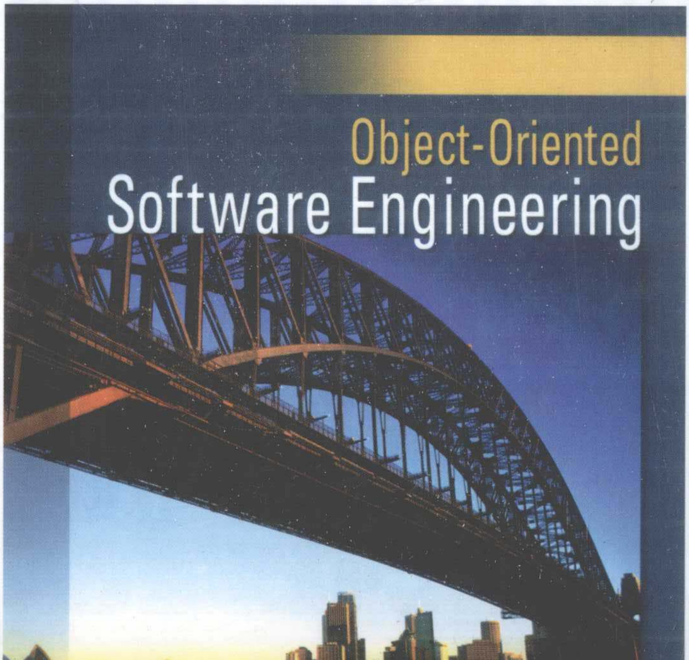
HZ BOOKS
华章教育

McGraw Hill Education

经 典 原 版 书 库

面向对象软件工程

（英文版）



（美） Stephen R. Schach
范德比尔特大学 著

机械工业出版社
China Machine Press

经 典 原 版 书 库

面向对象软件工程

（英文版）

Object-Oriented Software Engineerin

（美）Stephen R. Schach 著
范德比尔特大学

机 械 工 业 出 版 社
China Machine Press

本书版权登记号：图字：01-2008-1763

图书在版编目（CIP）数据

面向对象软件工程（英文版）/（美）沙赫（Schach，S. R.）著. —北京：机械工业出版社，2009.3

（经典原版书库）

书名原文：Object-Oriented Software Engineering

ISBN 978-7-111-26526-9

Ⅰ. 面…　Ⅱ. 沙…　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（2009）第031375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

北京京师印务有限公司印刷

2009年3月第1版第1次印刷

150mm × 214mm · 18.125印张

标准书号：ISBN 978-7-111-26526-9

定价：49.00元

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，不仅擘划了研究的范畴，还揭示了学术的源变，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和发展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到积极的推动作用，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分社较早意识到“出版要为教育服务”。自1998年开始，华章分社就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与Pearson，McGraw-Hill，Elsevier，MIT，John Wiley & Sons，Cengage等世界著名出版公司建立了良好的合作关系，从他们现有的数百种教材中甄选出Andrew S. Tanenbaum，Bjarne Stroustrup，Brain W. Kernighan，Dennis Ritchie，Jim Gray，Afred V. Aho，John E. Hopcroft，Jeffrey D. Ullman，Abraham Silberschatz，William Stallings，Donald E. Knuth，John L. Hennessy，Larry L. Peterson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、研究及珍藏。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”已经出版了近两百个品种，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。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

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。随着计算机科学与技术专业学科建设的不断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将步入一个新的阶段，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。华章分社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：

华章网站：	www.hzbook.com
电子邮件：	hzjsj@hzbook.com
联系电话：	（010）88379604
联系地址：	北京市西城区百万庄南街1号
邮政编码：	100037

HZ BOOKS
华章教育

Preface

The wheel has turned full circle.

In 1988, I wrote a textbook entitled *Software Engineering*. Virtually the only mention of the object-oriented paradigm in that book was one section that described object-oriented design.

By 1994, the object-oriented paradigm was starting to gain acceptance in the software industry, so I wrote a textbook called *Classical and Object-Oriented Software Engineering*. Six years later, however, the object-oriented paradigm had become more important than the classical paradigm. To reflect this change, I switched the order of the two topics in the title of the textbook I wrote in 2000, and called it *Object-Oriented and Classical Software Engineering*.

Nowadays, use of the classical paradigm is largely restricted to maintaining legacy software. Students learn C++ or Java as their first programming language, and object-oriented languages are used in subsequent computer science and computer engineering courses. Students expect that, when they graduate, they will work for a company that uses the object-oriented paradigm. The object-oriented paradigm has all but squeezed out the classical paradigm. And that is why I have written a textbook entitled *Object-Oriented Software Engineering*.

Features of This Book

- The Unified Process is still largely the methodology of choice for object-oriented software development. Throughout this book, the student is therefore exposed to both the theory and the practice of the Unified Process.
- In Chapter 1 ("The Scope of Object-Oriented Software Engineering"), the strengths of the object-oriented paradigm are analyzed in depth.
- The iterative-and-incremental life-cycle model has been introduced as early as possible, namely, in Chapter 2 ("Software Life-Cycle Models"). Agile processes are also discussed in this chapter.
- In Chapter 3 ("The Software Process"), the workflows (activities) and processes of the Unified Process are introduced, and the need for two-dimensional life-cycle models is explained.
- A wide variety of ways of organizing software teams are presented in Chapter 4 ("Teams"), including teams for agile processes and for open-source software development.
- Chapter 5 ("Tools of the Trade") includes information on important classes of CASE tools.
- The importance of continual testing is stressed in Chapter 6 ("Testing").
- Objects are the focus of attention in Chapter 7 ("From Modules to Objects").
- In Chapter 8 ("Reusability and Portability"), design patterns have been stressed.
- The new IEEE standard for software project management plans is presented in Chapter 9 ("Planning and Estimating").

- Chapter 10 ("The Requirements Workflow"), Chapter 11 ("The Analysis Workflow"), Chapter 12 ("The Design Workflow"), and Chapter 13 ("The Implementation Workflow") are largely devoted to the workflows (activities) of the Unified Process.
- The material in Chapter 13 ("The Implementation Workflow") clearly distinguishes between implementation and integration.
- The importance of postdelivery maintenance is stressed in Chapter 14 ("Postdelivery Maintenance").
- Chapter 15 ("More on UML") provides additional material on UML to prepare the student thoroughly for employment in the software industry. This chapter is of particular use to instructors who utilize this book for the two-semester software engineering course sequence. In the second semester, in addition to developing the team-based term project or a capstone project, the student can acquire additional knowledge of UML, beyond what is needed for this book.
- There are two running case studies. The MSG Foundation case study and the elevator problem case study have been developed using the Unified Process. Java and C++ implementations are available online at www.mhhe.com/schach.
- In addition to the two running case studies that are used to illustrate the complete life cycle, seven mini case studies highlight specific topics, such as the moving-target problem, stepwise refinement, design patterns, and postdelivery maintenance.
- I stress the importance of documentation, maintenance, reuse, portability, testing, and CASE tools. It is no use teaching students the latest ideas unless they appreciate the importance of the basics of object-oriented software engineering.
- Attention is paid to object-oriented life-cycle models, object-oriented analysis, object-oriented design, management implications of the object-oriented paradigm, and the testing and maintenance of object-oriented software. Metrics for the object-oriented paradigm also are included. In addition, many briefer references are made to objects, a paragraph or even only a sentence in length. The reason is that the object-oriented paradigm is not just concerned with how the various workflows are performed but rather permeates the way we think about software engineering. Object technology pervades this book.
- The software process underlies the book as a whole. To control the process, we have to be able to measure what is happening to the project. Accordingly, there is an emphasis on metrics. With regard to process improvement, there is material on the capability maturity model (CMM), ISO/IEC 15504 (SPICE), and ISO/IEC 12207; the people capability maturity model (P–CMM) has been included in the chapter on teams.
- The book is language independent; the few code examples are presented in C++ and Java, and I have made every effort to smooth over language-dependent details and ensure that the code examples are equally clear to C++ and Java users. For example, instead of using cout for C++ output and System.out.println for Java output, I have utilized the pseudocode instruction *print*. (The one exception is the second case study, where complete implementation details are given in both C++ and Java.)
- This book contains over 600 references. I have selected current research papers as well as classic articles and books whose message remains fresh and relevant. There is no question that object-oriented software engineering is a rapidly moving field, and

students therefore need to know the latest results and where in the literature to find them. At the same time, today's cutting-edge research is based on yesterday's truths, and I see no reason to exclude an older reference if its ideas are as applicable today as they originally were.

- With regard to prerequisites, it is assumed that the reader is familiar with one high-level object-oriented programming language such as C++ or Java. In addition, the reader is expected to have taken a course in data structures.

How This Book Is Organized

This book is written for both the traditional one-semester and the newer two-semester software engineering curriculum, now growing in popularity. In the traditional one-semester (or one-quarter) course, the instructor has to rush through the theoretical material to provide the students the knowledge and skills needed for the term project as soon as possible. The need for haste is so that the students can commence the term project early enough to complete it by the end of the semester. To cater to a one-semester, project-based software engineering course, Part 2 of this book covers the software life cycle, workflow by workflow, and Part 1 contains the theoretical material needed to understand Part 2. For example, Part 1 introduces the reader to CASE, metrics, and testing; each chapter of Part 2 contains a section on CASE tools for that workflow, a section on metrics for that workflow, and a section on testing during that workflow. Part 1 is kept short to enable the instructor to start Part 2 relatively early in the semester. Furthermore, the last two chapters of Part 1 (Chapters 8 and 9) may be postponed, and then taught in parallel with Part 2. As a result, the class can begin developing the term project as soon as possible.

We turn now to the two-semester software engineering curriculum. More and more computer science and computer engineering departments are realizing that the overwhelming preponderance of their graduates find employment as software engineers. As a result, many colleges and universities have introduced a two-semester (or two-quarter) software engineering sequence. The first course is largely theoretical (but often includes a small project of some sort). The second course comprises a major team-based term project. This is usually a capstone project. When the term project is in the second course, there is no need for the instructor to rush to start Part 2.

Therefore, an instructor teaching a one-semester (or one-quarter) sequence using this book covers most of Chapters 1 through 7 and then starts Part 2 (Chapters 10 through 15). Chapters 8 and 9 can be taught in parallel with Part 2 or at the end of the course while the students are implementing the term project. When teaching the two-semester sequence, the chapters of the book are taught in order; the class now is fully prepared for the team-based term project that it will develop in the following semester.

To ensure that the key software engineering techniques of Part 2 truly are understood, each is presented twice. First, when a technique is introduced, it is illustrated by means of the elevator problem. The elevator problem is the correct size for the reader to be able to see the technique applied to a complete problem, and it has enough subtleties to highlight both the strengths and weaknesses of the technique being taught. Then, the relevant portion of the MSG Foundation case study is presented. This detailed solution provides the second illustration of each technique.

The Problem Sets

This book has five types of problems. First, there are running object-oriented analysis and design projects at the end of Chapters 10, 11, and 12. These have been included because the only way to learn how to perform the requirements, analysis, and design workflows is from extensive hands-on experience.

Second, the end of each chapter contains a number of exercises intended to highlight key points. These exercises are self-contained; the technical information for all the exercises can be found in this book.

Third, there is a software term project. It is designed to be solved by students working in teams of three, the smallest number of team members that cannot confer over a standard telephone. The term project comprises 14 separate components, each tied to the relevant chapter. For example, design is the topic of Chapter 12, so in that chapter the component of the term project is concerned with software design. By breaking a large project into smaller, well-defined pieces, the instructor can monitor the progress of the class more closely. The structure of the term project is such that an instructor may freely apply the 14 components to any other project that he or she chooses.

Because this book has been written for use by graduate students as well as upper-class undergraduates, the fourth type of problem is based on research papers in the software engineering literature. In each chapter, an important paper has been chosen. The student is asked to read the paper and answer a question relating to its contents. Of course, the instructor is free to assign any other research paper; the For Further Reading section at the end of each chapter includes a wide variety of relevant papers.

The fifth type of problem relates to the case study. This type of problem has been included in response to a number of instructors who feel that their students learn more by modifying an existing product than by developing a new product from scratch. Many senior software engineers in the industry agree with that viewpoint. Accordingly, each chapter in which the case study is presented has problems that require the student to modify the case study in some way. For example, in one chapter the student is asked what the effect would have been of performing the steps of the object-oriented analysis in a different order. To make it easy to modify the source code of the case study, it is available on the World Wide Web at www.mhhe.com/schach.

The website also has material for instructors, including a complete set of PowerPoint lecture notes, and detailed solutions to all the exercises as well as to the term project.

Material on UML

This book makes substantial use of the Unified Modeling Language (UML). If the students do not have previous knowledge of UML, this material may be taught in two ways. I prefer to teach UML on a just-in-time basis; that is, each UML concept is introduced just before it is needed. The following table describes where the UML constructs used in this book are introduced.

Construct	Section in Which the Corresponding UML Diagram is Introduced
Class diagram, note, inheritance (generalization), aggregation, association, navigation triangle	Section 7.7
Use case	Section 10.4.3
Use-case diagram, use-case description	Section 10.7
Stereotype	Section 11.4
Statechart	Section 11.9
Interaction diagram (sequence diagram, communication diagram)	Section 11.18

Alternatively, Chapter 15 contains an introduction to UML, including material above and beyond what is needed for this book. Chapter 15 may be taught at any time; it does not depend on material in the first 14 chapters. The topics covered in Chapter 15 are given in the following table:

Construct	Section in Which the Corresponding UML Diagram is Introduced
Class diagram, aggregation, multiplicity, composition, generalization, association	Section 15.2
Note	Section 15.3
Use-case diagram	Section 15.4
Stereotype	Section 15.5
Interaction diagram	Section 15.6
Statechart	Section 15.7
Activity diagram	Section 15.8
Package	Section 15.9
Component diagram	Section 15.10
Deployment diagram	Section 15.11

Acknowledgments

I should like to thank the reviewers of this book:

Michael A. Aars,
Baylor University
Keith S. Decker,
University of Delaware
Xiaocong Fan,
The Pennsylvania State University
Adrian Fiech,
Memorial University
Sudipto Ghosh,
Colorado State University
Anita Kuchera,
Rensselaer Polytechnic Institute
Matthew R. McBride,
Southern Methodist University
Michael McCracken,
Georgia Institute of Technology
Rick Mercer,
University of Arizona
Richard J. Povinelli,
Marquette University

David C. Rine,
George Mason University
Keng Siau,
University of Nebraska-Lincoln

John Sturman,
Rensselaer Polytechnic Institute
Levent Yilmaz,
Auburn University

I warmly thank three individuals who have also made significant contributions to previous books I have written. First, Kris Irwin provided a complete solution to the term project, including implementing it in both Java and C++. Second, Jeff Gray implemented the MSG Foundation case study. Third, Lauren Ryder was a coauthor of the *Instructor's Solution Manual* and contributor to the PowerPoint slides.

I turn now to McGraw-Hill. I am truly grateful to Senior Managing Editor Faye Schilling for her willingness to take over the role of production manager mid-project. I am also most appreciative of her readiness to modify the schedule when needed. Developmental Editor Lora Kalb was a pillar of strength from start to finish; it was a real pleasure to work with Lora again. I also warmly thank copyeditor Lucy Mullins, proofreader Dorothy Wendell, and Production Manager Joyce Berendes. Finally, I am grateful to Brenda Rolwes in coordinating with cover designer Jenny Hobein from Studio Montage. Jenny transformed a photograph of Sydney Harbour Bridge into a striking cover.

I would like to thank the numerous instructors from all over the world who sent me e-mail regarding my other books. I look forward with anticipation to receiving instructors' feedback on this book also. My e-mail address is srs@vuse.vanderbilt.edu.

Students, too, continue to be most helpful. Once more I thank my students at Vanderbilt University for their provocative questions and constructive suggestions, both inside and outside the classroom. I also am most appreciative of the questions and comments e-mailed to me by students from all over the world. As with my previous books, I look forward keenly to student feedback on this book, too.

Finally, as always, I thank my family for their continual support. As with all my previous books, I did my utmost to try to ensure that family commitments took precedence over writing. However, when deadlines loomed, this was sometimes not possible. At such times, they were always understanding, and for this I am most grateful.

It is my privilege to dedicate my fourteenth book to my grandson, Jackson, with love.

Stephen R. Schach

Contents