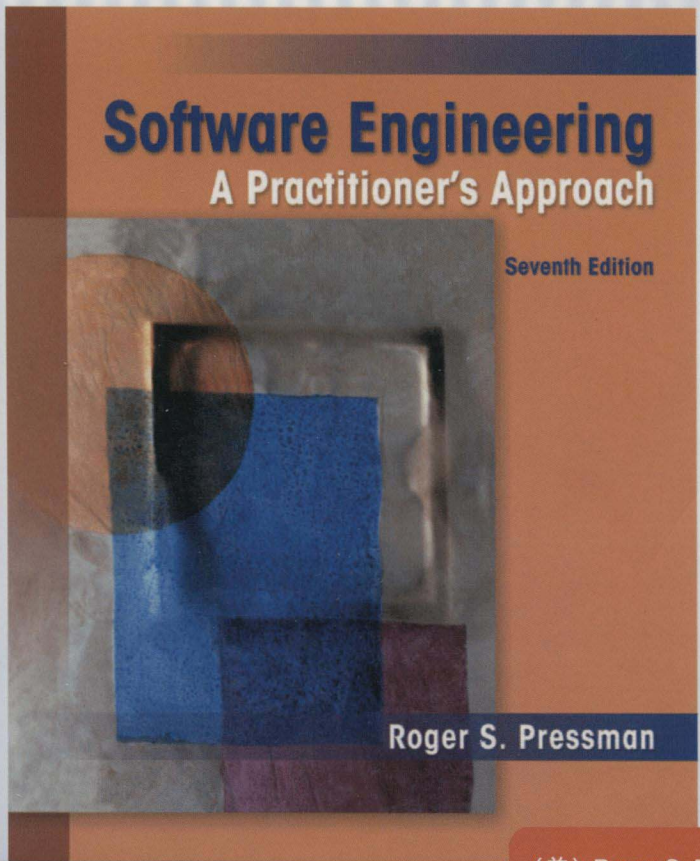
HZ BOOKS 华章教育
Mc Graw Hill Education
经 典 原 版 书 库

软件工程

实践者的研究方法

（英文版 · 第7版）



（美）Roger S. Pressman 著

机械工业出版社
China Machine Press

软件工程 实践者的研究方法 （英文版 · 第7版）

Software Engineering A Practitioner's Approach (Seventh Edition)

Roger Pressman编写的这部翔实而全面的软件工程指南，广泛适合软件工程专业的学生及投身软件工程实践或需要参与这种实践的软件开发人员和管理人员。

——《IEEE Software》

这是一本经典的现代教材，叙述清晰而又具有权威性。本书包含大量插图、例子、习题和参考资料……如果读者心存疑问："软件工程是什么？它现在在哪里？"那么最好阅读这本书。

——《ACM Computing Reviews》

作为一名软件工程实践者，我发现此书是无价的。对于我做过的所有项目，本书都有重大的参考价值。

——摘自Amazon.com的评论

本书自1982年发行第1版以来，一直受到软件工程界的高度重视，成为高等院校计算机相关专业软件工程课程的重要教学参考书。近30年来，它的各个后继版本一直都是软件专业人士熟悉的读物，在国际软件工程界享有无可质疑的权威地位。它在全面而系统地介绍软件工程的有关概念、原则、方法和工具方面获得了广大读者的好评。

本书在给出传统的、对学科发展具有深刻影响的方法时，又适当地介绍了当前正在发展的、具有生命力的新技术。本书第7版在结构和内容上均有调整、更新和充实，论述了很多人们称之为"21世纪工程学科"的重要主题。第7版更加突出软件过程，强调普遍使用的软件工程方法。

对第7版的内容做了如下划分，这样更便于课堂教学及自学使用：

第一部分 软件过程，介绍了说明性模型和敏捷过程模型。

第二部分 建模，介绍了现代分析与设计方法，新的重点放在基于UML的建模方面。

第三部分 质量管理，是第7版中新增加的内容，描述软件测试、质量保证、形式化验证技术和变更管理的各个方面。

第四部分 软件项目管理，介绍与计划、管理和控制软件项目有关的主题。

第五部分 软件工程高级课题，用专门的章节讲述软件过程改进及将来的软件工程趋势。

作者简介

Roger S. Pressman 是软件过程改善和软件工程技术方面的国际知名的权威人士。30多年来，他作为软件工程师、管理人员、教授、作者及咨询顾问始终工作在软件工程领域。Pressman博士著有6部著作，并撰写了很多技术文章，是多种行业期刊的固定撰稿人，曾任多种行业杂志的编委，多年来一直担任引《IEEE Software》杂志的Manager专栏的编辑。Pressman博士是知名的演讲者，曾在许多行业会议上演讲，他还是美国计算机协会(ACM)、美国电气与电子工程师协会(IEEE)等组织的成员。

Mc Graw Hill Education
http://www.mheducation.com

上架指导：计算机/软件工程
ISBN 978-7-111-31871-2
9 787111 318712

定价：75.00元

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

华章网站 http://www.hzbook.com
网上购书：www.china-pub.com

封面设计 · 杨宇梅

HZ BOOKS
华章教育

Mc Graw Hill

软件工程 实践者的研究方法

Software Engineering A Practitioner's Approach (Seventh Edition)

(美) Roger S. Pressman 著

英文版
第7版

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7版）

Software Engineering

A Practitioner's Approach

(Seventh Edition)

（美）Roger S. Pressman 著

机械工业出版社
China Machine Press

本书版权登记号：图字：01-2009-4562

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7版）/（美）普雷斯曼（Pressman, R. S.）著. —北京：机械工业出版社，2010.9

（经典原版书库）

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

ISBN 978-7-111-31871-2

Ⅰ. 软…　Ⅱ. 普…　Ⅲ. 软件工程-英文　Ⅳ. TP311.5

中国版本图书馆CIP数据核字（2010）第177825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：李俊竹

北京京师印务有限公司印刷

2010年10月第1版第1次印刷

150mm×214mm · 28.875印张

标准书号：ISBN 978-7-111-31871-2

定价：75.00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

华章教育
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 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.

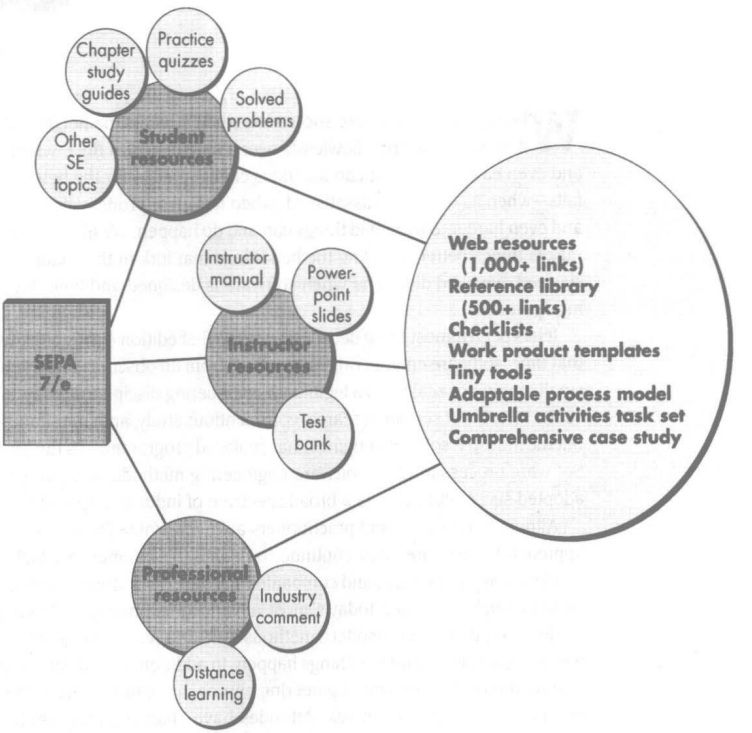
Although managers and practitioners alike recognize the need for a more disciplined approach to software, they continue to debate the manner in which discipline is to be applied. Many individuals and companies still develop software haphazardly, even as they build systems to service today's most advanced 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 seventh edition of *Software Engineering: A Practitioner's Approach* is intended to serve as a guide to a maturing engineering discipline. Like the six editions that preceded it, the seventh edition 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 seventh 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, a revised and updated "support system," illustrated in the figure, provides 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 Seventh Edition. The 32 chapters of the seventh edition have been reorganized into five parts. This organization, which differs considerably from the sixth edition, has been done to better compartmentalize topics and assist instructors who may not have the time to complete the entire book in one term.

Support System for SEPA, 7/e



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 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. Two new appendices provide brief tutorials on UML and object-oriented thinking for those who may be unfamiliar with these important topics.

The five-part organization of the seventh 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 seventh edition in this way, I have attempted to provide an instructor with a number of teaching options. In every case, the content of the seventh edition is complemented by the following elements of the *SEPA, 7/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 1000 categorized *Web References* allow a student to explore software engineering in greater detail and a *Reference Library* with links to over 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 seventh edition. These include a complete online *Instructor's Guide* (also downloadable) and supplementary teaching materials including a complete set of over 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 (CASE) tools, a comprehensive collection of Web-based resources, and an "adaptable process model" that provides a detailed task breakdown of the software engineering process.

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

Acknowledgments. My work on the seven editions of *Software Engineering: A Practitioner's Approach* has been the longest continuing technical project of my life. Even when the writing stops, information extracted from the technical literature continues to be assimilated and organized, and criticism and suggestions from readers worldwide is evaluated and cataloged. For this reason, my thanks to the many authors of books, papers, and articles (in both hardcopy and electronic media) who have provided me with additional insight, ideas, and commentary over nearly 30 years.

Special thanks go to Tim Lethbridge of the University of Ottawa, who assisted me 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. Special thanks also go to Bruce Maxim of the University of Michigan–Dearborn, who assisted me in developing much of the pedagogical website content that accompanies this book. Finally, I wish to thank the reviewers of the seventh edition: Their in-depth comments and thoughtful criticism have been invaluable.

Osman Balci,
Virginia Tech University
Max Fomitchev,
Penn State University
Jerry (Zeyu) Gao,
San Jose State University
Guillermo Garcia,
Universidad Alfonso X Madrid
Pablo Gervas,
Universidad Complutense de Madrid
SK Jain,
National Institute of Technology Hamirpur
Saeed Monemi,
Cal Poly Pomona
Ahmed Salem,
California State University
Vasudeva Varma,
IIIT Hyderabad

The content of the seventh edition of *Software Engineering: A Practitioner's Approach* has been shaped by industry professionals, university professors, and students who have used earlier editions of the book and have taken the time to communicate their suggestions, criticisms, and ideas. My thanks to each of you. In addition, my personal thanks go to our many industry clients worldwide, who certainly have taught me as much or more than I could ever teach them.

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. And finally, to 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

About the Author

Roger S. Pressman is an internationally recognized authority in software process improvement and software engineering technologies. For almost four decades, he has worked as a software engineer, a manager, a professor, an author, and a consultant, focusing on software engineering issues.

As an industry practitioner and manager, Dr. Pressman worked on the development of CAD/CAM systems for advanced engineering and manufacturing applications. He has also held positions with responsibility for scientific and systems programming.

After receiving a Ph.D. in engineering from the University of Connecticut, Dr. Pressman moved to academia where he became Bullard Associate Professor of Computer Engineering at the University of Bridgeport and director of the university's Computer-Aided Design and Manufacturing Center.

Dr. Pressman is currently president of R.S. Pressman & Associates, Inc., a consulting firm specializing in software engineering methods and training. He serves as principal consultant and has designed and developed *Essential Software Engineering,* a complete video curriculum in software engineering, and *Process Advisor,* a self-directed system for software process improvement. Both products are used by thousands of companies worldwide. More recently, he has worked in collaboration with *EdistaLearning* in India to develop comprehensive Internet-based training in software engineering.

Dr. Pressman has written many technical papers, is a regular contributor to industry periodicals, and is author of seven technical books. In addition to *Software Engineering: A Practitioner's Approach,* he has co-authored *Web Engineering* (McGraw-Hill), one of the first books to apply a tailored set of software engineering principles and practices to the development of Web-based systems and applications. He has also written the award-winning *A Manager's Guide to Software Engineering* (McGraw-Hill); *Making Software Engineering Happen* (Prentice Hall), the first book to address the critical management problems associated with software process improvement; and *Software Shock* (Dorset House), a treatment that focuses on software and its impact on business and society. Dr. Pressman has been on the editorial boards of a number of industry journals, and for many years, was editor of the "Manager" column in *IEEE Software.*

Dr. Pressman is a well-known speaker, keynoting a number of major industry conferences. He is a member of the IEEE, and Tau Beta Pi, Phi Kappa Phi, Eta Kappa Nu, and Pi Tau Sigma.

On the personal side, Dr. Pressman lives in South Florida with his wife, Barbara. An athlete for most of his life, he remains a serious tennis player (NTRP 4.5) and a single-digit handicap golfer. In his spare time, he has written two novels, The *Aymara Bridge* and *The Puppeteer,* and plans to begin work on another.

TABLE OF CONTENTS

Preface v
About the Author ix

CHAPTER 1 SOFTWARE AND SOFTWARE ENGINEERING 1

1.1 The Nature of Software 3
1.1.1 Defining Software 4
1.1.2 Software Application Domains 7
1.1.3 Legacy Software 9
1.2 The Unique Nature of WebApps 10
1.3 Software Engineering 12
1.4 The Software Process 14
1.5 Software Engineering Practice 17
1.5.1 The Essence of Practice 17
1.5.2 General Principles 19
1.6 Software Myths 21
1.7 How It All Starts 24
1.8 Summary 25
PROBLEMS AND POINTS TO PONDER 25
FURTHER READINGS AND INFORMATION SOURCES 26

PART ONE THE SOFTWARE PROCESS 29

CHAPTER 2 PROCESS MODELS 30

2.1 A Generic Process Model 31
2.1.1 Defining a Framework Activity 32
2.1.2 Identifying a Task Set 34
2.1.3 Process Patterns 35
2.2 Process Assessment and Improvement 37
2.3 Prescriptive Process Models 38
2.3.1 The Waterfall Model 39
2.3.2 Incremental Process Models 41
2.3.3 Evolutionary Process Models 42
2.3.4 Concurrent Models 48
2.3.5 A Final Word on Evolutionary Processes 49
2.4 Specialized Process Models 50
2.4.1 Component-Based Development 50
2.4.2 The Formal Methods Model 51
2.4.3 Aspect-Oriented Software Development 52
2.5 The Unified Process 53
2.5.1 A Brief History 54
2.5.2 Phases of the Unified Process 54
2.6 Personal and Team Process Models 56
2.6.1 Personal Software Process (PSP) 57
2.6.2 Team Software Process (TSP) 58
2.7 Process Technology 59
2.8 Product and Process 60

2.9 Summary 61
PROBLEMS AND POINTS TO PONDER 62
FURTHER READINGS AND INFORMATION SOURCES 63

CHAPTER 3 AGILE DEVELOPMENT 65

3.1 What Is Agility? 67
3.2 Agility and the Cost of Change 67
3.3 What Is an Agile Process? 68
3.3.1 Agility Principles 69
3.3.2 The Politics of Agile Development 70
3.3.3 Human Factors 71
3.4 Extreme Programming (XP) 72
3.4.1 XP Values 72
3.4.2 The XP Process 73
3.4.3 Industrial XP 77
3.4.4 The XP Debate 78
3.5 Other Agile Process Models 80
3.5.1 Adaptive Software Development (ASD) 81
3.5.2 Scrum 82
3.5.3 Dynamic Systems Development Method (DSDM) 84
3.5.4 Crystal 85
3.5.5 Feature Driven Development (FDD) 86
3.5.6 Lean Software Development (LSD) 87
3.5.7 Agile Modeling (AM) 88
3.5.8 Agile Unified Process (AUP) 89
3.6 A Tool Set for the Agile Process 91
3.7 Summary 91
PROBLEMS AND POINTS TO PONDER 92
FURTHER READINGS AND INFORMATION SOURCES 93

PART TWO MODELING 95

CHAPTER 4 PRINCIPLES THAT GUIDE PRACTICE 96

4.1 Software Engineering Knowledge 97
4.2 Core Principles 98
4.2.1 Principles That Guide Process 98
4.2.2 Principles That Guide Practice 99
4.3 Principles That Guide Each Framework Activity 101
4.3.1 Communication Principles 101
4.3.2 Planning Principles 103
4.3.3 Modeling Principles 105
4.3.4 Construction Principles 111
4.3.5 Deployment Principles 113
4.4 Summary 115
PROBLEMS AND POINTS TO PONDER 116
FURTHER READINGS AND INFORMATION SOURCES 116

CHAPTER 5 UNDERSTANDING REQUIREMENTS 119

5.1 Requirements Engineering 120
5.2 Establishing the Groundwork 125
5.2.1 Identifying Stakeholders 125

5.2.2 Recognizing Multiple Viewpoints 126
5.2.3 Working toward Collaboration 126
5.2.4 Asking the First Questions 127
5.3 Eliciting Requirements 128
5.3.1 Collaborative Requirements Gathering 128
5.3.2 Quality Function Deployment 131
5.3.3 Usage Scenarios 132
5.3.4 Elicitation Work Products 133
5.4 Developing Use Cases 133
5.5 Building the Requirements Model 138
5.5.1 Elements of the Requirements Model 139
5.5.2 Analysis Patterns 142
5.6 Negotiating Requirements 142
5.7 Validating Requirements 144
5.8 Summary 145
PROBLEMS AND POINTS TO PONDER 145
FURTHER READINGS AND INFORMATION SOURCES 146

CHAPTER 6 REQUIREMENTS MODELING: SCENARIOS, INFORMATION, AND ANALYSIS CLASSES 148

6.1 Requirements Analysis 149
6.1.1 Overall Objectives and Philosophy 150
6.1.2 Analysis Rules of Thumb 151
6.1.3 Domain Analysis 151
6.1.4 Requirements Modeling Approaches 153
6.2 Scenario-Based Modeling 154
6.2.1 Creating a Preliminary Use Case 155
6.2.2 Refining a Preliminary Use Case 158
6.2.3 Writing a Formal Use Case 159
6.3 UML Models That Supplement the Use Case 161
6.3.1 Developing an Activity Diagram 161
6.3.2 Swimlane Diagrams 162
6.4 Data Modeling Concepts 164
6.4.1 Data Objects 164
6.4.2 Data Attributes 164
6.4.3 Relationships 165
6.5 Class-Based Modeling 167
6.5.1 Identifying Analysis Classes 167
6.5.2 Specifying Attributes 171
6.5.3 Defining Operations 171
6.5.4 Class-Responsibility-Collaborator (CRC) Modeling 173
6.5.5 Associations and Dependencies 180
6.5.6 Analysis Packages 182
6.6 Summary 183
PROBLEMS AND POINTS TO PONDER 183
FURTHER READINGS AND INFORMATION SOURCES 184

CHAPTER 7 REQUIREMENTS MODELING: FLOW, BEHAVIOR, PATTERNS, AND WEBAPPS 186

7.1 Requirements Modeling Strategies 186
7.2 Flow-Oriented Modeling 187

7.2.1 Creating a Data Flow Model 188
7.2.2 Creating a Control Flow Model 191
7.2.3 The Control Specification 191
7.2.4 The Process Specification 192
7.3 Creating a Behavioral Model 195
7.3.1 Identifying Events with the Use Case 195
7.3.2 State Representations 196
7.4 Patterns for Requirements Modeling 199
7.4.1 Discovering Analysis Patterns 200
7.4.2 A Requirements Pattern Example: Actuator-Sensor 200
7.5 Requirements Modeling for WebApps 205
7.5.1 How Much Analysis Is Enough? 205
7.5.2 Requirements Modeling Input 206
7.5.3 Requirements Modeling Output 207
7.5.4 Content Model for WebApps 207
7.5.5 Interaction Model for WebApps 209
7.5.6 Functional Model for WebApps 210
7.5.7 Configuration Models for WebApps 211
7.5.8 Navigation Modeling 212
7.6 Summary 213
PROBLEMS AND POINTS TO PONDER 213
FURTHER READINGS AND INFORMATION SOURCES 214

CHAPTER 8 DESIGN CONCEPTS 215

8.1 Design within the Context of Software Engineering 216
8.2 The Design Process 219
8.2.1 Software Quality Guidelines and Attributes 219
8.2.2 The Evolution of Software Design 221
8.3 Design Concepts 222
8.3.1 Abstraction 223
8.3.2 Architecture 223
8.3.3 Patterns 224
8.3.4 Separation of Concerns 225
8.3.5 Modularity 225
8.3.6 Information Hiding 226
8.3.7 Functional Independence 227
8.3.8 Refinement 228
8.3.9 Aspects 228
8.3.10 Refactoring 229
8.3.11 Object-Oriented Design Concepts 230
8.3.12 Design Classes 230
8.4 The Design Model 233
8.4.1 Data Design Elements 234
8.4.2 Architectural Design Elements 234
8.4.3 Interface Design Elements 235
8.4.4 Component-Level Design Elements 237
8.4.5 Deployment-Level Design Elements 237
8.5 Summary 239
PROBLEMS AND POINTS TO PONDER 240
FURTHER READINGS AND INFORMATION SOURCES 240

CHAPTER 9 ARCHITECTURAL DESIGN 242

9.1 Software Architecture 243
- 9.1.1 What Is Architecture? 243
- 9.1.2 Why Is Architecture Important? 245
- 9.1.3 Architectural Descriptions 245
- 9.1.4 Architectural Decisions 246

9.2 Architectural Genres 246
9.3 Architectural Styles 249
- 9.3.1 A Brief Taxonomy of Architectural Styles 250
- 9.3.2 Architectural Patterns 253
- 9.3.3 Organization and Refinement 255

9.4 Architectural Design 255
- 9.4.1 Representing the System in Context 256
- 9.4.2 Defining Archetypes 257
- 9.4.3 Refining the Architecture into Components 258
- 9.4.4 Describing Instantiations of the System 260

9.5 Assessing Alternative Architectural Designs 261
- 9.5.1 An Architecture Trade-Off Analysis Method 262
- 9.5.2 Architectural Complexity 263
- 9.5.3 Architectural Description Languages 264

9.6 Architectural Mapping Using Data Flow 265
- 9.6.1 Transform Mapping 265
- 9.6.2 Refining the Architectural Design 272

9.7 Summary 273

PROBLEMS AND POINTS TO PONDER 274

FURTHER READINGS AND INFORMATION SOURCES 274

CHAPTER 10 COMPONENT-LEVEL DESIGN 276

10.1 What Is a Component? 277
- 10.1.1 An Object-Oriented View 277
- 10.1.2 The Traditional View 279
- 10.1.3 A Process-Related View 281

10.2 Designing Class-Based Components 282
- 10.2.1 Basic Design Principles 282
- 10.2.2 Component-Level Design Guidelines 285
- 10.2.3 Cohesion 286
- 10.2.4 Coupling 288

10.3 Conducting Component-Level Design 290
10.4 Component-Level Design for WebApps 296
- 10.4.1 Content Design at the Component Level 297
- 10.4.2 Functional Design at the Component Level 297

10.5 Designing Traditional Components 298
- 10.5.1 Graphical Design Notation 299
- 10.5.2 Tabular Design Notation 300
- 10.5.3 Program Design Language 301

10.6 Component-Based Development 303
- 10.6.1 Domain Engineering 303
- 10.6.2 Component Qualification, Adaptation, and Composition 304
- 10.6.3 Analysis and Design for Reuse 306
- 10.6.4 Classifying and Retrieving Components 307