Preface

I've been lucky in a lot of ways in my life; one of my great strokes of fortune was being in the right place with the right knowledge to write the first edition of this book in 1997. Back then, the chaotic world of object-oriented (OO) modeling was just beginning to unify under the Unified Modeling Language (UML). Since then, the UML has become the standard for the graphical modeling of software, not just for objects. My fortune is that this book has been the most popular book on the UML, selling more than a quarter of a million copies.

Well, that's very nice for me, but should you buy this book?

I like to stress that this is a brief book. It's not intended to give you the details on every facet of the UML, which has grown and grown over the years. My intention is to find that fraction of the UML that is most useful and tell you just that. Although a bigger book gives you more detail, it also takes longer to read. And your time is the biggest investment you'll make in a book. By keeping this book small, I've spent the time selecting the best bits to save you from having to do that selection yourself. (Sadly, being smaller doesn't mean proportionately cheaper; there is a certain fixed cost to producing a quality technical book.)

One reason to have this book is to begin to learn about the UML. Because this is a short book, it will quickly get you up to speed on the essentials of the UML. With that under your belt, you can go into more detail on the UML with the bigger books, such as the *User Guide* [Booch, UML user] or the *Reference Manual* [Rumbaugh, UML Reference].

This book can also act as a handy reference to the most common parts of the UML. Although the book doesn't cover everything, it's a lot lighter to carry around than most other UML books.

It's also an opinionated book. I've been working with objects for a long time now, and I have definite ideas about what works and what doesn't. Any book reflects the opinions of the author, and I don't try to hide mine. So if you're looking for something that has a flavor of objectivity, you might want to try something else.

Although many people have told me that this book is a good introduction to objects, I didn't write it with that in mind. If you are after an introduction to OO design, I suggest Craig Larman's book [Larman].

Many people who are interested in the UML are using tools. This book concentrates on the standard and on conventional usage of the UML and doesn't get into the details of what various tools support. Although the UML did resolve the tower of Babel of pre-UML notations, many annoying differences remain between what tools show and allow when drawing UML diagrams.

I don't say much in this book about Model Driven Architecture (MDA). Although many people consider the two to be the same thing, many developers use the UML without being interested in MDA. If you want to learn more about MDA, I would start with this book to get an overview of the UML first and then move on to a book that's more specific about MDA.

Although the main point of this book is the UML, I've also added bits of other material about techniques, such as CRC cards, that are valuable for OO design. The UML is just a part of what you need to succeed with objects, and I think that it's important to introduce you to some other techniques.

In a brief book like this, it's impossible to go into detail about how the UML relates to source code, particularly as there is no standard way of making that correspondence. However, I do point out common coding techniques for implementing pieces of the UML. My code examples are in Java and C#, as I've found that these languages are usually the most widely understood. Don't assume that I prefer those languages; I've done too much Smalltalk for that!

Why Bother with the UML?

Graphical design notations have been with us for a while. For me, their primary value is in communication and understanding. A good diagram can often help communicate ideas about a design, particularly when you want to avoid a lot of details. Diagrams can also help you understand either a software system or a business process. As part of a team trying to figure out something, diagrams both help understanding and communicate that understanding throughout a team. Although they aren't, at least yet, a replacement for textual programming languages, they are a helpful assistant.

Many people believe that in the future, graphical techniques will play a dominant role in software development. I'm more skeptical of that, but it's certainly useful to have an appreciation of what these notations can and can't do.

Of these graphical notations, the UML's importance comes from its wide use and standardization within the OO development community. The UML has

become not only the dominant graphical notation within the OO world but also a popular technique in non-OO circles.

Structure of the Book

Chapter 1 gives an introduction to the UML: what it is, the different meanings it has to different people, and where it came from.

Chapter 2 talks about software process. Although this is strictly independent of the UML, I think that it's essential to understand process in order to see the context of something like the UML. In particular, it's important to understand the role of iterative development, which has been the underlying approach to process for most of the OO community.

I've organized the rest of the book around the diagram types within the UML. Chapters 3 and 4 discuss the two most useful parts of the UML: class diagrams (core) and sequence diagrams. Even though this book is slim, I believe that you can get the most value out of the UML by using the techniques that I talk about in these chapters. The UML is a large and growing beast, but you don't need all of it.

Chapter 5 goes into detail on the less essential but still useful parts of class diagrams. Chapters 6 through 8 describe three useful diagrams that shed further light on the *structure* of a system: object diagrams, package diagrams, and deployment diagrams.

Chapters 9 through 11 show three further useful *behavioral* techniques: use cases, state diagrams (although officially known as state machine diagrams, they are generally called state diagrams), and activity diagrams. Chapters 12 through 17 are very brief and cover diagrams that are generally less important, so for these, I've only provided a quick example and explanation.

The inside covers summarize the most useful parts of the notation. I've often heard people say that these covers are the most valuable part of the book. You'll probably find it handy to refer to them as you're reading some of the other parts of the book.

Changes for the Third Edition

If you have earlier editions of this book, you're probably wondering what is different and, more important, whether you should buy the new edition.

The primary trigger for the third edition was the appearance of UML 2. UML 2 has added a lot of new stuff, including several new diagram types. Even familiar diagrams have a lot of new notation, such as interaction frames in sequence diagrams. If you want to be aware of what's happened but don't want to wade through the specification (I certainly don't recommend that!), this book should give you a good overview.

I've also taken this opportunity to completely rewrite most of the book, bringing the text and examples up to date. I've incorporated much that I've learned in teaching and using the UML over the past five years. So although the spirit of this ultrathin UML book is intact, most of the words are new.

Over the years, I've worked hard to keep this book as current as is possible. As the UML has gone through its changes, I've done my best to keep pace. This book is based on the UML 2 drafts that were accepted by the relevant committee in June 2003. It's unlikely that further changes will occur between that vote and more formal votes, so I feel that UML 2 is now stable enough for my revision to go into print. I'll post information any further updates on my Web site (**http://martinfowler.com**).

Acknowledgments

Over many years, many people have been part of the success of this book. My first thanks go Carter Shanklin and Kendall Scott. Carter was the editor at Addison-Wesley who suggested this book to me. Kendall Scott helped me put together the first two editions, working over the text and graphics. Between them, they pulled off the impossible in getting the first edition out in an impossibly short time, while keeping up the high quality that people expect from Addison-Wesley. They also kept pushing out changes during the early days of the UML when nothing seemed stable.

Jim Odell has been my mentor and guide for much of the early part of my career. He's also been deeply involved with the technical and personal issues of making opinionated methodologists settle their differences and agree to a common standard. His contribution to this book is both profound and difficult to measure, and I bet it's the same for the UML too.

The UML is a creature of standards, but I'm allergic to standards bodies. So to know what's going on, I need a network of spies who can keep me up to date on all the machinations of the committees. Without these spies, including Conrad Bock, Steve Cook, Cris Kobryn, Jim Odell, Guus Ramackers, and Jim

Rumbaugh, I would be sunk. They've all given me useful tips and answered stupid questions.

Grady Booch, Ivar Jacobson, and Jim Rumbaugh are known as the Three Amigos. Despite the playful jibes I've given them over the years, they have given me much support and encouragement with this book. Never forget that my jabs usually sprout from fond appreciation.

Reviewers are the key to a book's quality, and I learned from Carter that you can never have too many reviewers. The reviewers of the previous editions of this book were Simmi Kochhar Bhargava, Grady Booch, Eric Evans, Tom Hadfield, Ivar Jacobson, Ronald E. Jeffries, Joshua Kerievsky, Helen Klein, Jim Odell, Jim Rumbaugh, and Vivek Salgar.

The third edition also had a fine group of reviewers:

Conrad Bock
Andy Carmichael
Alistair Cockburn
Steve Cook
Luke Hohmann
Pavel Hruby
Jon Kern
Cris Kobryn
Craig Larman
Steve Mellor
Jim Odell
Alan O'Callaghan
Guus Ramackers
Jim Rumbaugh
Tim Seltzer

All these reviewers spent time reading the manuscript, and every one of them found at least one embarrassing howler. My sincere thanks to all of them. Any howlers that remain are entirely my responsibility. I will post an errata sheet to the books section of **martinfowler.com** when I find them.

The core team that designed and wrote the UML specification are Don Baisley, Morgan Björkander, Conrad Bock, Steve Cook, Philippe Desfray, Nathan Dykman, Anders Ek, David Frankel, Eran Gery, Øystein Haugen, Sridhar Iyengar, Cris Kobryn, Birger Møller-Pedersen, James Odell, Gunnar Övergaard, Karin Palmkvist, Guus Ramackers, Jim Rumbaugh, Bran Selic, Thomas Weigert, and Larry Williams. Without them, I would have nothing to write about.

Pavel Hruby developed some excellent Visio templates that I use a lot for UML diagrams; you can get them at **http://phruby.com**.

Many people have contacted me on the Net and in person with suggestions and questions and to point out errors. I haven't been able to keep track of you all, but my thanks are no less sincere.

The people at my favorite technical bookstore, SoftPro in Burlington, Massachusetts, let me spend many hours there looking at their stock to find how people use the UML in practice and fed me good coffee while I was there.

For the third edition, the acquisition editor was Mike Hendrickson. Kim Arney Mulcahy managed the project, as well as did the layout and clean-up of the diagrams. John Fuller, at Addison-Wesley, was the production editor, while Evelyn Pyle and Rebecca Rider helped with the copyediting and proofreading of the book. I thank them all.

Cindy has stayed with me while I persist in writing books. She then plants the proceeds in the garden.

My parents started me off with a good education, from which all else springs.

Martin Fowler
Melrose, Massachusetts
http://martinfowler.com

blank page