**Waterfall Model**

**Features of the SDLC Waterfall Model**

**Phases of SDLC Waterfall Model – Design**

**Advantages of the SDLC Waterfall Model**

**Disadvantages of the SDLC Waterfall Model**

**When to Use the SDLC Waterfall Model?**

**Waterfall Model**

The Waterfall Model is also known as the traditional waterfall software life cycle model.

* In the waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.
* The classical waterfall model divides the life cycle into a set of phases.
* The waterfall model considers that one phase can be started after completion of the previous phase, which is the output of one phase will be the input to the next phase.
* Due to this the development process can be considered as a sequential flow in the waterfall. Here the phases do not overlap with each other.

**Features of the SDLC Waterfall Model**

1. **Sequential Approach**: The waterfall model involves a sequential approach to software development, where each phase of the project is completed before moving on to the next one.
2. **Document-Driven:**The waterfall model relies heavily on documentation to ensure that the project is well-defined and the project team is working towards a clear set of goals.
3. **Quality Control:** The waterfall model places a high emphasis on quality control and testing at each phase of the project, to ensure that the final product meets the requirements and expectations of the stakeholders.
4. **Rigorous Planning**: The waterfall model involves a rigorous planning process, where the project scope, timelines, and deliverables are carefully defined and monitored throughout the project lifecycle.

**Phases of SDLC Waterfall Model – Design**

The Waterfall Model is a classical software development methodology that was first introduced by Winston W. Royce in 1970. It is a linear and sequential approach to software development that consists of several phases that must be completed in a specific order.

**6 Phases:**

**1. Requirements:** The first phase involves gathering requirements from stakeholders and analysing them to understand the scope and objectives of the project.

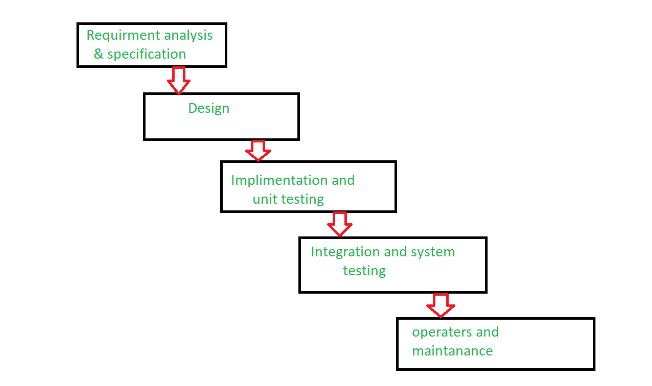
**2. Design:** Once the requirements are understood, the design phase begins. This involves creating a detailed design document that outlines the software architecture, user interface, and system components.

**3. Development:** The Development phase include implementation involves coding the software based on the design specifications. This phase also includes unit testing to ensure that each component of the software is working as expected.

**4. Testing:** In the testing phase, the software is tested as a whole to ensure that it meets the requirements and is free from defects.

**5. Deployment:** Once the software has been tested and approved, it is deployed to the production environment.

**6. Maintenance:** The final phase of the Waterfall Model is maintenance, which involves fixing any issues that arise after the software has been deployed and ensuring that it continues to meet the requirements over time.



**Advantages of the SDLC Waterfall Model**

* **Easy to Understand:** The Classical Waterfall Model is very simple and easy to understand.
* **Individual Processing:** Phases in the Classical Waterfall model are processed one at a time.
* **Properly Defined:** In the classical waterfall model, each stage in the model is clearly defined.
* **Clear Milestones:** The classical Waterfall model has very clear and well-understood milestones.
* **Properly Documented:** Processes, actions, and results are very well documented.
* **Reinforces Good Habits:** The Classical Waterfall Model reinforces good habits like define-before-design and design-before-code.
* **Working:** Classical Waterfall Model works well for smaller projects and projects where requirements are well understood.

**Disadvantages of the SDLC Waterfall Model**

* **No Feedback Path:** In the classical waterfall model evolution of software from one phase to another phase is like a waterfall. It assumes that no error is ever committed by developers during any phase. Therefore, it does not incorporate any mechanism for error correction.
* **Difficult to accommodate Change Requests:** This model assumes that all the customer requirements can be completely and correctly defined at the beginning of the project, but the customer’s requirements keep on changing with time. It is difficult to accommodate any change requests after the requirements specification phase is complete.
* **No Overlapping of Phases:** This model recommends that a new phase can start only after the completion of the previous phase. But in real projects, this can’t be maintained. To increase efficiency and reduce cost, phases may overlap.
* **Limited Flexibility:** The Waterfall Model is a rigid and linear approach to software development, which means that it is not well-suited for projects with changing or uncertain requirements. Once a phase has been completed, it is difficult to make changes or go back to a previous phase.
* **Limited Stakeholder Involvement:** The Waterfall Model is a structured and sequential approach, which means that stakeholders are typically involved in the early phases of the project (requirements gathering and analysis) but may not be involved in the later phases [(implementation, testing, and deployment).](https://www.geeksforgeeks.org/post-deployment-testing-in-software-testing/)
* **Late Defect Detection:** In the Waterfall Model, testing is typically done toward the end of the development process. This means that defects may not be discovered until late in the development process, which can be expensive and time-consuming to fix.
* **Lengthy Development Cycle:**The Waterfall Model can result in a lengthy development cycle, as each phase must be completed before moving on to the next. This can result in delays and increased costs if requirements change or new issues arise.

**When to Use the SDLC Waterfall Model?**

Here are some cases where the use of the Waterfall Model is best suited:

* **Well-understood Requirements:** Before beginning development, there are precise, reliable, and thoroughly documented requirements available.
* **Very Little Changes Expected:** During development, very little adjustments or expansions to the project’s scope are anticipated.
* **Small to Medium-Sized Projects**: Ideal for more manageable projects with a clear development path and little complexity.
* **Predictable:**Projects that are predictable, low-risk, and able to be addressed early in the development life cycle are those that have known, controllable risks.
* **Regulatory Compliance is Critical:** Circumstances in which paperwork is of utmost importance and stringent regulatory compliance is required.
* **Client Prefers a Linear and Sequential Approach**: This situation describes the client’s preference for a linear and sequential approach to project development.
* **Limited Resources**: Projects with limited resources can benefit from a set-up strategy, which enables targeted resource allocation.