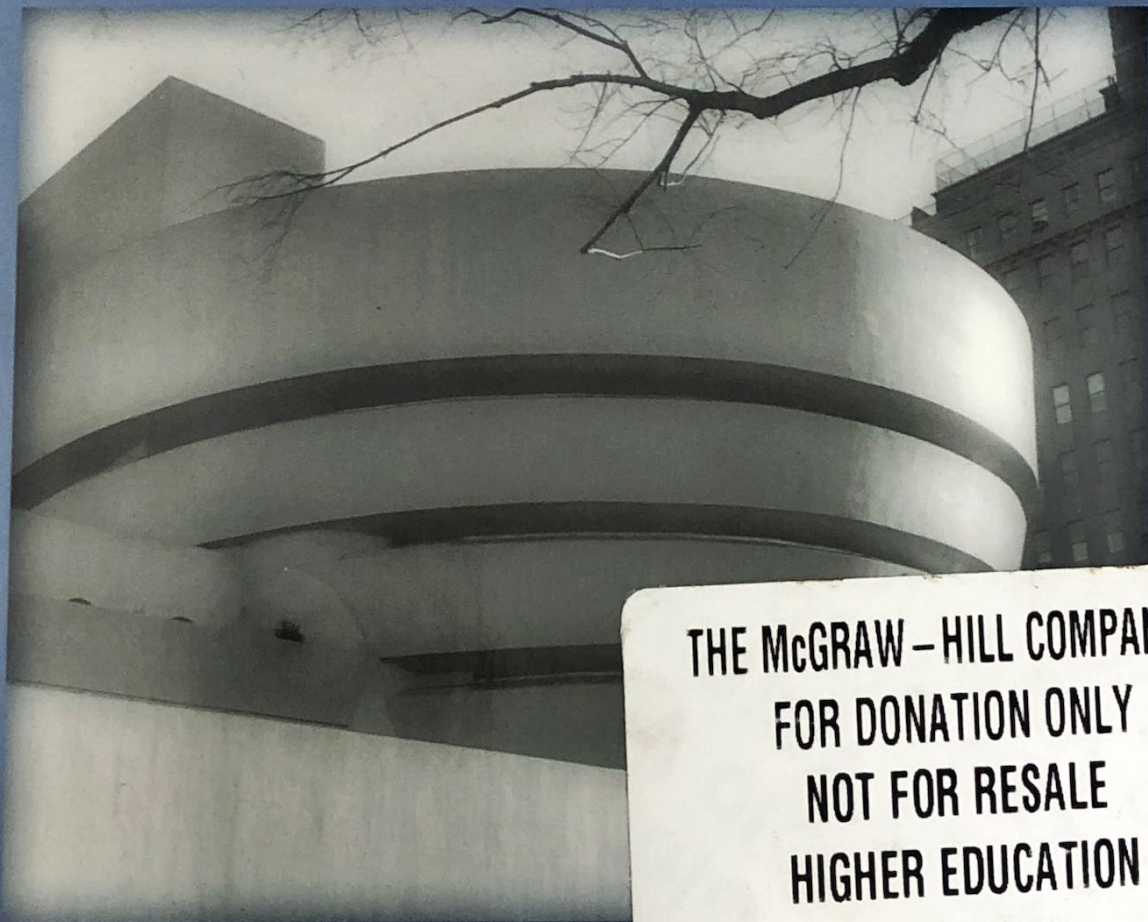


OBJECT-ORIENTED & CLASSICAL SOFTWARE ENGINEERING

SIXTH EDITION



THE MCGRAW-HILL COMPANIES
FOR DONATION ONLY
NOT FOR RESALE
HIGHER EDUCATION
23-ASA-006

STEPHEN R. SCHACH

Object-Oriented and Classical Software Engineering

Sixth Edition

Stephen R. Schach
Vanderbilt University



Higher Education

Boston Burr Ridge, IL Dubuque, IA Madison, WI 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



Higher Education

OBJECT-ORIENTED AND CLASSICAL SOFTWARE ENGINEERING, SIXTH EDITION

Published by McGraw-Hill, a business unit of The McGraw-Hill Companies, Inc., 1221 Avenue of the Americas, New York, NY 10020. Copyright © 2005, 2002, 1999, 1996, 1993, 1990 by The McGraw-Hill Companies, Inc. All rights reserved. No part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written consent of The McGraw-Hill Companies, Inc., including, but not limited to, in any network or other electronic storage or transmission, or broadcast for distance learning.

Some ancillaries, including electronic and print components, may not be available to customers outside the United States.

This book is printed on acid-free paper.

2 3 4 5 6 7 8 9 0 DOC/DOC 0 9 8 7 6 5 4

ISBN 0-07-286551-2

Publisher: *Elizabeth A. Jones*

Managing developmental editor: *Emily J. Lupash*

Marketing manager: *Dawn R. Bercier*

Senior project manager: *Jane Mohr*

Lead production supervisor: *Sandy Ludovissy*

Lead media project manager: *Audrey A. Reiter*

Senior coordinator of freelance design: *Michelle D. Whitaker*

Cover designer: *Christopher Reese*

Cover image: © *Sherman/Getty Images*

Compositor: *Interactive Composition Corporation*

Typeface: *10/12 Times Roman*

Printer: *R. R. Donnelley Crawfordsville, IN*

Library of Congress Cataloging-in-Publication Data

Schach, Stephen R.

Object-oriented and classical software engineering / Stephen R. Schach. — 6th ed.

p. cm.

Includes bibliographical references and index.

ISBN 0-07-286551-2

1. Software engineering. 2. Object-oriented programming (Computer science). 3. UML (Computer science). 4. C++ (Computer program language). I. Title.

QA76.758S318 2005

005.1'17—dc22

2003024034

CIP

To Sharon, David, and Lauren

The following are registered trademarks:

ADF	JBuilder	Requisite Pro
Analyst/Designer	Linux	Rhapsody
Ant	Lotus 1-2-3	Rose
Apache	Lucent Technologies	SilkTest
Apple	MacApp	Software through Pictures
AS/400	Macintosh	Solaris
AT&T	Macintosh Toolbox	SourceSafe
Bachman Product Set	MacProject	SPARCstation
Borland	Microsoft Foundation	Sun
Bugzilla	Class Library	Sun Enterprise
Capability Maturity Model	Motif	Sun Microsystems
CCC	MS-DOS	Sun ONE Studio
ClearQuest	MVS/360	System Architect
CMM	Natural	Together
Coca-Cola	Netscape	UNIX
CORBA	<i>New York Times</i>	VAX
CVS	Object C	Visual Component Library
DB2	Objective-C	Visual C++
e-Components	ObjectWindows Library	Visual J++
Emeraude	1-800-flowers.com	VM/370
Enterprise JavaBeans	Oracle	VMS
Excel	OS/360	<i>Wall Street Journal</i>
Focus	OS/370	WebSphere
Ford	OS/VS2	Win32
Foundation	Post-it note	Windows 95
FoxBASE	PowerBuilder	Windows 2000
GCC	Project	Windows NT
Hewlett-Packard	PureCoverage	X11
IBM	PVCS	XRunner
IMS/360	QAPartner	Zip disk
Java	Rational	ZIP code

Contents

Preface xv

PART ONE

INTRODUCTION TO SOFTWARE ENGINEERING 1

Chapter 1

The Scope of Software Engineering 3

- Learning Objectives 3
- 1.1 Historical Aspects 4
- 1.2 Economic Aspects 6
- 1.3 Maintenance Aspects 7
 - 1.3.1 *Classical and Modern Views of Maintenance* 9
 - 1.3.2 *The Importance of Postdelivery Maintenance* 11
- 1.4 Requirements, Analysis, and Design Aspects 13
- 1.5 Team Development Aspects 15
- 1.6 Why There Is No Planning Phase 16
- 1.7 Why There Is No Testing Phase 17
- 1.8 Why There Is No Documentation Phase 17
- 1.9 The Object-Oriented Paradigm 18
- 1.10 The Object-Oriented Paradigm in Perspective 23
- 1.11 Terminology 23
- 1.12 Ethical Issues 26
- Chapter Review 27
- For Further Reading 27
- Key Terms 28
- Problems 29
- References 30

Chapter 2

Software Life-Cycle Models 34

- Learning Objectives 34
- 2.1 Software Development in Theory 34
- 2.2 Winburg Mini Case Study 35

- 2.3 Lessons of the Winburg Mini Case Study 39
- 2.4 Teal Tractors Mini Case Study 39
- 2.5 Iteration and Incrementation 40
- 2.6 Winburg Mini Case Study Revisited 44
- 2.7 Risks and Other Aspects of Iteration and Incrementation 45
- 2.8 Managing Iteration and Incrementation 47
- 2.9 Other Life-Cycle Models 48
 - 2.9.1 *Code-and-Fix Life-Cycle Model* 48
 - 2.9.2 *Waterfall Life-Cycle Model* 49
 - 2.9.3 *Rapid-Prototyping Life-Cycle Model* 51
 - 2.9.4 *Extreme Programming and Agile Processes* 52
 - 2.9.5 *Synchronize-and-Stabilize Life-Cycle Model* 54
 - 2.9.6 *Spiral Life-Cycle Model* 54
- 2.10 Comparison of Life-Cycle Models 58
- Chapter Review 59
- For Further Reading 60
- Key Terms 60
- Problems 61
- References 61

Chapter 3

The Software Process 64

- Learning Objectives 64
- 3.1 The Unified Process 66
- 3.2 Iteration and Incrementation within the Object-Oriented Paradigm 67
- 3.3 The Requirements Workflow 68
- 3.4 The Analysis Workflow 70
- 3.5 The Design Workflow 72
- 3.6 The Implementation Workflow 73
- 3.7 The Test Workflow 74
 - 3.7.1 *Requirements Artifacts* 74
 - 3.7.2 *Analysis Artifacts* 74
 - 3.7.3 *Design Artifacts* 75
 - 3.7.4 *Implementation Artifacts* 75

- 3.8** Postdelivery Maintenance 77
- 3.9** Retirement 78
- 3.10** The Phases of the Unified Process 78
 - 3.10.1 *The Inception Phase* 79
 - 3.10.2 *The Elaboration Phase* 81
 - 3.10.3 *The Construction Phase* 82
 - 3.10.4 *The Transition Phase* 82
- 3.11** One- versus Two-Dimensional Life-Cycle Models 83
- 3.12** Improving the Software Process 84
- 3.13** Capability Maturity Models 85
- 3.14** Other Software Process Improvement Initiatives 88
- 3.15** Costs and Benefits of Software Process Improvement 89
 - Chapter Review 91
 - For Further Reading 91
 - Key Terms 92
 - Problems 92
 - References 93

Chapter 4

Teams 96

- Learning Objectives 96
- 4.1** Team Organization 96
- 4.2** Democratic Team Approach 98
 - 4.2.1 *Analysis of the Democratic Team Approach* 99
- 4.3** Classical Chief Programmer Team Approach 99
 - 4.3.1 *The New York Times Project* 101
 - 4.3.2 *Impracticality of the Classical Chief Programmer Team Approach* 102
- 4.4** Beyond Chief Programmer and Democratic Teams 102
- 4.5** Synchronize-and-Stabilize Teams 106
- 4.6** Extreme Programming Teams 106
- 4.7** People Capability Maturity Model 107
- 4.8** Choosing an Appropriate Team Organization 108
 - Chapter Review 109
 - For Further Reading 109
 - Key Terms 109
 - Problems 109
 - References 110

Chapter 5

The Tools of the Trade 112

- Learning Objectives 112
- 5.1** Stepwise Refinement 112
 - 5.1.1 *Stepwise Refinement Mini Case Study* 113
- 5.2** Cost-Benefit Analysis 118
- 5.3** Software Metrics 119
- 5.4** CASE 121
- 5.5** Taxonomy of CASE 122
- 5.6** Scope of CASE 123
- 5.7** Software Versions 127
 - 5.7.1 *Revisions* 127
 - 5.7.2 *Variations* 128
- 5.8** Configuration Control 129
 - 5.8.1 *Configuration Control during Postdelivery Maintenance* 131
 - 5.8.2 *Baselines* 131
 - 5.8.3 *Configuration Control during Development* 132
- 5.9** Build Tools 132
- 5.10** Productivity Gains with CASE Technology 133
 - Chapter Review 135
 - For Further Reading 135
 - Key Terms 135
 - Problems 136
 - References 137

Chapter 6

Testing 139

- Learning Objectives 139
- 6.1** Quality Issues 140
 - 6.1.1 *Software Quality Assurance* 141
 - 6.1.2 *Managerial Independence* 141
- 6.2** Non-Execution-Based Testing 142
 - 6.2.1 *Walkthroughs* 143
 - 6.2.2 *Managing Walkthroughs* 143
 - 6.2.3 *Inspections* 144
 - 6.2.4 *Comparison of Inspections and Walkthroughs* 146
 - 6.2.5 *Strengths and Weaknesses of Reviews* 147
 - 6.2.6 *Metrics for Inspections* 147

6.3	Execution-Based Testing	147
6.4	What Should Be Tested?	148
	6.4.1 Utility	149
	6.4.2 Reliability	149
	6.4.3 Robustness	149
	6.4.4 Performance	150
	6.4.5 Correctness	150
6.5	Testing versus Correctness Proofs	152
	6.5.1 Example of a Correctness Proof	152
	6.5.2 Correctness Proof Mini Case Study	156
	6.5.3 Correctness Proofs and Software Engineering	157
6.6	Who Should Perform Execution-Based Testing?	159
6.7	When Testing Stops	161
	Chapter Review	161
	For Further Reading	161
	Key Terms	162
	Problems	162
	References	164

Chapter 7

From Modules to Objects 166

	Learning Objectives	166
7.1	What Is a Module?	166
7.2	Cohesion	170
	7.2.1 Coincidental Cohesion	170
	7.2.2 Logical Cohesion	171
	7.2.3 Temporal Cohesion	172
	7.2.4 Procedural Cohesion	172
	7.2.5 Communicational Cohesion	173
	7.2.6 Functional Cohesion	173
	7.2.7 Informational Cohesion	174
	7.2.8 Cohesion Example	174
7.3	Coupling	175
	7.3.1 Content Coupling	176
	7.3.2 Common Coupling	176
	7.3.3 Control Coupling	178
	7.3.4 Stamp Coupling	178
	7.3.5 Data Coupling	180
	7.3.6 Coupling Example	180
	7.3.7 The Importance of Coupling	181

7.4	Data Encapsulation	182
	7.4.1 Data Encapsulation and Development	184
	7.4.2 Data Encapsulation and Maintenance	185
7.5	Abstract Data Types	191
7.6	Information Hiding	192
7.7	Objects	194
7.8	Inheritance, Polymorphism, and Dynamic Binding	198
7.9	The Object-Oriented Paradigm	200
	Chapter Review	203
	For Further Reading	203
	Key Terms	204
	Problems	204
	References	205

Chapter 8

Reusability and Portability 208

	Learning Objectives	208
8.1	Reuse Concepts	209
8.2	Impediments to Reuse	211
8.3	Reuse Case Studies	212
	8.3.1 Raytheon Missile Systems Division	212
	8.3.2 European Space Agency	214
8.4	Objects and Reuse	215
8.5	Reuse during Design and Implementation	215
	8.5.1 Design Reuse	215
	8.5.2 Application Frameworks	217
	8.5.3 Design Patterns	217
	8.5.4 Software Architecture	220
	8.5.5 Component-Based Software Engineering	222
8.6	Reuse and Postdelivery Maintenance	222
8.7	Portability	223
	8.7.1 Hardware Incompatibilities	224
	8.7.2 Operating System Incompatibilities	225
	8.7.3 Numerical Software Incompatibilities	225
	8.7.4 Compiler Incompatibilities	226
8.8	Why Portability?	229

- 8.9** Techniques for Achieving Portability 230
 - 8.9.1 *Portable System Software* 230
 - 8.9.2 *Portable Application Software* 231
 - 8.9.3 *Portable Data* 232
- Chapter Review 233
- For Further Reading 233
- Key Terms 234
- Problems 234
- References 236

Chapter 9

Planning and Estimating 240

- Learning Objectives 240
- 9.1** Planning and the Software Process 241
- 9.2** Estimating Duration and Cost 242
 - 9.2.1 *Metrics for the Size of a Product* 243
 - 9.2.2 *Techniques of Cost Estimation* 247
 - 9.2.3 *Intermediate COCOMO* 249
 - 9.2.4 *COCOMO II* 252
 - 9.2.5 *Tracking Duration and Cost Estimates* 253
- 9.3** Components of a Software Project Management Plan 254
- 9.4** Software Project Management Plan Framework 255
- 9.5** IEEE Software Project Management Plan 257
- 9.6** Planning Testing 260
- 9.7** Planning Object-Oriented Projects 261
- 9.8** Training Requirements 261
- 9.9** Documentation Standards 262
- 9.10** CASE Tools for Planning and Estimating 263
- 9.11** Testing the Software Project Management Plan 263
 - Chapter Review 263
 - For Further Reading 264
 - Key Terms 264
 - Problems 265
 - References 266

PART TWO

THE WORKFLOWS OF THE SOFTWARE LIFE CYCLE 269

Chapter 10

Requirements 271

- Learning Objectives 271
- 10.1** Determining What the Client Needs 271
- 10.2** Overview of the Requirements Workflow 272
- 10.3** Understanding the Domain 273
- 10.4** The Business Model 274
 - 10.4.1 *Interviewing* 274
 - 10.4.2 *Other Techniques* 275
 - 10.4.3 *Use Cases* 276
- 10.5** Initial Requirements 277
- 10.6** Initial Understanding of the Domain: The Osbert Oglesby Case Study 278
- 10.7** Initial Business Model: The Osbert Oglesby Case Study 279
- 10.8** Initial Requirements: The Osbert Oglesby Case Study 282
- 10.9** Continuing the Requirements Workflow: The Osbert Oglesby Case Study 284
- 10.10** The Test Workflow: The Osbert Oglesby Case Study 291
- 10.11** The Classical Requirements Phase 292
- 10.12** Rapid Prototyping 293
- 10.13** Human Factors 294
- 10.14** Reusing the Rapid Prototype 295
- 10.15** CASE Tools for the Requirements Workflow 296
- 10.16** Metrics for the Requirements Workflow 297
- 10.17** Challenges of the Requirements Workflow 297
 - Chapter Review 299
 - For Further Reading 299
 - Key Terms 300
 - Case Study Key Terms 300
 - Problems 300
 - References 301

Chapter 11

Classical Analysis 303

- Learning Objectives 303
- 11.1 The Specification Document 303
- 11.2 Informal Specifications 305
 - 11.2.1 *Correctness Proof Mini Case Study Redux* 306
- 11.3 Structured Systems Analysis 307
 - 11.3.1 *Sally's Software Shop Mini Case Study* 307
- 11.4 Structured Systems Analysis: The Osbert Oglesby Case Study 315
- 11.5 Other Semiformal Techniques 316
- 11.6 Entity-Relationship Modeling 317
- 11.7 Finite State Machines 319
 - 11.7.1 *Finite State Machines: The Elevator Problem Case Study* 321
- 11.8 Petri Nets 325
 - 11.8.1 *Petri Nets: The Elevator Problem Case Study* 328
- 11.9 Z 330
 - 11.9.1 *Z: The Elevator Problem Case Study* 330
 - 11.9.2 *Analysis of Z* 333
- 11.10 Other Formal Techniques 334
- 11.11 Comparison of Classical Analysis Techniques 335
- 11.12 Testing during Classical Analysis 335
- 11.13 CASE Tools for Classical Analysis 336
- 11.14 Metrics for Classical Analysis 337
- 11.15 Software Project Management Plan: The Osbert Oglesby Case Study 338
- 11.16 Challenges of Classical Analysis 338
- Chapter Review 339
- For Further Reading 339
- Key Terms 340
- Case Study Key Terms 340
- Problems 340
- References 342

Chapter 12

Object-Oriented Analysis 346

- Learning Objectives 346
- 12.1 The Analysis Workflow 347
- 12.2 Extracting the Entity Classes 348
- 12.3 Object-Oriented Analysis: The Elevator Problem Case Study 349
- 12.4 Functional Modeling: The Elevator Problem Case Study 349
- 12.5 Entity Class Modeling: The Elevator Problem Case Study 351
 - 12.5.1 *Noun Extraction* 352
 - 12.5.2 *CRC Cards* 354
- 12.6 Dynamic Modeling: The Elevator Problem Case Study 355
- 12.7 The Test Workflow: Object-Oriented Analysis 358
- 12.8 Extracting the Boundary and Control Classes 362
- 12.9 The Initial Functional Model: The Osbert Oglesby Case Study 363
- 12.10 The Initial Class Diagram: The Osbert Oglesby Case Study 365
- 12.11 The Initial Dynamic Model: The Osbert Oglesby Case Study 371
- 12.12 Extracting the Boundary Classes: The Osbert Oglesby Case Study 373
- 12.13 Extracting the Control Classes: The Osbert Oglesby Case Study 374
- 12.14 Refining the Use Cases: The Osbert Oglesby Case Study 374
- 12.15 Use-Case Realization: The Osbert Oglesby Case Study 377
 - 12.15.1 *Buy a Masterpiece Use Case* 377
 - 12.15.2 *Buy a Masterwork Use Case* 382
 - 12.15.3 *Buy Other Painting Use Case* 383
 - 12.15.4 *The Remaining Five Use Cases* 384
- 12.16 Incrementing the Class Diagram: The Osbert Oglesby Case Study 386
- 12.17 The Test Workflow: The Osbert Oglesby Case Study 388
- 12.18 The Specification Document in the Unified Process 389
- 12.19 More on Actors and Use Cases 390
- 12.20 CASE Tools for the Object-Oriented Analysis Workflow 391

- 12.21** Challenges of the Object-Oriented Analysis Workflow 391
- Chapter Review 392
- For Further Reading 392
- Key Terms 393
- Problems 393
- References 395

Chapter 13 Design 397

- Learning Objectives 397
- 13.1** Design and Abstraction 398
- 13.2** Operation-Oriented Design 398
- 13.3** Data Flow Analysis 399
 - 13.3.1 Mini Case Study: Word Counting 400
 - 13.3.2 Data Flow Analysis Extensions 405
- 13.4** Transaction Analysis 405
- 13.5** Data-Oriented Design 407
- 13.6** Object-Oriented Design 408
- 13.7** Object-Oriented Design: The Elevator Problem Case Study 409
- 13.8** Object-Oriented Design: The Osbert Oglesby Case Study 412
- 13.9** The Design Workflow 416
- 13.10** The Test Workflow: Design 418
- 13.11** The Test Workflow: The Osbert Oglesby Case Study 419
- 13.12** Formal Techniques for Detailed Design 419
- 13.13** Real-Time Design Techniques 419
- 13.14** CASE Tools for Design 421
- 13.15** Metrics for Design 421
- 13.16** Challenges of the Design Workflow 422
 - Chapter Review 423
 - For Further Reading 423
 - Key Terms 424
 - Problems 424
 - References 425

Chapter 14 Implementation 428

- Learning Objectives 428
- 14.1** Choice of Programming Language 428
- 14.2** Fourth-Generation Languages 431

- 14.3** Good Programming Practice 433
 - 14.3.1 Use of Consistent and Meaningful Variable Names 434
 - 14.3.2 The Issue of Self-Documenting Code 435
 - 14.3.3 Use of Parameters 436
 - 14.3.4 Code Layout for Increased Readability 437
 - 14.3.5 Nested **if** Statements 437
- 14.4** Coding Standards 439
- 14.5** Code Reuse 439
- 14.6** Integration 440
 - 14.6.1 Top-down Integration 440
 - 14.6.2 Bottom-up Integration 442
 - 14.6.3 Sandwich Integration 442
 - 14.6.4 Integration of Object-Oriented Products 443
 - 14.6.5 Management of Integration 444
- 14.7** The Implementation Workflow 445
- 14.8** The Implementation Workflow: The Osbert Oglesby Case Study 445
- 14.9** The Test Workflow: Implementation 446
- 14.10** Test Case Selection 446
 - 14.10.1 Testing to Specifications versus Testing to Code 446
 - 14.10.2 Feasibility of Testing to Specifications 446
 - 14.10.3 Feasibility of Testing to Code 447
- 14.11** Black-Box Unit-Testing Techniques 450
 - 14.11.1 Equivalence Testing and Boundary Value Analysis 450
 - 14.11.2 Functional Testing 451
- 14.12** Black-Box Test Cases: The Osbert Oglesby Case Study 452
- 14.13** Glass-Box Unit-Testing Techniques 455
 - 14.13.1 Structural Testing: Statement, Branch, and Path Coverage 455
 - 14.13.2 Complexity Metrics 457
- 14.14** Code Walkthroughs and Inspections 458
- 14.15** Comparison of Unit-Testing Techniques 458
- 14.16** Cleanroom 459

- 14.17** Potential Problems When Testing Objects 460
- 14.18** Management Aspects of Unit Testing 462
- 14.19** When to Rewrite Rather than Debug a Code Artifact 463
- 14.20** Integration Testing 464
- 14.21** Product Testing 465
- 14.22** Acceptance Testing 466
- 14.23** The Test Workflow: The Osbert Oglesby Case Study 467
- 14.24** CASE Tools for Implementation 467
 - 14.24.1 CASE Tools for the Complete Software Process 467*
 - 14.24.2 Integrated Development Environments 468*
 - 14.24.3 Environments for Business Applications 469*
 - 14.24.4 Public Tool Infrastructures 469*
 - 14.24.5 Potential Problems with Environments 470*
- 14.25** Metrics for the Implementation Workflow 470
- 14.26** Challenges of the Implementation Workflow 471
 - Chapter Review 471
 - For Further Reading 472
 - Key Terms 473
 - Problems 473
 - References 475

Chapter 15

Postdelivery Maintenance 479

- Learning Objectives 479
- 15.1** Why Postdelivery Maintenance Is Necessary 480
- 15.2** What Is Required of Postdelivery Maintenance Programmers? 480
- 15.3** Postdelivery Maintenance Mini Case Study 482
- 15.4** Management of Postdelivery Maintenance 484
 - 15.4.1 Defect Reports 484*
 - 15.4.2 Authorizing Changes to the Product 485*

- 15.4.3 Ensuring Maintainability 486*
- 15.4.4 The Problem of Repeated Maintenance 486*

- 15.5** Maintenance of Object-Oriented Software 487
- 15.6** Postdelivery Maintenance Skills versus Development Skills 489
- 15.7** Reverse Engineering 490
- 15.8** Testing during Postdelivery Maintenance 491
- 15.9** CASE Tools for Postdelivery Maintenance 492
- 15.10** Metrics for Postdelivery Maintenance 492
- 15.11** Postdelivery Maintenance: The Osbert Oglesby Case Study 493
- 15.12** Challenges of Postdelivery Maintenance 493
 - Chapter Review 493
 - For Further Reading 493
 - Key Terms 494
 - Problems 494
 - References 495

Chapter 16

More on UML 497

- Learning Objectives 497
- 16.1** UML Is *Not* a Methodology 497
- 16.2** Class Diagrams 498
 - 16.2.1 Aggregation 499*
 - 16.2.2 Multiplicity 500*
 - 16.2.3 Composition 501*
 - 16.2.4 Generalization 502*
 - 16.2.5 Association 502*
- 16.3** Notes 503
- 16.4** Use-Case Diagrams 503
- 16.5** Stereotypes 503
- 16.6** Interaction Diagrams 504
- 16.7** Statecharts 506
- 16.8** Activity Diagrams 509
- 16.9** Packages 511
- 16.10** Component Diagrams 511
- 16.11** Deployment Diagrams 511
- 16.12** Review of UML Diagrams 511

- 16.13** UML and Iteration 512
 - Chapter Review 512
 - For Further Reading 513
 - Key Terms 513
 - Problems 513
 - References 514

Bibliography 515

Appendix A

- Term Project: Ophelia's Oasis in the Amlet Desert 539

Appendix B

- Software Engineering Resources 542

Appendix C

- Requirements Workflow: The Osbert Oglesby Case Study 544

Appendix D

- Structured Systems Analysis: The Osbert Oglesby Case Study 545

Appendix E

- Analysis Workflow: The Osbert Oglesby Case Study 549

Appendix F

- Software Project Management Plan: The Osbert Oglesby Case Study 550

Appendix G

- Design Workflow: The Osbert Oglesby Case Study 555

Appendix H

- Implementation Workflow: The Osbert Oglesby Case Study (C++ Version) 563

Appendix I

- Implementation Workflow: The Osbert Oglesby Case Study (Java Version) 564

Appendix J

- Test Workflow: The Osbert Oglesby Case Study 565

- Author Index 567**

- Subject Index 570**