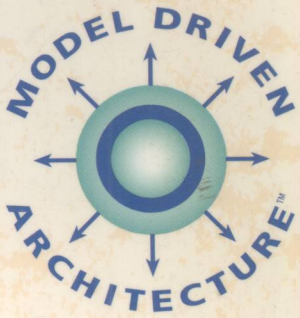
Enterprise Patterns and MDA

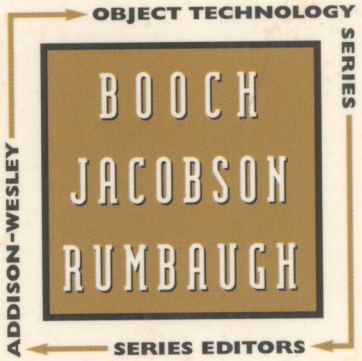
Building Better Software with Archetype Patterns and UML

JIM ARLOW

ILA NEUSTADT

Foreword by Richard Mark Soley, Ph.D.





JIM ARLOW AND ILA NEUSTADT

Enterprise Patterns and MDA

Building Better Software with Archetype Patterns and UML

Addison-Wesley

Boston • San Francisco • New York • Toronto • Montreal
London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Library of Congress Cataloging-in-Publication Data

Arlow, Jim, 1960–
Enterprise patterns and MDA : building better software with archetype patterns and UML / Jim Arlow, Ila Neustadt.
p. cm.
Includes bibliographical references and index.
ISBN 0-321-11230-X (pbk.)
1. Computer software—Development. 2. UML (Computer science) 3. Software patterns. 4. Object-Oriented programming (Computer science) 5. Business enterprises—Data processing. I. Neustadt, Ila. II. Title.

QA76.76.D47A735 2003
005.1'17—dc22

2003062962

ISBN 0-321-11230-X
Text printed on recycled paper
1 2 3 4 5 6 7 8 9 10—CRS—0706050403
First printing, December 2003

Praise for *Enterprise Patterns and MDA*

"The burgeoning field of Model Driven Architecture tools and worldwide support for the Unified Modeling Language are finally being met with high-quality books that explain standard modeling techniques in a way any developer can follow. This book meets an urgent need squarely and clearly, and explains with copious examples a powerful approach to building usable (and reusable!) assets and applications. Every enterprise developer needs this book."

—Richard Mark Soley, Ph.D.
Chairman & CEO
Object Management Group, Inc.

"I've never seen a system of business patterns as detailed as this one. The completeness that Arlow and Neustadt provide in these patterns is impressive. The explanations for why the patterns are formed the way they are and how they're interconnected are incredibly thorough. The patterns presented here have the potential to impact business applications in the same way the 'Gang of Four' patterns have impacted general software development."

—Steve Vinoski
Chief Engineer of Product Innovation
IONA Technologies

"[*Enterprise Patterns and MDA* is a] detailed, yet very readable, guide to designing business applications using reusable model components and Model Driven Architecture. It deserves a place on every application designer's desk."

—Andrew Watson
Vice President and Technical Director
Object Management Group, Inc.

"Design patterns are generally acknowledged as an effective approach to developing robust and highly reusable software. Now that Model Driven Architecture is raising software design to ever-higher levels of abstraction, it is only natural that pattern concepts should find application in advanced modeling techniques. With this book, Arlow and Neustadt have greatly advanced the state of the art of MDA by defining both a theory and a methodology for applying the concept of Archetype Patterns to business software modeling."

—John Poole
Distinguished Software Engineer
Hyperion Solutions Corporation

Enterprise Patterns and MDA

The Addison-Wesley Object Technology Series

Grady Booch, Ivar Jacobson, and James Rumbaugh, Series Editors

For more information, check out the series web site at www.awprofessional.com/otseries.

Ahmed/Umrysh, *Developing Enterprise Java Applications with J2EE™ and UML*

Arlow/Neustadt, *UML and the Unified Process: Practical Object-Oriented Analysis and Design*

Armour/Miller, *Advanced Use Case Modeling: Software Systems*

Bellin/Simone, *The CRC Card Book*

Binder, *Testing Object-Oriented Systems: Models, Patterns, and Tools*

Bittner/Spence, *Use Case Modeling*

Booch, *Object Solutions: Managing the Object-Oriented Project*

Booch, *Object-Oriented Analysis and Design with Applications, 2E*

Booch/Bryan, *Software Engineering with ADA, 3E*

Booch/Rumbaugh/Jacobson, *The Unified Modeling Language User Guide*

Box/Brown/Ewald/Sells, *Effective COM: 50 Ways to Improve Your COM and MTS-based Applications*

Carlson, *Modeling XML Applications with UML: Practical e-Business Applications*

Cockburn, *Surviving Object-Oriented Projects: A Manager's Guide*

Collins, *Designing Object-Oriented User Interfaces*

Conallen, *Building Web Applications with UML, 2E*

D'Souza/Wills, *Objects, Components, and Frameworks with UML: The Catalysis Approach*

Douglass, *Doing Hard Time: Developing Real-Time Systems with UML, Objects, Frameworks, and Patterns*

Douglass, *Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems*

Douglass, *Real-Time UML, 2E: Developing Efficient Objects for Embedded Systems*

Eeles/Houston/Kozaczynski, *Building J2EE™ Applications with the Rational Unified Process*

Fontoura/Pree/Rumpe, *The UML Profile for Framework Architectures*

Fowler, *Analysis Patterns: Reusable Object Models*

Fowler et al., *Refactoring: Improving the Design of Existing Code*

Fowler, *UML Distilled, 3E: A Brief Guide to the Standard Object Modeling Language*

Gomaa, *Designing Concurrent, Distributed, and Real-Time Applications with UML*

Graham, *Object-Oriented Methods, 3E: Principles and Practice*

Heinckiens, *Building Scalable Database Applications: Object-Oriented Design, Architectures, and Implementations*

Hofmeister/Nord/Dilip, *Applied Software Architecture*

Jacobson/Booch/Rumbaugh, *The Unified Software Development Process*

Jordan, *C++ Object Databases: Programming with the ODMG Standard*

Kleppe/Warmer/Bast, *MDA Explained: The Model Driven Architecture™: Practice and Promise*

Kroll/Kruchten, *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP*

Kruchten, *The Rational Unified Process, An Introduction, 2E*

Lau, *The Art of Objects: Object-Oriented Design and Architecture*

Leffingwell/Widrig, *Managing Software Requirements, 2E: A Use Case Approach*

Marshall, *Enterprise Modeling with UML: Designing Successful Software through Business Analysis*

Manassis, *Practical Software Engineering: Analysis and Design for the .NET Platform*

McGregor/Sykes, *A Practical Guide to Testing Object-Oriented Software*

Mellor/Balcer, *Executable UML: A Foundation for Model-Driven Architecture*

Naiburg/Maksimchuk, *UML for Database Design*

Oestereich, *Developing Software with UML: Object-Oriented Analysis and Design in Practice, 2E*

Page-Jones, *Fundamentals of Object-Oriented Design in UML*

Pohl, *Object-Oriented Programming Using C++, 2E*

Quatrani, *Visual Modeling with Rational Rose 2002 and UML*

Rector/Sells, *ATL Internals*

Reed, *Developing Applications with Visual Basic and UML*

Rosenberg/Scott, *Applying Use Case Driven Object Modeling with UML: An Annotated e-Commerce Example*

Rosenberg/Scott, *Use Case Driven Object Modeling with UML: A Practical Approach*

Royce, *Software Project Management: A Unified Framework*

Rumbaugh/Jacobson/Booch, *The Unified Modeling Language Reference Manual*

Schneider/Winters, *Applying Use Cases, 2E: A Practical Guide*

Shan/Earle, *Enterprise Computing with Objects: From Client/Server Environments to the Internet*

Smith/Williams, *Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software*

Stevens/Pooley, *Using UML, Updated Edition: Software Engineering with Objects and Components*

Unhelkar, *Process Quality Assurance for UML-Based Projects*

van Harmelen, *Object Modeling: Designing Interactive Systems*

Wake, *Refactoring Workbook*

Warmer/Kleppe, *The Object Constraint Language, Second Edition: Getting Your Models Ready for MDA*

White, *Software Configuration Management Strategies and Rational ClearCase®: A Practical Introduction*

The Component Software Series

Clemens Szyperski, Series Editor

For more information, check out the series web site at www.awprofessional.com/csseries.

Allen, *Realizing eBusiness with Components*

Apperly et al., *Service- and Component-based Development: Using the Select Perspective™ and UML*

Atkinson et al., *Component-Based Product Line Engineering with UML*

Cheesman/Daniels, *UML Components: A Simple Process for Specifying Component-Based Software*

Szyperski, *Component Software, 2E: Beyond Object-Oriented Programming*

Whitehead, *Component-Based Development: Principles and Planning for Business Systems*

To our family

Foreword

Not 200 meters from where I sit, there was a revolution.

Responding to a call from a famous night rider, a handful of men turned out in the middle of the night to protect their families, their lands, and a set of "human rights" they had in fact just discovered. Their intolerable treatment by a distant tyrant caused them to risk everything—their very lives—to protect their way of life. Some paid the ultimate price that morning, sending a message heard around the world, becoming the beacon of revolution that was to reverberate throughout the Age of Enlightenment down to today.

They wanted nothing less than to create a better society, a just society, different than what they saw in the distant tyrant's domain. They wanted to change everything, wipe the slate clean, and dissolve the political bonds that bound them to their past. Leaning on the previous hundred years of political philosophy, they believed themselves to have what would soon be termed "certain inalienable rights," and they meant to assert those rights.

The result was years of painful fighting, a trans-oceanic war between the world's greatest superpower and a band of rebels, led by a figure termed a "traitor" by the ranks of that superpower. Against all odds, the rebels won the war in a mere eight years, winning a place in history and control of their own destinies.

Yet . . . more than 200 years later, here I sit in that same country drinking a cup of coffee in my local coffeehouse, where stood 200 years ago another coffeehouse. Though this country and its former tyrannical, imperial owner have been separated for more than two centuries, we still share the same language. The legal systems are nearly identical; the political systems, while different, have strong and clear similarities. The cultures are closely related, closely enough to share the same sources of entertainment. In fact, these one-time enemies are considered to have a "special relationship" that transcends all other

diplomatic relationships, even to the exclusion of the closest neighbors and trading partners of both countries.

The message is clear: sometimes at least, revolutions are evolutionary. The urge to reinvent, to clear the decks and start again, quite often instead reifies an extant system—in this case, personal freedom in the context of a precedent-based legal system structured around a government of the people, for the people, and by the people.

Moving our focus from political history to the structure of scientific and industrial revolutions, we find much the same situation. Thomas Kuhn's view of the world is constantly invoked in the information technology industry, but more often as not, IT "paradigm shifts" are in fact only terminology shifts.

Model Driven Architecture (MDA), the vision of the Object Management Group (OMG) to move software development out of the depths of handicraft up to the heights of engineering, without doubt represents a paradigm shift. By focusing on architecture and encapsulating design "on paper" the way building architects capture blueprints, the OMG aims not only to lower initial software development costs but, more importantly, to decrease the ever-increasing costs of software integration and maintenance (which claim some 90 percent of software lifecycle resources today). At the same time, MDA starts with a graphical language rather than a textual one and forces developers to design before coding (or even instead of coding). Clearly there's a revolution under way.

At the same time, however, MDA represents just another level of abstraction, another level of compilation. The authors of the tome in your hands call MDA a "revolution" akin to the late-twentieth-century move from procedural to object-oriented programming languages; but even that was only another compiler-based level of abstraction (few instruction sets are object-oriented; object-oriented languages must be compiled to those non-OO instruction set architectures). In fact, the MDA revolution is already delivering benefits, without discarding that which came before; that is, it is an evolutionary revolution.

Every revolution, regardless of how well it replaces or expands the existing order, must have a language; political revolutions have their constitutions and declarations, and the MDA revolution focuses on software processes and assets. This book, while it briefly presents a software development process, focuses primarily (and in prescient and clear depth) on filling out a set of patterns to simplify the development of software. This book is, effectively, your dictionary of the new language of MDA, a set of basic blueprints that will accelerate the construction of the building you have in mind. Builders don't all need to reinvent steel I-beams, and software developers don't need to reinvent the product catalog.

In sum, you hold in your hands the keys to an evolutionary revolution, one that is already having its impact on the software development world. I must confess to a personal failing, that I find joy in reading encyclopedias and dictionaries. As I sip my coffee near the site of an evolutionary revolution, it's hard not to enjoy reading the declaration of another.

—Richard Mark Soley, Ph.D.
Lexington, Massachusetts, U.S.A.
October 2003

Introduction

About this book

We have called this book *Enterprise Patterns and MDA—Building Better Software with Archetypes Patterns and UML*. The first part of the title sets the theme for the book as a whole: we aim to provide you with a set of essential patterns for enterprise computing. These are not technical "design patterns"; rather, they are essential business patterns that are found, at least to some degree, in virtually all enterprises. You should find that one or more of these patterns are immediately applicable in software development projects. These patterns are high-value model components that you can easily use in your own UML models. Each pattern provides a solution for understanding and modeling a specific part of a business system. Furthermore, we show you how you can use the emerging discipline of Model Driven Architecture (MDA) to apply these patterns with a high degree of automation.

The second part of the title sets out how we achieve our goal of producing a set of patterns that are useful at the enterprise level. We introduce the new concepts of *archetypes* and *archetype patterns* in order to define a level of abstraction optimized for reuse *and* to the automation possibilities of MDA. These patterns are documented using the technique of *literate modeling*, which embeds the patterns, expressed as UML models, in a narrative such that the patterns can be understood, validated, and adapted even by nontechnical readers.

This is a practical book that gives you a useful set of archetype patterns and the theory you need to use them effectively in an enterprise context. We hope that this will save you a great deal of time and effort in your software development projects.

These patterns are valuable—a similar, but much less mature, set of patterns was recently independently valued at about $300,000 by a blue-chip company.

Using any one of these patterns, or even a pattern fragment, may save you many days or months of work. Perhaps even more important than this saving is the fact that the knowledge engineered into each archetype pattern may prevent you from making costly and time-consuming mistakes!

All of the patterns presented in this volume work together harmoniously and so provide a unified pattern language for talking about selling systems. This harmony greatly adds to their value.

At the time of this writing in 2003, we think we are at the start of a revolution in software development. Much as the 1990s saw an increase in the level of abstraction from procedural to object-oriented (OO) code, we believe that this decade will see a further, and more significant, increase in the level of abstraction. This will be a change from code-centric software development to model-centric software development through MDA.

We hope that the concepts, techniques, tools, and patterns that we describe in this book will help us all to make this revolution in software development a reality.

Our vision

One of the reasons this book came about was through boredom! After modeling for many years, we decided that we were often just doing the same old thing over and over again. At the right level of abstraction, most businesses seem to be made up of the same semantic elements—`Customer`, `Product`, `Order`, `Party`, and so on. In fact, so pervasive are some of these elements that it led us to the notion of business archetype patterns.

We speculated that most business systems could be assembled, like Lego bricks, from a sufficiently complete set of archetype patterns.

The essence of our vision is that archetype patterns should be treated as a type of "model component" that can be taken off the shelf, customized, and instantiated in your own models. This process can be done manually, but ideally it should be automated to as high a degree as possible by using an MDA tool.

Today, you can use a GUI builder to create graphical user interfaces rapidly from GUI components. The work we describe in Chapter 2 enables you to construct semantically correct and verifiable UML models rapidly from platform-independent, generic model components with a high degree of automation. We believe that this may be the future of software development. We call this *component-based modeling.*

This is reuse writ large—software systems are not considered to be composed of reusable classes, reusable code components, or even reusable subsystems, but

rather from the reusable semantic elements that are archetype patterns. In fact, to a great extent the essence of the business system lies in its archetypes and their patterns, rather than in any code or design artifacts. Coding practices, design practices, and even architectures come and go with technology changes, but the archetypes themselves survive, largely unchanged, sometimes over millennia.

Why we haven't done it sooner

We have wanted to write this book for several years, but there have been obstacles that we have only recently overcome.

- The state of the art of UML modeling. Until the recent MDA initiative of the Object Management Group (OMG), we did not really have the conceptual tools necessary to describe archetype patterns in good form.
- The problem of pattern variation. Business patterns often need to adapt their forms to a specific business context. We have now formulated a simple solution to this problem that allows us to create archetype patterns that are adaptable to different business environments.
- The problem of communicating UML models to a wide audience. In fact, we've had a good solution to this for a few years now, in the form of literate modeling (described in Chapter 3).
- UML modeling tool support. It's all very well presenting a theory of archetype patterns, but such a theory is useful to the average software engineer *only* if it can be put into practice. Modeling tools have recently come onto the market that can accommodate our requirements for archetype pattern automation.

The structure of this book

There are four main threads to this book:

1. The theory of archetypes and archetype patterns (Chapters 1 and 2)
2. Pattern automation using MDA (Chapter 2)
3. Increasing the business value of UML models by making them accessible to a wide audience through literate modeling (Chapter 3)
4. A valuable pattern catalog that you can use in your own models (Chapter 4 onward)

Chapters 1, 2, and 3 provide you with the theoretical basis for the rest of the book, and you will find that they cover a lot of new material.

In Chapter 1 we describe a new approach to dealing with the problem of pattern variation—how to adapt patterns for different usage contexts.

In Chapter 2 we show you how you can automate the process of using archetype patterns with an MDA-enabled UML modeling tool. The first two chapters are intimately related. The pattern automation described in Chapter 2 depends on the theory of archetypes and archetype patterns presented in Chapter 1.

Chapter 3 describes the technique of literate modeling that you can use to document your patterns. This chapter is pretty much self-contained. Literate modeling is a powerful way to communicate UML models to a wide audience.

Each of the first three chapters contains a summary that reiterates the key information in the chapter in a very concise outline form. This is great for revision and it is also a useful source of bullet points for presentations.

The pattern catalog can stand alone. If you choose to use the book primarily as a pattern catalog (Chapter 4 to Chapter 12), you can skip much of the theoretical background in the first three chapters. Use the pattern catalog as a valuable resource for your own models. Each of the pattern chapters ends with a brief summary that lists the key concepts and archetypes introduced in that chapter. Again, we do this in outline form.

Having said that the pattern catalog can stand alone, we believe that you will be able to apply the patterns much more effectively if you have at least a basic understanding of archetype theory first. You can find all you need to know in Chapter 1. All the patterns in the pattern catalog are a direct result of the application of the theories and techniques described in the first three chapters. The notions of archetypes, archetype patterns, pattern configuration, and literate modeling have allowed us to create much more complete and robust patterns than otherwise would have been possible.

Finally, we provide a glossary of archetypes, a bibliography, and a complete index.

How to use this book

In this section we present roadmaps for the various ways in which you might wish to use this book and some recommendations about how you might like to approach reading it.

Please be aware that this is *not* a beginner's book, so there may be prerequisites for some of the roads that you may want to travel. None of these prerequisites are particularly difficult to achieve, but it's always worth ensuring that you

have what you need, or at least know where to get it, before setting out on the journey!

Table 1

You are	Your goal in reading this book	Prerequisites	Useful references	Roadmap
OO analyst/designer OO programmer Architect	I want to understand the theory of archetypes and archetype patterns	A working knowledge of UML	[Arlow 2001]	Chapter 1
OO analyst/designer OO programmer Architect	I want to see how MDA tools can be used to automate the use of archetype (and other) patterns	A working knowledge of UML (some knowledge of MDA would also be helpful)	[Arlow 2001] [Kleppe 2003]	Chapter 1 Chapter 2
OO analyst/designer Architect	I want to improve my UML models and see how I can make them available to a wider audience	A working knowledge of UML	[Arlow 2001]	Chapter 3
OO analyst/designer Architect	I want to reuse the archetype patterns in my own models in an informal way by taking patterns, pattern fragments, or just ideas	A working knowledge of UML A working knowledge of the business domain in which you intend to apply the patterns	[Arlow 2001]	Pattern catalog
OO analyst/designer Architect	I want to reuse the archetype patterns in my own models in a formal way by understanding the theory behind them	A working knowledge of UML A working knowledge of the business domain in which you intend to apply the patterns	[Arlow 2001]	Chapter 1 Chapter 2 Chapter 3 Pattern catalog
OO analyst/designer OO programmer Architect Business analyst Project manager Software engineer	I want to use the pattern catalog to help me understand a particular business domain	Some knowledge of UML is desirable, but if you don't have this, you should still be able to understand most of the text of the literate models	[Arlow 2001]	Pattern catalog

In Table 1 we (naturally) reference our previous book, *UML and the Unified Process*, as a suitable source for readers who need some introductory material on UML. When we wrote that book, we always had in mind that it could serve as a useful precursor to more advanced texts such as this.

The pattern chapters contain a lot of information, and you may find it helpful to proceed as follows when reading each chapter.

1. Read the chapter's section on business context. As its name suggests, this section provides information that sets the pattern in its context within the business world.
2. Read the chapter's summary section. This will give you a clear idea of exactly what you can find in the chapter.
3. Look at the chapter roadmap. This will give you an overview of the archetypes and their relationships, and where they are discussed in the text.
4. Read the chapter.

Conventions

We have used the following conventions.

Archetypes, pleomorphs, attributes, operations, relationship names, relationship role names, and code fragments are in this font: `AnArchetype`, `APleomorph`, `anAttribute`, `anOperation()`, `aRelationshipName`, `aRoleName`, `some code`.

Archetype definitions look like this:

The `Money` archetype represents an `amount` of a specific `Currency`. This `Currency` is `acceptedIn` one or more `Locales`.

Term definitions look like this:

A business process is a sequence of business activities that, when executed, is designed to lead to some business benefit.

Acknowledgments

We'd like to thank our U.K. editor, Simon Plumtree, for seeing the potential of this project. We'd also like to thank our U.S. editor, Mary O'Brien and her team, Brenda Mulligan, Julie Nahil, Kim Arney Mulcahy, Chrysta Meadowbrooke, Kathy Benn McQueen, and Carol Noble for all their work and support.

Thanks to Dr. Wolfgang Emmerich and John Quinn for their contributions to the work on literate modeling described in Chapter 3. We'd like to acknowledge our friends in iO Software, Richard Hubert, Ronald Steinhau, and Marcus Munzert, without whom Chapter 2 would be considerably shorter. Fabrizio Ferrandina of Zuhlke Engineering has been of invaluable assistance by publicizing our work and introducing us to key contacts.

We have received essential help from John Watkins at IBM and Andy Carmichael at Borland, who provided us with tools to do the job. Andrew Watson of the OMG has been of tremendous assistance in bringing our work to the attention of that body. Liz Dobson of Zuhlke Engineering has always been helpful and enthusiastic, even when we had to turn down consulting opportunities in order to work on the manuscript.

Our reviewers, Richard Soley (OMG), John Poole (Hyperion Software), Steve Vinoski (IONA Technologies) and Fred Waskiewicz (OMG) helped us to improve the quality of the final manuscript, and Richard has written a fine foreword for us.

Once again, we have been fortunate to have had excellent support from Sue and David Epstein. In particular, David helped us with some key advice on writing when this book was still in the planning stage.

Finally, we'd like to thank our cats who, as always, have been an important part of the project. We have solved many modeling problems while stroking Homer. Paddy lies on top of our monitors and nonchalantly indicates modeling errors with paws and tail. Meg generates new ideas by juxtaposing unrelated manuscript pages in novel ways.