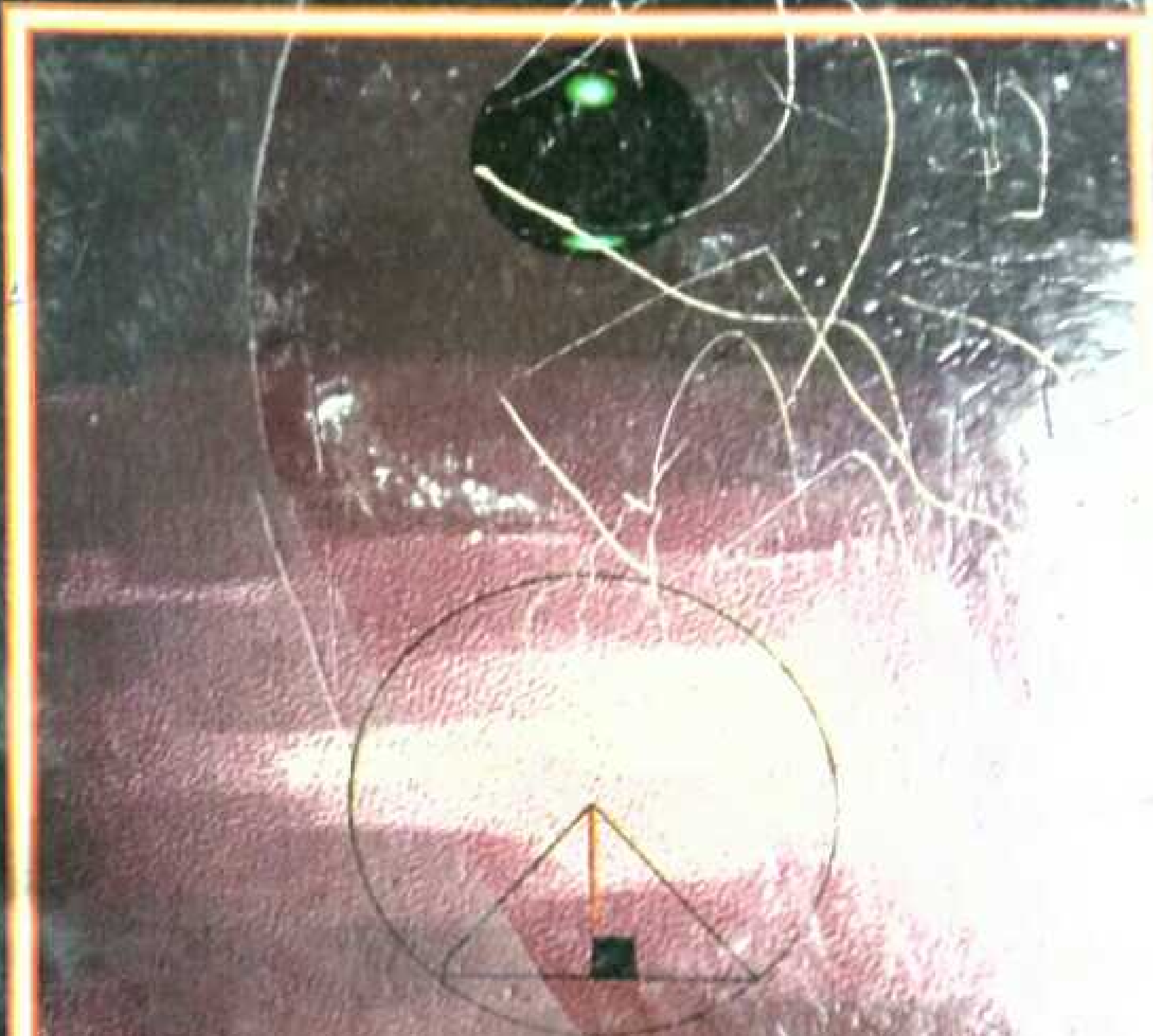
ROGER S. PRESSMAN

SOFTWARE ENGINEERING

A PRACTITIONER'S APPROACH



Roger S. Pressman
20th ANNIVERSARY EDITION
Software Engineering

FIFTH EDITION

Mc GRAW-HILL INTERNATIONAL EDITIONS

Schaum's Outline Series

CONTENTS AT A GLANCE

Preface xxv

PART ONE **The Product and the Process** 1

CHAPTER 1 The Product 3
CHAPTER 2 The Process 19

PART TWO **Managing Software Projects** 53

CHAPTER 3 Project Management Concepts 55
CHAPTER 4 Software Process and Project Metrics 79
CHAPTER 5 Software Project Planning 113
CHAPTER 6 Risk Analysis and Management 145
CHAPTER 7 Project Scheduling and Tracking 165
CHAPTER 8 Software Quality Assurance 193
CHAPTER 9 Software Configuration Management 225

PART THREE **Conventional Methods for Software Engineering** 243

CHAPTER 10 System Engineering 245
CHAPTER 11 Analysis Concepts and Principles 271
CHAPTER 12 Analysis Modeling 299
CHAPTER 13 Design Concepts and Principles 335
CHAPTER 14 Architectural Design 365
CHAPTER 15 User Interface Design 401
CHAPTER 16 Component-Level Design 423
CHAPTER 17 Software Testing Techniques 437
CHAPTER 18 Software Testing Strategies 477
CHAPTER 19 Technical Metrics for Software 507

PART FOUR **Object-Oriented Software Engineering** 539

CHAPTER 20 Object-Oriented Concepts and Principles 541
CHAPTER 21 Object-Oriented Analysis 571
CHAPTER 22 Object-Oriented Design 603

CHAPTER 23 Object-Oriented Testing 631

CHAPTER 24 Technical Metrics for Object-Oriented Systems 653

PART FIVE **Advanced Topics in Software Engineering** 671

CHAPTER 25 Formal Methods 673

CHAPTER 26 Cleanroom Software Engineering 699

CHAPTER 27 Component-Based Software Engineering 721

CHAPTER 28 Client/Server Software Engineering 747

CHAPTER 29 Web Engineering 769

CHAPTER 30 Reengineering 799

CHAPTER 31 Computer-Aided Software Engineering 825

CHAPTER 32 The Road Ahead 845

TABLE OF CONTENTS

PART ONE—THE PRODUCT AND THE PROCESS 1

✓ **CHAPTER 1** ✓ **THE PRODUCT 3**

- 1.1 The Evolving Role of Software 4
- 1.2 Software 6
  - 1.2.1 Software Characteristics 6
  - 1.2.2 Software Applications 9
- 1.3 Software: A Crisis on the Horizon? 11
- 1.4 Software Myths 12
- 1.5 Summary 15

REFERENCES 15

PROBLEMS AND POINTS TO PONDER 16

FURTHER READINGS AND INFORMATION SOURCES 17

✓ **CHAPTER 2** ✓ **THE PROCESS 19** ✓

- (2) 2.1 Software Engineering: A Layered Technology 20
  - 2.1.1 Process, Methods, and Tools 20
  - 2.1.2 A Generic View of Software Engineering 21
- 2.2 The Software Process 23
- 2.3 Software Process Models 26
- 2.4 The Linear Sequential Model 28
- 2.5 The Prototyping Model 30
- 2.6 The RAD Model 32
- (3) 2.7 Evolutionary Software Process Models 34
  - 2.7.1 The Incremental Model 35 ✓
  - 2.7.2 The Spiral Model 36 ✓
  - 2.7.3 The WINWIN Spiral Model 38 ✓
  - 2.7.4 The Concurrent Development Model 40
- 2.8 Component-Based Development 42
- 2.9 The Formal Methods Model 43
- 2.10 Fourth Generation Techniques 44
- 2.11 Process Technology 46
- 2.12 Product and Process 46
- 2.13 Summary 47

REFERENCES 47

PROBLEMS AND POINTS TO PONDER 49

FURTHER READINGS AND INFORMATION SOURCES 50

Process model.

CMM |
Proce
argument
Tech.
proces

Pro Technologi

PART TWO—MANAGING SOFTWARE PROJECTS 53

CHAPTER 3 PROJECT MANAGEMENT CONCEPTS 55

3.1 The Management Spectrum 56
3.1.1 The People 56
3.1.2 The Product 57
3.1.2 The Process 57
3.1.3 The Project 57
3.2 People 58
3.2.1 The Players 58
3.2.2 Team Leaders 59
3.2.3 The Software Team 60
3.2.4 Coordination and Communication Issues 65
3.3 The Product 67
3.3.1 Software Scope 67
3.3.2 Problem Decomposition 67
3.4 The Process 68
3.4.1 Melding the Product and the Process 69
3.4.2 Process Decomposition 70
3.5 The Project 71
3.6 The W^5HH Principle 73
3.7 Critical Practices 74
3.8 Summary 74
REFERENCES 75
PROBLEMS AND POINTS TO PONDER 76
FURTHER READINGS AND INFORMATION SOURCES 77

CHAPTER 4 SOFTWARE PROCESS AND PROJECT METRICS 79

4.1 Measures, Metrics, and Indicators 80
4.2 Metrics in the Process and Project Domains 81
4.2.1 Process Metrics and Software Process Improvement
4.2.2 Project Metrics 86
4.3 Software Measurement 87
4.3.1 Size-Oriented Metrics 88
4.3.2 Function-Oriented Metrics 89
4.3.3 Extended Function Point Metrics 91
4.4 Reconciling Different Metrics Approaches 94
4.5 Metrics for Software Quality 95
4.5.1 An Overview of Factors That Affect Quality 95
4.5.2 Measuring Quality 96
4.5.3 Defect Removal Efficiency 98
4.6 Integrating Metrics Within the Software Engineering Process
4.6.1 Arguments for Software Metrics 99
4.6.2 Establishing a Baseline 100
4.6.3 Metrics Collection, Computation, and Evaluation
4.7 Managing Variation: Statistical Quality Control 100
4.8 Metrics for Small Organizations 104
4.9 Establishing a Software Metrics Program 105
4.10 Summary 107
REFERENCES 107

PROBLEMS AND POINTS TO PONDER 109
FURTHER READINGS AND INFORMATION SOURCES 110

CHAPTER 5 SOFTWARE PROJECT PLANNING 113

5.1 Observations on Estimating 114
5.2 Project Planning Objectives 115
5.3 Software Scope 115
5.3.1 Obtaining Information Necessary for Scope 116
5.3.2 Feasibility 117
5.3.3 A Scoping Example 118
5.4 Resources 120
5.4.1 Human Resources 121
5.4.2 Reusable Software Resources 121
5.4.3 Environmental Resources 122
5.5 Software Project Estimation 123
5.6 Decomposition Techniques 124
5.6.1 Software Sizing 124
5.6.2 Problem-Based Estimation 126
5.6.3 An Example of LOC-Based Estimation 128
5.6.4 An Example of FP-Based Estimation 129
5.6.4 Process-Based Estimation 130
5.6.5 An Example of Process-Based Estimation 131
5.7 Empirical Estimation Models 132
5.7.1 The Structure of Estimation Models 132
5.7.2 The COCOMO Model 133
5.7.3 The Software Equation 135
5.8 The Make/Buy Decision 136
5.8.1 Creating a Decision Tree 137
5.8.2 Outsourcing 138
5.9 Automated Estimation Tools 139
5.10 Summary 140
REFERENCES 140
PROBLEMS AND POINTS TO PONDER 141
FURTHER READINGS AND INFORMATION SOURCES 142

CHAPTER 6 RISK ANALYSIS AND MANAGEMENT 145

6.1 Reactive versus Proactive Risk Strategies 146
6.2 Software Risks 146
6.3 Risk Identification 148
6.3.1 Assessing Overall Project Risk 149
6.3.2 Risk Components and Drivers 149
6.4 Risk Projection 151
6.4.1 Developing a Risk Table 151
6.4.2 Assessing Risk Impact 153
6.4.3 Risk Assessment 154
6.5 Risk Refinement 156
6.6 Risk Mitigation, Monitoring, and Management 156
6.7 Safety Risks and Hazards 158
6.8 The RMMM Plan 159

PROBLEMS AND POINTS TO PONDER 161
FURTHER READINGS AND INFORMATION SOURCES 162

CHAPTER 7 PROJECT SCHEDULING AND TRACKING 165

7.1 Basic Concepts 166
7.1.1 Comments on "Lateness" 167
7.2.1 Basic Principles 168
7.2 The Relationship Between People and Effort 170
7.2.1 An Example 170
7.2.2 An Empirical Relationship 171
7.2.3 Effort Distribution 172
7.3 Defining a Task Set for the Software Project 172
7.3.1 Degree of Rigor 173
7.3.2 Defining Adaptation Criteria 174
7.3.3 Computing a Task Set Selector Value 175
7.3.4 Interpreting the TSS Value and Selecting the Task Se
7.4 Selecting Software Engineering Tasks 177
7.5 Refinement of Major Tasks 178
7.6 Defining a Task Network 180
7.7 Scheduling 181
7.7.1 Timeline Charts 182
7.7.2 Tracking the Schedule 185
7.8 Earned Value Analysis 186
7.9 Error Tracking 187
7.10 The Project Plan 189
7.11 Summary 189
REFERENCES 189
PROBLEMS AND POINTS TO PONDER 190
FURTHER READINGS AND INFORMATION SOURCES 192

CHAPTER 8 SOFTWARE QUALITY ASSURANCE 193

8.1 Quality Concepts 194
8.1.1 Quality 195
8.1.2 Quality Control 196
8.1.3 Quality Assurance 196
8.1.4 Cost of Quality 196
8.2 The Quality Movement 198
8.3 Software Quality Assurance 199
8.3.1 Background Issues 200
8.3.2 SQA Activities 201
8.4 Software Reviews 202
8.4.1 Cost Impact of Software Defects 203
8.4.2 Defect Amplification and Removal 204
8.5 Formal Technical Reviews 205
8.5.1 The Review Meeting 206
8.5.2 Review Reporting and Record Keeping 207
8.5.3 Review Guidelines 207
8.6 Formal Approaches to SQA 209
8.7 Statistical Software Quality Assurance 209
8.8 Software Reliability 212
8.8.1 Measures of Reliability and Availability 212
8.8.2 Software Safety 213

8.9 Mistake-Proofing for Software 214
8.10 The ISO 9000 Quality Standards 216
8.10.1 The ISO Approach to Quality Assurance Systems 217
8.10.2 The ISO 9001 Standard 217
8.11 The SQA Plan 218
8.12 Summary 219
REFERENCES 220
PROBLEMS AND POINTS TO PONDER 221
FURTHER READINGS AND INFORMATION SOURCES 222

CHAPTER 9 SOFTWARE CONFIGURATION MANAGEMENT 225

9.1 Software Configuration Management 226
9.1.1 Baselines 227
9.1.2 Software Configuration Items 228
9.2 The SCM Process 230
9.3 Identification of Objects in the Software Configuration 230
9.4 Version Control 232
9.5 Change Control 234
9.6 Configuration Audit 237
9.7 Status Reporting 237
9.8 SCM Standards 238
9.9 Summary 238
REFERENCES 239
PROBLEMS AND POINTS TO PONDER 239
FURTHER READINGS AND INFORMATION SOURCES 240

E—CONVENTIONAL METHODS FOR SOFTWARE ENGINEERING 243

CHAPTER 10 SYSTEM ENGINEERING 245

10.1 Computer-Based Systems 246
10.2 The System Engineering Hierarchy 248
10.2.1 System Modeling 249
10.2.2 System Simulation 251
10.3 Business Process Engineering: An Overview 251
10.4 Product Engineering: An Overview 254
10.5 Requirements Engineering 256
10.5.1 Requirements Elicitation 256
10.5.2 Requirements Analysis and Negotiation 258
10.5.3 Requirements Specification 259
10.5.4 System Modeling 259
10.5.5 Requirements Validation 260
10.5.6 Requirements Management 261
10.6 System Modeling 262
10.7 Summary 265
REFERENCES 267
PROBLEMS AND POINTS TO PONDER 267
FURTHER READINGS AND INFORMATION SOURCES 268

CHAPTER 11 ANALYSIS CONCEPTS AND PRINCIPLES 271

11.1 Requirements Analysis 272
11.2 Requirements Elicitation for Software 274
11.2.1 Initiating the Process 274
11.2.2 Facilitated Application Specification Techniques 275
11.2.3 Quality Function Deployment 279
11.2.4 Use-Cases 280
11.3 Analysis Principles 282
11.3.1 The Information Domain 283
11.3.2 Modeling 285
11.3.3 Partitioning 286
11.3.4 Essential and Implementation Views 288
11.4 Software Prototyping 289
11.4.1 Selecting the Prototyping Approach 289
11.4.2 Prototyping Methods and Tools 290
11.5 Specification 291
11.5.1 Specification Principles 291
11.5.2 Representation 292
11.5.3 The Software Requirements Specification 293
11.6 Specification Review 294
11.7 Summary 294
REFERENCES 295
PROBLEMS AND POINTS TO PONDER 296
FURTHER READINGS AND INFORMATION SOURCES 297

CHAPTER 12 ANALYSIS MODELING 299

12.1 A Brief History 300
12.2 The Elements of the Analysis Model 301
12.3 Data Modeling 302
12.3.1 Data Objects, Attributes, and Relationships 302
12.3.2 Cardinality and Modality 305
12.3.3 Entity/Relationship Diagrams 307
12.4 Functional Modeling and Information Flow 309
12.4.1 Data Flow Diagrams 311
12.4.2 Extensions for Real-Time Systems 312
12.4.3 Ward and Mellor Extensions 312
12.4.4 Hatley and Pirbhai Extensions 315
12.5 Behavioral Modeling 317
12.6 The Mechanics of Structured Analysis 319
12.6.1 Creating an Entity/Relationship Diagram 319
12.6.2 Creating a Data Flow Model 321
12.6.3 Creating a Control Flow Model 324
12.6.4 The Control Specification 325
12.6.5 The Process Specification 327
12.7 The Data Dictionary 328
12.8 Other Classical Analysis Methods 330
12.9 Summary 331
REFERENCES 331
PROBLEMS AND POINTS TO PONDER 332
FURTHER READINGS AND INFORMATION SOURCES

CHAPTER 13 DESIGN CONCEPTS AND PRINCIPLES 335

13.1 Software Design and Software Engineering 336
13.2 The Design Process 338
13.2.1 Design and Software Quality 338
13.2.2 The Evolution of Software Design 339
13.3 Design Principles 340
13.4 Design Concepts 341
13.4.1 Abstraction 342
13.4.2 Refinement 343
13.4.3 Modularity 343
13.4.4 Software Architecture 346
13.4.5 Control Hierarchy 347
13.4.6 Structural Partitioning 348
13.4.7 Data Structure 349
13.4.8 Software Procedure 351
13.4.9 Information Hiding 351
13.5 Effective Modular Design 352
13.5.1 Functional Independence 352
13.5.2 Cohesion 353
13.5.3 Coupling 354
13.6 Design Heuristics for Effective Modularity 355
13.7 The Design Model 357
13.8 Design Documentation 358
13.9 Summary 359
REFERENCES 359
PROBLEMS AND POINTS TO PONDER 361
FURTHER READINGS AND INFORMATION SOURCES 362

CHAPTER 14 ARCHITECTURAL DESIGN 365

14.1 Software Architecture 366
14.1.1 What Is Architecture? 366
14.1.2 Why Is Architecture Important? 367
14.2 Data Design 368
14.2.1 Data Modeling, Data Structures, Databases, and the Data Warehouse 368
14.2.2 Data Design at the Component Level 369
14.3 Architectural Styles 371
14.3.1 A Brief Taxonomy of Styles and Patterns 371
14.3.2 Organization and Refinement 374
14.4 Analyzing Alternative Architectural Designs 375
14.4.1 An Architecture Trade-off Analysis Method 375
14.4.2 Quantitative Guidance for Architectural Design 376
14.4.3 Architectural Complexity 378
14.5 Mapping Requirements into a Software Architecture 378
14.5.1 Transform Flow 379
14.5.2 Transaction Flow 380
14.6 Transform Mapping 380
14.6.1 An Example 380
14.6.2 Design Steps 381
14.7 Transaction Mapping 389
14.7.1 An Example 390

14.8 Refining the Architectural Design 394
14.9 Summary 395
REFERENCES 396
PROBLEMS AND POINTS TO PONDER 397
FURTHER READINGS AND INFORMATION SOURCES 397

CHAPTER 15 USER INTERFACE DESIGN 401

15.1 The Golden Rules 402
15.1.1 Place the User in Control 402
15.1.2 Reduce the User's Memory Load 404
15.1.3 Make the Interface Consistent 404
15.2 User Interface Design 405
15.2.1 Interface Design Models 405
15.2.2 The User Interface Design Process 407
15.3 Task Analysis and Modeling 408
15.4 Interface Design Activities 410
15.4.1 Defining Interface Objects and Actions 410
15.4.2 Design Issues 413
15.5 Implementation Tools 415
15.6 Design Evaluation 416
15.7 Summary 418
REFERENCES 418
PROBLEMS AND POINTS TO PONDER 419
FURTHER READINGS AND INFORMATION SOURCES 420

CHAPTER 16 COMPONENT-LEVEL DESIGN 423

16.1 Structured Programming 424
16.1.1 Graphical Design Notation 425
16.1.2 Tabular Design Notation 427
16.1.3 Program Design Language 429
16.1.4 A PDL Example 430
16.2 Comparison of Design Notation 432
16.3 Summary 433
REFERENCES 433
PROBLEMS AND POINTS TO PONDER 434
FURTHER READINGS AND INFORMATION SOURCES 435

CHAPTER 17 SOFTWARE TESTING TECHNIQUES 437

17.1 Software Testing Fundamentals 438
17.1.1 Testing Objectives 439
17.1.2 Testing Principles 439
17.1.3 Testability 440
17.2 Test Case Design 443
17.3 White-Box Testing 444
17.4 Basis Path Testing 445
17.4.1 Flow Graph Notation 445
17.4.2 Cyclomatic Complexity 446
17.4.3 Deriving Test Cases 449
17.4.4 Graph Matrices 452
17.5 Control Structure Testing 454
17.5.1 Condition Testing 454

CHAPTER 19 TECHNICAL METRICS FOR SOFTWARE 507

19.1 Software Quality 508
19.1.1 McCall's Quality Factors 509
19.1.2 FURPS 511
19.1.3 ISO 9126 Quality Factors 513
19.1.4 The Transition to a Quantitative View 513
19.2 A Framework for Technical Software Metrics 514
19.2.1 The Challenge of Technical Metrics 514
19.2.2 Measurement Principles 515
19.2.3 The Attributes of Effective Software Metrics 516
19.3 Metrics for the Analysis Model 517
19.3.1 Function-Based Metrics 518
19.3.2 The Bang Metric 520
19.3.3 Metrics for Specification Quality 522
19.4 Metrics for the Design Model 523
19.4.1 Architectural Design Metrics 523
19.4.2 Component-Level Design Metrics 526
19.4.3 Interface Design Metrics 530
19.5 Metrics for Source Code 531
19.6 Metrics for Testing 532
19.7 Metrics for Maintenance 533
19.8 Summary 534
REFERENCES 534
PROBLEMS AND POINTS TO PONDER 536
FURTHER READING AND OTHER INFORMATION SOURCES 537

PART FOUR—OBJECT-ORIENTED SOFTWARE ENGINEERING 539

✓ CHAPTER 20 OBJECT-ORIENTED CONCEPTS AND PRINCIPLES 541

20.1 The Object-Oriented Paradigm 542
20.2 Object-Oriented Concepts 544
20.2.1 Classes and Objects 546
20.2.2 Attributes 547
20.2.3 Operations, Methods, and Services 548
20.2.4 Messages 548
20.2.5 Encapsulation, Inheritance, and Polymorphism 550
20.3 Identifying the Elements of an Object Model 553
20.3.1 Identifying Classes and Objects 553
20.3.2 Specifying Attributes 557
20.3.3 Defining Operations 558
20.3.4 Finalizing the Object Definition 559
20.4 Management of Object-Oriented Software Projects 560
20.4.1 The Common Process Framework for OO 560
20.4.2 OO Project Metrics and Estimation 562
20.4.3 An OO Estimating and Scheduling Approach 564
20.4.4 Tracking Progress for an OO Project 565
20.5 Summary 566
REFERENCES 566
PROBLEMS AND POINTS TO PONDER 567
FURTHER READINGS AND INFORMATION SOURCES 568

✓ 21.1 Object-Oriented Analysis 572
21.1.1 Conventional vs. OO Approaches 572
21.1.2 The OOA Landscape 573
21.1.3 A Unified Approach to OOA 575
21.2 Domain Analysis 576
21.2.1 Reuse and Domain Analysis 577
21.2.2 The Domain Analysis Process 577
21.3 Generic Components of the OO Analysis Model 579
21.4 The OOA Process 581
21.4.1 Use-Cases 581
21.4.2 Class-Responsibility-Collaborator Modeling 582
21.4.3 Defining Structures and Hierarchies 588
21.4.4 Defining Subjects and Subsystems 590
21.5 The Object-Relationship Model 591
21.6 The Object-Behavior Model 594
21.6.1 Event Identification with Use-Cases 594
21.6.2 State Representations 595
21.7 Summary 598
REFERENCES 599
PROBLEMS AND POINTS TO PONDER 600
FURTHER READINGS AND INFORMATION SOURCES 601

✓ CHAPTER 22 OBJECT-ORIENTED DESIGN 603

22.1 Design for Object-Oriented Systems 604
22.1.1 Conventional vs. OO Approaches 605
22.1.2 Design Issues 607
22.1.3 The OOD Landscape 608
22.1.4 A Unified Approach to OOD 610
22.2 The System Design Process 611
22.2.1 Partitioning the Analysis Model 612
22.2.2 Concurrency and Subsystem Allocation 613
22.2.3 The Task Management Component 614
22.2.4 The User Interface Component 615
22.2.5 The Data Management Component 615
22.2.6 The Resource Management Component 616
22.2.7 Intersubsystem Communication 616
22.3 The Object Design Process 618
22.3.1 Object Descriptions 618
22.3.2 Designing Algorithms and Data Structures 619
22.3.3 Program Components and Interfaces 621
22.4 Design Patterns 624
22.4.1 Describing a Design Pattern 624
22.4.2 Using Patterns in Design 625
22.5 Object-Oriented Programming 625
22.6 Summary 626
REFERENCES 627
PROBLEMS AND POINTS TO PONDER 628
FURTHER READINGS AND INFORMATION SOURCES 629

CHAPTER 23 OBJECT-ORIENTED TESTING 631

✓

23.1 Broadening the View of Testing 632
23.2 Testing OOA and OOD Models 633
 23.2.1 Correctness of OOA and OOD Models 633
 23.2.2 Consistency of OOA and OOD Models 634
23.3 Object-Oriented Testing Strategies 636
 23.3.1 Unit Testing in the OO Context 636
 23.3.2 Integration Testing in the OO Context 637
 23.3.3 Validation Testing in an OO Context 637
23.4 Test Case Design for OO Software 637
 23.4.1 The Test Case Design Implications of OO Concepts
 23.4.2 Applicability of Conventional Test Case Design Methods 638
 23.4.3 Fault-Based Testing 639
 23.4.4 The Impact of OO Programming on Testing 640
 23.4.5 Test Cases and the Class Hierarchy 641
 23.4.6 Scenario-Based Test Design 641
 23.4.7 Testing Surface Structure and Deep Structure 643
23.5 Testing Methods Applicable at the Class Level 644
 23.5.1 Random Testing for OO Classes 644
 23.5.2 Partition Testing at the Class Level 644
23.6 Interclass Test Case Design 645
 23.6.1 Multiple Class Testing 645
 23.6.2 Tests Derived from Behavior Models 647
23.7 Summary 648
REFERENCES 649
PROBLEMS AND POINTS TO PONDER 649
FURTHER READINGS AND INFORMATION SOURCES 650

CHAPTER 24 TECHNICAL METRICS FOR OBJECT-ORIENTED SYSTEMS 653

✓

24.1 The Intent of Object-Oriented Metrics 654
24.2 The Distinguishing Characteristics of Object-Oriented Metrics 654
 24.2.1 Localization 655
 24.2.2 Encapsulation 655
 24.2.3 Information Hiding 655
 24.2.4 Inheritance 656
 24.2.5 Abstraction 656
24.3 Metrics for the OO Design Model 656
24.4 Class-Oriented Metrics 658
 24.4.1 The CK Metrics Suite 658
 24.4.2 Metrics Proposed by Lorenz and Kidd 661
 24.4.3 The MOOD Metrics Suite 662
24.5 Operation-Oriented Metrics 664
24.6 Metrics for Object-Oriented Testing 664
24.7 Metrics for Object-Oriented Projects 665
24.8 Summary 666
REFERENCES 667
PROBLEMS AND POINTS TO PONDER 668
FURTHER READINGS AND INFORMATION SOURCES 669

CHAPTER 23 OBJECT-ORIENTED TESTING 631

✓

23.1 Broadening the View of Testing 632
23.2 Testing OOA and OOD Models 633
23.2.1 Correctness of OOA and OOD Models 633
23.2.2 Consistency of OOA and OOD Models 634
23.3 Object-Oriented Testing Strategies 636
23.3.1 Unit Testing in the OO Context 636
23.3.2 Integration Testing in the OO Context 637
23.3.3 Validation Testing in an OO Context 637
23.4 Test Case Design for OO Software 637
23.4.1 The Test Case Design Implications of OO Concepts
23.4.2 Applicability of Conventional Test Case Design Methods 638
23.4.3 Fault-Based Testing 639
23.4.4 The Impact of OO Programming on Testing 640
23.4.5 Test Cases and the Class Hierarchy 641
23.4.6 Scenario-Based Test Design 641
23.4.7 Testing Surface Structure and Deep Structure 643
23.5 Testing Methods Applicable at the Class Level 644
23.5.1 Random Testing for OO Classes 644
23.5.2 Partition Testing at the Class Level 644
23.6 Interclass Test Case Design 645
23.6.1 Multiple Class Testing 645
23.6.2 Tests Derived from Behavior Models 647
23.7 Summary 648
REFERENCES 649
PROBLEMS AND POINTS TO PONDER 649
FURTHER READINGS AND INFORMATION SOURCES 650

CHAPTER 24 TECHNICAL METRICS FOR OBJECT-ORIENTED SYSTEMS 653

✓

24.1 The Intent of Object-Oriented Metrics 654
24.2 The Distinguishing Characteristics of Object-Oriented Metrics
24.2.1 Localization 655
24.2.2 Encapsulation 655
24.2.3 Information Hiding 655
24.2.4 Inheritance 656
24.2.5 Abstraction 656
24.3 Metrics for the OO Design Model 656
24.4 Class-Oriented Metrics 658
24.4.1 The CK Metrics Suite 658
24.4.2 Metrics Proposed by Lorenz and Kidd 661
24.4.3 The MOOD Metrics Suite 662
24.5 Operation-Oriented Metrics 664
24.6 Metrics for Object-Oriented Testing 664
24.7 Metrics for Object-Oriented Projects 665
24.8 Summary 666
REFERENCES 667
PROBLEMS AND POINTS TO PONDER 668
FURTHER READINGS AND INFORMATION SOURCES 669

CHAPTER 25 FORMAL METHODS 673

25.1 Basic Concepts 674
25.1.1 Deficiencies of Less Formal Approaches 675
25.1.2 Mathematics in Software Development 676
25.1.3 Formal Methods Concepts 677
25.2 Mathematical Preliminaries 682
25.2.1 Sets and Constructive Specification 683
25.2.2 Set Operators 684
25.2.3 Logic Operators 686
25.2.4 Sequences 686
25.3 Applying Mathematical Notation for Formal Specification 687
25.4 Formal Specification Languages 689
25.5 Using Z to Represent an Example Software Component 690
25.6 The Ten Commandments of Formal Methods 693
25.7 Formal Methods—The Road Ahead 694
25.8 Summary 695
REFERENCES 695
PROBLEMS AND POINTS TO PONDER 696
FURTHER READINGS AND INFORMATION SOURCES 697

CHAPTER 26 CLEANROOM SOFTWARE ENGINEERING 699

26.1 The Cleanroom Approach 700
26.1.1 The Cleanroom Strategy 701
26.1.2 What Makes Cleanroom Different? 703
26.2 Functional Specification 703
26.2.1 Black-Box Specification 705
26.2.2 State-Box Specification 705
26.2.3 Clear-Box Specification 706
26.3 Cleanroom Design 706
26.3.1 Design Refinement and Verification 707
26.3.2 Advantages of Design Verification 710
26.4 Cleanroom Testing 712
26.4.1 Statistical Use Testing 712
26.4.2 Certification 714
26.5 Summary 714
REFERENCES 715
PROBLEMS AND POINTS TO PONDER 716
FURTHER READINGS AND INFORMATION SOURCES 717

CHAPTER 27 COMPONENT-BASED SOFTWARE ENGINEERING 721

27.1 Engineering of Component-Based Systems 722
27.2 The CBSE Process 724
27.3 Domain Engineering 725
27.3.1 The Domain Analysis Process 726
27.3.2 Characterization Functions 727
27.3.3 Structural Modeling and Structure Points 728
27.4 Component-Based Development 730
27.4.1 Component Qualification, Adaptation, and Composition 730

27.4.2 Component Engineering 734
27.4.3 Analysis and Design for Reuse 734
27.5 Classifying and Retrieving Components 735
27.5.1 Describing Reusable Components 736
27.5.2 The Reuse Environment 738
27.6 Economics of CBSE 739
27.6.1 Impact on Quality, Productivity, and Cost 73
27.6.2 Cost Analysis Using Structure Points 741
27.6.3 Reuse Metrics 741
27.7 Summary 742
REFERENCES 743
PROBLEMS AND POINTS TO PONDER 744
FURTHER READINGS AND INFORMATION SOURCES 745

CHAPTER 28 CLIENT/SERVER SOFTWARE ENGINEERING 747

28.1 The Structure of Client/Server Systems 748
28.1.1 Software Components for c/s Systems 750
28.1.2 The Distribution of Software Components 750
28.1.3 Guidelines for Distributing Application Subsystems
28.1.4 Linking c/s Software Subsystems 753
28.1.5 Middleware and Object Request Broker Architectu
28.2 Software Engineering for c/s Systems 755
28.3 Analysis Modeling Issues 755
28.4 Design for c/s Systems 755
28.4.1 Architectural Design for Client/Server Systems 7
28.4.2 Conventional Design Approaches for Application Software 757
28.4.3 Database Design 758
28.4.4 An Overview of a Design Approach 759
28.4.5 Process Design Iteration 761
28.5 Testing Issues 761
28.5.1 Overall c/s Testing Strategy 762
28.5.2 c/s Testing Tactics 763
28.6 Summary 764
REFERENCES 764
PROBLEMS AND POINTS TO PONDER 765
FURTHER READINGS AND INFORMATION SOURCES 766

CHAPTER 29 WEB ENGINEERING 769

29.1 The Attributes of Web-Based Applications 771
29.1.1 Quality Attributes 773
29.1.2 The Technologies 773
29.2 The WebE Process 774
29.3 A Framework for WebE 775
29.4 Formulating/Analyzing Web-Based Systems 776
29.4.1 Formulation 776
29.4.2 Analysis 778
29.5 Design for Web-Based Applications 779
29.5.1 Architectural Design 780
29.5.2 Navigation Design 783
29.5.3 Interface Design 785

29.6 Testing Web-Based Applications 786
29.7 Management Issues 787
29.7.1 The WebE Team 788
29.7.2 Project Management 789
29.7.3 SCM Issues for WebE 792
29.8 Summary 794
REFERENCES 795
PROBLEMS AND POINTS TO PONDER 796
FURTHER READINGS AND INFORMATION SOURCES 797

CHAPTER 30 REENGINEERING 799

30.1 Business Process Reengineering 800
30.1.1 Business Processes 800
30.1.2 Principles of Business Process Reengineering 801
30.1.3 A BPR Model 802
30.1.4 Words of Warning 804
30.2 Software Reengineering 804
30.2.1 Software Maintenance 804
30.2.2 A Software Reengineering Process Model 805
30.3 Reverse Engineering 809
30.3.1 Reverse Engineering to Understand Processing 810
30.3.2 Reverse Engineering to Understand Data 811
30.3.3 Reverse Engineering User Interfaces 812
30.4 Restructuring 813
30.4.1 Code Restructuring 814
30.4.2 Data Restructuring 814
30.5 Forward Engineering 814
30.5.1 Forward Engineering for Client/Server Architectures 816
30.5.2 Forward Engineering for Object-Oriented Architectures 817
30.5.3 Forward Engineering User Interfaces 818
30.6 The Economics of Reengineering 819
30.7 Summary 820
REFERENCES 820
PROBLEMS AND POINTS TO PONDER 822
FURTHER READINGS AND INFORMATION SOURCES 823

CHAPTER 31 COMPUTER-AIDED SOFTWARE ENGINEERING 825

31.1 What is CASE? 826
31.2 Building Blocks for CASE 826
31.3 A Taxonomy of CASE Tools 828
31.4 Integrated CASE Environments 833
31.5 The Integration Architecture 834
31.6 The CASE Repository 836
31.6.1 The Role of the Repository in I-CASE 836
31.6.2 Features and Content 837
31.7 Summary 841
REFERENCES 842
PROBLEMS AND POINTS TO PONDER 842
FURTHER READINGS AND INFORMATION SOURCES 843

CHAPTER 32 THE ROAD AHEAD 845

32.1 The Importance of Software—Revisited 846
32.2 The Scope of Change 847
32.3 People and the Way They Build Systems 847
32.4 The "New" Software Engineering Process 848
32.5 New Modes for Representing Information 849
32.6 Technology as a Driver 851
32.7 A Concluding Comment 852
REFERENCES 853
PROBLEMS AND POINTS TO PONDER 853
FURTHER READINGS AND INFORMATION SOURCES 853

PREFACE

When a 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.

In the 20 years since the first edition of this book was written, 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 applications.

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 the most advanced technologies of the day. 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 fifth edition of *Software Engineering: A Practitioner's Approach* is intended to serve as a guide to a maturing engineering discipline. The fifth edition, like the four 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 format and style of the fifth edition have undergone significant change, making the presentation more reader-friendly and the content more easily accessible.

The fifth edition is considerably more than a simple update. The book has been revised to accommodate the dramatic growth in the field and to emphasize new and important software engineering practices. In addition, a comprehensive Web site has been developed to complement the content of the book. The Web site, which I call