

Software Engineering

A PRACTITIONER'S APPROACH

SEVENTH EDITION

Roger S. Pressman, Ph.D.



Higher Education

Boston Burr Ridge, IL Dubuque, IA New York San Francisco St. Louis
Bangkok Bogotá Caracas Kuala Lumpur Lisbon London Madrid Mexico City
Milan Montreal New Delhi Santiago Seoul Singapore Sydney Taipei Toronto

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.

CONTENTS AT A GLANCE

CHAPTER 1 Software and Software Engineering 1

PART ONE THE SOFTWARE PROCESS 29

CHAPTER 2 Process Models 30

CHAPTER 3 Agile Development 65

PART TWO MODELING 95

CHAPTER 4 Principles that Guide Practice 96

CHAPTER 5 Understanding Requirements 119

CHAPTER 6 Requirements Modeling: Scenarios, Information, and Analysis Classes 148

CHAPTER 7 Requirements Modeling: Flow, Behavior, Patterns, and WebApps 186

CHAPTER 8 Design Concepts 215

CHAPTER 9 Architectural Design 242

CHAPTER 10 Component-Level Design 276

CHAPTER 11 User Interface Design 312

CHAPTER 12 Pattern-Based Design 347

CHAPTER 13 WebApp Design 373

PART THREE QUALITY MANAGEMENT 397

CHAPTER 14 Quality Concepts 398

CHAPTER 15 Review Techniques 416

CHAPTER 16 Software Quality Assurance 432

CHAPTER 17 Software Testing Strategies 449

CHAPTER 18 Testing Conventional Applications 481

CHAPTER 19 Testing Object-Oriented Applications 511

CHAPTER 20 Testing Web Applications 529

CHAPTER 21 Formal Modeling and Verification 557

CHAPTER 22 Software Configuration Management 584

CHAPTER 23 Product Metrics 613

PART FOUR MANAGING SOFTWARE PROJECTS 645

CHAPTER 24 Project Management Concepts 646

CHAPTER 25 Process and Project Metrics 666

CHAPTER 26	Estimation for Software Projects	691
CHAPTER 27	Project Scheduling	721
CHAPTER 28	Risk Management	744
CHAPTER 29	Maintenance and Reengineering	761

PART FIVE **ADVANCED TOPICS** 785

CHAPTER 30	Software Process Improvement	786
CHAPTER 31	Emerging Trends in Software Engineering	808
CHAPTER 32	Concluding Comments	833
APPENDIX 1	An Introduction to UML	841
APPENDIX 2	Object-Oriented Concepts	863
REFERENCES		871
INDEX		889

TABLE OF CONTENTS

Preface xxv

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

10.7	Summary	309
	PROBLEMS AND POINTS TO PONDER	310
	FURTHER READINGS AND INFORMATION SOURCES	311

CHAPTER 11 USER INTERFACE DESIGN 312

11.1	The Golden Rules	313
	11.1.1	Place the User in Control 313
	11.1.2	Reduce the User's Memory Load 314
	11.1.3	Make the Interface Consistent 316
11.2	User Interface Analysis and Design	317
	11.2.1	Interface Analysis and Design Models 317
	11.2.2	The Process 319
11.3	Interface Analysis	320
	11.3.1	User Analysis 321
	11.3.2	Task Analysis and Modeling 322
	11.3.3	Analysis of Display Content 327
	11.3.4	Analysis of the Work Environment 328
11.4	Interface Design Steps	328
	11.4.1	Applying Interface Design Steps 329
	11.4.2	User Interface Design Patterns 330
	11.4.3	Design Issues 331
11.5	WebApp Interface Design	335
	11.5.1	Interface Design Principles and Guidelines 336
	11.5.2	Interface Design Workflow for WebApps 340
11.6	Design Evaluation	342
11.7	Summary	344
	PROBLEMS AND POINTS TO PONDER	345
	FURTHER READINGS AND INFORMATION SOURCES	346

CHAPTER 12 PATTERN-BASED DESIGN 347

12.1	Design Patterns	348
	12.1.1	Kinds of Patterns 349
	12.1.2	Frameworks 352
	12.1.3	Describing a Pattern 352
	12.1.4	Pattern Languages and Repositories 353
12.2	Pattern-Based Software Design	354
	12.2.1	Pattern-Based Design in Context 354
	12.2.2	Thinking in Patterns 356
	12.2.3	Design Tasks 357
	12.2.4	Building a Pattern-Organizing Table 358
	12.2.5	Common Design Mistakes 359
12.3	Architectural Patterns	360
12.4	Component-Level Design Patterns	362
12.5	User Interface Design Patterns	364
12.6	WebApp Design Patterns	368
	12.6.1	Design Focus 368
	12.6.2	Design Granularity 369
12.7	Summary	370
	PROBLEMS AND POINTS TO PONDER	371
	FURTHER READING AND INFORMATION SOURCES	372

CHAPTER 13 WEBAPP DESIGN 373

13.1	WebApp Design Quality	374
13.2	Design Goals	377
13.3	A Design Pyramid for WebApps	378
13.4	WebApp Interface Design	378
13.5	Aesthetic Design	380
	13.5.1	Layout Issues 380
	13.5.2	Graphic Design Issues 381
13.6	Content Design	382
	13.6.1	Content Objects 382
	13.6.2	Content Design Issues 382
13.7	Architecture Design	383
	13.7.1	Content Architecture 384
	13.7.2	WebApp Architecture 386
13.8	Navigation Design	388
	13.8.1	Navigation Semantics 388
	13.8.2	Navigation Syntax 389
13.9	Component-Level Design	390
13.10	Object-Oriented Hypermedia Design Method (OOHDM)	390
	13.10.1	Conceptual Design for OOHDM 391
	13.10.2	Navigational Design for OOHDM 391
	13.10.3	Abstract Interface Design and Implementation 392
13.11	Summary	393
	PROBLEMS AND POINTS TO PONDER	394
	FURTHER READINGS AND INFORMATION SOURCES	395

PART THREE QUALITY MANAGEMENT 397

CHAPTER 14 QUALITY CONCEPTS 398

14.1	What Is Quality?	399
14.2	Software Quality	400
	14.2.1	Garvin's Quality Dimensions 401
	14.2.2	McCall's Quality Factors 402
	14.2.3	ISO 9126 Quality Factors 403
	14.2.4	Targeted Quality Factors 404
	14.2.5	The Transition to a Quantitative View 405
14.3	The Software Quality Dilemma	406
	14.3.1	"Good Enough" Software 406
	14.3.2	The Cost of Quality 407
	14.3.3	Risks 409
	14.3.4	Negligence and Liability 410
	14.3.5	Quality and Security 410
	14.3.6	The Impact of Management Actions 411
14.4	Achieving Software Quality	412
	14.4.1	Software Engineering Methods 412
	14.4.2	Project Management Techniques 412
	14.4.3	Quality Control 412
	14.4.4	Quality Assurance 413
14.5	Summary	413
	PROBLEMS AND POINTS TO PONDER	414
	FURTHER READINGS AND INFORMATION SOURCES	414

CHAPTER 15 REVIEW TECHNIQUES 416

- 15.1 Cost Impact of Software Defects 417
- 15.2 Defect Amplification and Removal 418
- 15.3 Review Metrics and Their Use 420
 - 15.3.1 Analyzing Metrics 420
 - 15.3.2 Cost Effectiveness of Reviews 421
- 15.4 Reviews: A Formality Spectrum 423
- 15.5 Informal Reviews 424
- 15.6 Formal Technical Reviews 426
 - 15.6.1 The Review Meeting 426
 - 15.6.2 Review Reporting and Record Keeping 427
 - 15.6.3 Review Guidelines 427
 - 15.6.4 Sample-Driven Reviews 429
- 15.7 Summary 430
- PROBLEMS AND POINTS TO PONDER 431
- FURTHER READINGS AND INFORMATION SOURCES 431

CHAPTER 16 SOFTWARE QUALITY ASSURANCE 432

- 16.1 Background Issues 433
- 16.2 Elements of Software Quality Assurance 434
- 16.3 SQA Tasks, Goals, and Metrics 436
 - 16.3.1 SQA Tasks 436
 - 16.3.2 Goals, Attributes, and Metrics 437
- 16.4 Formal Approaches to SQA 438
- 16.5 Statistical Software Quality Assurance 439
 - 16.5.1 A Generic Example 439
 - 16.5.2 Six Sigma for Software Engineering 441
- 16.6 Software Reliability 442
 - 16.6.1 Measures of Reliability and Availability 442
 - 16.6.2 Software Safety 443
- 16.7 The ISO 9000 Quality Standards 444
- 16.8 The SQA Plan 445
- 16.9 Summary 446
- PROBLEMS AND POINTS TO PONDER 447
- FURTHER READINGS AND INFORMATION SOURCES 447

CHAPTER 17 SOFTWARE TESTING STRATEGIES 449

- 17.1 A Strategic Approach to Software Testing 450
 - 17.1.1 Verification and Validation 450
 - 17.1.2 Organizing for Software Testing 451
 - 17.1.3 Software Testing Strategy—The Big Picture 452
 - 17.1.4 Criteria for Completion of Testing 455
- 17.2 Strategic Issues 455
- 17.3 Test Strategies for Conventional Software 456
 - 17.3.1 Unit Testing 456
 - 17.3.2 Integration Testing 459
- 17.4 Test Strategies for Object-Oriented Software 465
 - 17.4.1 Unit Testing in the OO Context 466
 - 17.4.2 Integration Testing in the OO Context 466
- 17.5 Test Strategies for WebApps 467
- 17.6 Validation Testing 467

	17.6.1	Validation-Test Criteria	468
	17.6.2	Configuration Review	468
	17.6.3	Alpha and Beta Testing	468
17.7		System Testing	470
	17.7.1	Recovery Testing	470
	17.7.2	Security Testing	470
	17.7.3	Stress Testing	471
	17.7.4	Performance Testing	471
	17.7.5	Deployment Testing	472
17.8		The Art of Debugging	473
	17.8.1	The Debugging Process	473
	17.8.2	Psychological Considerations	474
	17.8.3	Debugging Strategies	475
	17.8.4	Correcting the Error	477
17.9		Summary	478
		PROBLEMS AND POINTS TO PONDER	478
		FURTHER READINGS AND INFORMATION SOURCES	479

CHAPTER 18 TESTING CONVENTIONAL APPLICATIONS 481

18.1		Software Testing Fundamentals	482
18.2		Internal and External Views of Testing	484
18.3		White-Box Testing	485
18.4		Basis Path Testing	485
	18.4.1	Flow Graph Notation	485
	18.4.2	Independent Program Paths	487
	18.4.3	Deriving Test Cases	489
	18.4.4	Graph Matrices	491
18.5		Control Structure Testing	492
	18.5.1	Condition Testing	492
	18.5.2	Data Flow Testing	493
	18.5.3	Loop Testing	493
18.6		Black-Box Testing	495
	18.6.1	Graph-Based Testing Methods	495
	18.6.2	Equivalence Partitioning	497
	18.6.3	Boundary Value Analysis	498
	18.6.4	Orthogonal Array Testing	499
18.7		Model-Based Testing	502
18.8		Testing for Specialized Environments, Architectures, and Applications	503
	18.8.1	Testing GUIs	503
	18.8.2	Testing of Client-Server Architectures	503
	18.8.3	Testing Documentation and Help Facilities	505
	18.8.4	Testing for Real-Time Systems	506
18.9		Patterns for Software Testing	507
18.10		Summary	508
		PROBLEMS AND POINTS TO PONDER	509
		FURTHER READINGS AND INFORMATION SOURCES	510

CHAPTER 19 TESTING OBJECT-ORIENTED APPLICATIONS 511

19.1		Broadening the View of Testing	512
19.2		Testing OOA and OOD Models	513

	19.2.1	Correctness of OOA and OOD Models	513
	19.2.2	Consistency of Object-Oriented Models	514
19.3		Object-Oriented Testing Strategies	516
	19.3.1	Unit Testing in the OO Context	516
	19.3.2	Integration Testing in the OO Context	516
	19.3.3	Validation Testing in an OO Context	517
19.4		Object-Oriented Testing Methods	517
	19.4.1	The Test-Case Design Implications of OO Concepts	518
	19.4.2	Applicability of Conventional Test-Case Design Methods	518
	19.4.3	Fault-Based Testing	519
	19.4.4	Test Cases and the Class Hierarchy	519
	19.4.5	Scenario-Based Test Design	520
	19.4.6	Testing Surface Structure and Deep Structure	522
19.5		Testing Methods Applicable at the Class Level	522
	19.5.1	Random Testing for OO Classes	522
	19.5.2	Partition Testing at the Class Level	524
19.6		Interclass Test-Case Design	524
	19.6.1	Multiple Class Testing	524
	19.6.2	Tests Derived from Behavior Models	526
19.7		Summary	527
		PROBLEMS AND POINTS TO PONDER	528
		FURTHER READINGS AND INFORMATION SOURCES	528

CHAPTER 20 TESTING WEB APPLICATIONS 529

20.1		Testing Concepts for WebApps	530
	20.1.1	Dimensions of Quality	530
	20.1.2	Errors within a WebApp Environment	531
	20.1.3	Testing Strategy	532
	20.1.4	Test Planning	532
20.2		The Testing Process—An Overview	533
20.3		Content Testing	534
	20.3.1	Content Testing Objectives	534
	20.3.2	Database Testing	535
20.4		User Interface Testing	537
	20.4.1	Interface Testing Strategy	537
	20.4.2	Testing Interface Mechanisms	538
	20.4.3	Testing Interface Semantics	540
	20.4.4	Usability Tests	540
	20.4.5	Compatibility Tests	542
20.5		Component-Level Testing	543
20.6		Navigation Testing	545
	20.6.1	Testing Navigation Syntax	545
	20.6.2	Testing Navigation Semantics	546
20.7		Configuration Testing	547
	20.7.1	Server-Side Issues	547
	20.7.2	Client-Side Issues	548
20.8		Security Testing	548
20.9		Performance Testing	550
	20.9.1	Performance Testing Objectives	550
	20.9.2	Load Testing	551
	20.9.3	Stress Testing	552

20.10	Summary	553
	PROBLEMS AND POINTS TO PONDER	554
	FURTHER READINGS AND INFORMATION SOURCES	555

CHAPTER 21 FORMAL MODELING AND VERIFICATION 557

21.1	The Cleanroom Strategy	558
21.2	Functional Specification	560
	21.2.1 Black-Box Specification	561
	21.2.2 State-Box Specification	562
	21.2.3 Clear-Box Specification	562
21.3	Cleanroom Design	563
	21.3.1 Design Refinement	563
	21.3.2 Design Verification	564
21.4	Cleanroom Testing	566
	21.4.1 Statistical Use Testing	566
	21.4.2 Certification	567
21.5	Formal Methods Concepts	568
21.6	Applying Mathematical Notation for Formal Specification	571
21.7	Formal Specification Languages	573
	21.7.1 Object Constraint Language (OCL)	574
	21.7.2 The Z Specification Language	577
21.8	Summary	580
	PROBLEMS AND POINTS TO PONDER	581
	FURTHER READINGS AND INFORMATION SOURCES	582

CHAPTER 22 SOFTWARE CONFIGURATION MANAGEMENT 584

22.1	Software Configuration Management	585
	22.1.1 An SCM Scenario	586
	22.1.2 Elements of a Configuration Management System	587
	22.1.3 Baselines	587
	22.1.4 Software Configuration Items	589
22.2	The SCM Repository	590
	22.2.1 The Role of the Repository	590
	22.2.2 General Features and Content	591
	22.2.3 SCM Features	592
22.3	The SCM Process	593
	22.3.1 Identification of Objects in the Software Configuration	594
	22.3.2 Version Control	595
	22.3.3 Change Control	596
	22.3.4 Configuration Audit	599
	22.3.5 Status Reporting	600
22.4	Configuration Management for WebApps	601
	22.4.1 Dominant Issues	601
	22.4.2 WebApp Configuration Objects	603
	22.4.3 Content Management	603
	22.4.4 Change Management	606
	22.4.5 Version Control	608
	22.4.6 Auditing and Reporting	609
22.5	Summary	610
	PROBLEMS AND POINTS TO PONDER	611
	FURTHER READINGS AND INFORMATION SOURCES	612

CHAPTER 23 PRODUCT METRICS 613

- 23.1 A Framework for Product Metrics 614
 - 23.1.1 Measures, Metrics, and Indicators 614
 - 23.1.2 The Challenge of Product Metrics 615
 - 23.1.3 Measurement Principles 616
 - 23.1.4 Goal-Oriented Software Measurement 617
 - 23.1.5 The Attributes of Effective Software Metrics 618
- 23.2 Metrics for the Requirements Model 619
 - 23.2.1 Function-Based Metrics 620
 - 23.2.2 Metrics for Specification Quality 623
- 23.3 Metrics for the Design Model 624
 - 23.3.1 Architectural Design Metrics 624
 - 23.3.2 Metrics for Object-Oriented Design 627
 - 23.3.3 Class-Oriented Metrics—The CK Metrics Suite 628
 - 23.3.4 Class-Oriented Metrics—The MOOD Metrics Suite 631
 - 23.3.5 OO Metrics Proposed by Lorenz and Kidd 632
 - 23.3.6 Component-Level Design Metrics 632
 - 23.3.7 Operation-Oriented Metrics 634
 - 23.3.8 User Interface Design Metrics 635
- 23.4 Design Metrics for WebApps 636
- 23.5 Metrics for Source Code 638
- 23.6 Metrics for Testing 639
 - 23.6.1 Halstead Metrics Applied to Testing 639
 - 23.6.2 Metrics for Object-Oriented Testing 640
- 23.7 Metrics for Maintenance 641
- 23.8 Summary 642
- PROBLEMS AND POINTS TO PONDER 642
- FURTHER READINGS AND INFORMATION SOURCES 643

PART FOUR MANAGING SOFTWARE PROJECTS 645

CHAPTER 24 PROJECT MANAGEMENT CONCEPTS 646

- 24.1 The Management Spectrum 647
 - 24.1.1 The People 647
 - 24.1.2 The Product 648
 - 24.1.3 The Process 648
 - 24.1.4 The Project 648
- 24.2 People 649
 - 24.2.1 The Stakeholders 649
 - 24.2.2 Team Leaders 650
 - 24.2.3 The Software Team 651
 - 24.2.4 Agile Teams 654
 - 24.2.5 Coordination and Communication Issues 655
- 24.3 The Product 656
 - 24.3.1 Software Scope 656
 - 24.3.2 Problem Decomposition 656
- 24.4 The Process 657
 - 24.4.1 Melding the Product and the Process 657
 - 24.4.2 Process Decomposition 658
- 24.5 The Project 660
- 24.6 The W⁵HH Principle 661

24.7	Critical Practices	662
24.8	Summary	663
	PROBLEMS AND POINTS TO PONDER	663
	FURTHER READINGS AND INFORMATION SOURCES	664

CHAPTER 25 PROCESS AND PROJECT METRICS 666

25.1	Metrics in the Process and Project Domains	667
25.1.1	Process Metrics and Software Process Improvement	667
25.1.2	Project Metrics	670
25.2	Software Measurement	671
25.2.1	Size-Oriented Metrics	672
25.2.2	Function-Oriented Metrics	673
25.2.3	Reconciling LOC and FP Metrics	673
25.2.4	Object-Oriented Metrics	675
25.2.5	Use-Case-Oriented Metrics	676
25.2.6	WebApp Project Metrics	677
25.3	Metrics for Software Quality	679
25.3.1	Measuring Quality	680
25.3.2	Defect Removal Efficiency	681
25.4	Integrating Metrics within the Software Process	682
25.4.1	Arguments for Software Metrics	683
25.4.2	Establishing a Baseline	683
25.4.3	Metrics Collection, Computation, and Evaluation	684
25.5	Metrics for Small Organizations	684
25.6	Establishing a Software Metrics Program	686
25.7	Summary	688
	PROBLEMS AND POINTS TO PONDER	688
	FURTHER READINGS AND INFORMATION SOURCES	689

CHAPTER 26 ESTIMATION FOR SOFTWARE PROJECTS 691

26.1	Observations on Estimation	692
26.2	The Project Planning Process	693
26.3	Software Scope and Feasibility	694
26.4	Resources	695
26.4.1	Human Resources	695
26.4.2	Reusable Software Resources	696
26.4.3	Environmental Resources	696
26.5	Software Project Estimation	697
26.6	Decomposition Techniques	698
26.6.1	Software Sizing	698
26.6.2	Problem-Based Estimation	699
26.6.3	An Example of LOC-Based Estimation	701
26.6.4	An Example of FP-Based Estimation	702
26.6.5	Process-Based Estimation	703
26.6.6	An Example of Process-Based Estimation	704
26.6.7	Estimation with Use Cases	705
26.6.8	An Example of Use-Case-Based Estimation	706
26.6.9	Reconciling Estimates	707
26.7	Empirical Estimation Models	708
26.7.1	The Structure of Estimation Models	709
26.7.2	The COCOMO II Model	709
26.7.3	The Software Equation	711

26.8	Estimation for Object-Oriented Projects	712
26.9	Specialized Estimation Techniques	713
26.9.1	Estimation for Agile Development	713
26.9.2	Estimation for WebApp Projects	714
26.10	The Make/Buy Decision	715
26.10.1	Creating a Decision Tree	715
26.10.2	Outsourcing	717
26.11	Summary	718
	PROBLEMS AND POINTS TO PONDER	719
	FURTHER READINGS AND INFORMATION SOURCES	719

CHAPTER 27 PROJECT SCHEDULING 721

27.1	Basic Concepts	722
27.2	Project Scheduling	724
27.2.1	Basic Principles	725
27.2.2	The Relationship Between People and Effort	725
27.2.3	Effort Distribution	727
27.3	Defining a Task Set for the Software Project	728
27.3.1	A Task Set Example	729
27.3.2	Refinement of Software Engineering Actions	730
27.4	Defining a Task Network	731
27.5	Scheduling	732
27.5.1	Time-Line Charts	732
27.5.2	Tracking the Schedule	734
27.5.3	Tracking Progress for an OO Project	735
27.5.4	Scheduling for WebApp Projects	736
27.6	Earned Value Analysis	739
27.7	Summary	741
	PROBLEMS AND POINTS TO PONDER	741
	FURTHER READINGS AND INFORMATION SOURCES	743

CHAPTER 28 RISK MANAGEMENT 744

28.1	Reactive versus Proactive Risk Strategies	745
28.2	Software Risks	745
28.3	Risk Identification	747
28.3.1	Assessing Overall Project Risk	748
28.3.2	Risk Components and Drivers	749
28.4	Risk Projection	749
28.4.1	Developing a Risk Table	750
28.4.2	Assessing Risk Impact	752
28.5	Risk Refinement	754
28.6	Risk Mitigation, Monitoring, and Management	755
28.7	The RMMM Plan	757
28.8	Summary	759
	PROBLEMS AND POINTS TO PONDER	759
	FURTHER READINGS AND INFORMATION SOURCES	760

CHAPTER 29 MAINTENANCE AND REENGINEERING 761

29.1	Software Maintenance	762
29.2	Software Supportability	764

29.3	Reengineering	764
29.4	Business Process Reengineering	765
29.4.1	Business Processes	765
29.4.2	A BPR Model	766
29.5	Software Reengineering	768
29.5.1	A Software Reengineering Process Model	768
29.5.2	Software Reengineering Activities	770
29.6	Reverse Engineering	772
29.6.1	Reverse Engineering to Understand Data	773
29.6.2	Reverse Engineering to Understand Processing	774
29.6.3	Reverse Engineering User Interfaces	775
29.7	Restructuring	776
29.7.1	Code Restructuring	776
29.7.2	Data Restructuring	777
29.8	Forward Engineering	778
29.8.1	Forward Engineering for Client-Server Architectures	779
29.8.2	Forward Engineering for Object-Oriented Architectures	780
29.9	The Economics of Reengineering	780
29.10	Summary	781
	PROBLEMS AND POINTS TO PONDER	782
	FURTHER READINGS AND INFORMATION SOURCES	783

PART FIVE ADVANCED TOPICS 785

CHAPTER 30 SOFTWARE PROCESS IMPROVEMENT 786

30.1	What Is SPI?	787
30.1.1	Approaches to SPI	787
30.1.2	Maturity Models	789
30.1.3	Is SPI for Everyone?	790
30.2	The SPI Process	791
30.2.1	Assessment and Gap Analysis	791
30.2.2	Education and Training	793
30.2.3	Selection and Justification	793
30.2.4	Installation/Migration	794
30.2.5	Evaluation	795
30.2.6	Risk Management for SPI	795
30.2.7	Critical Success Factors	796
30.3	The CMMI	797
30.4	The People CMM	801
30.5	Other SPI Frameworks	802
30.6	SPI Return on Investment	804
30.7	SPI Trends	805
30.8	Summary	806
	PROBLEMS AND POINTS TO PONDER	806
	FURTHER READINGS AND INFORMATION SOURCES	807

CHAPTER 31 EMERGING TRENDS IN SOFTWARE ENGINEERING 808

31.1	Technology Evolution	809
31.2	Observing Software Engineering Trends	811

31.3	Identifying “Soft Trends”	812
31.3.1	Managing Complexity	814
31.3.2	Open-World Software	815
31.3.3	Emergent Requirements	816
31.3.4	The Talent Mix	816
31.3.5	Software Building Blocks	817
31.3.6	Changing Perceptions of “Value”	818
31.3.7	Open Source	818
31.4	Technology Directions	819
31.4.1	Process Trends	819
31.4.2	The Grand Challenge	821
31.4.3	Collaborative Development	822
31.4.4	Requirements Engineering	824
31.4.5	Model-Driven Software Development	825
31.4.6	Postmodern Design	825
31.4.7	Test-Driven Development	826
31.5	Tools-Related Trends	827
31.5.1	Tools That Respond to Soft Trends	828
31.5.2	Tools That Address Technology Trends	830
31.6	Summary	830
	PROBLEMS AND POINTS TO PONDER	831
	FURTHER READINGS AND INFORMATION SOURCES	831

CHAPTER 32 CONCLUDING COMMENTS 833

32.1	The Importance of Software—Revisited	834
32.2	People and the Way They Build Systems	834
32.3	New Modes for Representing Information	835
32.4	The Long View	837
32.5	The Software Engineer’s Responsibility	838
32.6	A Final Comment	839

APPENDIX 1 AN INTRODUCTION TO UML 841

APPENDIX 2 OBJECT-ORIENTED CONCEPTS 863

REFERENCES 871

INDEX 889