**The Software Development Life Cycle (SDLC) is the software development world’s spellcheck — it can flag errors in software creation before they’re discovered (at a much higher cost) in successive stages. But it’s much more than that, of course: SDLC can also lay out a plan for getting everything right the first time.**

The SDLC process involves several distinct stages, including planning, analysis, design, building, testing, deployment and maintenance. Here are six SDLC methodologies, or models, that development teams use in this effort.

**Here are the software development lifecycle methodologies**

* Agile
* Lean
* Waterfall
* Iterative
* Spiral
* DevOps

Each of these approaches varies in some ways from the others, but all have a common purpose: to help teams deliver high-quality software as quickly and cost-effectively as possible.

Reviewing a brief description of the six most common SDLC methodologies may help you decide which is best for your team:

**1. Agile**

The Agile model has been around for about a decade. But lately, it has become a [major driving force behind software development](https://sdtimes.com/agile-is-becoming-the-de-facto-standard-for-software-development-according-to-versionone/) in many organizations. Some businesses value the Agile methodology so much that they are now applying it to other types of projects, including non-tech initiatives.

In the Agile model, “fast failure” is a good thing. The approach produces ongoing release cycles, each featuring small, incremental changes from the previous release. At each iteration, the product is tested. The Agile model helps teams identify and address small issues on projects before they evolve into more significant problems, and engage business stakeholders and get their feedback throughout the development process.

As part of their embrace of this methodology, many teams are also applying an Agile framework known as Scrum to help structure more complex development projects. Scrum teams work in “sprints,” which usually last two to four weeks, to complete assigned tasks. Daily Scrum meetings help the whole team monitor progress throughout the project. And the ScrumMaster is tasked with keeping the team focused on its goal. (For more on Scrum, see the [Scrum Alliance website](https://www.scrumalliance.org/).)

**2. Lean**

The Lean model for software development is inspired by lean manufacturing practices and principles. The seven Lean principles (in this order) are: eliminate waste, amplify learning, decide as late possible, deliver as fast as possible, empower the team, build integrity in, and see the whole.

The Lean process is about working only on what must be worked on at the time, so there’s no room for multitasking. Project teams are also focused on finding opportunities to cut waste at every turn throughout the SDLC process, from dropping unnecessary meetings to reducing documentation.

The Agile model is actually a Lean method for the SDLC, but with some notable differences. One is how each prioritizes customer satisfaction: Agile makes it the top priority from the outset, creating a flexible process where project teams can respond quickly to stakeholder feedback throughout the SDLC. Lean, meanwhile, emphasizes the elimination of waste as a way to create more overall value for customers — which, in turn, helps to enhance satisfaction.

***Ready to fill an open role?***[***Use our candidate finder.***](https://www.roberthalf.com/browse-candidates)

**3. Waterfall**

Some experts argue that the Waterfall model was never meant to be a process model for real projects (check out the discussion on this topic on [StackExchange](https://stackexchange.com/)). Regardless, the Waterfall model is widely considered the oldest of the structured SDLC methodologies. It’s also a very straightforward approach: finish one phase, then move on to the next. No going back. Each stage relies on information from the previous stage and has its own project plan.

The downside of Waterfall is its rigidity. Sure, it’s easy to understand and simple to manage. But early delays can throw off the entire project timeline. With little room for revisions once a stage is completed, problems can’t be fixed until you get to the maintenance stage. This model doesn’t work well if flexibility is needed or if the project is long term and ongoing.

Even more rigid is the related Verification and Validation model — or V-shaped model. This linear development methodology sprang from the Waterfall approach. It’s characterized by a corresponding testing phase for each development stage. Like Waterfall, each stage begins only after the previous one has ended. This SDLC model can be useful, provided your project has no unknown requirements.

**4. Iterative**

The Iterative model is repetition incarnate. Instead of starting with fully known requirements, project teams implement a set of software requirements, then test, evaluate and pinpoint further requirements. A new version of the software is produced with each phase, or iteration. Rinse and repeat until the complete system is ready.

Advantages of the Iterative model over other common SDLC methodologies is that it produces a working version of the project early in the process, and makes it less expensive to implement changes. One disadvantage: Repetitive processes can consume resources quickly.

One example of an Iterative model is the Rational Unified Process (RUP), developed by IBM’s Rational Software division. As explained in this [document from IBM](https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf), RUP is a “process product” designed to enhance team productivity that also “captures many of the best practices in modern software development in a form that is suitable for a wide range of projects and organizations.”

RUP divides the development process into four phases: inception, when the idea for a project is set; elaboration, when the project is further defined, and resources are evaluated; construction, when the project is developed and completed; and transition, when the product is released. Each phase of the project involves business modeling, analysis and design, implementation, testing, and deployment.

**5. Spiral**

One of the most flexible SDLC methodologies, the Spiral model takes a cue from the Iterative model and its repetition; the project passes through four phases (planning, risk analysis, engineering and evaluation) over and over in a “spiral” until completed, allowing for multiple rounds of refinement.

The Spiral model is typically used for large projects. It enables development teams to build a highly customized product, and incorporate user feedback early on in the project. Another benefit of this SDLC model is risk management. Each iteration starts by looking ahead to potential risks, and figuring out how best to avoid or mitigate them.

**6. DevOps**

The DevOps methodology is the newcomer to the SDLC scene. As [this article explains](https://theagileadmin.com/what-is-devops/), it emerged from two trends: the application of Agile and Lean practices to operations work, and the general shift in business toward seeing the value of collaboration between development and operations staff at all stages of the SDLC process.

In a DevOps model, Developers and Operations teams work together closely — and sometimes as one team — to accelerate innovation and the deployment of higher-quality and more reliable software products and functionalities. Updates to products are small but frequent. Discipline, continuous feedback and process improvement, and automation of manual development processes are all hallmarks of the DevOps model.

Amazon Web Services [describes DevOps](https://aws.amazon.com/devops/what-is-devops/) like this: “DevOps is the combination of cultural philosophies, practices, and tools that increases an organization’s ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes.” So, like many SDLC models, DevOps is not only an approach to planning and executing work, but also a philosophy that demands significant mindset and culture changes in an organization.

Choosing the right SDLC model for your software development project will require careful thought. But keep in mind that a methodology for planning and guiding your project is only one ingredient for success. Even more important is assembling a solid team of skilled talent committed to moving the project forward through every unexpected challenge or setback.