

Software Engineering

A PRACTITIONER'S APPROACH

EIGHTH EDITION

Roger S. Pressman, Ph.D.
Bruce R. Maxim, Ph.D.

**Mc
Graw
Hill**
Education

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.

CONTENTS AT A GLANCE

CHAPTER 1	The Nature of Software	1
CHAPTER 2	Software Engineering	14

PART ONE **THE SOFTWARE PROCESS** 29

CHAPTER 3	Software Process Structure	30
CHAPTER 4	Process Models	40
CHAPTER 5	Agile Development	66
CHAPTER 6	Human Aspects of Software Engineering	87

PART TWO **MODELING** 103

CHAPTER 7	Principles That Guide Practice	104
CHAPTER 8	Understanding Requirements	131
CHAPTER 9	Requirements Modeling: Scenario-Based Methods	166
CHAPTER 10	Requirements Modeling: Class-Based Methods	184
CHAPTER 11	Requirements Modeling: Behavior, Patterns, and Web/Mobile Apps	202
CHAPTER 12	Design Concepts	224
CHAPTER 13	Architectural Design	252
CHAPTER 14	Component-Level Design	285
CHAPTER 15	User Interface Design	317
CHAPTER 16	Pattern-Based Design	347
CHAPTER 17	WebApp Design	371
CHAPTER 18	MobileApp Design	391

PART THREE **QUALITY MANAGEMENT** 411

CHAPTER 19	Quality Concepts	412
CHAPTER 20	Review Techniques	431
CHAPTER 21	Software Quality Assurance	448
CHAPTER 22	Software Testing Strategies	466
CHAPTER 23	Testing Conventional Applications	496
CHAPTER 24	Testing Object-Oriented Applications	523
CHAPTER 25	Testing Web Applications	540
CHAPTER 26	Testing MobileApps	567

CHAPTER 27	Security Engineering	584
CHAPTER 28	Formal Modeling and Verification	601
CHAPTER 29	Software Configuration Management	623
CHAPTER 30	Product Metrics	653

PART FOUR MANAGING SOFTWARE PROJECTS 683

CHAPTER 31	Project Management Concepts	684
CHAPTER 32	Process and Project Metrics	703
CHAPTER 33	Estimation for Software Projects	727
CHAPTER 34	Project Scheduling	754
CHAPTER 35	Risk Management	777
CHAPTER 36	Maintenance and Reengineering	795

PART FIVE ADVANCED TOPICS 817

CHAPTER 37	Software Process Improvement	818
CHAPTER 38	Emerging Trends in Software Engineering	839
CHAPTER 39	Concluding Comments	860
APPENDIX 1	An Introduction to UML	869
APPENDIX 2	Object-Oriented Concepts	891
APPENDIX 3	Formal Methods	899
REFERENCES		909
INDEX		933

TABLE OF CONTENTS

Preface xxvii

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	ComponentLevel 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

19.3.5	Quality and Security	425
19.3.6	The Impact of Management Actions	426
19.4	Achieving Software Quality	427
19.4.1	Software Engineering Methods	427
19.4.2	Project Management Techniques	427
19.4.3	Quality Control	427
19.4.4	Quality Assurance	428
19.5	Summary	428
	PROBLEMS AND POINTS TO PONDER	429
	FURTHER READINGS AND INFORMATION SOURCES	429

CHAPTER 20 REVIEW TECHNIQUES 431

20.1	Cost Impact of Software Defects	432
20.2	Defect Amplification and Removal	433
20.3	Review Metrics and Their Use	435
20.3.1	Analyzing Metrics	435
20.3.2	Cost-Effectiveness of Reviews	436
20.4	Reviews: A Formality Spectrum	438
20.5	Informal Reviews	439
20.6	Formal Technical Reviews	441
20.6.1	The Review Meeting	441
20.6.2	Review Reporting and Record Keeping	442
20.6.3	Review Guidelines	442
20.6.4	Sample-Driven Reviews	444
20.7	Post-Mortem Evaluations	445
20.8	Summary	446
	PROBLEMS AND POINTS TO PONDER	446
	FURTHER READINGS AND INFORMATION SOURCES	447

CHAPTER 21 SOFTWARE QUALITY ASSURANCE 448

21.1	Background Issues	449
21.2	Elements of Software Quality Assurance	450
21.3	SQA Processes and Product Characteristics	452
21.4	SQA Tasks, Goals, and Metrics	452
21.4.1	SQA Tasks	453
21.4.2	Goals, Attributes, and Metrics	454
21.5	Formal Approaches to SQA	456
21.6	Statistical Software Quality Assurance	456
21.6.1	A Generic Example	457
21.6.2	Six Sigma for Software Engineering	458
21.7	Software Reliability	459
21.7.1	Measures of Reliability and Availability	459
21.7.2	Software Safety	460
21.8	The ISO 9000 Quality Standards	461
21.9	The SQA Plan	463
21.10	Summary	463
	PROBLEMS AND POINTS TO PONDER	464
	FURTHER READINGS AND INFORMATION SOURCES	464

CHAPTER 22 SOFTWARE TESTING STRATEGIES 466

22.1	A Strategic Approach to Software Testing	466
22.1.1	Verification and Validation	468
22.1.2	Organizing for Software Testing	468
22.1.3	Software Testing Strategy—The Big Picture	469
22.1.4	Criteria for Completion of Testing	472
22.2	Strategic Issues	472
22.3	Test Strategies for Conventional Software	473
22.3.1	Unit Testing	473
22.3.2	Integration Testing	475
22.4	Test Strategies for Object-Oriented Software	481
22.4.1	Unit Testing in the OO Context	481
22.4.2	Integration Testing in the OO Context	481
22.5	Test Strategies for WebApps	482
22.6	Test Strategies for MobileApps	483
22.7	Validation Testing	483
22.7.1	Validation-Test Criteria	484
22.7.2	Configuration Review	484
22.7.3	Alpha and Beta Testing	484
22.8	System Testing	486
22.8.1	Recovery Testing	486
22.8.2	Security Testing	486
22.8.3	Stress Testing	487
22.8.4	Performance Testing	487
22.8.5	Deployment Testing	487
22.9	The Art of Debugging	488
22.9.1	The Debugging Process	488
22.9.2	Psychological Considerations	490
22.9.3	Debugging Strategies	491
22.9.4	Correcting the Error	492
22.10	Summary	493
	PROBLEMS AND POINTS TO PONDER	493
	FURTHER READINGS AND INFORMATION SOURCES	494

CHAPTER 23 TESTING CONVENTIONAL APPLICATIONS 496

23.1	Software Testing Fundamentals	497
23.2	Internal and External Views of Testing	499
23.3	White-Box Testing	500
23.4	Basis Path Testing	500
23.4.1	Flow Graph Notation	500
23.4.2	Independent Program Paths	502
23.4.3	Deriving Test Cases	504
23.4.4	Graph Matrices	506
23.5	Control Structure Testing	507
23.6	Black-Box Testing	509
23.6.1	Graph-Based Testing Methods	509
23.6.2	Equivalence Partitioning	511
23.6.3	Boundary Value Analysis	512
23.6.4	Orthogonal Array Testing	513

23.7	Model-Based Testing	516
23.8	Testing Documentation and Help Facilities	516
23.9	Testing for Real-Time Systems	517
23.10	Patterns for Software Testing	519
23.11	Summary	520
	PROBLEMS AND POINTS TO PONDER	521
	FURTHER READINGS AND INFORMATION SOURCES	521

CHAPTER 24 TESTING OBJECT-ORIENTED APPLICATIONS 523

24.1	Broadening the View of Testing	524
24.2	Testing OOA and OOD Models	525
24.2.1	Correctness of OOA and OOD Models	525
24.2.2	Consistency of Object-Oriented Models	526
24.3	Object-Oriented Testing Strategies	528
24.3.1	Unit Testing in the OO Context	528
24.3.2	Integration Testing in the OO Context	529
24.3.3	Validation Testing in an OO Context	529
24.4	Object-Oriented Testing Methods	529
24.4.1	The Test-Case Design Implications of OO Concepts	530
24.4.2	Applicability of Conventional Test-Case Design Methods	531
24.4.3	Fault-Based Testing	531
24.4.4	Scenario-Based Test Design	532
24.5	Testing Methods Applicable at the Class Level	532
24.5.1	Random Testing for OO Classes	532
24.5.2	Partition Testing at the Class Level	533
24.6	Interclass Test-Case Design	534
24.6.1	Multiple Class Testing	534
24.6.2	Tests Derived from Behavior Models	536
24.7	Summary	537
	PROBLEMS AND POINTS TO PONDER	538
	FURTHER READINGS AND INFORMATION SOURCES	538

CHAPTER 25 TESTING WEB APPLICATIONS 540

25.1	Testing Concepts for WebApps	541
25.1.1	Dimensions of Quality	541
25.1.2	Errors within a WebApp Environment	542
25.1.3	Testing Strategy	543
25.1.4	Test Planning	543
25.2	The Testing Process—An Overview	544
25.3	Content Testing	545
25.3.1	Content Testing Objectives	545
25.3.2	Database Testing	547
25.4	User Interface Testing	549
25.4.1	Interface Testing Strategy	549
25.4.2	Testing Interface Mechanisms	550
25.4.3	Testing Interface Semantics	552
25.4.4	Usability Tests	552
25.4.5	Compatibility Tests	554
25.5	Component-Level Testing	555

25.6	Navigation Testing	556
25.6.1	Testing Navigation Syntax	556
25.6.2	Testing Navigation Semantics	556
25.7	Configuration Testing	558
25.7.1	Server-Side Issues	558
25.7.2	Client-Side Issues	559
25.8	Security Testing	559
25.9	Performance Testing	560
25.9.1	Performance Testing Objectives	561
25.9.2	Load Testing	562
25.9.3	Stress Testing	562
25.10	Summary	563
	PROBLEMS AND POINTS TO PONDER	564
	FURTHER READINGS AND INFORMATION SOURCES	565

CHAPTER 26 TESTING MOBILEAPPS 567

26.1	Testing Guidelines	568
26.2	The Testing Strategies	569
26.2.1	Are Conventional Approaches Applicable?	570
26.2.2	The Need for Automation	571
26.2.3	Building a Test Matrix	572
26.2.4	Stress Testing	573
26.2.5	Testing in a Production Environment	573
26.3	Considering the Spectrum of User Interaction	574
26.3.1	Gesture Testing	575
26.3.2	Voice Input and Recognition	576
26.3.3	Virtual Key Board Input	577
26.3.4	Alerts and Extraordinary Conditions	577
26.4	Test Across Borders	578
26.5	Real-Time Testing Issues	578
26.6	Testing Tools and Environments	579
26.7	Summary	581
	PROBLEMS AND POINTS TO PONDER	582
	FURTHER READINGS AND INFORMATION SOURCES	582

CHAPTER 27 SECURITY ENGINEERING 584

27.1	Analyzing Security Requirements	585
27.2	Security and Privacy in an Online World	586
27.2.1	Social Media	587
27.2.2	Mobile Applications	587
27.2.3	Cloud Computing	587
27.2.4	The Internet of Things	588
27.3	Security Engineering Analysis	588
27.3.1	Security Requirement Elicitation	589
27.3.2	Security Modeling	590
27.3.3	Measures Design	591
27.3.4	Correctness Checks	591
27.4	Security Assurance	592
27.4.1	The Security Assurance Process	592
27.4.2	Organization and Management	593

27.5	Security Risk Analysis	594
27.6	The Role of Conventional Software Engineering Activities	595
27.7	Verification of Trustworthy Systems	597
27.8	Summary	599
	PROBLEMS AND POINTS TO PONDER	599
	FURTHER READINGS AND INFORMATION SOURCES	600

CHAPTER 28 FORMAL MODELING AND VERIFICATION 601

28.1	The Cleanroom Strategy	602
28.2	Functional Specification	604
	28.2.1 Black-Box Specification	605
	28.2.2 State-Box Specification	606
	28.2.3 Clear-Box Specification	607
28.3	Cleanroom Design	607
	28.3.1 Design Refinement	608
	28.3.2 Design Verification	608
28.4	Cleanroom Testing	610
	28.4.1 Statistical Use Testing	610
	28.4.2 Certification	612
28.5	Rethinking Formal Methods	612
28.6	Formal Methods Concepts	615
28.7	Alternative Arguments	618
28.8	Summary	619
	PROBLEMS AND POINTS TO PONDER	620
	FURTHER READINGS AND INFORMATION SOURCES	621

CHAPTER 29 SOFTWARE CONFIGURATION MANAGEMENT 623

29.1	Software Configuration Management	624
	29.1.1 An SCM Scenario	625
	29.1.2 Elements of a Configuration Management System	626
	29.1.3 Baselines	626
	29.1.4 Software Configuration Items	628
	29.1.5 Management of Dependencies and Changes	628
29.2	The SCM Repository	630
	29.2.1 General Features and Content	630
	29.2.2 SCM Features	631
29.3	The SCM Process	632
	29.3.1 Identification of Objects in the Software Configuration	633
	29.3.2 Version Control	634
	29.3.3 Change Control	635
	29.3.4 Impact Management	638
	29.3.5 Configuration Audit	639
	29.3.6 Status Reporting	639
29.4	Configuration Management for Web and MobileApps	640
	29.4.1 Dominant Issues	641
	29.4.2 Configuration Objects	642
	29.4.3 Content Management	643
	29.4.4 Change Management	646
	29.4.5 Version Control	648
	29.4.6 Auditing and Reporting	649

29.5	Summary	650
	PROBLEMS AND POINTS TO PONDER	651
	FURTHER READINGS AND INFORMATION SOURCES	651

CHAPTER 30 PRODUCT METRICS 653

30.1	A Framework for Product Metrics	654
30.1.1	Measures, Metrics, and Indicators	654
30.1.2	The Challenge of Product Metrics	655
30.1.3	Measurement Principles	656
30.1.4	Goal-Oriented Software Measurement	656
30.1.5	The Attributes of Effective Software Metrics	657
30.2	Metrics for the Requirements Model	659
30.2.1	Function-Based Metrics	659
30.2.2	Metrics for Specification Quality	662
30.3	Metrics for the Design Model	663
30.3.1	Architectural Design Metrics	663
30.3.2	Metrics for Object-Oriented Design	666
30.3.3	Class-Oriented Metrics—The CK Metrics Suite	667
30.3.4	Class-Oriented Metrics—The MOOD Metrics Suite	670
30.3.5	OO Metrics Proposed by Lorenz and Kidd	671
30.3.6	Component-Level Design Metrics	671
30.3.7	Operation-Oriented Metrics	671
30.3.8	User Interface Design Metrics	672
30.4	Design Metrics for Web and Mobile Apps	672
30.5	Metrics for Source Code	675
30.6	Metrics for Testing	676
30.6.1	Halstead Metrics Applied to Testing	676
30.6.2	Metrics for Object-Oriented Testing	677
30.7	Metrics for Maintenance	678
30.8	Summary	679
	PROBLEMS AND POINTS TO PONDER	679
	FURTHER READINGS AND INFORMATION SOURCES	680

PART FOUR MANAGING SOFTWARE PROJECTS 683

CHAPTER 31 PROJECT MANAGEMENT CONCEPTS 684

31.1	The Management Spectrum	685
31.1.1	The People	685
31.1.2	The Product	686
31.1.3	The Process	686
31.1.4	The Project	686
31.2	People	687
31.2.1	The Stakeholders	687
31.2.2	Team Leaders	688
31.2.3	The Software Team	689
31.2.4	Agile Teams	691
31.2.5	Coordination and Communication Issues	692
31.3	The Product	693
31.3.1	Software Scope	694
31.3.2	Problem Decomposition	694

31.4	The Process	694
31.4.1	Melding the Product and the Process	695
31.4.2	Process Decomposition	696
31.5	The Project	697
31.6	The W ² HH Principle	698
31.7	Critical Practices	699
31.8	Summary	700
	PROBLEMS AND POINTS TO PONDER	700
	FURTHER READINGS AND INFORMATION SOURCES	701

CHAPTER 32 PROCESS AND PROJECT METRICS 703

32.1	Metrics in the Process and Project Domains	704
32.1.1	Process Metrics and Software Process Improvement	704
32.1.2	Project Metrics	707
32.2	Software Measurement	708
32.2.1	Size-Oriented Metrics	709
32.2.2	Function-Oriented Metrics	710
32.2.3	Reconciling LOC and FP Metrics	711
32.2.4	Object-Oriented Metrics	713
32.2.5	Use Case-Oriented Metrics	714
32.2.6	WebApp Project Metrics	714
32.3	Metrics for Software Quality	716
32.3.1	Measuring Quality	717
32.3.2	Defect Removal Efficiency	718
32.4	Integrating Metrics within the Software Process	719
32.4.1	Arguments for Software Metrics	720
32.4.2	Establishing a Baseline	720
32.4.3	Metrics Collection, Computation, and Evaluation	721
32.5	Metrics for Small Organizations	721
32.6	Establishing a Software Metrics Program	722
32.7	Summary	724
	PROBLEMS AND POINTS TO PONDER	724
	FURTHER READINGS AND INFORMATION SOURCES	725

CHAPTER 33 ESTIMATION FOR SOFTWARE PROJECTS 727

33.1	Observations on Estimation	728
33.2	The Project Planning Process	729
33.3	Software Scope and Feasibility	730
33.4	Resources	731
33.4.1	Human Resources	731
33.4.2	Reusable Software Resources	732
33.4.3	Environmental Resources	732
33.5	Software Project Estimation	733
33.6	Decomposition Techniques	734
33.6.1	Software Sizing	734
33.6.2	Problem-Based Estimation	735
33.6.3	An Example of LOC-Based Estimation	736
33.6.4	An Example of FP-Based Estimation	738
33.6.5	Process-Based Estimation	739
33.6.6	An Example of Process-Based Estimation	740
33.6.7	Estimation with Use Cases	740

33.6.8	An Example of Estimation Using Use Case Points	742
33.6.9	Reconciling Estimates	742
33.7	Empirical Estimation Models	743
33.7.1	The Structure of Estimation Models	744
33.7.2	The COCOMO II Model	744
33.7.3	The Software Equation	744
33.8	Estimation for Object-Oriented Projects	746
33.9	Specialized Estimation Techniques	746
33.9.1	Estimation for Agile Development	746
33.9.2	Estimation for WebApp Projects	747
33.10	The Make/Buy Decision	748
33.10.1	Creating a Decision Tree	749
33.10.2	Outsourcing	750
33.11	Summary	752
	PROBLEMS AND POINTS TO PONDER	752
	FURTHER READINGS AND INFORMATION SOURCES	753

CHAPTER 34 PROJECT SCHEDULING 754

34.1	Basic Concepts	755
34.2	Project Scheduling	757
34.2.1	Basic Principles	758
34.2.2	The Relationship between People and Effort	759
34.2.3	Effort Distribution	760
34.3	Defining a Task Set for the Software Project	761
34.3.1	A Task Set Example	762
34.3.2	Refinement of Major Tasks	763
34.4	Defining a Task Network	764
34.5	Scheduling	765
34.5.1	Time-Line Charts	766
34.5.2	Tracking the Schedule	767
34.5.3	Tracking Progress for an OO Project	768
34.5.4	Scheduling for WebApp and Mobile Projects	769
34.6	Earned Value Analysis	772
34.7	Summary	774
	PROBLEMS AND POINTS TO PONDER	774
	FURTHER READINGS AND INFORMATION SOURCES	776

CHAPTER 35 RISK MANAGEMENT 777

35.1	Reactive versus Proactive Risk Strategies	778
35.2	Software Risks	778
35.3	Risk Identification	780
35.3.1	Assessing Overall Project Risk	781
35.3.2	Risk Components and Drivers	782
35.4	Risk Projection	782
35.4.1	Developing a Risk Table	783
35.4.2	Assessing Risk Impact	785
35.5	Risk Refinement	787
35.6	Risk Mitigation, Monitoring, and Management	788
35.7	The RMMM Plan	790
35.8	Summary	792

PROBLEMS AND POINTS TO PONDER	792
FURTHER READINGS AND INFORMATION SOURCES	793

CHAPTER 36 MAINTENANCE AND REENGINEERING 795

36.1	Software Maintenance	796
36.2	Software Supportability	798
36.3	Reengineering	798
36.4	Business Process Reengineering	799
	36.4.1 Business Processes	799
	36.4.2 A BPR Model	800
36.5	Software Reengineering	802
	36.5.1 A Software Reengineering Process Model	802
	36.5.2 Software Reengineering Activities	803
36.6	Reverse Engineering	805
	36.6.1 Reverse Engineering to Understand Data	807
	36.6.2 Reverse Engineering to Understand Processing	807
	36.6.3 Reverse Engineering User Interfaces	808
36.7	Restructuring	809
	36.7.1 Code Restructuring	809
	36.7.2 Data Restructuring	810
36.8	Forward Engineering	811
	36.8.1 Forward Engineering for Client-Server Architectures	812
	36.8.2 Forward Engineering for Object-Oriented Architectures	813
36.9	The Economics of Reengineering	813
36.10	Summary	814
PROBLEMS AND POINTS TO PONDER		815
FURTHER READINGS AND INFORMATION SOURCES		816

PART FIVE ADVANCED TOPICS 817

CHAPTER 37 SOFTWARE PROCESS IMPROVEMENT 818

37.1	What Is SPI?	819
	37.1.1 Approaches to SPI	819
	37.1.2 Maturity Models	821
	37.1.3 Is SPI for Everyone?	822
37.2	The SPI Process	823
	37.2.1 Assessment and Gap Analysis	823
	37.2.2 Education and Training	825
	37.2.3 Selection and Justification	825
	37.2.4 Installation/Migration	826
	37.2.5 Evaluation	827
	37.2.6 Risk Management for SPI	827
37.3	The CMMI	828
37.4	The People CMM	832
37.5	Other SPI Frameworks	832
37.6	SPI Return on Investment	834
37.7	SPI Trends	835
37.8	Summary	836
PROBLEMS AND POINTS TO PONDER		837
FURTHER READINGS AND INFORMATION SOURCES		837

CHAPTER 38 EMERGING TRENDS IN SOFTWARE ENGINEERING 839

38.1	Technology Evolution	840
38.2	Prospects for a True Engineering Discipline	841
38.3	Observing Software Engineering Trends	842
38.4	Identifying “Soft Trends”	843
38.4.1	Managing Complexity	845
38.4.2	Open-World Software	846
38.4.3	Emergent Requirements	846
38.4.4	The Talent Mix	847
38.4.5	Software Building Blocks	847
38.4.6	Changing Perceptions of “Value”	848
38.4.7	Open Source	848
38.5	Technology Directions	849
38.5.1	Process Trends	849
38.5.2	The Grand Challenge	851
38.5.3	Collaborative Development	852
38.5.4	Requirements Engineering	852
38.5.5	Model-Driven Software Development	853
38.5.6	Postmodern Design	854
38.5.7	Test-Driven Development	854
38.6	Tools-Related Trends	855
38.7	Summary	857
	PROBLEMS AND POINTS TO PONDER	857
	FURTHER READINGS AND INFORMATION SOURCES	858

CHAPTER 39 CONCLUDING COMMENTS 860

39.1	The Importance of Software—Revisited	861
39.2	People and the Way They Build Systems	861
39.3	New Modes for Representing Information	862
39.4	The Long View	864
39.5	The Software Engineer’s Responsibility	865
39.6	A Final Comment from RSP	867
APPENDIX 1	AN INTRODUCTION TO UML	869
APPENDIX 2	OBJECT-ORIENTED CONCEPTS	891
APPENDIX 3	FORMAL METHODS	899
REFERENCES		909
INDEX		933