**Software Development Life Cycle**

**SDLC Overview**

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 a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

* SDLC is the acronym of Software Development Life Cycle.
* It is also called as Software Development Process.
* SDLC is a framework defining tasks performed at each step in the software development process.
* ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.

**History of the SDLC**

The profession of “software developer” has existed since the first computers, and their operators, as far back as the days of ENIAC and vacuum tubes. Practices and methods for developing software have evolved over the decades since the invention of the computer. Those methods have adapted to the state of the art in computer hardware, development tools, and modern thinking about the organizational management of software development teams. With this progress, new methods of software development have grown out of private and public software development efforts around the world. These methods vary widely in approach, yet they share a common goal: to develop software as cheaply, efficiently, and effectively as possible.

**What is the SDLC?**

Software is a complex product that is developed and delivered through a series of steps. That is the one thing all the various methods have in common: one way or another, software, like all products, starts as an idea. The idea then becomes a document, or perhaps a prototype, depending on the method in use. Whether a document, diagram, or working software, the artifact created in one step becomes the input to the next step. Eventually, the software is delivered to the customer. The sequence of steps used by these methods is commonly referred to as the Software Development Lifecycle (SDLC.)

**How does the SDLC work?**

The process of software development is a never-ending cycle. The first release of a software application is rarely “finished.” There are almost always additional features and bug fixes waiting to be designed, developed, and deployed.

Reports from error monitoring software about usability and bugs feed back into the process of software development and become new feature requests and improvements to existing features. This is why the Software Development Life Cycle is the most general term for software development methods. The steps of the process and their order vary by method. Regardless of method, they typically run in cycles, starting over with each iteration.

It’s very difficult to carry out a complex, team effort such as software development without some kind of plan. Each software development methodology (several will be detailed below) is a plan framework for how to develop software. There is much debate about which method is best overall, which is best suited to a particular type of software, and how to measure success in software development. One thing, however, is certain: any plan is better than no plan.

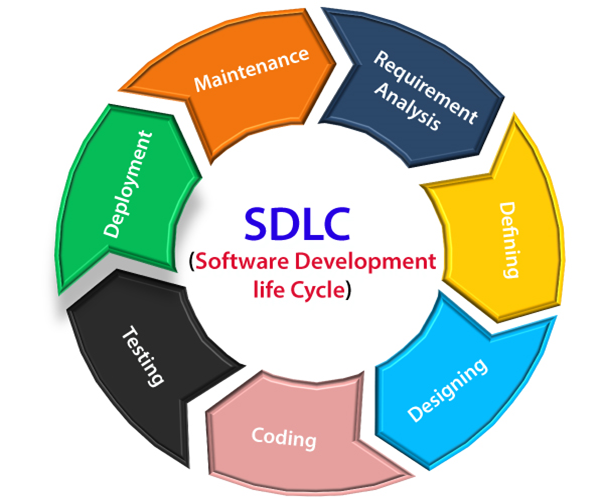
Without some kind of structured plan, software development teams tend to devolve into a “herd of cats.” Developers don’t know what they’re supposed to create. Project managers have no idea how much progress is made towards completion of a project. Without a plan, the business doesn’t even have a way to decide whether the final product meets their requirements.

**The Seven Phases of the SDLC**

These steps are (very) roughly the same from one methodology to another. They tend to occur in this order, though they can also be mixed together, such that several steps occur in parallel.

As we’ll discuss later, Agile methods tend to “wind together” all of these steps into a tight, rapidly-repeating cycle. Waterfall methods tend to take each of these steps in turn. Outputs from one become inputs to the following step.

**The following figure is a graphical representation of the various stages of a typical SDLC.**



1. **Requirements Analysis/Planning**

The planning phase involves aspects of project and product management. This may include:

Resource allocation (both human and materials)

* Capacity planning
* Project scheduling
* Cost estimation
* Provisioning

The outputs of the planning phase include: project plans, schedules, cost estimations, and procurement requirements. Ideally, Project Managers and Development staff collaborate with Operations and Security teams to ensure all perspectives are represented.

2. **Defining/Feasibility**

The business must communicate with IT teams to convey their requirements for new development and enhancement. The requirements phase gathers these requirements from business stakeholders and Subject Matter Experts (SMEs.)

**There are mainly five types of feasibility checks:**

* Economic: Can we complete the project within the budget or not?
* Legal: Can we handle this project as cyber law and other regulatory framework/compliance.
* Operation feasibility: Can we create operations which is expected by the client?
* Technical: Need to check whether the current computer system can support the software
* Schedule: Decide that the project can be completed within the given schedule or not.

Architects, Development teams, and Product Managers work with the SMEs to document the business processes that need to be automated through software. The output of this phase in a Waterfall project is usually a document that lists these requirements. Agile methods, by contrast, may produce a backlog of tasks to be performed.

3. **Design and prototyping**

Once the requirements are understood, software architects and developers can begin to design the software. The design process uses established patterns for application architecture and software development. Architects may use an architecture framework such as TOGAF to compose an application from existing components, promoting reuse and standardization.

Developers use proven Design Patterns to solve algorithmic problems in a consistent way. This phase may also include some rapid prototyping, also known as a spike, to compare solutions to find the best fit. The output of this phase includes:

* Design documents that list the patterns and components selected for the project
* Code produced by spikes, used as a starting point for development

There are two kinds of design documents developed in this phase:

**High-Level Design (HLD)**

* Brief description and name of each module
* An outline about the functionality of every module
* Interface relationship and dependencies between modules
* Database tables identified along with their key elements
* Complete architecture diagrams along with technology details

**Low-Level Design (LLD)**

* Functional logic of the modules
* Database tables, which include type and size
* Complete detail of the interface
* Addresses all types of dependency issues
* Listing of error messages
* Complete input and outputs for every module

4. **Coding/Software development**

This phase produces the software under development. Depending on the methodology, this phase may be conducted in time-boxed “sprints,” (Agile) or may proceed as a single block of effort (Waterfall.) Regardless of methodology, development teams should produce working software as quickly as possible. Business stakeholders should be engaged regularly, to ensure that their expectations are being met. The output of this phase is testable, functional software.

5. **Testing**

The testing phase of the SDLC is arguably one of the most important. It is impossible to deliver quality software without testing. There is a wide variety of testing necessary to measure quality:

* Code quality
* Unit testing (functional tests)
* Integration testing
* Performance testing
* Security testing

The best way to ensure that tests are run regularly, and never skipped for expediency, is to automate them. Tests can be automated using Continuous Integration tools, like Codeship, for example. The output of the testing phase is functional software, ready for deployment to a production environment.

6. **Deployment**

The deployment phase is, ideally, a highly automated phase. In high-maturity enterprises, this phase is almost invisible; software is deployed the instant it is ready. Enterprises with lower maturity, or in some highly regulated industries, the process involves some manual approvals. However, even in those cases it is best for the deployment itself to be fully automated in a continuous deployment model. Application Release Automation (ARA) tools are used in medium and large-size enterprises to automate the deployment of applications to Production environments. ARA systems are usually integrated with Continuous Integration tools. The output of this phase is the release to Production of working software.

7. **Operations and maintenance**

The operations and maintenance phase are the “end of the beginning,” so to speak. The Software Development Life Cycle doesn’t end here. Software must be monitored constantly to ensure proper operation. Bugs and defects discovered in Production must be reported and responded to, which often feeds work back into the process. Bug fixes may not flow through the entire cycle, however, at least an abbreviated process is necessary to ensure that the fix does not introduce other problems (known as a regression.)