**UNIT-1**

**INTRODUCTION TO SOFTWARE ENGINEERING**

Software Engineering

The term **software engineering** is the product of two words, **software**, and **engineering**.

The **software** is a collection of integrated programs.

**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.

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

**Evolving role of software:**

Software has dual role

1 Product 2. Vehicle

1 Software is a product: It produce ,manage and display the information.

2 Software is a vehicle for delivering the product.

**Changing Nature of Software:**

Software is everywhere .It keeps on changing from one situation to other in order to accommodate the user choice and according to evolution of technology ,requirements and situation .

1. **System Software:** System software is a collection of programs that are written to service other programs. Some system software processes complex but determinate, information structures.

**Ex:** compiler ,operating system

1. **Application Software:** Application software is defined as programs that solve a specific business need. Application in this area processes business or technical data in a way that facilitates business operation or management technical decision-making. In addition to conventional data processing applications, application software is used to control business functions in real-time.

**Ex:Ms office**

1. **Engineering and Scientific Software:** This software is used to facilitate the engineering function and task. however modern applications within the engineering and scientific area are moving away from conventional numerical algorithms. Computer-aided design, system simulation, and other interactive applications have begun to take a real-time and even system software characteristic.

**Ex:** Calculus,CAD

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

**Ex:** Microwave Oven

1. **Product-line Software:** Designed to provide a specific capability for use by many customers, product-line software can focus on the limited and esoteric marketplace or address the mass consumer market.

**Ex:** DBMS

1. **Web Application:** It is a client-server computer program that the client runs on the web browser. In their simplest form, Web apps can be little more than a set of linked hypertext files that present information using text and limited graphics.

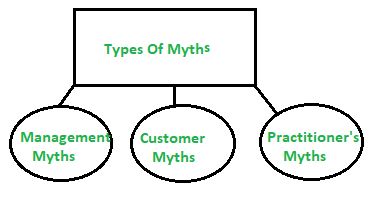
**Ex:** Websites

1. **Artificial Intelligence Software:** Artificial intelligence software makes use of a nonnumerical algorithm to solve a complex problem that is not amenable to computation or straightforward analysis. Applications within this area include robotics, expert systems, pattern recognition, artificial neural networks, theorem proving, and game playing.

**Ex: Robotics, Speech Recognition**

**Software Myths:**

Most, experienced experts have seen myths or superstitions (false beliefs or interpretations) or misleading attitudes (naked users) which creates major problems for management and technical people. The types of software-related myths are listed below.



*`Types of Software Myths*

**(i) Management Myths:**

**Myth 1:**

We have all the standards and procedures available for software development.

Fact:

* Software experts do not know all the requirements for the software development.
* And all existing processes are incomplete as new software development is based on new and different problem.

**Myth 2:**

The addition of the latest hardware programs will improve the software development.

Fact:

* The role of the latest hardware is not very high on standard software development; instead (CASE) Engineering tools help the computer, they are more important than hardware to produce quality and productivity.
* Hence, the hardware resources are misused.

**Myth 3:**

* With the addition of more people and program planners to Software development can help meet project deadlines (If lagging behind).

Fact:

* If software is late, adding more people will merely make the problem worse. This is because the people already working on the project now need to spend time educating the newcomers, and are thus taken away from their work. The newcomers are also far less productive than the existing software engineers, and so the work put into training them to work on the software does not immediately meet with an appropriate reduction in work.

**(ii)Customer Myths:**

The customer can be the direct users of the software, the technical team, marketing / sales department, or other company. Customer has myths leading to false expectations (customer) & that’s why you create dissatisfaction with the developer.

**Myth 1:**

A general statement of intent is enough to start writing plans (software development) and details of objectives can be done over time.

Fact:

* Official and detailed description of the database function, ethical performance, communication, structural issues and the verification process are important.
* Unambiguous requirements (usually derived iteratively) are developed only through effective and continuous  
  communication between customer and developer.

**Myth 2:**

Software requirements continually change, but change can be easily accommodated because software is flexible

Fact:

* It is true that software requirements change, but the impact of change varies with the time at which it is introduced. When requirements changes are requested early (before design or code has been started), the cost impact is relatively small. However, as time passes, the cost impact grows rapidly—resources have been committed, a design framework has been established, and change can cause upheaval that requires additional resources and major design modification.

**(iii)Practitioner’s Myths:**

**Myths 1:**

They believe that their work has been completed with the writing of the plan.

Fact:

* It is true that every 60-80% effort goes into the maintenance phase (as of the latter software release). Efforts are required, where the product is available first delivered to customers.

**Myths 2:**

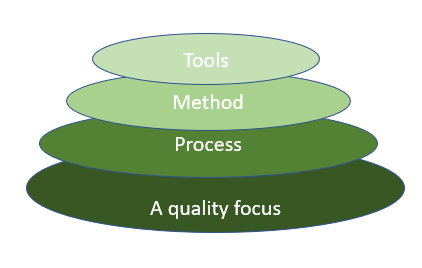
There is no other way to achieve system quality, until it is “running”.

Fact:

* Systematic review of project technology is the quality of effective software verification method. These updates are quality filters and more accessible than test.

Software Engineering A Layered Technology

[Software engineering](https://www.geeksforgeeks.org/software-engineering-introduction-to-software-engineering/)is a fully layered technology, to develop software we need to go from one layer to another. All the layers are connected and each layer demands the fulfillment of the previous layer.

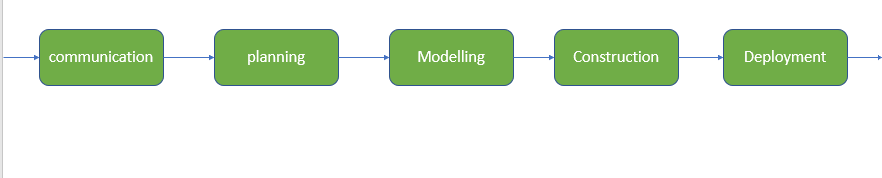


*Fig: The diagram shows the layers of software development*

### ****Layered technology is divided into four parts:****

**1.** **A quality focus:**It defines the continuous process improvement principles of software. It provides integrity that means providing security to the software so that data can be accessed by only an authorized person, no outsider can access the data. It also focuses on maintainability and usability.

**2. Process:**It is the foundation or base layer of software engineering. It is key that binds all the layers together which enables the development of software before the deadline or on time.  Process defines a framework that must be established for the effective delivery of software engineering technology. The software process covers all the activities, actions, and tasks required to be carried out for software development.



**Process activities are listed below:-**

* **Communication:** It is the first and foremost thing for the development of software. Communication is necessary to know the actual demand of the client.
* **Planning:**It basically means drawing a map for reduced the complication of development.
* **Modeling:**In this process, a model is created according to the client for better understanding.
* **Construction:**It includes the coding and testing of the problem.
* **Deployment:-** It includes the delivery of software to the client for evaluation and feedback.

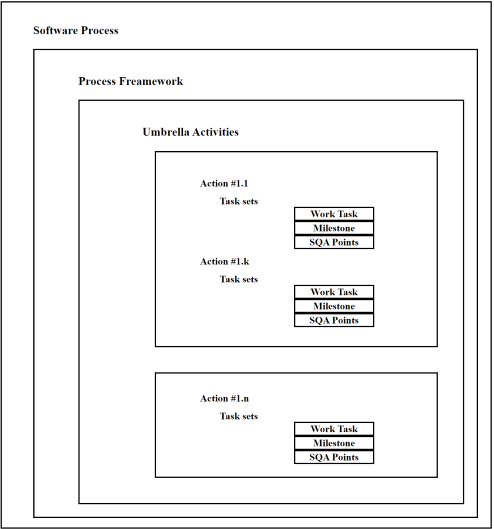
**3. Method:**During the process of software development the answers to all “how-to-do” questions are given by method. It has the information of all the tasks which includes communication, requirement analysis, design modeling, program construction, testing, and support.

**4.** **Tools:**Software engineering tools provide a self-operating system for processes and methods. Tools are integrated which means information created by one tool can be used by another.

**What is a Software Process Framework?**

Software Process Framework details the steps and chronological order of a process. Since it serves as a foundation for them, it is utilized in most applications. Task sets, umbrella activities, and process framework activities all define the characteristics of the software development process. Software Process includes:

1. **Tasks:** They focus on a small, specific objective.
2. **Action:**It is a set of tasks that produce a major work product.
3. **Activities:** Activities are groups of related tasks and actions for a major objective.



*Software Process Framework*

**Process Framework Activities**

The process framework is required for representing common process activities. Five framework activities are described in a process framework for software engineering. Communication, planning, modeling, construction, and deployment are all examples of framework activities. Each engineering action defined by a framework activity comprises a list of needed work outputs, project milestones, and software quality assurance (SQA) points.

**1. Communication**

By communication, customer requirement gathering is done. Communication with consumers and stakeholders to determine the system’s objectives and the software’s requirements.

**2. Planning**

Establish engineering work plan, describes technical risk, lists resources requirements, work produced and defines work schedule.

**3. Modeling**

Architectural models and design to better understand the problem and to work towards the best solution. The software model is prepared by:

* Analysis of requirements
* Design

**4. Construction**

Creating code, testing the system, fixing bugs, and confirming that all criteria are met. The software design is mapped into a code by:

* Code generation
* Testing

**5. Deployment**

In this activity, a complete or non-complete product or software is represented to the customers to evaluate and give feedback. On the basis of their feedback, we modify the product for the supply of better products.

**Umbrella Activities**

**F**ramework activities used for small project.

Umbrella activities used for large project

Umbrella Activities are that take place during a software development process for improved project management and tracking.

1. **Software project tracking and control:**This is an activity in which the team can assess progress and take corrective action to maintain the schedule. Take action to keep the project on time by comparing the project’s progress against the plan.
2. **Risk management:**The risks that may affect project outcomes or quality can be analyzed. Analyze potential risks that may have an impact on the software product’s quality and outcome.
3. **Software quality assurance:** These are activities required to maintain software quality. Perform actions to ensure the product’s quality.
4. **Formal technical reviews:**It is required to assess engineering work products to uncover and remove errors before they propagate to the next activity. At each level of the process, errors are evaluated and fixed.
5. **Software configuration management:**Managing of configuration process when any change in the software occurs.
6. **Work product preparation and production:** The activities to create models, documents, logs, forms, and lists are carried out.
7. **Reusability management:**It defines criteria for work product reuse. Reusable work items should be backed up, and reusable software components should be achieved.
8. **Measurement:** In this activity, the process can be defined and collected. Also, project and product measures are used to assist the software team in delivering the required software.

# Capability Maturity Model Integration (CMMI)

**Capability Maturity Model Integration (CMMI)** is a successor of CMM and is a more evolved model .It is a proven industry framework to improve product quality and development efficiency for both hardware and software.

It is the benchmark for measuring the maturity of an organization

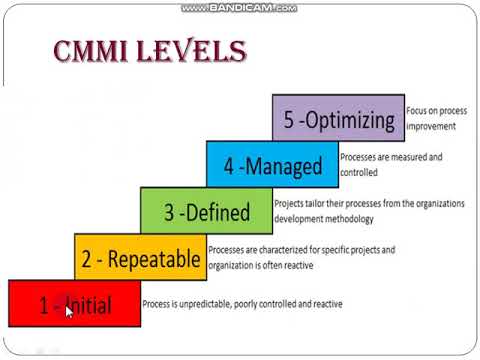
**Objectives of CMMI :**

1. Fulfilling customer needs and expectations.
2. Value creation for investors/stockholders.
3. Market growth is increased.
4. Improved quality of products and services.
5. Enhanced reputation in Industry.

**Levels of CMMI Maturity**

The five maturity levels are as follows:

1. **Initial**: The process at this level tends to be ad hoc and chaotic. A sense of unpredictability and uncertainty is involved.
2. **Managed:**Ensure proper project management as per requirements and processes are planned, measured, and controlled.
3. **Defined:** Processes are clearly characterized and understood, and are described in standards, procedures, tools, and methods.
4. **Quantitatively managed:**Quantitative objectives for process performance and quality are created for the purposes of managing the processes. Objectives are based on the requirements of customers and organizations.
5. **Optimized:**Continuous improvement of the process performance through both incremental and innovative technological improvements.



**process pattern:**

software process is defined as collection of patterns(provides a template). They can be used to describe a problem and solution associated with framework activity in some situations.

**Template:**

* **Pattern Name –** Meaningful name must be given to a pattern within context of software process (e.g. Technical Reviews).
* **Forces –** The issues that make problem visible and may affect its solution also environment in which pattern is encountered.

**Type:**

It is of three types :

1. **Stage pattern –** Problems associated with a framework activity for process are described by stage pattern. Establishing Communication might be an example of a staged pattern. This pattern would incorporate task pattern Requirements Gathering and others.
2. **Task-pattern –** Problems associated with a software engineering action or work task and relevant to successful software engineering practice (e.g., Requirements Gathering is a task pattern) are defined by task-pattern.
3. **Phase pattern –** Even when the overall flow of activities is iterative in nature, it defines sequence of framework activities that occurs within process. Spiral Model or Prototyping might be an example of a phase pattern.

**Software Process Assessment** (SPA) is a systematic evaluation of an organization’s software development processes. It aims to assess the effectiveness, efficiency, and maturity of these processes. Here are some key aspects of SPA:

1. **Purpose**:
   * Identify strengths and weaknesses in software processes.
   * Facilitate process improvement.
   * Enhance overall software quality and productivity.
2. **Types of Assessment**:
   * **Maturity Assessment**: Evaluates the maturity level of processes (e.g., CMMI levels).
   * **Capability Assessment**: Focuses on specific process areas.
   * **Compliance Assessment**: Ensures adherence to standards or guidelines.
3. **Steps in SPA**:
   * **Planning**: Define assessment scope, objectives, and criteria.
   * **Data Collection**: Gather information about processes, artifacts, and practices.
   * **Analysis**: Evaluate collected data against predefined criteria.
   * **Reporting**: Present findings, strengths, and areas for improvement.
   * **Recommendations**: Suggest actions for enhancing processes.
4. **Benefits**:
   * **Quality Improvement**: Identifies areas needing attention.
   * **Risk Reduction**: Mitigates risks associated with poor processes.
   * **Efficiency Enhancement**: Streamlines workflows.
   * **Organizational Learning**: Encourages continuous improvement.
5. **Personal Software Process (PSP)**:
   * **Focus**: PSP is centered around **individuals** to enhance their performance.
   * **Methodology**: It’s a **bottom-up approach** to software process improvement.
   * **Prescriptive**: PSP provides a well-defined set of tools and techniques.
   * **Key Features**:
     1. **Process-Focused**: Emphasizes disciplined software development.
     2. **Personalized**: Tailored to an individual’s skills and work habits.
     3. **Metrics-Driven**: Collects and analyzes data for progress measurement.
     4. **Incremental**: Breaks down development into manageable steps.
     5. **Quality-Focused**: Aims for high-quality, defect-free software.
   * **Advantages**:
     1. Improved productivity and quality.
     2. Personalized approach.
     3. Better estimation and continuous improvement.
   * **Disadvantages**:
     1. [Can be time-consuming and complex, especially during initial adoption](https://www.outlife.in/blog/team-development-performance-models).
6. **Team Software Process (TSP)**:
   * **Focus**: TSP emphasizes **team collaboration** throughout software development.
   * **Methodology**: It provides a structured approach to development.
   * **Key Features**:
     1. **Team-Focused**: Prioritizes collaboration and communication.
     2. **Process-Driven**: Follows a disciplined approach.
   * **Benefits**:
     1. Improved teamwork and communication.
     2. Risk reduction and efficiency enhancement.
     3. Organizational learning.
   * **Disadvantages**:
     1. Requires coordination and training.
     2. Achievable with effort and commitment

# 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

SDLC Cycle

SDLC Cycle represents the process of developing software. SDLC framework includes the following steps:



The stages of SDLC are as follows:

**Stage1: Planning and requirement analysis**

Requirement Analysis is the most important and necessary stage in SDLC.

The senior members of the team perform it with inputs from all the stakeholders and domain experts or SMEs in the industry.

Planning for the quality assurance requirements and identifications of the risks associated with the projects is also done at this stage.

Once the requirement is understood, the SRS (Software Requirement Specification) document is created. The developers should thoroughly follow this document and also should be reviewed by the customer for future reference.

**Stage2: Defining Requirements**

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.

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**

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: Developing the project**

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.

**Stage5: Testing**

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.

During this stage, unit testing, integration testing, system testing, acceptance testing are done.

**Stage6: Deployment**

Once the software is certified, and no bugs or errors are stated, then it is deployed.

After the software is deployed, then its maintenance begins.

**Stage7: Maintenance**

Once when the client starts using the developed systems, then the real issues come up and requirements to be solved from time to time.

This procedure where the care is taken for the developed product is known as maintenance.

**Objectives of SDLC**

**1.To ensure that the software is of high quality:**The SDLC includes testing and quality assurance phases, which help to ensure that the software is free of bugs and that it meets the requirements.

**2.[To manage risks and costs:](https://www.geeksforgeeks.org/software-risk-analysis/)**The SDLC helps organizations to identify and manage risks early in the development process, which can help to reduce costs and minimize the impact of any issues that do arise.

**3.To improve communication and collaboration:**The SDLC helps to ensure that all stakeholders, including customers, end-users, and developers, are involved in the development process and that their needs are taken into account.

**4.To improve efficiency and productivity:**The SDLC helps organizations to optimize the use of resources and to streamline the development process, which can improve efficiency and productivity.

**5.To increase the likelihood of a successful project outcome:** Following a well-defined SDLC process can greatly increase the chances of success of the project, as the process guides the team towards the goal in a systematic and efficient way.

**Process Models**

**Waterfall model**

**Incremental model**

**Prototype model**

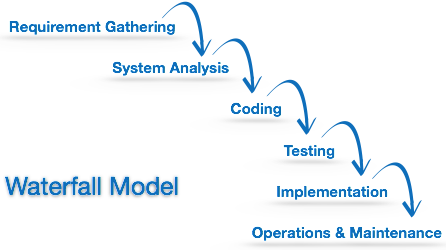
**Spiral model**

**1. Waterfall Model**

The waterfall model is a linear and sequential model, which means that a development phase cannot begin until the previous phase is completed. We cannot overlap phases in waterfall model.

Similarly waterfall model also works, once one phase of development is completed then we move to the next phase but cannot go back to the previous phase.

In the waterfall model, the output of one phase serves as the input for the other phase.

**Phases of Waterfall mode:**

**Advantages**

* This model is simple and easy to understand.
* This is very useful for small projects.
* This model is easy to manage.
* The end goal is determined early.
* Each phase of this model is well explained.
* This is a base model.
* **Disadvantages**
* In this model, complete and accurate requirements are expected at the beginning of the development process.
* We cannot go back to the previous phase due to which it is very difficult to change the requirements.
* Risk is not assessed in this, hence there is high risk and uncertainty in this model.
* Due to its sequential nature this model is not realistic in today’s world.
* This is not a good model for large and complex projects.

**INCREMENTAL MODEL**

Incremental Model is a process of software development where requirements divided into multiple standalone modules of the software development cycle.

In this model, each module goes through the requirements, design, implementation and testing phases. Every subsequent release of the module adds function to the previous release. The process continues until the complete system achieved.



## The various phases of incremental model are as follows:

**1. Requirement analysis:** In the first phase of the incremental model, the product analysis expertise identifies the requirements. And the system functional requirements are understood by the requirement analysis team. To develop the software under the incremental model, this phase performs a crucial role.

**2. Design & Development:** In this phase of the Incremental model of SDLC, the design of the system functionality and the development method are finished with success. When software develops new practicality, the incremental model uses style and development phase.

**3. Testing:** In the incremental model, the testing phase checks the performance of each existing function as well as additional functionality. In the testing phase, the various methods are used to test the behavior of each task.

**4. Implementation:** Implementation phase enables the coding phase of the development system. It involves the final coding that design in the designing and development phase and tests the functionality in the testing phase. After completion of this phase, the number of the product working is enhanced and upgraded up to the final system product

## When we use the Incremental Model?

* When the requirements are superior.
* A project has a lengthy development schedule.
* When Software team are not very well skilled or trained.
* When the customer demands a quick release of the product.
* You can develop prioritized requirements first.

## Advantage of Incremental Model

* Errors are easy to be recognized.
* Easier to test and debug
* More flexible.
* Simple to manage risk because it handled during its iteration.
* The Client gets important functionality early.

## Disadvantage of Incremental Model

* Need for good planning
* Total Cost is high.
* Well defined module interfaces are needed.

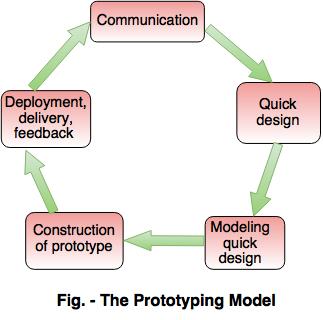
Evolutionary Process Models

* Evolutionary models are iterative type models.
* They allow to develop more complete versions of the software.

**Following are the evolutionary process models.**  
  
1. The prototyping model  
2. The spiral model

1. The Prototyping model

* Prototype is defined as first or preliminary form using which other forms are copied or derived.
* Prototype model is a set of general objectives for software.
* It does not identify the requirements like detailed input, output.
* It is software working model of limited functionality.
* In this model, working programs are quickly produced.

  
  
**The different phases of Prototyping model are:**  
  
**1.Communication**  
In this phase, developer and customer meet and discuss the overall objectives of the software.  
  
**2. Quick design**

* Quick design is implemented when requirements are known.
* It includes only the important aspects like input and output format of the software.
* It focuses on those aspects which are visible to the user rather than the detailed plan.
* It helps to construct a prototype.

**3. Modeling quick design**

* This phase gives the clear idea about the development of software because the software is now built.
* It allows the developer to better understand the exact requirements.

**4. Construction of prototype**  
The prototype is evaluated by the customer itself.  
  
**5. Deployment, delivery, feedback**

* If the user is not satisfied with current prototype then it refines according to the requirements of the user.
* The process of refining the prototype is repeated until all the  requirements of users are met.
* When the users are satisfied with the developed prototype then the system is developed on the basis of final prototype.

**Advantages of Prototyping Model**

* Prototype model need not know the detailed input, output, processes, adaptability of operating system and full machine interaction.
* In the development process of this model users are actively involved.
* The development process is the best platform to understand the system by the user.
* Errors are detected much earlier.
* Gives quick user feedback for better solutions.
* It identifies the missing functionality easily. It also identifies the confusing or difficult functions.

**Disadvantages of Prototyping Model:**

* The client involvement is more and it is not always considered by the developer.
* It is a slow process because it takes more time for development.
* Many changes can disturb the rhythm of the development team.
* It is a thrown away prototype when the users are confused with it.

2. The Spiral model

* Spiral model is a risk driven process model.
* It is used for generating the software projects.
* In spiral model, an alternate solution is provided if the risk is found in the risk analysis, then alternate solutions are suggested and implemented.
* It is a combination of prototype and sequential model or waterfall model.
* In one iteration all activities are done, for large project's the output is small.

Determine Objectives Identify and Resolve Risk

Customer Evaluation of prototype Develope next level

Fig: Spiral Model

**Advantages of Spiral Model**

* It reduces high amount of risk.
* It is good for large and critical projects.
* It gives strong approval and documentation control.
* In spiral model, the software is produced early in the life cycle process.

**Disadvantages of Spiral Model**

* It can be costly to develop a software model.
* It is not used for small projects.
* It needs better communication between the team members. This may not be achieved all the time.
* It requires to remember the status of the different activities.