

What is SDLC (Software Development Life Cycle)?

Software Development Life Cycle is the cost-effective and time-efficient process development teams use to design and build software. The goal is to minimize project risks through forward planning so that software meets customer expectations during production and beyond.

Importance of SDLC

1. Increased visibility of the project development process to all stakeholders.
2. Efficient planning, estimation, and scheduling.
3. Improved risk management and cost estimation.
4. Systematic software delivery and better customer satisfaction.

How does SDLC work?

SDLC outlines several tasks required to build software to build a software application. The development process goes through several stages as developers add new features and fix bugs in the software.

The details of the process vary for different teams. However, we outlined some common SDLC phases below.

Plan

The planning phase typically includes cost-benefit analysis, scheduling, resource estimation, and allocation. The development team collects requirements from stakeholders such as customers, internal and external experts, and managers to create a software requirement specification document.

The document sets expectations and defines common goals that aid in project planning. The team estimates costs, creates a schedule, and has a detailed plan to achieve their goals.

Design

In the design phase, software engineers analyze requirements and identify the best solutions to create the software. For example, they may consider integrating pre-existing modules, make technology choices, and identify development tools. They will look at how to best integrate the new software into the organization's existing IT infrastructure.

Implement

In the implementation phase, the development team codes the product. They analyze the requirements to identify smaller coding tasks they can do daily to achieve the final result.

Test

The development team combines automation and manual testing to check the software for bugs. Quality analysis includes testing the software for errors and checking if it meets customer requirements. Because many teams immediately test their code, the testing phase often runs parallel to the development phase.

Deploy

When teams develop software, they code and test on a different copy of the software than the one that the users have access to. The software customers use is called production, while other copies are said to be in the build or testing environments.

Having separate build and production environments ensures that customers can continue to use the software even while it is being changed or upgraded. The deployment phase includes several tasks to move the latest build copy to the production environment, such as packaging, environment configuration, and installation.

Maintain

In the maintenance phase, among other tasks, the team fixes bugs, resolves customer issues, and manages software changes. In addition, the team monitors overall system performance, security, and user experience to identify new ways to improve the existing software.

What are SDLC models?

A software development lifecycle (SDLC) model conceptually presents SDLC in an organized fashion to help organizations implement it. Different models arrange the SDLC phases in varying chronological order to optimize the development cycle. We look at some popular SDLC models below.

Waterfall

The waterfall model arranges all the phases sequentially so that each new phase depends on the outcome of the previous phase. Conceptually, the design flows from one phase down to the next, like that of a waterfall.

Pros and cons

The waterfall model provides discipline to project management and gives a tangible output at the end of each phase. However, there is little room for change once a phase is considered complete, as changes can affect the software's delivery time, cost, and quality. Therefore, the model is most suitable for small software development projects, where tasks are easy to arrange and manage and requirements can be pre-defined accurately.

Iterative

The iterative process suggests that teams begin software development with a small subset of requirements. Then, they iteratively enhance versions over time until the complete software is ready for production. The team produces a new software version at the end of each iteration.

Pros and cons

It's easy to identify and manage risks, as requirements can change between iterations. However, repeated cycles could lead to scope change and underestimation of resources.

Spiral

The spiral model combines the iterative model's small repeated cycles with the waterfall model's linear sequential flow to prioritize risk analysis. You can use the spiral model to ensure the software's gradual release and improvement by building prototypes at each phase.

Pros and cons

The spiral model is suitable for large and complex projects that require frequent changes. However, it can be expensive for smaller projects with a limited scope.

Agile

The agile model arranges the SDLC phases into several development cycles. The team iterates through the phases rapidly, delivering only small, incremental software changes in each cycle. They continuously evaluate requirements, plans, and results so that they can respond quickly to change. The agile model is both iterative and incremental, making it more efficient than other process models.

Pros and cons

Rapid development cycles help teams identify and address issues in complex projects early on and before they become significant problems. They can also engage customers and stakeholders to obtain feedback throughout the project lifecycle. However, overreliance on customer feedback could lead to excessive scope changes or end the project midway.

How does SDLC address security?

In traditional software development, security testing was a separate process from the software development lifecycle (SDLC). The security team discovered security flaws only after they had built the software. This led to a high number of bugs that remained hidden as well as increased security risks.

Today, most teams recognize that security is an integral part of the software development lifecycle. You can address security in SDLC following DevSecOps practices and conducting security assessments during the entire SDLC process.

DevSecOps

DevSecOps is the practice of integrating security testing at every stage of the software development process. It includes tools and processes that encourage collaboration between developers, security specialists, and operation teams to build software that can withstand modern threats. In addition, it ensures that security assurance activities such as code review, architecture analysis, and penetration testing are integral to development efforts.

How does SDLC compare with other lifecycle management methodologies?

The term software development lifecycle (SDLC) is frequently used in technology to refer to the entire process of technology innovation and support. We give other similar terms below.

Systems development lifecycle

The abbreviation SDLC can sometimes refer to the systems development lifecycle, the process of planning and creating an IT system. The system typically consists of several hardware and software components that work together to perform complex functions.

Software development lifecycle compared to systems development lifecycle

The software development lifecycle addresses only the development and testing of software components. On the other hand, system development is a broader superset involving the setup and management of the software, hardware, people, and processes that can make up a system. It can include tasks like organizational training and change management policies that don't fall under the software development umbrella.

Application lifecycle management

Application lifecycle management (ALM) is the creation and maintenance of software applications until they are no longer required. It involves multiple processes, tools, and people working together to manage every lifecycle aspect, such as ideation, design and development, testing, production, support, and eventual redundancy.

SDLC compared to ALM

SDLC describes the application development phase in greater detail. It is a part of ALM. ALM includes the entire lifecycle of the application and continues beyond SDLC. ALM can have multiple SDLCs during an application's lifecycle.