HZ BOOKS
华章教育

McGraw Hill Education

经典原版书库

软件工程

面向对象和传统的方法

(美) Stephen R. Schach 范德比尔特大学 著

(英文版·第8版)

Object-Oriented and Classical Software Engineering

Eighth Edition

机械工业出版社
China Machine Press

Stephen R. Schach

典 原 版 书 库

软件工程

面向对象和传统的方法

（英文版·第8版）

Object-Oriented and Classical Software Engineering (Eighth Edition)

（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：01-2011-1460

图书在版编目（CIP）数据

软件工程：面向对象和传统的方法（英文版 · 第8版）/（美）沙赫（Schach, S. R.）著. —北京：机械工业出版社，2011.5

（经典原版书库）

书名原文：Object-Oriented and Classical Software Engineering, Eighth Edition

ISBN 978-7-111-34196-3

I. 软…　II. 沙…　III. ①软件工程－英文　②面向对象语言－程序设计－英文　IV. ①TP311.5　②TP312

中国版本图书馆CIP数据核字（2011）第071255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
北京瑞德印刷有限公司印刷
2011年6月第1版第1次印刷
186mm × 240mm · 43 印张
标准书号：ISBN 978-7-111-34196-3
定价：79.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991；88361066
购书热线：（010）68326294；88379649；68995259
投稿热线：（010）88379604
读者信箱：hzjsj@hzbook.com

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公司较早意识到“出版要为教育服务”。自1998年开始，我们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：

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

华章科技图书出版中心

Preface

Almost every computer science and computer engineering curriculum now includes a required team-based software development project. In some cases, the project is only one semester or quarter in length, but a year-long team-based software development project is fast becoming the norm.

In an ideal world, every student would complete a course in software engineering before starting his or her team-based project ("two-stage curriculum"). In practice, however, many students have to start their projects partway through their software engineering course, or even at the beginning of the course ("parallel curriculum").

As explained in the next section, this book is organized in such a way that it can be used for both curricula.

How the Eighth Edition Is Organized

The book comprises two main parts: Part B teaches the students how to develop a software product; Part A provides the necessary theoretical background for Part B. The 18 chapters are organized as follows:

	Chapter 1	Introduction to software engineering
Part A	Chapters 2 through 9	Software engineering concepts
Part B	Chapters 10 through 17	Software engineering techniques
	Chapter 18	Emerging technologies

Chapter 10 is new. It contains a summary of the key material of Part A. When the two-stage curriculum is followed, the instructor teaches first Part A and then Part B (omitting Chapter 10, because the material of Chapter 10 will have been covered in depth in Part A). For the parallel curriculum, the instructor first teaches Part B (so that the students can start their projects as soon as possible), and then Part A. The material of Chapter 10 enables the students to understand Part B without first covering Part A.

This latter approach seems counterintuitive: Surely theory should always be taught before practice. In fact, curricular issues have forced many of the instructors who have used the seventh edition of this book to teach the material of Part B before Part A. Surprisingly, they have been most satisfied with the outcome. They report that their students have a greater appreciation of the theoretical material of Part A as a consequence of their project work. That is, team-based project work makes students more receptive to and understanding of the theoretical concepts that underlie software engineering.

In more detail, the material of the eighth edition may be taught in the following two ways:

1. Two-Stage Curriculum

	Chapter 1 (Introduction to software engineering)
Part A	Chapters 2 through 9 (Software engineering concepts)
Part B	Chapters 11 through 17 (Software engineering techniques)
	Chapter 18 (Emerging technologies)
	The students then commence their team-based projects in the following semester or quarter.

2. Parallel Curriculum

	Chapter 1 (Introduction to software engineering)
	Chapter 10 (Key material from Part A)
	The students now commence their team-based projects, in parallel with studying the material of Part B.
Part B	Chapters 11 through 17 (Software engineering techniques)
Part A	Chapters 2 through 9 (Software engineering concepts)
	Chapter 18 (Emerging technologies)

New Features of the Eighth Edition

- The book has been updated throughout.
- I have added two new chapters. As previously explained, Chapter 10, a summary of key points of Part A, has been included so that this book can be used when students start their team-based term projects in parallel with their software engineering course. The other new chapter, Chapter 18, gives an overview of 10 emerging technologies, including
  - Aspect-oriented technology
  - Model-driven technology
  - Component-based technology
  - Service-oriented technology
  - Social computing
  - Web engineering
  - Cloud technology
  - Web 3.0
  - Computer security
  - Model checking
- I have considerably expanded the material on design patterns in Chapter 8, including a new mini case study.
- Two theoretical tools have been added to Chapter 5: divide-and-conquer, and separation of concerns.
- The object-oriented analysis of the elevator problem of Chapter 13 now reflects a modern distributed, decentralized architecture.
- The references have been extensively updated, with an emphasis on current research.
- There are well over 100 new problems.
- There are new Just in Case You Wanted to Know boxes.

Features Retained from the Seventh Edition

- 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 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. Furthermore, as with all previous editions, numerous other life-cycle models are presented, compared, and contrasted. Particular attention is paid to agile processes.
- 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 ("The Tools of the Trade") includes information on important classes of CASE tools.
- The importance of continual testing is stressed in Chapter 6 ("Testing").
- Objects continue to be the focus of attention in Chapter 7 ("From Modules to Objects").
- Design patterns remain a central focus of Chapter 8 ("Reusability and Portability").
- The IEEE standard for software project management plans is again presented in Chapter 9 ("Planning and Estimating").
- Chapter 11 ("Requirements"), Chapter 13 ("Object-Oriented Analysis"), and Chapter 14 ("Design") are largely devoted to the workflows (activities) of the Unified Process. For obvious reasons, Chapter 12 ("Classical Analysis") is largely unchanged.
- The material in Chapter 15 ("Implementation") clearly distinguishes between implementation and integration.
- The importance of postdelivery maintenance is stressed in Chapter 16.
- Chapter 17 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.
- As before, there are two running case studies. The MSG Foundation case study and the Elevator Problem case study have been developed using the Unified Process. As usual, 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, eight mini case studies highlight specific topics, such as the moving target problem, stepwise refinement, design patterns, and postdelivery maintenance.
- In all the previous editions, I have stressed the importance of documentation, maintenance, reuse, portability, testing, and CASE tools. In this edition, all these concepts are stressed equally firmly. It is no use teaching students the latest ideas unless they appreciate the importance of the basics of software engineering.
- As in the seventh edition, particular 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 phases are performed but rather permeates the way we think about software engineering. Object technology again pervades this book.

- The software process is still the concept that underlies the book as a whole. To control the process, we have to be able to measure what is happening to the project. Accordingly, the emphasis on metrics continues. With regard to process improvement, the material on the capability maturity model (CMM), ISO/IEC 15504 (SPICE), and ISO/IEC 12207 has been retained.
- The book is still 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 new case study, where complete implementation details are given in both C++ and Java, as before.)
- As in the seventh edition, 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 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 a high-level programming language such as C, C#, C++, or Java. In addition, the reader is expected to have taken a course in data structures.

Why the Classical Paradigm Is Still Included

There is now almost unanimous agreement that the object-oriented paradigm is superior to the classical paradigm. Accordingly, many instructors who adopted the seventh edition of *Object-Oriented and Classical Software Engineering* chose to teach only the object-oriented material in that book. However, when asked, instructors indicated that they prefer to adopt a text that includes the classical paradigm.

The reason is that, even though more and more instructors *teach* only the object-oriented paradigm, they still *refer* to the classical paradigm in class; many object-oriented techniques are hard for the student to understand unless that student has some idea of the classical techniques from which those object-oriented techniques are derived. For example, understanding entity-class modeling is easier for the student who has been introduced, even superficially, to entity-relationship modeling. Similarly, a brief introduction to finite state machines makes it easier for the instructor to teach statecharts. Accordingly, I have retained classical material in the eighth edition, so that instructors have classical material available for pedagogical purposes.

The Problem Sets

As in the seventh edition, this book has five types of problems. First, there are running object-oriented analysis and design projects at the end of Chapters 11, 13, and 14. 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 15 separate components, each tied to the relevant chapter. For example, design is the topic of Chapter 14, 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 15 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; wherever possible, a paper related to object-oriented software engineering has been selected. 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 was first introduced in the third edition in response to a number of instructors who felt 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 to redesign the case study using a different design technique from the one used for the case study. In another 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 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 UML (Unified Modeling Language). 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 11.4.3
Use-case diagram, use-case description	Section 11.7
Stereotype	Section 13.1
Statechart	Section 13.6
Interaction diagram (sequence diagram, communication diagram)	Section 13.15

Alternatively, Chapter 17 contains an introduction to UML, including material above and beyond what is needed for this book. Chapter 17 may be taught at any time; it does not depend on material in the first 16 chapters. The topics covered in Chapter 17 are as follows:

Construct	Section in Which the Corresponding UML Diagram Is Introduced
Class diagram, aggregation, multiplicity, composition, generalization, association	Section 17.2
Note	Section 17.3
Use-case diagram	Section 17.4
Stereotype	Section 17.5
Interaction diagram	Section 17.6
Statechart	Section 17.7
Activity diagram	Section 17.8
Package	Section 17.9
Component diagram	Section 17.10
Deployment diagram	Section 17.11

Online Resources

A website to accompany the text is available at www.mhhe.com/schach. The website features Java and C++ implementations as well as source code for the MSG case study for students. For instructors, lecture PowerPoints, detailed solutions to all exercises and the term project, and an image library are available. For details, contact your sales representative.

Electronic Textbook Options

E-books are an innovative way for students to save money and create a greener environment at the same time. An e-book can save students about half the cost of a traditional textbook and offers unique features like a powerful search engine, highlighting, and the ability to share notes with classmates using e-books.

McGraw-Hill offers this text as an e-book. To talk about the e-book options, contact your McGraw-Hill sales representative or visit the site www.coursesmart.com to learn more.

Acknowledgments

I greatly appreciate the constructive criticisms and many helpful suggestions of the reviewers of the seven previous editions. Special thanks go to the reviewers of this edition, including

Ramzi Bualuan
University of Notre Dame
Ruth Dameron
University of Colorado, Boulder
Werner Krandick
Drexel University
Mike McCracken
Georgia Institute of Technology
Nenad Medvidovic
University of Southern California
Saeed Monemi
California Polytechnic University, Pomona

Taehyung Wang
California State University, Northridge

Xiaojun Qi
Utah State University

Jie Wei
City University of New York—City College

With regard to my publishers, McGraw-Hill, I am most grateful to copyeditor Kevin Campbell and designer Brenda Rolwes. A special word of thanks goes to Melissa Welch of Studio Montage, who transformed a photograph of Sydney Harbour Bridge at night into the stunning cover.

Special thanks also go to Jean Naudé (Vaal University of Technology, Secunda Campus) for co-authoring the Instructor's Solution Manual. In particular, Jean provided a complete solution for the term project, including implementing it in both Java and C++. In the course of working on the ISM, Jean made numerous constructive suggestions for improving this book. I am most grateful to Jean.

Finally, as always, I thank my wife, Sharon, for her continual support and encouragement. As with all my previous books, I did my utmost to ensure that family commitments took precedence over writing. However, when deadlines loomed, this was not always possible. At such times, Sharon always understood, and for this I am most grateful.

It is my privilege to dedicate my fifteenth book to my grandchildren, Jackson and Mikaela, with love.

Stephen R. Schach

Contents