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 method introduced by Scott Ambler in his 2002 book *Agile Modeling* [1,3]. Ambler was primarily concerned with developers’ overreliance on prescriptive processes to develop software since these processes often encourage or result in poor development practices. Ambler introduced AM, an agile method intended to maximize developers’ efficacy during the development process, particularly with regard to modeling and documentation [3]. Agile Modeling is a respectable effort at streamlining systems modeling and documentation, but developers should exercise caution when adopting it.

**Agile Modeling and the Agile Manifesto**

Ambler describes AM as a “practice-based methodology for effective modeling and documentation of software-based systems” [3]. Ambler argues that modeling is critical to the success of development projects, and AM aims to maximize the efficacy of this process. It is important to note, however, that AM is not a “complete software process” unto itself [1,3]. Agile Modeling focuses solely on improving modeling and documentation, not programming practices. As such, AM *must* be used with a base software process, such as Extreme Programming (XP), which will improve the modeling process and retain the base process’s benefits [1,3].

The core of AM lies in its values, principles, and practices. Agile Modeling has five values: communication; simplicity; feedback; courage; and, humility. It is worth noting that the first four values are borrowed directly from XP [1]. Improved communication between developers and the stakeholder leads to better, well-informed models. Simplicity aims to keep software developers from being overwhelmed with complex and confusing models. Timely, accurate, and useful feedback can keep a project on time and on budget. Courage is inherent in agile methods since adopting an agile methodology means eschewing the safety nets of prescriptive processes. Humility, Ambler argues, is the hallmark of a great programmer that knows her own limitations and is willing to accept help from others [1,3].

Agile Modeling also features a set of ten core principles. Agile Modeling’s core principles include: an emphasis on software; enabling the future; lightweight documentation; assumed simplicity; embracing change; incremental change; purposeful modeling; making multiple models; quality work; and, maximizing the stakeholder’s investment. The principles work together, directly or indirectly, to produce “high-quality software that meets the needs” of the stakeholder in an “effective manner” [3]. The lightweight and driven nature of AM allows the developers to modify the project without seriously upsetting or arresting development.

Agile Modeling’s values and principles are supplemented by core practices, which 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 appropriately applying artifacts, creating several models in parallel, iterating to other artifacts, and modeling in small increments. Teamwork entails modeling with others, involving the stakeholder, collectively owning the project, and publicly displaying the models. Simplicity is obtained by creating simple content, by simply depicting models, and by using simple tools. Validation, of course, concerns testability and proving models with code [3]. These practices do not occur in discrete phases; rather, these core practices are interdependent.

Agile Modeling’s values, principles, and practices all correspond 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. All of AM’s values work together to accomplish the “working software” value [2,3]. Agile Modeling’s principles and practices also correspond to the Agile Manifesto’s values. For example, AM emphasizes high quality software and lightweight documentation, which correspond to the Agile Manifesto’s “working software over comprehensive documentation” values [2,3]. Essentially, all of AM’s values, principles, and practices combine to address the Agile Manifesto’s values.

Agile Modeling’s values, principles, and practices also work together to address the twelve general principles of agile methods. For example, AM emphasizes delivering working software, embracing change, incremental and iterative development, and communication, which correspond to the first, second, third, and seventh principles of agile methods [2,3]. Agile Modeling’s emphasis on stakeholder involvement corresponds to the fourth and sixth general principles. Also, AM’s teamwork principles and practices address the fifth, eleventh, and twelfth general principles. Agile Modeling advocates traveling light, which should allow developers to “maintain a constant pace indefinitely,” and AM advocates “continuous attention to technical excellence” through high quality work [2,3]. Lastly, AM heavily emphasizes simplicity in its values, principles, and practices, which addresses the simplicity principle of agile methods [2,3].

**Extreme Programming, Team Sizes, and Project Complexity**

It is worth noting that AM is essentially an extension of and an improvement on XP; in fact, Ambler views AM as a “lead-in” to XP [7]. In addition to the four values AM borrows from XP, at least half of AM’s principles are also borrowed directly from XP. Extreme Programming is focused on systems development, and AM is focused on effectively modeling systems. Using AM in conjunction with XP can “best exploit the benefits of XP” while also incorporating the benefits of AM [3,7]. Agile Modeling and XP share a certain philosophical congruency, which lends itself to the methodologies’ compatibility [3].

Agile methods, such as AM, work best with certain team sizes and with certain levels of project complexity. In 2002, Ambler and other agile developers concluded that agile methods “work well for teams of 20 to 30 people,” which should be split into subgroups [4]. A study by Michael Hirsch in 2002 found that the Unified Process (UP) augmented with agile methodologies worked best with “small development teams of three to eight people” and that projects should last between one and four years [7]. Hirsch aside, the scarcity of research on AM makes it difficult to ascertain the method’s actual successes and failures. In fact, studies tend to incorporate AM in modified forms [3,7]. The scarcity and the inconsistency of the research literature makes it exceptionally difficult to objectively judge AM’s supposed benefits.

**Agile Modeling’s Strengths and Weaknesses**

Increased efficacy is AM’s greatest strength. Agile Modeling’s design philosophy forces developers to model more efficiently and effectively by striving for simplicity and quality. Agile Modeling’s principles and practices also allow for complex systems to be represented in simple terms [3,5]. Simplification of the system helps the developers to better understand the project and to better identify issues early in development. Also, the team does not have to maintain unwieldy documentation since AM advocates maintaining lightweight documentation [3,5]. Time saved can be put toward improving models or toward actually implementing the system.

Improved communication is AM’s other main strength. Agile Modeling’s values, principles, and practices facilitate improved communication between developers and the stakeholder [1,3]. The stakeholder’s active involvement in the project opens a line of discussion with the developers, which may help to clarify project requirements [3,6]. Agile Modeling also facilitates the development of camaraderie and collective ownership of the project. The modeling process is a collaborative effort, and the team’s work is displayed publicly to reinforce team cohesion [3,5]. These collaborative techniques help mitigate potential “finger-pointing” and ultimately increase the development team’s efficacy.

Despite having impressive strengths, AM has significant weaknesses. First, Ambler readily admits that AM is not a complete agile method; that is, AM is not sufficient [1,3]. In other words, AM *must* be used in conjunction with another development process. If a developer adopts AM, it will cost additional money and time to incorporate AM into the established base process. More importantly, Ambler admits that some teams may find themselves “not in a position to fully adopt AM”; this potentially contradicts Ambler’s assertion that AM should be adopted in full, not piecemeal [3]. Furthermore, there is no guarantee that AM will work in all situations, even when “conditions are perfect” [3]. It would appear, then, that AM is limited in its applications, and Ambler’s “all-or-nothing” philosophy could concern potential adopters.

Agile Modeling also lacks a significant presence in the research literature. Namely, there is a lack of empirical research supporting AM’s claimed benefits [7]. Within the extant studies and reports, AM is often lumped together with a base process, such as the UP [7]. This is reasonable since AM is not a complete agile method, but the lack of distinction makes it difficult to accurately interpret the studies’ results. Furthermore, the lack of empirical research makes it difficult to objectively evaluate AM’s supposed benefits [7].

**Conclusion**

Agile Modeling is a relatively new entry into the agile methods canon. Agile Modeling is not a full agile method, but it does have the potential to bring additional benefits to a development team’s established base process. Ultimately, AM aims to maximize modeling efficacy and to minimize the amount of documentation maintained. While AM’s strengths are noteworthy, developers considering adopting AM would be wise to approach it with caution.

Works Cited

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

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

[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.