# Software Development Life Cycle

Software is a collection of multiple actions designed to accomplish specific tasks. Software is a collection of instructions that specify how a system should operate. To develop any software, we must adhere to a set of procedures. A Software Development Life Cycle (SDLC) is comprised of critical phases that are critical for developers, including planning, analysis, design, and implementation, which are discussed in detail in the section below. There have been several software development life cycle (SDLC) models developed: waterfall, spiral, V-Model, rapid prototyping, incremental, and synchronize and stabilize.

## Traditional Phases of SDLC

The Software Development Life Cycle (SDLC) is a term that refers to the stages of work that go into developing software applications. Each phase denotes a role or responsibility that software contributors must comprehend, manage, and optimize to deliver their software services quickly and efficiently. These stages of work include the following:

* Requirements gathering
* Software Design
* Software Development
* Test and Integration
* Deployment
* Operationalization and Maintenance.

### Requirements Gathering

The team identifies, collects, and defines current problems, requirements, requests, and customer expectations regarding the software application or service during this stage of work. Defining software or product requirements equips teams with the foresight and context necessary to successfully deliver and produce software solutions.

### Software Design

The team makes software design decisions during this phase of work, including the architecture and implementation of the software solution. This may include the creation of design documents and coding guidelines, as well as discussions about the tools, practices, runtime environments, and frameworks.

### Software Development

By implementing the solution, teams achieve the goals and outcomes established during the software requirements gathering phase. The development process may involve multidisciplinary teams, novel technologies, and unexpected obstacles; however, development teams typically collaborate with technology leads and product or project managers to unblock the process, make decisions, or provide support.

### Test And Integration

This phase of work involves packaging and testing a software implementation to ensure its quality. Testing or quality assurance ensures that implemented solutions meet established quality and performance standards.

### Deployment

This process entails provisioning infrastructure on-premises or in the cloud and developing a software deployment strategy for delivering changes to a customer.

### Operationalization And Maintenance

This stage of work involves operationalizing the software to ensure that there are no issues or incidents during deployment. This process may include incident resolution or management if any issues or changes affecting a customer or user base occur.

**Waterfall Model**

* Waterfall model is the Linear, Sequential development model.
* Requirement should be clear before going to next phase of design.
* Testing is carried out once the code has been fully developed. Each work-product or activity is completed before moving on to next
* Each phase of development proceeds in order without any overlapping.
* Each phase schedule for the tasks to be completed within a specified time period
* The documentation and testing happens at the end of each phase, which helps in maintaining the quality of the project.
* In the waterfall model each step is frozen before the next step. That is the requirements are frozen before the design starts, and once the design is frozen the coding starts etc. But what will the testing team do till then so is very time consuming and high costing
* In waterfall model the defect were found very late in the development life cycle as test team was not involved from the beginning of the project.
* Tester role will be involved in testing phase only

Diagram

Description automatically generated

Fig1: Waterfall Model Life Cycle

**Pros**

* Requirement is clear before development starts.
* Each phase is completed in specified period after that it moves to next phase.
* The number of resources required to implement this model are minimal.
* Each phase proper documentation is followed for the quality of the development.

**Cons**

* The problems with one phase are never solved completely during that phase and in fact many problems regarding a particular phase arise after the phase is signed off, this result in badly structured system.
* If client want the requirement to be changed, it will not be implemented in the current development process

1. **V-Model (Modified version of Water fall Model)**

* The V model (Validation & Verification model)[11] is a modified version of the Waterfall method
* As opposed to the Waterfall method, this one was not designed in a linear axis; instead the stages turn back upwards after the coding phase is done.
* This developmental process is balanced and relies on the verification from the previous steps before proceeding forward.
* The product from every phase needs to be checked and approved before moving forward.
* In v model developer and tester works parallel.
* In V model, based on the requirements the System test cases are prepared, and based on the HLD (High level document)the Integration Test cases are prepared, and based on the LLD(Low-level document)the Integration Test cases are prepared. And then the coding is done. Once coding is completed, unit, integration and system testing happens in the sequence.
* In V-model, gives relationship between each development stages and Testing stages.

Diagram

Description automatically generated

*Figure 2: V- Model Life Cycle*

* 1. **Pros**
* Same as Waterfall model
* V-Model, the advantage is that Tester role will be involved in the requirement phase itself.
* Requirement Changes is possible in any phase.
  1. **Cons**
* The biggest disadvantage of V-model is that it’s very rigid and the least flexible.
* If any changes happen mid way, not only the requirements documents but also the test documentation needs to be updated.
* It is not proposed for short term projects as it requires reviews at each stage.

In our case study, if a client wishes to modify a requirement, it is possible to do so; however, documentation created during the requirement phase, such as functional specifications, high level design, low level design, unit testing, system testing, and integration testing, must be updated. Generally, the V-Model is used in larger organisations due to the increased resource requirements.

1. Incremental Model

The incremental model is a method of software development in which requirements are broken down into multiple self-contained modules throughout the software development cycle. Each module in this model goes through the phases of requirements, design, implementation, and testing. Each subsequent release of the module incorporates new functionality. The process is repeated until the entire system is achieved.

Diagram

Description automatically generated

Fig3; Incremental Model

**4.1. The various phases of incremental model are as follows:**

**1. Requirement analysis:** The incremental model's first phase involves the identification of requirements by the product analysis expertise. Additionally, the requirement analysis team comprehends the system's functional requirements. This phase is critical for developing software incrementally.

**2. Design & Development:** This phase of the SDLC's incremental model successfully completes the design of the system's functionality and development method. When software develops new functionality, the incremental model makes use of style and phase development.

**3. Testing:** The testing phase of the incremental model verifies the performance of each existing function as well as any new functionality. Various methods are used to evaluate the behaviour of each task during the testing phase.

**4. Implementation:** The implementation phase enables the development system's coding phase. It entails the final coding of the designs created during the designing and development phases, as well as the testing of the functionality created during the testing phase. After this phase is complete, the number of working products is increased and upgraded to the final system product.

**4.2. 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.

**4.3. 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.

**4.4. Disadvantage of Incremental Model**

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

1. **The Iterative Model**

Iterative development is a method of scheduling rework in which time is set aside to revise and improve various components of the system. In this case, feature code is designed, developed, and tested in cyclical fashion. A detailed requirement is not required to begin, as improvement begins with identifying and executing software components. This procedure is then repeated to create a new version of the software at the end of each model cycle. Iterative development breaks down the product development of a large application into manageable pieces. As such, it is defined as the process of segmenting the development of a large application into smaller chunks.

Diagram

Description automatically generated

Fig 4: Iterative Model

* 1. **Advantages**

A significant feature of the iterative model is that it starts with a working model of the entire system, which makes it easier to quickly identify configuration errors and implement corrective actions in terms of cost planning. Several other noteworthy characteristics of the model include the following:

**In born Variation:** It is fairly obvious that the majority of software development life cycles will include some type of variation demonstrating the product's deployment phase at a specific stage. Nonetheless, the iterative model simplifies this significantly by ensuring that more current emphases are incrementally enhanced renditions of previous cycles.

**Fast Turnaround:** While it may appear that each phase of the iterative model is similar to the phases of a more conventional model such as the waterfall model—and thus that the procedure will take a long time—the beauty of the iterative procedure is that each stage can be successfully shortened into smaller time allotments; whatever is necessary to meet the task's requirements or constraints.While it may appear that each phase of the iterative model is similar to the phases of a more conventional model such as the waterfall model—and thus that the procedure will take a long time—the beauty of the iterative procedure is that each stage can be successfully shortened into smaller time allotments; whatever is necessary to meet the task's requirements or constraints.

**Suited for Agile Associations:** While a well-organized process such as the waterfall model may work admirably well for large associations involving several colleagues, the iterative model truly shines when applied to a smaller, more nimble group. When combined with the energy of current form control frameworks, a complete "cycle process" can be adequately performed by a group of individual colleagues, from planning and organising to executing and testing, with little or no outside input or assistance.

**Easy Adaptability:** A critical advantage of the iterative model is its ability to rapidly adapt to the constantly changing requirements of both the undertaking and the customer. Indeed, even significant changes to the hidden code structure or usage are possible.

**Enables Risk Analysis:** This is also referred to as "Project Impact Analysis" and is performed during the analysis phase of the SDLC in order to identify inconsistencies in the system requirements. The Iterative model incorporates a process for identifying and evaluating project/goal success factors.

**Progress Measurement:** Additionally, the Iterative model supports progress measurement. Following each iteration, stakeholders can observe the tangible development of the system, and the development team can track progress following each build. This helps to avoid project stagnation and serves as a motivator for the team.

* 1. **The Disadvantages**

The primary disadvantage of the Iterative SDLC Model is that it is only suitable for large-scale system or software development projects; it is difficult to break down a small programming framework into manageable augmentations/modules. The flaws are extremely prevalent.

**Costly Late-Stage Issues:** While this is not a concern for all activities, because of the minimal pre-coding and usage preparation required when using an iterative model, it is possible that an unanticipated issue in the plan or hidden framework design will surface late in the task. Solving this could potentially have a detrimental effect on the overall duration and cost of the project, necessitating numerous future cycles to resolve a single issue.

**Increased Pressure on User Engagement:** On the other hand, Unlike the waterfall model, which places a premium on client/customer engagement during the initial phases of the venture during a brief time to get down to business period, the iterative model frequently necessitates client engagement throughout the procedure. This is frequently an appalling commitment, as each new emphasis will almost certainly require testing and input from clients in order to legitimately assess any significant changes.

**Feature Creep:** Not only does the iterative model require client feedback throughout the process, but it also implies that the task may be prone to undesirable element crawl, in which clients encounter changes in each cycle and are slanted to continually advance new demands for additional features to be added to future forms.

**Management Complexity:** As a disadvantage of complexity management, its methodology focuses on the analysis and optimization of enterprise complexity. Because the effects of complexity affect all business processes along the value chain, complexity management requires a holistic approach, and management will be extremely difficult due to iteration on each build or development.

1. **Spiral Model**

Boehm first proposed the spiral model, which is a critical software development life cycle model. It is a synthesis of evolutionary, waterfall, and prototyping modelling techniques. It is typically used for projects that are too large, too expensive, or too complicated to manage on their own. The spiral model is unique in its ability to manage risks. Although the spiral model is represented by a spiral, the number of loops is subjective and entirely dependent on the project. The greater the wind speed, the greater the project payload, and the continuous phase's success is determined by the angular rate.

Due to the fact that the Spiral Model is consistent with the development process, i.e. it poses less risk to development companies and customers, it is widely used in the code business. The spiral model is typically used when risk assessment is critical and there is a budget constraint, on projects involving moderate to high risk, and on projects with compound needs.

* 1. DESIGN OF SPIRAL MODEL

The spiral model instils the concept of repetitive development with an organised, supervised component of the waterfall model. This spiral model combines a recurring process with a successive model, resulting in a waterfall model with a strong emphasis on risk analysis. Per processor, additional product releases or filters are possible.

The spiral model is a critical component of the Software Development Life Cycle. This model enables risk management. It appears to be a spiral-like structure composed of several loops in the following diagram. The precise number of spiral loops is unknown and may vary by project. Each spiral loop represents a phase of the Software Development Process. The appropriate number of steps required for product development may be changed by the project's administrator based on the project's risk assessment. Because the project manager accurately determines the number of steps, the project manager plays a critical role in the development of software using the Spiral model.

The radius of the model can be used to determine the price of the software product. Additionally, the angular attributes can be used to determine the product's development thus far in the current cycle.

Diagram

Description automatically generated

Fig 5: Spiral Model

* 1. HANDLING OF RISK IN SPIRAL MODEL

Risk is defined as any unfavourable event that could jeopardise the successful completion of a project. The most critical feature of this model is its organisation and management of unknown risks that arise during development. This type of risk assessment is easily facilitated by the development of a specific type of prototype. The spiral model assists in mitigating such risks by providing a build-up model at each stage of software development.

Additionally, the Prototyping Model aids in risk management. However, risks must be fully assessed prior to beginning work on the software project's development. However, in the case of a genuine project, risk can be identified after development work has begun. As a result, we cannot use the Prototyping Model in this case. At each phase of the spiral model, product characteristics are analysed, and existing risks are identified and resolved. As a result, this model is much more workable when combined with other models.

* 1. **Advantages**  
     Below are some advantages of the Spiral Model.

1. **Risk Handling:** Spiral Model is the best development model to follow for projects with numerous unknown risks that occur throughout the development process, as it incorporates risk analysis and risk management at each phase.
2. **Good for large projects:** It is recommended to use the Spiral Model in large and complex projects.
3. **Flexibility in Requirements:** Change requests in the Requirements at later phase can be incorporated accurately by using this model.
4. **Customer Satisfaction:** Customers can observe the product's development during the early stages of software development, and thus become familiar with the system by using it prior to the final product's completion.
   1. **Disadvantages**  
      Below are some main disadvantages of the spiral model.
5. **Complex:** The Spiral Model is much more complex than other SDLC models.
6. **Expensive:** Spiral Model is not suitable for small projects as it is expensive.
7. **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.
8. **Difficulty in time management:** As the number of phases is unknown at the start of the project, so time estimation is very difficult.
9. **Prototyping Model**

Prototyping is defined as the process of developing a working replication of a product or system that has to be engineered. It offers a small scale facsimile of the end product and is used for obtaining customer feedback as described below:

Diagram

Description automatically generated

Fig: 6 Prototyping Model

The Prototyping Model is one of the most widely used phases of the Software Development Life Cycle (SDLC models). This model is used when customers are unsure of the precise project requirements in advance. In this model, an end product prototype is developed, tested, and refined repeatedly based on customer feedback until a final acceptable prototype is achieved, which serves as the foundation for developing the final product.

The system is partially implemented prior to or during the analysis phase in this process model, allowing customers to see the product early in its life cycle. The process begins with customer interviews and the development of an incomplete high-level paper model. This document is used to create the initial prototype, which will only include the bare minimum functionality requested by the customer. After the customer identifies the issues, the prototype is fine-tuned to eliminate them. The process is repeated until the user approves the prototype and is satisfied with the working model.

There are four types of model available:

**Rapid Throwaway Prototyping:** This technique is advantageous for exploring ideas and obtaining customer feedback on each one. A developed prototype does not have to be included in the final accepted prototype in this method. Customer feedback assists in avoiding unnecessary design flaws, resulting in a higher-quality final prototype.

**Evolutionary Prototyping:** The prototype is initially developed and then incrementally refined based on customer feedback until it is finally accepted. In comparison to Rapid Throwaway Prototyping, it is a more efficient method that saves both time and effort. This is because creating a new prototype for each iteration of the process can be extremely frustrating for developers.The prototype is initially developed and then incrementally refined based on customer feedback until it is finally accepted. In comparison to Rapid Throwaway Prototyping, it is a more efficient method that saves both time and effort. This is because creating a new prototype for each iteration of the process can be extremely frustrating for developers.

**Incremental Prototyping:** In this type of incremental prototyping, the final desired product is divided into smaller prototypes and developed separately. Finally, once all individual components have been properly developed, the various prototypes are collectively merged into a single final product in the order in which they were created. It is a very efficient approach that reduces the complexity of the development process by dividing the goal into sub-parts and developing each sub-part independently. The time required to complete a project is significantly reduced because all components of the system are prototyped and tested concurrently. Of course, there is a possibility that the pieces will not fit together properly due to a lack of planning during the development phase – this can only be resolved by meticulously plotting the entire system before prototyping begins.

**Extreme Prototyping:**This method is mainly used for web development. It is consists of three sequential independent phases:

**D.1)**In this phase a basic prototype with all the existing static pages are presented in the HTML format.

**D.2)**  In the 2nd phase, Functional screens are made with a simulate data process using a prototype services layer.

**D.3)** This is the final step where all the services are implemented and associated with the final prototype.

This method of Extreme Prototyping ensures a robust and rapid project cycle and delivery, and keeps the entire developer team focused on product delivery rather than on discovering all possible needs and specifications and adding superfluous features.

Diagram

Description automatically generated

Fig 7: **Extreme Prototyping**

* 1. **Advantages –**
* The customers get to see the partial product early in the life cycle. This ensures a greater level of customer satisfaction and comfort.
* New requirements can be easily accommodated as there is scope for refinement.
* Missing functionalities can be easily figured out.
* Errors can be detected much earlier thereby saving a lot of effort and cost, besides enhancing the quality of the software.
* The developed prototype can be reused by the developer for more complicated projects in the future.
* Flexibility in design.
  1. **Disadvantages**
* Costly w.r.t time as well as money.
* There may be too much variation in requirements each time the prototype is evaluated by the customer.
* Poor Documentation due to continuously changing customer requirements.
* It is very difficult for developers to accommodate all the changes demanded by the customer.
* There is uncertainty in determining the number of iterations that would be required before the prototype is finally accepted by the customer.
* After seeing an early prototype, the customers sometimes demand the actual product to be delivered soon.
* Developers in a hurry to build prototypes may end up with sub-optimal solutions.
* The customer might lose interest in the product if he/she is not satisfied with the initial prototype.
  1. **Use**

When the product's requirements are unclear or unstable, the Prototyping Model should be used. Additionally, it can be used if requirements change frequently. This model is suitable for developing user interfaces, high-technology software-intensive systems, and systems with sophisticated algorithms and interfaces. Additionally, it is an excellent choice for demonstrating the product's technical feasibility.