Agile Database Techniques—Effective Strategies for the Agile Software Developer

Agile Database Techniques: Effective Strategies for the Agile Software Developer

by Scott W. Ambler

Table of Contents

Agile Database Techniques—Effective Strategies for the Agile Software Developer
Foreword by Jon Kern
Foreword by Douglas K. Barry
Introduction
Part One - Setting the Foundation
Chapter 1 - The Agile Data Method
Chapter 2 - From Use Cases to Databases — Real-World UML
Chapter 3 - Data Modeling 101
Chapter 4 - Data Normalization
Chapter 5 - Class Normalization
Chapter 6 - Relational Database Technology, Like It or Not
Chapter 7 - The Object-Relational Impedance Mismatch
Chapter 8 - Legacy Databases — Everything You Need to Know But Are Afraid to Deal With
Part Two - Evolutionary Database Development
Chapter 9 - Vive L’olution
Chapter 10 - Agile Model-Driven Development (AMDD)
Chapter 11 - Test-Driven Development (TDD)
Chapter 12 - Database Refactoring
Chapter 13 - Database Encapsulation Strategies
Chapter 14 - Mapping Objects to Relational Databases
Chapter 15 - Performance Tuning
Chapter 16 - Tools for Evolutionary Database Development
Part Three - Practical Data-Oriented Development Techniques
Chapter 17 - Implementing Concurrency Control
Chapter 18 - Finding Objects in Relational Databases
Chapter 19 - Implementing Referential Integrity and Shared Business Logic
Chapter 20 - Implementing Security Access Control
Chapter 21 - Implementing Reports
Chapter 22 - Realistic XML
Part Four - Adopting Agile Database Techniques
Chapter 23 - How You Can Become Agile
Chapter 24 - Bringing Agility into Your Organization
Appendix - Database Refactoring Catalog
References and Suggested Reading

Index
List of Figures
List of Tables
List of Examples

Agile Database Techniques—Effective Strategies for the Agile Software Developer

Scott W. Ambler

Vice President and Publisher: Joseph B. Wikert
Executive Editor: Robert M. Elliott
Development Editor: James H. Russell
Editorial Manager: Kathryn A. Malm
Senior Production Editor: Angela Smith
Text Design & **Composition:** Wiley Composition Services

Published simultaneously in Canada

Library of Congress Cataloging-in-Publication Data:

ISBN: 0-471-20283-5

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

To my parents, Bill and Loreen.
Thanks muchly.

Acknowledgments

I'd like to thank the following for their ideas and, more importantly, for their skepticisms: Dave Astels, Philippe Back, Francois Beauregard, Graham Berrisford, Charles Betz, Chris Britton, Steven Brown, Steve Cohen, Mike Colbert, Warren Cotton, Dale Emery, Neal Fishman, Adam Geras, Steven Gordon, Jason Gorman, Michael M. Gorman, David C. Hay, Michael Haynes, Mats Helander, Daniel Honig, Ron Jeffries, Jon Kern, Jan Emil Larsen, Kevin Light, Floyd Marinescu, Les Munday, John Nalbone, Pan Wei Ng, Paul Oldfield, Oscar Pearce, Chris Roffler, Dave Rooney, John Roth, Adrian Rutter, Yonn M. Samuels, Dwight W. Seeley, Paul Tiseo, Jason P. Tryon, Patrick Vanhuyse, Micheal Vizdos, Michael Vorburger, Sebastian Ware, David Waters, Gary Williams, Dawn M. Wolthuis, and Michael Zimmer.

I'd also like to thank my editors at John Wiley & Sons, Inc., including James H. Russell, Angela Smith, Terri Hudson, Kathryn A. Malm, and Robert M. Elliott.

About the Author

Scott W. Ambler is a senior consultant with Ronin International, Inc. (www._ronin-intl.com), a firm specializing in helping organizations to adopt new software-development techniques. Scott is a senior contributing editor with *Software Development* (www.sdmagazine.com) and the (co)author of numerous books, including *Agile Modeling*, *Mastering EJB,* Second Edition, and *The Object Primer*. In his spare time Scott is an avid photographer and martial artist, studying karate, tai chi, and yoga. Scott lives just north of Toronto, Canada, although he travels all around the world to work with clients (as far as he's concerned the more exotic the locale the better).

Foreword by Jon Kern

“Got database?”* Then get agile!

The agile movement was given shape in the shadows of 11,000-foot peaks of Snowbird, Utah, in February 2001. Ever since, people and pundits alike have been talking and practicing agility. Many development groups, tired of the failed promise of heavyweight processes and death marches towards uncertain goals in uncertain timeframes, are finding comfort in a “human-readable” set of development philosophies and principles.

Scott Ambler has a strong voice in the agile community, founding the Agile Modeling Forum in February 2001. To all who “know” Scott, it is clear that he is passionate about helping development teams succeed for their stakeholders and, ultimately, their customers.

There are many resources that address object modeling, agile techniques, UML (Unified Modeling Language), language-specific intricacies, database design, SQL, and so on. Many good books present information on how to develop a good object-oriented application. Likewise, there are excellent tomes on techniques for developing and tuning databases. It is less likely that you will run across books that describe an evolutionary, agile approach to data-oriented development.

In this book, Scott addresses this key area of application development — the database. He extends the reach of agile techniques across the application team, from developer to database architect, demonstrating that agile techniques are no longer the sole domain of the development folks. The DBA can also apply the same principles to the database — developing incrementally and iteratively, just as developers do with their code base. Now, DBAs will be able to understand the agile methodologies as applied to data-oriented development. They will gain insight and learn how to fit into the larger team, how to leverage their extensive experience with a given DBMS, and how to effectively — and efficiently — support the team’s persistence needs. Even if you work on a small team without “designated” DBAs, this book will be very helpful for its insights into the critical techniques for addressing the common persistence problems facing all development teams.

*With apologies to the National Dairy Council

For those of us who have had the pleasure of introducing development teams to object-oriented methodologies, the data-modeling aspect is *always* an interesting topic. On the one hand, good object models look a lot like well-formed entity-relationship diagrams (if yours don’t, well, get mentored on object modeling!) and many object modelers drive the database design from their class diagrams. On the other hand, some database experts will insist the world revolves around the database (especially where legacy databases have been driving the business for the last decade). Both positions have merit — yet neither is entirely correct. Scott presents a give-and-take, evolutionary methodology that establishes balance within the team. He points out that real-world applications often have more than one option for addressing data concerns throughout the development iterations.

Most modern development methodologies are iterative in nature and require an evolutionary approach. Most hard-core data modelers may be more familiar with a waterfall approach with big, up-front design. This can sometimes cause friction within the team and result in turf

warfare. This book will teach the developer about database basics and teach the DBA the needed skills to be a member of an agile development project. Effectively intertwining agile object development and agile database development can only help teams in their quest for success.

I wish I had a book like this eight years ago when I was developing my first major thin-client, object-oriented application with a data management layer and all the associated other ilks that come along for the ride. *You* will be able to avoid many of the lessons from the school of "hard knocks" by using this book. If you have ever considered "dirty flags," two-phase commits in a distributed environment, or the struggle between "who" is in charge of referential integrity (the database or the objects), then you will benefit greatly from this work. And, if you aren't sure what these terms mean, then you *really* must consider this book!

Because almost every (business) application-development project confronts the need for data storage, this book will be an invaluable resource to most development teams. You'll want to be sure to have enough copies for both your development *and* database folks. Developers enhance their skills by learning about agile database techniques, and DBAs learn how to orient their database development techniques along more agile lines — more effectively supporting the development effort. In short, everybody stands to win. So grab a stimulating cup of something, study up, and then let the collaboration begin!

Jon Kern
Coauthor, *Agile Manifesto*

Foreword by Douglas K. Barry

I want everyone reading this foreword to turn immediately to Chapter 23 of this book. Look at the table containing recommendations on how to become more agile. Do the entries in the table make you feel a little uncomfortable? Good. Do you think these recommendations are unnecessary? Why is that? Take my advice: You really need to do what Scott is suggesting. These are the first steps you can take to improve the chances you will have a successful project. And they may just be very uncomfortable steps to take.

Some mental discomfort is good for people who want to make a change. There is no question that change is needed in how we build (or fail to build) our software systems. This includes our databases. If all we do is what we find to be comfortable, then there is little chance for change.

To me, Scott is suggesting in Chapter 23 that it is good to stand in the other people's shoes for a while. *Really* stand. Talking with other people and expressing empathy for their situation is good, but not good enough. Actually trying to do another person's job is a very different experience.

I know, because I have stood in many people's shoes. Early in my career, I was a data-modeling guru at a large corporation. Then, I got involved in software design and had to deal with other people's database designs. After that, I was the CIO of a startup database company. That was followed by many years in database-related standards development. At

the same time, I started helping people to understand what is likely to be the best architecture for their needs — which is what I am currently doing. Let me tell you, it has been an education, and I have often felt uncomfortable. But I think I am better off for it. Based on my experience, it appears that Scott is showing you a good way to start on your own path.

Throughout this book, Scott includes practical suggestions for using agile techniques in database development. You might not always agree, but it will possibly challenge your thinking. And that is good as well.

Scott also offers common-sense design suggestions for developing a database and for the mapping of data between different types of systems. These suggestions are important, and you do not always find them in the basic modeling texts.

The uncommon suggestions for becoming agile and the common-sense design suggestions make this a good, all-around book for someone looking to go beyond a basic modeling text. You will find workable, real-world advice here.

Douglas K. Barry
Founder and Principal, Barry & Associates, Inc.
(www.barryandassociates.com)

Introduction

An Agile Introduction: This is a really good book. Buy it. Read it. Spread the word.

Since the early 1990s, I've been working with both object and relational database (RDB) technologies to build business applications, and since the mid-1990s I've done a fair bit of writing on the subject. These writings have appeared in *Software Development* (www.sdmagazine.com), in several of my books (in particular *Building Object Applications That Work* and *The Object Primer*), and on my personal Web site (www.ambysoft.com). The two white papers at my site, one on mapping objects to RDBs and the other describing the design of a persistence layer, have proven to be incredibly popular, with several hundred thousand downloads over the years. The persistence layer paper has even been used as the basis for several open source products. Although it's been very rewarding for me to share my ideas through these writings, I never took the time to collect this work in one place, nor have I written everything that I have to say about the topic. This book rectifies this situation.

As a consultant, I've worked with object and data professionals, their related technologies, and of course their techniques. In doing so, I've worked in traditional environments that take a near-serial approach to development as well as more modern environments that take an agile and evolutionary approach to development. Over time, I've worked on many different project teams in various roles. Data-oriented issues were important, and sometimes even critical, to the success of each project. Although traditional project teams seemed to have a handle on how to deal with data issues the more agile ones often struggled — in part because the data professionals in those organizations preferred to take a serial approach and in part because the object developers didn't appreciate the importance of data-oriented issues. Being an

ex-data-specialist (oh no, my horrible secret is out!) and being experienced in object technology, I often found ways for the two groups to work together. My experience was that data professionals were often overly focused on data to the exclusion of the wide variety of challenges faced by object developers and similarly object developers had little or no data-related experience. So, I would help the two groups find ways to work together, to mentor them in each other's techniques, and to help them overcome what is known as the object-relational impedance mismatch. For these two groups to work together effectively, they each need to understand and appreciate what the other group is focused on, and I would even call into question the wisdom of having separate groups to begin with. This book describes the skills that both data professionals and object professionals require in order to build modern-day software.

As a methodologist I have actively tried to find ways to develop software effectively, and over the years have run the gambit from prescriptive approaches such as my work with process patterns (www.ambysoft.com/processPatternsPage.html) and the Enterprise Unified Process (EUP) (www.enterpriseunifiedprocess.info) to agile approaches such as Agile Modeling (AM) (www.agilemodeling.com) and now agile database techniques. In part, this book is an extension of AM to help describe how data professionals can take an evolutionary (iterative and incremental) approach to development. Although many people within the data community are adamantly opposed to evolutionary approaches, interestingly enough I've often found that those opposed to it have never actually tried it; the reality is that agile software development is real and here to stay. For data professionals to remain relevant, they must be prepared to work in an agile manner, otherwise project teams will very likely find ways to work around them (I suspect you see this sort of thing happen within your organization all of the time). My experience, on actual projects, is that you can in fact be very successful by taking an agile approach to data-oriented activities if you choose to do so. Many people will tell you that it won't work, but all they're really saying is that they either can't make it work or they don't want to. This book describes numerous, proven techniques that support evolutionary data-oriented development.

When I first started writing this book, I intended its focus to be on the agile data (AD) method (www.agiledata.org). This method, summarized in Chapter 1, describes how data professionals and application developers can work together effectively on agile projects. It also describes how enterprise professionals, such as enterprise architects and data administrators, can support agile development teams effectively. Because I was taking an iterative and incremental approach to the development of the book, I quickly realized that the real value lay in detailed development techniques instead of yet another methodology. So I refocused.

The Audience for This Book

Who is the audience for this book? The simple answer is anyone who is part of, or at least interacting with, an agile software-development team. The more complicated answer is:

Agile/extreme programmers. Chances are pretty good that the software that you're building manipulates data: therefore, you'll need to adopt many of the techniques described in this book.

Database administrators. This book describes how you can succeed working on an agile software-development team. Read it from cover to cover.

Data administrators. You'll need to support more and more agile development teams as times goes on, and therefore you need to understand how they work and why they work this way. This book will provide the insight that you require to help these teams be effective.

Architects. Agile, evolutionary development is quickly becoming the norm in most organizations. This book describes techniques that you can adopt to work effectively on these teams.

Team leads/coaches/managers. To lead an agile software-development team effectively, you must understand the techniques that your team uses, why they apply those techniques, and the implications of doing so. This book not only describes these techniques but also discusses their trade-offs, enabling you to help your team make intelligent decisions.

Why the Focus on Agile DBAs?

Although most of the skills that I describe in this book are applicable to both application developers and database administrators (DBAs), I choose to present them from the point of view of an agile DBA. An agile DBA focuses on data-oriented issues, including traditional database administration as well as any application development involving data. Agile DBAs will also collaborate with enterprise professionals to ensure that the efforts of the project team reflect enterprise realities. The important thing is that they do this work in an agile manner. The role of agile DBA can be held by several people on your project, can be shared on a rotating basis by several people, or can be held by a single person. Although the skillset of an agile DBA can seem formidable, and it is, you'll find that you can gain these skills over time by working with others who already have skills that you're missing, by training, and by simply trying them out for yourself.

An Overview

This book is organized into four parts. The first part sets the foundation by describing the fundamental skills and philosophies that all IT professionals require to be effective at data-oriented activities. The second part describes techniques that enable evolutionary database development, showing that it is possible to take an iterative and incremental approach to data-oriented development. The third part provides an overview of detailed implementation techniques for effectively using object technology, relational database technology, and XML (Extensible Markup Language) technology together. The fourth part wraps up with a discussion of how to successfully adopt the concepts described in this book.

Part I

A significant problem in the IT industry is that most data books do not cover object-oriented development issues, and most object books seem to ignore data issues. This needs to stop. Part I describes the fundamental skills and knowledge that everyone on an agile project team should have. This includes the basics of object orientation, relational databases, the object-relational impedance mismatch, data modeling, and how to deal with legacy data issues. Without this common base of knowledge it is very difficult for application developers and data professionals to work together effectively.

Part II

Part II focuses on how to take an evolutionary approach to data. This section sets the foundation for a model-driven development (MDD) approach, or more accurately, an agile model-driven development (AMDD) approach where your application code and database schemas are based on agile models. This isn't the only way to work; you may decide to take a test-driven development (TDD) approach instead, or better yet, combine it with AMDD. Both methods support evolutionary development but because MDD is very common within the data community, I suspect that developers will gravitate more towards an AMDD approach rather than a TDD approach. However, some agile developers, particularly extreme programmers, prefer TDD over AMDD. Luckily, the two approaches work very well together, so it really doesn't matter which you choose. The implication is that TDD will become more important to data professionals in the coming years. This part also describes database refactoring, an evolutionary technique that enables you to improve your database design in small steps. In many ways, database refactoring is normalization after the fact. Chapters describing mapping objects to relational databases, performance tuning, database encapsulation, and supporting tools are included in this part because they enable evolutionary development.

Part III

Part III focuses on implementation techniques and strategies such as concurrency control, security access control, finding objects in relational databases, referential integrity, and the effective use of XML. An important observation is that many of these topics are traditionally thought of as data issues, but as you'll see there is far more to them than this — it isn't a black-and-white world.

Part IV

Part IV describes strategies for adopting agile database techniques. This chapter provides advice for individuals who want to become agile software developers and for organizations that want to adopt agile techniques.

Part One: Setting the Foundation

Chapter List

Chapter 1: The Agile Data Method

Chapter 2: From Use Cases to Databases — Real-World UML

Chapter 3: Data Modeling 101

Chapter 4: Data Normalization

Chapter 5: Class Normalization

Chapter 6: Relational Database Technology, Like It or Not

Chapter 7: The Object-Relational Impedance Mismatch

Chapter 8: Legacy Databases — Everything You Need to Know But Are Afraid to Deal With

Part Overview

This part describes fundamental skills and knowledge that everyone on an agile project team should have. Why should you invest your time reading these chapters? Without this common base of knowledge it is very difficult for application developers and data professionals to work together effectively. A significant problem in the IT industry is that most data books do not cover object-oriented development issues and most object books seem to ignore data issues. Furthermore, leading agile books have all but ignored data and enterprise issues until now. I think it's time that we all decide to start investing the time to learn about the wide range of issues that we commonly face on a daily business. Although you may feel that you have a very good understanding of one or more of these topics my advice is to skim the chapters describing your areas of expertise because I suspect I've presented many new insights on these "old topics".

Chapter 1: The Agile Data Method. Explores how application developers, database administrators (DBAs), enterprise architects, and data administrators can work together effectively in an agile environment.

Chapter 2: From Use Cases to Databases — Real-World UML. Object technology is the norm for modern projects; therefore, it is critical for everyone to understand the basics of object orientation and the Unified Modeling Language (UML) 2.*x* (including UML data modeling).

Chapter 3: Data Modeling 101. Data modeling is a fundamental skill that all software professionals, including object professionals, require if they wish to store data effectively.