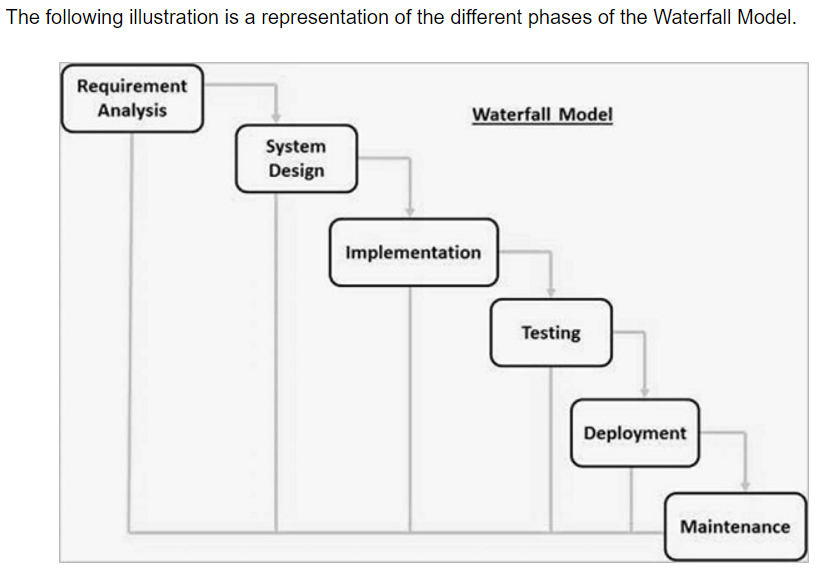
**Waterfall Model**

It is also referred to as a **linear-sequential life cycle model**. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases. The Waterfall model is the earliest SDLC approach that was used for software development.

The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.



**The sequential phases in Waterfall model are –**

* **Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
* **System Design** − The requirement specifications from first phase are studied in this phase and the system design is prepared.
* **Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
* **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
* **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

**Functional testing:** it tests what the product does. It checks the operations and actions of an application. It checks as per the customer requirements. What to test. Example, a login page must show textboxes to enter the username and password

**Non-functional testing:** it checks the behavior of an application. It checks as per the customer expectation. Defines how to test. Example, test if a login page is getting loaded in 5 seconds

* **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also, to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

**Advantages of Waterfall Model**

* Easy to understand and use
* Easy to manage
* Works well for smaller and low budget projects where requirements are very well understood
* Clearly defined stages and well understood
* Easy to arrange tasks
* Process and results are well documented

**Disadvantages of Waterfall Model**

* All requirements must be known upfront
* No feedback
* No parallelism
* No working software is produced until late during the life cycle
* Not a good model for complex and object-oriented projects
* Poor model for long and ongoing projects
* Cannot accommodate changing requirements
* 60% efforts in maintenance

**When to use Waterfall Model**

* Requirements are clear and fixed that may not change
* There are no ambiguous requirements (no confusion)
* It is good to use this model when the technology is well understood
* The project is short and cost is low
* Risk is zero or minimum