Kali Burres, 41

Professor Christine James

CIS 121-00

10/25/2019

Waterfall vs Agile Software Development and Project Management

The Waterfall and Agile methods are two approaches that have formed to define and guide the process of software development. Because Agile was developed out of a frustration with the downfalls of Waterfall, the two methods tend to vary directly in their strengths and weaknesses. Both present strong options for the organization of a team and formation and execution of a plan when developing an application. Before they can be fully compared, however, they must be understood on their own.

The Waterfall, or top-down, software development and project management method is "essentially…linear", focusing on accomplishing tasks one at a time and in succession, with an emphasis on thoroughness, structure, and reaching milestones (Grech). This is accomplished by dividing all the necessary phases of development into defined projects of their own and tackling them in order, finishing each project before moving on to the next stage. The stages are as follows: Requirement gathering, analysis, design, coding, testing, and deployment. During the Requirement Gathering phase, a team undergoes the painstaking process of identifying each and every function the software in question should have, methodically analyzing and recording them for the reference of the next team. The resulting document is called a specifications document. The Analysis phase involves analyzing the specifications document and then determining how the requirements stated in it will be put into place. It is during this phase that specific models and business logic are discussed and agreed upon. During the design phase decisions related to the programming of the application begin to take place. Not only are graphics, layout, and aesthetic design chosen, but the external services needed, and programming language to be used in the coding phase are decided on. Then in the coding phase the actual source code is written by a team of computer programmers. The testing phase involves taking the produced source code and testing each and every requirement outlined in the specifications document for proper compliance and function in order to systematically discover all bugs in the code. When these undoubtable bugs are discovered, the coding phase begins again. This cycle continues until no bugs are found. Finally, the deployment phase involves the training of employees, official documentation, and maintenance work that is necessary to launch and maintain a piece of software. Because of the segregation between stages, each of these stages is often completed by a completely different team.

There are many advantages to using the waterfall method. The distinct divisions between each phase, each phase can be worked on by a team that is uniquely specialized to the task at hand. This also encourages outsourcing to specialists, which in many situations will help yield a better product. The rigid structure of the waterfall approach also forces organization, something critical especially in large projects. The meticulous planning in the first phase also allows for easy changes in the beginning, as nothing has been coded yet. Further along in the process, however, waterfall looses that flexibility. It has been criticized for its nonadaptive structure, which also causes consumer feedback to go unappreciated and mid-process ideas to be ignored. This often results in a lesser-quality product being released in order to stay on time and budget. Delaying the testing period until the end can also come with grave consequences, as important (and expensive) bugs may not be found until the end.

The Agile project management approach, most commonly through the scrum method, focuses on iterative and incremental processes. Unlike its counterpart, the Agile methodology prioritizes flexibility, interaction, and collaboration. Developed in 2001 to find an alternative to the time-consuming, rigid, and often flawed top-down design method, The Agile method comes with its own manifesto which outlines the priorities and goals of the method by claiming, "Individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, [and] responding to change over following a plan" (Manifesto).

The Scrum process works within a series of usually two or three week periods known as sprints. Each sprint is more or less a miniature version of the waterfall method, focusing on only a small handful of tasks. Throughout one sprint all the necessary development phases for those tasks are completed, and then at the end of the duration a new sprint begins. The scrum process can be outlined by the following elements: backlog creation, sprint planning, sprint backlog, scrum team and daily scrum, sprint review, and implementation. The product backlog is the waterfall equivalent of a specifications document, although the two differ greatly in their details. During the sprint planning phase the goals for the sprint are defined and responsibilities are assigned, and that outline of expectations is used to create the sprint backlog, a smaller backlog focusing exclusively on the tasks for the sprint at hand. Then the sprint begins. During this time the development team meets daily to report on current successes and struggles, to give advice and make plans in addition to working individually on assigned responsibilities. At the end of the sprint, the team meets again to review the progress made in the current sprint and reports that progress to the customer. The product backlog is also adjusted (if necessary) after every sprint, as opposed to a specifications document which is never changed once it is complete.

Where waterfall fails for its rigidity and impersonality, Agile excels. “Agile begins with the assumption that not all of the work is of equal criticality”, allowing the team to get the most important tasks done in the order that they see fit and make smaller adjustments later (Grech). It usually involves one team throughout the whole process, allowing them to work together well and take responsibility over the whole project. Because the work is accomplished in small increments, agile provides the capability to welcome change instead of fearing it. The scope of the project does not need to be known in advance, and consumer input is welcome and easily incorporated, resulting in a product more satisfying to the customer. Like its counterpart, however, the method has its downfalls. The agile method proves financially riskier than waterfall, and provides almost no opportunity for outsourcing or specialization. Teams have also been known to struggle to manage their projects organizationally without the structure waterfall provides (“Agile or Waterfall”).

Both Waterfall and Agile software development carry their advantages, however, recent research suggests that the best solution is not found in either, but both. Since both strategies carry glaring errors which the other solves nicely, some software development teams have attempted a merging of the two strategies in order to become a more well-rounded team with a higher chance for success. And it seems to be working. In a recent study, over three quarters of applications developed with a mix of both methods scored higher on robustness, security, and changeability than applications developed with only one method or the other (“Agile or Waterfall”). Many computer scientists see the divide as silly. After all, computer programming is all about change and adaptability, and we have failed to do just that in our project management strategies, instead clinging to the hard-enforced either-or between two, while credible, faulty methods. As Dr. Bill Curtis put it, “It’s time to take religion out of software development and get back to sound software engineering" ("Agile or Waterfall"). Combining these strategies won't be easy, however, and the endeavor will undoubtedly provide challenges of its own. The important thing remains understanding the nature of the project at hand and discerning what method will suit it best, and always remaining open to learn, grow, and adapt in order to achieve the best results.

Works Cited

Applications built with either agile or waterfall methods alone are more susceptible to security, reliability, performance, and cost issues." Database and Network Journal, Oct. 2014, p. 21. Gale OneFile: Computer Science, https://link.gale.com/apps/doc/A388428316/CDB?u=philbibu&sid=CDB&xid=26df6b5d. Accessed 21 Oct. 2019.

Grech, Tony. "The Intersection of Agile and Waterfall." *Industrial Engineer: IE*, vol. 47, no. 8, Aug. 2015, pp. 47–49. *EBSCOhost*, search.ebscohost.com/login.aspx?direct=true&AuthType=ip,sso&db=iih&AN=109483551&site=ehost-live&custid=s9004953.

"Manifesto for Agile Software Development." *Manifesto for Agile Software Development*, agilemanifesto.org/.