Agile Modeling

Iteration Modeling: An Agile Best Practice

At the beginning of each Construction iteration an agile team will typically plan (estimate and schedule) the work that they will do that iteration. To estimate each requirement accurately you must understand the work required to implement it, and this is where modeling comes in. You discuss how you're going to implement each requirement, modeling where appropriate to explore or communicate ideas. This modeling in effect is the analysis and design of the requirements being implemented that iteration. In this article I discuss:

- How Much Iteration Modeling?
- Why Iteration Modeling?
- How Does Iteration Modeling Fit In?

How Much Iteration Modeling?

My experience is that a two-week iteration will have roughly half a day of iteration planning, including modeling, whereas for a four-week iteration this effort will typically take a day. The goal is to accurately plan the work for the iteration, identify the highest-priority work items to be addressed and how you will do so. In other words, to think things through in the short term. The goal isn't to produce a comprehensive Gantt chart, or detailed specifications for the work to be done. Figure 1 depicts the sketch for the design of a student information editing screen. This sketch was made on a whiteboard by the product owner, the person responsible for describing and prioritizing requirements for the team, when asked to describe the details behind the "A student maintains their basic information" user story which is to be implemented this iteration. Notice how the sketch is fairly rough but gets the idea across -- it would have been drawn in a few minutes by the product owner as they explained to the team what they expected the system to do. Based on this information the team can now accurately estimate the work to be done to implement this screen. Previously they had no idea that the product owner expected this screen to list the person's seminar history.

Figure 1. Sketch of a screen to be built during this iteration.

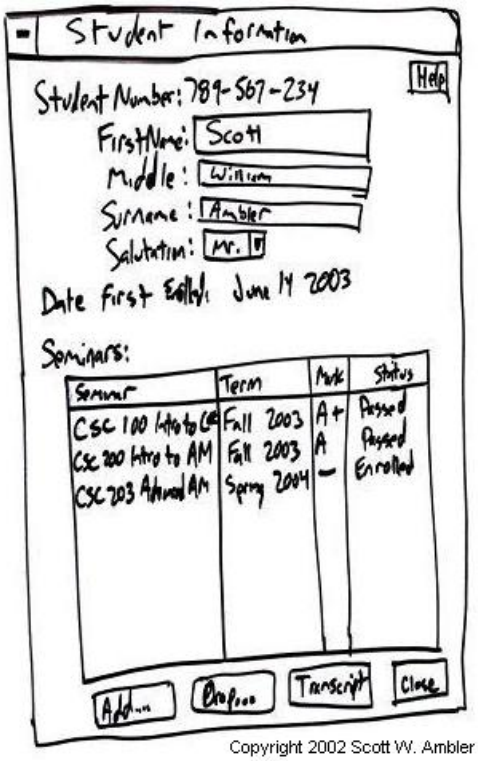


Figure 2 depicts a quick sketch of a physical data model (PDM) sketched out by the architecture owner on the team as he explained the concept of introducing an associative table, Enrollment, to implement the many-to-many association between Student and Course. Although some details are clearly missing, this is enough information required for the architecture owner to explain to the other developers on the team why her estimate was higher that everyone else's for implementing this requirement. Remember, the goal at this point in time is to simply plan the work. If more details are required, as shown in Figure 3, they can be identified through more detailed model storming throughout the iteration and/or in the form of tests via a test-driven development (TDD) approach. The level of detail in Figure 3 isn't required at this point in time -- remember, agile models are just barely good enough (JBGE) for the current situation, and at this point in time you only need enough information to plan the iteration.

Figure 2. Sketch of a physical data model.

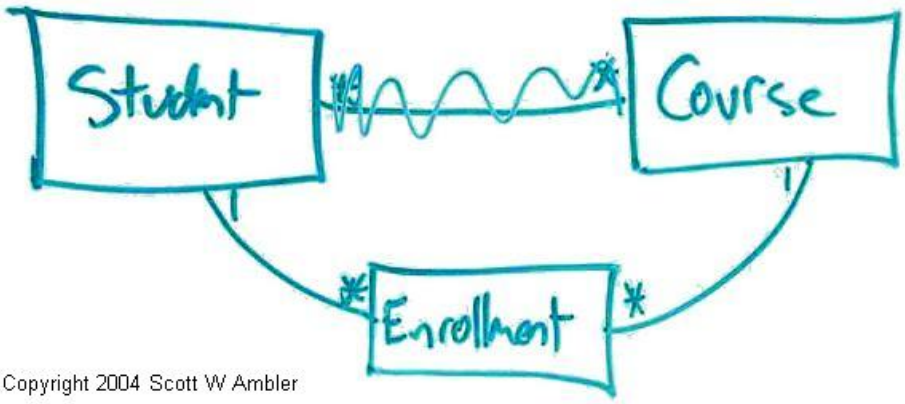
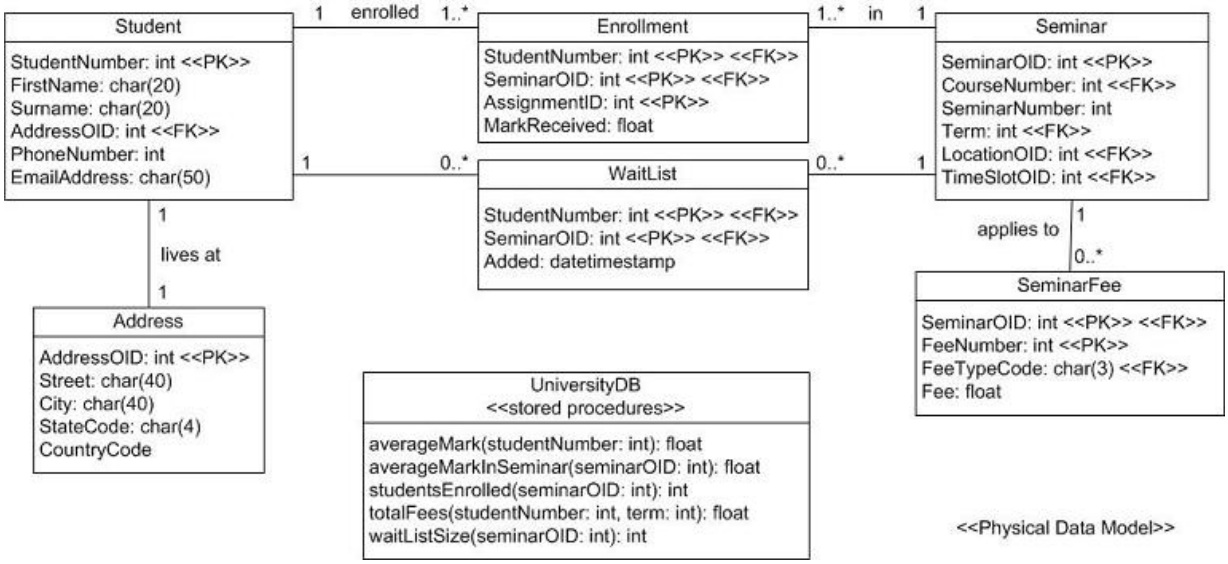


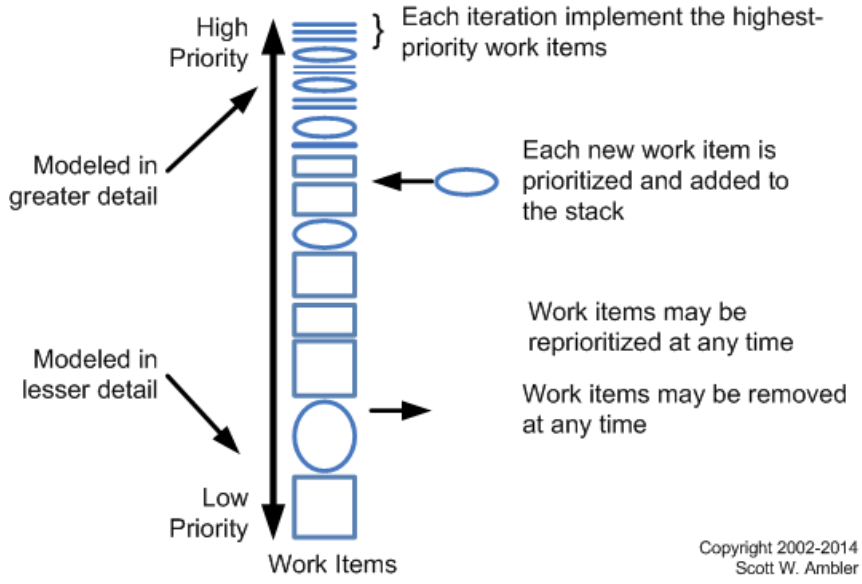
Figure 3. Detailed data model (not created during Iteration Modeling).



Why Iteration Modeling?

An often neglected aspect of Mike Cohn's planning poker is the required modeling activities implied by the technique. Agile teams implement requirements in priority order, see Figure 3, pulling an iteration's worth of work off the top of the stack. To do this successfully you must be able to accurately estimate the work required for each requirement, then based on your previous iteration's velocity (a measure of how much work you accomplished) you pick that much work off the stack. For example, if last iteration you accomplished 15 points worth of work then the assumption is that all things being equal you'll be able to accomplish that much work this iteration. This activity is often referred to as the "planning game" or simply iteration planning.

Figure 4. Agile requirements change management process.



How Does Iteration Modeling Fit In?

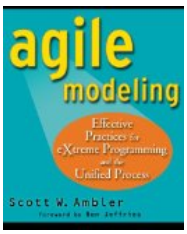
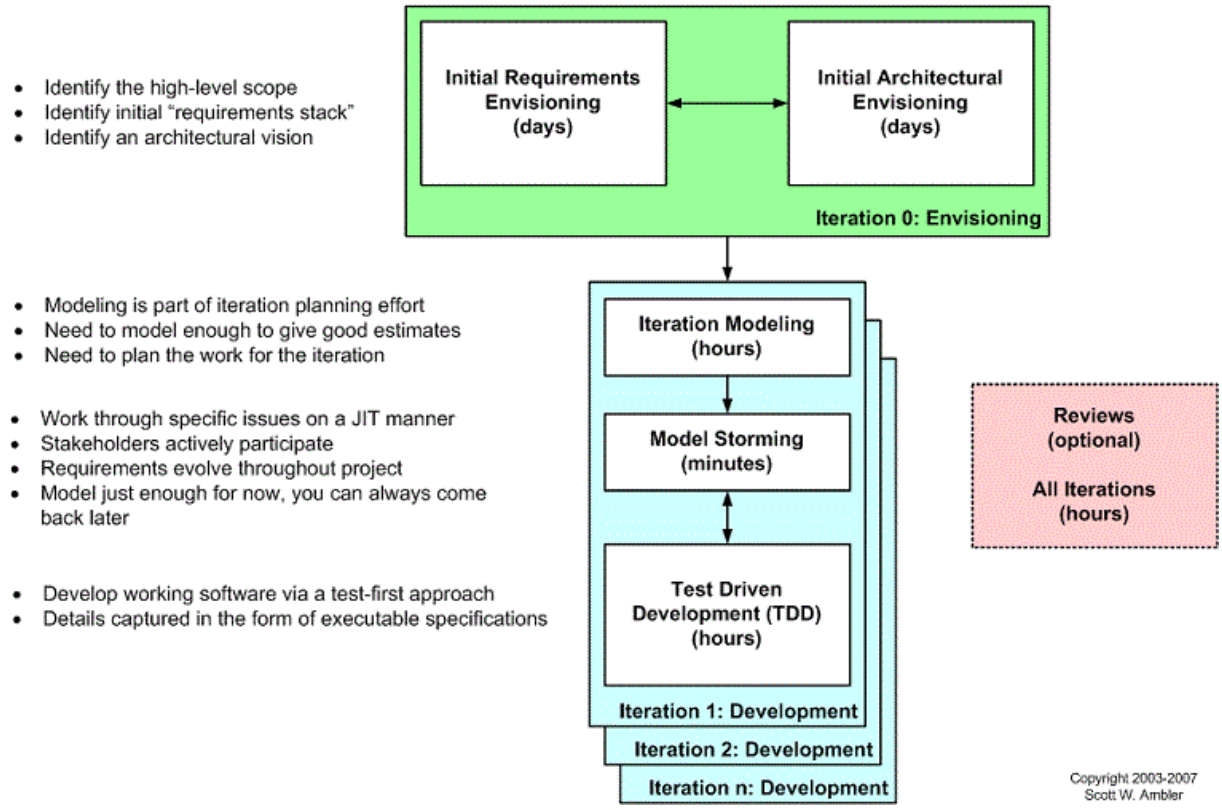


Figure 5 depicts the high-level lifecycle for Agile Model Driven Development (AMDD) for the release of a system. Iteration modeling occurs at the beginning of each iteration as part of the overall iteration planning activities. It is only one of several points in time that you'll model on an agile project: You do some initial requirements and architecture envisioning at the beginning of the project to enable your team to get going in the right direction, you do iteration modeling, and you do just in time (JIT) model storming to explore the detailed requirements and design throughout an iteration.

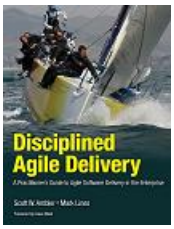
Figure 5. The AMDD lifecycle: Modeling activities throughout the lifecycle of a project.



Let Us Help

We actively work with clients around the world to improve their information technology (IT) practices, typically in the role of mentor/coach, team lead, or trainer. A full description of what we do, and how to contact us, can be found at Scott Ambler + Associates.

Recommended Reading



This book, Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise describes the Disciplined Agile Delivery (DAD) process decision framework. The DAD framework is a people-first, learning-oriented hybrid agile approach to IT solution delivery. It has a risk-value delivery lifecycle, is goal-driven, is enterprise aware, and provides the foundation for scaling agile. This book is particularly important for anyone who wants to understand how agile works from end-to-end within an enterprise setting. Data professionals will find it interesting because it shows how agile modeling and agile database techniques fit into the overall solution delivery process. Enterprise professionals will find it interesting beause it explicitly promotes the idea that disciplined agile teams should be enterprise aware and therefore work closely with enterprise teams. Existing agile developers will find it interesting because it shows how to extend Scrum-based and Kanban-based strategies to provide a coherent, end-to-end streamlined delivery process.

The Object Primer 3rd Edition: Agile Model Driven Development with UML 2 is an important reference book for agile modelers, describing how to develop 35 types of agile models including all 13 UML 2 diagrams. Furthermore, this book describes the fundamental programming and testing techniques for successful agile solution delivery. The book also shows how to move from your agile models to source code, how to succeed at implementation techniques such as refactoring and test-driven development(TDD). The Object Primer also includes a chapter overviewing the critical database development techniques (database refactoring, object/relational mapping, legacy analysis, and database access coding) from my award-winning Agile Database Techniquesbook.