

经 典 原 版 书 库

软件工程

实践者的研究方法

[美] 罗杰 S. 普莱斯曼 布鲁斯 R. 马克西姆 著
Roger S. Pressman Bruce R. Maxim

(英文精编版·第8版)

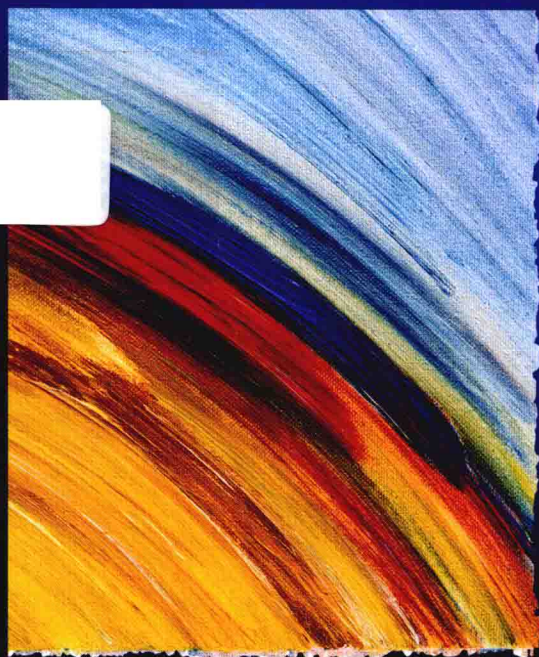
Eighth Edition

Software Engineering

A PRACTITIONER'S APPROACH

Roger S.
PRESSMAN

Bruce R.
MAXIM



经典原版书库

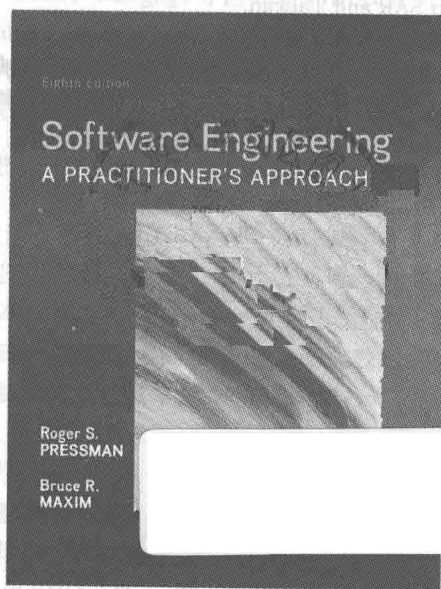
软件工程

实践者的研究方法

(英文精编版·第8版)

Software Engineering

A Practitioner's Approach (Eighth Edition · English Abridgement)



[美] 罗杰 S. 普莱斯曼 布鲁斯 R. 马克西姆 著
 Roger S. Pressman Bruce R. Maxim



机械工业出版社
China Machine Press

图书在版编目 (CIP) 数据

软件工程:实践者的研究方法 (英文精编版·第8版)/(美)普莱斯曼 (Pressman, R. S.), (美)马克西姆 (Maxim, B. R.) 著. —北京:机械工业出版社, 2015.11
(经典原版书库)

书名原文: Software Engineering: A Practitioner's Approach, Eighth Edition

ISBN 978-7-111-49931-2

I. 软… II. ①普… ②马… III. 软件工程—英文 IV. TP311.5

中国版本图书馆 CIP 数据核字 (2015) 第 073981 号

本书版权登记号: 图字: 01-2014-7783

Roger S. Pressman, Bruce R. Maxim: Software Engineering: A Practitioner's Approach, Eighth Edition (ISBN 978-0-07-802212-8).

Copyright © 2015 by McGraw-Hill Education.

All rights reserved. No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including without limitation photocopying, recording, taping, or any database, information or retrieval system, without the prior written permission of the publisher.

This authorized English Abridgement is jointly published by McGraw-Hill Education and China Machine Press. This edition is authorized for sale in the People's Republic of China only, excluding Hong Kong, Macao SAR and Taiwan.

Copyright © 2016 by McGraw-Hill Education and China Machine Press.

本授权英文影印删减版由麦格劳-希尔(亚洲)教育出版公司和机械工业出版社合作出版。此版本仅限在中华人民共和国境内(不包括香港、澳门特别行政区及台湾)销售。未经许可之出口,视为违反著作权法,将受法律之制裁。

未经出版者预先书面许可,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 McGraw-Hill 公司防伪标签,无标签者不得销售。

版权所有,侵权必究。

出版发行: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

责任校对:殷虹

印刷:北京诚信伟业印刷有限公司

版次:2016年1月第1版第1次印刷

开本:186mm×240mm 1/16

印张:37.5

书号:ISBN 978-7-111-49931-2

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

投稿热线:(010) 88379604

购书热线:(010) 68326294 88379649 68995259

读者信箱:hzjsj@hzbook.com

版权所有·侵权必究

封底无防伪标均为盗版

本书法律顾问:北京大成律师事务所 韩光/邹晓东

出版者的话

文艺复兴以来，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；也正是这样的优势，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, Brian W. Kernighan, Dennis Ritchie, Jim Gray, Alfred 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相助，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，有的还专门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，我们的联系方式如下：

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



华章科技图书出版中心

ADAPTER'S FOREWORD

Purpose

The original of this book is an excellent work of Dr. Pressman and Dr. Maxim. The 8th edition emphasizes on the new software engineering process and techniques, and is definitive on basically all the subjects that are listed by the Software Engineering Body of Knowledge (SWEBOK, <http://www.swebok.org/>). The extensive examples, references and online exercises are also quite helpful to the students. There are numbers of universities and colleges in China that are taking this book as their textbooks and keep tracking through its different versions.

As the idea of bilingual teaching is promoted in the higher education institutes in China, more and more schools are getting interested in this book. However, over nine hundred pages of the original book make it too difficult for a Chinese student with an average English reading ability to comprehend the fundamentals of software engineering in a semester. In order to introduce this book to more Chinese college students, a compression of the original book is made to better fit the book into the syllabus of an undergraduate course with only the core contents, and to reduce the students' reading load.

What's Compressed

We noticed that MobileApps and security engineering are the two very important new concepts added into this version. However due to the limited class time we finally decide to keep only the security engineering and leave mobile applications to the graduate courses. Similar to what we have done to the 7th edition, basically we would concentrate on the fundamental concepts of common frameworks, and leave WebApps, MobileApps, metrics and measurements, and some advanced topics to the graduate level courses.

The chapters that we would take to the graduate level and hence would be excluded in the compressed version are: Chapter 16 (Pattern-based Design), Chapter 17 (WebApp Design), Chapter 18 (MobileApp Design), Chapter 20 (Review Techniques), Chapter 25 (Testing Web Applications), Chapter 26 (Testing MobileApps), Chapter 28 (Formal Modeling and Verification), Chapter 36 (Maintenance and Reengineering), and Part Five (Advanced Topics). Chapter 30 (Product Metrics) is considered advanced as well, except that the first Section 30.1 (A Framework for Product Metrics) is suitable for the undergraduates to learn. Thus we plan to move Section 30.1 to Chapter 21 (Software Quality Assurance) and add it to be Section 21.10.

Other compressions are made to keep everything consistent with the compressed 7th edition:

1. All the sections and sub-sections related to WebApp and MobileApps will be removed. They are: Section 11.5 (Requirements Modeling for Web and Mobile Apps), Section 14.5 (Component-level Design for Mobile Apps), Section 15.5 (WebApp and Mobile

Interface Design), Sections 22.5-22.6 (Test Strategies for WebApps, and Test Strategies for MobileApps), Section 29.4 (Configuration Management for Web and MobileApps), Section 32.2.6 (WebApp Project Metrics), and Section 34.5.4 (Scheduling for WebApp and Mobile Projects).

2. In Chapter 3, Section 3.5 (Process Assessment and Improvement) and Section 4.4-4.5 (Personal and Team Process Models, Process Technology) are considered as the advanced topics and hence will not appear in the compressed version.
3. The content of Chapter 7 (Principles that Guide Practice) will eventually be introduced in the following chapters. We would remove this chapter to save the students some reading time.
4. In Chapter 8, Sections 8.2.5-8.2.6 (Nonfunctional Requirements, and Traceability), and Sections 8.6-8.8 (Negotiating Requirements, Requirements Monitoring, and Validating Requirements) are considered as the advanced topics and hence will not appear in the compressed version.
5. In Chapter 23, Sections (and sub-sections) 23.4.4 (Graph Matrices), 23.6.1 (Graph-Based Testing Methods), 23.6.4 (Orthogonal Array Testing), 23.8-23.10 (Testing Documentation and Help Facilities, Testing for Real-time Systems, and Patterns for Software Testing) are considered as the advanced topics and hence will not appear in the compressed version.
6. We believe that it would be enough to simply introduce some basic concepts on metrics and estimation to the undergraduates. Hence we plan to remove Sections 32.4-32.6 (Metrics for Process and Projects – Integrating Metrics within the Process, Metrics for Small Organizations, and Establishing a Software Metrics Program), and Sections 33.9-33.10 (Specialized Estimation Techniques and the Make/Buy Decision).

Acknowledgments

We feel grateful to Roger S. Pressman and Bruce R. Maxim, the authors of the original book, and McGraw Hill, the original publisher, for allowing us to compress the original 941-page book into about six hundred pages. It is their understanding and generosity that make it possible for more Chinese students to enjoy this distinguished book. Since the compression is made by simply removing some chapters or sections, we hope to keep as far as possible the elegance of Dr. Pressman and Dr. Maxim's writing style.

Thanks to everyone at China Machine Press who have put in great effort to make this kind of cooperation possible.

The compression is made according to my own experience in teaching and practicing in software engineering. It is certainly wild open for discussions and suggestions on further improvements. Your comments are important to us, and would be very much appreciated.

Yue Chen
Zhejiang University
(chenyue@cs.zju.edu.cn)

PREFACE

When computer software succeeds—when it meets the needs of the people who use it, when it performs flawlessly over a long period of time, when it is easy to modify and even easier to use—it can and does change things for the better. But when software fails—when its users are dissatisfied, when it is error prone, when it is difficult to change and even harder to use—bad things can and do happen. We all want to build software that makes things better, avoiding the bad things that lurk in the shadow of failed efforts. To succeed, we need discipline when software is designed and built. We need an engineering approach.

It has been almost three and a half decades since the first edition of this book was written. During that time, software engineering has evolved from an obscure idea practiced by a relatively small number of zealots to a legitimate engineering discipline. Today, it is recognized as a subject worthy of serious research, conscientious study, and tumultuous debate. Throughout the industry, software engineer has replaced programmer as the job title of preference. Software process models, software engineering methods, and software tools have been adopted successfully across a broad spectrum of industry segments.

Although managers and practitioners alike recognize the need for a more disciplined approach to software, they continue to debate the manner in which discipline is to be applied. Many individuals and companies still develop software haphazardly, even as they build systems to service today's most advanced technologies. Many professionals and students are unaware of modern methods. And as a result, the quality of the software that we produce suffers, and bad things happen. In addition, debate and controversy about the true nature of the software engineering approach continue. The status of software engineering is a study in contrasts. Attitudes have changed, progress has been made, but much remains to be done before the discipline reaches full maturity.

The eighth edition of *Software Engineering: A Practitioner's Approach* is intended to serve as a guide to a maturing engineering discipline. The eighth edition, like the seven editions that preceded it, is intended for both students and practitioners, retaining its appeal as a guide to the industry professional and a comprehensive introduction to the student at the upper-level undergraduate or first-year graduate level.

The eighth edition is considerably more than a simple update. The book has been revised and restructured to improve pedagogical flow and emphasize new and important software engineering processes and practices. In addition, we have further enhanced the popular “support system” for the book, providing a comprehensive set of student, instructor, and professional resources to complement the content of the book. These resources are presented as part of a website (www.mhhe.com/pressman) specifically designed for *Software Engineering: A Practitioner's Approach*.

The Eighth Edition. The 26 chapters of the eighth edition are organized into four parts. This organization better compartmentalizes topics and assists instructors who may not have the time to complete the entire book in one term.

Part 1, *The Process*, presents a variety of different views of software process, considering all important process models and addressing the debate between prescriptive and agile process philosophies. Part 2, *Modeling*, presents analysis and design methods with an emphasis on object-oriented techniques and UML modeling. Part 3, *Quality Management*, presents the concepts, procedures, and methods that enable a software team to assess software quality, conduct SQA procedures, and apply an effective testing strategy and tactics. Part 4, *Managing Software Projects*, presents topics that are relevant to those who plan, manage, and control a software development project. Continuing in the tradition of past editions, a series of sidebars is used throughout the book to present the trials and tribulations of a (fictional) software team and to provide supplementary materials about methods and tools that are relevant to chapter topics.

Acknowledgments. Special thanks go to Tim Lethbridge of the University of Ottawa who assisted us in the development of UML and OCL examples, and developed the case study that accompanies this book, and Dale Skrien of Colby College, who developed the UML tutorial in Appendix 1. Their assistance and comments were invaluable. In addition, we'd like to thank Austin Krauss, Senior Software Engineer at Treyarch, for providing insight into software development in the video game industry. We also wish to thank the reviewers of the eighth edition: Manuel E. Bermudez, University of Florida; Scott DeLoach, Kansas State University; Alex Liu, Michigan State University; and Dean Mathias, Utah State University. Their in-depth comments and thoughtful criticism have helped us make this a much better book.

Special Thanks. BRM: I am grateful to have had the opportunity to work with Roger on the eighth edition of this book. During the time I have been working on this book my son Benjamin shipped his first MobileApp and my daughter Katherine launched her interior design career. I am quite pleased to see the adults they have become. I am very grateful to my wife, Norma, for the enthusiastic support she has given me as I filled my free time with working on this book.

RSP: As the editions of this book have evolved, my sons, Mathew and Michael, have grown from boys to men. Their maturity, character, and success in the real world have been an inspiration to me. Nothing has filled me with more pride. They now have children of their own, Maya and Lily, who start still another generation. Both girls are already wizards on mobile computing devices. Finally, to my wife Barbara, my love and thanks for tolerating the many, many hours in the office and encouraging still another edition of "the book."

Roger S. Pressman

Bruce R. Maxim

ABOUT THE AUTHORS

Roger S. Pressman is an internationally recognized consultant and author in software engineering. For more than four decades, he has worked as a software engineer, a manager, a professor, an author, a consultant, and an entrepreneur.

Dr. Pressman is president of R. S. Pressman & Associates, Inc., a consulting firm that specializes in helping companies establish effective software engineering practices. Over the years he has developed a set of techniques and tools that improve software engineering practice. He is also the founder of Teslaccessories, LLC, a start-up manufacturing company that specializes in custom products for the Tesla Model S electric vehicle.

Dr. Pressman is the author of nine books, including two novels, and many technical and management papers. He has been on the editorial boards of *IEEE Software* and *The Cutter IT Journal* and was editor of the “Manager” column in *IEEE Software*.

Dr. Pressman is a well-known speaker, keynoting a number of major industry conferences. He has presented tutorials at the International Conference on Software Engineering and at many other industry meetings. He has been a member of the ACM, IEEE, and Tau Beta Pi, Phi Kappa Phi, Eta Kappa Nu, and Pi Tau Sigma.

Bruce R. Maxim has worked as a software engineer, project manager, professor, author, and consultant for more than thirty years. His research interests include software engineering, human computer interaction, game design, social media, artificial intelligence, and computer science education.

Dr. Maxim is associate professor of computer and information science at the University of Michigan—Dearborn. He established the GAME Lab in the College of Engineering and Computer Science. He has published a number of papers on computer algorithm animation, game development, and engineering education. He is coauthor of a best-selling introductory computer science text. Dr. Maxim has supervised several hundred industry-based software development projects as part of his work at UM-Dearborn.

Dr. Maxim’s professional experience includes managing research information systems at a medical school, directing instructional computing for a medical campus, and working as a statistical programmer. Dr. Maxim served as the chief technology officer for a game development company.

Dr. Maxim was the recipient of several distinguished teaching awards and a distinguished community service award. He is a member of Sigma Xi, Upsilon Pi Epsilon, Pi Mu Epsilon, Association of Computing Machinery, IEEE Computer Society, American Society for Engineering Education, Society of Women Engineers, and International Game Developers Association.

TABLE OF CONTENTS

CHAPTER 1 THE NATURE OF SOFTWARE 1

1.1	The Nature of Software	3
1.1.1	Defining Software	4
1.1.2	Software Application Domains	6
1.1.3	Legacy Software	7
1.2	The Changing Nature of Software	9
1.2.1	WebApps	9
1.2.2	Mobile Applications	9
1.2.3	Cloud Computing	10
1.2.4	Product Line Software	11
	PROBLEMS AND POINTS TO PONDER	12
	FURTHER READINGS AND INFORMATION SOURCES	12

CHAPTER 2 SOFTWARE ENGINEERING 14

2.1	Defining the Discipline	15
2.2	The Software Process	16
2.2.1	The Process Framework	17
2.2.2	Umbrella Activities	18
2.2.3	Process Adaptation	18
2.3	Software Engineering Practice	19
2.3.1	The Essence of Practice	19
2.3.2	General Principles	21
2.4	Software Development Myths	23
2.5	How It All Starts	26
	PROBLEMS AND POINTS TO PONDER	27
	FURTHER READINGS AND INFORMATION SOURCES	27

PART ONE THE SOFTWARE PROCESS 29

CHAPTER 3 SOFTWARE PROCESS STRUCTURE 30

3.1	A Generic Process Model	31
3.2	Defining a Framework Activity	32
3.3	Identifying a Task Set	34
3.4	Process Patterns	35
	PROBLEMS AND POINTS TO PONDER	37
	FURTHER READINGS AND INFORMATION SOURCES	38

CHAPTER 4 PROCESS MODELS 39

4.1 Prescriptive Process Models 40
 4.1.1 The Waterfall Model 40
 4.1.2 Incremental Process Models 42
 4.1.3 Evolutionary Process Models 44
 4.1.4 Concurrent Models 48
 4.1.5 A Final Word on Evolutionary Processes 50
 4.2 Specialized Process Models 51
 4.2.1 Component-Based Development 52
 4.2.2 The Formal Methods Model 52
 4.2.3 Aspect-Oriented Software Development 53
 4.3 The Unified Process 54
 4.3.1 A Brief History 55
 4.3.2 Phases of the Unified Process 55
 4.4 Product and Process 57
 PROBLEMS AND POINTS TO PONDER 59
 FURTHER READINGS AND INFORMATION SOURCES 59

CHAPTER 5 AGILE DEVELOPMENT 60

5.1 What Is Agility? 62
 5.2 Agility and the Cost of Change 62
 5.3 What Is an Agile Process? 63
 5.3.1 Agility Principles 64
 5.3.2 The Politics of Agile Development 65
 5.4 Extreme Programming 66
 5.4.1 The XP Process 66
 5.4.2 Industrial XP 69
 5.5 Other Agile Process Models 71
 5.5.1 Scrum 72
 5.5.2 Dynamic Systems Development Method 73
 5.5.3 Agile Modeling 74
 5.5.4 Agile Unified Process 76
 5.6 A Tool Set for the Agile Process 77
 PROBLEMS AND POINTS TO PONDER 78
 FURTHER READINGS AND INFORMATION SOURCES 79

CHAPTER 6 HUMAN ASPECTS OF SOFTWARE ENGINEERING 81

6.1 Characteristics of a Software Engineer 82
 6.2 The Psychology of Software Engineering 83
 6.3 The Software Team 84
 6.4 Team Structures 86
 6.5 Agile Teams 87
 6.5.1 The Generic Agile Team 87
 6.5.2 The XP Team 88
 6.6 The Impact of Social Media 89
 6.7 Software Engineering Using the Cloud 91
 6.8 Collaboration Tools 92
 6.9 Global Teams 93

PROBLEMS AND POINTS TO PONDER 94

FURTHER READINGS AND INFORMATION SOURCES 95

PART TWO MODELING 97**CHAPTER 7 UNDERSTANDING REQUIREMENTS 98**

7.1	Requirements Engineering	99
7.2	Establishing the Groundwork	105
7.2.1	Identifying Stakeholders	106
7.2.2	Recognizing Multiple Viewpoints	106
7.2.3	Working toward Collaboration	107
7.2.4	Asking the First Questions	107
7.3	Eliciting Requirements	108
7.3.1	Collaborative Requirements Gathering	109
7.3.2	Quality Function Deployment	112
7.3.3	Usage Scenarios	112
7.3.4	Elicitation Work Products	113
7.3.5	Agile Requirements Elicitation	114
7.3.6	Service-Oriented Methods	114
7.4	Developing Use Cases	115
7.5	Building the Analysis Model	120
7.5.1	Elements of the Analysis Model	120
7.5.2	Analysis Patterns	123
7.5.3	Agile Requirements Engineering	124
7.5.4	Requirements for Self-Adaptive Systems	124
7.6	Avoiding Common Mistakes	125
	PROBLEMS AND POINTS TO PONDER	125
	FURTHER READINGS AND OTHER INFORMATION SOURCES	126

CHAPTER 8 REQUIREMENTS MODELING: SCENARIO-BASED METHODS 128

8.1	Requirements Analysis	129
8.1.1	Overall Objectives and Philosophy	130
8.1.2	Analysis Rules of Thumb	131
8.1.3	Domain Analysis	132
8.1.4	Requirements Modeling Approaches	133
8.2	Scenario-Based Modeling	135
8.2.1	Creating a Preliminary Use Case	135
8.2.2	Refining a Preliminary Use Case	138
8.2.3	Writing a Formal Use Case	139
8.3	UML Models That Supplement the Use Case	141
8.3.1	Developing an Activity Diagram	142
8.3.2	Swimlane Diagrams	143
	PROBLEMS AND POINTS TO PONDER	144
	FURTHER READINGS AND INFORMATION SOURCES	145

CHAPTER 9 REQUIREMENTS MODELING: CLASS-BASED METHODS 146

- 9.1 Identifying Analysis Classes 147
- 9.2 Specifying Attributes 150
- 9.3 Defining Operations 151
- 9.4 Class-Responsibility-Collaborator Modeling 154
- 9.5 Associations and Dependencies 160
- 9.6 Analysis Packages 161

PROBLEMS AND POINTS TO PONDER 162

FURTHER READINGS AND INFORMATION SOURCES 163

CHAPTER 10 REQUIREMENTS MODELING: BEHAVIOR, PATTERNS, AND WEB/MOBILE APPS 164

- 10.1 Creating a Behavioral Model 165
- 10.2 Identifying Events with the Use Case 165
- 10.3 State Representations 166
- 10.4 Patterns for Requirements Modeling 169
 - 10.4.1 Discovering Analysis Patterns 170
 - 10.4.2 A Requirements Pattern Example: Actuator-Sensor 171

PROBLEMS AND POINTS TO PONDER 175

FURTHER READINGS AND INFORMATION SOURCES 176

CHAPTER 11 DESIGN CONCEPTS 177

- 11.1 Design within the Context of Software Engineering 178
- 11.2 The Design Process 181
 - 11.2.1 Software Quality Guidelines and Attributes 181
 - 11.2.2 The Evolution of Software Design 183
- 11.3 Design Concepts 184
 - 11.3.1 Abstraction 185
 - 11.3.2 Architecture 185
 - 11.3.3 Patterns 186
 - 11.3.4 Separation of Concerns 187
 - 11.3.5 Modularity 187
 - 11.3.6 Information Hiding 188
 - 11.3.7 Functional Independence 189
 - 11.3.8 Refinement 190
 - 11.3.9 Aspects 190
 - 11.3.10 Refactoring 191
 - 11.3.11 Object-Oriented Design Concepts 191
 - 11.3.12 Design Classes 192
 - 11.3.13 Dependency Inversion 194
 - 11.3.14 Design for Test 195
- 11.4 The Design Model 196
 - 11.4.1 Data Design Elements 197
 - 11.4.2 Architectural Design Elements 197
 - 11.4.3 Interface Design Elements 198
 - 11.4.4 Component-Level Design Elements 200
 - 11.4.5 Deployment-Level Design Elements 201

PROBLEMS AND POINTS TO PONDER 202

FURTHER READINGS AND INFORMATION SOURCES 203

CHAPTER 12 ARCHITECTURAL DESIGN 204

12.1	Software Architecture	205
12.1.1	What Is Architecture?	205
12.1.2	Why Is Architecture Important?	206
12.1.3	Architectural Descriptions	207
12.1.4	Architectural Decisions	208
12.2	Architectural Genres	209
12.3	Architectural Styles	210
12.3.1	A Brief Taxonomy of Architectural Styles	210
12.3.2	Architectural Patterns	215
12.3.3	Organization and Refinement	215
12.4	Architectural Considerations	216
12.5	Architectural Decisions	218
12.6	Architectural Design	219
12.6.1	Representing the System in Context	219
12.6.2	Defining Archetypes	221
12.6.3	Refining the Architecture into Components	222
12.6.4	Describing Instantiations of the System	224
12.6.5	Architectural Design for Web Apps	225
12.6.6	Architectural Design for Mobile Apps	226
12.7	Assessing Alternative Architectural Designs	226
12.7.1	Architectural Description Languages	228
12.7.2	Architectural Reviews	229
12.8	Lessons Learned	230
12.9	Pattern-based Architecture Review	230
12.10	Architecture Conformance Checking	231
12.11	Agility and Architecture	232
	PROBLEMS AND POINTS TO PONDER	234
	FURTHER READINGS AND INFORMATION SOURCES	234

CHAPTER 13 COMPONENT-LEVEL DESIGN 236

13.1	What Is a Component?	237
13.1.1	An Object-Oriented View	237
13.1.2	The Traditional View	239
13.1.3	A Process-Related View	242
13.2	Designing Class-Based Components	242
13.2.1	Basic Design Principles	243
13.2.2	Component-Level Design Guidelines	246
13.2.3	Cohesion	247
13.2.4	Coupling	249
13.3	Conducting Component-Level Design	250
13.4	Component-Level Design for WebApps	256
13.4.1	Content Design at the Component Level	257
13.4.2	Functional Design at the Component Level	257
13.5	Designing Traditional Components	257
13.6	Component-Based Development	258
13.6.1	Domain Engineering	259
13.6.2	Component Qualification, Adaptation, and Composition	259
13.6.3	Architectural Mismatch	261
13.6.4	Analysis and Design for Reuse	262

13.6.5 Classifying and Retrieving Components 262

PROBLEMS AND POINTS TO PONDER 264

FURTHER READINGS AND INFORMATION SOURCES 264

CHAPTER 14 USER INTERFACE DESIGN 266

14.1 The Golden Rules 267

14.1.1 Place the User in Control 267

14.1.2 Reduce the User's Memory Load 268

14.1.3 Make the Interface Consistent 270

14.2 User Interface Analysis and Design 271

14.2.1 Interface Analysis and Design Models 271

14.2.2 The Process 272

14.3 Interface Analysis 274

14.3.1 User Analysis 274

14.3.2 Task Analysis and Modeling 275

14.3.3 Analysis of Display Content 280

14.3.4 Analysis of the Work Environment 280

14.4 Interface Design Steps 281

14.4.1 Applying Interface Design Steps 281

14.4.2 User Interface Design Patterns 283

14.4.3 Design Issues 284

14.5 Design Evaluation 286

PROBLEMS AND POINTS TO PONDER 288

FURTHER READINGS AND INFORMATION SOURCES 289

PART THREE QUALITY MANAGEMENT 291**CHAPTER 15 QUALITY CONCEPTS 292**

15.1 What Is Quality? 293

15.2 Software Quality 294

15.2.1 Garvin's Quality Dimensions 295

15.2.2 McCall's Quality Factors 296

15.2.3 ISO 9126 Quality Factors 298

15.2.4 Targeted Quality Factors 298

15.2.5 The Transition to a Quantitative View 300

15.3 The Software Quality Dilemma 300

15.3.1 "Good Enough" Software 301

15.3.2 The Cost of Quality 302

15.3.3 Risks 304

15.3.4 Negligence and Liability 305

15.3.5 Quality and Security 305

15.3.6 The Impact of Management Actions 306

15.4 Achieving Software Quality 307

15.4.1 Software Engineering Methods 307

15.4.2 Project Management Techniques 307

15.4.3 Quality Control 307

15.4.4 Quality Assurance 308

PROBLEMS AND POINTS TO PONDER 308

FURTHER READINGS AND INFORMATION SOURCES 309

CHAPTER 16 SOFTWARE QUALITY ASSURANCE 310

- 16.1 Background Issues 311
- 16.2 Elements of Software Quality Assurance 312
- 16.3 SQA Processes and Product Characteristics 314
- 16.4 SQA Tasks, Goals, and Metrics 314
 - 16.4.1 SQA Tasks 315
 - 16.4.2 Goals, Attributes, and Metrics 316
- 16.5 Formal Approaches to SQA 318
- 16.6 Statistical Software Quality Assurance 318
 - 16.6.1 A Generic Example 319
 - 16.6.2 Six Sigma for Software Engineering 320
- 16.7 Software Reliability 321
 - 16.7.1 Measures of Reliability and Availability 321
 - 16.7.2 Software Safety 322
- 16.8 The ISO 9000 Quality Standards 323
- 16.9 The SQA Plan 325
- 16.10 A Framework for Product Metrics 325
 - 16.10.1 Measures, Metrics, and Indicators 325
 - 16.10.2 The Challenge of Product Metrics 326
 - 16.10.3 Measurement Principles 327
 - 16.10.4 Goal-Oriented Software Measurement 327
 - 16.10.5 The Attributes of Effective Software Metrics 328
- PROBLEMS AND POINTS TO PONDER 329
- FURTHER READINGS AND INFORMATION SOURCES 330

CHAPTER 17 SOFTWARE TESTING STRATEGIES 332

- 17.1 A Strategic Approach to Software Testing 332
 - 17.1.1 Verification and Validation 334
 - 17.1.2 Organizing for Software Testing 334
 - 17.1.3 Software Testing Strategy—The Big Picture 335
 - 17.1.4 Criteria for Completion of Testing 338
- 17.2 Strategic Issues 338
- 17.3 Test Strategies for Conventional Software 339
 - 17.3.1 Unit Testing 339
 - 17.3.2 Integration Testing 341
- 17.4 Test Strategies for Object-Oriented Software 347
 - 17.4.1 Unit Testing in the OO Context 347
 - 17.4.2 Integration Testing in the OO Context 347
- 17.5 Validation Testing 348
 - 17.5.1 Validation-Test Criteria 348
 - 17.5.2 Configuration Review 349
 - 17.5.3 Alpha and Beta Testing 349
- 17.6 System Testing 350
 - 17.6.1 Recovery Testing 350
 - 17.6.2 Security Testing 351
 - 17.6.3 Stress Testing 351
 - 17.6.4 Performance Testing 352
 - 17.6.5 Deployment Testing 352
- 17.7 The Art of Debugging 353
 - 17.7.1 The Debugging Process 353

17.7.2	Psychological Considerations	354
17.7.3	Debugging Strategies	355
17.7.4	Correcting the Error	357

PROBLEMS AND POINTS TO PONDER 357

FURTHER READINGS AND INFORMATION SOURCES 358

CHAPTER 18 TESTING CONVENTIONAL APPLICATIONS 360

18.1	Software Testing Fundamentals	361
18.2	Internal and External Views of Testing	363
18.3	White-Box Testing	364
18.4	Basis Path Testing	364
	18.4.1 Flow Graph Notation	364
	18.4.2 Independent Program Paths	366
	18.4.3 Deriving Test Cases	368
18.5	Control Structure Testing	370
18.6	Black-Box Testing	372
	18.6.1 Equivalence Partitioning	372
	18.6.2 Boundary Value Analysis	373
18.7	Model-Based Testing	374

PROBLEMS AND POINTS TO PONDER 375

FURTHER READINGS AND INFORMATION SOURCES 375

CHAPTER 19 TESTING OBJECT-ORIENTED APPLICATIONS 377

19.1	Broadening the View of Testing	378
19.2	Testing OOA and OOD Models	379
	19.2.1 Correctness of OOA and OOD Models	379
	19.2.2 Consistency of Object-Oriented Models	380
19.3	Object-Oriented Testing Strategies	382
	19.3.1 Unit Testing in the OO Context	382
	19.3.2 Integration Testing in the OO Context	383
	19.3.3 Validation Testing in an OO Context	383
19.4	Object-Oriented Testing Methods	383
	19.4.1 The Test-Case Design Implications of OO Concepts	384
	19.4.2 Applicability of Conventional Test-Case Design Methods	385
	19.4.3 Fault-Based Testing	385
	19.4.4 Scenario-Based Test Design	386
19.5	Testing Methods Applicable at the Class Level	386
	19.5.1 Random Testing for OO Classes	386
	19.5.2 Partition Testing at the Class Level	387
19.6	Interclass Test-Case Design	388
	19.6.1 Multiple Class Testing	388
	19.6.2 Tests Derived from Behavior Models	390

PROBLEMS AND POINTS TO PONDER 391

FURTHER READINGS AND INFORMATION SOURCES 392

CHAPTER 20 SECURITY ENGINEERING 393

20.1	Analyzing Security Requirements	394
20.2	Security and Privacy in an Online World	395