

## What is SDLC?

The software development lifecycle (SDLC) is the cost-effective and time-efficient process that development teams use to design and build high-quality software. The goal of SDLC is to minimize project risks through forward planning so that software meets customer expectations during production and beyond. This methodology outlines a series of steps that divide the software development process into tasks you can assign, complete, and measure.

## Why is SDLC important?

Software development can be challenging to manage due to changing requirements, technology upgrades, and cross-functional collaboration. The software development lifecycle (SDLC) methodology provides a systematic management framework with specific deliverables at every stage of the software development process. As a result, all stakeholders agree on software development goals and requirements upfront and also have a plan to achieve those goals.

Here are some benefits of SDLC:

* Increased visibility of the development process for all stakeholders involved
* Efficient estimation, planning, and scheduling
* Improved risk management and cost estimation
* Systematic software delivery and better customer satisfaction

## 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 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.