THE SOFTWARE DEVELOPMENT LIFECYCLE (SDLC): AN INTRODUCTION

The Software Development Lifecycle (SDLC) describes the systematic approach to developing software. In this article, we'll look at:

What is the software development lifecycle?

The SDLC helps to ensure high quality software is built and released to end-users quickly and at an optimized cost. How you determine the quality of your software might vary, but general measurements include:

- The robustness of the software functionality
- Overall performance
- Security
- Ultimately, the user experience

Regardless of which software development you subscribe to—Agile, Waterfall, or other

variations—this lifecycle can apply.

Who uses the SDLC?

Not so long ago, Watt S. Humphrey, known as the father of quality in software, remarked:

"Every business is a software business."

More recently, Microsoft CEO Satya Nadella repeated the quote: "Every company is a software company".

Of course, we can point to many specific technology companies who develop software. If there's an app, someone developed it.

But business organizations that aren't "in software" rely on on software and technology to do business (which is to say, all of them). These organizations will need to adapt at least some off-the-shelf solutions, likely to tweak software to align and optimize with their unique business operations.

That's why people beyond developers or engineers should understand the SDLC approach: many stakeholders might be involved in various stages. Plus, cross-functional teams might adopt the SDLC to collaborate on Agile- and DevOps-based projects. Following modern SDLC practices and frameworks can significantly improve the software development process.

So, who uses the SDLC? In short, everyone.

Stages of the SDLC

The SDLC follows a series of phases involved in software development. Depending on the SDLC framework, these phases may be adopted sequentially or in parallel. (More on this below.)

The SDLC workflows may involve repeated transitions or iterations across the phases before reaching the final phase.

Phase 1: Requirement Analysis

The initial stage of the SDLC involves stakeholders from tech, business, and leadership segments of the organization. In this initial state, you'll:

- Analyze and translate business questions into engineering problems by considering a variety of factors: cost, performance, functionality, and risk.
- Evaluate he broad scope of the project and then identify available resources.
- Consider project opportunities and risks across the technical and business aspect for every decision choice in each SDLC phase.

This stage may continue for a prolonged period and includes provision for strategic changes as the SDLC evolves.

(Learn how to write software requirements specifications, also known as SRS.)

Phase 2: Feasibility Study

During this stage, evaluate the requirements for feasibility. Not every single requirement will be feasible for your current scope. The goal of this stage is to quantify the opportunities and risk of addressing the agreed requirements with the variety of resources and strategies you have available.

The feasibility study evaluates the following key aspects, among others:

- **Economic:** Is it financially viable to invest in the project based on the available resources?
- **Legal:** What is the scope of regulations and the organization's capacity to guarantee compliance?
- **Operational:** Can we satisfy the requirements within scope definition according to the proposed operational framework and workflows?
- **Technical:** What is the availability of technology and HR resources to support the SLDC process?
- **Schedule:** Can we finish the project in time?

Executive decision makers should answer and document these questions and study them carefully—before proceeding with the software design and implementation process.

Phase 3: Architectural Design

Next, the appropriate technical and business stakeholders document, review, and evaluate the design specifications and choices against the risk, opportunities, practical modalities, and constraints.

In this phase, you'll have technical documentation that specifies:

- Systems architecture
- Configurations
- Data structure
- Resource procurement model

Desired output can include prototypes, pseudocode, minimal viable products (MVPs) and/or architecture reports and diagrams that include the necessary technology details:

- High-level design details include the desired functionality of software and system modules.
- Low-level design details can include the functional logic, interface details, dependency issues, and errors.

Phase 4: Software Development

Implementation follows the design phase. Several independent teams and individuals collaborate on feature development and coding activities. Frequently, individual developers will build their own codebase within the development environment, then merge it with the collaborating teams in a common build environment.

While the requirements analysis and design choices are already defined, feedback from the development teams is reviewed for potential change in direction of the design strategies.

This is the longest process in the SDLC pipeline and it assists subsequent phases of software testing and deployment.

(Explore behavior-driven development, one approach to developing software.)

Phase 5: Testing

In this phase, you'll use testing to:

- Investigate the performance of the software
- Discover and identify potential issues to fix or address

Testing teams develop a test plan based on the predefined software requirements. The testing plan should:

- Identify the resources available for testing
- Provide instructions and assignments for testers
- Select types of tests to be conducted
- Determine what to report to technical executives and decision makers

Testers often work collectively with development teams and rework the codebase to improve test results.

It is very common for teams to repeat the development and testing phases several times, before moving onto the final stages of deploying and and releasing the software.

(Consider the benefits of testing automation and testing frameworks, like regression testing, BDD, and TDD.)

Phase 6: Deployment

You've reached the final phase of the SDLC pipeline when your finished product has passed the necessary tests. Now, make it available for release to end users in the real environment. Several procedures and preparation activities are involved before a software product can be shipped, including:

- Documentation
- Transferring ownership and licensing,
- Deploying and installing the product on customer systems

(Learn more about the deployment and release stages.)

Traditional vs modern SDLC methodologies

With traditional SDLC methodologies, such as Waterfall, these phases are performed independently in series by disparate teams. Under the Agile methodology, these phases are performed in short, iterative, incremental sprints.

bmc

Agile	Waterfall
Iterative development in short sprints	Sequential development process in pre-defined phases
Flexible and adaptive methodology	The process is documented and follows the fixed structure and requirements agreed in the beginning of the process
Feedback-based approach: Sprints lead to short build updates that are evaluated on and guide the future direction of the development process.	Limited and delayed feedback: The software quality and requirements fulfilment isn't evaluated until the final phase of the development processes when testers and customer feedback is requested.
A provision for adaptability: Project development requirements and scope is expected to change over the course of the iterative development process.	The requirements and scope are definitive once agreed upon.
The SDLC phases overlap and begin early in the SDLC: planning, requirements, designing, developing, testing, and maintenance.	The SDLC phases are followed in order, with no overlap. Members of one functional group are not involved in another phase that doesn't belong to their job responsibilities.
Follows a mindset of collaboration and communication. The requirements, challenges, progress, and changes are discussed between all stakeholders on a continuous basis.	Follows a project-focused mindset with the aim of fully completing the SDLC process.
Responsibilities and hierarchical structure can be interchangeable between team members.	Fixed individual responsibilities, particularly in management positions.
All team members focused on end-to-end completion (achieved sequentially) for the projects.	Team members focused on their responsibilities only during their respective SDLC phases.
Suitable for short projects in high-risk situations.	Suitable for straightforward projects in predictable circumstances.
Limited dependencies as the focus is less on implementation specifics, and more toward the mindset.	Strict dependencies in technologies, processes, projects and people.

(Agile vs Waterfall SDLC methodologies)

An SDLC pipeline and framework can be as varied as the number of organizations adopting them—virtually every company tries to adopt a strategy that works best for their organization.

In today's era of software development, however, these stages are not always followed sequentially. Modern SDLC frameworks such as DevOps and Agile encourage cross-functional organizations to share responsibilities across these phases conducted in parallel.

For instance, the DevOps SDLC framework encourages Devs, Ops, and QA personnel to work together for continuous development, testing and deployment activities. Additionally, the testing procedure is shifted left and early in the SDLC pipeline so that software defects are identified before it's too late to fix them.

Related reading

- BMC DevOps Blog
- How & Why To Become a Software Factory
- Differences Between Continuous Integration (CI), Delivery (CD), and Deployment
- Orchestration in SDLC for DevOps
- Agile Roles and Responsibilities
- 15 Best Practices for Building a Microservices Architecture