UML与面向对象设计影印丛书

UML参考手册

THE UNIFIED MODELING LANGUAGE REFERENCE MANUAL

(美) JAMES RUMBAUGH
IVAR JACOBSON
GRADY BOOCH 编著

The definitive reference to the UML from the original designers

OBJECT TECHNOLOGY SERIES
BOOCH
JACOBSON
RUMBAUGH
ADDISON-WESLEY
SERIES EDITORS

科学出版社
www.sciencep.com

Pearson Education

UML 与面向对象设计影印丛书

UML 参考手册

The Unified Modeling Language Reference Manual

（美） James Rumbaugh
Ivar Jacobson Grady Booch 编著

科学出版社
北京

图字：01-2003-6986 号

内 容 简 介

UML 是一种用于建立面向对象系统模型的标准标记法。本书首先概述了 UML 的历史、基本概念、目标及使用方法，然后按字母顺序列出了 UML 的所有术语及标准元素，从语义、表示法和用途等方面详尽地介绍了 UML 的构成和概念。本书的三位作者是面向对象方法最早的倡导者，也是 UML 的原创人员。

本书可供广大软件开发人员、系统用户和工程技术人员查询和参考。

图书在版编目(CIP)数据

UML 参考手册=The Unified Modeling Language Reference Guide/（美）James Rumbaugh 等编著. 影印本. —北京：科学出版社，2004
（UML 与面向对象设计影印丛书）
ISBN 7-03-012467-7

Ⅰ.U… Ⅱ.R… 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（2003）第 099959 号

策划编辑：李佩乾/责任编辑：舒 立
责任印制：吕春珉/封面制作：东方人华平面设计室

科 学 出 版 社 出版
北京东黄城根北街16号
邮政编码：100717
http://www.sciencep.com
双 青 印 刷 厂 印刷

科学出版社发行 各地新华书店经销

*

2004 年 1 月第 一 版 开本：787×1092 1/16
2004 年 1 月第一次印刷 印张：35 1/2
印数：1—3 000 字数：674 000

定价：60.00 元

（如有印装质量问题，我社负责调换〈环伟〉）

影印前言

随着计算机硬件性能的迅速提高和价格的持续下降，其应用范围也在不断扩大。交给计算机解决的问题也越来越难，越来越复杂。这就使得计算机软件变得越来越复杂和庞大。20 世纪 60 年代的软件危机使人们清醒地认识到按照工程化的方法组织软件开发的必要性。于是软件开发方法从 60 年代毫无工程性可言的手工作坊式开发，过渡到 70 年代结构化的分析设计方法、80 年代初的实体关系开发方法，直到面向对象的开发方法。

面向对象的软件开发方法是在结构化开发范型和实体关系开发范型的基础上发展而来的，它运用分类、封装、继承、消息等人类自然的思维机制，允许软件开发者处理更为复杂的问题域和其支持技术，在很大程度上缓解了软件危机。面向对象技术发端于程序设计语言，以后又向软件开发的早期阶段延伸，形成了面向对象的分析和设计。

20 世纪 80 年代末 90 年代初，先后出现了几十种面向对象的分析设计方法。其中，Booch, Coad/Yourdon、OMT 和 Jacobson 等方法得到了面向对象软件开发界的广泛认可。各种方法对许多面向对象的概念的理解不尽相同，即便概念相同，各自技术上的表示法也不同。通过 90 年代不同方法流派之间的争论，人们逐渐认识到不同的方法既有其容易解决的问题，又有其不容易解决的问题，彼此之间需要进行融合和借鉴；并且各种方法的表示也有很大的差异，不利于进一步的交流与协作。在这种情况下，统一建模语言(UML)于 90 年代中期应运而生。

UML 的产生离不开三位面向对象的方法论专家 G. Booch、J. Rumbaugh 和 I. Jacobson 的通力合作。他们从多种方法中吸收了大量有用的建模概念，使 UML 的概念和表示法在规模上超过了以往任何一种方法，并且提供了允许用户对语言做进一步扩展的机制。UML 使不同厂商开发的系统模型能够基于共同的概念，使用相同的表示法，呈现彼此一致的模型风格。1997 年 11 月 UML 被 OMG 组织正式采纳为标准的建模语言，并在随后的几年中迅速地发展为事实上的建模语言国际标准。

UML 在语法和语义的定义方面也做了大量的工作。以往各种关于面向对象方法的著作通常是以比较简单的方式定义其建模概念，而以主要篇幅给出过程指导，论述如何运用这些概念来进行开发。UML 则以一种建模语言的姿态出现，使用语言学中的一些技术来定义。尽管真正从语言学的角度看它还有许多缺陷，但它在这方面所做的努力却是以往的各种建模方法无法比拟的。

从 UML 的早期版本开始，便受到了计算机产业界的重视，OMG 的采纳和大公司的支持把它推上了实际上的工业标准的地位，使它拥有越来越多的用户。它被广泛地用

于应用领域和多种类型的系统建模，如管理信息系统、通信与控制系统、嵌入式实时系统、分布式系统、系统软件等。近几年还被运用于软件再工程、质量管理、过程管理、配置管理等方面。而且它的应用不仅仅限于计算机软件，还可用于非软件系统，例如硬件设计、业务处理流程、企业或事业单位的结构与行为建模，等等。

在 UML 陆续发布的几个版本中，逐步修正了前一个版本中的缺陷和错误。即将发布的 UML2.0 版本将是对 UML 的又一次重大的改进。将来的 UML 将向着语言家族化、可执行化、精确化等理念迈进，为软件产业的工程化提供更有力的支撑。

本丛书收录了与面向对象技术和 UML 有关的十几本书，反映了面向对象技术最新的发展趋势以及 UML 的新的研究动态。其中涉及对面向对象建模理论研究与实践的有这样几本书：《面向对象系统架构及设计》主要讨论了面向对象的基本概念、静态设计、永久对象、动态设计、设计模式以及体系结构等近几年来面向对象技术领域中的新的理论知识与方法；《用 UML 进行用况对象建模》主要介绍了面向对象的需求阶段、分析阶段、设计阶段中用况模型的建立方法与技术；《高级用况建模》介绍了在建立用况模型中需要注意的高级的问题与技术；《UML 面向对象设计基础》则侧重于经典的面向对象理论知识的阐述；《UML 参考手册》列出了 UML 的所有术语和标准元素，从语义、表示法和用途等方面详尽地介绍了 UML 的构成和概念。

涉及 UML 在特定领域的运用的有这样几本：《UML 实时系统开发》讨论了进行实时系统开发时需要对 UML 进行扩展的技术；《用 UML 构建 Web 应用程序》讨论了运用 UML 进行 Web 应用建模所应该注意的技术与方法；《面向对象系统测试：模型、视图与工具》介绍了将 UML 应用于面向对象的测试领域所应掌握的方法与工具；《对象、构件、框架与 UML 应用》讨论了如何运用 UML 对面向对象的新技术——构件-框架技术建模的方法策略；《UML 与 Visual Basic 应用程序开发》主要讨论了从 UML 模型到 Visual Basic 程序的建模与映射方法；《XML 程序的 UML 建模》讲解了如何将 XML 与 UML 结合，创建动态的 Web 应用程序、实现最优的 B2B 应用集成；《构建可扩展数据库应用程序》介绍了商务模式和数据库模式的建模方法以及集成系统的程序实现；《UML 与并行分布式实时应用程序设计》对 UML 在并行分布式实时系统开发中的应用作了全面而详细的介绍，尤其对面向对象方法解决此类系统特有的问题作了有针对性的讲解；《UML 与 J2EE 企业应用程序开发》系统介绍了使用 J2EE 开发企业级软件工程时，将 UML 建模技术应用到软件开发各个阶段的方法。

介绍面向对象编程技术的有两本书：《COM 高手心经》和《ATL 技术内幕》，深入探讨了面向对象的编程新技术——COM 和 ATL 技术的使用技巧与技术内幕。

还有一本《Executable UML 技术内幕》，这本书介绍了可执行 UML 的理念与其支持技术，使得模型的验证与模拟以及代码的自动生成成为可能，也代表着将来软件开发的一种新的模式。

总之，这套书所涉及的内容包含了对软件生命周期的全过程建模的方法与技术，同时也对近年来的热点领域建模技术、新型编程技术作了深入的介绍，有些内容已经涉及到了前沿领域。可以说，每一本都很经典。

有鉴于此，特向软件领域中不同程度的读者推荐这套书，供大家阅读、学习和研究。

北京大学计算机系　蒋严冰 博士

敬告读者：

本书光盘如有需求，请向我社索取，费用 20.00 元/张。

联系人：朱敏　电话：010-62622941　E-mail: zhumin@abook.cn

Preface

Goals

This book is intended to be a complete and useful reference to the Unified Modeling Language (UML) for the developer, architect, project manager, system engineer, programmer, analyst, contracting officer, customer, and anyone else who needs to specify, design, build, or understand complex software systems. It provides a full reference to the concepts and constructs of UML, including their semantics, syntax, notation, and purpose. It is organized to be a convenient but thorough reference for the working professional developer. It also attempts to provide additional detail about issues that may not be clear from the standards documents and to provide a rationale for many decisions that went into the UML.

This book is not intended as a guide to the UML standards documents or to the internal structure of the metamodel contained in them. The details of the metamodel are of interest to methodologists and UML tool builders, but most other developers have little need for the arcane details of the Object Management Group (OMG) documents. This book provides all the details of UML that most developers need; in many cases, it makes information explicit that must otherwise be sought between the lines of the original documents. For those who do wish to consult the source documents, they are included on the accompanying CD.

This book is intended as a reference for those who already have some understanding of object-oriented technology. For beginners, the original books by us and by other authors are listed in the bibliography; although some of the notation has changed, books such as [Rumbaugh-91], [Booch-94], [Jacobson-92], and [Meyer-88] provide an introduction to object-oriented concepts that is still valid and therefore unnecessary to duplicate here. For a tutorial introduction to UML that shows how to model a number of common problems, see *The Unified Modeling Language User Guide* [Booch-99]. Those who already know an object-oriented method, such as OMT, Booch, Objectory, Coad-Yourdon, or Fusion, should be able to read the *Reference Manual* and use it to understand UML notation and semantics; to learn UML quickly, they may nevertheless find it useful to read the *User Guide.*

UML does not require a particular development process, and this book does not describe one. Although UML may be used with a variety of development processes, it was designed to support an iterative, incremental, use-case–driven process with a strong architectural focus—the kind we feel is most suitable for the development of modern, complex systems. *The Unified Software Development Process* [Jacobson-99] describes the kind of process we believe complements the UML and best supports software development.

Outline of the Book

The UML Reference Manual is organized into three parts: an overview of UML history and of modeling, a survey of UML concepts, and an alphabetical encyclopedia of UML terms and concepts.

The first part is a survey of UML—its history, purposes, and uses—to help you understand the origin of UML and the need it tries to fill.

The second part is a brief survey of UML views so that you can put all the concepts into perspective. The survey provides a brief overview of the views UML supports and shows how the various constructs work together. This part begins with an example that walks through various UML views and then contains one chapter for each kind of UML view. This survey is not intended as a full tutorial or as a comprehensive description of concepts. It serves mainly to summarize and relate the various UML concepts and provides starting points for detailed readings in the encyclopedia.

The third part contains the reference material organized for easy access to each topic. The bulk of the book is an alphabetical encyclopedia of all of the concepts and constructs in UML. Each UML term of any importance has its own entry in the encyclopedia. The encyclopedia is meant to be complete; therefore, everything in the concept overview in Part 2 is repeated in more detail in the encyclopedia. The same or similar information has sometimes been included in multiple encyclopedia articles so that the reader can conveniently find it.

The reference part also contains an alphabetic list of UML standard elements. A standard element is a feature predefined using the UML extensibility mechanisms. The standard elements are extensions that are felt to be widely useful.

Appendices show the UML metamodel, a summary of UML notation, and some standard sets of extensions for particular domains. There is a brief bibliography of major object-oriented books, but no attempt has been made to include a comprehensive citation of sources of ideas for UML or other approaches. Many of the books in the bibliography contain excellent lists of references to books and journal articles for those interested in tracking the development of the ideas.

Encyclopedia Article Formatting Conventions

The encyclopedia part of the book is organized as an alphabetical list of entries, each describing one concept in some detail. The articles represent a flat list of UML concepts at various conceptual levels. A high-level concept typically contains a summary of its subordinate concepts, each of which is fully described in a separate article. The articles are highly cross-referenced. This flat encyclopedia organization permits the description of each concept to be presented at a fairly uniform level of detail, without constant shifts in level for the nested descriptions that would be necessary for a sequential presentation. The hypertext format of the document should also make it convenient for reference. It should not be necessary to use the index much; instead go directly to the main article in the encyclopedia for any term of interest and follow cross-references. This format is not necessarily ideal for learning the language; beginners are advised to read the overview description of UML found in Part 2 or to read introductory books on UML, such as the *UML User Guide* [Booch-99].

Encyclopedic articles have the following divisions, although not all divisions appear in all articles.

Brief definition

The name of the concept appears in boldface, set to the left of the body of the article. A brief definition follows in normal type. This definition is intended to capture the main idea of the concept, but it may simplify the concept for concise presentation. Refer to the main article for precise semantics.

Semantics

This section contains a detailed description of the meaning of the concept, including constraints on its uses and its execution consequences. Notation is not covered in this section, although examples use the appropriate notation. General semantics are given first. For concepts with subordinate structural properties, a list of the properties follows the general semantics, often under the subheading *Structure*. In most cases, the properties appear as a table in alphabetical order by property name, with the description of each property on the right. If a property has a brief enumerated list of choices, they may be given as an indented sublist. In more complicated cases, the property is given its own article to avoid excessive nesting. When properties require more explanation than permitted by a table, they are described in normal text with run-in headers in boldface italics. In certain cases, the main concept is best described under several logical subdivisions rather than one list. In such cases, additional sections follow or replace the *Structure* subsection. Although several organizational mechanisms have been used, their structure should be obvious to the reader.

Notation

This section contains a detailed description of the notation for the concept. Usually, the notation section has a form that parallels the preceding semantics section, which it references, and it often has the same divisions. The notation section usually includes one or more diagrams to illustrate the concept. The actual notation is printed in black ink. To help the reader understand the notation, many diagrams contain annotations in blue ink. Any material in blue is commentary and is not part of the actual notation.

Example

This subsection contains examples of notation or illustrations of the use of the concept. Frequently, the examples also treat complicated or potentially confusing situations.

Discussion

This section describes subtle issues, clarifies tricky and frequently confused points, and contains other details that would otherwise digress from the more descriptive semantics section. A minority of articles have a discussion section.

This section also explains certain design decisions that were made in the development of the UML, particularly those that may appear counterintuitive or that have provoked strong controversy. Only a fraction of articles have this section. Simple differences in taste are generally not covered.

Standard elements

This section lists standard constraints, tags, stereotypes, and other conventions that are predefined for the concept in the article. This section is fairly rare.

Syntax Conventions

Syntax expressions. Syntax expressions are given in a modified BNF format in a sans serif font. To avoid confusing literal values and syntax productions, literal values that appear in the target sentence are printed in black ink, and the names of syntax variables and special syntax operators are printed in blue ink.

Text printed in black ink appears in that form in the target string.

Punctuation marks (they are always printed in black) appear in the target string.

Any word printed in blue ink represents a variable that must be replaced by another string or another syntax production in the target string. Words may contain letters and hyphens. If a blue word is italicized or underlined, the actual replacement string must be italicized or underlined.

In code examples, comments are printed in blue ink to the right of the code text. Subscripts and overbars are used as syntax operators as follows:

$\text{expression}_{\text{opt}}$	The expression is optional.
$\text{expression}_{\text{list},}$	A comma-separated list of the expression may appear. If there is zero or one repetition, there is no separator. Each repetition may have a separate substitution. If a different punctuation mark than a comma appears in the subscript, then it is the separator.
$\overline{= \text{expression}}_{\text{opt}}$	An overbar ties together two or more terms that are considered a unit for optional or repeated occurrences. In this example, the equal sign and the expression form one unit that may be omitted or included. The overbar is unnecessary if there is only one term.

Two-level nesting is avoided.

Literal strings. In running text, language keywords, names of model elements, and sample strings from models are shown in a sans serif font.

Diagrams. In diagrams, blue text and arrows are annotations, that is, explanations of the diagram notation that do not appear in an actual diagram. Any text and symbols in black ink are actual diagram notation.

CD

This book is accompanied by a CD containing the full text of the book in Adobe Reader (PDF) format. Using Adobe Reader, the viewer can easily search the book for a word or phrase. The CD version also contains a clickable table of contents, index, Adobe Reader thumbnails, and extensive hot links in the body of the articles. Simply click on one of the links to jump to the encyclopedia article for the word or phrase.

The CD also contains the full text of the OMG UML specifications, included by the permission of the Object Management Group.

We feel that this CD will be a useful on-line reference to UML for advanced users.

For More Information

Additional source files and up-to-date information on further work on UML and related topics can be found on the World Wide Web sites www.rational.com and www.omg.org.

Acknowledgments

We want to thank many people who made the UML possible. First, we must thank Rational Software Corporation, especially Mike Devlin and Paul Levy, who had the vision to bring us together, start the unification work, and stay the course during the four years that were required to bring the work to successful completion. We also thank the Object Management Group for providing the framework that brought together many diverse viewpoints and merged them together into a broad consensus that was much greater than any one contribution.

We particularly want to thank Cris Kobryn, who led the technical team that prepared the UML standard and who managed to achieve a consensus among an extremely strong-willed group of persons (and the three of us were not the least of his problems). His diplomatic skills and technical balance kept the UML effort from foundering amid many differences of opinion. Cris also reviewed the book and provided countless useful suggestions.

We would like to thank Gunnar Övergaard for reviewing the book thoroughly, as well as for his perseverance in completing many sections of the UML documents that were not fun to write but were necessary to its formal correctness.

We want to thank Karin Palmkvist for an exceedingly thorough review that uncovered many bugs in technical content, as well as many flaws in grammar, phrasing, and presentation.

We would also like to thank Mike Blaha, Conrad Bock, Perry Cole, Bruce Douglass, Martin Fowler, Eran Gery, Pete McBreen, Guus Ramackers, Tom Schultz, Ed Seidewitz, and Bran Selic for their helpful reviews.

Most of all, we want to thank the scores or even hundreds of persons who contributed to the community of ideas from which UML was drawn—ideas in object-oriented technology, software methodology, programming languages, user interfaces, visual programming, and numerous other areas of computer science. It is impossible to list them all, or indeed to track even the major chains of influence, without a major scholarly effort, and this is an engineering book, not a historical review. Many are well known, but many good ideas came from those who did not have the good fortune to be widely recognized.

On a more personal note, I wish to thank Professor Jack Dennis, who inspired my work in modeling and the work of many other students, more than twenty-five years ago. The ideas from his Computations Structures Group at MIT have borne much fruit, and they are not the least of the sources of UML. I must also thank Mary Loomis and Ashwin Shah, with whom I developed the original ideas of OMT, and my former colleagues at GE R&D Center, Mike Blaha, Bill Premerlani, Fred Eddy, and Bill Lorensen, with whom I wrote the OMT book.

Finally, without the patience of my wife, Madeline, and my sons, Nick and Alex, there would have been no UML and no book about it.

James Rumbaugh
Cupertino, California
November 1998

Contents

Part 4: Appendices