Points (FP)**: Function Points measure the size of a project based on the functionality delivered to the end-user. Function Points consider factors such as inputs, outputs, inquiries, files, and interfaces to quantify the functionality provided by the software.

**Use Case Points (UCP)**: Use Case Points measure the size of a project based on the number and complexity of use cases identified for the system.

**Story Points**: Story Points are a measure used in Agile methodologies such as Scrum to estimate the size and effort required for user stories or product backlog items.

**CMM model?**

The Capability Maturity Model (CMM) is a framework used to assess and improve the maturity of an organization's software development processes. It defines five levels of maturity, each representing a stage of process improvement:

**Initial**: Processes are ad hoc and unpredictable. Success depends on individual effort and heroics, and there is little consistency or control over outcomes.

**Managed**: Processes are documented, standardized, and managed. Basic project management techniques are implemented to track cost, schedule, and quality.

**Defined**: Processes are well-defined, documented, and standardized across the organization. There is a focus on continuous process improvement, and processes are tailored to meet specific project needs.

**Quantitatively Managed**: Processes are quantitatively measured and controlled. Data-driven decision-making is used to manage process performance and make improvements.

**Optimizing**: Continuous process improvement is institutionalized, and organizations focus on innovation and optimization. Processes are continuously monitored, evaluated, and refined to improve efficiency and effectiveness.