Joshua Byrd
IBG FastTrack Summer
June 10th, 2020

SDLC Report

There are many different Software Development Lifecycle methodologies and each of them comes with certain features that make them beneficial to use. This report will cover some of the key SDLC methodologies along with their pros and cons. Waterfall is one of the oldest SDLC methodologies that software engineers used. This method has a very linear structure. The 6 stages include requirement analysis, system design, implementation, testing, deployment, and maintenance. Some of the pros to this method are that it defines the goals and deliverables at an early stage of the project. In addition, each stage is clearly defined and understood. Also, the management of the project can be somewhat easy since there are little to no unnecessary changes in the development process. Some of the cons of this method include that this may not be very suitable for long term projects. Also, towards the later stages of this method it is practically impossible to make certain changes. There is a high amount of documentation that is required for this model. The final working model of the software will only be ready at the very end of the life cycle so this can be viewed as a con for clients.

Agile is a popular SDLC model that has emerged during the past decade. This methodology uses an incremental approach. Through ongoing release cycles, software engineers are able to make the necessary changes to the product. Scrum teams use sprints as a way to complete assigned tasks in short time frames; 2-4 weeks. That being said, one of the pros of this method is that through fast development and testing, existing gaps and issues in the requirements can be recognized and addressed. In addition, less documentation is required. Project managers and stakeholders are actively engaged in the development of the software as they are able to provide feedback throughout the process. Some of the cons of this model include that it is difficult to estimate the exact requirements/effort needed at the very beginning of the project. This leads to there being high-risk probability when clients and end users are unsure about the requirements. This is why it is important for project managers to be as clear as possible when starting projects and explaining the requirements to their team.

The Iterative SDLC methodology produces a new and better version of the software during each phase. This is repeated until the final software is complete. The software is tested, evaluated, and then further requirements are identified to make the correct improvements to it. The pros of this model include having the capacity to plan parallel developments. In addition, each iteration is easy to manage, and changes can be accommodated as the project progresses. Flaws and bugs in the software can be identified much earlier in the process rather than later. Less time for documentation is required, giving more time for designing. The cons of this model include that it is not suitable for smaller projects. In addition, it can tend to require a high number of resources to be completed. While all the requirements are not gathered at the beginning of the process, it is possible that there are design issues that can come up later in the process.

The Lean methodology is another key SDLC model that takes inspiration from lean manufacturing practices and principles. This model follows seven principles: eliminate waste, amplify/refine learning, decide as late as possible, deliver as fast as possible, empower the team, conceptual integrity, see the whole/operating from the top-level. Lean prioritizes the elimination of waste to provide more overall value for their clients. To do this, teams skip unimportant meetings and reduce documentation. Pros of this model include being able to integrate teams to optimize collaboration. Lean shares similarities with Agile and DevOps so those models are able to work well with it. Lean model is easily scalable and makes it a good fit for large projects. Also, lean can deliver more functionality in a shorter time span than some other models. Some of the cons of this model include that it is very team dependent meaning that if one member of the team lacks experience then the whole project can be a failure. In addition, providing a high quality of documentation is imperative to the success of a project using lean.

The V model is short for Verification and Validation Model. This model is inspired by the Waterfall model however, V model uses testing during each stage of the development process. When one stage is completed, the next then begins. Some of the pros of the V model include its ability to avoid a downward flow of defects. There is proactive tracking of the potential defects that can be found in the software. This model is a great fit for small projects with understandable requirements. The planning and designing for these projects are completed in the beginning of the process so this saves time. Cons of this model include its rigidness and how its less flexible than the Waterfall model. Also, an early prototype is not created so clients only get the final product at the end of the process. In addition, requirements and test documents must be updated if there are any changes during the development of the software.

The Spiral methodology is a flexible SDLC model that allows teams to build a highly customized product. There are 4 phases in the Spiral methodology; planning, risk analysis, engineering, and evaluation. THese phases are gone through repeatedly until the project is completed. THe pros of this model include its emphasis on customer feedback, better risk management with repeated development, and faster development and systematic addition of features. In addition, this model allows new changes to be made at later stages of the development. Cost estimation is also fairly easy as the prototype is built in increments.
The cons of this model include its demand for risk management expertise, required documentation, and the high risk of not meeting budget or schedule deadlines. ANother con of this model is that it is not fit for smaller projects.

Rapid Application Development (RAD) is an accelerated development process with quick prototyping. This model focuses on producing more functional software and prototyping. The four phases of RAD are requirement planning, user design, construction and cutover. The first two phases are very important because they go through multiple iterations until the product is complete and fulfills all of its requirements. One of the pros of this model is that the client stays updated during the development process. They are able to provide feedback which can then be implemented into the product. Another pro is that the tasks of the project are achievable and broken up for teams to work on. Some of the cons of this model are that it is highly dependent on a strong team as well as it requires highly skilled and experienced designers and developers. Also, RAD is an expensive model and it is not feasible for projects with low budgets.