Agile Modeling:

An Agile Methodology for Systems Modeling

by

Michael Beaver

Dr. Patricia Roden

CS 455

30 Jan. 2014

Agile Modeling: An Agile Methodology for Systems Modeling

Agile Modeling (AM) is an agile methodology introduced and detailed by Scott Ambler in his 2002 book *Agile Modeling* [2,3]. Ambler was primarily concerned by developers’ reliance on prescriptive processes to develop software since these processes are management-oriented and unfriendly to developers. Ambler introduced AM, an agile method that maximizes developers’ efficacy and efficiency during the development process, particularly with regard to modeling [3]. On the face of it, AM appears to be an amiable effort at streamlining systems modeling, but developers should be wary of adopting AM outright.

**Agile Modeling and the Agile Manifesto**

Ambler describes AM as a “chaordic, practice-based methodology for effective modeling and documentation of software-based systems” [3]. AM is not a rigid, prescriptive process, like the Waterfall Method; rather, AM is a collection of guiding principles and practices. Ambler argues that modeling is critical to the success of software development teams, and AM aims to increase the efficacy and the efficiency of this process. It is important to note, however, that AM is not a “complete software process” unto itself [2,3]. Agile Modeling focuses solely on improving modeling, not necessarily programming practices. Ambler argues that AM ought to be used as a supplement to a base software process, such as Extreme Programming (XP). The end result is a custom process that improves the modeling process in addition to the base process’s benefits [2,3].

The core of AM lies in its values, principles, and practices. Agile Modeling has four values: communication; simplicity; feedback; courage; and, humility. It is worth noting that the first four values are also the values of XP [2]. Effective modeling, according to Ambler, leads to improved communication among developers and stakeholders. Simplicity aims to keep software developers from being overwhelmed with complex and potentially confusing or contradictory models. Timely, accurate, and useful feedback can often be the difference between a project being on time and on budget or delayed and over budget. Courage is inherent in agile methods since following an agile methodology means eschewing the safety nets of more prescriptive processes. Humility, Ambler argues, is the hallmark of the best programmers that know their own limitations [2,3].

Agile Modeling also features a set of ten core principles. Agile Modeling’s core principles include: software as the primary goal; enabling the future as the secondary goal; light documentation; assumed simplicity; embracing change; incremental change; purposeful modeling; multiple models; quality work; and, maximizing the stakeholder’s investment. The principles work together, directly or indirectly, to produce a “high-quality software that meets the needs” of the stakeholders in an “effective manner” [3]. The lightweight and driven nature of AM allows the stakeholders to request changes, significant or trivial, without upsetting or arresting the entire development process.

Agile Modeling’s values and principles are supplemented by core practices. These core practices, according to Ambler, form “the heart of Agile Modeling” [3]. The practices are divided into four categories: iterative and incremental modeling; teamwork; simplicity; and, validation [3]. Iterative and incremental modeling involves appropriate application of artifacts, creating several models in parallel, iterating to other artifacts, and modeling in small increments. Teamwork entails modeling with others, involving the stakeholder in the project, collective ownership of the project, and the public display of the models. Simplicity is obtained by creating simple content, by depicting models in simple terms, and by using simple tools. Validation, of course, concerns testability and proving models with code [3]. These practices are not discrete phases or steps to be followed during the development process. Rather, each practice either directly or indirectly impacts another practice.

Agile Modeling’s values, principles, and practices all correspond directly or indirectly to the values of the Agile Manifesto. For example, AM values communication, feedback, courage, and humility, which correspond to the Agile Manifesto’s “individuals and interactions,” “customer collaboration,” and “responding to change” values [1,3]. All of AM’s values work together to accomplish the Agile Manifesto’s “working software” value [1,3]. Agile Modeling’s principles also correspond to the Agile Manifesto’s major and minor values. For example, AM advocates software as the primary goal and traveling light, which correspond to the Agile Manifesto’s “working software over comprehensive documentation” values [1,3]. Similarly, AM has two principles concerning embracing and reacting to change, which correspond directly to the Agile Manifesto’s “responding to change over following a plan” values [1,3]. Essentially, all of AM’s values, principles, and practices work together to address, either directly or indirectly, the values of the Agile Manifesto,

Agile Modeling’s values, principles, and practices also work together to address the twelve general principles of agile methods. For example, AM emphasizes communication, the software, and incremental and iterative development, which correspond to the first, third, and seventh principles of agile methods. Also, AM is highly receptive change, which addresses the second principle of agile methods [1,3]. Agile Modeling’s principles and practices of stakeholder involvement correspond to the fourth and sixth principles of agile methods. Also, AM’s complementary teamwork principles and practices address the fifth, eleventh, and twelfth principles of agile methods. Agile Modeling advocates traveling light, which should allow developers to “maintain a constant pace indefinitely” [1,3]. Agile Modeling has a principle of “quality work,” which corresponds directly to the general agile methods principle of “continuous attention to technical excellence” [1,3]. Lastly, AM heavily emphasizes simplicity in its values, principles, and practices; hence, AM addresses the simplicity principle of agile methods [1,3]. It is easy to see, then, that the values, principles, and practices of AM were carefully crafted to address the twelve principles of agile methods.

It is worth noting that AM is essentially an improvement on and an extension of XP; in fact, Ambler views AM as a “lead-in” to XP [7]. Extreme Programming is focused on systems development, and AM is focused on effectively and efficiently developing systems models. Using AM in conjunction with XP would “best exploit the benefits of XP” while simultaneously incorporating the benefits of AM [3,7]. It is interesting to note that AM also adopted nine of XP’s eighteen principles. It would seem that AM and XP share a certain philosophical congruency, which lends itself to the methodologies’ compatibility [3]. Agile Modeling is heavily derived from XP, and its adoption could possibly benefit the users of XP.

**Small Team Sizes, Simple Projects, and a Lack of Supporting Research**

Agile methods, such as AM, work best with certain team sizes and with certain levels of project complexity. In a 2002 workshop, Ambler and other members of the agile community concluded that agile methodologies “work well for teams of 20 to 30 people,” which should be split into smaller subgroups [4]. A study by Hirsch in 2002 found that the Unified Process (UP) augmented with agile methodologies worked best with “small development teams of three to eight people” [7]. The same study also concluded that projects for an agile development team should be small, lasting between one and four years in duration [7]. Agile methods, especially AM, are meant to produce software quickly and efficiently. Hence, it makes sense that agile methodologies would work best on short projects with small development teams.

Unfortunately, the scarcity of research on AM makes it difficult to ascertain the methodology’s actual successes and failures. Agile Modeling has supporters that ardently argue its benefits, but there appear to be no extensive studies that support these supposed benefits [3,7]. In other words, AM may not have seen any reported failures, but there is a lack of substantial research detailing its successes. It is even more unfortunate that studies tend to incorporate AM incompletely, in modified forms, or the authors treat the methodology as part of another methodology altogether [7]. These inconsistencies in the research literature make it exceptionally difficult to accurately and empirically judge AM’s supposed benefits.

**Strengths and Weaknesses of Agile Modeling**

Agile Modeling has two main strengths: increased efficacy and improved communication. The design philosophy of AM, if fully adopted, forces developers to model more efficiently and effectively [3,5]. There are also the driving principles of excellence and simplicity. These principles have corresponding practices which allow for seemingly complex systems to be represented in relatively simple terms [3]. This simplification of the system potentially helps the team members better understand the stakeholder’s needs and identify potential issues early in the development process. Also, the team does not have to concern itself with maintaining vast amounts of documentation since AM advocates maintaining a minimum amount of documentation [3,5]. The time saved from maintaining lightweight documentation can be put toward making better models or toward the actual implementation of the system.

Improving communication was one of Ambler’s chief concerns when developing AM. The values, principles, and practices of AM interact to improve communication between developers themselves and with the stakeholder [2,3]. The stakeholder’s active involvement in the project opens a line of communication between the developers and the stakeholder, which may help clarify the requirements of the project [3,6]. Agile Modeling also facilitates the development of a sense of camaraderie and collective ownership of the project. The modeling process is a collaborative effort, and the work is displayed publicly to reinforce team cohesion [3,5]. Obviously these collaborative techniques help mitigate potential “finger-pointing” and increase the development team’s efficacy.

Despite having impressive strengths, AM has two potentially damaging weaknesses. First, Ambler readily admits that AM is not a full agile methodology; hence, it is not wholly sufficient [2,3]. In other words, AM has to be adopted in conjunction with another development process. If a developer were willing to adopt a particular base process, then AM would be an *additional* methodology that would cost money and time to adopt. More importantly, Ambler also admits that some teams may find themselves “not in a position to fully adopt AM” [3]. This is particularly damaging to AM because Ambler also proclaims that to properly use AM, it should be adopted in *full* [3]. If a developer did adopt AM, there is no guarantee that it would work in all situations, even when “conditions are perfect” [3]. It would appear that AM is limited in its applications, and Ambler’s “all-or-nothing” philosophy could scare away adopters.

Agile Modeling lacks a significant presence in the research literature. Namely, there is a lack of research evidence supporting AM’s claimed benefits [7]. There are some case studies that have explored AM, but there are only a few empirical reports to support objective investigations. Within these studies and reports, however, AM is often lumped together with a base process, such as the UP [7]. This is sensible since AM is not a full-fledged agile method, but the blurred lines between AM and the base process make it difficult to judge the results of such studies. If we wish to be particularly charitable, we could attribute that the lack of substantial research to the AM’s relative novelty.

**Conclusion**

[Conclusion]

Works Cited

[1] Agile Alliance. “Manifesto for Agile Software Development.” *AgileManfiesto.org*. Agile Alliance, 2001. Web. 14 Jan. 2014.

[2] Abrahamsson, Pekka, Outi Salo, Jussi Ronkainen, and Juhani Warsta. *Agile Software Development: Review and Analysis.* Espoo, Finland: VIT Publications 478, 2002. Web.

[3] Ambler, Scott. *Agile Modeling: Effective Practices for Extreme Programming and the Unified Process*. New York: John Wiley & Sons, 2002. Web.

[4] ---. “Agile Development Best Dealt with in Small Groups: Development Techniques Work Best in Environments that Undergo Continual Change.” *Computing Canada* 28.9 (2002): 9. Web.

[5] Ambler, Scott W. “Values, Principles, and Practices Equal Success: Agile Modeling Extends Extreme Programming Ideas.” *Computing Canada* 27.10 (2001): 11. Web.

[6] ---. “Know the User Before Implementing a System: The First Step is to Identify Project Stakeholders.” *Computing Canada* 28.3 (2002): 13. Web.

[7] Erickson, John, Kalle Lyytinen, and Keng Siau. “Agile Modeling, Agile Software Development, and Extreme Programming: The State of Research.” *Journal of Database Development* 16.4 (2005): 88-100. Web.