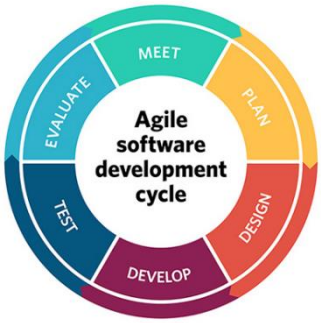
Skills

By Christopher Lebovitz

Class: CSC 424

Date: 4/16/21

For this paper, I will be writing about three software development methodologies I want to learn about. These methodologies divide work into smaller steps or subprocesses to improve design, project management, and product management. These methodologies are also known as software development life cycles. The software methodologies didn't emerge until around the 1960s with the main idea to "pursue the development of information systems in a very deliberate, structured and methodical way, requiring each stage of the life cycle—from the inception of the idea to delivery of the final system—to be carried out rigidly and sequentially (Elliott, 2004).". Since then, several different types of methods have developed software development. The methods I will be talking about are Agile, Waterfall, and Lean.



Agile is when a team discovers requirements and develops solutions through a collaborative effort of self-organizing and cross-functional teams with their product owners or end-user in the company. It is meant to be more flexible methods such as Waterfall, which were very rigid and required a lot of micromanaging to develop a program. It was created the early as 1957, but it has rapidly gained popularity since then and is the most common software development method you will find implemented at companies worldwide. In 2001, about 17 developers discussed the future of software development. They shared a common frustration about how the current design methods were running the design lifecycle. What came out of that meeting is a document that is called the agile manifesto.

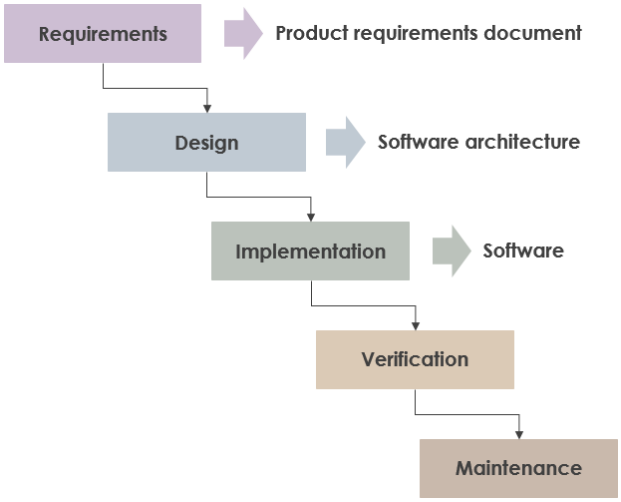
The Agile manifesto lays out 12 principles:

1. Customer satisfaction by early and continuous delivery of valuable software.
2. Welcome changing requirements, even in late development.
3. Deliver working software frequently (weeks rather than months)
4. Close, daily cooperation between business people and developers
5. Projects are built around motivated individuals, who should be trusted
6. Face-to-face conversation is the best form of communication (co-location)
7. Working software is the primary measure of progress
8. Sustainable development, able to maintain a constant pace
9. Continuous attention to technical excellence and good design
10. Simplicity—the art of maximizing the amount of work not done—is essential
11. Best architectures, requirements, and designs emerge from self-organizing teams
12. Regularly, the team reflects on how to become more effective, and adjusts accordingly

These principals really show the flexibility of Agile and how it prioritizes individuals and their interaction over tools, and it introduces feedback from the customer a lot earlier than other methodologies. Agile is broken down into short sprints or iterations, which are short time frames

that can last a week up to four weeks, depending on the level of effort the ask is. In Agile, a development team is also self-organizing, meaning that the team is the one who decides what the best approach the worked that is asked of them rather than being directed by a project manager.

In Waterfall, a project is broken down into a linear series of phases, and each one depends on the output of the previous phase. This methodology is the opposite of Agile. It is stricter when it comes to the development process because it flows in one direction, so it's called Waterfall. Waterfall was first used in manufacturing and construction where the method's high structure was needed to be finalized much sooner because changing anything after gathering requirements would incur a massive cost.



There are 6 phases in Waterfall, and when one ends, the other begins. They are System, Analysis, Design, Coding, Testing, Operations/Maintenance. The first phase is the requirements phase, where those in project management and the developers get together to start drafting the requirements for the program. The Analysis phase is where the technical step of completing the products are defined, for example, business rules, schemas, and models. Design is the third phase that establishes the Software architecture of the program. The Coding Phase is where actual development begins. During this phase, unit testing plays a significant role and is used to ensure that the code that was implemented works as the developer expects it to. Here is where integration happens to ensure what is being developed works with what was already developed and put in place.

Testing is where debugging and manual testing occurred. This is where the product owners and other people see who are knowledgeable get to have a hands-on with the product and identify any issue and have the developers correct them. The last phase is operations or maintenance, depending on who you are talking to, but here is where the development process is being wrapped up, and the code is being packaged to be delivered to the customer and installed onto their machines. Support for the product happens in this stage. It is most likely why some people refer to this stage as maintenance instead of operations.

Lean is a translation of manufacturing ideology and practices that were ported from the Toyota Production System. This method originated from a book called *lean software development*, written by Mary and Tom Poppendieck in 2013. The book goes over the lean principles that Toyota used and compared them o corresponding agile practices.

There are seven lean principles outlined in the book: eliminate waste, amplify learning, decide as late as possible, deliver as fast as possible, empower the team, build integrity, and optimize the whole. When talking about the first method in Lean, it's referring to removing or not adding anything that may be considered as not having value to the product. These wastes may include features that were not asked for, defects, work that was partially completed, and management activities. Amplifying learning is where the learning process is sped up by using short cycles such as sprints and increasing feedback from customers using short sessions to help the developers gain as much knowledge about the product, domain, and intent the client wants for the program.

When deciding as late as possible, decisions should be delayed as long as possible until all the facts and knowledge have been gathered to make an informed decision so that mistakes aren't made out of ignorance. On the opposite end, the team should deliver as fast as possible to receive feedback sooner and be incorporated in the next sprint. Empowering the team is different than most business relations ships between developers and managers. Instead of the managers telling the devs what to do, the roles are flipped. The dev teaches the managers about their needs, and it's the manager's responsibility to address those needs and remove any challenges they may have. When talking about building integrity, lean means that from conception to production, the program should be developed as thoroughly as possible with knowledgeable customers by using testing and functional testing and making sure that those working on the program understand the problem entirely. The last principle is optimize the whole, and what it means by that is that the problem should be broken down into small pieces and worked on, tested, and optimized so that when it comes time to put the pieces together, it is more optimized than it would have been if the problem was worked in one large chunk.

Looking at these different software development methodologies, it's plain to see their need in the workplace. They help organize a team, break down the development process into manageable pieces, and promote delivering a working program.