Designing Object-Oriented C++ Applications

Using the Booch Method

Robert Cecil Martin

Object Mentor Associates

Prentice Hall, Englewood Cliffs, New Jersey 07632

Contents

Forward v

Preface vii

About This Book viii
Goals/Purpose viii
Audience viii
Anatomy and Physiology of Design ix
Software Is Hard ix
OOD Can Make Software "Softer" xi
Acknowledgements xii

Contents xiv

Figures xxv

C++ Listings xxx

00

OOverview 1

Introduction 1

Part 1: Some Common Questions about OOD 2
What is object-oriented design? 2
Is OOD better than previous software-design disciplines? If so, why? 3
Is OOD a revolution, or has it evolved from "older" design methods? 4
How is OOD different from more traditional software methods? 5
Software is software. Isn't OOD just a new face painted on an old canvas? 6
Will OOD solve all my design problems? 6

What can I really expect from OOD? 6
Is C++ a "true" object-oriented programming language? 7
Why was the Booch notation chosen for this book? 7

Part 2: Tutorial 8

Objects 8
Abstraction 9
Abstraction of State 11
Abstraction of Behavior 12
Collaboration Among Objects 13
Polymorphism 14
Classes 15
Specifying State 15
Instance Variables 15
Class Variables 16
Specifying Behavior 17
Instance Methods 17
Class Methods 18
Class Relationships 18
The "Contains" Relationship 18
The "Uses" Relationship 19
The "Inheritance" Relationship 19
Multiple Inheritance 21
Abstract Classes 22

Summary 23

Exercises 24

1

Static and Dynamic Design 26

Introduction 27

Connecting Requirements to the Design 28

Static and Dynamic Models in OOD 28
A Bill-of-Materials Case Study, in C++ 29
A C++ Example of a Static Model 29
Containment by Value 31
Containment by Reference 33
Inheritance 35
Containment with Cardinality 36
Polymorphic Containment 37
A C++ Example of a Dynamic Model 38
Iteration Between the Models 41

Why Is This Better Than Writing Code? 43

XRef: An Example of Object-Oriented Analysis and Design 43

XRef References: A Static Model 45

Dynamic Scenario: Printing the Cross-Reference Report 46

Rethinking the Static Model in Light of the Dynamic Requirements 47

Dynamic Scenario: Finding and Adding References 49

Making the Parser Reusable 51

Depicting Reusability with Class Categories 51

Reuse of the Identifier Class 53

The XRef Application 54

Converting the XRef Design into C++ 55

Summary of the XRef Example 57

The Mark IV Special Coffee Maker 60

Specification of the Mark IV Hardware Interface Functions 61

Designing the Mark IV Software 63

The Control Model: A Finite State Machine for the Coffee Maker 64

Finding the Objects 65

Object-Oriented Analysis: Finding the Underlying Abstractions 66

Assigning Responsibilities to the Abstractions 67

Reusing the Abstractions 71

CoffeeMaker Categories 78

Implementation of the Mark IV Coffee Maker 79

The Design and the Code Aren't a Perfect Match 80

Summary 80

Exercises 81

Mark IV Coffee Maker Implementation 84

The UI Category 84

The Warmer Category 86

The Sprayer Category 88

The CoffeeMaker Category 90

The MarkIV Category 92

2

Managing Complexity **106**

Introduction 107

Managing vs. Reducing Complexity 107

Abstraction: "The Most Powerful Tool" 108

Product Costing Policy: Case Study. 109

Grouping 109
The Open-Closed Principle 109
Using Grouping Strategies to Close a Function 115
Hiding (Restricting Visibility) 119
The Problem of Too Much Visibility 120
Hiding and Closure 121
Abstraction and Hiding 121
Completing the Product / Policy Design 123
The Cost of Complexity Management 127
The Efficiency of the Diagrams 128
The Product Costing Code 128

Managing Complexity with Abstraction 137
Polymorphism 137
Total Typed Polymorphism 137
Partial, Untyped Polymorphism 138
ISA and the Liskov Substitution Principle 139
Mathematical vs. Polymorphic Relationships 140
Factoring Instead of Deriving 142
Managing Complexity with Aggregation 145
Restricting Visibility by the use of Friendship 148

Case Study: The Design of a Container Library 149
Anonymous Containers 160
Summary of the Container Case Study 163

Summary 164

Exercises 165

Container Class Listings 166

3

Analysis and Design 189

Introduction 189

Case Study: A Batch Payroll Application 191
Specification 191
Analysis by Noun Lists 192
Analysis by Use-Cases 194
Adding Employees 194
We Are Already Making Design Decisions 196
Deleting Employees 196
Posting Time Cards 197

Posting Sales Receipts 197
Posting a Union Service Charge 198
Changing Employee Details 199
Payday 200
Reflection: What Have We Learned? 201
The Viability of Real-World Models 203
Finding the Underlying Abstractions 204
The Schedule Abstraction 204
Payment Methods 206
Affiliations 206
Transactions 206
Adding Employees 208
Deleting Employees 208
Time Cards, Sales Receipts, and Service Charges 209
Changing Employees 214
Paying Employees 220
Main Program 222
Application Framework 223
The Database 224

Summary of Payroll Design 225

High-Level Closure Using Categories 226
Class Categories 226
Category Structure and Notation 228
Circularity in the Category Structure 229
Resolving Issues of Circularity 230
The Category Structure Is Always Flexible 231

Cohesion, Closure, and Reusability 231
The Cohesion of Common Closure 232
Creating a Hierarchy of Closed Categories 233
The Main Sequence: Plotting Stability vs. Generality 235
The Abstraction vs. Stability Characteristics of Traditional Software Methods 236
The Impact of Abstract Classes on the Main Sequence 238
Categories That Deviate from the Main Sequence 238
The Category Is the Granule of Reuse 239
Cohesion of Policy and Function 240
Reflections on Cohesion 241
Reflections Upon the Payroll Application 241
Coupling and Encapsulation 243
Afferent and Efferent Coupling 243
Controlling Coupling with Encapsulation 244

Metrics 246
Applying the Metrics to the Payroll Application 247

Object Factories 250
Rethinking the Cohesion Boundaries 251
The Object Factory for TransactionImplementation 252
Initializing the Factories 252
The Final Category Structure 254
Reflections upon Object Factories 257
Exercises 257
A Partial C++ Implementation of Payroll 258
Categories and Namespaces 258
The Header Files 259
The PayrollDomain Category 259
The PayrollFactory Category 262
The PayrollImplementation Category 263

4

Paradigm Crossings 271

Introduction 271
The Object-Oriented Paradigm 272
The C++ Interpretation of the OO Paradigm 273
Crossing the Procedural-Paradigm Boundary 273
Wrapping Procedural Programs in OO Classes 274
Wrapping Procedural Servers within Classes 274
"Objectifying" Procedural Clients 279
Managing the PP-to-OO Transition 281
The Representational Paradigm 283
Representational Modeling Is Not Object-Oriented Modeling 284
The OO / Representational Difference 285
The Need for Pure Representations 285
Representational Models and Run-Time Type Identification 286
When Is a Representational Model More Appropriate Than an OO Model? 287
The Representational Model of a State Machine Compiler 287
The SMC Representational Model in C++ 291
Crossing the OO/Representational Boundary (Interpreting the Model) 296
Categories and Representational Models 298
Representational Reflections 299
The Relational Paradigm 299
Tuples and Tables 300
The Normal Forms of a Relation 301

The Relational Representation of the Payroll Application 302
Using a Relational Database 302
A Wide Chasm 305
Crossing the Chasm 305
Surrogation 306
Relational Reflections 307
The Multiprocessing Paradigm 309
Lightweight Processes in an OO Environment 309
Multiprocessing Pollution 313
Heavyweight Processes 313
Surrogation between Heavyweight Processes. 314
Summary 319
Exercises 319

5

High-Level OOAD: A Case Study 321

Introduction 322
Case Study: The Requirements Document 323
Analyzing the Requirements 324
Requirement 1: Doors, Locks, and Security Card Readers 324
Reflections 336
Requirement 2: Break-in Detection 337
Reflections 340
Requirement 3: Fire and Smoke Detectors 341
Reflections 344
Requirement 4: Security-Guard Patrol Tracking 345
Reflections 355
Requirement 5: Multiple Levels of Security 355
Reflections 361
Requirement 6: Lockdown 361
Reflections 368
Requirement 7: Emergency Evacuation 368
Reflections 370
Requirement 8: Security Control Centers 372
Reflections 380
Requirement 9: Events, Violations, and Alarms 380
Summary 383

Exercises 383

6

Physical Architecture 385

Introduction 385

Reviewing the Logical Design 386
- Breaking Dependency Cycles 386
- Breaking Unwanted Dependencies 390
- Accidental Duplication 390
- Cycles Revisited 391
- Reflection 393

SecurityZone Category 394
- Keeping the Dynamic Scenarios Current 398
- Breaking Unwanted Dependencies 399
- Metric Analysis 401
  - *Reflection on Metrics* 403

Splitting Detectors 403

Timers and Patrols 404

Sensors 409

Security Card Reader 410

GUI Control Center 411

Security System and Clearance-Access Policy 412

The Final Category Diagram 414
- Physical Representation 414
- How Reusable Is This 415
- Splitting up the O/S-Specific Category 416
- The Main Category 417
- Metric Analysis 417

Summary 418

Exercises 419