HZ BOOKS 华章教育
Mc Graw Hill Education

经 典 原 版 书 库

软件工程

实践者的研究方法

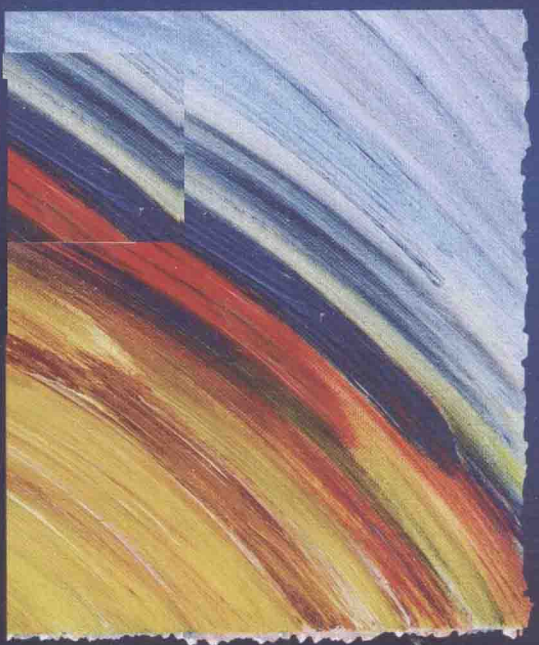
（美）Roger S. Pressman Bruce R. Maxim 著

（英文版·第8版）

Eighth Edition

Software Engineering

A PRACTITIONER'S APPROACH



Roger S. PRESSMAN

Bruce R. MAXIM

机械工业出版社
China Machine Press

经 典 原 版 书 库

软件工程

实践者的研究方法

（英文版·第8版）

Software Engineering

A Practitioner's Approach (Eighth Edition)

（美）Roger S. Pressman Bruce R. Maxim 著

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，2015.1
（经典原版书库）
书名原文：Software Engineering: A Practitioner's Approach, Eighth Edition

ISBN 978-7-111-48950-4

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（2014）第307858号

本书版权登记号：图字：01-2014-6189

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：董纪丽
印　　刷：北京市荣盛彩色印刷有限公司　　版　　次：2015年2月第1版第1次印刷
开　　本：186mm×240mm　1/16　　印　　张：60.5
书　　号：ISBN 978-7-111-48950-4　　定　　价：119.00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

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，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

华章科技图书出版中心

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 39 chapters of the eighth edition are organized into five 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. Pattern-based design and design for Web and mobile applications are also considered. Part 3, *Quality Management*, presents the concepts, procedures, techniques, and methods that enable a software team to assess software quality, review software engineering work products, conduct SQA procedures, and apply an effective testing strategy and tactics. In addition, formal modeling and verification methods are also considered. Part 4, *Managing Software Projects*, presents topics that are relevant to those who plan, manage, and control a software development project. Part 5, *Advanced Topics*, considers software process improvement and software engineering trends. 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.

The five-part organization of the eighth edition enables an instructor to "cluster" topics based on available time and student need. An entire one-term course can be built around one or more of the five parts. A software engineering survey course would select chapters from all five parts. A software engineering course that emphasizes analysis and design would select topics from Parts 1 and 2. A testing-oriented software engineering course would select topics from Parts 1 and 3, with a brief foray into Part 2. A "management course" would stress Parts 1 and 4. By organizing the eighth edition in this way, we have attempted to provide an instructor with a number of teaching options. In every case the content of the eighth edition is complemented by the following elements of the *SEPA, 8/e Support System*.

Student Resources. A wide variety of student resources includes an extensive online learning center encompassing chapter-by-chapter study guides, practice quizzes, problem solutions, and a variety of Web-based resources including software engineering checklists, an evolving collection of "tiny tools," a comprehensive case study, work product templates, and many other resources. In addition, over 1,000 categorized *Web References* allow a student to explore software engineering in greater detail and a *Reference Library* with links to more than 500 downloadable papers provides an in-depth source of advanced software engineering information.

Instructor Resources. A broad array of instructor resources has been developed to supplement the eighth edition. These include a complete online *Instructor's Guide* (also downloadable) and supplementary teaching materials including a complete set of more than 700 *PowerPoint Slides* that may be used for lectures, and a test bank. Of course, all resources available for students (e.g, tiny tools, the Web References, the downloadable Reference Library) and professionals are also available.

The *Instructor's Guide for Software Engineering: A Practitioner's Approach* presents suggestions for conducting various types of software engineering courses, recommendations for a variety of software projects to be conducted in conjunction with a course, solutions to selected problems, and a number of useful teaching aids.

Professional Resources. A collection of resources available to industry practitioners (as well as students and faculty) includes outlines and samples of software engineering documents and other work products, a useful set of software engineering checklists,

a catalog of software engineering tools, a comprehensive collection of Web-based resources, and an "adaptable process model" that provides a detailed task breakdown of the software engineering process.

McGraw-Hill Connect® Computer Science provides online presentation, assignment, and assessment solutions. It connects your students with the tools and resources they'll need to achieve success. With Connect **Computer Science** you can deliver assignments, quizzes, and tests online. A robust set of questions and activities are presented and aligned with the textbook's learning outcomes. As an instructor, you can edit existing questions and author entirely new problems. Integrate grade reports easily with Learning Management Systems (LMS), such as WebCT and Blackboard—and much more. ConnectPlus® **Computer Science** provides students with all the advantages of Connect **Computer Science**, plus 24/7 online access to a media-rich eBook, allowing seamless integration of text, media, and assessments. To learn more, visit **www.mcgrawhillconnect.com**

McGraw-Hill LearnSmart® is available as a standalone product or an integrated feature of McGraw-Hill Connect **Computer Science**. It is an adaptive learning system designed to help students learn faster, study more efficiently, and retain more knowledge for greater success. LearnSmart assesses a student's knowledge of course content through a series of adaptive questions. It pinpoints concepts the student does not understand and maps out a personalized study plan for success. This innovative study tool also has features that allow instructors to see exactly what students have accomplished and a built-in assessment tool for graded assignments. Visit the following site for a demonstration. **www.mhlearnsmart.com**

Powered by the intelligent and adaptive LearnSmart engine, **SmartBook™** is the first and only continuously adaptive reading experience available today. Distinguishing what students know from what they don't, and honing in on concepts they are most likely to forget, SmartBook personalizes content for each student. Reading is no longer a passive and linear experience but an engaging and dynamic one, where students are more likely to master and retain important concepts, coming to class better prepared. SmartBook includes powerful reports that identify specific topics and learning objectives students need to study.

When coupled with its online support system, the eighth edition of *Software Engineering: A Practitioner's Approach,* provides flexibility and depth of content that cannot be achieved by a textbook alone.

With this edition of *Software Engineering: A Practitioner's Approach,* Bruce Maxim joins me (Roger Pressman) as a coauthor of the book. Bruce brought copious software engineering knowledge to the project and has added new content and insight that will be invaluable to readers of this edition.

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

Preface iv

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
1.3 Summary 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
2.6 Summary 27
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
3.5 Process Assessment and Improvement 37
3.6 Summary 38
PROBLEMS AND POINTS TO PONDER 38
FURTHER READINGS AND INFORMATION SOURCES 39

CHAPTER 4 PROCESS MODELS 40

4.1 Prescriptive Process Models 41
4.1.1 The Waterfall Model 41
4.1.2 Incremental Process Models 43
4.1.3 Evolutionary Process Models 45
4.1.4 Concurrent Models 49
4.1.5 A Final Word on Evolutionary Processes 51
4.2 Specialized Process Models 52
4.2.1 Component-Based Development 53
4.2.2 The Formal Methods Model 53
4.2.3 Aspect-Oriented Software Development 54
4.3 The Unified Process 55
4.3.1 A Brief History 56
4.3.2 Phases of the Unified Process 56
4.4 Personal and Team Process Models 59
4.4.1 Personal Software Process 59
4.4.2 Team Software Process 60
4.5 Process Technology 61
4.6 Product and Process 62
4.7 Summary 64
PROBLEMS AND POINTS TO PONDER 64
FURTHER READINGS AND INFORMATION SOURCES 65

CHAPTER 5 AGILE DEVELOPMENT 66

5.1 What Is Agility? 68
5.2 Agility and the Cost of Change 68
5.3 What Is an Agile Process? 69
5.3.1 Agility Principles 70
5.3.2 The Politics of Agile Development 71
5.4 Extreme Programming 72
5.4.1 The XP Process 72
5.4.2 Industrial XP 75
5.5 Other Agile Process Models 77
5.5.1 Scrum 78
5.5.2 Dynamic Systems Development Method 79
5.5.3 Agile Modeling 80
5.5.4 Agile Unified Process 82
5.6 A Tool Set for the Agile Process 83
5.7 Summary 84
PROBLEMS AND POINTS TO PONDER 85
FURTHER READINGS AND INFORMATION SOURCES 85

CHAPTER 6 HUMAN ASPECTS OF SOFTWARE ENGINEERING 87

6.1 Characteristics of a Software Engineer 88
6.2 The Psychology of Software Engineering 89
6.3 The Software Team 90
6.4 Team Structures 92
6.5 Agile Teams 93
6.5.1 The Generic Agile Team 93
6.5.2 The XP Team 94

6.6 The Impact of Social Media 95
6.7 Software Engineering Using the Cloud 97
6.8 Collaboration Tools 98
6.9 Global Teams 99
6.10 Summary 100
PROBLEMS AND POINTS TO PONDER 101
FURTHER READINGS AND INFORMATION SOURCES 102

PART TWO MODELING 103

CHAPTER 7 PRINCIPLES THAT GUIDE PRACTICE 104

7.1 Software Engineering Knowledge 105
7.2 Core Principles 106
7.2.1 Principles That Guide Process 106
7.2.2 Principles That Guide Practice 107
7.3 Principles That Guide Each Framework Activity 109
7.3.1 Communication Principles 110
7.3.2 Planning Principles 112
7.3.3 Modeling Principles 114
7.3.4 Construction Principles 121
7.3.5 Deployment Principles 125
7.4 Work Practices 126
7.5 Summary 127
PROBLEMS AND POINTS TO PONDER 128
FURTHER READINGS AND INFORMATION SOURCES 129

CHAPTER 8 UNDERSTANDING REQUIREMENTS 131

8.1 Requirements Engineering 132
8.2 Establishing the Groundwork 138
8.2.1 Identifying Stakeholders 139
8.2.2 Recognizing Multiple Viewpoints 139
8.2.3 Working toward Collaboration 140
8.2.4 Asking the First Questions 140
8.2.5 Nonfunctional Requirements 141
8.2.6 Traceability 142
8.3 Eliciting Requirements 142
8.3.1 Collaborative Requirements Gathering 143
8.3.2 Quality Function Deployment 146
8.3.3 Usage Scenarios 146
8.3.4 Elicitation Work Products 147
8.3.5 Agile Requirements Elicitation 148
8.3.6 Service-Oriented Methods 148
8.4 Developing Use Cases 149
8.5 Building the Analysis Model 154
8.5.1 Elements of the Analysis Model 154
8.5.2 Analysis Patterns 157
8.5.3 Agile Requirements Engineering 158
8.5.4 Requirements for Self-Adaptive Systems 158
8.6 Negotiating Requirements 159

8.7 Requirements Monitoring 160
8.8 Validating Requirements 161
8.9 Avoiding Common Mistakes 162
8.10 Summary 162
PROBLEMS AND POINTS TO PONDER 163
FURTHER READINGS AND OTHER INFORMATION SOURCES 164

CHAPTER 9 REQUIREMENTS MODELING: SCENARIO-BASED METHODS 166

9.1 Requirements Analysis 167
9.1.1 Overall Objectives and Philosophy 168
9.1.2 Analysis Rules of Thumb 169
9.1.3 Domain Analysis 170
9.1.4 Requirements Modeling Approaches 171
9.2 Scenario-Based Modeling 173
9.2.1 Creating a Preliminary Use Case 173
9.2.2 Refining a Preliminary Use Case 176
9.2.3 Writing a Formal Use Case 177
9.3 UML Models That Supplement the Use Case 179
9.3.1 Developing an Activity Diagram 180
9.3.2 Swimlane Diagrams 181
9.4 Summary 182
PROBLEMS AND POINTS TO PONDER 182
FURTHER READINGS AND INFORMATION SOURCES 183

CHAPTER 10 REQUIREMENTS MODELING: CLASS-BASED METHODS 184

10.1 Identifying Analysis Classes 185
10.2 Specifying Attributes 188
10.3 Defining Operations 189
10.4 Class-Responsibility-Collaborator Modeling 192
10.5 Associations and Dependencies 198
10.6 Analysis Packages 199
10.7 Summary 200
PROBLEMS AND POINTS TO PONDER 201
FURTHER READINGS AND INFORMATION SOURCES 201

CHAPTER 11 REQUIREMENTS MODELING: BEHAVIOR, PATTERNS, AND WEB/MOBILE APPS 202

11.1 Creating a Behavioral Model 203
11.2 Identifying Events with the Use Case 203
11.3 State Representations 204
11.4 Patterns for Requirements Modeling 207
11.4.1 Discovering Analysis Patterns 208
11.4.2 A Requirements Pattern Example: Actuator-Sensor 209
11.5 Requirements Modeling for Web and Mobile Apps 213
11.5.1 How Much Analysis Is Enough? 214
11.5.2 Requirements Modeling Input 214
11.5.3 Requirements Modeling Output 215
11.5.4 Content Model 216

11.5.5 Interaction Model for Web and Mobile Apps 217
11.5.6 Functional Model 218
11.5.7 Configuration Models for WebApps 219
11.5.8 Navigation Modeling 220
11.6 Summary 221
PROBLEMS AND POINTS TO PONDER 222
FURTHER READINGS AND INFORMATION SOURCES 222

CHAPTER 12 DESIGN CONCEPTS 224

12.1 Design within the Context of Software Engineering 225
12.2 The Design Process 228
12.2.1 Software Quality Guidelines and Attributes 228
12.2.2 The Evolution of Software Design 230
12.3 Design Concepts 231
12.3.1 Abstraction 232
12.3.2 Architecture 232
12.3.3 Patterns 233
12.3.4 Separation of Concerns 234
12.3.5 Modularity 234
12.3.6 Information Hiding 235
12.3.7 Functional Independence 236
12.3.8 Refinement 237
12.3.9 Aspects 237
12.3.10 Refactoring 238
12.3.11 Object-Oriented Design Concepts 238
12.3.12 Design Classes 239
12.3.13 Dependency Inversion 241
12.3.14 Design for Test 242
12.4 The Design Model 243
12.4.1 Data Design Elements 244
12.4.2 Architectural Design Elements 244
12.4.3 Interface Design Elements 245
12.4.4 Component-Level Design Elements 247
12.4.5 Deployment-Level Design Elements 248
12.5 Summary 249
PROBLEMS AND POINTS TO PONDER 250
FURTHER READINGS AND INFORMATION SOURCES 251

CHAPTER 13 ARCHITECTURAL DESIGN 252

13.1 Software Architecture 253
13.1.1 What Is Architecture? 253
13.1.2 Why Is Architecture Important? 254
13.1.3 Architectural Descriptions 255
13.1.4 Architectural Decisions 256
13.2 Architectural Genres 257
13.3 Architectural Styles 258
13.3.1 A Brief Taxonomy of Architectural Styles 258
13.3.2 Architectural Patterns 263
13.3.3 Organization and Refinement 263
13.4 Architectural Considerations 264

13.5 Architectural Decisions 266
13.6 Architectural Design 267
13.6.1 Representing the System in Context 267
13.6.2 Defining Archetypes 269
13.6.3 Refining the Architecture into Components 270
13.6.4 Describing Instantiations of the System 272
13.6.5 Architectural Design for Web Apps 273
13.6.6 Architectural Design for Mobile Apps 274
13.7 Assessing Alternative Architectural Designs 274
13.7.1 Architectural Description Languages 276
13.7.2 Architectural Reviews 277
13.8 Lessons Learned 278
13.9 Pattern-based Architecture Review 278
13.10 Architecture Conformance Checking 279
13.11 Agility and Architecture 280
13.12 Summary 282
PROBLEMS AND POINTS TO PONDER 282
FURTHER READINGS AND INFORMATION SOURCES 283

CHAPTER 14 COMPONENT-LEVEL DESIGN 285

14.1 What Is a Component? 286
14.1.1 An Object-Oriented View 286
14.1.2 The Traditional View 288
14.1.3 A Process-Related View 291
14.2 Designing Class-Based Components 291
14.2.1 Basic Design Principles 292
14.2.2 Component-Level Design Guidelines 295
14.2.3 Cohesion 296
14.2.4 Coupling 298
14.3 Conducting Component-Level Design 299
14.4 Component-Level Design for WebApps 305
14.4.1 Content Design at the Component Level 306
14.4.2 Functional Design at the Component Level 306
14.5 Component-Level Design for Mobile Apps 306
14.6 Designing Traditional Components 307
14.7 Component-Based Development 308
14.7.1 Domain Engineering 308
14.7.2 Component Qualification, Adaptation, and Composition 309
14.7.3 Architectural Mismatch 311
14.7.4 Analysis and Design for Reuse 312
14.7.5 Classifying and Retrieving Components 312
14.8 Summary 313
PROBLEMS AND POINTS TO PONDER 315
FURTHER READINGS AND INFORMATION SOURCES 315

CHAPTER 15 USER INTERFACE DESIGN 317

15.1 The Golden Rules 318
15.1.1 Place the User in Control 318
15.1.2 Reduce the User's Memory Load 319
15.1.3 Make the Interface Consistent 321

15.2 User Interface Analysis and Design 322
15.2.1 Interface Analysis and Design Models 322
15.2.2 The Process 323
15.3 Interface Analysis 325
15.3.1 User Analysis 325
15.3.2 Task Analysis and Modeling 326
15.3.3 Analysis of Display Content 331
15.3.4 Analysis of the Work Environment 331
15.4 Interface Design Steps 332
15.4.1 Applying Interface Design Steps 332
15.4.2 User Interface Design Patterns 334
15.4.3 Design Issues 335
15.5 WebApp and Mobile Interface Design 337
15.5.1 Interface Design Principles and Guidelines 337
15.5.2 Interface Design Workflow for Web and Mobile Apps 341
15.6 Design Evaluation 342
15.7 Summary 344
PROBLEMS AND POINTS TO PONDER 345
FURTHER READINGS AND INFORMATION SOURCES 346

CHAPTER 16 PATTERN-BASED DESIGN 347

16.1 Design Patterns 348
16.1.1 Kinds of Patterns 349
16.1.2 Frameworks 351
16.1.3 Describing a Pattern 352
16.1.4 Pattern Languages and Repositories 353
16.2 Pattern-Based Software Design 354
16.2.1 Pattern-Based Design in Context 354
16.2.2 Thinking in Patterns 354
16.2.3 Design Tasks 356
16.2.4 Building a Pattern-Organizing Table 358
16.2.5 Common Design Mistakes 359
16.3 Architectural Patterns 359
16.4 Component-Level Design Patterns 360
16.5 User Interface Design Patterns 362
16.6 WebApp Design Patterns 364
16.6.1 Design Focus 365
16.6.2 Design Granularity 365
16.7 Patterns for Mobile Apps 366
16.8 Summary 367
PROBLEMS AND POINTS TO PONDER 368
FURTHER READINGS AND INFORMATION SOURCES 369

CHAPTER 17 WEBAPP DESIGN 371

17.1 WebApp Design Quality 372
17.2 Design Goals 374
17.3 A Design Pyramid for WebApps 375
17.4 WebApp Interface Design 376

17.5 Aesthetic Design 377
17.5.1 Layout Issues 378
17.5.2 Graphic Design Issues 378
17.6 Content Design 379
17.6.1 Content Objects 379
17.6.2 Content Design Issues 380
17.7 Architecture Design 381
17.7.1 Content Architecture 381
17.7.2 WebApp Architecture 384
17.8 Navigation Design 385
17.8.1 Navigation Semantics 385
17.8.2 Navigation Syntax 387
17.9 Component-Level Design 387
17.10 Summary 388
PROBLEMS AND POINTS TO PONDER 389
FURTHER READINGS AND INFORMATION SOURCES 389

CHAPTER 18 MOBILEAPP DESIGN 391

18.1 The Challenges 392
18.1.1 Development Considerations 392
18.1.2 Technical Considerations 393
18.2 Developing MobileApps 395
18.2.1 MobileApp Quality 397
18.2.2 User Interface Design 398
18.2.3 Context-Aware Apps 399
18.2.4 Lessons Learned 400
18.3 MobileApp Design—Best Practices 401
18.4 Mobility Environments 403
18.5 The Cloud 405
18.6 The Applicability of Conventional Software Engineering 407
18.7 Summary 408
PROBLEMS AND POINTS TO PONDER 409
FURTHER READINGS AND INFORMATION SOURCES 409

PART THREE QUALITY MANAGEMENT 411

CHAPTER 19 QUALITY CONCEPTS 412

19.1 What Is Quality? 413
19.2 Software Quality 414
19.2.1 Garvin's Quality Dimensions 415
19.2.2 McCall's Quality Factors 416
19.2.3 ISO 9126 Quality Factors 418
19.2.4 Targeted Quality Factors 418
19.2.5 The Transition to a Quantitative View 420
19.3 The Software Quality Dilemma 420
19.3.1 "Good Enough" Software 421
19.3.2 The Cost of Quality 422
19.3.3 Risks 424
19.3.4 Negligence and Liability 425