Preface

I know there is a need for this book. I have been modeling application and enterprise data for 25 years now—the last 15 as a consultant to dozens of organizations. I'm often asked how I conceive software models, and why I construct them a certain way.

Methodology books (including my own prior books) give an answer for beginners. They present concepts, notation, and a simple process. This new book provides the next level of techniques for building models, for those who have mastered the basics. When I build models, my thought processes revolve around patterns. This book provides detailed patterns as a basis for more expertly building data models.

This book focuses on databases and not on programming, because most commercial software is built around a database. The database representation (the data model) sets the scope for the software, determines its flexibility, affects its quality, and influences whether the software succeeds or fails.

This book can help readers avoid mistakes. I know that with tried and tested patterns I could have avoided some of my past modeling mistakes.

This book can help readers build better models. I rifled through past consulting projects and noted the improvements in my recent models. If this book had been available earlier in my career, I could have had better models and built them more quickly.

Who Should Read This Book?

The new book has multiple audiences. It is targeted at practitioners but is also suitable for advanced courses.

- **Application architects**. Application architects help determine the focus of an application and drive that focus into the resulting software. They have to pin down the requirements, understand the requirements, determine what is in and out of scope, and set the

key abstractions in the software. These tasks revolve around models and often data models. The skillful use of patterns is intrinsic to building quality models.

- **Enterprise architects**. These architects reach beyond a single application and address the needs of an entire enterprise. The suite of applications for an enterprise must provide the required business functionality and work well together. There is no better way to harmonize applications than by modeling them in-depth and aligning the models. With models, gaps become apparent and problems can be resolved.
- **Data modelers**. Experienced modelers will find the new book to be a helpful reference for its concise and in-depth advice. Beginning modelers can use the new book as a resource for learning.
- **Database administrators**. In many organizations DBAs do more than just attend to the day-to-day servicing of the database (backups, tuning, authorization, and so forth). Many DBAs create data models and maintain the models as applications evolve.
- **Programmers**. Databases pervade commercial applications. Nevertheless, many programmers have trouble with databases and are tentative about how to apply them. This book can help programmers represent their data, a mind-set that is essential for successful use of databases.
- **Courses**. Many universities offer advanced, special-topics courses as part of their graduate curriculum; this book could be used for such a course. The book is also suitable for commercial data modeling courses.

What You Will Find

My usage of the term *pattern* is different than the literature but consistent with the spirit of past work. I treat pattern as an overarching term encompassing mathematical templates, antipatterns, archetypes, identity, and canonical models.

Part I (Chapters 2–7) concerns ***mathematical templates***, abstract data structures that lie at the core of many application models. I use a deliberate style in presenting each template. First I present UML and IDEF1X diagrams. Then I show representative queries, followed by tables populated with sample data. Finally I present one or more examples using the template. I explain the trade-offs for alternative templates. The discussion of each template is mostly self contained; once you find an appropriate template you can understand it by reading only a few pages.

Part II (Chapters 8–9) provides another perspective with ***antipatterns***, characterizations of common software flaws. When developers find an antipattern, they should substitute the correction. The literature has antipatterns for programming code, but antipatterns also apply to data models.

Part III (Chapter 10) covers ***archetypes***, deep concepts that are prominent and cut across applications. Developers should keep these concepts in mind as they construct models. The use of an archetype can lead to a conceptual breakthrough. By necessity, my list is arbitrary

and incomplete. The models and explanation are also incomplete, so readers will need to add detail as they include these concepts.

Part IV (Chapter 11) focuses on ***identity***, that property of an entity that distinguishes each entity from all others. Identity is a prominent concern in databases because developers must have some way for finding and referring to things. This chapter emphasizes conceptual aspects of identity and minimizes discussion of implementation.

Part V (Chapters 12–15) discusses several ***canonical models***, submodels that often arise during application modeling.

Part VI (Chapter 16) covers ***relational database design***, how to take a UML model, prepare a corresponding IDEF1X model, and then finally create SQL code.

Appendices A and B explain the UML and IDEF1X notations. Appendix C collects major concepts from throughout the book and defines them.

Comparison with Other Books

Several existing books claim to cover data modeling patterns. They are informative books, but their emphasis is on seed models and not patterns. A ***seed model*** is a starter model that is specific to a particular application. In contrast, I define a ***pattern*** as a model fragment that transcends individual applications. Some of the books have latent patterns, but they are implicit and mingled with application content. All of the authors cited below use a different notation.

- Martin Fowler. *Analysis Patterns: Reusable Object Models*. Boston, Massachusetts: Addison-Wesley, 1997.

  Fowler's work is closest in spirit to this book. His is an excellent book but really does not discuss patterns. Instead it presents seed models with occasional commentary on the underlying patterns. Fowler presents models for various applications and evolves the models as he gradually complicates the requirements. Oddly, Fowler uses a database-oriented notation with object-oriented jargon.

- David C. Hay. *Data Model Patterns: Conventions of Thought*. New York: Dorset House, 1996.

  This is another excellent book, but it does not cover true patterns. Hay presents seed models for many applications. Hay uses a database notation that is less concise than the UML.

- Len Silverston. *The Data Model Resource Book, Volumes 1 and 2*. New York: Wiley, 2001.

  This book is similar in style to Hay's book, but covers a wider variety of applications. Silverston uses Richard Barker's notation.

- Len Silverston and Paul Agnew. *The Data Model Resource Book, Volume 3*. New York: Wiley, 2009.

Volume 3 is more abstract than Silverston's prior two volumes and has some deep insights about patterns. This book is most pertinent to Chapter 10 on archetypes. Volume 3 continues to use Richard Barker's notation.

- Jim Arlow and Ila Neustadt. *Enterprise Patterns and MDA: Building Better Software with Archetype Patterns and UML*. Boston, Massachusetts: Addison-Wesley, 2004.

  Most of Arlow and Neustadt's book discusses seed models for several application domains. They use the UML notation, but include programming-oriented aspects that I omit.

Acknowledgments

I would like to thank the following reviewers of this book for their thoughtful and helpful comments: Paul Brown, Donna Burbank, Peter Chang, Alex Ebel, Chary Gottumukkala, Jack Hilty, Bill Huth, Steve Johnson, Patti Lee, Rod Sprattling, Joseph R. Stephen, Toby Teorey, Sam Wegner, Rui Xu, and Roberto Zicari.

In particular I would like to thank Paul Brown for his deep insights. He caught several errors, had many suggestions, and caused me to redouble my efforts to explain the patterns clearly.

Of course, a book like this is not written in a vacuum. I also thank the many colleagues and companies with whom I've worked and interacted over the years.

I used several tools in the writing of this book. Specifically, I used Enterprise Architect to create the UML models and then rekeyed them with the Framemaker desktop publishing software for a precise layout. I used ERwin CE to create IDEF1X models and also typeset them with Framemaker. I tested the SQL code with Microsoft's SQL Server.

Michael Blaha
Chesterfield, Missouri, USA
blaha@computer.org