Preface

The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. The UML gives you a standard way to write a system's blueprints, covering conceptual things, such as business processes and system functions, as well as concrete things, such as classes written in a specific programming language, database schemas, and reusable software components.

This book teaches you how to use the UML effectively.

Goals

In this book, you will

- Learn what the UML is, what it is not, and why the UML is relevant to the process of developing software-intensive systems
- Master the vocabulary, rules, and idioms of the UML and, in general, learn how to "speak" the language effectively
- Understand how to apply the UML to solve a number of common modeling problems

The user guide provides a reference to the use of specific UML features. However, it is not intended to be a comprehensive reference manual for the UML; that is the focus of another book, *The Unified Modeling Language Reference Manua* l (Rumbaugh, Jacobson, Booch, Addison-Wesley, 1999).

The user guide describes a development process for use with the UML. However, it is not intended to provide a complete reference to that process; that is the focus of yet another book, *The Unified Software Development Process* (Jacobson, Booch, Rumbaugh, Addison-Wesley, 1999).

Finally, this book provides hints and tips for using the UML to solve a number of common modeling problems, but it does not teach you how to model. This is similar to a user guide for a programming language that teaches you how to use the language but does not teach you how to program.

Audience

The UML is applicable to anyone involved in the production, deployment, and maintenance of software. The user guide is primarily directed to members of the development team who create UML models. However, it is also suitable to those who read them, working together to understand, build, test, and release a software-intensive system. Although this encompasses almost every role in a software development organization, the user guide is especially relevant to analysts and end users (who specify the required structure and behavior of a system), architects (who design systems that satisfy those requirements), developers (who turn those architectures into executable code), quality assurance personnel (who verify and validate the system's structure and behavior), librarians (who create and catalogue components), and project and program managers (who generally wrestle with chaos, provide leadership and direction, and orchestrate the resources necessary to deliver a successful system).

The user guide assumes a basic knowledge of object-oriented concepts. Experience in an object-oriented programming language or method is helpful but not required.

How to Use This Book

For the developer approaching the UML for the first time, the user guide is best read linearly. You should pay particular attention to **Chapter 2**, which presents a conceptual model of the UML. All chapters are structured so that each builds upon the content of the previous one, thus lending itself to a linear progression.

For the experienced developer seeking answers to common modeling problems using the UML, this book can be read in any order. You should pay particular attention to the common modeling problems presented in each chapter.

Organization and Special Features

The user guide is organized into seven major sections:

- **Section 1 Getting Started**
- **Section 2 Basic Structural Modeling**
- **Section 3 Advanced Structural Modeling**
- **Section 4 Basic Behavioral Modeling**
- **Section 5 Advanced Behavioral Modeling**
- **Section 6 Architectural Modeling**
- **Section 7 Wrapping Up**

The user guide contains three appendices: a summary of the UML notation, a list of standard UML elements, and a summary of the Rational Unified Process. A glossary of common terms is also provided.

Each chapter addresses the use of a specific UML feature, and most are organized into the following four sections:

1. Getting Started
2. Terms and Concepts
3. Common Modeling Techniques

4. Hints and Tips

The third section introduces and then solves a set of common modeling problems. To make it easy for you to browse the guide in search of these use cases for the UML, each problem is identified by a distinct heading, as in the following example.

Modeling Architectural Patterns

Each chapter begins with a summary of the features it covers, as in the following example.

> **In this chapter**
>
> - Active objects, processes, and threads
> - Modeling multiple flows of control
> - Modeling interprocess communication
> - Building thread-safe abstractions

Similarly, parenthetical comments and general guidance are set apart as notes, as in the following example.

> **Note**
>
> You can specify more complex multiplicities by using a list, such as `0..1, 3..4, 6..*`, which would mean "any number of objects other than 2 or 5."

*Components are discussed in **Chapter 25**.*

The UML is semantically rich. Therefore, a presentation about one feature may naturally involve another. In such cases, cross references are provided in the left margin, as on this page.

Blue highlights are used in figures to distinguish text that explains a model from text that is part of the model itself. Code is distinguished by displaying it in a monospace font, as in `this example.`

A Brief History of the UML

Object-oriented modeling languages appeared sometime between the mid 1970s and the late 1980s as methodologists, faced with a new genre of object-oriented programming languages and increasingly complex applications, began to experiment with alternative approaches to analysis and design. The number of object-oriented methods increased from fewer than 10 to more than 50 during the period between 1989 and 1994. Many users of these methods had trouble finding a modeling language that met their needs completely, thus fueling the so-called method wars. Learning from experience, new generations of these methods began to appear, with a few clearly prominent methods emerging, most notably Booch, Jacobson's OOSE (Object-Oriented Software Engineering), and Rumbaugh's OMT (Object Modeling Technique). Other important methods included Fusion, Shlaer-Mellor, and Coad-Yourdon. Each of these was a complete method, although each was recognized as having strengths and weaknesses. In simple terms, the Booch method was particularly expressive during the design and construction phases of projects, OOSE provided excellent support for use cases as a way to drive requirements capture, analysis, and

high-level design, and OMT-2 was most useful for analysis and data-intensive information systems. The behavioral component of many object-oriented methods, including the Booch method and OMT, was the language of statecharts, invented by David Harel. Prior to this object-oriented adoption, statecharts were used mainly in the realm of functional decomposition and structured analysis, and led to the development of executable models and tools that generated full running code.

A critical mass of ideas started to form by the mid 1990s, when Grady Booch (Rational Software Corporation), Ivar Jacobson (Objectory), and James Rumbaugh (General Electric) began to adopt ideas from each other's methods, which collectively were becoming recognized as the leading object-oriented methods worldwide. As the primary authors of the Booch, OOSE, and OMT methods, we were motivated to create a unified modeling language for three reasons. First, our methods were already evolving toward each other independently. It made sense to continue that evolution together rather than apart, eliminating the potential for any unnecessary and gratuitous differences that would further confuse users. Second, by unifying our methods, we could bring some stability to the object-oriented marketplace, allowing projects to settle on one mature modeling language and letting tool builders focus on delivering more useful features. Third, we expected that our collaboration would yield improvements for all three earlier methods, helping us to capture lessons learned and to address problems that none of our methods previously handled well.

As we began our unification, we established three goals for our work:

1. To model systems, from concept to executable artifact, using object- oriented techniques
2. To address the issues of scale inherent in complex, mission-critical systems
3. To create a modeling language usable by both humans and machines

Devising a language for use in object-oriented analysis and design is not unlike designing a programming language. First, we had to constrain the problem: Should the language encompass requirements specification? Should the language be sufficient to permit visual programming? Second, we had to strike a balance between expressiveness and simplicity. Too simple a language would limit the breadth of problems that could be solved; too complex a language would overwhelm the mortal developer. In the case of unifying existing methods, we also had to be sensitive to the installed base. Make too many changes, and we would confuse existing users; resist advancing the language, and we would miss the opportunity of engaging a much broader set of users and of making the language simpler. The UML definition strives to make the best trade-offs in each of these areas.

The UML effort started officially in October 1994, when Rumbaugh joined Booch at Rational. Our project's initial focus was the unification of the Booch and OMT methods. The version 0.8 draft of the Unified Method (as it was then called) was released in October 1995. Around the same time, Jacobson joined Rational and the scope of the UML project was expanded to incorporate OOSE. Our efforts resulted in the release of the UML version 0.9 documents in June 1996. Throughout 1996, we invited and received feedback from the general software engineering community. During this time, it also became clear that many software organizations saw the UML as strategic to their business. We established a UML consortium, with several organizations willing to dedicate resources to work toward a strong and complete UML definition. Those partners contributing to the UML 1.0 definition included Digital Equipment Corporation, Hewlett-Packard, I-Logix, Intellicorp, IBM, ICON Computing, MCI Systemhouse, Microsoft, Oracle, Rational, Texas Instruments, and Unisys. This collaboration resulted in the UML 1.0, a modeling language that was well-defined, expressive, powerful, and applicable to a wide spectrum of problem domains. UML 1.0 was offered for standardization to the Object Management Group (OMG) in January 1997, in response to their request for proposal for a standard modeling language.

Between January 1997 and July 1997, the original group of partners was expanded to include virtually all of the other submitters and contributors of the original OMG response, including Andersen Consulting, Ericsson, ObjecTime Limited, Platinum Technology, PTech, Reich Technologies, Softeam, Sterling Software, and Taskon. A semantics task force was formed, led by Cris Kobryn of MCI Systemhouse and administered by Ed Eykholt of Rational, to formalize the UML specification and to integrate the UML with other standardization efforts. A revised version of the UML (version 1.1) was offered to the OMG for standardization in July 1997. In September 1997, this version was accepted by the OMG Analysis and Design Task Force (ADTF) and the OMG Architecture Board and then put up for vote by the entire OMG membership. UML 1.1 was adopted by the OMG on November 14, 1997.

Maintenance of the UML was then taken over by the OMG Revision Task Force (RTF), led by Cris Kobryn. The RTF released an editorial revision, UML 1.2, in June 1998. In fall 1998, the RTF released UML 1.3, which this user guide describes, providing some technical cleanup.

Acknowledgments

Grady Booch, Ivar Jacobson, and James Rumbaugh began the UML effort and throughout the project were its original designers, but the final product was a team effort among all the UML partners. Although all partners came with their own perspectives, areas of concern, and areas of interest, the overall result has benefited from the contributions of each of them and from the diversity of their experience and viewpoints.

The core UML team included

- Hewlett-Packard: Martin Griss
- I-Logix: Eran Gery, David Harel
- IBM: Steve Cook, Jos Warmer
- ICON Computing: Desmond D'Souza
- Intellicorp and James Martin and Company: James Odell
- MCI Systemhouse: Cris Kobryn, Joaquin Miller
- ObjecTime: John Hogg, Bran Selic
- Oracle: Guus Ramackers
- Platinum Technology: Dilhar DeSilva
- Rational Software: Grady Booch, Ed Eykholt, Ivar Jacobson, Gunnar Overgaard, Karin Palmkvist, James Rumbaugh
- Taskon: Trygve Reenskaugh
- Texas Instruments/Sterling Software: John Cheesman, Keith Short
- Unisys: Sridhar Iyengar, G.K. Khalsa

Cris Kobryn deserves a special acknowledgment for his leadership in directing the UML technical team during the development of UML 1.1, 1.2, and 1.3.

We also acknowledge the contributions, influence, and support of the following individuals. In some cases, individuals mentioned here have not formally endorsed the UML but are

nonetheless appreciated for their influence: Jim Amsden, Hernan Astudillo, Colin Atkinson, Dave Bernstein, Philip Bernstein, Michael Blaha, Conrad Bock, Mike Bradley, Ray Buhr, Gary Cernosek, James Cerrato, Michael Jesse Chonoles, Magnus Christerson, Dai Clegg, Geoff Clemm, Peter Coad, Derek Coleman, Ward Cunningham, Raj Datta, Philippe Desfray, Mike Devlin, Bruce Douglass, Staffan Ehnebom, Maria Ericsson, Johannes Ernst, Don Firesmith, Martin Fowler, Adam Frankl, Eric Gamma, Dipayan Gangopadhyay, Garth Gullekson, Rick Hargrove, Tim Harrison, Richard Helm, Brian Hendersen-Sellers, Michael Hirsch, Bob Hodges, Yves Holvoet, Jon Hopkins, John Hsia, Glenn Hughes, Ralph Johnson, Anneke Kleppe, Philippe Kruchten, Paul Kyzivat, Martin Lang, Grant Larsen, Reed Letsinger, Mary Loomis, Jeff MacKay, Joe Marasco, Robert Martin, Terri McDaniel, Jim McGee, Mike Meier, Randy Messer, Bertrand Meyer, Greg Meyers, Fred Mol, Luis Montero, Paul Moskowitz, Andy Moss, Jan Pachl, Paul Patrick, Woody Pidcock, Bill Premerlani, Jeff Price, Jerri Pries, Terry Quatrani, Mats Rahm, Rudolf Riess, Rich Reitman, Erick Rivas, Kenny Rubin, Jim Rye, Danny Sabbahr, Tom Schultz, Colin Scott, Ed Seidewitz, Keith Short, Gregson Sui, Jeff Sutherland, Dan Tasker, Andy Trice, Dave Tropeano, Dan Uhlar, John Vlissides, Larry Wall, Paul Ward, Alan Willis, Rebecca Wirfs-Brock, Bryan Wood, Ed Yourdon, and Steve Zeigler.

The development of the UML was an open process, and via the OTUG (Object Technology User's Group) we received thousands of e-mail messages from all over the world. Although we cannot mention every submitter by name, we do thank all of them for their comments and suggestions. We really did read each message, and the UML is better because of this broad international feedback.

A special acknowledgment also goes to a small band of lab rats (Loud and Boisterous RATional Students) who participated in a user guide course led by Booch in early 1997, during which they offered excellent ideas and gave much constructive criticism that helped fine-tune the contents of this book: Hernan Astudillo, Robin Brown, Robert Bundy, Magnus Christerson, Adam Frankl, Nookiah Kolluru, Ron Krubek, Grant Larsen, Dean Leffingwell, Robert Martin, Mills Ripley, Hugo Sanchez, Geri Schneider, Tom Schultz, Andy Trice, Dan Uhlar, and Lloyd Williams. Thanks go to the madmen at Number Six Software and to the folks who provided a technical review of this book: Jack Carter, Tim Budd, Bruce Douglass, Martin Fowler, Cris Kobryn, Philippe Kruchten, Ron Lusk, Terry Quatrani, and David Rine.

For More Information

The most current information about the UML, including its formal specification, may be found on the Internet at **http://www.rational.com** and **http://www.omg.org**. The work of the revision task force may be found at *uml.shl.com.*

There are several electronic forums that are appropriate for general discussion about the UML, including the Internet news groups *comp.software-eng* and *comp.object* and the public mailing lists **otug@rational.com** and **uml-rtf@omg.org**.

Grady Booch

Lakewood, Colorado

September 1998

egb@rational.com

Part I: Getting Started