Software Engineering unit 1

Software Engineering (Dr. A.P.J. Abdul Kalam Technical University)

Software Engineering UNIT-1

Software Engineering

- We can define software engineering as an engineering branch associated with the development of software product using well defined scientific principles, methods and procedures.
- The outcome of software engineering is an efficient and reliable software product. Software engineering helps to reduce this programming complexity.
- Software engineering is the branch of computer science that deals with the design, development, testing, and maintenance of software applications.
- Software engineers apply engineering principles and knowledge of programming languages to build software solutions for end users.

NEED OF SOFTWARE ENGINEERING

The need of software engineering arises because of higher rate of change in user requirements and environment on which the software is working.

Large software - It is easier to build a wall than to a house or building, likewise, as the size of software become large engineering has to step to give it a scientific process.

Scalability- Scalability is the measure of a system's ability to increase or decrease in performance and cost in response to changes in application and system processing demands.

Dynamic Nature- The always growing and adapting nature of software hugely depends upon the environment in which the user works. If the nature of software is always changing, new enhancements need to be done in the existing one. This is where software engineering plays a good role.

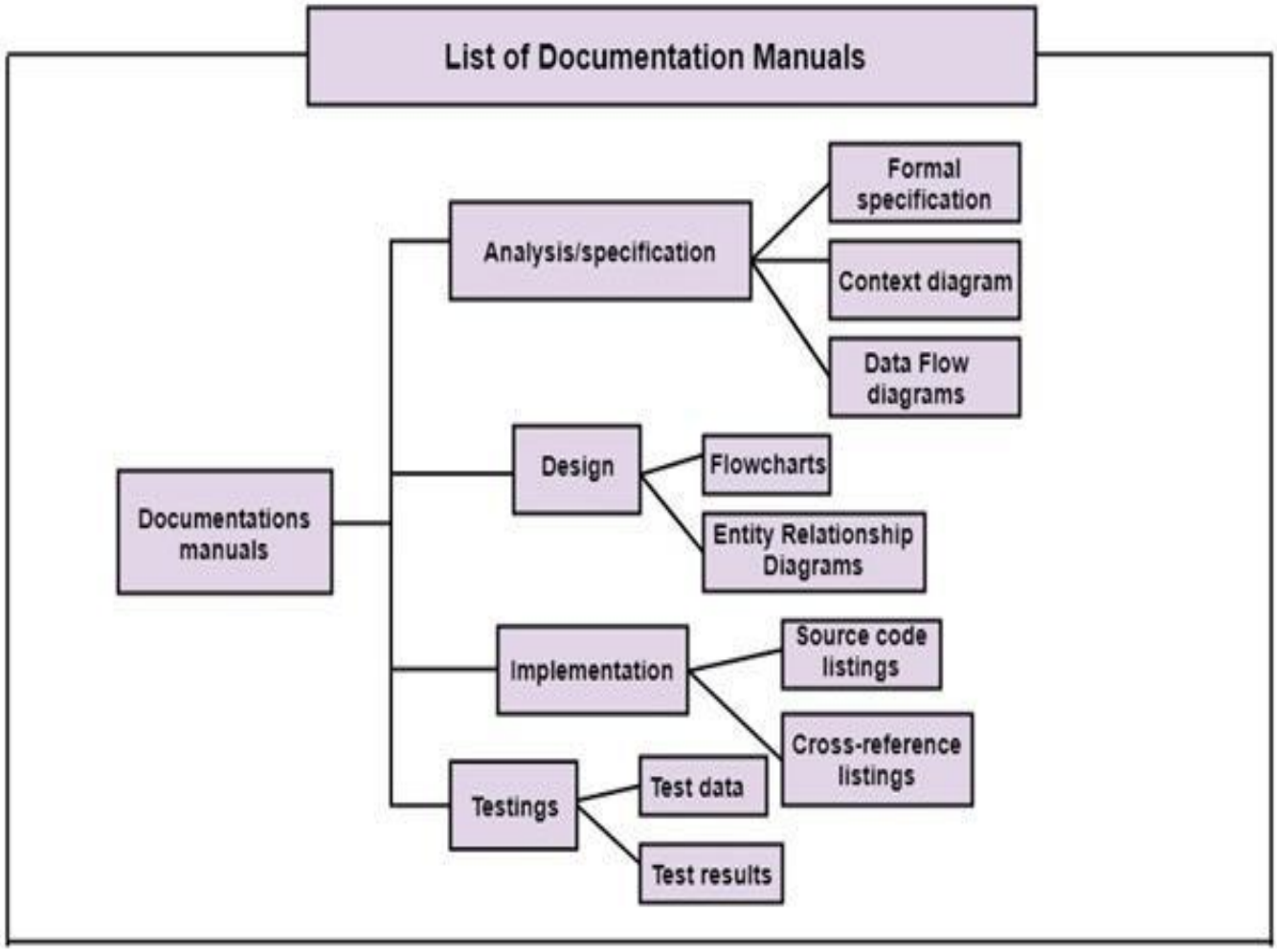
Quality Management - Better process of software development provides better and quality software product.

Cost- As hardware industry has shown its skills and huge manufacturing has lower down the price of computer and electronic hardware. But the cost of software remains high if proper process is not adapted

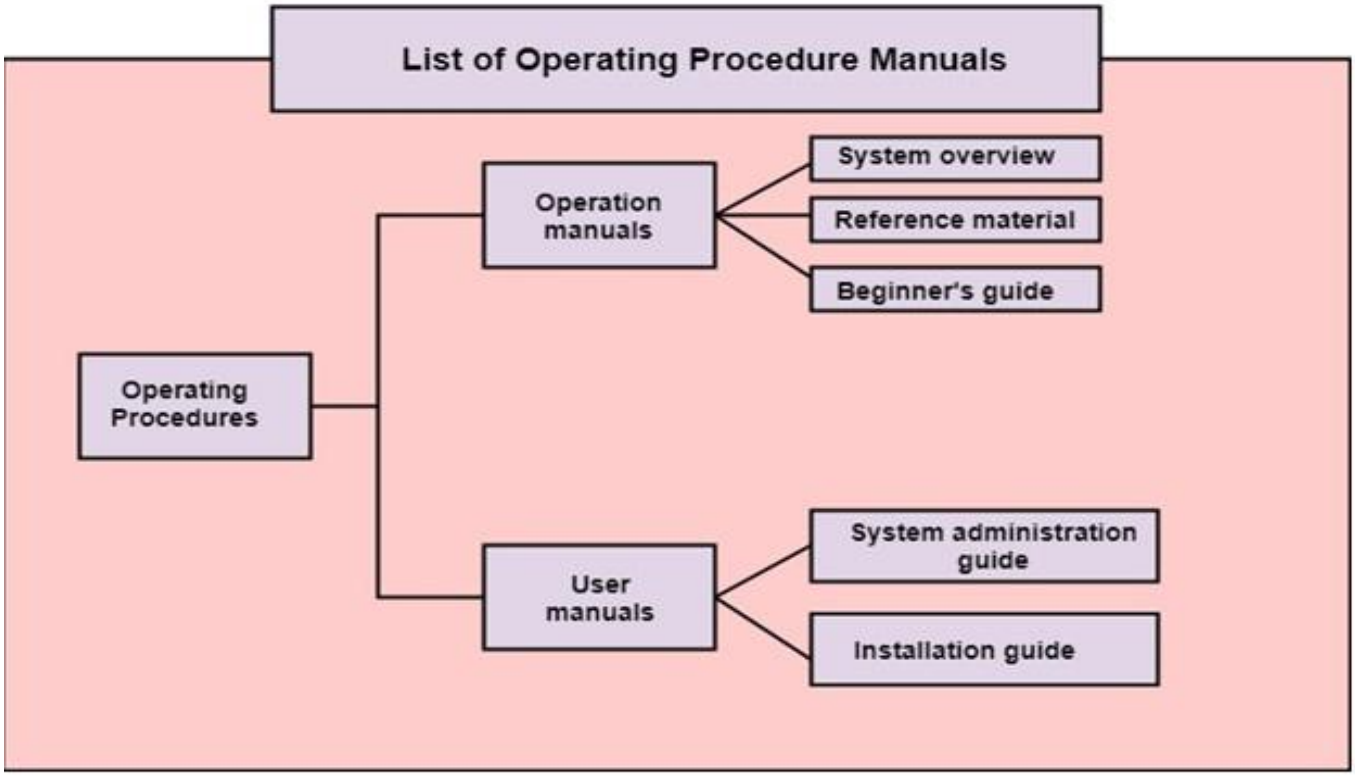
Components of Software Engineering

1. **Program:** Program is a combination of source code & object code.

2. Documentation: Documentation consists of different types of manuals. Examples of documentation manuals are: Data Flow Diagram, Flow Charts, ER diagrams, etc.



3.<u>Operating Procedures</u>: Operating Procedures consist of instructions to set up and use the software system and instructions on how react to the system failure. Example of operating system procedures manuals is: installation guide, Beginner's guide, reference guide, system administration guide, etc.



CHARACTERESTICS OF GOOD SOFTWARE

1-Operational

This tells us how well software works in operations. It can be measured on:

1. Budget
2. Usability
3. Efficiency
4. Correctness
5. Functionality
6. Dependability
7. Security
8. Safety

2-Transitional

1. This aspect is important when the software is moved from one platform to another:
2. Portability
3. Interoperability
4. Reusability
5. Adaptability

3-Maintenance

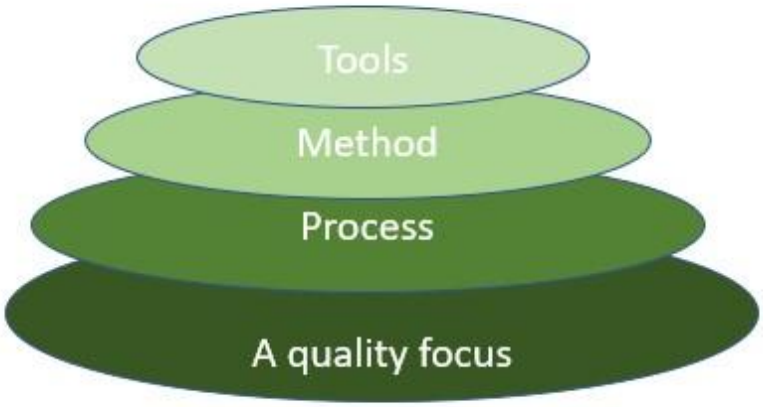
This aspect briefs about how well software has the capabilities to maintain itself in the ever-changing environment:

1. Modularity
2. Maintainability
3. Flexibility
4. Scalability

In short, Software engineering is a branch of computer science, which uses well-defined engineering concepts required to produce efficient, durable, scalable, in-budget and on-time software products

Layered Technology in Software Engineering

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.



Layer 1 — Tools

The first layer involves choosing the semi-automated and automated tools that will become the framework for the project. 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.

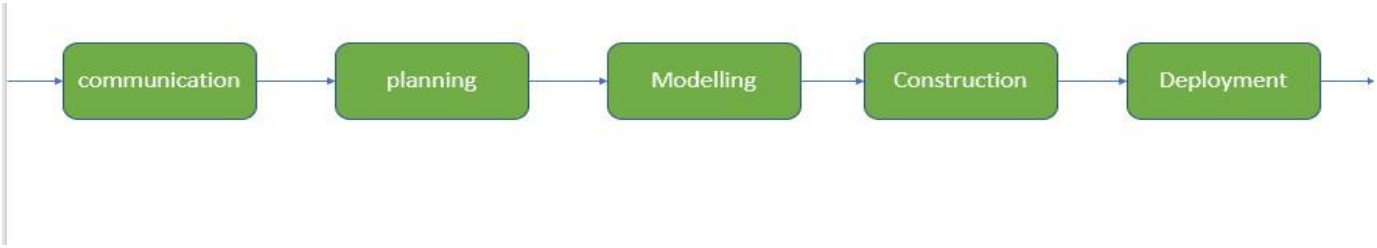
Layer 2 — Method

The second layer establishes the methods of developing the software. This includes any technical knowledge and resources required for development. Some tasks include choosing methods for:

- Communication
- Analysis
- Modeling
- Program construction
- Testing and support

Layer 3 — Process

Layer three focuses on the framework that needs to be established to deliver software effectively. This layer can be broken down into five sub layers:



Layer 4 — A Quality Focus

At this point, the software is developed and refined to a point, but it is critical to apply quality control to the finished product. Besides testing the end product to ensure that it meets the client's specifications, it also needs real-world testing to determine how efficient, usable, and reusable it will be, and it needs to explore how much resource maintenance will require.

Software Crisis

Software Crisis is a term used in computer science for the difficulty of writing useful and efficient computer programs in the required time. The software crisis was due to using the same workforce, same methods, same tools even though rapidly increasing in software demand, the complexity of software, and software challenges.

Factors contributing to the software crisis

- Poor project management.
- Lack of adequate training in software engineering.
- Less skilled project members.
- Low productivity improvements.

Solution of Software Crisis: There is no single solution to the crisis. One possible solution to a software crisis is Software Engineering because software engineering is a

systematic, disciplined, and quantifiable approach. For preventing software crises, there are some guidelines:

- Reduction in software over budget.
- The quality of software must be high.
- Less time is needed for a software project.
- Experienced and skilled people working over the software project.
- Software must be delivered.
- Software must meet user requirements.

Software Engineering Process :

It is a engineering process which is mainly related to computers and programming and developing different kinds of applications through the use of information technology.

There are four basic key process activities:

1.Software Specifications –In this process, detailed description of a software system to be developed with its functional and non-functional requirements.

2.Software Development –In this process, designing, programming, documenting, testing, and bug fixing is done.

3.Software Validation –In this process, evaluation software product is done to ensure that the software meets the business requirements as well as the end users needs.

4.Software Evolution – It is a process of developing software initially, then timely updating it for various reasons.

Conventional Engineering Process : It is a engineering process which is highly based on empirical knowledge and is about building cars, machines and hardware.

S.No.	**Software Engineering Process**	**Conventional Engineering Process**
1.	**Software Engineering Process is a process which majorly involves computer science, information technology and discrete mathematics.**	**Conventional Engineering Process is a process which majorly involves science, mathematics and empirical knowledge.**
2.	**It is mainly related with computers, programming and writing codes for building applications.**	**It is about building cars, machines, hardware, buildings etc.**
3.	**In Software Engineering Process construction and development cost is low.**	**In Conventional Engineering Process construction and development cost is high.**
4.	**It can involve the application of new and untested elements in software projects.**	**It usually applies only known and tested principles to meet product requirements.**
5.	**In Software Engineering Process, most development effort goes into building new designs and features.**	**In Conventional Engineering Process, most development efforts are required to change old designs.**

6.	**It majorly emphasize on quality.**	**It majorly emphasize on mass production.**

Quality Attributes

The following factors are used to measure Software Development Quality. Each attribute can be used to measure product performance. These attributes can be used for Quality assurance as well as Quality control:

1) Reliability

Measure if the product is reliable enough to sustain in any condition

2) Maintainability

Different versions of the product should be easy to maintain. For development, it should be easy to add code to the existing system, should be easy to upgrade for new features and new technologies from time to time.

3) Usability

This can be measured in terms of ease of use. The application should be userfriendly. It should be easy to learn. Navigation should be simple.

4) Portability

This can be measured in terms of Costing issues related to porting, Technical issues related to porting, and Behavioral issues related to porting.

5) Correctness

The application should be correct in terms of its functionality, calculations used internally and the navigation should be correct. This means that the application should adhere to functional requirements.

6) Efficiency

It is measured in terms of time required to complete any task given to the system.

7) Integrity or Security

Integrity comes with security. System integrity or security should be sufficient to prevent unauthorized access to system functions, prevent information loss, ensure that the software is protected from virus infection, and protect the privacy of data entered into the system.

8) Testability

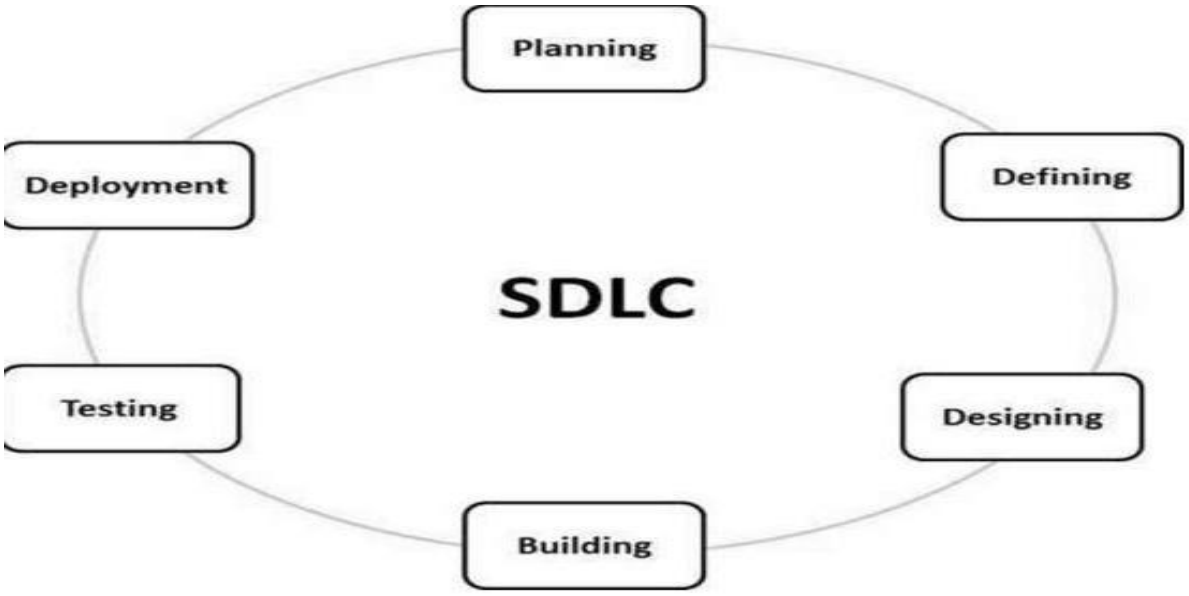
The system should be easy to test and find defects. If required, it should be easy to divide into different modules for testing.

9) Flexibility

Should be flexible enough to modify

SDLC-SOFTWARE DEVELOPMENT LIFE CYCLE

- Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality software.
- The SDLC aims to produce high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.
- It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software.



Step 1: Planning and Requirement Analysis

It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry.

Step 2: Defining Requirements

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through an **SRS (Software Requirement Specification)** document which consists of all the product requirements to be designed and developed during the project life cycle.

Step 3: Designing the Product Architecture

Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.

A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules

Step 4: Building or Developing the Product

The programming code is generated as per DDS during this stage. Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code.

Step 5: Testing the Product

This stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

Step 6: Deployment in the Market and Maintenance

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment.

SDLC Models

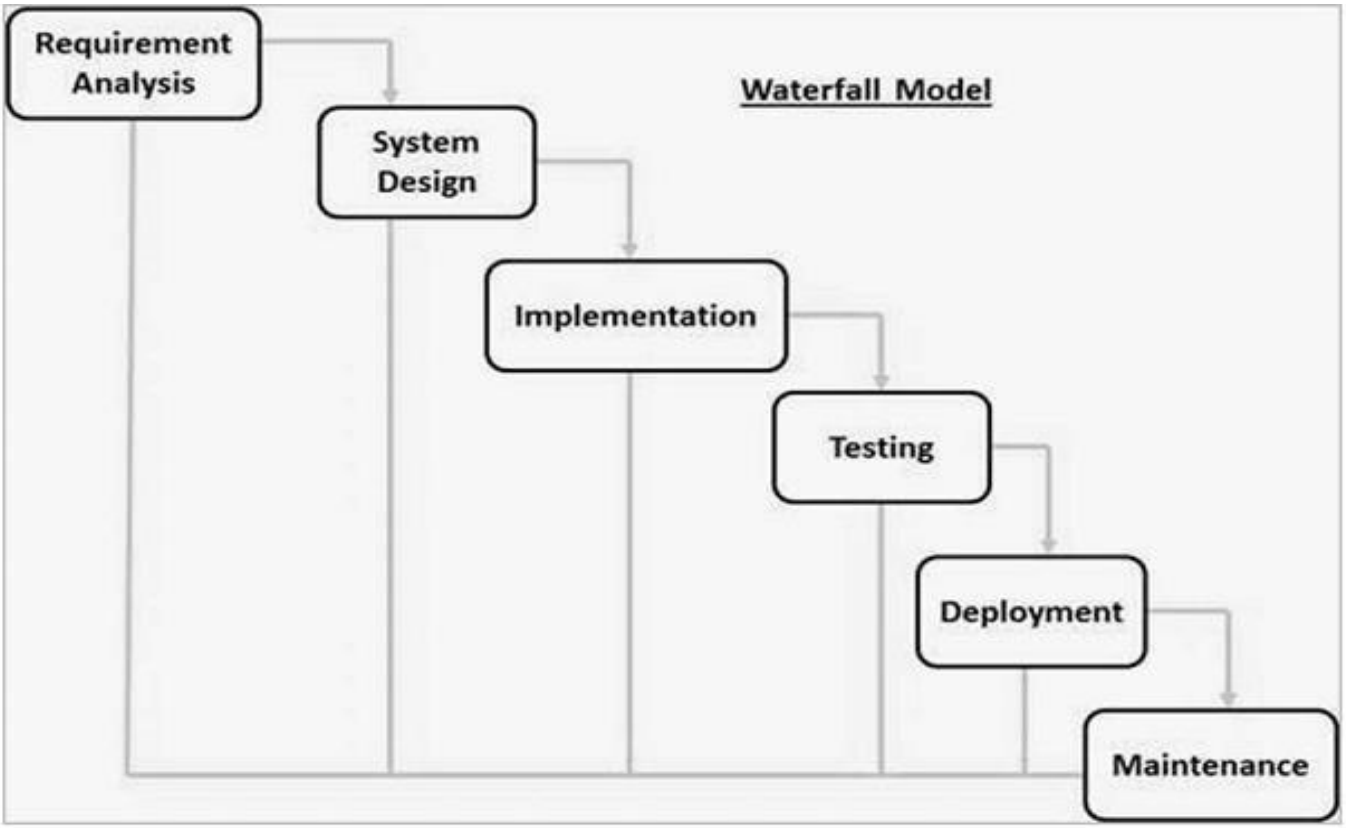
- There are various software development life cycle models defined and designed which are followed during the software development process.
- Waterfall Model
- Iterative Model
- Spiral Model
- V-Model
- Big Bang Model

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**.

Waterfall model is introduced by Royce in year 1970. Waterfall model follows the SDLC approach and states that "the phases are organized in a linear order and the output of one phase becomes the input for the next phase."

The waterfall model is a sequential (non-iterative) design process, used in software development processes, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of conception, initiation, analysis, design, construction, testing, production/implementation and maintenance.



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.

Advantages of waterfall model:

- This model is simple and easy to understand and use.
- It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
- In this model phases are processed and completed one at a time. Phases do not overlap.
- Waterfall model works well for smaller projects where requirements are very well understood.

Disadvantages of waterfall model:

- Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
- 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

When to use the waterfall model:

- This model is used only when the requirements are very well known, clear and fixed.
- Product definition is stable.
- Technology is understood.
- There are no ambiguous requirements
- Ample resources with required expertise are available freely
- The project is short.

Limitations of Waterfall Model

- The model implies that you should attempt to complete a given stage before moving on to the next stage
  - Does not account for the fact that requirements constantly change.
  - It also means that customers cannot use anything until the entire system is complete.
- The model makes no allowances for prototyping.
- It implies that you can get the requirements right by simply writing them down and reviewing them.
- The model implies that once the product is finished, everything else is maintenance.

Iterative Model

Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented.

At each iteration, design modifications are made and new functional capabilities are added.

Also called Incremental Model

Project is break into small modules which can be delivered

A working version of software is produced during the first module.

Each subsequent release of the module adds functionality to the previous release. The process continues till the complete system is achieved.

Model very successfully when working with new technology

More than one iteration can be going at the same time

Advantages of Iterative Model

- Quick software is released during early phases of Software life cycle
- Less costly to change requirements as compared to other models
- Easier to develop and test when iterations are small
- Customer can give his feedback quickly
- More than one iteration can be going parallel at same time

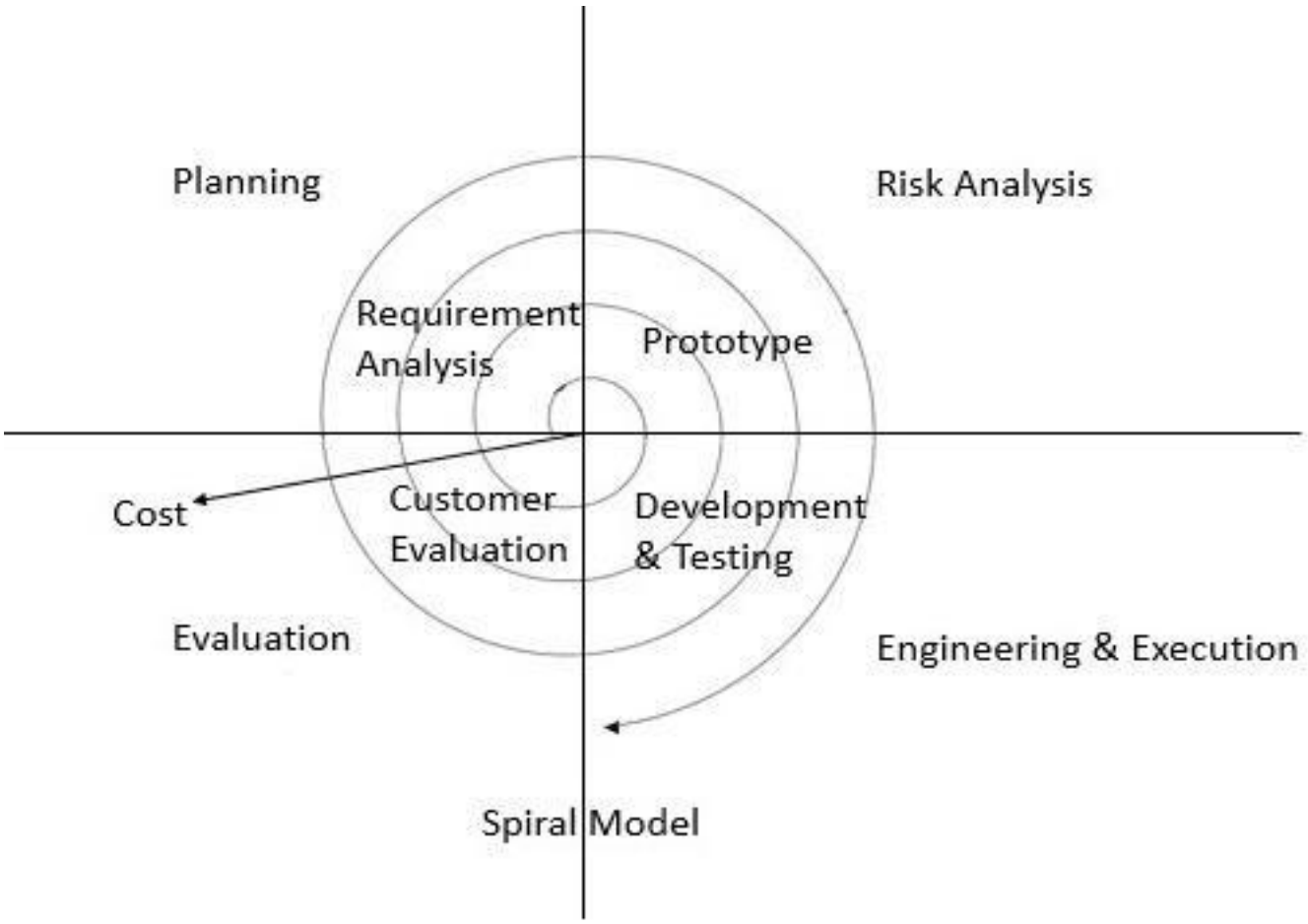
Dis-Advantages of Iterative Model

- Comparatively more resources are required
- Skilled Manager is needed to manage otherwise Project costing is increased
- Project started with complete Project architecture design, can result issues in future.
- Cost is higher than Waterfall model

Spiral Model

- The spiral model combines the idea of iterative development with the systematic, controlled aspects of the waterfall model.
- This Spiral model is a combination of iterative development process model and sequential linear development model i.e. the waterfall model with a very high emphasis on risk analysis.
- It allows incremental releases of the product or incremental refinement through each iteration around the spiral.

Phases of Spiral model



Objectives determination and identify alternative solutions: Requirements are gathered from the customers and the objectives are identified, elaborated, and analyzed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant.

Identify and resolve Risks: During the second quadrant, all the possible solutions are evaluated to select the best possible solution. Then the risks associated with that solution are identified and the risks are resolved using the best possible strategy. At the end of this quadrant, the Prototype is built for the best possible solution.

Develop next version of the Product:

During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available.

Review and plan for the next Phase:

In the fourth quadrant, the Customers evaluate the so far developed version of the software. In the end, planning for the next phase is started.

Why Spiral Model is called Meta Model?

- The Spiral model is called a Meta-Model because it subsumes all the other SDLC models. For example, a single loop spiral actually represents the Iterative Waterfall Model.
- The spiral model incorporates the stepwise approach of the Classical Waterfall Model. The spiral model uses the approach of the Prototyping Model by building a prototype at the start of each phase as a risk-handling technique.
- Also, the spiral model can be considered as supporting the Evolutionary model – the iterations along the spiral can be considered as evolutionary levels through which the complete system is built.

Advantages of Spiral Model

Risk Handling: The projects with many unknown risks that occur as the development proceeds, in that case, Spiral Model is the best development model to follow due to the risk analysis and risk handling at every phase.

Good for large projects: It is recommended to use the Spiral Model in large and complex projects.

Flexibility in Requirements: Change requests in the Requirements at later phase can be incorporated accurately by using this model.

Customer Satisfaction: Customer can see the development of the product at the early phase of the software development and thus, they habituated with the system by using it before completion of the total product.

Iterative and Incremental Approach: The Spiral Model provides an iterative and incremental approach to software development, allowing for flexibility and adaptability in response to changing requirements or unexpected events.

Emphasis on Risk Management: The Spiral Model places a strong emphasis on risk management, which helps to minimize the impact of uncertainty and risk on the software development process.

Improved Communication: The Spiral Model provides for regular evaluations and reviews, which can improve communication between the customer and the development team.

Improved Quality: The Spiral Model allows for multiple iterations of the software development process, which can result in improved software quality and reliability

Disadvantages of Spiral Model

Complex: The Spiral Model is much more complex than other SDLC models.

Expensive: Spiral Model is not suitable for small projects as it is expensive.

Too much dependability on Risk Analysis: The successful completion of the project is very much dependent on Risk Analysis. Without very highly experienced experts, it is going to be a failure to develop a project using this model.

Difficulty in time management: As the number of phases is unknown at the start of the project, so time estimation is very difficult.

Complexity: The Spiral Model can be complex, as it involves multiple iterations of the software development process.

Time-Consuming: The Spiral Model can be timeconsuming, as it requires multiple evaluations and reviews.

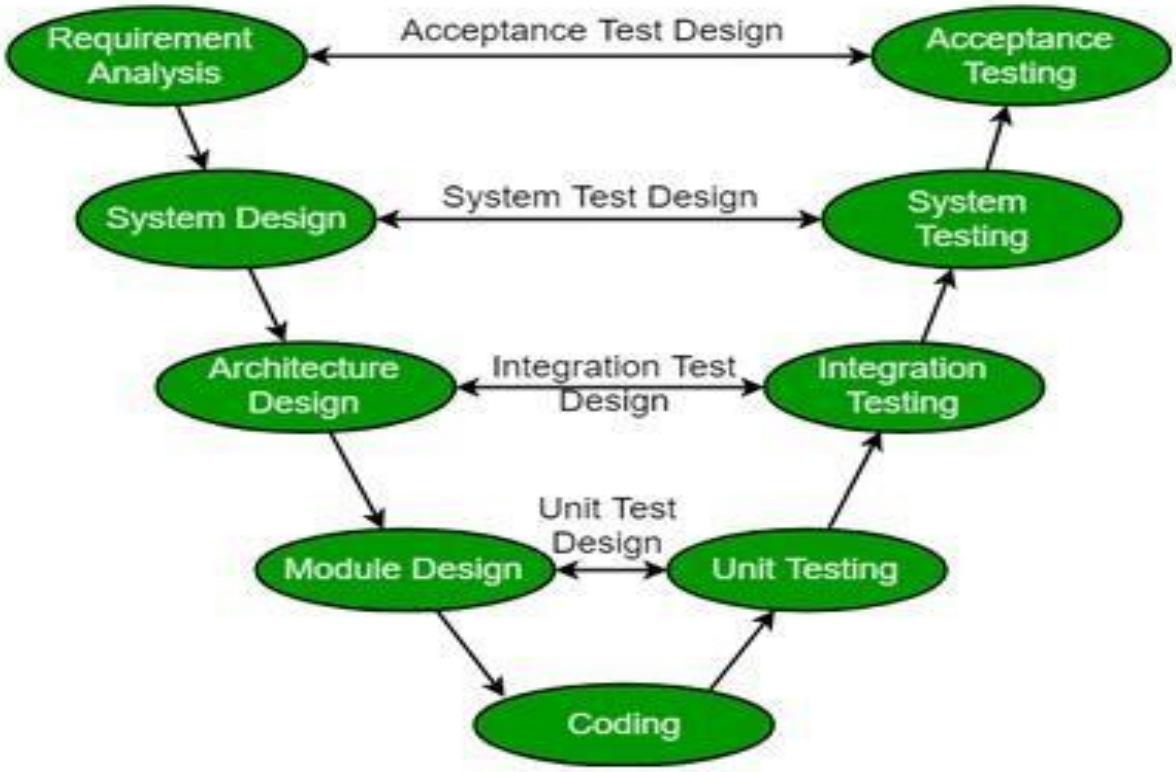
Resource Intensive: The Spiral Model can be resourceintensive, as it requires a significant investment in planning, risk analysis, and evaluations.

Uses of the spiral model

Projects in which changes may be required at any time
- long term projects that are not feasible due to altered economic priorities
- medium to high risk projects
- projects in which cost and risk analysis is important
- projects that would benefit from the creation of a prototype
- projects with unclear or complex requirements.

V-Model

- V-Model also referred to as the Verification and Validation Model.
- In this each phase of SDLC must complete before the next phase starts.
- It follows a sequential design process same as the waterfall model.
- Testing of the device is planned in parallel with a corresponding stage of development.



Verification:

It involves static analysis technique (review) done without executing code. It is the process of evaluation of the product development phase to find whether specified requirements meet.

Validation:

It involves dynamic analysis method (functional, non-functional), testing is done by executing code. Validation is the process to classify the software after the completion of the development process to determine whether the software meets the customer expectations and requirements.

So V-Model contains Verification phases on one side of the Validation phases on the other side. Verification and Validation phases are joined by coding phase in V-shape. Thus it is called V-Model.

Design Phase

Requirement Analysis: This phase contains detailed communication with the customer to understand their requirements and expectations. This stage is known as Requirement Gathering.

System Design: This phase contains the system design and the complete hardware and communication setup for developing product.

Architectural Design: System design is broken down further into modules taking up different functionalities. The data transfer and communication between the internal modules and with the outside world (other systems) is clearly understood.

Module Design: In this phase the system breaks down into small modules. The detailed design of modules is specified, also known as Low-Level Design (LLD).

Testing Phase

Unit Testing: Unit Test Plans are developed during module design phase. These Unit Test Plans are executed to eliminate bugs at code or unit level.

Integration testing: After completion of unit testing Integration testing is performed. In integration testing, the modules are integrated and the system is tested. Integration testing is performed on the Architecture design phase. This test verifies the communication of modules among themselves.

System Testing: System testing test the complete application with its functionality, inter dependency, and communication. It tests the functional and non-functional requirements of the developed application.

User Acceptance Testing (UAT): UAT is performed in a user environment that resembles the production environment. UAT verifies that the delivered system meets user's requirement and system is ready for use in real world.

Advantages

- This is a highly disciplined model and Phases are completed one at a time.
- V-Model is used for small projects where project requirements are clear.
- Simple and easy to understand and use.
- This model focuses on verification and validation activities early in the life cycle thereby enhancing the probability of building an error-free and good quality product.
- It enables project management to track progress accurately.
- Clear and Structured Process.
- Better Communication between the customer and the development team.

Disadvantages

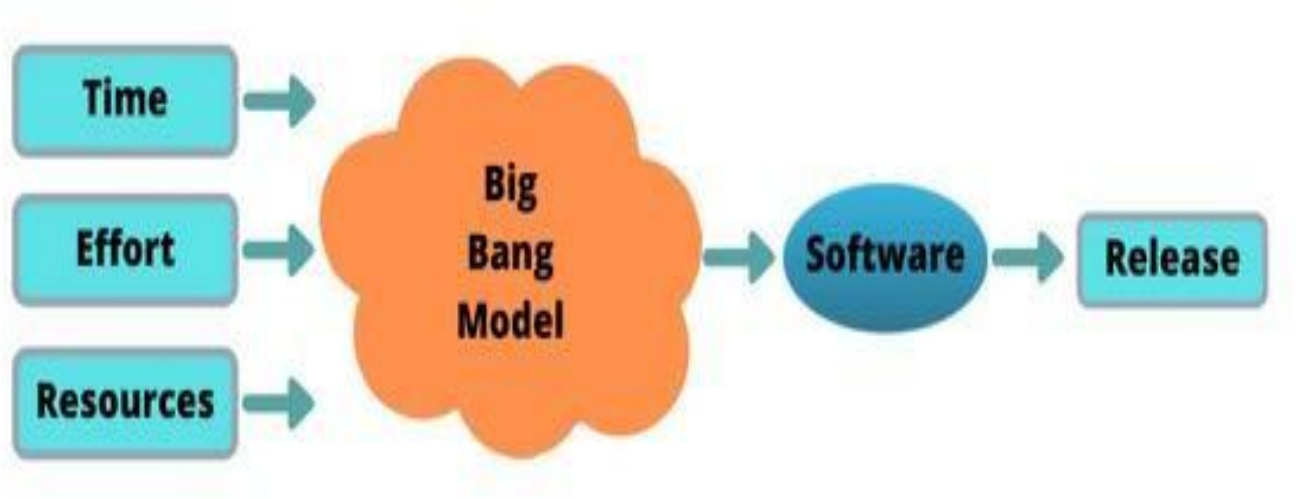
- High risk and uncertainty.
- It is not a good for complex and object-oriented projects.
- It is not suitable for projects where requirements are not clear and contains high risk of changing.
- This model does not support iteration of phases

- It does not easily handle concurrent events.
- Inflexibility

Big bang model

In this model, developers do not follow any specific process. Development begins with the necessary funds and efforts in the form of inputs. And the result may or may not be as per the customer's requirement, because in this model, even the customer requirements are not defined.

This model is ideal for small projects like academic projects or practical projects. One or two developers can work together on this model.



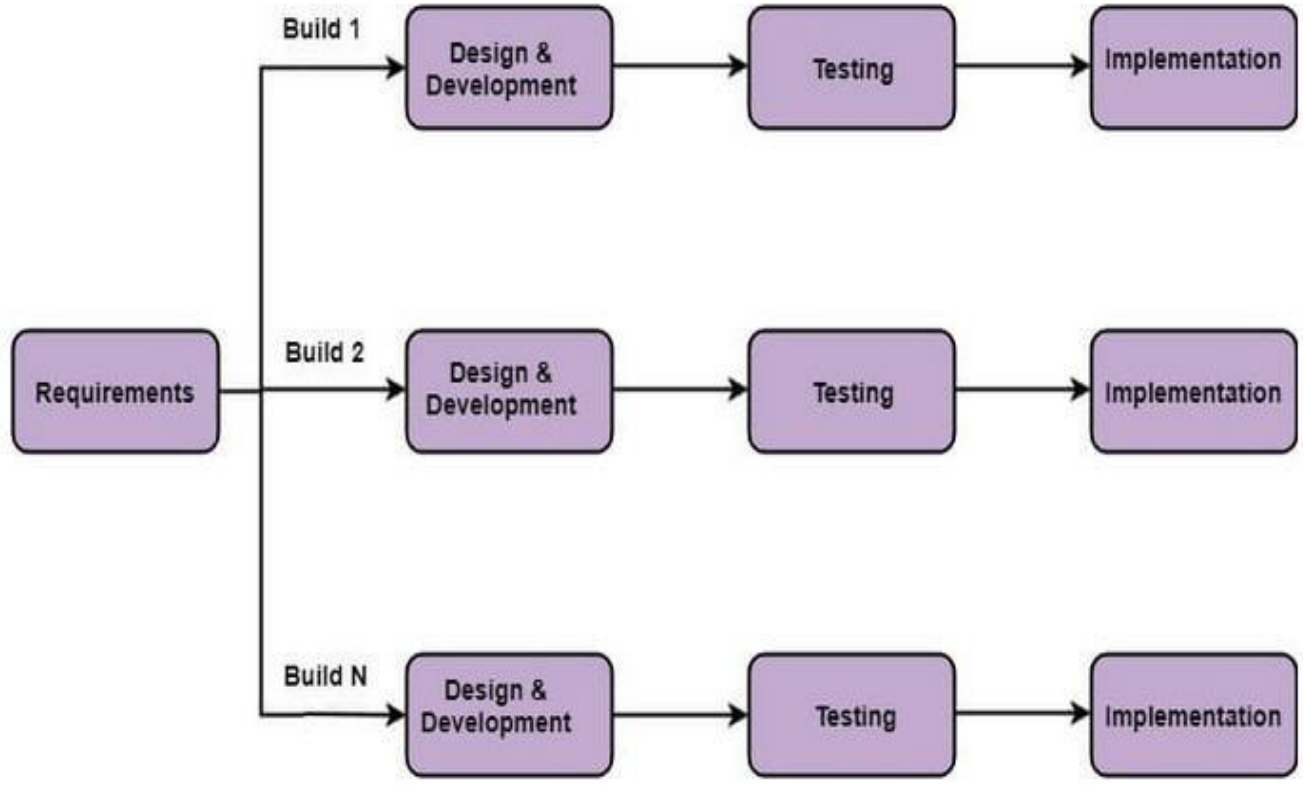
Big Bang SDLC MODEL

Advantage(Pros) of Big Bang Model

- There is no planning required.
- Simple Model.
- Few resources required.
- Easy to manage.
- Flexible for developers

Disadvantage(Cons) of Big Bang Model

- There are high risk and uncertainty.
- Not acceptable for a large project.
- If requirements are not clear that can cause very expensive.
- 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.



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.

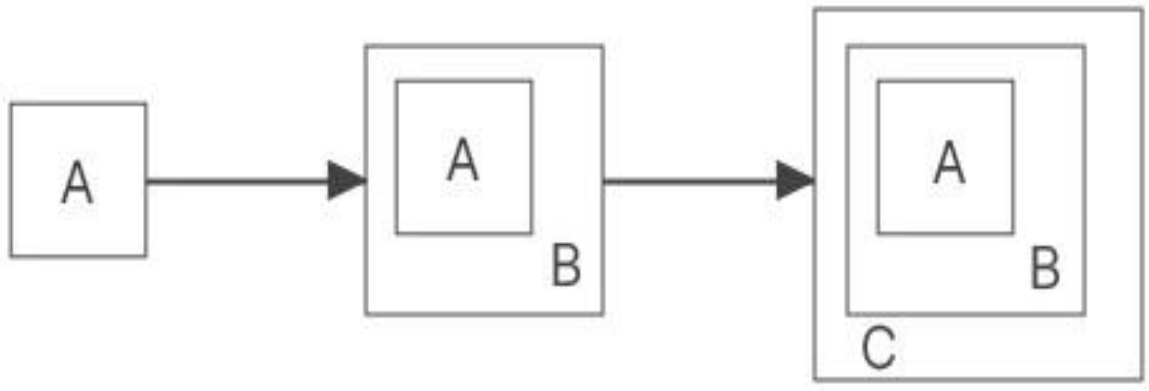
Disadvantage of Incremental Model

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

What is Evolutionary model?

Software requirements are fragmented into several chunks that can be built and transferred incrementally. In other words, you develop a skeleton of the product and in successive versions, you add more functionalities. Services are not required for the core module (first stage) of the system.

"Iterative" + "Incremental model" = "Evolutionary model"



Evolutionary Development Of Software Development

Where to use Evolutionary Model?

- Very useful for large projects.
- Well suited for projects using object-oriented development.
- When a client prefers to have the product in increments so that he can began using the specific characteristics as they are provided rather than waiting for the entire thing to be manufactured and delivered, this model is frequently adopted.

Advantages of the Model

- Risk analysis is better.
- It supports changing environment.
- Initial operating time is less.
- Better suited for large mission-critical projects.
- During the life cycle software is produced early which facilitates customer evaluation and feedback.

Disadvantage of the Model

- Management complexity is more.
- Not suitable for smaller projects.
- Can be costly to use.
- Highly skilled resources are required for risk analysis.