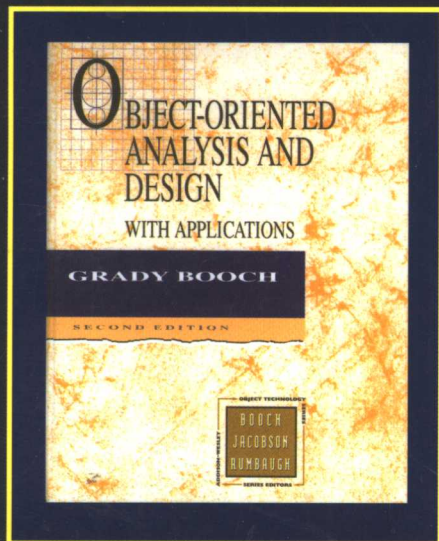


Object-Oriented Analysis and Design
with Applications

面向对象分析与设计

(第二版·影印版)

[美] Grady Booch 著



UML 创始人 Grady Booch 代表作 ■

获美国《Software Development》杂志 Jolt 大奖 ■

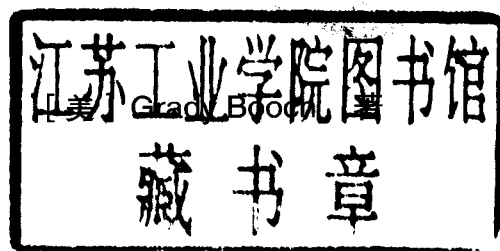
面向对象分析与设计领域公认权威之作 ■

原 版 风 暴 系 列

Object-Oriented Analysis and Design
with Applications

面向对象分析与设计

(第二版·影印版)



B1285966

中国电力出版社

OBJECT-ORIENTED ANALYSIS AND DESIGN with Applications, 2nd edition (ISBN 0-8053-5340-2)

Grady Booch

Copyright © 1994 Addison Wesley Longman, Inc.

Original English Language Edition Published by Addison Wesley Longman, Inc.

All rights reserved.

Reprinting edition published by PEARSON EDUCATION ASIA LTD and CHINA ELECTRIC POWER PRESS, Copyright © 2003.

本书影印版由 Pearson Education 授权中国电力出版社在中国境内（香港、澳门特别行政区和台湾地区除外）独家出版、发行。

未经出版者书面许可，不得以任何方式复制或抄袭本书的任何部分。

本书封面贴有 Pearson Education 防伪标签，无标签者不得销售。

北京市版权局著作合同登记号：图字：01-2003-3853

仅限于中华人民共和国境内（不包括中国香港、澳门特别行政区和中国台湾地区）销售发行。

图书在版编目（CIP）数据

面向对象分析与设计：第2版 / （美）布奇著．—影印本．—北京：中国电力出版社，2003.10
（原版风暴系列）

ISBN 7-5083-1807-2

I.面... II.布... III.面向对象语言—程序设计—英文 IV.TP312

中国版本图书馆 CIP 数据核字（2003）第 086082 号

丛 书 名：原版风暴系列

书 名：面向对象分析与设计（第二版·影印版）

编 著：（美）Grady Booch

责任编辑：陈维宁

出版发行：中国电力出版社

地址：北京市三里河路6号 邮政编码：100044

电话：（010）88515918 传 真：（010）88518169

印 刷：汇鑫印务有限公司

开 本：787×1092 1/16

印 张：33.5 插 页：2

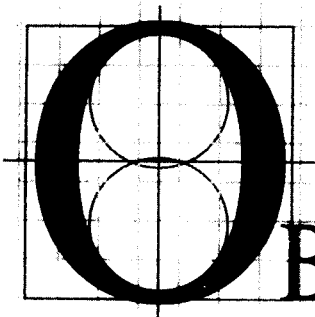
书 号：ISBN 7-5083-1807-2

版 次：2003年10月北京第一版

2003年10月第一次印刷

定 价：55.00 元

版权所有 翻印必究



OBJECT-ORIENTED ANALYSIS AND DESIGN with Applications

SECOND EDITION

Grady Booch

Rational
Santa Clara, California

Mankind, under the grace of God, hungers for spiritual peace, esthetic achievements, family security, justice, and liberty, none directly satisfied by industrial productivity. But productivity allows the sharing of the plentiful rather than fighting over scarcity; it provides time for spiritual, esthetic, and family matters. It allows society to delegate special skills to institutions of religion, justice, and the preservation of liberty.

HARLAN MILLS
DPMA and Human Productivity

As computer professionals, we strive to build systems that are useful and that work; as software engineers, we are faced with the task of creating complex systems in the presence of scarce computing and human resources. Over the past few years, object-oriented technology has evolved in diverse segments of the computer sciences as a means of managing the complexity inherent in many different kinds of systems. The object model has proven to be a very powerful and unifying concept.

Changes to the First Edition

Since the publication of the first edition of *Object-Oriented Design with Applications*, object-oriented technology has indeed moved into the mainstream of industrial-strength software development. We have encountered the use of the object-oriented paradigm throughout the world, for such diverse domains as the administration of banking transactions; the automation of bowling alleys; the management of public utilities; and the mapping of the human genome. Many of the next generation operating systems, database systems, telephony systems, avionics systems, and multimedia applications are being written using object-oriented techniques. Indeed, many such projects have chosen to use

object-oriented technology simply because there appears to be no other way to economically produce an enduring and resilient programming system.

Over the past several years, hundreds of projects have applied the notation and process described in *Object-Oriented Design with Applications*.^{*} Through our own work with several of these projects, as well as the kind contribution of many individuals who have taken the time to communicate with us, we have found ways to improve our method, in terms of better articulating the process, adding and clarifying certain semantics otherwise missing or difficult to express in the notation, and simplifying the notation where possible.

During this time, many other methods have also appeared, including the work of Jacobson, Rumbaugh, Coad and Yourdon, Constantine, Shlaer and Mellor, Martin and Odell, Wasserman, Goldberg and Rubin, Embley, Wirfs-Brock, Goldstein and Alger, Henderson-Sellers, Firesmith, and others. Rumbaugh's work is particularly interesting, for as he points out, our methods are more similar than they are different. We have surveyed many of these methods, interviewed developers and managers who have applied them, and where possible, tried these methods ourselves. Because we are more interested in helping projects succeed with object-oriented technology rather than dogmatically hanging on to practices solely for emotional or historical reasons, we have tried to incorporate the best from each of these methods in our own work. We gratefully acknowledge the fundamental and unique contributions each of these people has made to the field.

It is in the best interests of the software development industry, and object-oriented technology in particular, that there be standard notations for development. Therefore, this edition presents a unified notation that, where possible, eliminates the cosmetic differences between our notation and that of others, particularly Jacobson's and Rumbaugh's. As before, and to encourage the unrestricted use of the method, this notation is in the public domain.

The goals, audience, and structure of this edition remain the same as for the first edition. However, there are five major differences between this edition and the original publication.

First, Chapter 5 has been expanded to provide much more specific detail about the unified notation. To enhance the reader's understanding of this notation, we explicitly distinguish between its fundamental and advanced elements. In addition, we have given special attention to how the various views of the notation integrate with one another.

Second, Chapters 6 and 7, dealing with the process and pragmatics of object-oriented analysis and design, have been greatly expanded. We have also changed the title of this second edition to reflect the fact that our process does indeed encompass analysis as well as design.

Third, we have chosen to express all programming examples in the main text using C++. This language is rapidly becoming the de facto standard in

^{*} Including my own projects. Ultimately, I'm a developer, not just a methodologist. The first question you should ask any methodologist is if he or she uses their own methods to develop software.

many application domains; additionally, most professional developers who are versed in other object-oriented programming languages can read C++. This is not to say that we view other languages – such as Smalltalk, CLOS, Ada, or Eiffel – as less important. The focus of this book is on analysis and design, and because we need to express concrete examples, we choose to do so in a reasonably common programming language. Where applicable, we describe the semantics unique to these other languages and their impact upon the method.

Fourth, this edition introduces several new application examples. Certain idioms and architectural frameworks have emerged in various application domains, and these examples take advantage of these practices. For example, client/server computing provides the basis of a revised application example.

Finally, almost every chapter provides references to and discussion of the relevant object-oriented technology that has appeared since the first edition.

Goals

This book provides practical guidance on the construction of object-oriented systems. Its specific goals are:

- To provide a sound understanding of the fundamental concepts of the object model
- To facilitate a mastery of the notation and process of object-oriented analysis and design
- To teach the realistic application of object-oriented development within a variety of problem domains

The concepts presented herein all stand on a solid theoretical foundation, but this is primarily a pragmatic book that addresses the practical needs and concerns of the software engineering community.

Audience

This book is written for the computer professional as well as for the student.

- For the practicing software engineer, we show you how to effectively use object-oriented technology to solve real problems.
- In your role as an analyst or architect, we offer you a path from requirements to implementation, using object-oriented analysis and design. We develop your ability to distinguish “good” object-oriented architectures from “bad” ones, and to trade off alternate designs when the perversity of the real world intrudes. Perhaps most important, we offer you fresh approaches to reasoning about complex systems.

- For the program manager, we provide insight on how to allocate the resources of a team of developers, and on how to manage the risks associated with complex software systems.
- For the tool builder and the tool user, we provide a rigorous treatment of the notation and process of object-oriented development as a basis for computer-aided software engineering (CASE) tools.
- For the student, we provide the instruction necessary for you to begin acquiring several important skills in the science and art of developing complex systems.

This book is also suitable for use in undergraduate and graduate courses as well as in professional seminars and individual study. Because it deals primarily with a method of software development, it is most appropriate for courses in software engineering and advanced programming, and as a supplement to courses involving specific object-oriented programming languages.

Structure

The book is divided into three major sections – Concepts, The Method, and Applications – with considerable supplemental material woven throughout.

Concepts

The first section examines the inherent complexity of software and the ways in which complexity manifests itself. We present the object model as a means of helping us manage this complexity. In detail, we examine the fundamental elements of the object model: abstraction, encapsulation, modularity, hierarchy, typing, concurrency, and persistence. We address basic questions such as “What is a class?” and “What is an object?” Because the identification of meaningful classes and objects is the key task in object-oriented development, we spend considerable time studying the nature of classification. In particular, we examine approaches to classification in other disciplines, such as biology, linguistics, and psychology, then apply these lessons to the problem of discovering classes and objects in software systems.

The Method

The second section presents a method for the development of complex systems based on the object model. We first present a graphic notation for object-oriented analysis and design, followed by its process. We also examine the pragmatics of object-oriented development – in particular, its place in the software development life cycle and its implications for project management.

Applications

The final section offers a collection of five complete, nontrivial examples encompassing a diverse selection of problem domains: data acquisition, application frameworks, client/server information management, artificial intelligence, and command and control. We have chosen these particular problem domains because they are representative of the kinds of complex problems faced by the practicing software engineer. It is easy to show how certain principles apply to simple problems, but because our focus is on building useful systems for the real world, we are more interested in showing how the object model scales up to complex applications. Some readers may be unfamiliar with the problem domains chosen, so we begin each application with a brief discussion of the fundamental technology involved (such as database design and blackboard system architecture). The development of software systems is rarely amenable to cookbook approaches; therefore, we emphasize the incremental development of applications, guided by a number of sound principles and well-formed models.

Supplemental Material

A considerable amount of supplemental material is woven throughout the book. Most chapters have boxes that provide information on important topics, such as the mechanics of method dispatch in different object-oriented programming languages. We also include an appendix on object-oriented programming languages, in which we consider the distinction between object-based and object-oriented programming languages and the evolution and essential properties of both categories of languages. For those readers who are unfamiliar with certain object-oriented programming languages, we provide a summary of the features of a few common languages, with examples. We also provide a glossary of common terms and an extensive classified bibliography that provides references to source material on the object model. Lastly, the end pages provide a summary of the notation and process of the object-oriented development method.

Available apart from the text, and new to the second edition, is an Instructor's Guide containing suggested exercises, discussion questions, and projects, which should prove very useful in the classroom. *The Instructor's Guide with Exercises* (ISBN 0-8053-5341-0) has been developed by Mary Beth Rosson from IBM's Thomas J. Watson laboratory. Qualified instructors may receive a free copy from their local sales representatives or by emailing aw.cse@aw.com. Questions, suggestions, and contributions to the Instructor's Guide may be emailed to rosson@watson.ibm.com.

Tools and training that support the Booch method are available from a variety of sources. For further information, contact Rational at any of the numbers listed on the last page of this book. Additionally, Addison-Wesley can provide educational users with software that supports this notation.

11/14/97/9

Using this Book

This book may be read from cover to cover or it may be used in less structured ways. If you are seeking a deep understanding of the underlying concepts of the object model or the motivation for the principles of object-oriented development, you should start with Chapter 1 and continue forward in order. If you are primarily interested in learning the details of the notation and process of object-oriented analysis and design, start with Chapters 5 and 6; Chapter 7 is especially useful to managers of projects using this method. If you are most interested in the practical application of object-oriented technology to a specific problem domain, select any or all of Chapters 8 through 12.

Acknowledgments

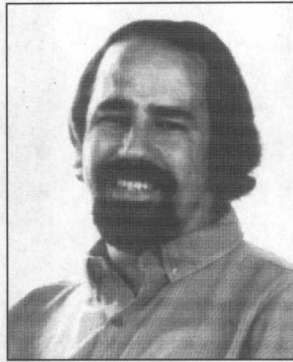
This book is dedicated to my wife, Jan, for her loving support.

Through both the first and second editions, a number of individuals have shaped my ideas on object-oriented development. For their contributions, I especially thank Sam Adams, Mike Akroid, Glenn Andert, Sid Bailin, Kent Beck, Daniel Bobrow, Dick Bolz, Dave Bulman, Dave Bernstein, Kayvan Carun, Dave Collins, Steve Cook, Damian Conway, Jim Coplien, Brad Cox, Ward Cunningham, Tom DeMarco, Mike Devlin, Richard Gabriel, William Genemaras, Adele Goldberg, Ian Graham, Tony Hoare, Jon Hopkins, Michael Jackson, Ralph Johnson, James Kempf, Norm Kerth, Jordan Kreindler, Doug Lea, Phil Levy, Barbara Liskov, Cliff Longman, James MacFarlane, Masoud Milani, Harlan Mills, Robert Murray, Steve Neis, Gene Ouye, Dave Parnas, Bill Riddel, Mary Beth Rosson, Kenny Rubin, Jim Rumbaugh, Kurt Schmucker, Ed Seidewitz, Dan Shiffman, Dave Stevenson, Bjarne Stroustrup, Dave Thomas, Mike Vilot, Tony Wasserman, Peter Wegner, Iseult White, John Williams, Lloyd Williams, Mario Wolczko, Niklaus Wirth, and Ed Yourdon.

A large part of the pragmatics of this book derives from my involvement with complex software systems being developed around the world at companies such as Apple, Alcatel, Andersen Consulting, AT&T, Autotrol, Bell Northern Research, Boeing, Borland, Computer Sciences Corporation, Contel, Ericsson, Ferranti, General Electric, GTE, Holland Signaal, Hughes Aircraft Company, IBM, Lockheed, Martin Marietta, Motorola, NTT, Philips, Rockwell International, Shell Oil, Symantec, Taligent, and TRW. I have had the opportunity to interact with literally hundreds of professional software engineers and their managers, and I thank them all for their help in making this book relevant to real-world problems.

A special acknowledgment goes to Rational for their support of my work. Thanks also to my editor, Dan Joraanstad, for his encouragement during this project, and to Tony Hall, whose cartoons brighten what would otherwise be just another stuffy technical book. Finally, thanks to my three cats, Camy, Annie, and Shadow, who kept me company on many a late night of writing.

ABOUT THE AUTHOR



Grady Booch, Chief Scientist at Rational Software Corporation, is recognized throughout the international software development community for his pioneering work in object methods and applications. He is a featured columnist in **Object Magazine** and **C++ Report**, and the author of several best-selling books on software engineering and object-oriented development. Grady Booch also edits and contributes to the **Object-Oriented Software Engineering Series** published by Addison-Wesley.

Preface v

The First Section: Concepts 1

Chapter 1: Complexity 3

- 1.1 The Inherent Complexity of Software 3
- 1.2 The Structure of Complex Systems 9
- 1.3 Bringing Order to Chaos 16
- 1.4 On Designing Complex Systems 21

Sidebar: Categories of Analysis and Design Methods 18

Chapter 2: The Object Model 27

- 2.1 The Evolution of the Object Model 28
- 2.2 Elements of the Object Model 40
- 2.3 Applying the Object Model 72

Sidebar: Foundations of the Object Model 36

Chapter 3: Classes and Objects 81

- 3.1 The Nature of an Object 81
- 3.2 Relationships Among Objects 97
- 3.3 The Nature of a Class 103
- 3.4 Relationships Among Classes 106
- 3.5 The Interplay of Classes and Objects 135
- 3.6 On Building Quality Classes and Objects 136
- Sidebar: Invoking a Method 118

Chapter 4: Classification 145

- 4.1 The Importance of Proper Classification 146
- 4.2 Identifying Classes and Objects 150
- 4.3 Key Abstractions and Mechanisms 162
- Sidebar: A Problem of Classification 151

The Second Section: The Method 169**Chapter 5: The Notation 171**

- 5.1 Elements of the Notation 172
- 5.2 Class Diagrams 176
- 5.3 State Transition Diagrams 199
- 5.4 Object Diagrams 208
- 5.5 Interaction Diagrams 217
- 5.6 Module Diagrams 219
- 5.7 Process Diagrams 223
- 5.8 Applying the Notation 226

Chapter 6: The Process 229

- 6.1 First Principles 230
- 6.2 The Micro Development Process 234
- 6.3 The Macro Development Process 248

Chapter 7: Pragmatics 267

- 7.1 Management and Planning 268
- 7.2 Staffing 271
- 7.3 Release Management 275

- 7.4 Reuse 277
- 7.5 Quality Assurance and Metrics 278
- 7.6 Documentation 281
- 7.7 Tools 282
- 7.8 Special Topics 285
- 7.8 The Benefits and Risks of Object-Oriented Development 287

The Third Section: Applications 291

Chapter 8: Data Acquisition: Weather Monitoring Station 293

- 8.1 Analysis 294
 - 8.2 Design 312
 - 8.3 Evolution 318
 - 8.4 Maintenance 325
- Sidebar: Weather Monitoring Station Requirements 294

Chapter 9: Frameworks: Foundation Class Library 327

- 9.1 Analysis 328
 - 9.2 Design 333
 - 9.3 Evolution 365
 - 9.4 Maintenance 372
- Sidebar: Foundation Class Library Requirements 329

Chapter 10: Client/Server Computing: Inventory Tracking 377

- 10.1 Analysis 378
 - 10.2 Design 400
 - 10.3 Evolution 410
 - 10.4 Maintenance 412
- Sidebar: Inventory Tracking System Requirements 379

Chapter 11: Artificial Intelligence: Cryptanalysis 413

- 11.1 Analysis 414

11.2 Design 421

11.3 Evolution 438

11.4 Maintenance 442

Sidebar: Cryptanalysis Requirements 415

Chapter 12: Command and Control: Traffic Management 445

12.1 Analysis 446

12.2 Design 455

12.3 Evolution 464

12.4 Maintenance 468

Sidebar: Traffic Management System Requirements 448

Afterword 471

Appendix: Object-Oriented Programming Languages 473

A.1 Concepts 474

A.2 Smalltalk 475

A.3 Object Pascal 479

A.4 C++ 480

A.5 Common Lisp Object System 484

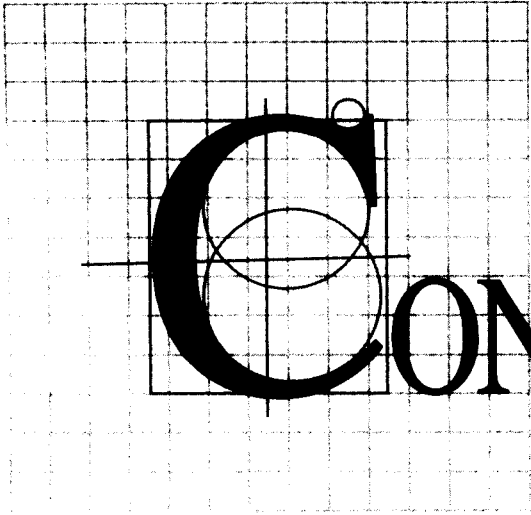
A.6 Ada 486

A.7 Eiffel 487

A.8 Other Object-Oriented Programming Languages 489

Notes 491

Glossary 511



C CONCEPTS

Sir Isaac Newton secretly admitted to some friends: He understood how gravity *behaved*, but not how it *worked*!

LILY TOMLIN

The Search for Signs of Intelligent Life in the Universe

