The Software Development Life Cycle (SDLC): 7 Phases and 5 Models

QUICK SUMMARY

Building software is a huge job, which is why digital product teams rely on the software development life cycle (SDLC). Here's what you need to know.

Building software is a huge job, which is why digital product teams rely on the software development life cycle (SDLC). The SDLC usually takes the form of one of 5 different methodologies and follows 7 main development stages. Knowing what needs to be done in the SDLC process can help product managers guide the entire project to completion. It also helps PMs understand the milestones and communicate progress to stakeholders. Let’ s jump in!

What is the SDLC?

The software development life cycle is a process that development teams use to create awesome software that's top-notch in terms of quality, cost-effectiveness, and time efficiency. The main goal is to minimize risks and make sure the software meets the customer's expectations both during and after production.

This process is about creating a detailed plan to guide the development of the product and then breaking down the development process into smaller modules that can be assigned, completed, and measured to make the whole thing more manageable.

Benefits of SDLC for the Product Team

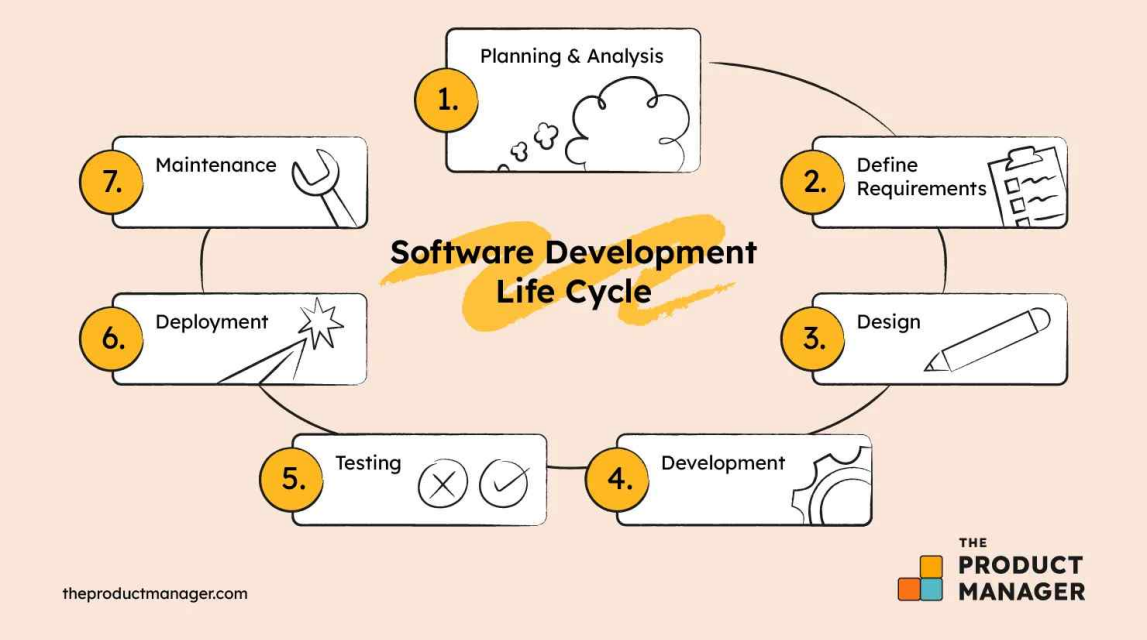
Managing changing requirements, staying on top of new technology, and working collaboratively can be challenging for the product team. That's where the SDLC comes in handy. The SDLC provides a framework for the product team to manage the development process systematically, with clear goals and deliverables at every stage. By using SDLC, the product team can ensure that all stakeholders agree on software development goals and requirements upfront and have a plan to achieve them.

Here are some specific benefits of using SDLC for the product team:

*Increased visibility of the development process for all stakeholders involved
*More efficient estimation, planning, and scheduling
*Improved risk management and cost estimation
*A systematic approach to delivering software that meets customer expectations and improves satisfaction

The 7 Phases of the Software Development Life Cycle

The SDLC process will look a little different for every team and product. However, these are the stages that most SDLC frameworks have in common:



The software development life cycle can be an iterative process.

1. Planning & Analysis

The first phase of the SDLC is the project planning stage where you are gathering business requirements from your client or stakeholders. This phase is when you evaluate the feasibility of creating the product, revenue potential, the cost of production, the needs of the end-users, etc.

To properly decide what to make, what not to make, and what to make first, you can use a feature prioritization framework that takes into account the value of the software/update, the cost, the time it takes to build, and other factors.

Once it is decided that the software project is in line with business and stakeholder goals, feasible to create, and addresses user needs, then you can move on to the next phase.

2. Define Requirements

This phase is critical for converting the information gathered during the planning and analysis phase into clear requirements for the development team. This process guides the development of several important documents: a software requirement specification (SRS), a Use Case document, and a Requirement Traceability Matrix document.

3. Design

The design phase is where you put pen to paper—so to speak. The original plan and vision are elaborated into a software design document (SDD) that includes the system design, programming language, templates, platform to use, and application security measures. This is also where you can flowchart how the software responds to user actions.

In most cases, the design phase will include the development of a prototype model. Creating a pre-production version of the product can give the team the opportunity to visualize what the product will look like and make changes without having to go through the hassle of rewriting code.

4. Development

The actual development phase is where the development team members divide the project into software modules and turn the software requirement into code that makes the product.

This SDLC phase can take quite a lot of time. It’ s important to have a set timeline and milestones so the software developers understand the expectations and you can keep track of the progress in this stage.

In some cases, the development stage can also merge with the testing stage where certain tests are run to ensure there are no critical bugs.

5. Testing

Before getting the software product out the door to the production environment, it’ s important to have your quality assurance team perform validation testing to make sure it is functioning properly and does what it’ s meant to do. The testing process can also help hash out any major user experience issues and security issues.

In some cases, software testing can be done in a simulated environment. Other simpler tests can also be automated.

The types of testing to do in this phase:

*Performance testing: Assesses the software's speed and scalability under different conditions
*Functional testing: Verifies that the software meets the requirements
*Security testing: Identifies potential vulnerabilities and weaknesses
*Unit-testing : Tests individual units or components of the software
*Usability testing: Evaluates the software's user interface and overall user experience
*Acceptance testing: Also termed end-user testing, beta testing, application testing, or field testing, this is the final testing stage to test if the software product delivers on what it promises

6. Deployment

During the deployment phase, your final product is delivered to your intended user. You can automate this process and schedule your deployment depending on the type. For example, if you are only deploying a feature update, you can do so with a small number of users (canary release). If you are creating brand-new software, you can learn more about the different stages of the software release life cycle (SRLC).

7. Maintenance

The maintenance phase is the final stage of the SDLC if you’ re following the waterfall structure of the software development process. However, the industry is moving towards a more agile software development approach where maintenance is only a stage for further improvement.

In the maintenance stage, users may find bugs and errors that were missed in the earlier testing phase. These bugs need to be fixed for better user experience and retention. In some cases, these can lead to going back to the first step of the software development life cycle.

The SDLC phases can also restart for any new features you may want to add in your next release/update.

Common SDLC Models

In software development, there are various frameworks, or “models,” of the Software Development Lifecycle (SDLC), which arrange the development process in different ways. These models help organizations implement SDLC in an organized way. Here are some of the most commonly used software life cycle models.

1. Agile Model

This model arranges the SDLC phases into several development cycles, with the team delivering small, incremental software changes in each cycle. The Agile methodology is highly efficient, and rapid development cycles help teams identify issues early on, but overreliance on customer feedback could lead to excessive scope changes or project termination. It's best for software development projects that require flexibility and the ability to adapt to change over time.

2. Waterfall Model

This model arranges all the phases sequentially, with each new phase depending on the outcome of the previous one. It provides structure to project management, but there is little room for changes once a phase is complete, so it's best for small, well-defined projects.

3. Iterative Model

With this model, the team begins development with a small set of requirements and iteratively enhances versions until the software is ready for production. It's easy to manage risks, but repeated cycles could lead to scope change and underestimation of resources. This model is best for projects that require high flexibility in their requirements and have the resources to handle multiple iterations.

4. Spiral Model

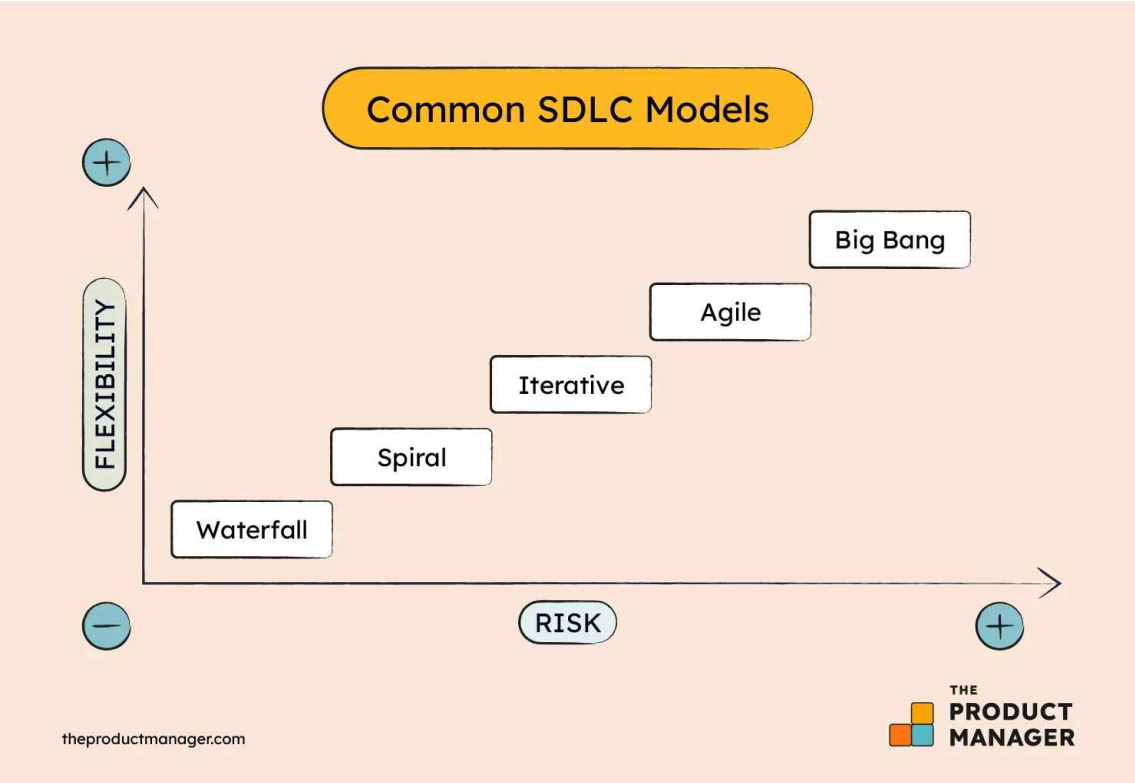
This model combines the iterative model's repeated cycles with the waterfall model's linear flow to prioritize risk analysis. It's best for complex projects with frequent changes but can be expensive for smaller projects.

5. Big Bang Model

The Big Bang Model is a unique approach where developers jump right into coding without much planning. This means that requirements are implemented as they come, without any kind of clear roadmap. If

changes are needed, it can require a complete revamp of the software.

While this model isn't great for larger projects, it’ s best for academic or practice projects, or smaller projects with only one or two developers. Essentially, it's a model that works well when requirements aren't well understood and there's no set release date in sight.



SDLC vs Other Lifecycle Management Methodologies

As you may know, SDLC is not the only lifecycle management process in the glossary of product management terms. Here are some similar terms and what distinguishes them from the SDLC:

SDLC vs. ALM (Application lifecycle management)

ALM is a term that describes the creation and maintenance of software applications, from ideation to design, development, testing, production, support, and eventual retirement. Sound a lot like SDLC? They might appear similar on paper, but some key differences include:

*SDLC focuses on the development phase of an application, while ALM takes a more comprehensive approach, covering the entire lifecycle of the application.
*Multiple ALM tools, processes, and teams need to work together to manage different stages of the application, including development.
*There may be multiple SDLCs within an application's lifecycle that fall under the larger ALM framework.

SDLC vs. systems development lifecycle

Sometimes, people use the term SDLC to refer to the systems development lifecycle, which is the process of planning and creating an IT system. This system typically consists of multiple hardware and software components that work together to perform complex functions.

So, what’ s the difference?

*SDLC only covers the development and testing of software components
*Systems development is a broader process that encompasses the setup and management of hardware, software, people, and processes needed for a complete system.
*While SDLC focuses on the software product only, systems development can include tasks like organizational training and change management that aren't necessarily part of software development.

SDLC vs STLC (Software Testing Lifecycle)

You might have also heard about the software testing lifecycle (STLC). The STLC refers to the set of activities that ensure software

quality by detecting bugs and defects before the product release. It has phases similar to the SDLC but with different objectives and deliverables.

There are several key differences between SDLC and STLC, such as:

*SDLC is focused on software development, while STLC is focused on software testing.
*SDLC aims to build a software product that meets the user requirements, while STLC aims to ensure that the software is bug-free and reliable.
*SDLC consists of various phases, such as planning, design, coding, testing, and deployment, while STLC has different phases, such as test planning, test case development, test execution, and test closure.

MORE ARTICLES

* How to Create More Intuitive Experiences: The Smartest UX Design Research Process
* 7 Growth Hacking Examples To Steal For Your Strategy
* 7 Deadly Sins Of Product Design (And How To Avoid Them)

SDLC vs DevOps

Another buzzword in the software development industry is DevOps. DevOps is a set of practices that combines software development (Dev) and IT operations (Ops) to enable faster and more frequent software delivery. It involves collaboration, automation, and monitoring throughout the software development lifecycle.

Here are the distinctions between SDLC and DevOps:

*SDLC is a methodology for managing software development, while DevOps is a cultural shift that promotes collaboration between development and operations teams.
*SDLC focuses on delivering software that meets the user requirements, while DevOps focuses on delivering software that meets the business objectives.
*SDLC involves different phases, such as planning, design, coding, testing, and deployment, while DevOps involves continuous integration, continuous delivery, and continuous monitoring.

SDLC vs PDLC (Product development lifecycle)

The product development lifecycle (PDLC) is a comprehensive process that covers the entire lifecycle of a product, from ideation to

retirement. It includes product planning, market research, product design, development, testing, launch, marketing, and support.

Here are some key differences between SDLC and PDLC:

*SDLC is focused on software development, while PDLC is focused on product development.
*SDLC consists of various phases, such as planning, design, coding, testing, and deployment, while PDLC includes additional phases, such as market research, product planning, and marketing.
*SDLC aims to build software that meets the user requirements, while PDLC aims to build a product that meets the market needs and generates revenue.

SDLC vs SRLC (Software Release Life Cycle)

The software requirements lifecycle (SRLC) is a process that focuses on gathering, documenting, and validating software requirements. It includes eliciting requirements from stakeholders, analyzing and prioritizing them, documenting them in a requirements specification, and validating them.

Here are some key differences between SDLC and SRLC:

*SDLC is focused on software development, while SRLC is focused on software requirements management.
*SDLC consists of various phases, such as planning, design, coding, testing, and deployment, while SRLC includes additional phases, such as requirements elicitation, analysis, and validation.
*SDLC aims to build software that meets the user requirements, while SRLC aims to ensure that the software requirements are complete, correct, and unambiguous before development starts.