Dr S.Vengatesh Kumar
Associate Professor, Department of Computer Applications

UNIT I

SOFTWARE ENGINEERING

1. System software

   If you think of software as being in layers, the system software is the bottom layer: it sits between the hardware and the application software.

   Operating systems like Windows, macOS, Android and iOS are examples of system software. Operating systems are loaded into RAM when the device starts up, and have access to the hard drive.

   2. Utility software

   Utility software is part of the system software and performs specific tasks to keep the computer running. Utility software is always running in the background. Examples of utility software are security and optimisation programs.

   Security programs include anti-virus software that scans and removes viruses. Most computers will include some sort of anti-virus software, but you can add your own.

   Optimisation programs can include tools for system clean-up, disk defragmentation, and file compression. These tools are typically installed as part of the operating system. They have access to the hard drive to keep it tidy.

2. 3.The operating system

   Application software also uses the operating system to talk to the hardware on the computer and to other software. When a web browser wants to load a web page, it is the operating system that controls access to the internet and fetches the information from the web.

   Similarly, the operating system also provides the application software information about what key is being pressed, and about the mouse: where it is, what it clicked, and where it's moving on screen.

   Application software relies heavily on the operating system to do these tasks and send it all this information.

3. Application software

   This is everything else! Anything that is not an operating system or a utility is an application or app. So a word processor, spreadsheet, web browser, and graphics software are all examples of application software, and they can do many specific tasks.

   You can remove and add applications on your computer using the operating system.

Application software like a word processor regularly directs the operating system to load and save files from and to the hard drive. When you are working on a file, it is saved temporarily in the RAM. It is only when you choose to save it that it is written to the hard drive.

This is why, if the computer crashes while you're working on a file, you may lose any changes you didn't save. Data stored in the RAM is volatile. The data is lost when the RAM loses power

Changing Nature Of Software

There are seven broad categories of computer software having different forms, uses and challenges to software engineers.

1. System Software: It is a collection of programs written to service other programs. Some system software are compilers, editors, file management utilities. They process complex but determinate information structure. Other system applications can be operating system components, drivers, networking software. They process largely indeterminate data. In either case the system software area is characterized by heavy interaction with computer hardware, heavy usage by multiple users, concurrent operation that requires scheduling, resource sharing, and sophisticated process management, complex data structure, and multiple external interfaces.

2. Application software: These consist of standalone programs that solve a specific business need. Applications in this area process business or technical data in a way that facilitates business operations or management/technical decision making. These are used to control business functions in real time.

3. Engineering/scientific software: These software range from astronomy to volcanology, from automotive stress analysis to space shuttle orbital dynamics and from molecular biology to automated manufacturing.

4. Embedded software: These software resides within a product or system and is used to implement and control features and functions for the end user and for the system itself. These software can perform limited and esoteric functions or provide significant function and control capability.

5. Product line software: These are designed to provide a specific capability for use by many different customers. They focus on a limited and esoteric market place. These software addresses mass consumer markets.

6. Web-applications: Web apps or web applications are evolving into sophisticated computing environments that not only provide standalone features, computing functions and content to the end user but also integrated with corporate databases and business applications.

7. Artificial intelligence software: It makes use of non numerical algorithms to solve complex problems that are not amenable to computations or straightaway analysis. Applications within this area include robotics, expert systems pattern recognition, artificial neural networks, theorem proving and game playing.

8. Ubiquitous computing: The rapid growth of wireless networking may soon lead to true distributed computing. The challenge for software engineers will be develop systems and application software that will allow small devices, personal computers and enterprise system to communicate across vast network.

9. Net sourcing: The www is rapidly becoming a computing engine as well as content provider. The challenge for software engineers is to architect simple and sophisticated applications that provide benefit to target end user markets worldwide.

10. Open source: The challenge for software engineers to build source code that is self descriptive but more importantly to develop techniques that will enable both customers and developers to know what changes have been made and how those changes manifest themselves within the software.

In the new era the challenge for software engineers is to build applications that will facilitate mass communication and mass product distribution using the concepts that are only now forming.

Cloud Computing Architecture

As we know, cloud computing technology is used by both small and large organizations to store the information in cloud and access it from anywhere at anytime using the internet connection.

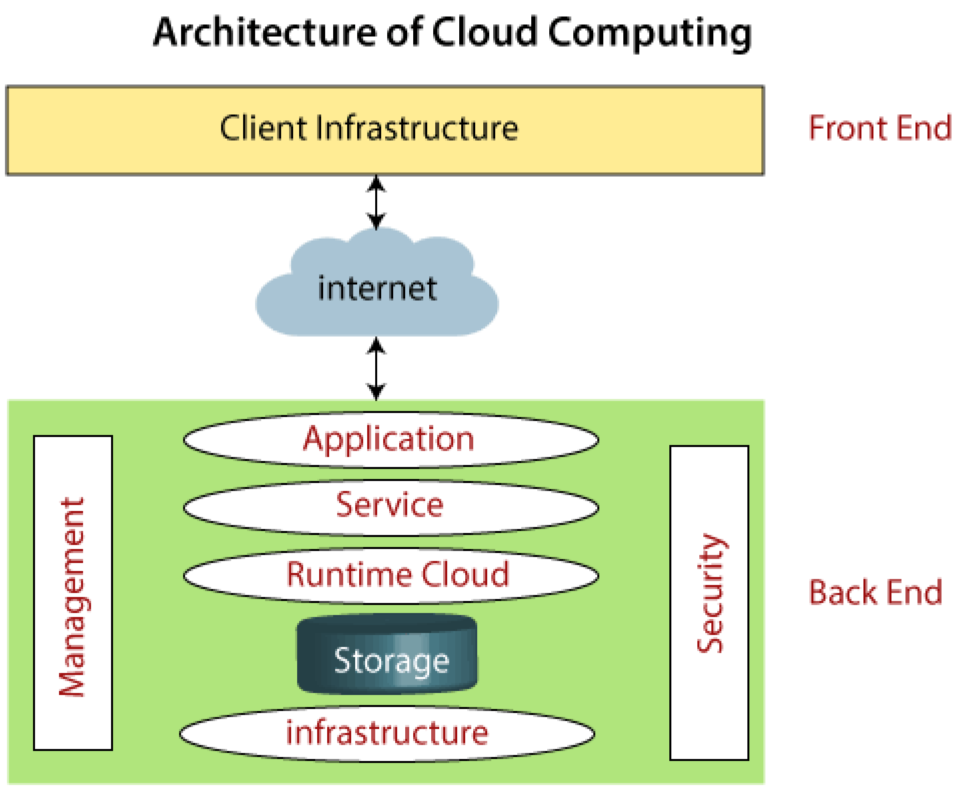
Cloud computing architecture is a combination of service-oriented architecture and event-driven architecture.

Cloud computing architecture is divided into the following two parts -

Front End

Back End

The below diagram shows the architecture of cloud computing -



Front End

The front end is used by the client. It contains client-side interfaces and applications that are required to access the cloud computing platforms. The front end includes web servers (including Chrome, Firefox, internet explorer, etc.), thin & fat clients, tablets, and mobile devices.

Back End

The back end is used by the service provider. It manages all the resources that are required to provide cloud computing services. It includes a huge amount of data storage, security mechanism, virtual machines, deploying models,

SOFTWARE ENGINEERING

Software is a set of instructions, data or programs used to operate computers and execute specific tasks. It is the opposite of hardware, which describes the physical aspects of a computer. Software is a generic term used to refer to applications, scripts and programs that run on a device. It can be thought of as the variable part of a computer, while hardware is the invariable part.

The two main categories of software are application software and system software. An application is software that fulfils a specific need or performs tasks. System software is designed to run a computer's hardware and provides a platform for applications to run on top of.

Other types of software include programming software, which provides the programming tools software developers need; middleware, which sits between system software and applications; and driver software, which operates computer devices and peripherals.

Early software was written for specific computers and sold with the hardware it ran on. In the 1980s, software began to be sold on floppy disks, and later on CDs and DVDs. Today, most software is purchased and directly downloaded over the internet. Software can be found on vendor websites or application service provider websites.

Examples and types of software

Among the various categories of software, the most common types include the following:

Application software. The most common type of software, application software is a computer software package that performs a specific function for a user, or in some cases, for another application. An application can be self-contained, or it can be a group of programs that run the application for the user. Examples of modern applications include office suites, graphics software, databases and database management programs, web browsers, word processors, software development tools, image editors and communication platforms.

Examples and types of software

System software. These software programs are designed to run a computer's application programs and hardware. System software coordinates the activities and functions of the hardware and software. In addition, it controls the operations of the computer hardware and provides an environment or platform for all the other types of software to work in. The OS is the best example of system software; it manages all the other computer programs. Other examples of system software include the firmware, computer language translators and system utilities.

Examples and types of software

Driver software. Also known as device drivers, this software is often considered a type of system software. Device drivers control the devices and peripherals connected to a computer, enabling them to perform their specific tasks. Every device that is connected to a computer needs at least one device driver to function. Examples include software that comes with any nonstandard hardware, including special game controllers, as well as the software that enables standard hardware, such as USB storage devices, keyboards, headphones and printers.

Examples and types of software

Middleware. The term middleware describes software that mediates between application and system software or between two different kinds of application software. For example, middleware enables Microsoft Windows to talk to Excel and Word. It is also used to send a remote work request from an application in a computer that has one kind of OS, to an application in a computer with a different OS. It also enables newer applications to work with legacy ones.

Examples and types of software

Programming software. Computer programmers use programming software to write code. Programming software and programming tools enable developers to develop, write, test and debug other software programs. Examples of programming software include assemblers, compilers, debuggers and interpreters.

IST OUT THE RULES OF THUMB FOR CREATING AN ANALYSE MODEL

The Four Phases

The life of a software system can be represented as a series of cycles. A cycle ends with the release of a version of the system to customers.

Within the Unified Process, each cycle contains four phases. A phase is simply the span of time between two major milestones, points at which managers make important decisions about whether to proceed with development and, if so, what's required concerning project scope, budget, and schedule.

1. Inception

The primary goal of the Inception phase is to establish the case for the viability of the proposed system.

The tasks that a project team performs during Inception include the following:

Defining the scope of the system (that is, what's in and what's out)

Outlining a candidate architecture, which is made up of initial versions of six different models Identifying critical risks and determining when and how the project will address them Starting to make the business case that the project is worth doing, based on initial estimates of cost, effort, schedule, and product quality The concept of candidate architecture is discussed in the section "Architecture-Centric" later in this chapter. The six models are covered in the next major section of this chapter, "The Five Workflows."

The major milestone associated with the Inception phase is called Life-Cycle Objectives. The indications that the project has reached this milestone include the following:

The major stakeholders agree on the scope of the proposed system.

The candidate architecture clearly addresses a set of critical high-level requirements.

The business case for the project is strong enough to justify a green light for continued development.

.

2. Elaboration

The primary goal of the Elaboration phase is to establish the ability to build the new system given the financial constraints, schedule constraints, and other kinds of constraints that the development project faces. The tasks that a project team performs during Elaboration include the following:

Capturing a healthy majority of the remaining functional requirements Expanding the candidate architecture into a full architectural baseline, which is an internal release of the system focused on describing the architecture Addressing significant risks on an ongoing basis Finalizing the business case for the project and preparing a project plan that contains sufficient detail to guide the next phase of the project (Construction)

2. Elaboration

The architectural baseline contains expanded versions of the six models initialized during the Inception phase.

The major milestone associated with the Elaboration phase is called Life-Cycle Architecture. The indications that the project has reached this milestone include the following:

Most of the functional requirements for the new system have been captured in the use case model.

The architectural baseline is a small, skinny system that will serve as a solid foundation for ongoing development.

The business case has received a green light, and the project team has an initial project plan that describes how the Construction phase will proceed.

3. Construction

The primary goal of the Construction phase is to build a system capable of operating successfully in beta customer environments.

During Construction, the project team performs tasks that involve building the system

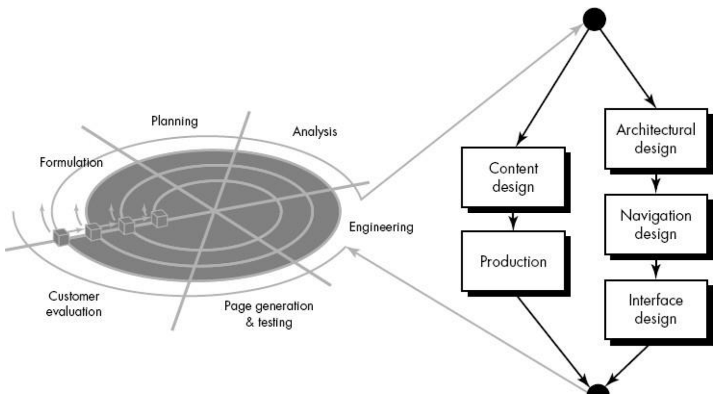
iteratively and incrementally (see "Iterations and Increments" later in this chapter), making sure that the viability of the system is always evident in executable form.
The major milestone associated with the Construction phase is called Initial Operational Capability. The project has reached this milestone if a set of beta customers has a more or less fully operational system in their hands.

4. Transition

The primary goal of the Transition phase is to roll out the fully functional system to customers.
During Transition, the project team focuses on correcting defects and modifying the system to correct previously unidentified problems.
The major milestone associated with the Transition phase is called Product Release.
Chapter 10 describes the details of the Transition phase

Software Engineering-A Framework for Web Applications

As Web Apps evolve from static, content-directed information sources to dynamic, user-directed application environments, the need to apply solid management andengineering principles grows in importance. To accomplish this, it is necessary to develop a Web E framework that encompasses an effective process model, populated by framework activities4 and engineering tasks. A process model for
Web E is suggested in figure.



The Web E process begins with a formulation—an activity that identifies the goals and objectives of the Web App and establishes the scope for the first increment. Planning estimates overall project cost, evaluates risks associated with the development effort, and defines a finely granulated development schedule for the initial Web App increment, with a more coarsely granulated schedule for subsequent increments. Analysis establishes technical requirements for the Web App and identifies the content items that will be incorporated. Requirements for graphic design (aesthetics) are also defined.

The engineering activity incorporates two parallel tasks illustrated on the right side of figure. Content design and production are tasks performed by nontechnical members of the Web E team. The intent of these tasks is to design, produce, and/or acquire all text, graphics, audio, and video content that are to become integrated into the Web App. At the same time, a set of technical design tasks are conducted.

Page generation is a construction activity that makes heavy use of automated tools for Web App creation. The content defined in the engineering activity is merged with the architectural, navigation, and interface designs to produce executable Web pages in HTML, XML, and other process-oriented languages (e.g., Java).Integration with component middleware (i.e., CORBA, DCOM, or JavaBeans) is also accomplished during this activity. Testing exercises Web App navigation; attempts to uncover errors in applets, scripts, and forms; and helps ensure that the Web App will operate correctly in different environments (e.g., with different browsers).

Each increment produced as part of the Web E process is reviewed during customer evaluation. This is the point at which changes are requested (scope extensions occur). These changes are integrated into the next path through the incremental process flow.

Spiral Model

The spiral model, initially proposed by Boehm, is an evolutionary software process model that couples the iterative feature of prototyping with the controlled and systematic aspects of the linear sequential model. It implements the potential for rapid development of new versions of the software. Using the spiral model, the software is developed in a series of incremental releases. During the early iterations, the additional release may be a paper model or prototype. During later iterations, more and more complete versions of the engineered system are produced.

The Spiral Model is shown in fig:

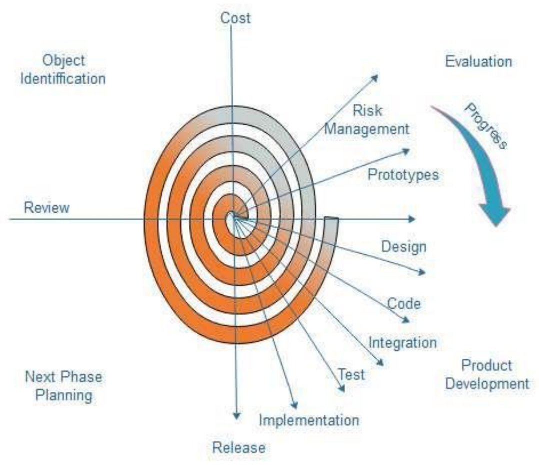


Fig. Spiral Model

The Spiral Model is shown in fig:

Each cycle in the spiral is divided into four parts:

Objective setting: Each cycle in the spiral starts with the identification of purpose for that cycle, the various alternatives that are possible for achieving the targets, and the constraints that exists.

Risk Assessment and reduction: The next phase in the cycle is to calculate these various alternatives based on the goals and constraints. The focus of evaluation in this stage is located on the risk perception for the project.

Development and validation: The next phase is to develop strategies that resolve uncertainties and risks. This process may include activities such as benchmarking, simulation, and prototyping.

Planning: Finally, the next step is planned. The project is reviewed, and a choice made whether to continue with a further period of the spiral. If it is determined to keep, plans are drawn up for the next step of the project.

The development phase depends on the remaining risks. For example, if performance or user-interface risks are treated more essential than the program development risks, the next phase may be an evolutionary development that includes developing a more detailed prototype for solving the risks.

The **risk-driven** feature of the spiral model allows it to accommodate any mixture of a specification-oriented, prototype-oriented, simulation-oriented, or another type of approach. An essential element of the model is that each period of the spiral is completed by a review that includes all the products developed during that cycle, including plans for the next cycle. The spiral model works for development as well as enhancement projects.

When to use Spiral Model?

o When deliverance is required to be frequent.

o When the project is large

o When requirements are unclear and complex

o When changes may require at any time

o Large and high budget projects

Advantages

o High amount of risk analysis

o Useful for large and mission-critical projects.

Disadvantages

o Can be a costly model to use.

o Risk analysis needed highly particular expertise

o Doesn't work well for smaller projects.

Umbrella Activities in Software Engineering

Umbrella activities are a set of steps or procedures that the software engineering team follows to maintain the progress, quality, change and risks of the overall software development process. The framework described in the generic view of Software Engineering is complemented by several umbrella activities.

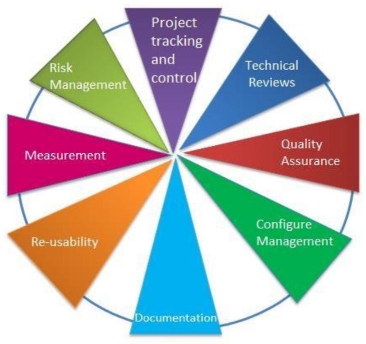


Fig: Umbrella Activities in Software Engineering

i. Software Project tracking and control: It allows the software team to assess progress against the project plan and take necessary action to maintain the schedule.

ii. Formal Technical Reviews: It assesses software engineering work products to uncover and remove errors before they are propagated to the next action or activity.

.

iii. Software Quality Assurance: It defines and conducts the activities required to ensure software quality. It is a methodology of checking the software development process with a predefined set of standards.

iv. Software Configure Management: It is the task of tracking and controlling changes in the software development part. It manages the effects of change throughout the software process.

v. Documentation: At first, all the project planning and other activities should be hard-copy documents and then the production gets started here

vi. Re-usability Management: It defines criteria for work product reuse and establishes mechanisms to achieve reusable components.

vii. Measurement: It defines process, project and product measures that assist the team in delivering software that meets customer's needs, it can be used in conjunction with all other frameworks.

viii. Risk Management: It assesses risks that may affect the outcome of the project or the quality of the product

UNIFIED PHASES

The Four Phases

The life of a software system can be represented as a series of cycles. A cycle ends with the release of a version of the system to customers.

Within the Unified Process, each cycle contains four phases. A phase is simply the span of time between two major milestones, points at which managers make important decisions about whether to proceed with development and, if so, what's required concerning project scope, budget, and schedule.

1.Inception

The primary goal of the Inception phase is to establish the case for the viability of the proposed system.

The tasks that a project team performs during Inception include the following:

Defining the scope of the system (that is, what's in and what's out)

Outlining a candidate architecture, which is made up of initial versions of six different models

Identifying critical risks and determining when and how the project will address them

Starting to make the business case that the project is worth doing, based on initial estimates of cost, effort, schedule, and product quality

.

1.Inception

The concept of candidate architecture is discussed in the section "Architecture-Centric" later in this chapter. The six models are covered in the next major section of this chapter, "The Five Workflows."

The major milestone associated with the Inception phase is called Life-Cycle Objectives. The indications that the project has reached this milestone include the following:

The major stakeholders agree on the scope of the proposed system.

The candidate architecture clearly addresses a set of critical high-level requirements.

The business case for the project is strong enough to justify a green light for continued development.

.

2. Elaboration

The primary goal of the Elaboration phase is to establish the ability to build the new system given the financial constraints, schedule constraints, and other kinds of constraints that the development project faces.

The tasks that a project team performs during Elaboration include the following:

Capturing a healthy majority of the remaining functional requirements

Expanding the candidate architecture into a full architectural baseline, which is an internal release of the system focused on describing the architecture

Addressing significant risks on an ongoing basis

Finalizing the business case for the project and preparing a project plan that contains sufficient detail to guide the next phase of the project (Construction)

2. Elaboration

The architectural baseline contains expanded versions of the six models initialized during the Inception phase.

The major milestone associated with the Elaboration phase is called Life-Cycle Architecture. The indications that the project has reached this milestone include the following:

Most of the functional requirements for the new system have been captured in the use case model.

The architectural baseline is a small, skinny system that will serve as a solid foundation for ongoing development.

The business case has received a green light, and the project team has an initial project plan that describes how the Construction phase will proceed.

3.Construction

The primary goal of the Construction phase is to build a system capable of operating successfully in beta customer environments.

During Construction, the project team performs tasks that involve building the system iteratively and incrementally (see "Iterations and Increments" later in this chapter), making sure that the viability of the system is always evident in executable form.

The major milestone associated with the Construction phase is called Initial Operational Capability. The project has reached this milestone if a set of beta customers has a more or less fully operational system in their hands.

Chapter 9 describes the details of the Construction phase.

4.Transition

The primary goal of the Transition phase is to roll out the fully functional system to customers.

During Transition, the project team focuses on correcting defects and modifying the system to correct previously unidentified problems.

The major milestone associated with the Transition phase is called Product Release.

Waterfall Model -

The Waterfall Model was the 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.

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.

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 this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

The following illustration is a representation of the different phases of the Waterfall Model. 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. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.

 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.

 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 - Advantages

The advantages of waterfall development are 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.

Some of the major advantages of the Waterfall Model are as follows −

- Simple and easy to understand and use
- Easy to manage due to the rigidity of the model. Each phase has
specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well
understood.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.

Waterfall Model - Disadvantages

The disadvantage of waterfall development is that it does not allow 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 major disadvantages of the Waterfall Model are as follows −

- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- 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.
- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.
- Adjusting scope during the life cycle can end a project.
- Integration is done as a "big-bang. at the very end, which doesn't
allow identifying any technological or business bottleneck or
challenges early.