
# WATERFALL MODEL

# Waterfall Management Guide

These days there is a strong push for Agile Management, as opposed to Waterfall. Personally at Castellan Systems we believe that the agility should be applied to the project development lifecycle rather than the project management. A strong project management process is equally, if not more, appropriate to an "Agile" environment. The more agile or dynamic a project environment is, the more important strong project management principles and processes are. In a fast moving project environment things can go wrong very quickly and if the project management disciplines are not strongly applied these problems may not be picked up until it's too late.

> The term "Waterfall" refers to a traditional software development methodology where the project is defined sequentially and through clear project phases.

The term "Waterfall" refers to a traditional software development methodology where the project is defined sequentially and through clear project phases. This is a common approach to large-scale projects where little change is expected to the overall project plan. This is a distinct approach from Agile project planning, which is designed to accommodate rapid changes to the schedule.

# What Is Waterfall Methodology?

# Waterfall Model

The Waterfall Model was first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

Waterfall model is the earliest SDLC approach that was used for software development.

The Waterfall approach to systems analysis and design was the first established modern approach to building a system. This method was originally defined by Winston W. Royce in 1970. It quickly gained support from managers because everything flows logically from the beginning of a project through the end. Sources differ when it comes to the specific steps in the Waterfall process, and I will detail some of these differences in the next paragraph. However, the basic underlying logic and steps present themselves in each interpretation.

The Waterfall Model illustrates the software development process in a linear sequential flow; hence it is also referred to as a linear-sequential life cycle model. This means that any phase in the development process begins only if the previous phase is complete. In waterfall model phases do not overlap.

# Waterfall Project Management

There are many ways to think of how to employ a methodology and many points along a project at which to choose different methods to get different kind of work done. There are also different kinds of project methods in use depending upon your country of origin. In the U.K., the majority of project professionals adhere to a PRINCE2 methodology, an acronym for **PR**ojects **IN C**ontrolled **E**nvironments, which addresses the larger questions of cost, time, resources, etc. Whereas in the U.S., a number of certified project managers follow the Project Management Institute's PMBOK Guide for formal project management practice. It's a book that collects the processes used in classic project management, with fundamental practices needed to achieve organizational results.

On a more tactical level, in both of those methodologies, the projects tend to adhere to the Waterfall method of planning project schedules according to a fixed, cascading plan. It is a classic model, one in which the executives and stakeholders on the project will want to see fixed costs and schedules, typically used for large infrastructure projects such as bridges, tunnels, construction and manufacturing.

# Waterfall Model Design

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

Following is a diagrammatic representation of different phases of waterfall model:

The sequential phases in Waterfall model are:

- **Requirements:** All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc.
- **Design:** The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture.
- **Development:** With inputs from 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.
- **Testing:** All the units developed in the Design phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment:** Once the functional and nonfunctional testing is done, the product is deployed in the customer environment or released into the market.
- **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.


All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model phases do not overlap.

# Waterfall Model Application

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are:

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

# Waterfall Model Pros & Cons

## Advantage

The advantage of waterfall development is that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one.

Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order.

## Disadvantage

The disadvantage of waterfall development is that it does not allow for much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.

The following table lists out the pros and cons of Waterfall model:

| Pros | Cons |
|---|---|
| • Simple and easy to understand and use.<br>• Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.<br>• Phases are processed and completed one at a time.<br>• Works well for smaller projects where requirements are very well understood.<br>• Clearly defined stages.<br>• Well understood milestones.<br>• Easy to arrange tasks.<br>• Process and results are well documented. | • No working software is produced until late during the life cycle.<br>• High amounts of risk and uncertainty.<br>• Not a good model for complex and object-oriented projects.<br>• Poor model for long and ongoing projects.<br>• Not suitable for the projects where requirements are at a moderate to high risk of changing. So risk and uncertainty is high with this process model.<br>• It is difficult to measure progress within stages.<br>• Cannot accommodate changing requirements.<br>• Adjusting scope during the life cycle can end a project.<br>• Integration is done as a "big-bang. At the very end, which doesn't allow identifying any technological or business bottleneck or challenges early. |

# Iterative Waterfall Model

As mentioned above, one of the disadvantages of the Model is that it does not allow for much reflection or revision; to get around this Waterfall can also be applied in an iterative manner.

An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which can then be reviewed in order to identify further requirements. This process is then repeated, producing a new version of the software for each cycle of the model.


For each cycle of the model shown above, a decision has to be made as to whether the software produced by the cycle will be discarded, or kept as a starting point for the next cycle (sometimes referred to as incremental prototyping). Eventually a point will be reached where the requirements are complete and the software can be delivered, or it becomes impossible to enhance the software as required, and a fresh start has to be made.

The iterative life cycle model can be likened to producing software by successive approximation. Drawing an analogy with mathematical methods that use successive approximation to arrive at a final solution, the benefit of such methods depends on how rapidly they converge on a solution.

The key to successful use of an iterative software development life cycle is rigorous validation of requirements, and verification (including testing) of each version of the software against those requirements within each cycle of the model.

The first three phases of the Iterative Waterfall Model is in fact an abbreviated form of the Waterfall Model of development. Each cycle of the model produces software that requires testing at the unit level, for software integration, for system integration and for acceptance. As the software evolves through successive cycles, tests have to be repeated and extended to verify each version of the software.

But one thing to keep in mind is that in the Iterative Waterfall Model the team is doing the same thing but they are treating each story as a miniature project. They do all the analysis for one story, then all the design for one story, then all the coding and testing for one story. This is an iterative waterfall process and should not be confused with an agile process.


Let's look at a simple example to illustrate iterative waterfall. When we work iteratively we create rough product or product piece in one iteration, then review it and improve it in next iteration and so on until it's finished. When we're creating a painting, in the first iteration the whole painting is sketched roughly, then in the second iteration colors are filled and in the third iteration finishing is done. Hence, in iterative model the whole product is developed step by step.

When to use iterative model:

- Requirements of the complete system are clearly defined and understood.
- When the project is big.
- Major requirements must be defined; however, some details can evolve with time.

The following table lists out the pros and cons of Iterative Waterfall model:

| Pros | Cons |
|---|---|
| • In iterative model we can only create a high-level design of the application before we actually begin to build the product and define the design solution for the entire product. Later on we can design and built a skeleton version of that, and then evolved the design based on what had been built.<br>• In iterative model we are building and improving the product step by step. Hence we can track the defects at early stages. This avoids the downward flow of the defects.<br>• In iterative model we can get the reliable user feedback. When presenting sketches and blueprints of the product to users for their feedback, we are effectively asking them to imagine how the product will work.<br>• In iterative model less time is spent on documenting and more time is given for designing. | • Each phase of an iteration is rigid with no overlaps.<br>• Costly system architecture or design issues may arise because not all requirements are gathered up front for the entire lifecycle. |

## Acknowledgements

These articles were in part originally published by TutorialsPoint.com



3rd Floor, Vamsiram's Jyothi Celestia, Kavuri Hills, Phase-2, Madhapur, Hyderabad, INDIA-500081