**Software Engineering :-**

* Software Engineering provides a standard procedure to design and develop a software.

**What is Software Engineering?**

* The term software engineering is the product of two words, software, and engineering.
* The software is a collection of integrated programs.
* Software subsists of carefully-organized instructions and code written by developers on any of various particular computer languages.
* Computer programs and related documentation such as requirements, design models and user manuals.
* **Engineering** is the application of **scientific** and **practical** knowledge to **invent, design, build, maintain**, and **improve frameworks, processes, etc**.
* **Software Engineering** is an engineering branch related to the evolution of software product using well-defined scientific principles, techniques, and procedures. The result of software engineering is an effective and reliable software product.
* IEEE defines software engineering as:
  + The application of a systematic,disciplined,quantifiable approach to the development,operation and maintenance of software; that is, the application of engineering to software.

**Why is Software Engineering required?**

* Software Engineering is required due to the following reasons:
  + To manage Large software
  + For more Scalability
  + Cost Management
  + To manage the dynamic nature of software
  + For better quality Management

**Objectives of Software Engineering**

1. Maintainability
2. Efficiency
3. Correctness
4. Reusability
5. Testability
6. Reliability
7. Portability
8. Adaptability
9. Interoperability

## Program vs. Software

* Software is more than programs. Any program is a subset of software, and it becomes software only if documentation & operating procedures manuals are prepared.

## Software Development Life Cycle (SDLC)

* A software life cycle model (also termed process model) is a pictorial and diagrammatic representation of the software life cycle. A life cycle model represents all the methods required to make a software product transit through its life cycle stages. It also captures the structure in which these methods are to be undertaken.



## The stages of SDLC are as follows:

* **Stage1: Planning and requirement analysis:-**
  1. The senior members of the team perform it with inputs from all the stakeholders and domain experts or SMEs in the industry.
  2. Planning for the quality assurance requirements and identifications of the risks associated with the projects is also done at this stage.
* **Stage2: Defining Requirements :-**
  1. Once the requirement analysis is done, the next stage is to certainly represent and document the software requirements and get them accepted from the project stakeholders.
  2. This is accomplished through "SRS"- Software Requirement Specification document which contains all the product requirements to be constructed and developed during the project life cycle.
* **Stage3: Designing the Software**
  1. The next phase is about to bring down all the knowledge of requirements, analysis, and design of the software project. This phase is the product of the last two, like inputs from the customer and requirement gathering.
* **Stage4:Coding**
  1. This step is also known as programming phase. The implementation of software design starts in terms of writing program code in the suitable programming language and developing error-free executable programs efficiently.
* **Stage5:Testing**
  1. After the code is generated, it is tested against the requirements to make sure that the products are solving the needs addressed and gathered during the requirements stage.
  2. During this stage, unit testing, integration testing, system testing, acceptance testing are done.
* **Stage6: Developing the project**
  1. In this phase of SDLC, the actual development begins, and the programming is built. The implementation of design begins concerning writing code. Developers have to follow the coding guidelines described by their management and programming tools like compilers, interpreters, debuggers, etc. are used to develop and implement the code.
* **Stage7: Deployment**
  1. Once the software is certified, and no bugs or errors are stated, then it is deployed.
  2. Then based on the assessment, the software may be released as it is or with suggested enhancement in the object segment.
  3. After the software is deployed, then its maintenance begins.
* **Stage8: Maintenance**
  1. Once when the client starts using the developed systems, then the real issues come up and requirements to be solved from time to time.
  2. This procedure where the care is taken for the developed product is known as maintenance.

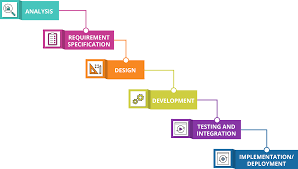
## SDLC Models

There are different software development life cycle models specify and design, which are followed during the software development phase. These models are also called "**Software Development Process Models**." Each process model follows a series of phase unique to its type to ensure success in the step of software development.

For detail info visit below site:-

https://www.tutorialspoint.com/sdlc/sdlc\_quick\_guide.htm

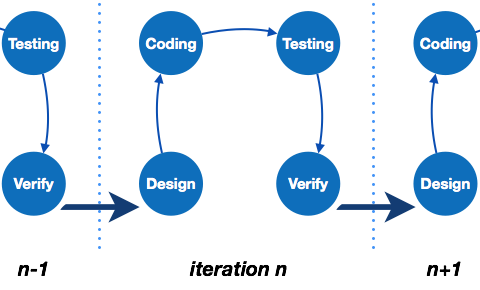
### **Waterfall Model**

****

* + Waterfall model is the simplest model of software development paradigm. It says the all the phases of SDLC will function one after another in linear manner. That is, when the first phase is finished then only the second phase will start and so on.
  + This model assumes that everything is carried out and taken place perfectly as planned in the previous stage and there is no need to think about the past issues that may arise in the next phase. This model does not work smoothly if there are some issues left at the previous step. The sequential nature of model does not allow us go back and undo or redo our actions.
  + This model is best suited when developers already have designed and developed similar software in the past and are aware of all its domains.
  + **Analysis** – To determine whether is would be financially and technically feasible to develop the software
  + **Requirements Specification** – to understand the exact requirement
  + **Design** – Transforming requirement specification into design models
  + **Development** – Actual coding and unit testing
  + **Testing and Integration** – Resultant system testing
  + **Maintenance/Deployement** – Resultant system at customer environment and changes in it.
  + Advantages
    1. The model is very simple and easy to understand
    2. Each phase in this model is processed one at a time
    3. It has very clear and well understood milestones
    4. Reinforces good habits like define-before-design and design-before-develop
    5. Model works well with smaller projects and projects where requirements are well/clearly understood
  + Drawbacks
    1. No feedback mechanism
    2. Difficult to accommodate change request
    3. No overlapping of phases
    4. Risk of failure

### **Iterative Model**

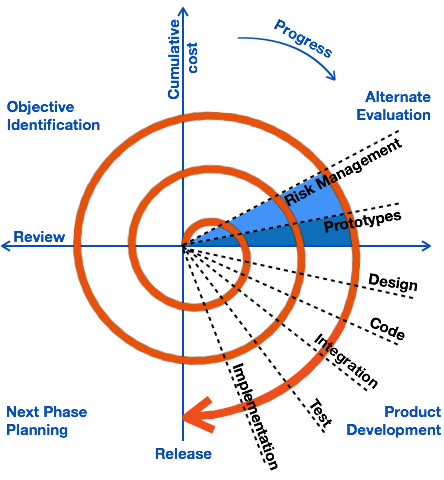
* + This model leads the software development process in iterations. It projects the process of development in cyclic manner repeating every step after every cycle of SDLC process.



* + The software is first developed on very small scale and all the steps are followed which are taken into consideration. Then, on every next iteration, more features and modules are designed, coded, tested and added to the software. Every cycle produces a software, which is complete in itself and has more features and capabilities than that of the previous one.
  + After each iteration, the management team can do work on risk management and prepare for the next iteration. Because a cycle includes small portion of whole software process, it is easier to manage the development process but it consumes more resources.

### **Spiral Model**

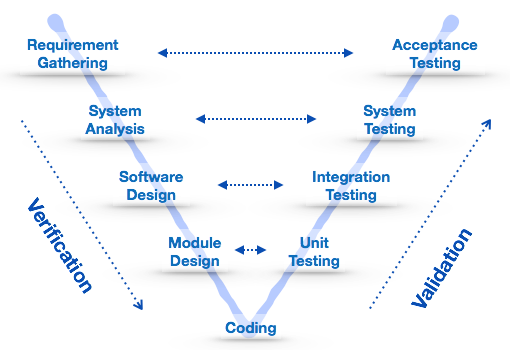
* + Spiral model is a combination of both, iterative model and one of the SDLC model. It can be seen as if you choose one SDLC model and combine it with cyclic process (iterative model).



* + This model considers risk, which often goes un-noticed by most other models. The model starts with determining objectives and constraints of the software at the start of one iteration. Next phase is of prototyping the software. This includes risk analysis. Then one standard SDLC model is used to build the software. In the fourth phase of the plan of next iteration is prepared.

### **V – model**

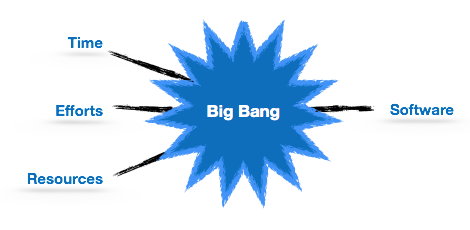
* + The major drawback of waterfall model is we move to the next stage only when the previous one is finished and there was no chance to go back if something is found wrong in later stages. V-Model provides means of testing of software at each stage in reverse manner.



* + At every stage, test plans and test cases are created to verify and validate the product according to the requirement of that stage. For example, in requirement gathering stage the test team prepares all the test cases in correspondence to the requirements. Later, when the product is developed and is ready for testing, test cases of this stage verify the software against its validity towards requirements at this stage.
  + This makes both verification and validation go in parallel. This model is also known as verification and validation model.

### **Big Bang Model**

* + This model is the simplest model in its form. It requires little planning, lots of programming and lots of funds. This model is conceptualized around the big bang of universe. As scientists say that after big bang lots of galaxies, planets and stars evolved just as an event. Likewise, if we put together lots of programming and funds, you may achieve the best software product.



* + For this model, very small amount of planning is required. It does not follow any process, or at times the customer is not sure about the requirements and future needs. So the input requirements are arbitrary.
  + This model is not suitable for large software projects but good one for learning and experimenting.

### [Incremental Model](https://www.javatpoint.com/software-engineering-incremental-model)



* + Characteristics of an Incremental module includes
    1. System development is broken down into many mini development projects
    2. Partial systems are successively built to produce a final total system
    3. Highest priority requirement is tackled first
    4. Once the requirement is developed, requirement for that increment are frozen

|  |  |
| --- | --- |
| **Incremental Phases** | **Activities performed in incremental phases** |
| **Requirement Analysis** | * Requirement and specification of the software are collected |
| **Design** | * Some high-end function are designed during this stage |
| **Code** | * Coding of software is done during this stage |
| **Test** | * Once the system is deployed, it goes through the testing phase |

* + When to use Incremental models?
  + Requirements of the system are clearly understood
  + When demand for an early release of a product arises
  + When software engineering team are not very well skilled or trained
  + When high-risk features and goals are involved
  + Such methodology is more in use for web application and product based companies
  + Advantages and Disadvantages of Incremental Model

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| The software will be generated quickly during the software life cycle | It requires a good planning designing |
| It is flexible and less expensive to change requirements and scope | Problems might cause due to system architecture as such not all requirements collected up front for the entire software lifecycle |
| Throughout the development stages changes can be done | Each iteration phase is rigid and does not overlap each other |
| This model is less costly compared to others | Rectifying a problem in one unit requires correction in all the units and consumes a lot of time |
| A customer can respond to each building |  |
| Errors are easy to be identified |  |



**Requirement Engineering**

* **Requirements engineering (RE)** refers to the process of defining, documenting, and maintaining requirements in the engineering design process. Requirement engineering provides the appropriate mechanism to understand what the customer desires, analyzing the need, and assessing feasibility, negotiating a reasonable solution, specifying the solution clearly, validating the specifications and managing the requirements as they are transformed into a working system. Thus, requirement engineering is the disciplined application of proven principles, methods, tools, and notation to describe a proposed system's intended behavior and its associated constraints.

**Requirement Engineering Process**

* It is a four-step process, which includes -
  + Feasibility Study
  + Requirement Elicitation and Analysis
  + Software Requirement Specification
  + Software Requirement Validation
  + Software Requirement Management



1. Feasibility Study:

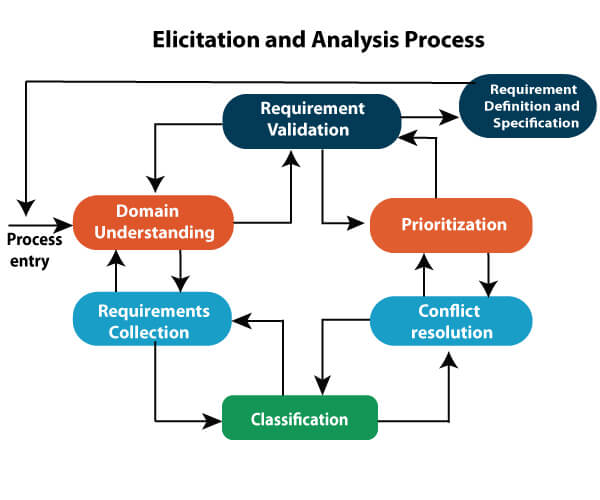
* + The objective behind the feasibility study is to create the reasons for developing the software that is acceptable to users, flexible to change and conformable to established standards.
  + **Types of Feasibility:**
    - **Technical Feasibility** - Technical feasibility evaluates the current technologies, which are needed to accomplish customer requirements within the time and budget.
    - **Operational Feasibility** - Operational feasibility assesses the range in which the required software performs a series of levels to solve business problems and customer requirements.
    - **Economic Feasibility** - Economic feasibility decides whether the necessary software can generate financial profits for an organization.

2. Requirement Elicitation and Analysis:

* + This is also known as the **gathering of requirements**. Here, requirements are identified with the help of customers and existing systems processes, if available.
  + Analysis of requirements starts with requirement elicitation. The requirements are analyzed to identify inconsistencies, defects, omission, etc. We describe requirements in terms of relationships and also resolve conflicts if any.

Triggers in SQL (Hindi)

* + **Problems of Elicitation and Analysis**
    - Getting all, and only, the right people involved.
    - Stakeholders often don't know what they want
    - Stakeholders express requirements in their terms.
    - Stakeholders may have conflicting requirements.
    - Requirement change during the analysis process.
    - Organizational and political factors may influence system requirements.



3. Software Requirement Specification:

* + Software requirement specification is a kind of document which is created by a software analyst after the requirements collected from the various sources - the requirement received by the customer written in ordinary language. It is the job of the analyst to write the requirement in technical language so that they can be understood and beneficial by the development team.
  + The models used at this stage include ER diagrams, data flow diagrams (DFDs), function decomposition diagrams (FDDs), data dictionaries, etc.
    - **Data Flow Diagrams:** Data Flow Diagrams (DFDs) are used widely for modeling the requirements. DFD shows the flow of data through a system. The system may be a company, an organization, a set of procedures, a computer hardware system, a software system, or any combination of the preceding. The DFD is also known as a data flow graph or bubble chart.
    - **Data Dictionaries:** Data Dictionaries are simply repositories to store information about all data items defined in DFDs. At the requirements stage, the data dictionary should at least define customer data items, to ensure that the customer and developers use the same definition and terminologies.
    - **Entity-Relationship Diagrams:** Another tool for requirement specification is the entity-relationship diagram, often called an "*E-R diagram*." It is a detailed logical representation of the data for the organization and uses three main constructs i.e. data entities, relationships, and their associated attributes.

4. Software Requirement Validation:

* + After requirement specifications developed, the requirements discussed in this document are validated. The user might demand illegal, impossible solution or experts may misinterpret the needs. Requirements can be the check against the following conditions -
  + If they can practically implement
  + If they are correct and as per the functionality and specially of software
  + If there are any ambiguities
  + If they are full
  + If they can describe
  + **Requirements Validation Techniques**
    - Requirements reviews/inspections: systematic manual analysis of the requirements.
    - Prototyping: Using an executable model of the system to check requirements.
    - Test-case generation: Developing tests for requirements to check testability.
    - Automated consistency analysis: checking for the consistency of structured requirements descriptions.
* **Software Requirements:**
  + Largely software requirements must be categorized into two categories:
  + **Functional Requirements:**
    - Functional requirements define a function that a system or system element must be qualified to perform and must be documented in different forms. The functional requirements are describing the behavior of the system as it correlates to the system's functionality.
  + **Non-functional Requirements:**
    - This can be the necessities that specify the criteria that can be used to decide the operation instead of specific behaviors of the system.  
      Non-functional requirements are divided into two main categories:
    - **Execution qualities** like security and usability, which are observable at run time.
    - **Evolution qualities** like testability, maintainability, extensibility, and scalability that embodied in the static structure of the software system.