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. AM’s core principles include: software is the primary goal; enabling the future is the secondary goal; documentation is light; simplicity is assumed; change is embraced; change occurs in increments; modeling is purposeful; multiple models are made; work is of high quality; and, the stakeholder’s investment is maximized [3]. The intended result of adherence to each principle is the production of “high-quality software that meets the needs” of the project stakeholders in an “effective manner” [3]. Ideally, a development team will maintain a “light” level of documentation to maximize the team’s agility and the effectiveness of responses to change. Simple, purposeful models are easier to understand than complex models and still convey the relevant information appropriately. Of course, complex projects will require multiple models, but each model should be as simple and as purposeful as possible. If these principles are followed, then less time will be spent updating potentially irrelevant or obsolete documents and models, which will allow for an increase in the quality of work produces. Also, the stakeholder may take a more active role in development since less development time is wasted on maintaining outdated documents and models [3]. AM’s lightweight and driven nature allows for stakeholders to request changes, significant or trivial, without upsetting and 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. The goal of iterative and incremental modeling is to develop accurate and well-defined models by using artifacts appropriately within a system, creating multiple models in parallel, iterating to other artifacts—that is, temporarily stopping work on one artifact and moving to another—and modeling in small increments. Teamwork is made more effective when multiple people work on models together, the stakeholders actively participate, the models are collectively owned, and the models are displayed publicly. The focus of teamwork is to build camaraderie between developers and improve communication between developers and the stakeholders. The aim of simplicity is to make content and models as simple as possible while using the simplest tools possible. Finally, validation is a vital practice in that it ensures the correctness of models by testing and by proving models with code [3]. Of course, 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 highly 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 correspond to the Agile Manifesto’s value of “working software” [1,3]. AM’s principles also correspond to the Agile Manifesto’s major and minor values. For example, Agile Manifesto advocates software as the primary goal, assumed simplicity, and traveling light, which correspond to the Agile Manifesto’s “working software over comprehensive documentation” value [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” value [1,3]. As another example, AM places strong emphasis on teamwork in practice, which corresponds directly to the Agile Manifesto’s “individuals and interactions” value [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 advocates communication, primary emphasis on 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]. AM mandates significant stakeholder interaction; in fact, improved communication is an outcome of AM. The stakeholder principles and practices, therefore, correspond to the fourth and sixth principles of agile methods. Similarly, another outcome of AM is the improved communication and camaraderie between developers. The corresponding teamwork principles and practices address the fifth, eleventh, and twelfth principle of agile methods [1,3]. AM advocates traveling light, which reduces the amount of time wasted on obsolete assets. Hence, developers following the AM methodology should be able to “maintain a constant pace indefinitely,” unless a significant hurdle is encountered [1,3]. AM 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, it 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 selected to address the twelve principles of agile methods.

It is worth noting that AM is essentially an improvement on and extension of XP. In fact, Ambler views AM as a “lead-in” to XP [7]. XP is focused on systems development, and AM is focused on effectively and efficiently developing systems models. Hence, the two methodologies appear to be a perfect match. 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 adopted nine of XP’s eighteen principles, in addition to four values adopted. It would seem that AM and XP share a certain philosophical congruency that lends itself to fully compatible or at least nearly compatible uses of the methodologies [3]. AM is heavily derived from XP, and, if Ambler is to be believed, its adoption can only benefit the users of XP.

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

Agile methodologies, 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”; however, a large group should be split into smaller subgroups [4]. A study by Hirsch in 2002 had similar results. The study found that the UP augmented with agile methodologies worked best with “small development teams of three to eight people” [7]. The same study also concluded that the ideal project for an agile development team should be small, lasting between one and four years in duration [7]. Although the Hirsch study did not employ AM, it would not be unreasonable to think AM would have similar results. Agile methodologies, 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. AM has supporters that ardently argue its successes and 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 the supposed benefits offered by AM.

**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 to strive for excellence and simplicity. These principles have corresponding practices which allow for seemingly complex systems to be represented in relatively simple terms [3]. This is accomplished by making multiple models, making the models as simple as possible, and using the appropriate models for the appropriate tasks. This simplification of the system potentially helps the team members better understand the stakeholder’s needs. Also, the team does not have to concern itself with maintaining vast amounts of documentation since AM advocates maintaining the most minimum amount of documentation possible [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 the AM methodology. The values, principles, and practices of AM interact in such a way to improve communication between developers themselves and with the stakeholder [2,3]. The stakeholder’s active role in the project development lifecycle forces the developers to form a relationship with the stakeholder. This active relationship opens the line of communication between the developers and the stakeholder. As the developers and the stakeholder maintain a constant line of communication, the requirements of the project are potentially made much clearer [3,6]. A better understanding of the project requirements and of the stakeholder’s needs allows the developers save time that could be potentially wasted during costly maintenance late in the development lifecycle. Apart from the stakeholder, AM also leads the development team to develop a sense of camaraderie and collective ownership over the project. The modeling process is a collaborative effort, and the work is displayed publicly for all to see [3,5]. Obviously this collaborative technique helps mitigate potential “finger-pointing” and builds a sense of cohesion between the developers.

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, preferably agile, development process. If a developer were willing to adopt a particular process—for example, XP—then AM would be an *additional* methodology the developer would have to spend money and time adopting. 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 the AM methodology, it has to be adopted in *full* [3]. If a developer did adopt AM, there is no guarantee that it would work in certain situations, even when “conditions are perfect” [3]. It would appear, then, that AM is limited in its application, and its “all-or-nothing” philosophy may scare away potential adopters.

Perhaps less damaging, AM lacks a significant presence in the research literature. Namely, there is a lack of research evidence supporting the claimed benefits of AM [7]. That is not to say, however, that AM has no presence in the literature. There are some case studies that have explored AM, but there are only a few empirical reports to support objective investigations [7]. Within these studies and reports, however, AM is often lumped together with a base process, such as the UP [7]. This makes sense because AM is not a full-fledged agile methodology, but the blurred lines between AM and the base process detract from the legitimacy of AM. If we wish to be particularly charitable, we could attribute that the lack of substantial research supporting AM to the methodology’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.