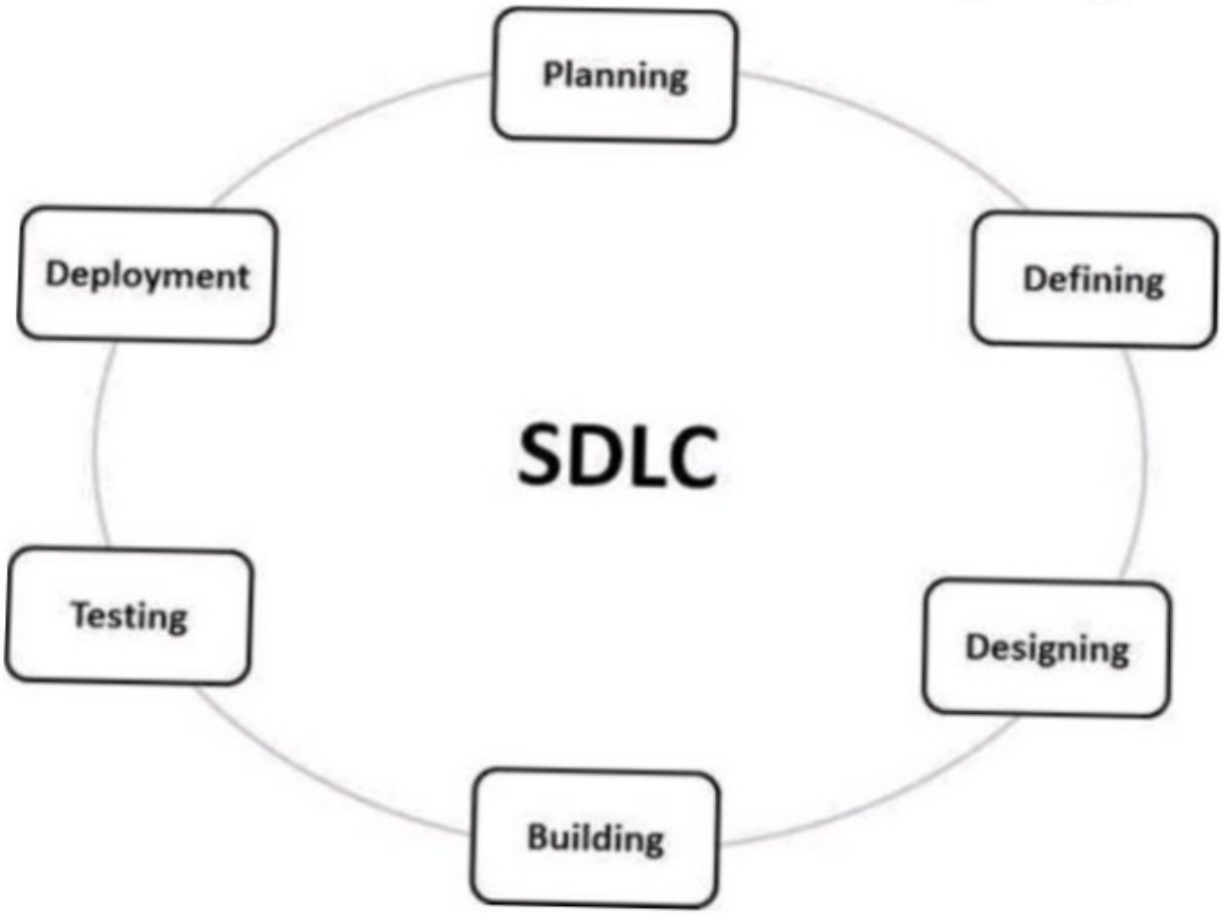
3.Software development process models :

What is SDLC?

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

The following figure is a graphical representation of the various stages of a typical SDLC.



A typical Software Development Life Cycle consists of the following stages –

Stage 1: Planning and Requirement Analysis

Requirement analysis is the most important and fundamental stage in SDLC. 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. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

Stage 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.

Stage 3: Designing the Product Architecture

SRS is the reference for product architects to come out with the best architecture for the product to be developed. 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.

This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

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 (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

Stage 4: Building or Developing the Product

In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code. Different high level programming languages such as C, C++, Pascal, Java and PHP are used for coding. The programming language is chosen with respect to the type of software being developed.

Stage 5: Testing the Product

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, 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.

Stage 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. Sometimes product deployment happens in stages as per the business strategy of that organization. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).

Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

4.SDLC Models

There are various software development life cycle models defined and designed which are followed during the software development process. These models are also referred as Software Development Process Models". Each process model follows a Series of steps unique to its type to ensure success in the process of software development.

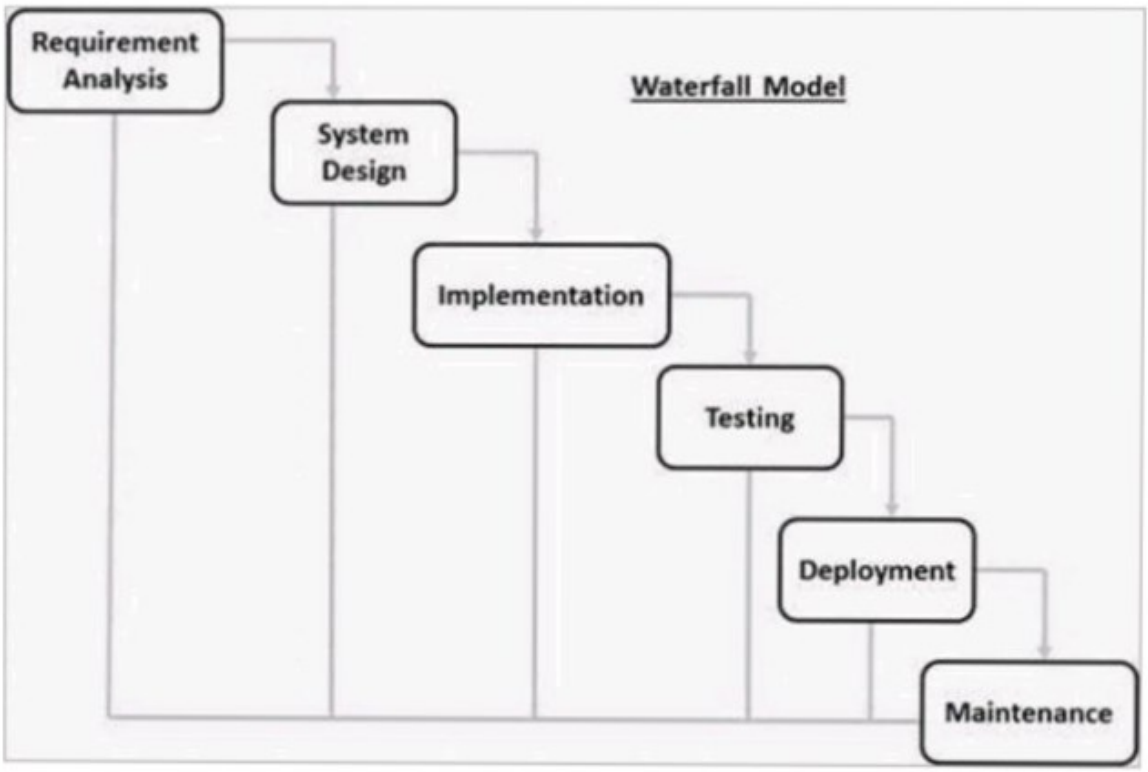
Following are the most important and popular SDLC models followed in the industry:

1. Waterfall Model
2. Prototyping
3. Iterative Model
4. Relational unified process
5. Time boxing model
6. Extreme programming
7. Agile process

1.Waterfall Model:

- Waterfall model is an example of a Sequential model. In this model, the software development activity is divided into different phases and each phase consists of series of tasks and has different objectives.
- It is divided into phases and output of one phase becomes the input of the next phase. It is mandatory for a phase to be completed before the next phase starts. In short, there is no overlapping in Waterfall model.
- In waterfall, development of one phase starts only when the previous phase is complete. Because of this nature, each phase of waterfall model is quite precise well defined. Since the phases fall from higher level to lower level, like a waterfall, It's named as waterfall model.

Pictorial representation of waterfall model:



Pros and Cons of waterfall model:

Advantages of using Waterfall model :

- Simple and easy to understand and use.
- For smaller projects, waterfall model works well and yield the appropriate results.
- Since the phases are rigid and precise, one phase is done one at a time, it is easy to maintain.

- The entry and exit criteria are well defined, so it easy and systematic to proceed with quality.
- Results are well documented.

Disadvantages of using Waterfall model:

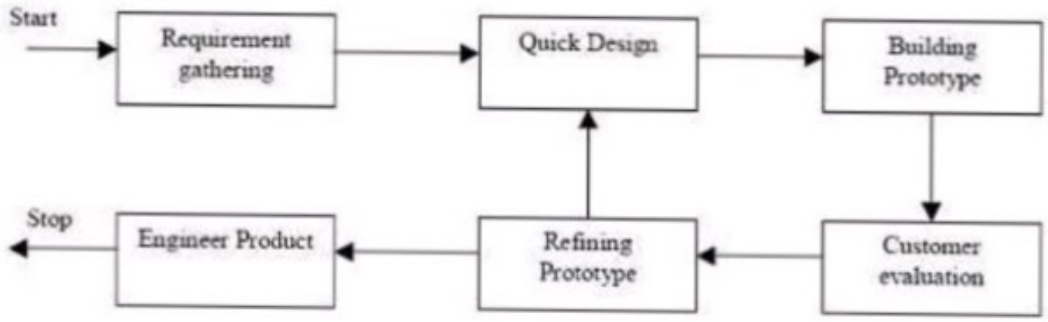
- Cannot adopt the changes in requirements
- It becomes very difficult to move back to the phase. For example, if the application has now moved to the testing stage and there is a change in requirement, It becomes difficult to go back and change it.
- Delivery of the final product is late as there is no prototype which is demonstrated intermediately.
- For bigger and complex projects, this model is not good as a risk factor is higher.
- Not suitable for the projects where requirements are changed frequently.
- Does not work for long and ongoing projects.
- Since the testing is done at a later stage, it does not allow identifying the challenges and risks in the earlier phase so the risk mitigation strategy is difficult to prepare.

Conclusion:
In the waterfall model, it is very important to take the sign off of the deliverables of each phase. As of today most of the projects are moving with Agile and Prototype models, Waterfall model still holds good for smaller projects. If requirements are straightforward and testable, Waterfall model will yield the best results.

2.Prototyping :

- The basic idea in **Prototype model** is that instead of freezing the requirements before a design or coding can proceed, a throwaway prototype is built to understand the requirements.
- This prototype is developed based on the currently known requirements. Prototype model is a **software development model**. By using this prototype, the client can get an "actual feel" of the system, since the interactions with prototype can enable the client to better understand the requirements of the desired system.
- Prototyping is an attractive idea for complicated and large systems for which there is no manual process or existing system to help determining the requirements.
- The prototype are usually not complete systems and many of the details are not built in the prototype. The goal is to provide a system with overall functionality.

Diagram of Prototype model:



Prototyping Model

Advantages of Prototype model:

- Users are actively involved in the development
- Since in this methodology a working model of the system is provided, the users get a better understanding of the system being developed.
- Errors can be detected much earlier.
- Quicker user feedback is available leading to better solutions.
- Missing functionality can be identified easily

.

Disadvantages of Prototype model:

- Leads to implementing and then repairing way of building systems.
- Practically, this methodology may increase the complexity of the system as scope of the system may expand beyond original plans.

When to use Prototype model:

- Prototype model should be used when the desired system needs to have a lot of interaction with the end users.
- Typically, online systems, web interfaces have a very high amount of interaction with end users, are best suited for Prototype model. It might take a while for a system to be built that allows ease of use and needs minimal training for the end user.
- Prototyping ensures that the end users constantly work with the system and provide a feedback which is incorporated in the prototype to result in a useable system. They are excellent for designing good human computer interface systems.

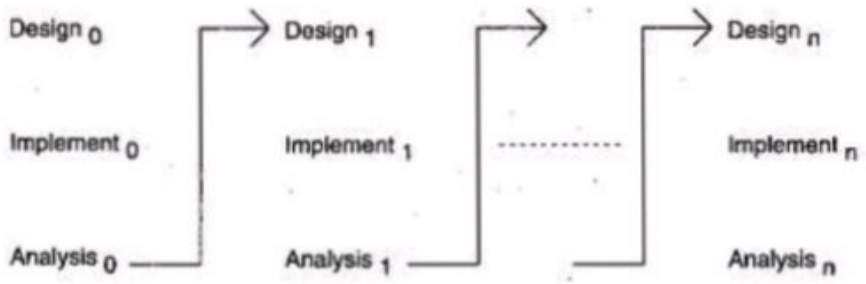
3.Iterative Model :

An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which can then be reviewed in order to identify further requirements. This process is then repeated, producing a new version of the software for each cycle of the model.

For example:

In the diagram above when we work **iteratively** we create rough product or product piece in one iteration, then review it and improve it in next iteration and so on until it's finished. As shown in the image above, in the first iteration the whole painting is sketched roughly, then in the second iteration colors are filled and in the third iteration finishing is done. Hence, in iterative model the whole product is developed step by step.

Diagram of Iterative model:



Advantages of Iterative model:

- In iterative model we can only create a high-level design of the application before we actually begin to build the product and define the design solution for the entire product. Later on we can design and built a skeleton version of that, and then evolved the design based on what had been built.
- In iterative model we are building and improving the product step by step. Hence we can track the defects at early stages. This avoids the downward flow of the defects.
- In iterative model we can get the reliable user feedback. When presenting sketches and blueprints of the product to users for their feedback, we are effectively asking them to imagine how the product will work.
- In iterative model less time is spent on documenting and more time is given for designing.