The Software Development Life Cycle

Amanda Mason

Ivy Tech Community College

SDEV120: Computing Logic

Hieu Tran

November 9th, 2025

The Software Development Life Cycle

Think of the Software Development Life Cycle (SDLC) as the essential roadmap we use to turn a fuzzy idea into a solid, working application. It all starts with Phase 1: *Planning*, where we define the project's overall scope, budget, and timeline, followed by Phase 2: *Feasibility Analysis* to ensure the entire concept is technically possible and financially viable. Once approved, we move to Phase 3: *System Design*, which is like drawing the architectural blueprint, mapping out how the system components will interact before a single line of code is written. Then comes Phase 4: *Implementation* - the actual coding - followed immediately by Phase 5: *Testing*, where we implement practices such as automated testing, unit testing, integration testing, and system testing to identify and fix bugs before moving on to the next phase, Phase 6: *Deployment*. In this phase, the software is made available on-premises or in the cloud so users can access it, and from there on out, the team handles Phase 7: *Maintenance*—because software is never truly "finished," it just keeps evolving. (Atlassian, 2025)

Basic Software Development Life Cycle Models

These days, when starting an SDLC journey, teams usually choose between two main models: *Waterfall* and *Agile*. *Waterfall* is the traditional, linear approach—all planning is completed, then all the design, then all the building, and so on, moving strictly in sequence. It's great when you have a perfectly clear, unchanging final destination, but making detours or major adjustments later can feel like ripping up the map and starting over. In contrast, *Agile* is all about flexibility and frequent check-ins. Software is developed in small chunks (iterations), constantly collaborating and getting user feedback so adjustments can be made to the plan as requirements inevitably shift. If goals might change during the trip, *Agile* is the far safer, more adaptable choice. (Google, 2025)

Modularization and Program Design

Once the team knows how they plan to move forward with building the software, program design and modularization are critical for what will be built. Modularization is simply breaking the complex application down into smaller, self-contained units, or "modules." Think of it like building with LEGO bricks; each brick does one specific job, making the overall structure easier to understand, test independently, and replace without breaking the whole toy. Good program design ensures these individual modules are organized cleanly and work together seamlessly, following best practices that keep them separate and easy to manage. By focusing on this structured approach early on, software is created that is much more flexible, easier to maintain, and far less likely to suffer from tangled, hard-to-fix bugs. (Google, 2025)

References

Atlassian. (2025). *The Complete Guide to SDLC (software development life cycle)*. https://www.atlassian.com/agile/software-development/sdlc

Google. (n.d.). *Common SDLC Models*. Google search. https://www.google.com/search?q=sdlc%2Bmodels