Software Design

From Programming to Architecture

WILEY
INTERNATIONAL
EDITION
RESTRICTED: NOT FOR SALE IN NORTH AMERICA

Eric Braude

Preface vii
Acknowledgments xi

PROLOGUE *THE SOFTWARE PROCESS* 1

0.1 Introduction to Software Process 1
0.1.1 The Phases of Software Process 1
0.1.2 Styles of Software Process 2
0.1.3 Common Procedures for Software Process: Writing Against Prior Phases; Inspection 4
0.2 Requirements Analysis 4
0.2.1 Styles of Requirements Analysis 5
0.2.2 Tips on Analyzing and Specifying Requirements 7
0.3 Design (What this book covers!) 7
0.3.1 The Meaning of "Software Design" 8
0.3.2 The Manner in Which This Book Teaches Design 8
0.4 Coding 9
0.4.1 When Does Design Stop and Coding Begin? 9
0.4.2 Basic Coding Tips 9
0.5 Testing 15
0.5.1 Testing and Correctness 15
0.5.2 Types of Testing 15
0.5.3 Tips on Testing 16
0.6 Maintenance 17
0.6.1 The Meaning of "Maintenance" 17
0.6.2 How Maintenance Considerations Affect Process 17
Summary of Software Process 18
Exercises 18

PART I. ***DESIGN PRINCIPLES, THE UNIFED MODELING LANGUAGE, AND CODE-LEVEL DESIGN***

CHAPTER 1 *PROGRAMMING REVIEW AND INTRODUCTION TO SOFTWARE DESIGN* 21

1.1 "Software Design:" Its Meaning and How this Book Introduces It 21
1.1.1 Part I of this Book: Design Principles, the Unified Modeling Language, and Code-Level Design 22
1.1.2 Part II of this Book: Design Patterns 22
1.1.3 Part III of this Book: Components 22
1.1.4 Part IV of this Book: Object-Oriented Analysis and Design 22
1.2 Documenting Functions 22
1.2.1 Java Coding Standards Used in this Book 23
1.2.2 Specifying What a Function Does 23
1.2.3 Describing *How* a Function Satisfies Its Specification 25
1.2.4 Error Processing 27
1.3 Review of Good Programming Habits for Writing Functions 27
1.4 Goals of Software Design 27
1.4.1 An Example 28
1.4.2 Correctness, Sufficiency, Modularity, and Readability 31
1.4.3 Robustness 31
1.4.4 Flexibility 36
1.4.5 Reusability 37
1.4.6 Efficiency 37
1.4.7 Other Design Goals: Reliability and Usability 37
1.5 The Need for a Software Design Notation 38
Chapter Summary 38
Exercises 38
Appendix: Nonrobust Code for *CommandLineCalculator* Example 41

CHAPTER 2 *OBJECT-ORIENTATION* 44

2.1 The Goals of Object-Orientation 44
2.2 Classes and Objects 46
2.2.1 Introducing Classes 46
2.2.2 Instantiation 47
2.2.3 The Members of a Class 47

2.2.4 The "Client" Concept in Object-Orientation 48
2.3 Key Features of Object-Orientation 49
2.3.1 Inheritance 50
2.3.2 Polymorphism 50
2.3.3 Interfaces and Encapsulation 55
2.4 Issues to Be Addressed 57
Chapter Summary 58
Exercises 58

CHAPTER 3 *THE UNIFIED MODELING LANGUAGE* 61

3.1 Goals of the Unified Modeling Language 61
3.2 Classes in UML 61
3.3 Class Relationships in UML 63
3.3.1 Inheritance 63
3.3.2 Aggregation 63
3.3.3 Dependency 65
3.3.4 Association 65
3.3.5 An Example 66
3.4 Use Cases 67
3.4.1 What Are Use Cases? 67
3.4.2 Collections of Use Cases 68
3.4.3 Combining Use Cases 69
3.5 Sequence Diagrams 69
3.5.1 The Parts of a Sequence Diagram 70
3.6 State Models 72
3.6.1 The Meaning of *State* 72
3.6.2 Events 73
3.6.3 Transitions 73
3.6.4 State-Transition Diagrams 74
3.7 Activity Diagrams 74
3.7.1 Activity Diagram Notation 74
3.7.2 Example Using Activity Diagrams 75
3.8 An Example 76
Chapter Summary 79
Exercises 80

CHAPTER 4 *DESIGN PRINCIPLES I: CORRECTNESS AND ROBUSTNESS* 82

4.1 Correctness and Sufficiency 82
4.1.1 Approaches to Correctness 82
4.1.2 Interfaces to Modules 88
4.1.3 Modularization 91
4.1.4 Refactoring for Correctness and Sufficiency 93
4.2 Robustness 94
4.2.1 Verifying Input (Ensuring Environmental Robustness) 94
4.2.2 Initializing to Improve Robustness 95
4.2.3 Parameter Passing Techniques to Improve Robustness 96
4.2.4 Enforcing Intentions 98
4.3 Design Details: How Much is Enough? 99
Chapter Summary 100
Exercises 100

CHAPTER 5 *DESIGN PRINCIPLES II: FLEXIBILITY, REUSABILITY, AND EFFICIENCY* 104

5.1 Flexibility 104
5.1.1 The Goals of Flexibility 104
5.1.2 Designs for Adding *More of the Same Kind* of Functionality 104
5.1.3 Design for Adding *Different* Functionality 105
5.1.4 Design for Flexibility 107
5.2 Reusability 107
5.2.1 The Goals of Reusability 108
5.2.2 Reusability of Function Design 108
5.2.3 Class Selection for Reuse 109
5.2.4 Class Combination for Reuse 110
5.3 Efficiency 114
5.3.1 Speed Efficiency 116
5.3.2 Storage Efficiency 118
5.4 Trade-offs Among Robustness, Flexibility, Reusability, and Efficiency 118
Chapter Summary 120
Exercises 121

PART II. *DESIGN PATTERNS*

CHAPTER 6 *INTRODUCTION TO DESIGN PATTERNS* 123

6.1 Recurring Design Purposes 123
6.1.1 An Example of a Recurring Design Purpose 124
6.2 What Are Design Patterns? 125
6.2.1 Example Application: Without Applying Design Patterns 126
6.2.2 Example Application: Applying a Design Pattern 126

6.3 Summary of Design Patterns by Type: *Creational, Structural,* and *Behavioral* 130
6.3.1 Creational Design Patterns 131
6.3.2 Structural Design Patterns 131
6.3.3 Behavioral Design Patterns 133
6.4 Characteristics of Design Patterns: *Viewpoints, Roles,* and *Levels* 136
6.4.1 Two Viewpoints for Describing a Pattern: *Static* and *Dynamic* 136
6.4.2 Two Levels of a Pattern: *Abstract* and *Concrete* 138
6.4.3 Three Roles Involved in Pattern Usage: *Pattern Application, Client,* and *Setup* 139
6.5 Design Pattern Forms: *Delegation* and *Recursion* 141
6.5.1 The Origins and Meaning of *Delegation* 141
6.5.2 The *Delegation* Design Pattern Form 141
6.5.3 The *Recursion* Design Pattern Form ... 143
Conclusion and Summary 144
Exercises 145

CHAPTER 7 *CREATIONAL DESIGN PATTERNS* 147

7.1 The Purposes of Creational Design 147
7.2 *Factory* 148
7.2.1 Design Purpose of *Factory* 148
7.2.2 The *Factory* Interface for Clients 148
7.2.3 The *Factory* Class Model 148
7.2.4 Sequence Diagram for *Factory* 150
7.2.5 Example *Factory* Application: E-Mail Generation 150
7.2.6 Use of *Factory* in the Java API 151
7.2.7 Comments on *Factory* 152
7.3 *Singleton* 153
7.3.1 Design Purpose of *Singleton* 153
7.3.2 The *Singleton* Interface for Clients ... 153
7.3.3 The *Singleton* Class Model 154
7.3.4 Example *Singleton* Application: Experiment 155
7.3.5 Use of *Singleton* in the Java API 155
7.3.6 Comments on *Singleton* 156
7.4 *Abstract Factory* 157
7.4.1 Design Purpose of *Abstract Factory* 157
7.4.2 The *Abstract Factory* Interface For Clients 158
7.4.3 The *Abstract Factory* Class Model 159
7.4.4 The *Abstract Factory* Sequence Diagram 161
7.4.5 Example *Abstract Factory* Application: "Word Processor" 161
7.4.6 Use of *Abstract Factory* in the Java API 163
7.4.7 Comments on *Abstract Factory* 164
7.5 *Prototype* 166
7.5.1 Design Purpose of *Prototype* 166
7.5.2 The *Prototype* Interface for Clients 167
7.5.3 The *Prototype* Class Model 169
7.5.4 The *Prototype* Sequence Diagram 169
7.5.5 Example *Prototype* Applications: Customer Information Entry 170
7.5.6 *Prototype* and the Java API 173
7.5.7 Comments on *Prototype* 178
Summary 179
Exercises 179
Appendix A: Code Listing for E-Mail Generation Example (*Factory* Design Pattern Application, Section 7.2) 183
Appendix B: Code Listing for Experiment Example (*Singleton* Design Pattern Application, Section 7.3) 188
Appendix C: Code Listing for Word Processor Example (*Abstract Factory* Design Pattern Application, Section 7.4) 189
Appendix D: Code Listing for Customer Information Entry Example (*Prototype* Design Pattern Application, Section 7.5) 199

CHAPTER 8 *STRUCTURAL DESIGN PATTERNS* 205

8.1 Structural Design Purposes 205
8.2 *Façade:* Interfacing for a Collection of Classes 205
8.2.1 The Design Purpose of *Façade* 205
8.2.2 Interface for Clients of *Façade* 206
8.2.3 The *Façade* Class Model 206
8.2.4 Example *Façade* Applications 206
8.2.5 Application of *Façade* in the Java API 210
8.2.6 Comments on *Façade* 210

8.3 *Decorator:* Adding and Removing Responsibilities of a Class at Runtime........211
8.3.1 The Design Purpose of *Decorator*.....211
8.3.2 Interface for Clients of *Decorator*.....212
8.3.3 The *Decorator* Class Model...............212
8.3.4 The *Decorator* Sequence Diagram....212
8.3.5 Example Application of *Decorator:* Customer / Account..........................213
8.3.6 Applications of *Decorator* in the Java API..215
8.3.7 Comments on *Decorator*....................215
8.4 *Composite:* Representing Trees of Objects..216
8.4.1 The Design Purpose of *Composite*....216
8.4.2 Interface for Clients of *Composite*.....216
8.4.3 The *Composite* Class Model..............217
8.4.4 The *Composite* Sequence Diagram...218
8.4.5 Example *Composite* Application: Bank / Teller..218
8.4.6 The *Composite* Pattern in the Java API...220
8.4.7 Comments on *Composite*....................221
8.5 *Adapter:* Interfacing in a Flexible Manner...221
8.5.1 The Design Purpose of *Adapter*.........221
8.5.2 Interface for Clients of *Adapter*.........222
8.5.3 The *Adapter* Class Model...................222
8.5.4 The *Adapter* Sequence Diagram........222
8.5.5 Example Applications of *Adapter*.....223
8.5.6 *Adapter* and the Java API...................225
8.5.7 Comments on *Adapter*........................225
8.6 *Flyweight:* Managing Large Numbers of Barely Distinguishable Objects....................226
8.6.1 The Design Purpose of *Flyweight*......226
8.6.2 Interface for Clients of *Flyweight*......227
8.6.3 The *Flyweight* Class Model...............227
8.6.4 *Flyweight* Sequence Diagram............228
8.6.5 Example *Flyweight* Application: Text Magnifier...................................228
8.6.6 *Flyweight* in the Java API..................232
8.6.7 Comments on *Flyweight*.....................232
8.7 *Proxy:* Avoiding Unnecessary Operations.....233
8.7.1 The Design Purpose of *Proxy*............233
8.7.2 Interface for Clients of *Proxy*............233
8.7.3 The *Proxy* Class Model.......................234
8.7.4 *Proxy* Sequence Diagram...................234
8.7.5 *Proxy* in the Java API..........................235
8.7.6 Example Applications of *Proxy:* Telephone Numbers..........................235
8.7.7 Comments on *Proxy*............................237
Summary of Structural Design Patterns................237
Exercises..238
Appendix A: Source Code for Section 8.2.4.2 "Banking" Application (*Façade* Design Pattern Application)................................242
Appendix B: Source Code for Customer / Account Example in Section 8.3.5 (*Decorator* Design Pattern Application)................................250
Appendix C: Source Code for Bank / Teller Example in Section 8.4.5 (*Composite* Design Pattern Application)............257
Appendix D: Source Code for Text Magnifier Example in Section 8.6.5 (*Flyweight* Design Pattern Application)............262
Appendix E: Source Code for "Telephone Numbers" Example in Section 8.7.6 (*Prototype* Design Pattern Application)............270

CHAPTER 9 *BEHAVIORAL DESIGN PATTERNS*..276

9.1 Behavioral Design Purposes............................276
9.2 *Interpreter:* Parsing Expressions....................276
9.2.1 *Interpreter* Design Purposes and Examples..276
9.2.2 *Interpreter* Interfaces for Clients........277
9.2.3 The *Interpreter* Class Model...............278
9.2.4 The *Interpreter* Sequence Diagram...278
9.2.5 Example *Interpreter* Application: Network Assembly............................279
9.2.6 *Interpreter* in the Java API..................282
9.2.7 Comments on *Interpreter*....................282
9.3 *Iterator:* Visiting the Members of a Collection...283
9.3.1 *Iterator* Design Purposes and Examples..283
9.3.2 *Iterator* Interfaces for Clients.............284
9.3.3 The *Iterator* Class Model....................286
9.3.4 Example *Iterator* Application: Organization Chart............................287
9.3.5 *Iterator* in the Java API.......................290
9.3.6 Comments on *Iterator*.........................294
9.4 *Mediator*...295
9.4.1 The Design Purpose of *Mediator*......295
9.4.2 The *Mediator* Class Model.................297
9.4.3 Sequence Diagram for *Mediator*........298
9.4.4 Example of *Mediator*...........................298
9.4.5 *Mediator* in the Java API.....................299
9.4.6 Comments on *Mediator*......................300

9.5 *Observer* 301
9.5.1 The Design Purposes of *Observer* 301
9.5.2 *Observer* Interfaces for Clients 301
9.5.3 The *Observer* Class Model 302
9.5.4 Example *Observer* Applications 302
9.5.5 *Observer* in the Java API 304
9.5.6 Comments on *Observer* 305
9.6 *State* 306
9.6.1 The Design Purposes of *State* 306
9.6.2 *State* Interfaces for Clients 306
9.6.3 The *State* Class Model 306
9.6.4 Example *State* Applications 307
9.6.5 *State* in the Java API 309
9.6.6 Comments on *State* 309
9.7 *Chain of Responsibility* 309
9.7.1 The Design Purpose of *Chain of Responsibility* 309
9.7.2 Interface to *Chain of Responsibility* for Clients 310
9.7.3 The *Chain of Responsibility* Class Model 310
9.7.4 Example *Chain of Responsibility* Application: Customer Information 311
9.7.5 *Chain of Responsibility* in the Java API 313
9.7.6 Comments on *Chain of Responsibility* 313
9.8 *Command* 314
9.8.1 The Design Purposes of *Command* 314
9.8.2 Interface to *Command* for Clients 314
9.8.3 The *Command* Class Model 314
9.8.4 Example *Command* Applications 315
9.8.5 *Command* in the Java API 318
9.8.6 Comments on *Command* 319
9.9 *Template* 319
9.9.1 The Design Purpose of *Template* 319
9.9.2 Interface to *Template* for Clients 320
9.9.3 The *Template* Class Model 320
9.9.4 Example *Template* Application: Quadratic Solutions 320
9.9.5 Application of *Template* in the Java API 323
9.9.6 Comments on *Template* 323
Summary 324
Exercises 324
Appendix A: Source Code for Network Assembly Example (Application of *Interpreter* Design Pattern) 328
Appendix B: Source Code for "Organization Chart" Example (Application of *Iterator* Design Pattern) 337
Appendix C: Source Code for Mutual Funds Example (Application of *Observer* Design Pattern) 350
Appendix D: Source Code for Customer Information Example (Application of *Chain of Responsibility* Design Pattern) 356
Appendix E: Source Code for Undo Example (Application of *Command* Design Pattern) 367
Appendix F: Source Code for Quadratic Solutions Example (Application of *Template* Design Pattern) 377

PART III. *COMPONENTS*

CHAPTER 10 *INTRODUCTION TO COMPONENTS* 384

10.1 Definition of Components 385
10.2 A Demonstration of Component Use 386
10.3 What Components Consist of 389
10.3.1 Properties, Methods, Events 389
10.3.2 Manifests 390
10.3.3 Introspection and Metadata 390
10.4 UML Component Notation 391
10.5 The Phases of a Component Lifetime 391
10.5.1 Design/Implementation Time 392
10.5.2 Instance Creation Time 393
10.5.3 Assembly Time 393
10.5.4 Deployment Time 394
10.5.5 Execution Time 394
10.6 The CORBA Component Standards 395
Summary of This Chapter 396
Exercises 397
Appendix: Source Code for *Juggler* 398

CHAPTER 11 *JAVABEANS* 404

11.1 The Goals of JavaBeans 404
11.2 Definition of JavaBeans 404
11.2.1 An Example: Estimating Chair Production 405
11.2.2 The Phases of a Bean 406
11.3 Phase 1: Creating Bean Classes 407
11.3.1 Example: Creating the *ChairMaker* Bean 412
11.4 Phase 2: Creating Multiple-Class Beans 415

11.5 Phase 3: Creating Bean Instances 415
11.6 Phase 4: Combining and Deploying Beans ... 416
11.6.1 Phase 4a: Combining Beans in a Bean Environment 416
11.6.2 Phase 4b: Using Beans in Applications 419
11.7 Connecting Beans via Property Changes: "Bound" Variables 421
11.8 Using Beans in Java Server Pages 424
Summary 425
Exercises 426

CHAPTER 12 *MICROSOFT ASSEMBLIES* 428

12.1 Microsoft .NET Overview 428
12.1.1 Goals of .NET 428
12.1.2 The Architecture of .NET 429
12.1.3 .NET Assemblies (Components) 431
12.2 Elements of the C# Language 431
12.3 A Simple HelloWorld .NET Assembly 435
12.4 A .NET Component Containing Multiple Classes 437
12.5 Metadata in .NET Assemblies 438
12.5.1 Summary of .NET Metadata 438
12.5.2 Viewing Metadata 439
12.5.3 Versioning 441
12.5.4 Attributes 444
12.6 Identifying an Assembly 449
12.6.1 Private Assemblies 449
12.6.2 Shared Assemblies 453
Summary 456
Exercises 456

PART IV. ***OBJECT-ORIENTED ANALYSIS AND DESIGN***

CHAPTER 13 *REQUIREMENTS AND DOMAIN CLASSES* 458

13.1 Characteristics of Object-Oriented Analysis and Design 458
13.1.1 NonOOA&D Approaches to Application Development 459
13.1.2 The Basic OOA&D Approach 461
13.2 The *Encounter* Case Study Introduced 463
13.3 Obtaining Domain Classes 464
13.3.1 Use Case Requirements Description 465
13.3.2 Converting Use Cases to Sequence Diagrams 466
13.3.3 Harvesting Domain Classes from Sequence Diagrams 467
13.3.4 Finding Classes from Other Sources 467
13.4 Using Domain Classes to Organize Requirements 475
13.4.1 An Example Requirements Document Fragment 475
Summary of This Chapter 477
Exercises 477

CHAPTER 14 *ARCHITECTURES AND FRAMEWORKS* 479

14.1 The Meaning of Software Architectures 479
14.2 Models of OO Analysis and Design 481
14.2.1 Use Case Model 481
14.2.2 Class Models 482
14.2.3 Data Flow Models 482
14.2.4 State Models 485
14.3 Software Architectures 487
14.3.1 Goals for Architecture and Modularization 487
14.3.2 Modularization, Cohesion, and Coupling 487
14.3.3 Using *Façade* Design Pattern 490
14.3.4 Standard Software Architectures 491
14.4 Frameworks 496
14.4.1 The Meaning and Usage of Frameworks 496
14.4.2 Framework Usages: Toolkits, Abstract Architecture, and Control 498
14.4.3 Goals for Frameworks 498
14.4.4 Framework Development and Examples 500
14.5 Finalizing an Application Design 513
14.5.1 Designing Control 514
14.5.2 Designing for Interfaces 519
14.5.3 Connecting to Databases 524
14.5.4 Putting the Finishing Touches on Class Models 530
14.5.5 Completing the List of Functions 532
Summary 534
Exercises 534

REFERENCES 537

INDEX 539