# 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 and adapt, maintain communication with clients instead of renegotiating contract, and evolutionary development cycles instead of planned in phases. These solve problems that are inherent in software development, and are not handled well in past methodologies such as Waterfall.

## 1 Client Contact with Small, Cross-Functional Teams

Communication with the client during the life of software is critically important. It ensures that the product matches what the client is expecting. In the past, client requirements are constructed into a contract for the developer to create, but that allows for divergence if every detail is ambiguous. Agile Methodologies that teams and a client representative create a relationship to stimulate communication.

### 1.1 Client in the Room

Agile Methodlogies advocate for someone with domain knowledge be present during development so that specific details and questions can be answered immediately. Of course this would be hard and costly to maintain, so good communication should be maintained in its stead.

### 1.2 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.

### 1.3 Scrums and Meetings

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

## 2 Iterative Cycles

One strength of the Agile Methodology is how the software development life cycle is separated into easier to handle segments. 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.

### 2.1 Iterations

The structured phases of feature and problem development of the application

### 2.2 Separation into Easier Phases

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

## 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] G. O. Barnett. History of the development of medical information systems at the laboratory of computer science at massachusetts general hospital. In *Proceedings of ACM Conference on History of Medical Informatics*, HMI '87, pages 43–49, New York, NY, USA, 1987. ACM.

[2] Wilson McCoy, Jeff B. Pelz, Cecilia Ovesdotter Alm, Pengcheng Shi, Cara Calvelli, and Anne Haake. Linking uncertainty in physicians' narratives to diagnostic correctness. In *Proceedings of the Workshop on Extra-Propositional Aspects of Meaning in Computational Linguistics*, ExProM '12, pages 19–27, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[3] Lipo Wang, Feng Chu, and Wei Xie. Accurate cancer classification using expressions of very few genes. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 4(1):40–53, January 2007.

[4] Kathryn Womack, Wilson McCoy, Cecilia Ovesdotter Alm, Cara Calvelli, Jeff B. Pelz, Pengcheng Shi, and Anne Haake. Disfluencies as extra-propositional indicators of cognitive processing. In *Proceedings of the Workshop on Extra-Propositional Aspects of Meaning in Computational Linguistics*, ExProM '12, pages 1–9, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[5] F. Zhang, A. Bilas, A. Dhanantwari, K. N. Plataniotis, R. Abiprojo, and S. Stergiopoulos. Parallelization and performance of 3d ultrasound imaging beamforming algorithms on modern clusters. In *Proceedings of the 16th International Conference on Supercomputing*, ICS '02, pages 294–304, New York, NY, USA, 2002. ACM.