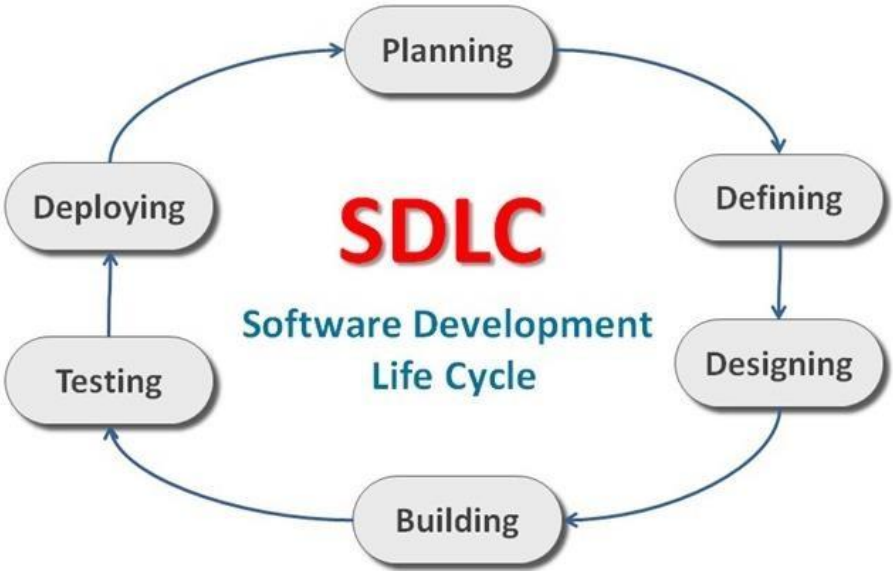
Software Development Life Cycle [SDLC]:

- 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.
- 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.



1. **Planning:** Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team. This information is then used to plan the basic project approach.
2. **Defining Requirement:** 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.

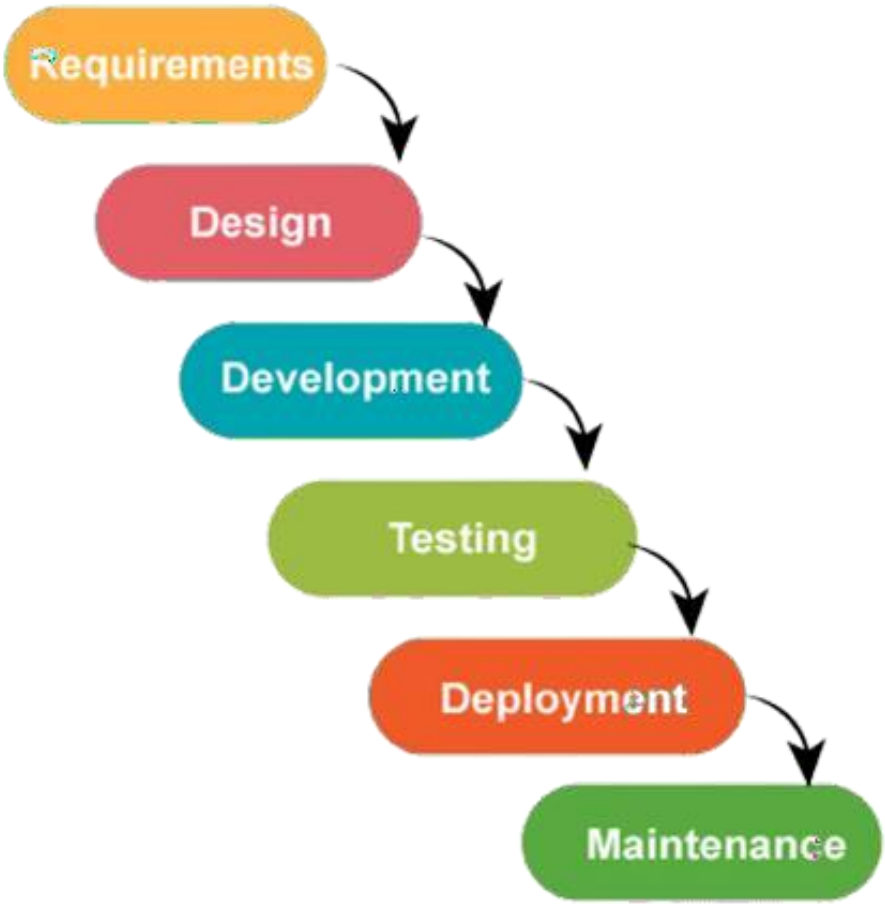
3. **Designing the architecture:** A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules.
4. **Building or developing software:** Developers must follow the coding guidelines defined 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.
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.
6. **Deployment 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.

Software Process Models:

- A software engineering must follow some strategies that include the process, methods, tools and generic phases.
- This strategy is referred as a **Process Model** or **Software Engineering Paradigm.**
- A process model for any software is chosen based on the nature of project and application, the methods and tools to be used, and control that are required.
- A system too large for one person to build is usually also too large to build without an overall plan that coordinates the people working on it, the tasks that need to be done, and the artifacts that are produced.
- Researchers and practitioners have identified a number of software development process models for this coordination.

1. Waterfall Model:

- The Waterfall Model was first Process Model to be introduced.
- It is also referred to as a **Linear-Sequential Model**.
- In a waterfall model, each phase must be completed fully before the next phase can begin.
- This type of software development model is basically used for the project which is small and there are no uncertain requirements.
- It says that 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.
- In waterfall model, phases do not overlap.



Applications: Some situations where the use of Waterfall model is most appropriate are:

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

Advantages:

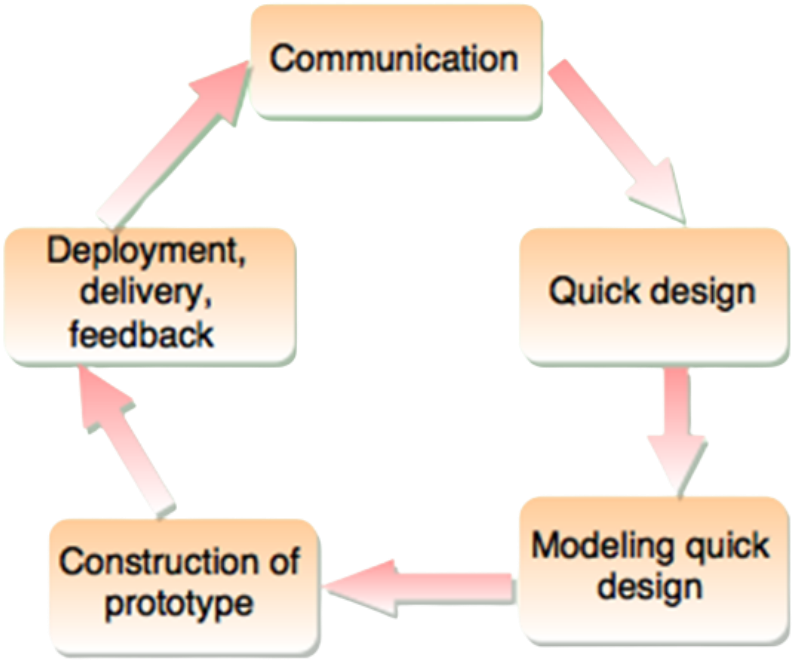
- It provides templates.
- It is widely used procedural model.
- It provides clear project objectives and stable project requirements which are essential for quality software development.
- In this model progress of the system is measurable.

Disadvantages:

- Changes can cause confusions the project team proceeds.
- All the requirements must be predefined.
- Customer must have patience as this model is time consuming.
- Backtracking is not possible.
- It leads to "blocking states" as the project team members must wait for other team members to complete the tasks.

2. Prototype Model:

- The prototype model requires that before carrying out the development of actual software, a working prototype of the system should be built.
- A prototype is a toy implementation of the system.
- A prototype usually turns out to be a very crude version of the actual system, possible exhibiting limited functional capabilities, low reliability, and inefficient performance as compared to actual software.
- In many instances, the client only has a general view of what is expected from the software product.
- Prototyping is used to allow the users evaluate developer proposals and try them out before implementation.
- It also helps understand the requirements which are user specific and may not have been considered by the developer during product design.



a) **Communication:** The feedback and the review comments are discussed during this stage and some negotiations happen with the customer based on factors like, time and budget constraints and technical feasibility of actual implementation. The changes accepted are again incorporated in the new Prototype developed and the cycle repeats until customer expectations are met.

b) **Quick plan:** According to the communication done with the user and team members, planning for the next prototype is defined in this stage. This defines the due date of development of the prototype, when to deploy, etc.

c) **Modeling Quick design:** This step involves understanding the very basics product requirements especially in terms of user interface. The more intricate details of the internal design and external aspects like performance and security can be ignored at this stage.

d) **Construction of prototype:** The initial Prototype is developed in this stage, where the very basic requirements are showcased and user interfaces are provided. These features may not exactly work in the

same manner internally in the actual software developed and the workarounds are used to give the same look and feel to the customer in the prototype developed.

e) **Deployment, delivery & feedback:** The prototype developed is then presented to the customer and the other important stakeholders in the project. The feedback is collected in an organized manner and used for further enhancements in the product under development.

Applications: Some situations where the use of Prototype model is most appropriate are:

- Software Prototyping is most useful in development of systems having high level of user interactions such as online systems.
- Systems which need users to fill out forms or go through various screens before data is processed can use prototyping very effectively to give the exact look and feel even before the actual software is developed.
- Software that involves too much of data processing and most of the functionality is internal with very little user interface does not usually benefit from prototyping.

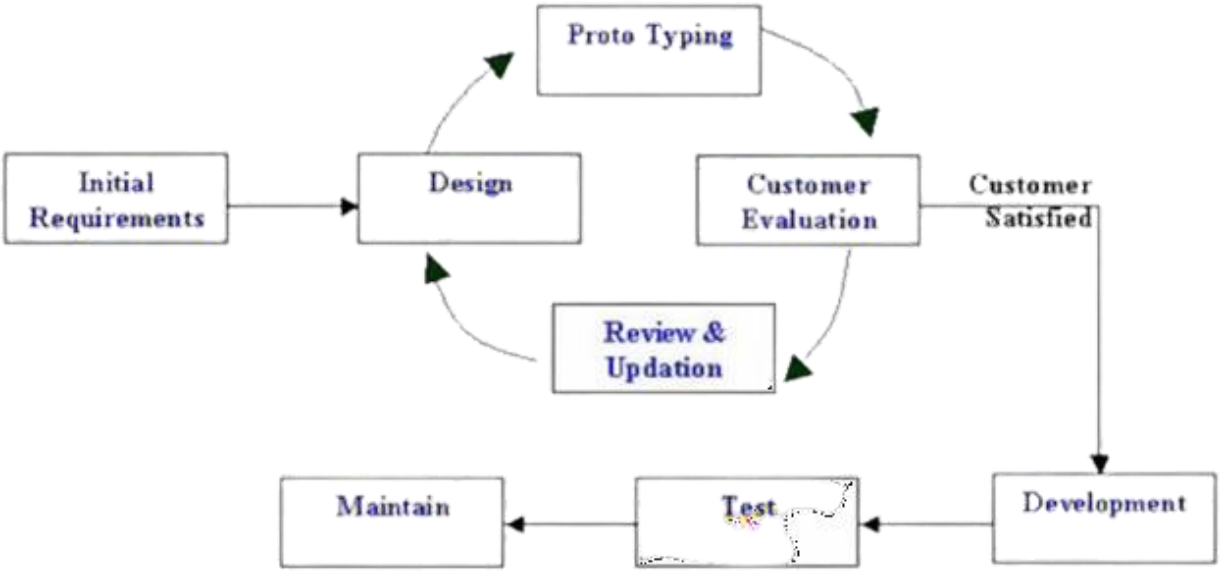
Advantages:

- Since a working model of the system is displayed, the users get a better understanding of the system being developed.
- It could serve as the first system.
- Customer doesn't need to wait long as in the linear model.
- Feedback from customers is received periodically and the changes don't come as a last minute surprise.

Disadvantages:

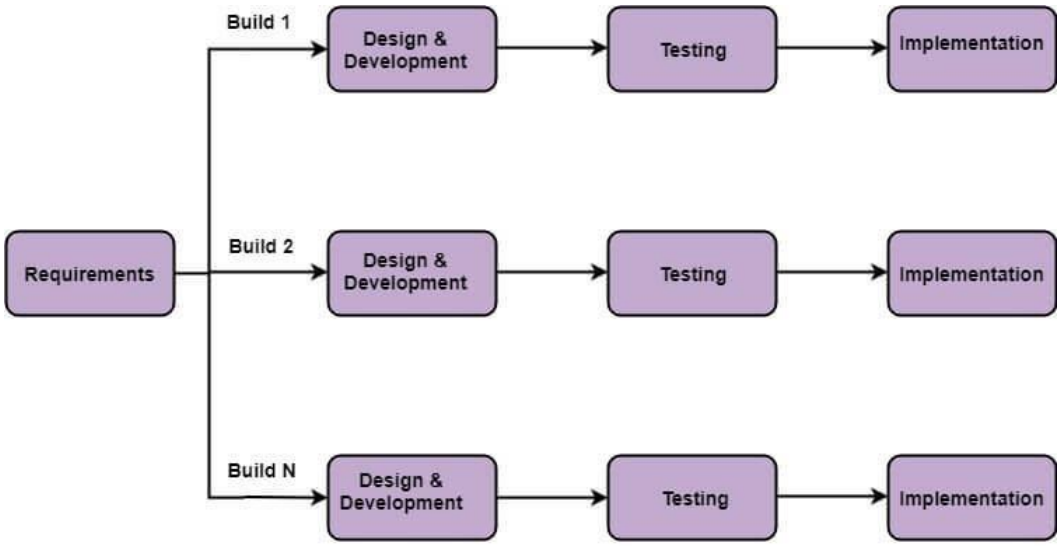
- Customer could believe the prototype as the working version.
- Developer could also make compromises as he quick fixes to the prototype and make it as a working version.

- Often clients expect that a new minor changes to the prototype will more than suffice their needs.



3. Iterative/Incremental 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.
- The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).
- This process may be described as an "evolutionary acquisition" or "incremental build" approach.
- The key to successful use of an iterative software development lifecycle is rigorous validation of requirements, and verification & testing of each version of the software against those requirements within each cycle of the model. As the software evolves through successive cycles, tests have to be repeated and extended to verify each version of the software.



Applications: Some situations where the use of Incremental/Iterative model is most appropriate are:

- Requirements of the complete system are clearly defined and understood.
- Major requirements must be defined; however, some functionalities or requested enhancements may evolve with time.
- There is a time to the market constraint.
- A new technology is being used and is being learnt by the development team while working on the project.
- Resources with needed skill set are not available and are planned to be used on contract basis for specific iterations.
- There are some high risk features and goals which may change in the future.

Advantages:

- Parallel development can be planned.
- Progress can be measured.
- Less costly to change the scope/requirements.
- Testing and debugging during smaller iteration is easy.
- Better suited for large and mission-critical projects.

- Risks are identified and resolved during iteration; and each iteration is an easily managed milestone.

Disadvantages:

- System architecture or design issues may arise because not all requirements are gathered in the beginning of the entire life cycle.
- Defining increments may require definition of the complete system.
- Not suitable for smaller projects.
- Management complexity is more.

4. Spiral Model:

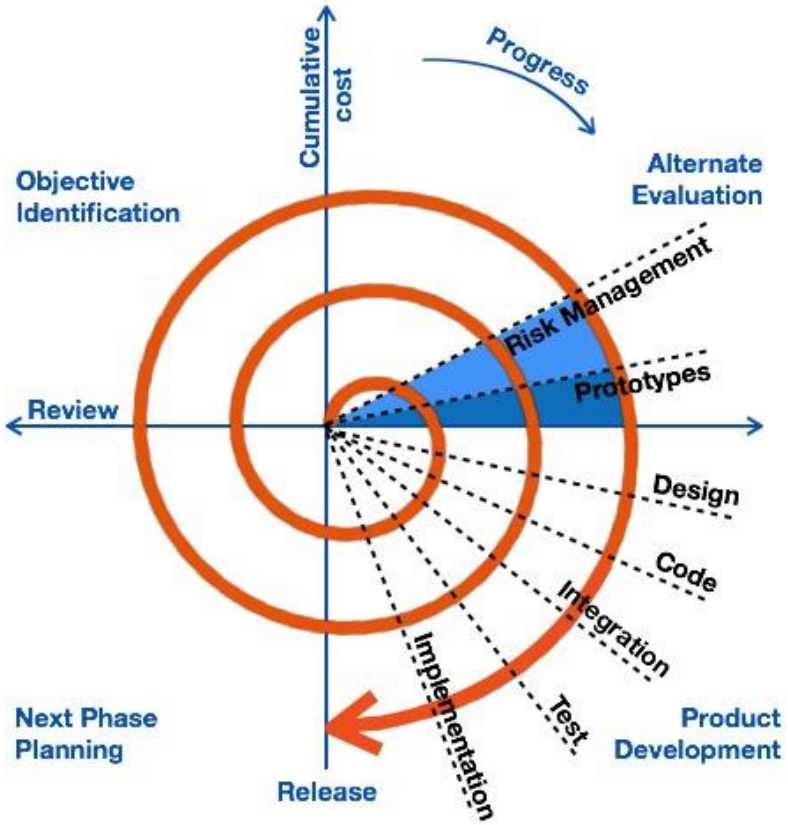
- Spiral Model is a software development life cycle model which is highly used for risk driven models.
- It consist of number of loops which are forming a spiral shape where each loop is called phase of software development cycle.
- The spiral model has four phases: Identification, Design, Build and Evaluation/Risk Analysis.
- A software project repeatedly passes through these phases in iterations called Spirals.
- The spiral model has four phases.
- A software project repeatedly passes through these phases in iterations called Spirals.

a) **Identification:** This phase starts with gathering the business requirements in the baseline spiral. In the subsequent spirals as the product matures, identification of system requirements, subsystem requirements and unit requirements are all done in this phase. This also includes understanding the system requirements by continuous communication between the customer and the system analyst. At the end of the spiral the product is deployed in the identified market.

b) **Design:** Design phase starts with the conceptual design in the baseline spiral and involves architectural design, logical design of modules, physical product design and final design in the subsequent spirals.

c) **Construct or Build:** Construct phase refers to production of the actual software product at every spiral. In the baseline spiral when the product is just thought of and the design is being developed a POC (Proof of Concept) is developed in this phase to get customer feedback. Then in the subsequent spirals with higher clarity on requirements and design details a working model of the software called build is produced with a version number. These builds are sent to customer for feedback.

d) **Evaluation and Risk Analysis:** Risk Analysis includes identifying, estimating, and monitoring technical feasibility and management risks, such as schedule slippage and cost overrun. After testing the build, at the end of first iteration, the customer evaluates the software and provides feedback.



Applications: Some situations where the use of Spiral model is most appropriate are:

- When costs there is a budget constraint and risk evaluation is important.
- For medium to high-risk projects.
- Long-term project commitment because of potential changes to economic priorities as the requirements change with time.
- Customer is not sure of their requirements which is usually the case.
- Requirements are complex and need evaluation to get clarity.
- New product line which should be released in phases to get enough customer feedback.
- Significant changes are expected in the product during the development cycle.

Advantages:

- Changing requirements can be accommodated.
- Requirements can be captured more accurately.
- Development can be divided into smaller parts and more risky parts can be developed earlier which helps better risk management.

Disadvantages:

- End of project may not be known early.
- Not suitable for small or low risk projects and could be expensive for small projects.
- Spiral may go indefinitely.
- Large number of intermediate stages requires excessive documentation.