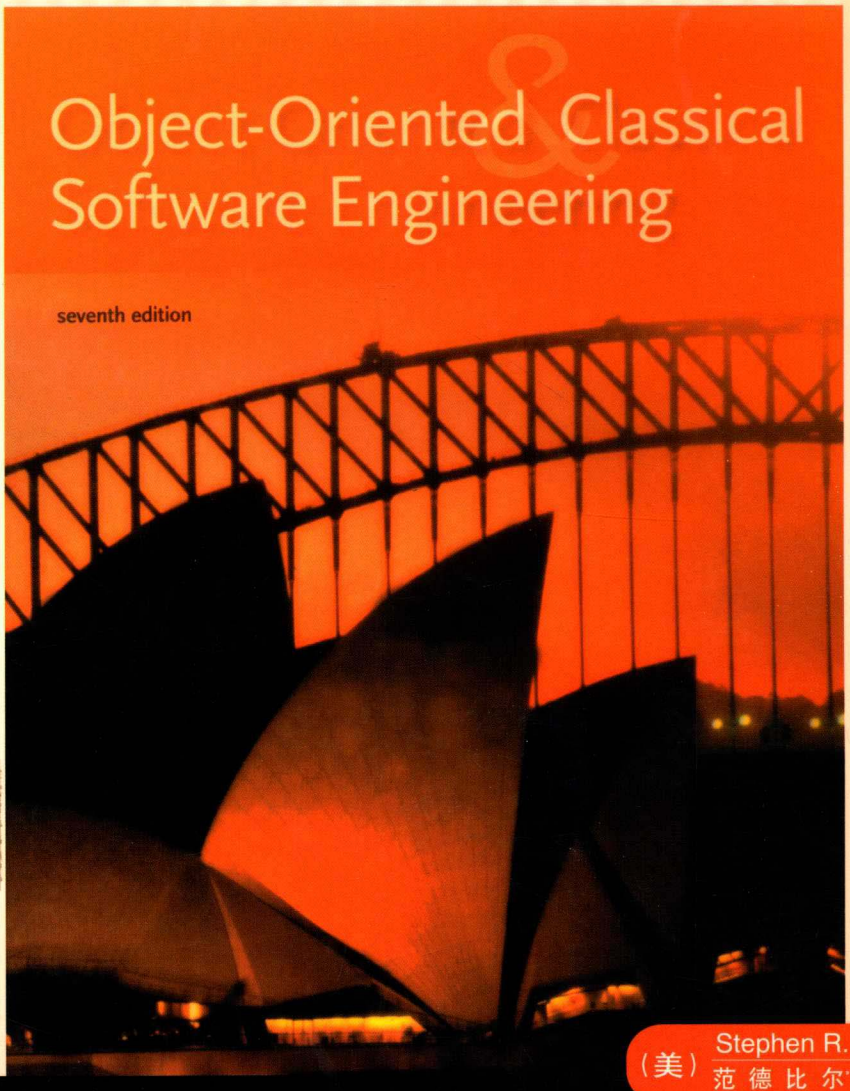
经 典 原 版 书 库

# 软件工程

## 面向对象和传统的方法

（英文版 · 第7版）




（美）Stephen R. Schach 范德比尔特大学 著

机械工业出版社
China Machine Press

Mc Graw Hill Education

经 典 原 版 书 库

# 软件工程

## 面向对象和传统的方法

（英文版 · 第7版）

Object-Oriented Classical Software Engineering

(Seventh Edition)

（美）Stephen R. Schach 范德比尔特大学 著

机械工业出版社
China Machine Press

Stephen R. Schach: Object-Oriented Classical Software Engineering, Seventh Edition (ISBN 13: 978-0-07-319126-3 ISBN 10: 0-07-319126-4).



**本书版权登记号：图字：01-2007-0471**

**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7版）/（美）沙赫（Schach, S. R.）著. -北京：机械工业出版社，2007.2

（经典原版书库）

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

ISBN 978-7-111-20822-8

Ⅰ. 软… Ⅱ. 沙… Ⅲ. 软件工程-高等学校-教材-英文 Ⅳ. TP311.5

中国版本图书馆CIP数据核字（2007）第010250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

北京诚信伟业印刷有限公司印刷 · 新华书店北京发行所发行

2007年2月第1版第1次印刷

170mm × 242mm · 40.25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，不仅擘划了研究的范畴，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 | | | |

The following are registered trademarks:

ADF
Analyst/Designer
Ant
Apache
Apple
AS/400
AT&T
Bachman Product Set
Bell Laboratories
Borland
Bugzilla
Capability Maturity Model
ClearCase
ClearQuest
CMM
Coca-Cola
CORBA
CppUnit
CVS
DB2
Eclipse
e-Components
Emeraude
Enterprise JavaBeans
eServer
Excel
Firefox
Focus
Ford
Foundation Class Library
FoxBASE
GCC
Hewlett-Packard
IBM
IMS/360
Jackpot Source Code Metrics
Java
JBuilder
JUnit
Linux
Lotus 1-2-3
Lucent Technologies
MacApp
Macintosh
Macintosh Toolbox
MacProject
Microsoft
Motif
MS-DOS
MVS/360
Natural
Netscape
*New York Times*
Object C
Objective-C
ObjectWindows Library
1-800-flowers.com
Oracle
Oracle Developer Suite
OS/360
OS/370
OS/VS2
Palm Pilot
Parasoft
Post-it Note
PowerBuilder
PREfix
PREfast
Project
PureCoverage
PVCS
QARun
Rational
Requisite Pro
Rhapsody
Rose
SBC Communications
SilkTest
SLAM
Software through Pictures
Solaris
SourceSafe
SPARCstation
Sun
Sun Enterprise
Sun Microsystems
Sun ONE Studio
System Architect
Together
UNIX
VAX
Visual Component Library
Visual C++
Visual J++
VM/370
VMS
*Wall Street Journal*
WebSphere
Win32
Windows 95
Windows 2000
Windows NT
X11
Xrunner
XUnit
Zip disk
ZIP Code
zSeries

# Preface

The sixth edition of this book, published in 2004, was vastly different to the fifth edition. More than half of the sixth edition consisted of new material, much of it on the Unified Process, and the topics that were retained from the fifth edition were rewritten from a more modern viewpoint.

Many instructors were kind enough to provide feedback on the sixth edition. They approved of the drastic changes I had made in the sixth edition, but suggested that the aim of the seventh edition should be to update the sixth edition, rather than once again make widespread changes. In addition, a number of instructors have asked for a new running case study in Part 2 of the book.

Accordingly, there are two types of changes in the seventh edition:

- The book has been updated throughout. In particular, I have considerably expanded the material on agile processes and open-source software development in Chapters 2 and 4. The references have been updated, and there are new problems in every chapter.
- I have replaced the Osbert Oglesby case study, used to illustrate the techniques of software development in Chapters 10 through 15, with the MSG Foundation case study.

## Features Retained from the Sixth 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.
- 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").
- The material on interoperability, which was removed from Chapter 8 ("Reusability and Portability") in the sixth edition, has not been replaced. In both the fourth and the fifth editions, these sections became hopelessly out of date during the 6 months it took to publish the books. In my opinion, the field is moving too fast to be included in a textbook; an instructor wishing to include interoperability in a software engineering course should obtain up-to-the-minute material from the Internet.

- The new IEEE standard for software project management plans is again presented in Chapter 9 ("Planning and Estimating").
- Chapter 10 ("Requirements"), Chapter 12 ("Object-Oriented Analysis"), and Chapter 13 ("Design") are largely devoted to the workflows (activities) of the Unified Process. For obvious reasons, Chapter 11 ("Classical Analysis") has not been changed, other than the case study.
- The material in Chapter 14 ("Implementation") clearly distinguishes between implementation and integration.
- The importance of postdelivery maintenance is stressed in Chapter 15.
- Chapter 16 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, seven mini case studies highlight specific topics, such as the moving-target problem, stepwise refinement, 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 sixth 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 concerned just 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 still is 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 is retained. 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 people capability maturity model (P–CMM) has been added to the chapter on teams.
- 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 sixth 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 one high-level programming language such as C, C++, Ada, or Java. In addition, the reader is expected to have taken a course in data structures.

## Why the Classical Paradigm Still Is Included

There is now almost unanimous agreement that the object-oriented paradigm is superior to the classical paradigm. Nevertheless, I feel that attempting to eliminate all mention of the classical paradigm is unwise.

First, it is impossible to appreciate why object-oriented technology is superior to classical technology without fully understanding the classical approach and how it differs from the object-oriented approach. For example, the object-oriented paradigm uses an iterative and incremental life-cycle model. To show why such a life-cycle model is needed, it is essential to explain in detail the differences between classical life-cycle models like the waterfall model and the iterative and incremental life-cycle model of the object-oriented paradigm. Therefore, all through the book, I have included material on the classical paradigm so that the student can clearly appreciate the differences between the classical paradigm and the object-oriented paradigm.

The second reason why I have included both paradigms is that technology transfer is a slow process. Notwithstanding the impact of Y2K on accelerating the switch to the object-oriented paradigm, the majority of software organizations still have not yet adopted the object-oriented paradigm. It therefore is likely that many of the students who use this book will be employed by organizations that use classical software engineering techniques. Furthermore, even when an organization uses the object-oriented approach for developing new software, existing software still has to be maintained, and this legacy software is not object oriented. Therefore, excluding classical material would be unfair to many of the students who use this text.

A third reason for including both paradigms is that a student who is employed at an organization considering making the transition to object-oriented technology will be able to advise that organization regarding both the strengths and the weaknesses of the new paradigm. So, as in the previous edition, the classical and object-oriented approaches are compared, contrasted, and analyzed.

## How the Seventh Edition Is Organized

Like the sixth edition of this book, the seventh edition 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 the seventh edition covers most of Chapters 1 through 7 and then starts Part 2 (Chapters 10 through 16). 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 they 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, whenever 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

As in the previous edition, this book has five types of problems. First, there are running object-oriented analysis and design projects at the end of Chapters 10, 12, and 13. 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 13, 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 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 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 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 12.1 |
| Statechart | Section 12.6 |
| Interaction diagram (sequence diagram, collaboration diagram) | Section 12.15 |

Alternatively, Chapter 16 contains an introduction to UML, including material above and beyond what is needed for this book. Chapter 16 may be taught at any time; it does not

depend on material in the first 15 chapters. The topics covered in Chapter 16 are as given in the following table.

| Construct | Section in Which the Corresponding UML Diagram Is Introduced |
|---|---|
| Class diagram, aggregation, multiplicity, composition, generalization, association | Section 16.2 |
| Note | Section 16.3 |
| Use-case diagram | Section 16.4 |
| Stereotype | Section 16.5 |
| Interaction diagram | Section 16.6 |
| Statechart | Section 16.7 |
| Activity diagram | Section 16.8 |
| Package | Section 16.9 |
| Component diagram | Section 16.10 |
| Deployment diagram | Section 16.11 |

## Acknowledgments

I greatly appreciate the constructive criticisms and many helpful suggestions of the reviewers of the six previous editions, including

**Arvin Agah**
*University of Kansas*

**Kiumi Akingbehin**
*University of Michigan, Dearborn*

**Phil Bernhard**
*Clemson University*

**Dan Berry**
*The Technion*

**Don Bickerstaff**
*Eastern Washington University*

**Richard J. Botting**
*California State University, San Bernardino*

**Michael Buckley**
*State University New York, Buffalo*

**Catherine Lowry Campbell**
*New Jersey Institute of Technology*

**James Cardow**
*Air Force Institute of Technology*

**Betty Cheng**
*Michigan State University*

**David Cheriton**
*Stanford University*

**Thaddeus R. Crews, Jr.**
*Western Kentucky University*

**Buster Dunsmore**
*Purdue University*

**Eduardo B. Fernandez**
*Florida Atlantic University*

**Michael Godfrey**
*Cornell University*

**Bob Goldberg**
*IBM*

**Donald Gotterbarn**
*East Tennessee State University*

**Frances Grodzinsky**
*Sacred Heart University*

**Jim Han**
*Florida Atlantic University*

**Scott Hawker**
*University of Alabama*

**Thomas B. Horton**
*Florida Atlantic University*

**Greg Jones**
*Utah State University*

**Peter E. Jones**
*University of Western Australia*
**Gail Kaiser**
*Columbia University*
**Laxmikant V. Kale**
*University of Illinois*
**Helene Kershner**
*University of Buffalo*
**Werner Krandick**
*Drexel University*
**Owen Lavin**
*DePaul University*
**Chung Lee**
*California State Polytechnic, Pomona*
**Richar A. Lejk**
*University of North Carolina, Chapel Hill*
**Bill McCracken**
*Georgia Institute of Technology*
**Susan Mengel**
*Texas Tech University*
**Everald E. Mills**
*Seattle University*
**Fred Mowle**
*Purdue University*
**Donald Needham**
*United States Naval Academy*
**Ron New**
*Johns Hopkins University*
**David Notkin**
*University of Washington*
**Andy Podgurski**
*Case Western Reserve University*
**Hal Render**
*University of Colorado, Colorado Springs*
**David C. Rine**
*George Mason University*
**David S. Rosenblum**
*University of California, Irvine*
**Shmuel Rotenstreich**
*George Washington University*
**Mansur Samadzadeh**
*Oklahoma State University*
**John H. Sayler**
*University of Michigan*
**Wendel Scarborough**
*Azusa Pacific University*
**Bob Schuerman**
*State College, Pennsylvania*
**Gerald B. Sheble**
*Iowa State*
**Fred Strauss**
*Polytechnic University*
**K. C. Tai**
*North Carolina State University*
**Toby Teorey**
*University of Michigan*
**Jie We**
*City University of New York*
**Laurie Werth**
*University of Texas, Austin*
**Lee White**
*Case Western Reserve University*
**David Workman**
*University of Central Florida*
**George W. Zobrist**
*University of Missouri, Rolla*

In addition, special thanks go to the reviewers of this edition, including

**Robert M. Cubert**
*University of Florida*
**Michael Hoffman**
*California State University, Long Beach*
**Saeed S. Monemi**
*California State Polytechnic University, Pomona*
**Linda Ott**
*Michigan Technological University*

**James Purtilo**
*University of Maryland*
**Steven Shaffer**
*Pennsylvania State University*
**Fred Strauss**
*Polytechnic University*

I would like to thank the numerous instructors from all over the world who sent me e-mail regarding the sixth edition. As always, I am exceedingly appreciative of their suggestions, comments, and criticisms. In particular, I thank Professor Michael Haugrud (Minnesota State University Moorhead) for suggesting improvements to the Winburg mini case study. I look forward with anticipation to receiving instructors' feedback on this edition 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 on the sixth edition e-mailed to me by students from all over the world. Special thanks go to Benjamin Polak, a student at the University of Erlangen, Germany, for suggesting improvements to the elevator problem case study. As with the previous editions, I look forward keenly to student feedback on this edition, too.

I warmly thank three individuals who have also made significant contributions to previous editions of this book. First, Kris Irwin once again 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 again a co-author of the *Instructor's Solution Manual* and contributor to the PowerPoint slides.

I turn now to McGraw-Hill. My publisher Alan Apt and my developmental editor Rebecca Olson provided assistance and guidance throughout the project. I thank copyeditor Lucy Mullins for her many helpful suggestions. It was a pleasure to work with marketing manager Michael Weitz, media manager Christina Nelson, production supervisor Kara Kudronowicz, and proofreader Carrie Barker. The striking cover of this book was designed by Michelle Whitaker and Christopher Reese, both of whom I warmly thank. Finally, I particularly wish to thank production manager Lora Kalb for her endless help and for keeping the project on schedule. She invariably went the extra mile—and more.

As always, I enjoyed working with the highly competent compositors at Interactive Composition Corporation. I also thank ICC project manager Deepti Hingle.

Finally, as always, I thank my wife, Sharon, for her 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, she was always understanding, and for this I am most grateful.

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

*Stephen R. Schach*

# Contents