SDLC Activities

SDLC provides a series of steps to be followed to design and develop a software product efficiently. SDLC framework includes the following steps:

(a) **Software Analysis:** This phase studies the problem or requirements of software in detail. These requirements define the processes to be managed during the software development. After analyzing the requirements of the user, a requirement statement known as software requirement specification (SRS) is developed. After analyses, planning for the project begins. It includes developing plans that describes the activities to be performed during the project, such as software configuration management plans, project and scheduling, and the quality assurance plans. In addition, the resources required during the project are also determined.

(b) **Software Design:** In this phase the requirements are given a 'defined' form. Design can be defined as a process of deciding information structures, in terms of efficiency, flexibility, and reusability. During this phase, strategic and tactical decisions are made to meet the required functional and quality requirements of a system. Software design serves as the blueprint for the implementation of requirement in the software system. Each element of the analysis model in the analysis phase provides information that is required to create design models. The requirement specification of software, together with data, functional, and behavioral models provides a platform to feed the design task to meet required functional and quality requirements of a system.

(c) **Software Coding:** This phase can be defined as a process of translating the software requirements into a programming language using tools that are available. Writing a software code requires a thorough knowledge of programming language and its tools. Therefore, it is important to choose the appropriate programming language according to the user requirements. A program code is efficient if it makes optimal use of resources and contains minimum errors.

(d) **Software Testing:** This testing is performed to assure that software is free from errors. Efficient testing improves the quality of software. To achieve this, software testing requires a thorough analysis of the software in a systematic manner. Test plan is created to test software in a planned and systematic manner. In addition, software testing is performed to ensure

that software produces the correct outputs. This implies that outputs produced should be according to user requirements.

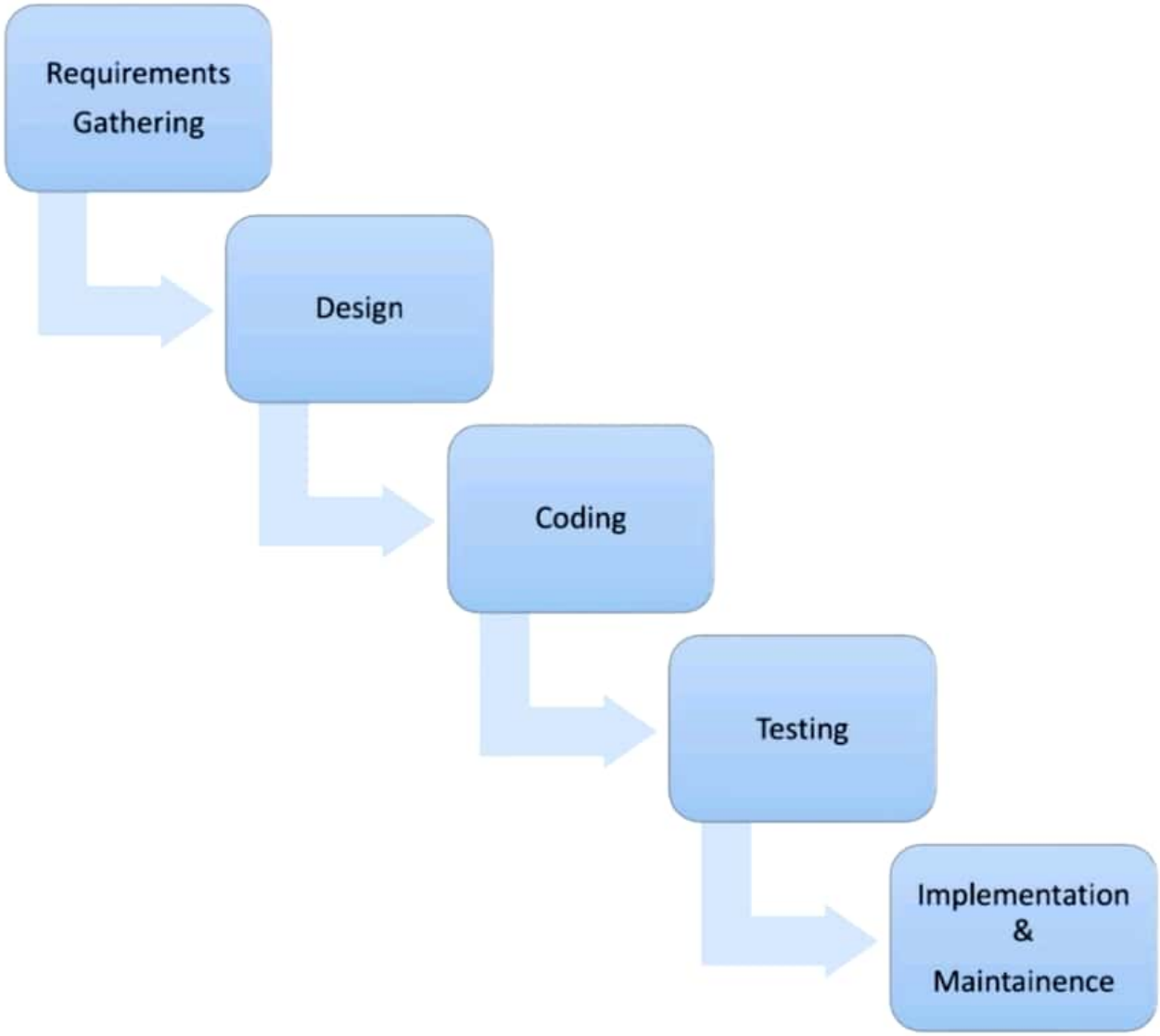
(e) **Software Maintenance**: This phase comprises of a set of software engineering activities that occur after software is delivered to the user. The objective of software maintenance is to make software operational according to user requirements. The need of software maintenance is to provide continuity of service. This implies that software maintenance focuses on fixing errors, recovering from failures, such as hardware failures, or incompatibility of hardware with software. In addition, it facilitates future maintenance work by modifying the software code and databases used in the software.

Waterfall Model

In waterfall model (also known as classical life cycle model) the development of software proceeds linearly and sequentially from requirement analysis to design, coding, testing, integration, implementation, and maintenance. Thus, this model is also known as linear sequential model.

This model is simple to understand, and represents processes, which are easy to manage and measure. Waterfall model comprises of different phases and each phase has its distinct goal. Figure shows that once a phase is completed, the development of software proceeds to the next phase. Each phase modifies the intermediate product to develop a new

product as an output. The new product becomes the input of the next process.



- **Requirement analysis:** Focuses on the requirements of the software which is to be developed. It determines the processes that are incorporated during the development of software. To specify the requirements' users specification should be clearly understood and the requirements should be analysed. This phase involves interaction between user

and software engineer, and produces a document known as software requirement specification (SRS).

- **Design:** Determines the detailed process of developing software after the requirements are analysed. It utilises software requirements defined by the user and translates them into a software representation. In this phase, the emphasis is on finding a solution to the problems defined in the requirement analysis phase. The software engineer, in this phase is mainly concerned with the data structure, algorithmic detail, and interface representations.

- **Coding:** Emphasises on translation of design into a programming language using the coding style and guidelines. The programs created should be easy to read and understand. All the programs written are documented according to the specification.

- **Testing**: Ensures that the product is developed according to the requirements of the user. Testing is performed to verify that the product is functioning efficiently with minimum errors. It focuses on the internal logics and external functions of the software and ensures that all the statements have been exercised (tested). Note that testing is a multi-stage activity, which emphasises verification and validation of the product.

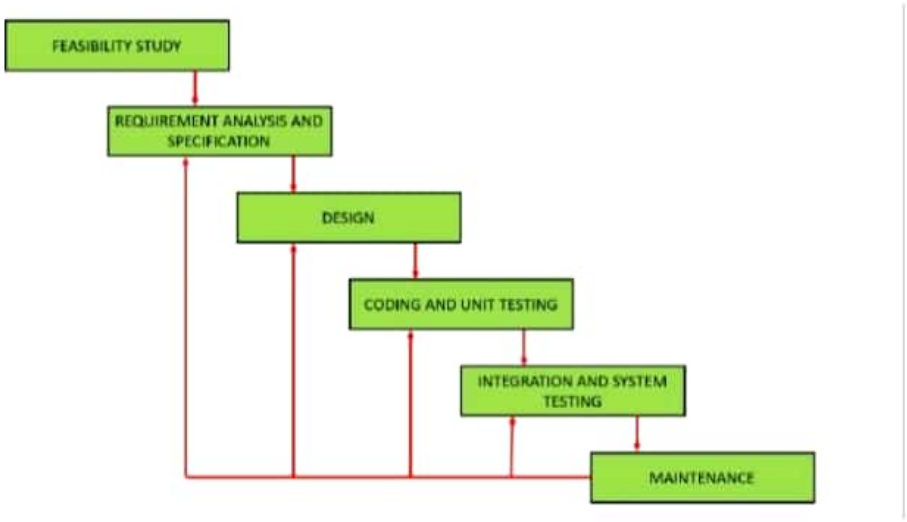
- **Implementation and maintenance:** Delivers fully functioning operational software to the user. Once the software is accepted and deployed at the user's end, various changes occur due to changes in external environment (these include upgrading new operating system or addition of a new peripheral device). The changes also occur due to changing requirements of the user and the changes occurring in the field of technology. This phase focuses on modifying software, correcting errors, and improving the performance of the software.

Advantages	Disadvantages
• Relatively simple to understand • Each phase of development proceeds sequentially. • Allows managerial control where a schedule with deadlines is set for each stage of development. • Helps in controlling schedules, budgets, and documentation.	• Requirements need to be specified before the development proceeds. • The changes of requirements in later phases of the waterfall model cannot be done. This implies that once an application is in the testing phase, it is difficult to incorporate changes at such a late phase. • No user involvement and working version of the software is available when the software is developed. • Does not involve risk management.

Iterative Waterfall Model

In a practical software development project, the classical waterfall model is hard to use. So, Iterative waterfall model can be thought of as incorporating the necessary changes to the classical waterfall model to make it usable in practical software development projects. It is almost same as the classical waterfall model except some changes are made to increase the efficiency of the software development.

The iterative waterfall model provides feedback paths from every phase to its preceding phases, which is the main difference from the classical waterfall model.



When errors are detected at some later phase, these feedback paths allow correcting errors committed by programmers during some phase. The feedback paths allow the phase to be reworked in which errors are committed and these changes are reflected

in the later phases. But, there is no feedback path to the stage – feasibility study, because once a project has been taken, does not give up the project easily. It is good to detect errors in the same phase in which they are committed. It reduces the effort and time required to correct the errors.

Phase Containment of Errors: The principle of detecting errors as close to their points of commitment as possible is known as Phase containment of errors.

Advantages of Iterative Waterfall Model

- **Feedback Path:** In the classical waterfall model, there are no feedback paths, so there is no mechanism for error correction. But in iterative waterfall model feedback path from one phase to its preceding phase allows correcting the errors that are committed and these changes are reflected in the later phases.
- **Simple:** Iterative waterfall model is very simple to understand and use. That's why it is one of the most widely used software development models.

Drawbacks of Iterative Waterfall Model

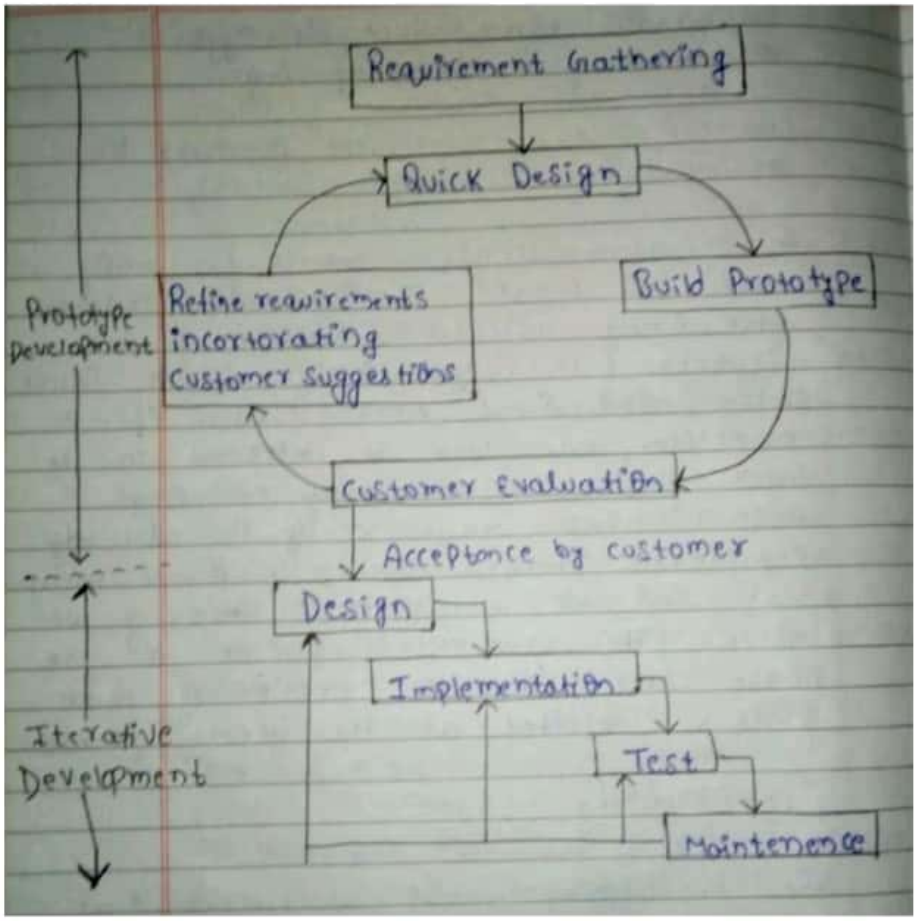
- **Difficult to incorporate change requests:** The major drawback of the iterative waterfall model is that all the requirements must be clearly stated before starting of the development phase. Customer may change requirements after some time but the iterative waterfall model does not leave any scope to incorporate change requests that are made after development phase starts.
- **Incremental delivery not supported:** In the iterative waterfall model, the full software is completely developed and tested before delivery to the customer. There is no scope for any intermediate delivery. So, customers have to wait long for getting the software.
- **Overlapping of phases not supported:** Iterative waterfall model assumes that one phase can start after completion of the previous phase, But in

real projects, phases may overlap to reduce the effort and time needed to complete the project.

- **Risk handling not supported:** Projects may suffer from various types of risks. But, Iterative waterfall model has no mechanism for risk handling.

Prototyping Model

The prototyping model is applied when there is an absence of detailed information regarding input and output requirements in the software. Prototyping model is developed on the assumption that it is often difficult to know all the requirements at the beginning of a project. It is usually used when there does not exist a system or in case of large and complex system where there is no manual process to determine the requirements.



Prototyping model increases flexibility of the development process by allowing the user to interact and experiment with a working representation of the product known as **prototype**.
A prototype gives the user an actual feel of the system.

1. Requirements gathering and analysis: Prototyping model begins with requirements analysis, and the requirements of the system are defined in detail. The user is interviewed to know the requirements of the system.

2. Quick design: When requirements are known, a preliminary design or a quick design for the system is created. It is not a detailed design, however, it includes the important aspects of the system, which gives an idea of the system to the user. Quick design helps in developing the prototype.

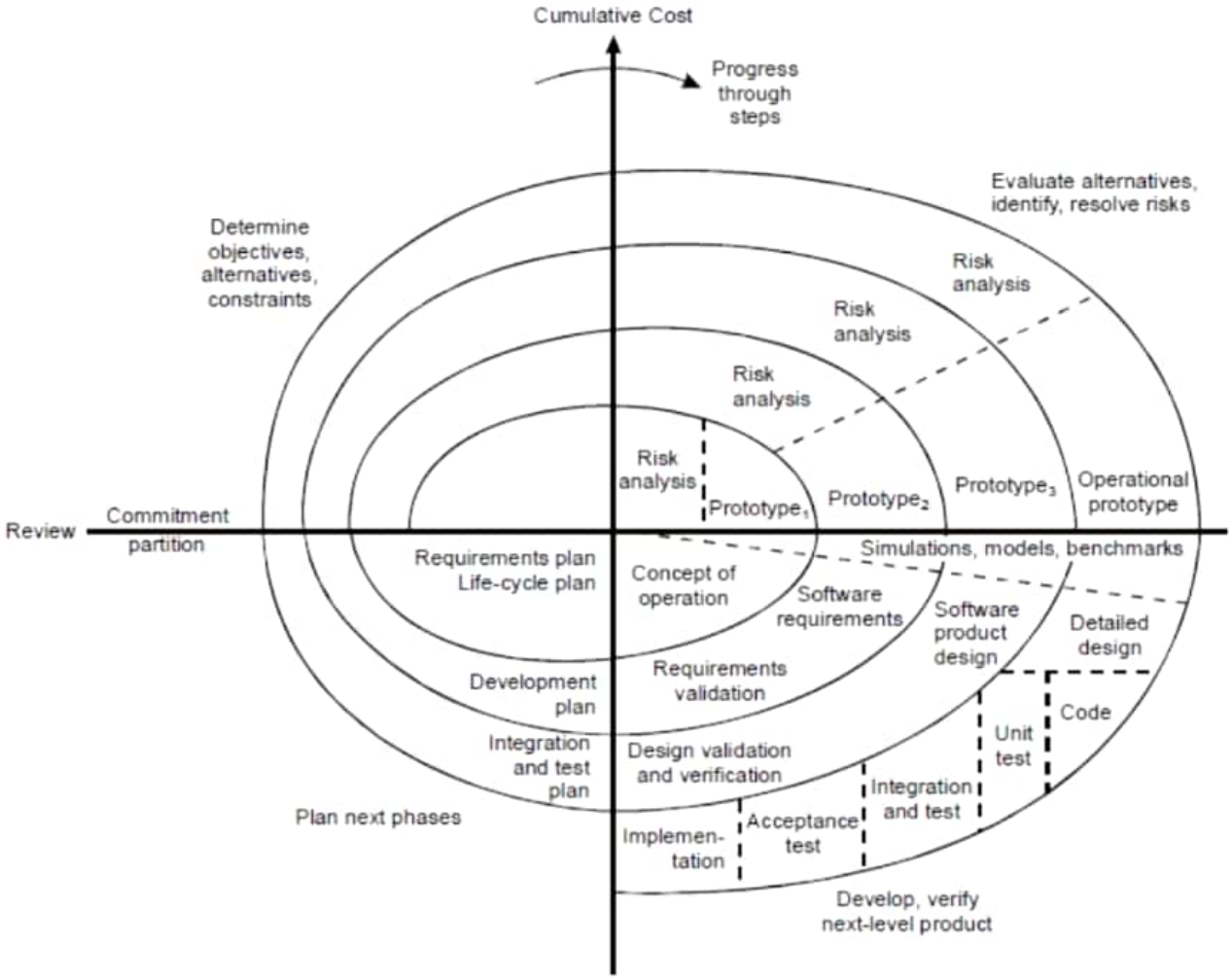
3. Build prototype: Information gathered from quick design is modified to form a prototype. The first prototype of the required system is developed from quick design. It represents a 'rough' design of the required system.

4. Assessment or user evaluation: Next, the proposed system is presented to the user for consideration as a part of development process. The users thoroughly evaluate the prototype and recognize its strengths and weaknesses, such as what is to be added or removed. Comments and suggestions are collected from the users and are provided to the developer.

5. Prototype refinement: Once the user evaluates the prototype, it is refined according to the requirements. The developer revises the prototype to make it more effective and efficient according to the user requirement. If the user is not satisfied with the developed prototype, then a new prototype is developed with the additional information provided by the user. The new prototype is evaluated in the same manner, as the previous prototype. This process continues until all the requirements specified by the user are met. Once the user is satisfied with the developed prototype, a final system is developed based on the final prototype.

Advantages	Disadvantages
• Provides a working model to the user early in the process, enabling early assessment and increasing user confidence. • There is a great involvement of users in software development. Hence, the requirements of the users are met to the greatest extent. • Helps in reducing risks associated with the project.	• If the user is not satisfied by the developed prototype, then a new prototype is developed. This process goes on until a perfect prototype is developed. Thus, this model is time consuming and expensive. • Prototyping can lead to false expectations. It often creates a situation where user believes that the development of the system is finished when it is not.

Spiral Model



In 1980's Boehm introduced a process model known as spiral model. The spiral model comprises of activities organized in a spiral, which has many cycles. This model combines the features of prototyping model and waterfall model and is advantageous for large, complex and expensive projects. The spiral model determines requirement problems in developing the prototypes. In addition, spiral model guides and measures the need of risk management in each cycle of the spiral model. IEEE defines spiral

model as **"a model of the software development process in which the constituent activities, typically requirements analysis, preliminary and detailed design, coding, integration, and testing, are performed iteratively until the software is complete."**

The objective of spiral model is to emphasize management to evaluate and resolve risks in the software project. Different areas of risks in the software project are project overruns, changed requirements, loss of key project personnel, delay of necessary hardware, competition from other software developers, and technological breakthroughs, which obsolete the project. the steps involved in the model are listed below:

1. Each cycle of the first quadrant commences with identifying the goals for that cycle. In addition, it determines other alternatives, which are possible in accomplishing those goals.

2. The next step in the cycle evaluates alternatives based on objectives and constraints. This process identifies the areas of uncertainty and focuses on significant sources of the project risks. Risk signifies that there is a possibility that the objectives of the project cannot be accomplished. If so the formulation of a cost effective strategy for resolving risks is followed. Figure shows the strategy, which includes prototyping, simulation, benchmarking administrating user questionnaires or risk resolution technique.

3. The development of the software depends on remaining risks. The third quadrant develops the final software while considering the risks that can occur. Risk management considers the time

and effort to be devoted to each project activity, such as planning, configuration management, quality assurance, verification, and testing.

4. The last quadrant plans the next step, and includes planning for the next prototype and thus, comprises of requirements plan, development plan, integration plan, and test plan.

One of the key features of the spiral model is that each cycle is completed by a review conducted by the individuals or users. This includes the review of all the intermediate products, which are developed during the cycles. In addition, it includes the plan for next cycle and the resources required for the cycle.

Advantages	Disadvantages
• Avoids the problems resulting in risk-driven approach in the software. • Specifies a mechanism for software quality assurance activities. • Spiral model is utilised by complex and dynamic projects. • Re-evaluation after each step allows changes in user perspectives, technology advances or financial perspectives.	• Assessment of project risks and its resolution is not an easy task. • Difficult to estimate budget and schedule in the beginning, as some of the analysis is not done until the design of the software is developed.