

Software Engineering

A PRACTITIONER'S APPROACH

NINTH EDITION

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



TABLE OF CONTENTS

Preface xvii

CHAPTER 1 SOFTWARE AND SOFTWARE ENGINEERING 1

- 1.1 The Nature of Software 4
 - 1.1.1 Defining Software 5
 - 1.1.2 Software Application Domains 7
 - 1.1.3 Legacy Software 8
- 1.2 Defining the Discipline 8
- 1.3 The Software Process 9
 - 1.3.1 The Process Framework 10
 - 1.3.2 Umbrella Activities 11
 - 1.3.3 Process Adaptation 11
- 1.4 Software Engineering Practice 12
 - 1.4.1 The Essence of Practice 12
 - 1.4.2 General Principles 14
- 1.5 How It All Starts 15
- 1.6 Summary 17

PART ONE THE SOFTWARE PROCESS 19

CHAPTER 2 PROCESS MODELS 20

- 2.1 A Generic Process Model 21
- 2.2 Defining a Framework Activity 23
- 2.3 Identifying a Task Set 23
- 2.4 Process Assessment and Improvement 24
- 2.5 Prescriptive Process Models 25
 - 2.5.1 The Waterfall Model 25
 - 2.5.2 Prototyping Process Model 26
 - 2.5.3 Evolutionary Process Model 29
 - 2.5.4 Unified Process Model 31
- 2.6 Product and Process 33
- 2.7 Summary 35

CHAPTER 3 AGILITY AND PROCESS 37

3.1 What Is Agility? 38

3.2 Agility and the Cost of Change 39

3.3 What Is an Agile Process? 40

 3.3.1 Agility Principles 40

 3.3.2 The Politics of Agile Development 41

3.4 Scrum 42

 3.4.1 Scrum Teams and Artifacts 43

 3.4.2 Sprint Planning Meeting 44

 3.4.3 Daily Scrum Meeting 44

 3.4.4 Sprint Review Meeting 45

 3.4.5 Sprint Retrospective 45

3.5 Other Agile Frameworks 46

 3.5.1 The XP Framework 46

 3.5.2 Kanban 48

 3.5.3 DevOps 50

3.6 Summary 51

CHAPTER 4 RECOMMENDED PROCESS MODEL 54

4.1 Requirements Definition 57

4.2 Preliminary Architectural Design 59

4.3 Resource Estimation 60

4.4 First Prototype Construction 61

4.5 Prototype Evaluation 64

4.6 Go, No-Go Decision 65

4.7 Prototype Evolution 67

 4.7.1 New Prototype Scope 67

 4.7.2 Constructing New Prototypes 68

 4.7.3 Testing New Prototypes 68

4.8 Prototype Release 68

4.9 Maintain Release Software 69

4.10 Summary 72

CHAPTER 5 HUMAN ASPECTS OF SOFTWARE ENGINEERING 74

5.1 Characteristics of a Software Engineer 75

5.2 The Psychology of Software Engineering 75

5.3	The Software Team	76
5.4	Team Structures	78
5.5	The Impact of Social Media	79
5.6	Global Teams	80
5.7	Summary	81

PART TWO MODELING 83

**CHAPTER 6 PRINCIPLES THAT
GUIDE PRACTICE 84**

6.1	Core Principles	85
6.1.1	Principles That Guide Process	85
6.1.2	Principles That Guide Practice	86
6.2	Principles That Guide Each Framework Activity	88
6.2.1	Communication Principles	88
6.2.2	Planning Principles	91
6.2.3	Modeling Principles	92
6.2.4	Construction Principles	95
6.2.5	Deployment Principles	98
6.3	Summary	100

CHAPTER 7 UNDERSTANDING REQUIREMENTS 102

7.1	Requirements Engineering	103
7.1.1	Inception	104
7.1.2	Elicitation	104
7.1.3	Elaboration	104
7.1.4	Negotiation	105
7.1.5	Specification	105
7.1.6	Validation	105
7.1.7	Requirements Management	106
7.2	Establishing the Groundwork	107
7.2.1	Identifying Stakeholders	107
7.2.2	Recognizing Multiple Viewpoints	107
7.2.3	Working Toward Collaboration	108
7.2.4	Asking the First Questions	108
7.2.5	Nonfunctional Requirements	109
7.2.6	Traceability	109

10.4	Architectural Considerations	193
10.5	Architectural Decisions	195
10.6	Architectural Design	196
10.6.1	Representing the System in Context	196
10.6.2	Defining Archetypes	197
10.6.3	Refining the Architecture into Components	198
10.6.4	Describing Instantiations of the System	200
10.7	Assessing Alternative Architectural Designs	201
10.7.1	Architectural Reviews	202
10.7.2	Pattern-Based Architecture Review	203
10.7.3	Architecture Conformance Checking	204
10.8	Summary	204

CHAPTER 11 COMPONENT-LEVEL DESIGN 206

11.1	What Is a Component?	207
11.1.1	An Object-Oriented View	207
11.1.2	The Traditional View	209
11.1.3	A Process-Related View	211
11.2	Designing Class-Based Components	212
11.2.1	Basic Design Principles	212
11.2.2	Component-Level Design Guidelines	215
11.2.3	Cohesion	216
11.2.4	Coupling	218
11.3	Conducting Component-Level Design	219
11.4	Specialized Component-Level Design	225
11.4.1	Component-Level Design for WebApps	226
11.4.2	Component-Level Design for Mobile Apps	226
11.4.3	Designing Traditional Components	227
11.4.4	Component-Based Development	228
11.5	Component Refactoring	230
11.6	Summary	231

CHAPTER 12 USER EXPERIENCE DESIGN 233

12.1	User Experience Design Elements	234
12.1.1	Information Architecture	235
12.1.2	User Interaction Design	236
12.1.3	Usability Engineering	236
12.1.4	Visual Design	237

12.2	The Golden Rules	238
12.2.1	Place the User in Control	238
12.2.2	Reduce the User's Memory Load	239
12.2.3	Make the Interface Consistent	240
12.3	User Interface Analysis and Design	241
12.3.1	Interface Analysis and Design Models	241
12.3.2	The Process	242
12.4	User Experience Analysis	243
12.4.1	User Research	244
12.4.2	User Modeling	245
12.4.3	Task Analysis	247
12.4.4	Work Environment Analysis	248
12.5	User Experience Design	249
12.6	User Interface Design	250
12.6.1	Applying Interface Design Steps	251
12.6.2	User Interface Design Patterns	252
12.7	Design Evaluation	253
12.7.1	Prototype Review	253
12.7.2	User Testing	255
12.8	Usability and Accessibility	255
12.8.1	Usability Guidelines	257
12.8.2	Accessibility Guidelines	259
12.9	Conventional Software UX and Mobility	261
12.10	Summary	261

CHAPTER 13 DESIGN FOR MOBILITY 264

13.1	The Challenges	265
13.1.1	Development Considerations	265
13.1.2	Technical Considerations	266
13.2	Mobile Development Life Cycle	268
13.2.1	User Interface Design	270
13.2.2	Lessons Learned	271
13.3	Mobile Architectures	273
13.4	Context-Aware Apps	274
13.5	Web Design Pyramid	275
13.5.1	WebApp Interface Design	275
13.5.2	Aesthetic Design	277
13.5.3	Content Design	277
13.5.4	Architecture Design	278
13.5.5	Navigation Design	280

17.6.1	A Generic Example	347
17.6.2	Six Sigma for Software Engineering	349
17.7	Software Reliability	350
17.7.1	Measures of Reliability and Availability	350
17.7.2	Use of AI to Model Reliability	351
17.7.3	Software Safety	352
17.8	The ISO 9000 Quality Standards	353
17.9	The SQA Plan	354
17.10	Summary	355

CHAPTER 18 SOFTWARE SECURITY ENGINEERING 356

18.1	Why Software Security Information Is Important	357
18.2	Security Life-Cycle Models	357
18.3	Secure Development Life-Cycle Activities	359
18.4	Security Requirements Engineering	360
18.4.1	SQUARE	360
18.4.2	The SQUARE Process	360
18.5	Misuse and Abuse Cases and Attack Patterns	363
18.6	Security Risk Analysis	364
18.7	Threat Modeling, Prioritization, and Mitigation	365
18.8	Attack Surface	366
18.9	Secure Coding	367
18.10	Measurement	368
18.11	Security Process Improvement and Maturity Models	370
18.12	Summary	370

CHAPTER 19 SOFTWARE TESTING—COMPONENT LEVEL 372

19.1	A Strategic Approach to Software Testing	373
19.1.1	Verification and Validation	373
19.1.2	Organizing for Software Testing	374
19.1.3	The Big Picture	375
19.1.4	Criteria for “Done”	377

19.2	Planning and Recordkeeping	378
19.2.1	Role of Scaffolding	379
19.2.2	Cost-Effective Testing	379
19.3	Test-Case Design	381
19.3.1	Requirements and Use Cases	382
19.3.2	Traceability	383
19.4	White-Box Testing	383
19.4.1	Basis Path Testing	384
19.4.2	Control Structure Testing	386
19.5	Black-Box Testing	388
19.5.1	Interface Testing	388
19.5.2	Equivalence Partitioning	389
19.5.3	Boundary Value Analysis	389
19.6	Object-Oriented Testing	390
19.6.1	Class Testing	390
19.6.2	Behavioral Testing	392
19.7	Summary	393

CHAPTER 20 SOFTWARE TESTING— INTEGRATION LEVEL 395

20.1	Software Testing Fundamentals	396
20.1.1	Black-Box Testing	397
20.1.2	White-Box Testing	397
20.2	Integration Testing	398
20.2.1	Top-Down Integration	398
20.2.2	Bottom-Up Integration	399
20.2.3	Continuous Integration	400
20.2.4	Integration Test Work Products	402
20.3	Artificial Intelligence and Regression Testing	402
20.4	Integration Testing in the OO Context	404
20.4.1	Fault-Based Test-Case Design	405
20.4.2	Scenario-Based Test-Case Design	406
20.5	Validation Testing	407
20.6	Testing Patterns	409
20.7	Summary	409

**CHAPTER 21 SOFTWARE TESTING—SPECIALIZED
TESTING FOR MOBILITY 412**

- 21.1 Mobile Testing Guidelines 413
- 21.2 The Testing Strategies 414
- 21.3 User Experience Testing Issues 415
 - 21.3.1 Gesture Testing 415
 - 21.3.2 Virtual Keyboard Input 416
 - 21.3.3 Voice Input and Recognition 416
 - 21.3.4 Alerts and Extraordinary Conditions 417
- 21.4 Web Application Testing 418
- 21.5 Web Testing Strategies 418
 - 21.5.1 Content Testing 420
 - 21.5.2 Interface Testing 421
 - 21.5.3 Navigation Testing 421
- 21.6 Internationalization 423
- 21.7 Security Testing 423
- 21.8 Performance Testing 424
- 21.9 Real-Time Testing 426
- 21.10 Testing AI Systems 428
 - 21.10.1 Static and Dynamic Testing 429
 - 21.10.2 Model-Based Testing 429
- 21.11 Testing Virtual Environments 430
 - 21.11.1 Usability Testing 430
 - 21.11.2 Accessibility Testing 433
 - 21.11.3 Playability Testing 433
- 21.12 Testing Documentation and Help Facilities 434
- 21.13 Summary 435

**CHAPTER 22 SOFTWARE CONFIGURATION
MANAGEMENT 437**

- 22.1 Software Configuration Management 438
 - 22.1.1 An SCM Scenario 439
 - 22.1.2 Elements of a Configuration Management System 440
 - 22.1.3 Baselines 441

22.1.4	Software Configuration Items	441
22.1.5	Management of Dependencies and Changes	442
22.2	The SCM Repository	443
22.2.1	General Features and Content	444
22.2.2	SCM Features	444
22.3	Version Control Systems	445
22.4	Continuous Integration	446
22.5	The Change Management Process	447
22.5.1	Change Control	448
22.5.2	Impact Management	451
22.5.3	Configuration Audit	452
22.5.4	Status Reporting	452
22.6	Mobility and Agile Change Management	453
22.6.1	e-Change Control	453
22.6.2	Content Management	455
22.6.3	Integration and Publishing	455
22.6.4	Version Control	457
22.6.5	Auditing and Reporting	458
22.7	Summary	458

CHAPTER 23 SOFTWARE METRICS AND ANALYTICS 460

23.1	Software Measurement	461
23.1.1	Measures, Metrics, and Indicators	461
23.1.2	Attributes of Effective Software Metrics	462
23.2	Software Analytics	462
23.3	Product Metrics	463
23.3.1	Metrics for the Requirements Model	464
23.3.2	Design Metrics for Conventional Software	466
23.3.3	Design Metrics for Object-Oriented Software	468
23.3.4	User Interface Design Metrics	471
23.3.5	Metrics for Source Code	473
23.4	Metrics for Testing	474
23.5	Metrics for Maintenance	476
23.6	Process and Project Metrics	476

- 23.7 Software Measurement 479
- 23.8 Metrics for Software Quality 482
- 23.9 Establishing Software Metrics Programs 485
- 23.10 Summary 487

PART FOUR MANAGING SOFTWARE PROJECTS 489

CHAPTER 24 PROJECT MANAGEMENT CONCEPTS 490

- 24.1 The Management Spectrum 491
 - 24.1.1 The People 491
 - 24.1.2 The Product 491
 - 24.1.3 The Process 492
 - 24.1.4 The Project 492
- 24.2 People 493
 - 24.2.1 The Stakeholders 493
 - 24.2.2 Team Leaders 493
 - 24.2.3 The Software Team 494
 - 24.2.4 Coordination and Communications Issues 496
- 24.3 Product 497
 - 24.3.1 Software Scope 497
 - 24.3.2 Problem Decomposition 497
- 24.4 Process 498
 - 24.4.1 Melding the Product and the Process 498
 - 24.4.2 Process Decomposition 498
- 24.5 Project 500
- 24.6 The W⁵HH Principle 501
- 24.7 Critical Practices 502
- 24.8 Summary 502

CHAPTER 25 CREATING A VIABLE SOFTWARE PLAN 504

- 25.1 Comments on Estimation 505
- 25.2 The Project Planning Process 506

25.3	Software Scope and Feasibility	507
25.4	Resources	507
25.4.1	Human Resources	508
25.4.2	Reusable Software Resources	509
25.4.3	Environmental Resources	509
25.5	Data Analytics and Software Project Estimation	509
25.6	Decomposition and Estimation Techniques	511
25.6.1	Software Sizing	511
25.6.2	Problem-Based Estimation	512
25.6.3	An Example of LOC-Based Estimation	512
25.6.4	An Example of FP-Based Estimation	514
25.6.5	An Example of Process-Based Estimation	515
25.6.6	An Example of Estimation Using Use Case Points	517
25.6.7	Reconciling Estimates	518
25.6.8	Estimation for Agile Development	519
25.7	Project Scheduling	520
25.7.1	Basic Principles	521
25.7.2	The Relationship Between People and Effort	522
25.8	Defining a Project Task Set	523
25.8.1	A Task Set Example	524
25.8.2	Refinement of Major Tasks	524
25.9	Defining a Task Network	525
25.10	Scheduling	226
25.10.1	Time-Line Charts	526
25.10.2	Tracking the Schedule	528
25.11	Summary	530

CHAPTER 26 RISK MANAGEMENT 532

26.1	Reactive Versus Proactive Risk Strategies	533
26.2	Software Risks	534
26.3	Risk Identification	535
26.3.1	Assessing Overall Project Risk	536
26.3.2	Risk Components and Drivers	537
26.4	Risk Projection	538

26.4.1	Developing a Risk Table	538
26.4.2	Assessing Risk Impact	540
26.5	Risk Refinement	542
26.6	Risk Mitigation, Monitoring, and Management	543
26.7	The RMMM Plan	546
26.8	Summary	547

CHAPTER 27 A STRATEGY FOR SOFTWARE SUPPORT 549

27.1	Software Support	550
27.2	Software Maintenance	552
27.2.1	Maintenance Types	553
27.2.2	Maintenance Tasks	554
27.2.3	Reverse Engineering	555
27.3	Proactive Software Support	557
27.3.1	Use of Software Analytics	558
27.3.2	Role of Social Media	559
27.3.3	Cost of Support	559
27.4	Refactoring	560
27.4.1	Data Refactoring	561
27.4.2	Code Refactoring	561
27.4.3	Architecture Refactoring	561
27.5	Software Evolution	562
27.5.1	Inventory Analysis	563
27.5.2	Document Restructuring	564
27.5.3	Reverse Engineering	564
27.5.4	Code Refactoring	564
27.5.5	Data Refactoring	564
27.5.6	Forward Engineering	565
27.6	Summary	565

PART FIVE ADVANCED TOPICS 567

CHAPTER 28 SOFTWARE PROCESS IMPROVEMENT 568

28.1	What Is SPI?	569
28.1.1	Approaches to SPI	569
28.1.2	Maturity Models	570
28.1.3	Is SPI for Everyone?	571

28.2	The SPI Process	571
28.2.1	Assessment and GAP Analysis	572
28.2.2	Education and Training	573
28.2.3	Selection and Justification	573
28.2.4	Installation/Migration	574
28.2.5	Evaluation	575
28.2.6	Risk Management for SPI	575
28.3	The CMMI	576
28.4	Other SPI Frameworks	579
28.4.1	SPICE	579
28.4.2	TickIT Plus	579
28.5	SPI Return on Investment	580
28.6	SPI Trends	580
28.7	Summary	581

CHAPTER 29 EMERGING TRENDS IN SOFTWARE ENGINEERING 583

29.1	Technology Evolution	584
29.2	Software Engineering as a Discipline	585
29.3	Observing Software Engineering Trends	586
29.4	Identifying “Soft Trends”	587
29.4.1	Managing Complexity	588
29.4.2	Open-World Software	589
29.4.3	Emergent Requirements	590
29.4.4	The Talent Mix	591
29.4.5	Software Building Blocks	591
29.4.6	Changing Perceptions of “Value”	592
29.4.7	Open Source	592
29.5	Technology Directions	593
29.5.1	Process Trends	593
29.5.2	The Grand Challenge	594
29.5.3	Collaborative Development	595
29.5.4	Requirements Engineering	596
29.5.5	Model-Driven Software Development	596
29.5.6	Search-Based Software Engineering	597
29.5.7	Test-Driven Development	598
29.6	Tools-Related Trends	599
29.7	Summary	600

CHAPTER 30 CONCLUDING COMMENTS 602

30.1	The Importance of Software—Revisited	603
30.2	People and the Way They Build Systems	603
30.3	Knowledge Discovery	605
30.4	The Long View	606
30.5	The Software Engineer's Responsibility	607
30.6	A Final Comment from RSP	609
APPENDIX 1	An Introduction to UML	611
APPENDIX 2	Data Science for Software Engineers	629
REFERENCES		639
INDEX		659