SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

Software development life cycle is a structure process that enables the production of high-quality, less costly software, in the shortest possible time that meets and exceeds all the customer’s expectations demands.

IMPORTANCE of SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

The software development life cycle is necessary as it is used to achieve the following:

* Delivers quality systems which meet or exceed the customer’s expectations when promised and within cost estimates.
* Provides a frame work for developing quality system using an indefinable, measure, and repeatable process.
* Establishes a project management structure to ensure that each system development project is effectively managers, throughout the system development life cycle.
* Identify and assign the roles and responsibilities of all including fictional and technical managers throughout the system development life cycle.
* Ensures that system development requirement are well defined and subsequently satisfied.

The stages of System Development Life cycle

1. Planning stage (requirement)
2. Analysis
3. Design
4. Development
5. Testing
6. Documentation
7. Implementation
8. Maintenance

**PLANNING Stage (Requirement)**

This is the first stage in of the cycle in this stage the planning phase encompasses all aspects of project and product management. This typically includes resource allocation, capacity planning, project scheduling, cost estimation, and provisioning.

During the planning phase, the development team collects input from stakeholders involved in the project; customers, sales, internal and external experts, and developers. This input is synthesized into a detailed definition of the requirements for creating the desired software. The team also determines what resources are required to satisfy the project requirements, and then infers the associated cost.

Expectations are clearly defined during this stage as well; the team determines not only what is desired in the software, but also what is NOT. The tangible deliverables produced from this phase include project plans, estimated costs, projected schedules, and procurement needs.

**Analysis**

In this stage the very phase the consumers requirements are analyzed .the project goals are converted into the defined functions that the organization intends to develop. The primary operations that will take place in this phase are;

1. Gathering software requirement
2. Creating process diagram
3. Performing a detailed analysis

Gathering software requirements-

This process is accomplished by identifying the relevant stakeholders (customers) desires, then establishing the projects goals and its desired objectives within the following

* User stories (imagination)
* Functional decompositions
* Features description

These are relevant so that the purpose of the user stories and available to all the members of the project team or individual software engr.

Then the requirements confirmation ensures all the stake holders customers and the production team come to an understanding of what you’re trying to build. This involves reviewing and refining the requirements. It will very like require additional elicitation and revision of the documentation as well

CREATING PROCESS DIAGRAM-

This is like creating a flowchart but actual it is graphical way of describing a process it constituents task and their sequences throughout the process. A PDF help with the brainstorming and commutates about the tasks of the process design. The level of the detail can decide be decide by the team including the time required time and it reduces the probability of missing failure modes or results.

Software Engineering should be able to produce the PFD. A good way to go about creating the PFD is to first log the major tasks of the process. Then add the detailed tasks, and the steps necessary to realize each task. Next, have a walk through the PFD with the stakeholders, e.g., fellow engineers and creative team, to debug the PFD. Perform the process using the PFD to verify the PFD. Add any discovered missing steps.

PERFORMING A DETAILED ANALYSIS –

This was probably done in all the stages getting down every single note down and putting them into consideration which was done throughout your analyzation if the team are a proper and well disciple.

**DESIGN STAGE**

This phase plays a crucial rule in the mobile app development industry. Here the software is designed to satisfy the identified requirements in the previous phases, in this phase one or more design are created to achieve the project result. Depending on the project subject, the design phase products include diorama’s, flow-charts, sketches site trees, HTML screen designs, photo impressions, prototypes, and UML schemas

The project supervisors use these designs to choose the definitive design that can be produced in the project. The development phase follows it. Once you have selected the design in the definition phase, you cannot make changes in the project’s later stage.

**DEVELOPMENT STAGE**

This phase features a key step in the project: software programing. The previous stages laid foundation for this stage (phase).the following stages (phases) ensure that the products functions as required

Objectives of this Phase or stage

* Creating a credible software program
* Testing and integrating the units into larger components
* Preparing the practical environment of the software
* Approval to the pro the testing phase or stage

**TESTING STAGE**

Once the software engineers build the software, then it is deployed in the testing environment. Then the testing team tests the functionality of the entire system. In this fifth phase of SDLC, the testing is done to ensure that the entire application works according to the customer requirements.

After testing, the team might find some bugs or defects and communicate the same with the developers. The development team then fixes the bugs and send it to QA for a re-test. This process goes on until the software is stable, bug-free and working according to the business requirements of that system.

**DOCUMENTATION**

Documentation in software engineering is the umbrella term that encompasses all written documents and materials dealing with software product development. All software development products, whether created by a small team or a large corporation, require some related documentation. And different types of documents are created through the whole[software development lifecycle](https://www.altexsoft.com/whitepapers/estimating-software-engineering-effort-project-and-product-development-approach/) (SDLC). Documentation exists to explain product functionality, unify project-related information, and allow for discussing all significant questions arising between stakeholders and developers.

These is very necessary to understand the level of work achieved at each stage.

**IMPLEMENTATION STAGE**

Implementation phase in SDLC is the process of configuring the software for certain conditions of use, as well as training customers to work with the product. This stage begins after the software has been tested and accepted by the team. At the time, a program is installed to support the intended business functions. The software performance is compared with the targets set at the planning phase.

It carries out a full range of works on the implementation of software into the software according to the customer’s wishes, starting from the installation, adaptation and adjustment of programs and up to integration with devices, commissioning and user training.

Types

The methods of software implementation can be different and depend on the task that the developers initially faced. According to the situation suggests using three ways:

* Fresh implementation. It can be defined as a process in which manual recording is replaced by new software. In this case there are some tasks in the form of file conversion, user training, etc.
* Replacement implementation. The old software is replaced by new programs. It demands thorough planning to overcome difficulties may arise.
* Modified implementation. The old software is replaced by new with some changes. This type of implementation is easily, because the file modification area is not so large.

Each method can bring real benefits to the business and improve the final result

Finally ……..

**MAINTANCE STAGE**

 The maintenance phase of the SDLC occurs after the product is in full operation. Maintenance of software can include software upgrades, repairs, and fixes of the software if it breaks.

Software applications often need to be upgraded or integrated with new systems the customer deploys. It's often necessary to provide additional testing of the software or version upgrades. During the maintenance phase, errors or defects may exist, which would require repairs during additional testing of the software. Monitoring the performance of the software is also included during the maintenance phase.

SOFTWARE DEVELOPMENT LIFE CYCLE MODELS

Software Development life cycle (SDLC) is a spiritual(MENTAL) model used in project management that defines the stages include in an information system development project, from an initial feasibility study to the maintenance of the completed application.

There are different software development life cycle models specify and design, which are followed during the software development phase. These models are also called "**Software Development Process Models**." Each process model follows a series of phase unique to its type to ensure success in the step of software development

The different types of SOFTWARE DEVELOPMENT LIFE CYCLE MODELS

(Sdlc) models:

1. Waterfall Models
2. V- shaped model
3. Protocol model
4. Spiral model
5. Iterative increment model
6. Big Bang model
7. Agile

WATER FALL MODEL

Waterfall is the oldest and most straightforward of the structured SDLC methodologies — finish one phase, then move on to the next. No going back.

Each stage relies on information from the previous stage and has its own project plan. Waterfall is easy to understand and simple to manage.

But early delays can throw off the entire project timeline.

And since there is little room for revisions once a stage is completed, problems can’t be fixed until you get to the maintenance stage.

This model doesn’t work well if flexibility is needed or if the project is long term and ongoing.

Advantages

1. Waterfall model is the simple model which can be easily understood and is the one in which all the phases are done step by step
2. Deliverables of each phase are well defined, and this leads to no complexity and makes the project easily manageable.

Disadvantages

1. Waterfall model is time-consuming & cannot be used in the short duration projects as in this model a new phase cannot be started until the ongoing phase is completed
2. Waterfall model cannot be used for the projects which have uncertain requirement or wherein the requirement keeps on changing as this model expects the requirement to be clear in the requirement gathering and analysis phase itself and any change in the later stages would lead to cost higher as the changes would be required in all the phases

V-shaped model

Also known as the Verification and Validation model, the V-shaped model grew out of Waterfall and is characterized by a corresponding testing phase for each development stage.

Like Waterfall, each stage begins only after the previous one has ended. But the difference is that as it each phase is tested immediately as it is finished making harder to go back and make changes.

Advantages

1. It is a simple and easily understandable model
2. V –model approach is good for smaller projects wherein the requirement is defined and it freezes in the early stage.
3. It is a systematic and disciplined model which results in a high-quality product

Disadvantages

1. V-shaped model is not good for ongoing projects
2. Requirement change at the later stage would cost too high.

Protocol model

The prototype model is a model in which the prototype is developed prior to the actual software.

Prototype models have limited functional capabilities and inefficient performance when compared to the actual software. Dummy functions are used to create prototypes. This is a valuable mechanism for understanding the customers’ needs

Software prototypes are built prior to the actual software to get valuable feedback from the customer. Feedbacks are implemented and the prototype is again reviewed by the customer for any change. This process goes on until the model is accepted by the customer

Advantages

1. Involvement of a customer from the initial stage reduces any confusion in the requirement or understanding of any functionality.

Disadvantages

1. Since the customer is involved in every phase, the customer can change the requirement of the end product which increases the complexity of the scope and may increase the delivery time of the product

SPIRAL MODEL

One of the most flexible SDLC methodologies, the Spiral model takes a cue from the Iterative model and its repetition; the project passes through four phases over and over in a “spiral” until completed, allowing for multiple rounds of refinement.

This model allows for the building of a highly customized product, and user feedback can be incorporated from early on in the project.

But the risk you run is creating a never-ending spiral for a project that goes on and on

Advantages

1. Risk Analysis is done extensively using the prototype models.
2. Any enhancement or change in the functionality can be done in the next iteration

Disadvantages

1. The spiral model is best suited for large projects only.
2. The cost can be high as it might take a large number of iterations which can lead to high time to reach the final product.

Iterative increment model

The Iterative model is repetition incarnate. Instead of starting with fully known requirements, you implement a set of software requirements, then test, evaluate and pinpoint further requirements.

A new version of the software is produced with each phase, or iteration. Rinse and repeat until the complete system is ready.

Advantages

1. Any change in the requirement can be easily done and would not cost as there is a scope of incorporating the new requirement in the next iteration.
2. Risk is analyzed & identified in the iterations.
3. Defects are detected at an early stage.
4. As the product is divided into smaller chunks it is easy to manage the product

Disadvantages

1. Complete requirement and understanding of a product are required to break down and build incrementally

Big bang model

A bit of an anomaly among SDLC methodologies, the Big Bang model follows no specific process, and very little time is spent on planning.

The majority of resources are thrown toward development, and even the client may not have a solid grasp of the requirements.

This is one of the SDLC models typically used for small projects with only one or two [software engineers](http://www.roberthalf.com.au/blog/software-engineer-vs-systems-engineer-whats-difference).

Big Bang is not recommended for large or complex projects, as it’s a high-risk model; if the requirements are misunderstood in the beginning, you could get to the end and realise the project may have to be started all over again.

Advantages

1. It’s a very simple Model.
2. Less Planning and scheduling is required.
3. The developer has the flexibility to build the software of their own

Disadvantages

1. Big Bang models cannot be used for large, ongoing & complex projects.
2. High risk and uncertainty

Agile model

By breaking the product into cycles, the Agile model quickly delivers a working product and is considered a very realistic development approach.

The model produces ongoing releases, each with small, incremental changes from the previous release. At each iteration, the product is tested.

This model emphasises interaction, as the customers, developers and testers work together throughout the project.

But since this model depends heavily on customer interaction, the project can head the wrong way if the customer is not clear on the direction he or she wants to go

Advantages

1. It allows more flexibility to adapt to the changes.
2. The new feature can be added easily.
3. Customer satisfaction as the feedback and suggestions are taken at every stage.

Disadvantages

1. Lack of documentation.
2. Agile needs experienced and highly skilled resources.
3. If a customer is not clear about how exactly they want the product to be, then the project would fail.

IMPORTANCE OF SOFTWARE DEVELOPMENT LIFE CYCLE

The software development life cycle adds value to software development in the following ways:

* It provides an effective framework and method to develop software applications.
* It helps in effectively planning before starting the actual development. SDLC allows developers to analyze the requirements.
* It helps in reducing unnecessary costs during development. During the initial phases, developers can estimate the costs and predict costly mistakes.
* It enables developers to design and build high-quality software products. This is because they follow a systematic process that allows them to test the software before it is rolled out.
* It provides a basis when evaluating the effectiveness of the software. This further enhances the software product