SDLC

SDLC (System Development Life Cycle) is a process that provides a well-structured flow of phases, and helps an organization quickly produce high-quality, production-ready software.

The main SDLC models are waterfall and agile. Technical writers have a specific role in each model.

### Stages of the System Development Life Cycle

There are six primary stages of the modern system development life cycle, though some models also include maintenance as a seventh stage.

**1. Planning**

*“What problem are we fixing?”*  
Project leaders evaluate the terms of the project; calculate labor and material costs, create a timetable with target goals, and define the project’s teams and leadership structure.

Planning also includes feedback from stakeholders, such as potential customers, developers, subject-matter experts, and sales reps.

Planning clearly defines the scope and purpose of the project and sets boundaries to keep the project from expanding or shifting from its original purpose.

**2. Defining Requirements**

*“What do we want?”*  
Analyzes the system’s functional requirements to ensure they meet the expectations of the target audience. Expectations are finalized and documented.   
  
A feasibility study ensures that a system makes sense from financial, technological, and organizational standpoints.

**3. Designing**

*“How will we get what we want?”*

Describes the design functions and operations, including screen layouts, business rules, process diagrams, and other documentation. These design elements are intended to describe the system in sufficient detail, such that developers and engineers can develop and deliver the system with minimal additional design input.

**4. Building or Development**

*“Let’s create what we want”*

Marks the end of the preliminary part of the process and signifies the beginning of production. Software engineers write code and fine-tune the technologies involved in the project.

**5. Testing And Integration**

*“Did we get what we wanted?”*

Tests the system and all related procedures to assess if it performs as expected and minimizes the possibility that users encounter bugs and glitches.

**6. Deployment and Implementation**

*“Let’s start using what we got”*

Releases to production environment. After implementation, the system is available to end-users.

**7. Maintenance**

*“Let’s get this closer to perfect”*

Software enters the stage of continuous maintenance, via bug checks and updates to adapt to the changing needs of its end users. This can spawn new development cycles and releases.

### Waterfall Model

The waterfall model is linear and straightforward, and requires development teams to finish one phase of the project completely before moving on to the next.

Each stage has a separate project plan and takes information from the previous stage.

**Pros:**

* Simple to understand and follow
* Clear milestones and deadlines
* Clear documentation

**Cons**:

* Prone to early delays
* Inflexible to scope changes
* Working product emerges only on later stages of the cycle

#### **Technical writers in the waterfall team**

The waterfall approach produces a lot of internal content before you get near anything customer-facing: high-level plans, detailed plans, technical specs, architecture graphics, and so on. For technical writers whose main focus is not the internal project documentation but rather the customer-facing elements such as help documents, installation guides, and security guides, there can be a significant lead time between project initiation and availability of raw material.

### 

### Agile Model

Agile is based on the idea that the development of software components should not be meticulously planned from start to finish. Instead, the developers rely on a set of repeatable and manageable *sprints* that culminate in a series of tests. The team then evaluates the results and supplements with feedback from the customer. Only then is the next sprint started.

The aim in agile is to be able to adjust the arc of a project when the requirements change, or the desired solution is no longer achievable in the form originally planned.

**Pros:**

* Fast development and testing with continuous feedback
* Quick detection and resolution of issues
* Accommodating of changes or enhancements

**Cons:**

* Reliant on prior experience in agile software development
* Unclear perspective on what the final product will look like
* Less emphasis on design and documentation

#### **Technical writers in the agile team**

Agile’s ideal is that users should be able to use the software without any documentation. In practice, however, users need documentation to fully use software above a certain level of complexity.

Therefore, technical writers should be tightly integrated into the agile teams. Frequently, writers do not focus on a single component, but rather a whole range of applications. Consequently, they are members on a number of teams at the same time, which can cause conflicts with deadlines and resources.

### 

### Waterfall Model vs. Agile

| **Waterfall** | **Agile** |
| --- | --- |
| Divides SDLC into distinct phases | Separates SDLC into sprints |
| Sequential design process | Incremental approach |
| All phases are completed once | Iterative development approach |
| The testing phase comes after the building phases | Testing is performed concurrently with software development |
| Changes are not expected | Requirements are expected to change and evolve |
| The project manager plays an essential role in every stage | Agile team members are interchangeable and projects are managed by the entire team |

# 

# Sources

<https://www.clouddefense.ai/blog/system-development-life-cycle>

<https://www.intellectsoft.net/blog/what-is-system-development-life-cycle/>

<https://www.guru99.com/waterfall-vs-agile.html>

<https://project-management.com/agile-vs-waterfall/>

<https://www.forecast.app/blog/agile-vs-waterfall>

<https://phoenixnap.com/blog/software-development-life-cycle>

<https://www.quanos-content-solutions.com/blog/en/2020/12/03/agile-documentation-how-development-and-technical-writing-complement-each-other/>

<https://technicallywriteit.com/wp-content/uploads/2016/08/Waterfall_stepxstep.png>

<https://technicallywriteit.com/waterfall-vs-agile/>

<https://en.wikipedia.org/wiki/Systems_development_life_cycle>