The Advantages of Agile Development

John Foley

13 April 2014

Abstract

Agile Methodologies have taken the software engineering industry by storm, especially in this modern era of internet technology and application production. The Agile Manfiesto, published in 2001, introduced many terms and ideas to describe the current set of methodologies that we call Agile. There are many problems in software engineering that Agile Methodologies are designed to address, such as ability to react to change, maintain communication with clients, and evolutionary development cycles. These problems are inherent in software development, and are not handled well in past methodologies such as Waterfall.

1 Iterative Cycles

One strength of the Agile Methodology is how the software development life cycle is split up. Agile thinks of the time it takes to solve a specific feature or problem as an iteration, typically two weeks long, but is up to the production manager. This allows for structured development of the application, and breaks up the requirements from the user into smaller, easier to handle chunks.

1.1 Iterations

The structured phases of feature and problem development of the application

1.2 Separation into Easier Phases

Each iteration allows for the application to be developed in parts. This allows for flexible changes and extremely reactive development. [1]

2 Small, Cross-functional Teams

2.1 Small Teams and Scrums

Teams are kept small in order to maintain inter-team communication. Internal workings are easier to maintain and manage, as well as becoming more comfortable with whom a developer works with.

2.2 Scrums and Meetings

Scrums are quick, efficient meetings for teams to keep on track. Scrums are held often and typically report driven.

3 Evolutionary Development

3.1 Test Driven Development

The use of Test Driven Development allows for an application that ensures that past work done on the application is still working with the addition of new code.

3.2 Increment

Each iteration has a specific goal in mind. After each iteration is complete, the application is one step further, and considered incremented forward.

4 Past Methodologies

The Waterfall Methodology has been the primary methodology used for software development for decades, and is still used today. These methodologies are effective, but do not address major problems that lead to either failure of a software development project or shortcomings of an application.

4.1 Phases Slow Reaction Time

Waterfall is typically held with several phases, one of which is the design phase, followed by implementation phase. These phases are essentially "closed door" meetings for developers from clients, and so divergence and thus failure to respond to change is a problem.

4.2 Maintain Communication

Waterfall is not known to maintain strong client communication, and without constant updates and checks on progress, the project will inevitably diverge from what the client is asking for.

4.3 Evolution of the Product

Waterfall methodologies advises development to be carried out in phases, without clear benchmarks to be made. A common problem is that the project is developed all at once and changing subcomponents becomes difficult and overwhelms a project.

References

[1] Lan Cao, Balasubramaniam Ramesh, and Tarek Abdel-Hamid. Modeling dynamics in agile software development. *ACM Trans. Manage. Inf. Syst.*, 1(1):5:1–5:26, December 2010.

[2] André Janus. Towards a common agile software development model (asdm). *SIGSOFT Softw. Eng. Notes*, 37(4):1–8, July 2012.

[3] Oualid Ktata and Ghislain Lévesque. Agile development: Issues and avenues requiring a substantial enhancement of the business perspective in large projects. In *Proceedings of the 2Nd Canadian Conference on Computer Science and Software Engineering*, C3S2E '09, pages 59–66, New York, NY, USA, 2009. ACM.

[4] Magnus Thorstein Sletholt, Jo Hannay, Dietmar Pfahl, Hans Christian Benestad, and Hans Petter Langtangen. A literature review of agile practices and their effects in scientific software development. In *Proceedings of the 4th International Workshop on Software Engineering for Computational Science and Engineering*, SECSE '11, pages 1–9, New York, NY, USA, 2011. ACM.

[5] Christoph Johann Stettina and Werner Heijstek. Necessary and neglected?: An empirical study of internal documentation in agile software development teams. In *Proceedings of the 29th ACM International Conference on Design of Communication*, SIGDOC '11, pages 159–166, New York, NY, USA, 2011. ACM.