**Software Development Methodologies: a comparison of Traditional and Agile approaches**

**Student**: Patrick Walsh

**Date**: 3/21/2021

**School**: University of Maryland Global Campus

**Course**: SDEV 360 7380

**Professor**: Dr. Alla Webb

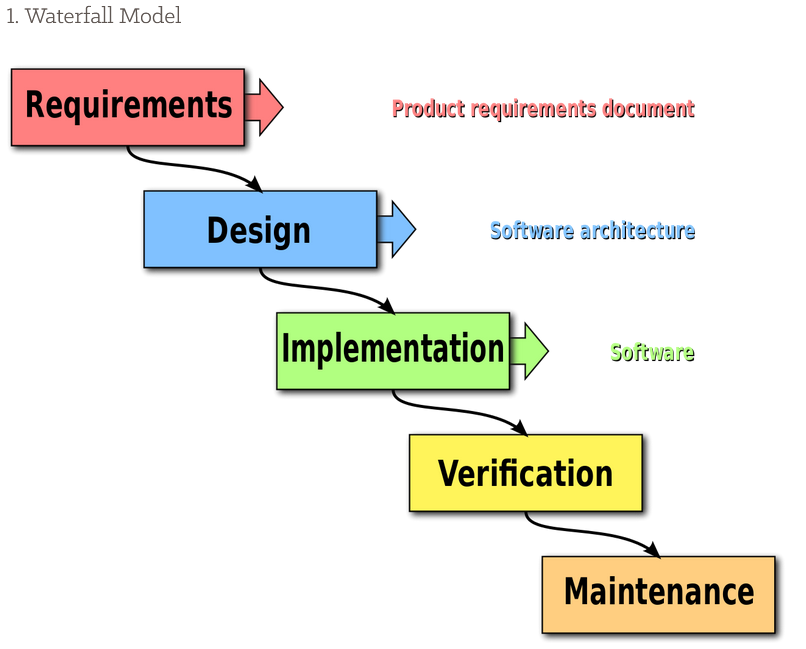
**Software development approaches**

The Software Development Lifecycle requires more than developers sitting in front of a computer writing code. There are numerous approaches and standards that govern how software development teams execute projects. This paper will focus on two of the primary methods of software development: Traditional and Agile. Each method has its own set of advantages and disadvantages, so when it comes to real-world implementation, there is no one-size-fits-all.

**Traditional: a planned approach to software development**

Within the Traditional software development method, one of the most popular approaches is known as the Waterfall approach. Waterfall is a sequential process, planned from beginning to end (or top to bottom like a waterfall) at the outset when requirements are being defined. In this approach, the entire project is divided into separate phases, with the result of the preceding phase acting as the input for the next phase (Rungta, 2021).

**Figure 1**, *Waterfall approach phases*



Note. As seen in Figure 1 (copyright 2020 by Creative Commons), the project is broken up into a Requirements Phase, a Design Phase, an Implementation Phase, a Verification Phase, and a Maintenance Phase (Insight Editor, 2020).

In the Requirements Phase, the development team gathers the requirements from the customer so that the desired end-product is well defined and understood. The project timeline, budget, and scope of work are all planned out in advance during the Requirements Phase before any code is written. Once the Requirements have been defined, the team then moves into the Design Phase where the software is designed. After design, the code is written in the Implementation Phase. After Implementation, the code is deployed and tested in the Verification Phase, and performance is monitored during the Maintenance Phase. The Waterfall approach is rigid, requiring each phase to be completed in sequence without returning to previous phases to make changes. For this reason, it is vitally important to rigorously document each phase of the project. It also means that the Waterfall approach is best suited for projects that do not anticipate unforeseen changes during development (Insight Editor, 2020).

One disadvantage to the Waterfall approach is that it is difficult to take into consideration unforeseen changes in technology which could affect the requirements of the project after work has already begun. The customer will not see the finished product until the end of the project. So, if the customer changes the requirements in the middle of the process, it is very difficult to go back and make these changes because everything has already been planned out. The Waterfall approach is effective when the customer has a clear picture of exactly what they want to see in an end-product, but if the customer wants to provide feedback and make changes during the development process, the Waterfall approach is not ideal (Insight Editor, 2020). Instead, the Agile approach is a better fit.

**Agile: an iterative approach to software development**

Agile is an incremental approach to software development that seeks to make the development process more flexible and customer-driven by integrating continuous development and feedback into the process. Rather than planning out the entire project from beginning to end at the outset as in the Traditional Waterfall approach, the Agile method breaks the project down into numerous short phases, with emphasis on delivering functional software as soon as possible and then making continuous improvements throughout the project (Flora, 2014). The Agile approach allows development teams to be highly flexible and respond to changing requirements either due to customer feedback or to changes in the environment. Unlike the Traditional Waterfall approach, the Agile approach does not require that the project be conducted in sequential phases. Rather, the project is carried out in a series of short phases, sometimes called sprints, that allow the team to adapt to new and changing requirements throughout the project. Unlike the Traditional Waterfall approach, the Agile approach does not rely heavily on written documentation (Flora, 2014). Instead, the Agile approach emphasizes effective communication within the team in the face of evolving requirements and customer feedback (Flora, 2014).

There are numerous approaches within Agile, but one of the most well-known approaches is Extreme Programming (XP). XP is a development philosophy that emphasizes a lightweight, collaborative approach (Flora, 2014). It is best suited for small development teams of 2-12 members, preferably working in the same physical location on short projects, normally 1-3 weeks (Flora, 2014). XP has 4 core values: Communication, Simplicity, Feedback, and Courage (Flora, 2014). Teams using the XP approach must practice good communication and willingness to adapt to rapidly changing requirements. In order to facilitate a collaborative mindset, XP teams follow certain practices like pair programming which involves two developers sitting at the same workstation and coding a piece of software together (Tripathi & Mishra, 2017). While not all projects are suitable for the XP approach, projects whose requirements are unclear or likely to change are ideal because of XP’s openness to change. However, without an experienced team that practices strong collaboration and communication, the XP approach may not have good results (Tripathi & Mishra, 2017).

There are other Agile approaches which are commonly used in software development, including Agile Modelling, SCRUM, Crystal methodologies family, Feature-Driven Development, and Adaptive Software Development (Tripathi & Mishra, 2017). While this paper will not go into the details of each of these approaches, it should be noted that Agile methodologies in general are best suited for projects that have requirements which are unclear and where the customer is unsure what they want to see in the end-product. Agile is also ideal for responding to unforeseen changes in technology that can affect requirements or feasibility of approach.

One of the downsides to the Agile methodology is that it can make the process chaotic if the team is not all on the same page and requirements are not being effectively communicated by project managers at all steps of the project. Due to the dynamic nature of the Agile approach, the requirements and scope of work can go through numerous changes throughout the development lifecycle. For this reason, the Agile approach requires experienced team members who work together and communicate effectively to keep everyone on the same page.

**Deciding which approach to use**

To summarize, two of the primary methods of software development are the Traditional and Agile approaches. Traditional approaches such as the Waterfall approach require extensive planning and documentation at the outset and are implemented in sequential phases. The Traditional approaches are best suited for projects where the customer has clearly defined requirements from the outset and these requirements are not expected to change during the course of project development. Agile approaches, on the other hand, seek to make the software development process more incremental and flexible, allowing continuous customer feedback throughout the project and adapting to changing requirements as the project develops. Agile approaches such as XP emphasize good communication and collaboration among team members to adapt to rapidly changing requirements as the project develops, while Traditional approaches like Waterfall rely heavily on good documentation. While the Agile approach can yield good results with teams that are experienced and practice good communication, this approach can lead to chaos and confusion if teamwork and collaboration breaks down.

**References**

Flora, H. (2014). A Systematic Study on Agile Software Development Methodologies and Practices. *International Journal of Computer Science and Information Technologies.* 5. 3626-3637.

Insight Editor. (2020, February 4). *Software Development Models: Processes in Application Engineering*. Insight Canada. https://ca.insight.com/en\_CA/content-and-resources/2016/07152016-types-of-software-development-models.html

Rungta, K. (2021, March 8). *SDLC: Phases & Models of Software Development Life Cycle*. Retrieved from https://www.guru99.com/software-development-life-cycle-tutorial.html#11

Tripathi, V. Y., & Mishra, A. (2017, November 11). Case Study of Agile Methodologies in Field of Software Development. *International Journal of Engineering Research & Technology*. 6(11). 2278-0181