Artifact 1: CS 250: Software Development Lifecycle

The following information is based around my current education knowledge regarding the Software Development Lifecycle. There are various models that are utilized within the SDLC process. The following are the ones I am most familiar with: Waterfall, Agile, Iterative and Incremental. Most of the models contain the following core stages of SDLC:

1. Planning – In this initial phase, the team defines the scope and purpose of the software. The thorough planning within this phase lays the foundation for the development teams to stay on track throughout the project.
2. Designing – In this phase, the team creates a software design plan outlining the architecture, interfaces, and components. Within this stage it is determined how the software will function, appear, and interact. This process transforms requirements either technical or business into a detailed software blueprint.
3. Building – After having a solid design plan, the developers can start building the software and the coding and configuration takes place during this phase. The project team takes the software design and brings it to life within this phase that is critical.
4. Testing – Before releasing the software, the team verifies and validates the project works as expected. Rigorous testing identifies bugs and code issues. The testing done during this phase ensures the software functions correctly and securely on a scale.
5. Deploying – Once all the software tests are passed, we move into this phase. The project is turned into an application to be installed and configured into production. A smooth deployment introduces the solution to users sooner with minimal disruption.
6. Maintaining – However, after deployment there is a phase that requires the development team to maintain the application and enhancing the software. Updates and patches fix issues and add new features. Proper maintenance of the software ensures a long-term software success story after the initial release.

The delivery of quality software requires careful coordination between many roles. The key roles and responsibilities within SDLC are the following:

1. Project Managers – Oversee the entire project lifecycle along with planning tasks, setting timelines, and managing resources of all kinds.
2. Developers – Write, debug, and review the source code.
3. QA Testers – Verify the software works by identifying bugs and defects.
4. Product Managers – Drive business objectives and prioritizes features.

SDLC can be a great framework to follow but teams should choose a specific methodology for their project. The following models are most used:

1. Waterfall – Sequential linear stages
2. Agile – Iterative cycles and incremental delivery
3. DevOps – Integration of development and operations

However, something to note is that requirements, team culture, and project size more often than note influences which methodology becomes ideal. The most common team implementation is known as agile frameworks like Scrum that are used specifically to accelerate delivery. The main thing to mention is that it doesn’t really matter which model is utilized, but utilizing the SDLC is a requirement and following the process leads to a higher-quality solution built securely with the budget and schedule in mind.

A great roadmap for the development teams to build, test, and maintain the software applications comes from comprehensive documentation. Robust documentation at all stages of the SDLC improves efficiency and quality. The following are the types of documentation that are commonly created:

1. Requirements Documents – These outline business and user needs that the software MUST meet. These are created during the planning and influence all downstream processes.
2. Technical Specifications – The technical implementation details are made easily understood by designers and developers. The specifications are updated because they evolve throughout the lifecycle as the software begins to take shape.
3. Test Plans – The testing scope, scenarios, cases, and expected results are all defined. This type of documentation is great for contributing to software stability and security.

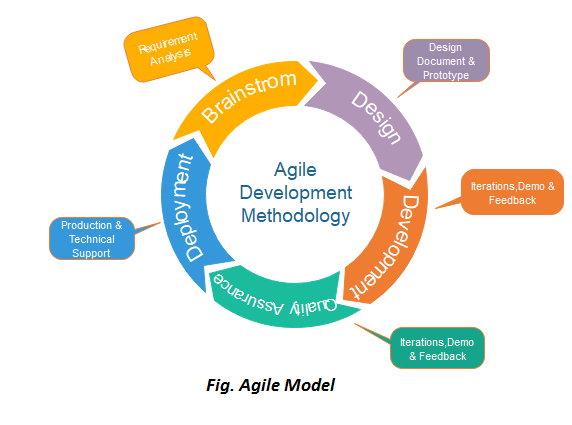
These are great documents that can enable stakeholders to make informed decisions during each stage of the SDLC. Continuity is created when team members change, and tribal knowledge is lost.

Specialized tools optimize workflows during coding, testing and deployment. Selecting the best tools for the job can accelerate development and delivery. The following are great categories of tools to take into consideration:

1. Source Control Management – These track changes and enable collaboration between developers.
2. IDEs – These are known as integrated development environments for writing, running, and debugging code.
3. Build Tools – These automate compiling of source code and managing dependencies for projects.
4. CI/CI Tools – These automate builds, tests, and releases.
5. Test Automation – Repetitive test cases are turned into automatic scripts.

The right choice and mixture of tools can enable the teams on projects to develop higher quality software faster.

The preliminary model we'll discuss is the waterfall software development process - Image Source: Adobe Stock.

Along with the two following images detailing the waterfall and agile methodologies the following are some great other types of flowcharts and diagrams that help with the SDLC:

1. UML (Unified Modeling Language) Diagram – These are most useful for designing software and databases.
2. Class Diagram – Are utilized to show classes, attributes, and methods within a system and are great at describing the relationships between each other.
3. Sequence Diagram – These are great to show in what order operations occur are best utilized to display the flow of activities with user input when beginning the sequence or with database activities.
4. Object Diagram – These work in an abstract manner and utilized in the planning stages of software development.
5. Use Case Diagram – These are great at mapping our requirements of a system.

Now here are some examples of the previous different types of diagrams and flowcharts.

Here we have a UML Class Diagram

A diagram of a bank account

Description automatically generated

Here we have a sequence diagram.

A diagram of a workflow

Description automatically generated

Here we have an object diagram.

A diagram of a company

Description automatically generated

Here we have a use case diagram.

A diagram of a car

Description automatically generated