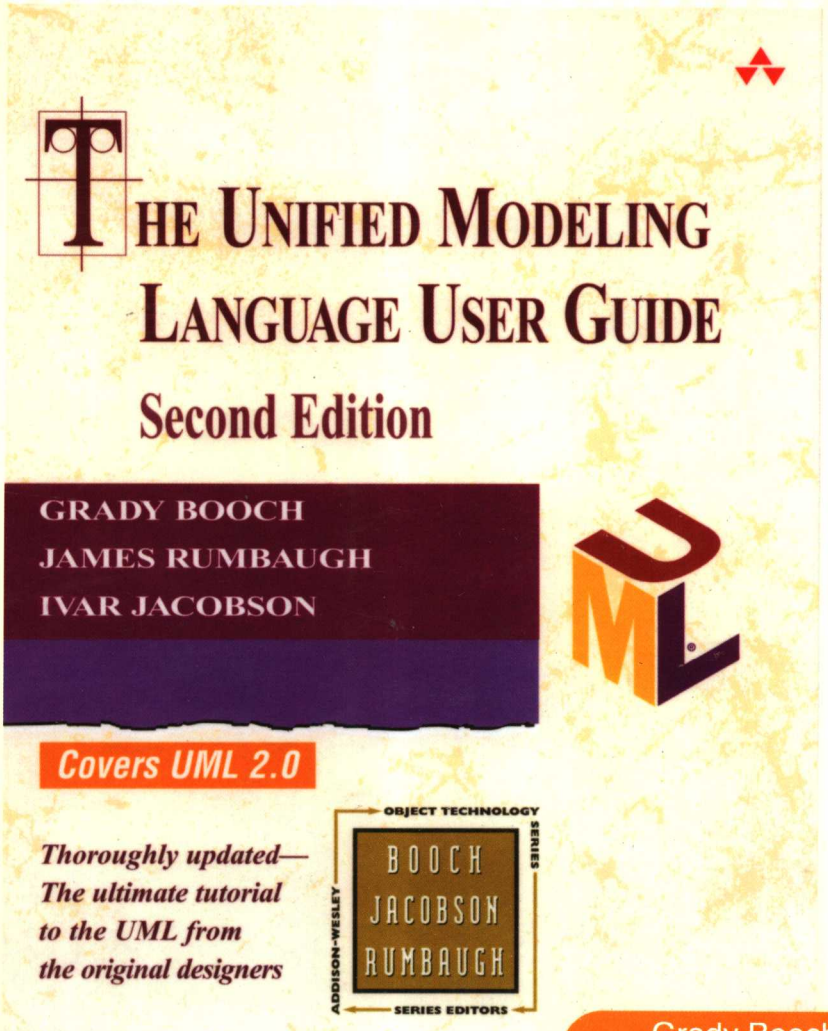
经 典 原 版 书 库

UML用户指南

（英文版 · 第2版）



（美）Grady Booch
James Rumbaugh 著
Ivar Jacobson

机械工业出版社
China Machine Press

经 典 原 版 书 库

UML用户指南

（英文版 · 第2版）

The Unified Modeling Language User Guide

(Second Edition)

Grady Booch
（美）James Rumbaugh 著
Ivar Jacobson

机械工业出版社
China Machine Press

本书版权登记号：图字：01-2006-1934

图书在版编目（CIP）数据

UML用户指南（英文版·第2版）/（美）布赫（Booch, G.）等著. -北京：机械工业出版社，2006.4

（经典原版书库）

书名原文：The Unified Modeling Language User Guide, Second Edition

ISBN 7-111-18827-6

Ⅰ.U…　Ⅱ.布…　Ⅲ.面向对象语言，UML-程序设计-英文　Ⅳ.TP312

中国版本图书馆CIP数据核字（2006）第029826号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）

责任编辑：迟振春

北京牛山世兴印刷厂印刷·新华书店北京发行所发行

2006年4月第1版第1次印刷

718mm×1020mm 1/16·31印张

定价：59.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：（010）68326294

出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅illuminate了研究的范畴，还揭橥了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短、从业人员较少的现状下，美国等发达国家在其计算机科学发展的几十年间积淀的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章图文信息有限公司较早意识到“出版要为教育服务”。自1998年开始，华章公司就将工作重点放在了遴选、移译国外优秀教材上。经过几年的不懈努力，我们与Prentice Hall，Addison-Wesley，McGraw-Hill，Morgan Kaufmann等世界著名出版公司建立了良好的合作关系，从它们现有的数百种教材中甄选出Tanenbaum，Stroustrup，Kernighan，Jim Gray等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及庋藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专程为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍，为进一步推广与发展打下了坚实的基础。

随着学科建设的初步完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都步入一个新的阶段。为此，华章公司将加大引进教材的力度，在“华章教育”的总规划之下出版三个系列的计算机教材：除“计算机科学丛书”之外，对影印版的教材，则单独开辟出“经典原版书库”；同时，引进全美通行的教学辅导书“Schaum's Outlines”系列组成“全美经典学习指导系列”。为了保证这三套丛书的权威性，同时也为了更好地为学校和老师们服务，华章公司聘请了中国科学院、北京大学、清华大学、国防科技大学、复旦大学、上海交通大学、南京大学、浙江大学、中国科技大学、哈尔

滨工业大学、西安交通大学、中国人民大学、北京航空航天大学、北京邮电大学、中山大学、解放军理工大学、郑州大学、湖北工学院、中国国家信息安全测评认证中心等国内重点大学和科研机构在计算机的各个领域的著名学者组成“专家指导委员会”，为我们提供选题意见和出版监督。

这三套丛书是响应教育部提出的使用外版教材的号召，为国内高校的计算机及相关专业的教学度身订造的。其中许多教材均已为M. I. T.，Stanford，U.C. Berkeley，C. M. U. 等世界名牌大学所采用。不仅涵盖了程序设计、数据结构、操作系统、计算机体系结构、数据库、编译原理、软件工程、图形学、通信与网络、离散数学等国内大学计算机专业普遍开设的核心课程，而且各具特色——有的出自语言设计者之手、有的历经三十年而不衰、有的已被全世界的几百所高校采用。在这些圆熟通博的名师大作的指引之下，读者必将在计算机科学的宫殿中由登堂而入室。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证，但我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。教材的出版只是我们的后续服务的起点。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

电子邮件：hzjsj@hzbook.com
联系电话：（010）68995264
联系地址：北京市西城区百万庄南街1号
邮政编码：100037

专家指导委员会

（按姓氏笔画顺序）

尤晋元	王　珊	冯博琴	史忠植	史美林
石教英	吕　建	孙玉芳	吴世忠	吴时霖
张立昂	李伟琴	李师贤	李建中	杨冬青
邵维忠	陆丽娜	陆鑫达	陈向群	周伯生
周克定	周傲英	孟小峰	岳丽华	范　明
郑国梁	施伯乐	钟玉琢	唐世渭	袁崇义
高传善	梅　宏	程　旭	程时端	谢希仁
裘宗燕	戴　葵			

To my loving wife, Jan, and my goddaughter, Elyse, both of whom make me whole.

—Grady Booch

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.

This book covers UML version 2.0.

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 Manual, Second Edition* (Rumbaugh, Jacobson, Booch, Addison-Wesley, 2005).

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 forming 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 parts:

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

The user guide contains two appendices: a summary of the UML notation and a summary of the Rational Unified Process. A glossary of common terms is also provided. An index follows.

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: Abstract operations map to what C++ calls pure virtual operations; leaf operations in the UML map to C++ nonvirtual operations.

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 indicate explanations about a model, as opposed to the model itself, which is always shown in black. Code is distinguished by displaying it in a monospace font, as in `this example`.

Acknowledgement. The authors wish to thank Bruce Douglass, Per Krol, and Joaquin Miller for their assistance in reviewing the manuscript of the second edition.

A Brief History of the UML

The first object-oriented language is generally acknowledged to be Simula-67, developed by Dahl and Nygaard in Norway in 1967. This language never had a large following, but its concepts were a major inspiration for later languages. Smalltalk became widely available in the early 1980s, followed by other object-oriented languages such as Objective C, C++, and Eiffel in the late 1980s. Object-oriented modeling languages appeared in the 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. A few methods gained prominence, including Booch's method, 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 was most useful for analysis and data-intensive information systems.

⊖ 原书为双色印刷，而本书为单色印刷，故显示不出来。——编辑注

A critical mass of ideas started to form by the mid 1990s when Grady Booch (Rational Software Corporation), James Rumbaugh (General Electric), Ivar Jacobson (Objectory), and others 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, OMT, and OOSE 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 to engage a much broader set of users and to make 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. Mary Loomis was instrumental in convincing the Object Management Group (OMG) to issue a request for proposals (RFP) for a standard modeling language. UML 1.0 was offered for standardization to the OMG in January 1997 in response to their RFP.

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.

For several years, UML was maintained by an OMG Revision Task Force, which produced versions 1.3, 1.4, and 1.5. From 2000 to 2003, a new and expanded set of partners produced an updated specification of UML, version 2.0. This version was reviewed for a year by a Finalization Task Force (FTF) headed by Bran Selic of IBM, and the official version of UML 2.0 was adopted by the OMG in early 2005. UML 2.0 is a major revision of UML 1 and includes a large number of additional features. In addition, many changes were made to previous constructs based on experience with the previous version. The actual UML specification documents are found on the OMG Website at www.omg.org.

UML is the work of a large number of individuals, and the ideas in it come from a wide range of previous works. It would be a major historical research project to reconstruct a complete list of sources, and even more difficult to identify the many predecessors who have influenced UML in manners large and small. As with all scientific research and engineering practice, UML is a small hill atop a large mountain of previous experience.

CONTENTS