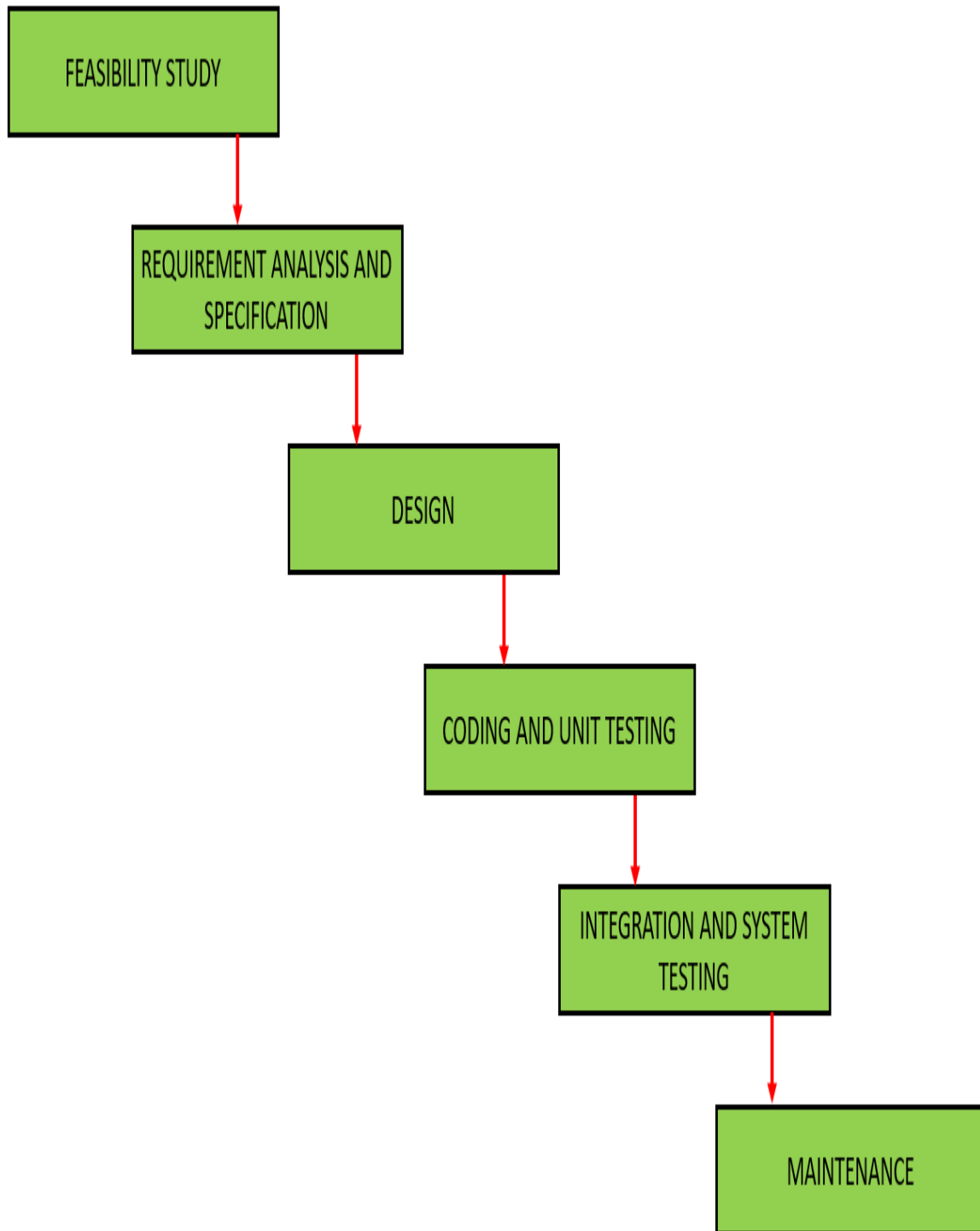# Software Engineering | Classical Waterfall Model

Classical waterfall model is the basic **software development life cycle** model. It is very simple but idealistic. Earlier this model was very popular but nowadays it is not used. But it is very important because all the other software development life cycle models are based on the classical waterfall model.
Classical waterfall model divides the life cycle into a set of phases. This model considers that one phase can be started after completion of the previous phase. That is the output of one phase will be the input to the next phase. Thus the development process can be considered as a sequential flow in the waterfall. Here the phases do not overlap with each other. The different sequential phases of the classical waterfall model

are shown in the below figure:

```
┌─────────────────────┐
│  FEASIBILITY STUDY  │
└──────────┬──────────┘
           │
           ▼
   ┌───────────────────────────┐
   │ REQUIREMENT ANALYSIS AND  │
   │      SPECIFICATION        │
   └─────────────┬─────────────┘
                 │
                 ▼
        ┌──────────────────┐
        │      DESIGN      │
        └─────────┬────────┘
                  │
                  ▼
          ┌─────────────────────────┐
          │ CODING AND UNIT TESTING │
          └────────────┬────────────┘
                       │
                       ▼
             ┌──────────────────────────┐
             │ INTEGRATION AND SYSTEM   │
             │         TESTING          │
             └────────────┬─────────────┘
                          │
                          ▼
                  ┌──────────────────┐
                  │   MAINTENANCE    │
                  └──────────────────┘
```

Let us now learn about each of these phases in brief details:

1. **Feasibility Study**: The main goal of this phase is to determine whether it would be financially and technically feasible to develop the software.

The feasibility study involves understanding the problem and then determine the various possible strategies to solve the problem. These different identified solutions are analyzed based on their benefits and drawbacks, The best solution is chosen and all the other phases are carried out as per this solution strategy.

2. **Requirements analysis and specification**: The aim of the requirement analysis and specification phase is to understand the exact requirements of the customer and document them properly. This phase consists of two different activities.

   - **Requirement gathering and analysis:** Firstly all the requirements regarding the software are gathered from the customer and then the gathered requirements are analyzed. The goal of the analysis part is to remove incompleteness (an incomplete requirement is one in which some parts of the actual requirements have been omitted) and inconsistencies (inconsistent requirement is one in which some part of the requirement contradicts with some other part).
   - **Requirement specification:** These analyzed requirements are documented in a software requirement specification (SRS) document. SRS document serves as a contract between development team and customers. Any future dispute between the customers and the developers can be settled by examining the SRS document.

3. **Design**: The aim of the design phase is to transform the requirements specified in the SRS document into a structure that is suitable for implementation in some programming language.

4. **Coding and Unit testing**: In coding phase software design is translated into source code using any suitable programming language. Thus each designed module is coded. The aim of the unit testing phase is to check whether each module is working properly or not.

5. **Integration and System testing**: Integration of different modules are undertaken soon after they have been coded and unit tested. Integration of various modules is carried out incrementally over a number of steps. During each integration step, previously planned modules are added to the partially integrated system and the resultant system is tested. Finally, after all the modules have been successfully integrated and tested, the full working system is obtained and system testing is carried out on this.
   System testing consists three different kinds of testing activities as described below :

   - **Alpha testing:** Alpha testing is the system testing performed by the development team.
   - **Beta testing:** Beta testing is the system testing performed by a friendly set of customers.
   - **Acceptance testing:** After the software has been delivered, the customer performed the acceptance testing to determine whether to accept the delivered software or to reject it.

6. **Maintainence:** Maintenance is the most important phase of a software life cycle. The effort spent on maintenance is the 60% of the total effort spent to develop a full software. There are basically three types of maintenance :
   - **Corrective Maintenance:** This type of maintenance is carried out to correct errors that were not discovered during the product development phase.
   - **Perfective Maintenance:** This type of maintenance is carried out to enhance the functionalities of the system based on the customer's request.
   - **Adaptive Maintenance:** Adaptive maintenance is usually required for porting the software to work in a new environment such as work on a new computer platform or with a new operating system.

Advantages of Classical Waterfall Model

Classical waterfall model is an idealistic model for software development. It is very simple, so it can be considered as the basis for other software development life cycle models. Below are some of the major advantages of this SDLC model:

- This model is very simple and is easy to understand.
- Phases in this model are processed one at a time.
- Each stage in the model is clearly defined.
- This model has very clear and well undestood milestones.
- Process, actions and results are very well documented.
- Reinforces good habits: define-before- design, design-before-code.
- This model works well for smaller projects and projects where requirements are well understood.

Drawbacks of Classical Waterfall Model

Classical waterfall model suffers from various shortcomings, basically we can't use it in real projects, but we use other software development lifecycle models which are based on the classical waterfall model. Below are some major drawbacks of this model:

- **No feedback path:** In classical waterfall model evolution of a software from one phase to another phase is like a waterfall. It assumes that no error is ever committed by developers during any phases. 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 actually customers' 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 new phase can start only after the completion of the previous phase. But in real projects, this can't be maintained. To increase the efficiency and reduce the cost, phases may overlap.