Agile Modeling: A Brief Overview

Scott W. Ambler

President, Ronin International
scott.ambler@ronin-intl.com

Abstract: Agile Modeling (AM) is a practice-based methodology for effective modeling of software-based systems. Where the Unified Modeling Language (UML) defines a subset of the modeling techniques that software professionals require, AM defines practices that enables developers to model in an efficient and effective manner. This paper provides a brief overview of AM's values, principles, and practices; defined what agile models are; and summarizes the scope of AM.

Agile Modeling (AM) [Am01b; Am02] is a practice-based methodology for effective modeling of software-based systems. The AM methodology is a collection of practices - guided by principles and values - that are meant to be applied by software professionals on a day-to-day basis. AM is not a prescriptive process, in other words it does not define detailed procedures for how to create a given type of model, instead it provides advice for how to be effective as a modeler. AM is "touchy-feely" in that it is not hard and fast - think of AM as an art, not a science.

1 Why AM?

Why do we want to be effective at modeling? Because modeling is an important part of any software process. Agile software processes such as eXtreme Programming (XP) [Be00] and Dynamic Systems Development Method (DSDM) [St97] include modeling activities. Yes, even XP includes modeling techniques such as user stories, Class Responsibility Collaborator (CRC) models [Be00; Am01c], and sketches. Contrary to what XP's detractors will tell you XP does not abandon modeling, instead it minimizes modeling efforts by taking a test-first approach to design in which you develop your tests before you develop your code. This forces you to think through how you will build your software before you actually build it, exactly as traditional design modeling does. XP fulfills some of the goals of modeling, understanding what it is you're building, in different ways and therefore requires less modeling - there is absolutely nothing wrong with that. Prescriptive software processes also include modeling activities, in the case of the Unified Process [JBR99] three of the six core process workflows focus on modeling and my own OOSP has a project stage simply called "Model".

You typically model for one of two purposes - either to understand what it is you are building or to aid your communication efforts. You may choose to model the requirements of your system, perhaps with a use case diagram or a collection of business rule definitions. Similarly, you may choose to develop models to analyze those requirements, to formulate a high-level architecture for your system, or a detailed design for it.

In each of these cases your goal is to gain a better understanding of one or more aspects of your system, in other words you use models to help you to explore what it is you are working on. Furthermore, you may use models to communicate within your team or with individuals or groups external to your team. A data model helps to communicate the structure of your database to people writing Java source code that interacts with that database. A user interface flow diagram communicates the overall structure of your system's user interface to the people working on individual screens, web pages, or reports. An activity diagram communicates the business processes that your system proposes to support to the project stakeholders providing funding to your project team. In short, modeling is critical to your project team's success, but how do you model in an effective and agile manner? That is the fundamental question that AM addresses.

The AM methodology fulfills three fundamental goals. First, it defines and shows how to put into practice a collection of values, principles, and practices pertaining to effective, light-weight modeling. The secret of AM isn't the modeling techniques themselves - such as use case models, class models, data models, or user interface models -but how they are applied. Second, AM addresses the issue of how to apply modeling techniques on software projects taking an agile approach such as XP. Sometimes it is significantly more productive for a developer to draw some bubbles and lines to think through an idea, or to compare several different approaches to solving a problem, than it is simply start hacking out code. Third, AM addresses how you can improve your modeling activities following a heavy-weight approach to software development, and in particular project teams that have adopted an instantiation of the Unified Process such as the Rational Unified Process (RUP) [Kr00] or the Enterprise Unified Process (EUP) [Am01a]. Although you must be following an agile software process to truly be agile modeling, more on this in a moment, you may still adopt and benefit from many of AM's practices on non-agile projects. This is similar to non-XP teams benefiting from adoption of some of its practices such as pair programming or refactoring - they aren't truly doing XP but have still improved their productivity by adopting a portion of it.

2 AM Values

The values of AM includes those of XP - communication, simplicity, feedback, and courage - and extends it with humility. It is critical to have effective communication within your development team as well as with and between all project stakeholders. You should strive to develop the simplest solution possible that meets all of your needs and to obtain feedback regarding your efforts often and early. Furthermore you should have the courage to make and stick to your decisions, and to have the humility to admit that you may not know everything, that others have value to add to your project efforts.

3 AM Principles

The principles of AM include the importance of assuming simplicity when you are modeling and embracing change as you are working because requirements do in fact change

over time. You should recognize that incremental change of your system over time enables agility and that you should strive to obtain rapid feedback on your work to ensure that it accurately reflects the needs of your project stakeholders. Agile modelers realize that software is your primary goal, although they balance this with the recognition that enabling the next effort is your secondary goal. You should model with a purpose, if you don't know why you are working on something then you shouldn't be doing so, and that you need multiple models in your development toolkit to be effective. A critical concept is that models are not necessarily documents, a realization that enables you travel light by discarding most of your models once they have fulfilled their purpose. Agile modelers believe that content is more important than representation, that there are many ways you can model the same concept yet still get it right. To be an effective modeler you need to know your models. To be an effective teammate you should realize that everyone can learn from everyone else, you should work with people's instincts, and that open and honest communication is often the best policy to follow to ensure effective teamwork. Finally, a focus on quality work is important because nobody likes to produce sloppy work and that local adaptation of AM to meet the exact needs of your environment is important.

4 AM Practices

To model in an agile manner you will apply AM's practices appropriately. Fundamental practices include creating several models in parallel, applying the right artifact(s) for the situation, and iterating to another artifact to continue moving forward at a steady pace. Modeling in small increments, and not attempting to create the magical "all encompassing model" from your ivory tower, is also fundamental to your success as an agile modeler. Because models are only abstract representations of software, abstractions that may not be accurate, you should strive to prove it with code to show that your ideas actually work in practice and not just in theory. Active stakeholder participation is critical to the success of your modeling efforts because your project stakeholders know what they want and can provide you with the feedback that you require. There are two fundamental reasons why you create models, either you model to understand an issue (such as how to design part of your system) or you model to communicate what your team is doing (or has done). The principle of assume simplicity is a supported by the practices of creating simple content by focusing only on the aspects that you need to model and not attempting to creating a highly detailed modeling, depicting models simply via use of simple notations, and using the simplest tools to create your models. You travel light by discarding temporary models and updating models only when it hurts. Communication is enabled by displaying models publicly, either on a wall or internal web site, through collective ownership of your project artifacts, through applying modeling standards, and by modeling with others. Your development efforts are greatly enhanced when you consider testability, apply patterns gently, and reuse existing artifacts. Because you often need to integrate with other systems, including legacy databases as well as web-based services, you will find that you need to formalize contract models with the owners of those systems.

At its core AM is simply a collection of techniques that reflect the principles and values shared by many experienced software developers. If there is such a thing as agile modeling, then are there also agile models? Yes.

5 When is a Model Agile?

To understand AM you need to understand the difference between a model and an agile model. A model is an abstraction that describes one or more aspects of a problem or a potential solution addressing a problem. Traditionally, models are thought of as zero or more diagrams plus any corresponding documentation. However non-visual artifacts such collections of CRC cards, a textual description of one or more business rules, or the structured English description of a business process are also considered to be models. An agile model is a model that is just barely good enough. But how do you know when a model is good enough? Agile models are good enough when they exhibit the following traits:

(1) They fulfill their purpose and no more.
(2) They are understandable.
(3) They are sufficiently accurate.
(4) They are sufficiently consistent.
(5) They are sufficiently detailed.
(6) They provide positive value.
(7) They are as simple as possible.

6 What Is(n't) Agile Modeling?

I am a firm believer that when you are describing the scope of something, be it a system or in the case of AM a methodology, that you should describe both what it is and what it isn't. The following points describe the scope of AM:

(1) AM is an attitude, not a prescriptive process.
(2) AM is a supplement to existing methods, it is not a complete software development methodology.
(3) AM is a way to work together effectively to meet the needs of project stakeholders.
(4) AM is effective and is about being effective.
(5) AM is something that works in practice, it isn't an academic theory.
(6) AM is not a silver bullet.
(7) AM is for the average developer, but is not a replacement for competent people.
(8) AM is not an attack on documentation.
(9) AM is not an attack on CASE tools.
(10) AM is not for everyone.
(11) AM is complementary to the UML.

7 Conclusion

Agile Modeling (AM) is a new methodology for effective modeling. I invite you to get involved with AM by applying its values, principles, and practices on your next project and sharing your experiences with the rest of the world via the AM mailing list (visit www.agilemodeling.com/feedback.htm for details).

Bibliography

[AB00] Abel, K.; Bibel, U: Formatierungsrichtlinien für Tagungsbände. Format-Verlag, Bonn, 2000.

[ABC01] Abraham, N.; Bibel, U.; Corleone, P.: Formatting Contributions for LNI. In (Glück, H.I. Hrsg.): Proc. 7th Int. Conf. on Formatting of Workshop-Proceedings, New York, 1999. Noah & Sons, San Francisco, 2001; S. 46-53.

[Am01a] Ambler, S.W. Enterprise Unified Process White Paper. www.ronin-intl.com/publications/unifiedProcess.htm. 2001

[Am01b] Ambler, S.W. The Agile Modeling Home Page. www.agilemodeling.com. 2001.

[Am01c] Ambler, S.W. The Object Primer 2nd Edition: The Application Developer's Guide to Object Orientation. New York: Cambridge University Press. www.ambysoft.com/theObjectPrimer.html. 2001.

[Am02] Ambler, S.W. Agile Modeling. New York: John Wiley & Sons Publishing. 2002

[Be00] Beck, K. Extreme Programming Explained - Embrace Change. Reading, MA: Addison Wesley Longman, Inc. 2000.

[JBR99] Jacobson, I., Booch, G., & Rumbaugh, J. The Unified Software Development Process. Reading, MA: Addison Wesley Longman, Inc. 1999

[KR00] Kruchten, P. The Rational Unified Process 2nd Edition: An Introduction. Reading, MA: Addison Wesley Longman, Inc. 2000.

[St97] Stapleton, J. DSDM: Dynamic Systems Development Method. Harlow, England: Addison Wesley.

Biography

Scott W. Ambler is President and a senior consultant of Ronin International, www.ronin-intl.com, a software services consulting firm that specializes in software process mentoring, Agile Modeling (AM), and object/component-based software architecture and development. He is also founder and thought leader of the Agile Modeling (AM) methodology, www.agilemodeling.com. Scott is the author of the books The Object Primer2nd Edition (2001), Building Object Applications That Work (1997), and Process Patterns (1998), and co-author of The Elements of Java Style (2000), all published by Cambridge University Press. He is author of the forthcoming Agile Modeling (Autumn 2001) from John Wiley & Sons. He is also co-editor with Larry Constantine of the Unified Process series from R&D books (2000-2001). Scott is a contributing editor with Software Development magazine (www.sdmagazine.com), a contributor to IBM DeveloperWorks (www.ibm.com/developer), and a columnist with Computing Canada.