Waterfall versus Agile

Which is the Right Development Methodology for Your Project?

Content assembled by R. Max Wideman, FPMI

Published here April, 2018

I frequently receive, or come across, comments, suggestions and recommendations for improving our understanding of practical project management. I typically pursue the better ones with the author to see if I can give them broader publicity. This I can do by working them into one of my Musings, or perhaps into comments on a Book Review or Guest paper. The following is one such text, indeed several years old, but the author has requested not to be identified. Nevertheless, after several more months of contemplation, I still think it is worth exposing in this months Max's Musings.

But first of all, given the title "Waterfall vs. Agile", we must be clear that we are talking about the choice of implementation methodology for one particular group of projects. This group falls under the heading of *Info Technology (& High Tech work)*, which is Type 4 in my simplified Typology.[1] By implication, the methodology described as "Agile" is generally *not* suited to the other specific groups: Construction, Healthcare, and Manufacturing.[2]

Let's look at an illustration of the comparison between Waterfall and Agile, assuming that both approaches involve seven identifiably different phases in their Product Development life span, see Figure 1.[3]

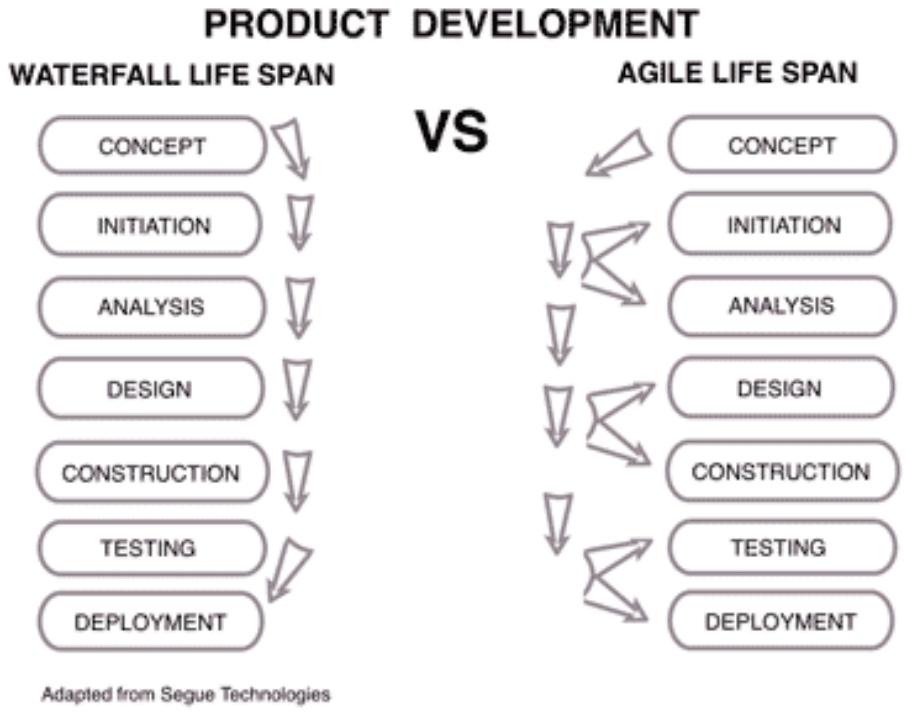

Figure 1: Product Development Life Spans for Waterfall and Agile methodologies

Assuming that this illustration is a realistic version of the comparison between the Waterfall and Agile approaches,[4] the arrangement of arrows representing progress, looks much more work-intensive in the case of Agile, compared to Waterfall. However, the reality is that Waterfall requires virtual completion, unanimity amongst the performing stakeholders of each phase, and signoff, before moving on to the next one, because there is no turning back. Imagine designing and building a railroad route or new highway

development, or any other type of major infrastructure project for that matter, and then up-rooting and going somewhere else? This is just not acceptable. Yes, I know it has been done, but at the cost of great public outrage and expense.

The Agile approach, on the other hand, says: "OK, we know where we are, and we know its not perfect, or there's more to be done. But let's 'send up a trial balloon' (representing the start of the next phase), and let us learn from that!" This represents a huge saving of time as it effectively cuts off any lingering doubts that could otherwise extend the current phase. And if it doesn't work, in IT projects you can much more easily go back and rework, with the loss of time and cost minimalized, compared to the alternative.

Now, suppose you are faced with a new IT category project. One of the first decisions you face is "Which product development methodology should we use?" This is a topic that usually gets a lot of discussion (and often heated debate).

If this is not something you've had to face before, descriptions of the two development methodologies is in order. Put very simply, each is the way of organizing the work of software development. This is NOT about a style of project management, nor a specific technical approach,[5] although you will often hear these terms all thrown together or used interchangeably. Hence:
Waterfall: Perhaps better recognized as the "traditional" approach, and
Agile: a specific type of Rapid Application Development, newer than Waterfall, but not that new. It is often implemented using "Scrum".[6]
Both of these are usable, mature methodologies.

Our original author shares her thoughts on the strengths and weaknesses of each in the following pages.

The Waterfall Methodology

Waterfall is a linear approach to software development. In this methodology, the sequence of events is something like:

1. Gather and document requirements
2. Design
3. Code and unit test
4. Perform system testing
5. Perform user acceptance testing (UAT)
6. Fix any issues
7. Deliver the finished product

In a true Waterfall development project, each of these represents a distinct phase of software development, and each phase generally finishes before the next one can begin. There is also typically a stage-gate between each; for example, requirements must be reviewed and approved by the customer before design can begin. There are good things and bad about the Waterfall approach. On the positive side:

- Developers and customers agree on what will be delivered early in the development lifecycle. This makes planning and designing more straightforward.
- Progress is more easily measured, as the full scope of the work is known in advance.
- Through out the development effort, it is possible for various members of the team to be involved or to continue with other work, depending on the active phase of the project. For

example, business analysts can learn about and document what needs to be done, while the developers are working on other projects. Testers can prepare test scripts from requirements documentation while coding is underway.
- Except for reviews, approvals, status meetings, etc., a customer presence is not strictly required after the requirements phase.
- Because design is completed early in the development life span, this approach lends itself to projects where multiple software components must be designed (sometimes in parallel) for integration with external systems.
- Finally, the software can be designed completely and more carefully, based upon a more complete understanding of all software deliverables. This provides a better software design with less likelihood of the "piecemeal effect," a development phenomenon that can occur as pieces of code are defined and subsequently added to an application where they may or may not fit well.

Here are some issues that our author has encountered using a pure Waterfall approach:
- One area that almost always falls short is the effectiveness of requirements. In my opinion, gathering and documenting requirements in a way that is meaningful to a customer is often the most difficult part of software development. Specific details, provided early in the project, are required with this approach, but customers are sometimes intimidated by such details. In addition, customers are not always able to visualize an application from a requirements document. Wireframes and mockups can help, but there's no question that most end users have some difficulty putting these elements together with written requirements to arrive at a good picture of what they will be getting.
- Another potential drawback of pure Waterfall development is the possibility that the customer will be dissatisfied with their delivered software product. As all deliverables are based upon documented requirements, a customer may not see what will be delivered until it's almost finished. By that time, changes can be difficult (and costly) to implement.

The Agile Methodology

The sequence of the development phases, as shown in Figure 1 earlier, are essentially the same. However, Agile is an iterative, team-based approach to development. This approach emphasizes the rapid delivery of an application in complete functional components. Rather than creating tasks and schedules, all time is "time-boxed" into phases called "sprints." Each sprint has a defined duration (usually in weeks) with a running list of deliverables, planned at the start of the sprint. Deliverables are prioritized by business value as determined by the customer. If all planned work for the sprint cannot be completed, work is reprioritized and the information is used for future sprint planning.

As work is completed, it can be reviewed and evaluated by the project team and customer, through daily builds and end-of-sprint demos. Agile relies on a very high level of customer involvement throughout the project, but especially during these reviews.

Some advantages of the Agile approach:[7]
- The customer has frequent and early opportunities to see the work being delivered, and to make decisions and changes throughout the development project.
- The customer gains a strong sense of ownership by working extensively and directly with the project team throughout the project.
- If time to market for a specific application is a greater concern than releasing a full feature set at initial launch, Agile can more quickly produce a basic version of working software which can be

built upon in successive iterations.
- Development is often more user-focused, likely a result of more and frequent direction from the customer.

Some disadvantages of the Agile approach:
- The very high degree of customer involvement, while great for the project, may present problems for some customers who simply may not have the time or interest for this type of participation.
- Agile works best when development team members are fully dedicated to the project.
- Because Agile focuses on time-boxed delivery and frequent reprioritization, it is possible that some items set for delivery will not be completed within the allotted timeframe. Additional sprints (beyond those initially planned) may be needed, adding to the project cost. In addition, customer involvement often leads to additional features requested throughout the project. Again, this can add to the overall time and cost of the project as compared to its business case.
- The close working relationships required in an Agile project are more easily managed when the team members are located in the same physical space. This is not always possible, but there are a variety of ways to handle this issue, such as webcams, collaboration tools, etc.
- The iterative nature of Agile development may lead to a frequent refactoring if the full scope of the system is not considered in the initial architecture and design. Without this refactoring, the system can suffer from a reduction in overall quality. This becomes more pronounced in larger-scale implementations, or with systems that include a high level of integration.

Making the Choice

So, how do you choose between Agile and Waterfall? For example, in the original author's case they may change the game a little (which is what most software development organizations do anyway). They may define their own process through a variation on the traditional Waterfall methodology. Such modifications include use of prototyping where possible to provide the customer a better view of their finished product early in the design/development cycle.

This helps to improve the team's understanding of requirements and communication with the customer. After the primary framework of the application is completed according to high-level requirements, they continue to develop and also to reach out to the customer for refinement of requirements. In this way, they strive to be as iterative (Agile) as possible without compromising their overall system architecture.

Once they've decided which basic methodology to use, they can further refine the process to best fit their project goals. Ultimately, although the way in which they do the work is important, delivering a solid and maintainable product that satisfies the customer is what really counts.

---

[1] See http://www.maxwideman.com/papers/challenge/comments.htm and http://www.maxwideman.com/papers/glossary6_1/goal.htm
[2] Although even in these groups, it is not unusual to set up a trial section to test the feasibility of any new technology, or to test the particular challenges to be faced in execution.
[3] This illustration is a modified version of the one presented with the original text, both for clarification and copyright concerns.
[4] A comparison though obviously simplified.
[5] Actually, I rather think it is. That is, it is about the preferred management of the development of the product.
[6] See http://www.maxwideman.com/guests/stateofart/specific.htm
[7] For more Agile Development benefits, please see *8 Benefits of Agile Software Development*