Software Development Life Cycle, SDLC for short, is a well-defined, structured sequence of stages in software engineering to develop the intended software product.

SDLC has some specific phase. This are-

- project identification
- feasibility study
- system analysis
- system design
- system development
- system testing
- system implementation
- system maintenance
- system documentation

1) **Project identification**: - in this phase the analyst focus the basic objective and identification need for the corresponding software. In this phase the analyst set up some meeting with the corresponding client for making the desired software.

2) **Feasibility study**: - feasibility defines in the three views for making particular software for the client.

a) Technical b) financial c) social feasibility.

3) **System analysis**: - analysis defines how and what type of desired software we have to make for the client. It has some pen and paper base. Exercise through which the analyst focused for there desired goals.

4) **System design**: - in this phase the analyst draw the corresponding diagrams related to the particular software. in this phase the design include in the form of flow chat, data flow diagram, ntt relationship diagram (NRD).

5) **System development**: - development refers in the form of coding, error checking and debarking for the particular software. This phase deals with the developer activity for making a successfully software.

6) **System testing**: - testing refers whatever analyst and developer done will it be correct and error free to the desired software. In the S.E there some testing technique to which we can check whether project is error free. The main testing techniques are

- white box testing
- black box testing
- ad hope testing
- system testing
- unit testing
- alpha testing
- beta testing

7) **System implementation**: - after completing the testing phase we have to implement a particular product or system according to the customer need. In the implementation phase some design and other user activity part may be changed as per customer need.

8) **System maintenance**: - after implementation the users use the particular software to there corresponding operation to active there job. In this phase the software maintained from the user or developer side after spanning some times of use of particular software. In this phase the related hardware, software and other utilities are also maintained.

9) **System documentation**: - documentation refers the approach and guidelines for the user as well as the customer to the related software. The documentation refers some writing instruction for how to use for related hardware requirement, and also some maintains factors for the users.

What is SRS?

A software requirements specification (SRS) is a description of a software system to be developed. It lays out functional and non-functional requirements and may include a set of use cases that describe user interactions that the software must provide.

Why SRS?

In order to fully understand one's project, it is very important that they come up with an SRS listing out their requirements, how are they going to meet them and how will they complete the project. It helps the team to save upon their time as they are able to comprehend how are going to go about the project. Doing this also enables the team to find out about the limitations and risks early on.

Characteristics of a good SRS document:

Correctness:

User review is used to ensure the correctness of requirements stated in the SRS. SRS is said to be correct if it covers all the requirements that are actually expected from the system.

Completeness:

Completeness of SRS indicates every sense of completion including the numbering of all the pages, resolving the to be determined parts to as much extent as possible as well as covering all the functional and non-functional requirements properly.

Consistency:

Requirements in SRS are said to be consistent if there are no conflicts between any set of requirements. Examples of conflict include differences in terminologies used at separate places, logical conflicts like time period of report generation, etc.

Unambiguousness:

A SRS is said to be unambiguous if all the requirements stated have only 1 interpretation. Some of the ways to prevent unambiguousness include the use of modelling techniques like ER diagrams, proper reviews and buddy checks, etc.

Ranking for importance and stability:

There should a criterion to classify the requirements as less or more important or more specifically as desirable or essential. An identifier mark can be used with every requirement to indicate its rank or stability.

Modifiability:

SRS should be made as modifiable as possible and should be capable of easily accepting changes to the system to some extent. Modifications should be properly indexed and cross-referenced.

Verifiability:

A SRS is verifiable if there exists a specific technique to quantifiably measure the extent to which every requirement is met by the system. For example, a

requirement starting that the system must be user-friendly is not verifiable and listing such requirements should be avoided.

Traceability:

One should be able to trace a requirement to design component and then to code segment in the program. Similarly, one should be able to trace a requirement to the corresponding test cases.

Design Independence:

There should be an option to choose from multiple design alternatives for the final system. More specifically, the SRS should not include any implementation details.

Testability:

A SRS should be written in such a way that it is easy to generate test cases and test plans from the document.

Understandable by the customer:

An end user maybe an expert in his/her specific domain but might not be an expert in computer science. Hence, the use of formal notations and symbols should be avoided to as much extent as possible. The language should be kept easy and clear.

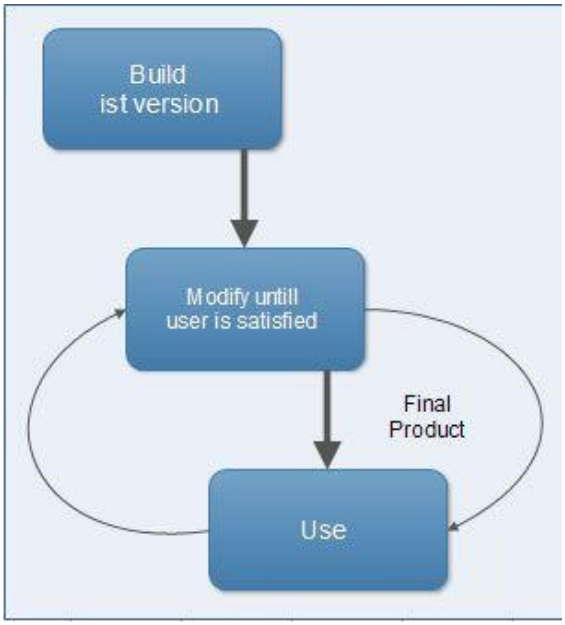
Right level of abstraction:

If the SRS is written for the requirements phase, the details should be explained explicitly. Whereas, for a feasibility study, fewer details can be used. Hence, the level of abstraction varies according to the purpose of the SRS.

What is build and fix model or ad hoc model? and Explain its Advantages and Disadvantages

In the **build and fix model** (also referred to as an **ad hoc model**), the software is developed without any specification or design. An initial product is built, which is then repeatedly modified until it (software) satisfies the user. That is, the software is developed and delivered to the user. The user checks whether the

desired functions ‘are present. If not, then the software is changed according to the needs by adding, modifying or deleting functions. This process goes on until the user feels that the software can be used productively. However, the lack of design requirements and repeated modifications result in loss of acceptability of software. Thus, software engineers are strongly discouraged from using this development approach.



This model includes the following two phases.

- **Build:** In this phase, the software code is developed and passed on to the next phase.
- **Fix:** In this phase, the code developed in the build phase is made error free. Also, in addition to the corrections to the code, the code is modified according to the user’s requirements.

Various advantages and disadvantages associated with the build and fix model are listed in Table.

Advantages and Disadvantages of Build and Fix Model

Advantages

1. Requires less experience to execute or manage other than the ability to program.
2. Suitable for smaller software.

3. Requires less project planning.

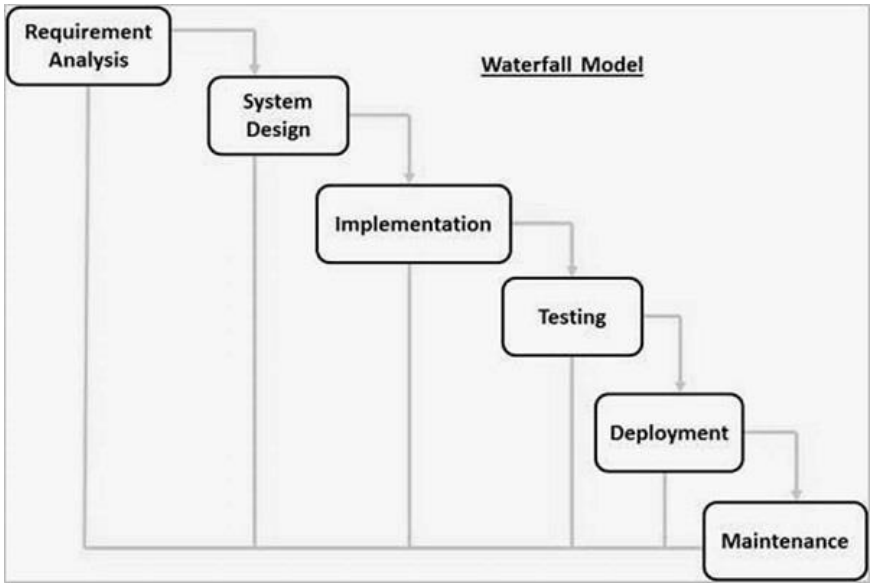
Disadvantages

1. No real means is available of assessing the progress, quality, and risks.
2. Cost of using this process model is high as it requires rework until user's requirements are accomplished.
3. Informal design of the software as it involves unplanned procedure.
4. Maintenance of these models is problematic.

The 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. It is very simple to understand and use.

The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.



The sequential phases in Waterfall model are −

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.

Waterfall Model - Advantages

- Simple and easy to understand and use
- Easy to manage due to the rigidity(not able to be changed) of the model. Each phase has specific deliverables and a review process.
- Works well for smaller projects where requirements are very well understood.
- Easy to arrange tasks.
- Process and results are well documented.

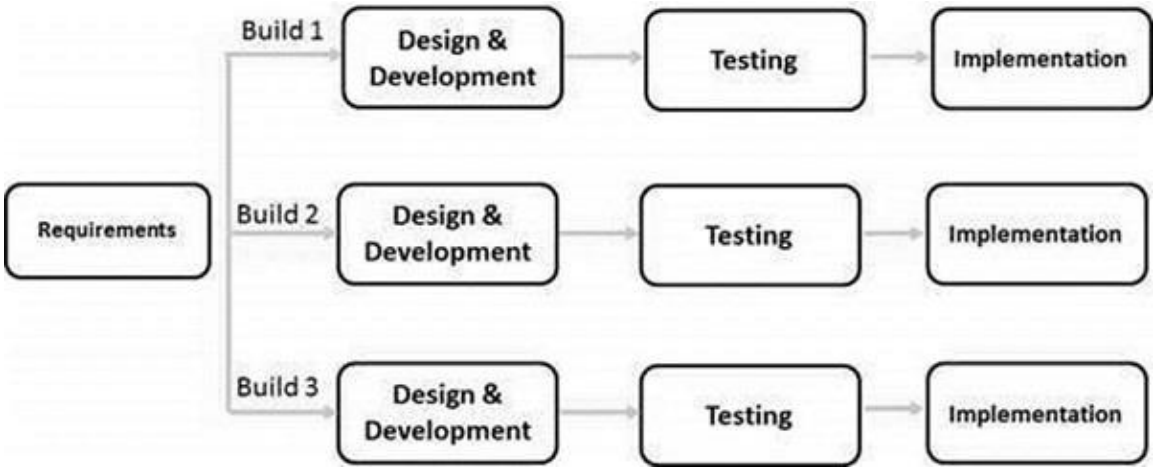
Waterfall Model - Disadvantages

The major disadvantages of the Waterfall Model are as follows −

- No working software is produced until late during the life cycle.
- Not a good model for complex and object-oriented projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.
- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.

Iterative Model

In the Iterative model, iterative process starts with a simple implementation of a small set of the software requirements and iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed.



In this incremental model, the whole requirement is divided into various builds. During each iteration, the development module goes through the requirements, design, implementation and testing phases. Each subsequent release of the module adds function to the previous release. The process continues till the complete system is ready as per the requirement.

Advantages

- Some working functionality can be developed quickly and early in the life cycle.
- Results are obtained early and periodically.
- Parallel development can be planned.
- Progress can be measured.
- Less costly to change the scope/requirements.
- Easier to manage risk - High risk part is done first.

Disadvantages

- More resources may be required.
- Although cost of change is lesser, but it is not very suitable for changing requirements.
- More management attention is required.
- Not suitable for smaller projects.
- Management complexity is more.
- Highly skilled resources are required for risk analysis.

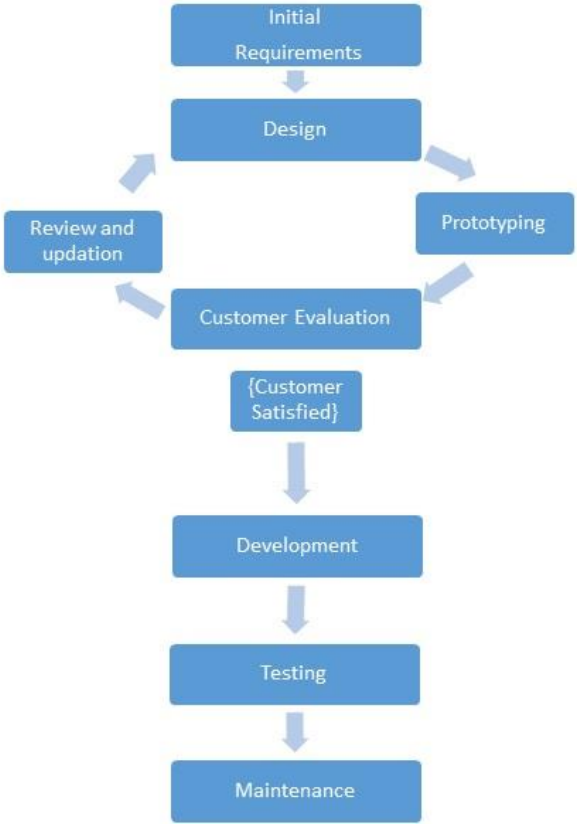
Prototyping Model

Prototype is a working model of software with some limited functionality.

Software prototyping is becoming very popular as a software development model, as it enables to understand customer requirements at an early stage of development. It helps get valuable feedback from the customer and helps software designers and developers understand about what exactly is expected from the product under development.

Advantage of Prototype Model

- Reduce the risk of incorrect user requirement
- Good where requirement are changing/uncommitted
- Support early product marketing
- Reduce Maintenance cost.
- Errors can be detected much earlier as the system is made side by side.



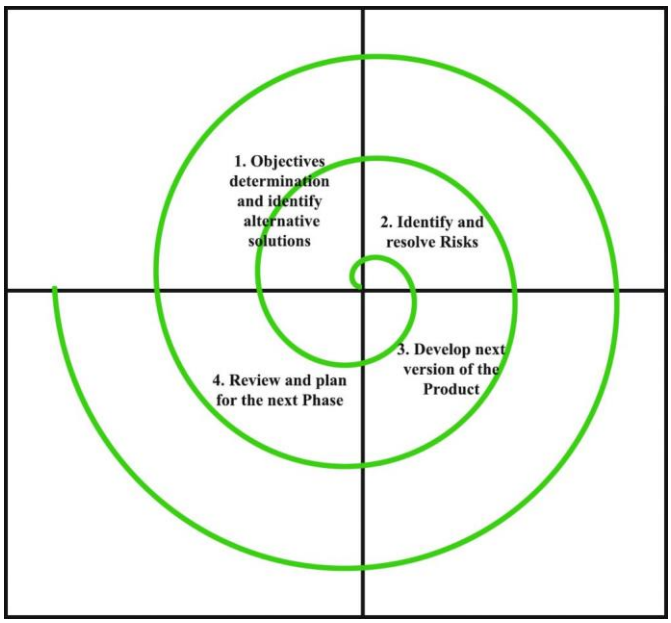
Disadvantage of Prototype Model

- Require extensive customer collaboration
- May be too customer specific, no broad market
- Difficult to know how long the project will last.
- Special tools & techniques are required to build a prototype.
- It is a time-consuming process.

Spiral Model

Spiral model is one of the most important Software Development Life Cycle models, which provides support for Risk Handling. In its diagrammatic representation, it looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project.

Each phase of the Spiral Model is divided into four quadrants as shown in the above figure. The functions of these four quadrants are discussed below-



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.

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.

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.

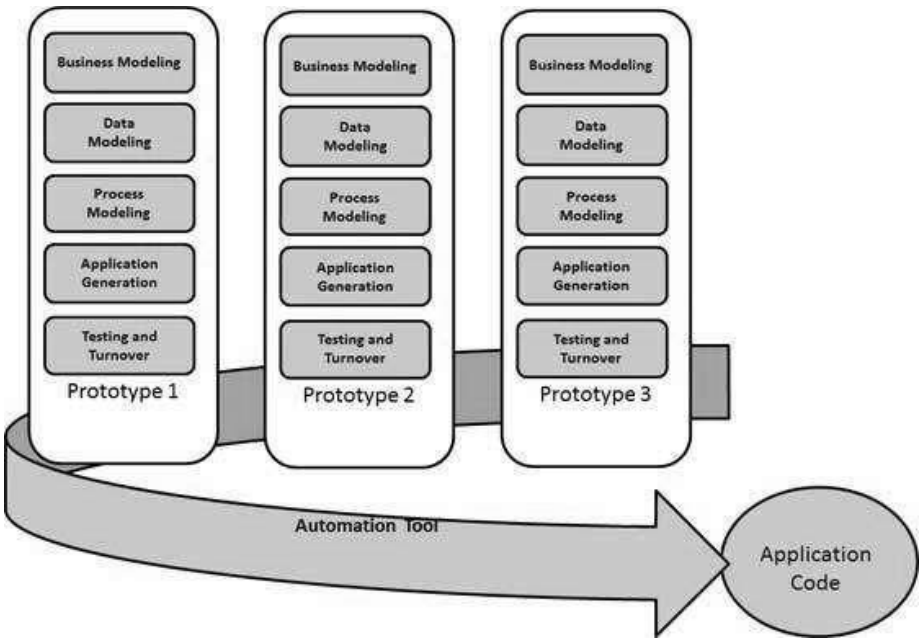
Too much dependability on Risk Analysis: The successful completion of the project is very much dependent on Risk Analysis. Without very highly experienced experts, it is going to be a failure to develop a project using this model.

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

Rapid application development

Rapid application development is a software development methodology that uses minimal planning in favor of rapid prototyping.

In the RAD model, the functional modules are developed in parallel as prototypes and are integrated to make the complete product for faster product delivery. Since there is no detailed preplanning, it makes it easier to incorporate the changes within the development process.



Business Modelling

A complete business analysis is performed to find the vital information for business, how it can be obtained, how and when is the information processed and what are the factors driving successful flow of information.

Data Modelling

The information gathered in the Business Modelling phase is reviewed and analyzed to form sets of data objects vital for the business. The attributes of all data sets is identified and defined. The relation between these data objects are established and defined in detail.

Process Modelling

The process model for any changes or enhancements to the data object sets is defined in this phase. Process descriptions for adding, deleting, retrieving or modifying a data object are given.

Application Generation

The actual system is built and coding is done by using automation tools to convert process and data models into actual prototypes.

Testing and Turnover

The data flow and the interfaces between all the components need to be thoroughly tested with complete test coverage. Since most of the programming components have already been tested, it reduces the risk of any major issues.

The advantages of the RAD Model are as follows −

- Changing requirements can be accommodated.
- Progress can be measured.
- Reduced development time.
- Increases reusability of components.
- Integration from very beginning solves a lot of integration issues.

The disadvantages of the RAD Model are as follows −

- Dependency on technically strong team members for identifying business requirements.
- Only system that can be modularized can be built using RAD.
- Requires highly skilled developers/designers.
- Management complexity is more.
- Requires user involvement throughout the life cycle.

Selection Process Criteria for a SDLC Model

Selection Process parameters plays an important role in software development as it helps to choose the best suitable software life cycle model. Following are the parameters which should be used to select a SDLC.

1. Requirements characteristics :

- Reliability of Requirements
- How often the requirements can change
- Types of requirements
- Number of requirements
- Can the requirements be defined at an early stage

- Requirements indicate the complexity of the system

2. Development team :

- Team size
- Experience of developers on similar type of projects
- Level of understanding of user requirements by the developers
- Environment
- Experience on technologies to be used
- Availability of training

3. User involvement in the project :

- Expertise of user in project
- Involvement of user in all phases of the project
- Experience of user in similar project in the past

4. Project type and associated risk :

- Stability of funds
- Tightness of project schedule
- Availability of resources
- Type of project
- Size of the project
- Expected duration for the completion of project
- Complexity of the project
- Level and the type of associated risk