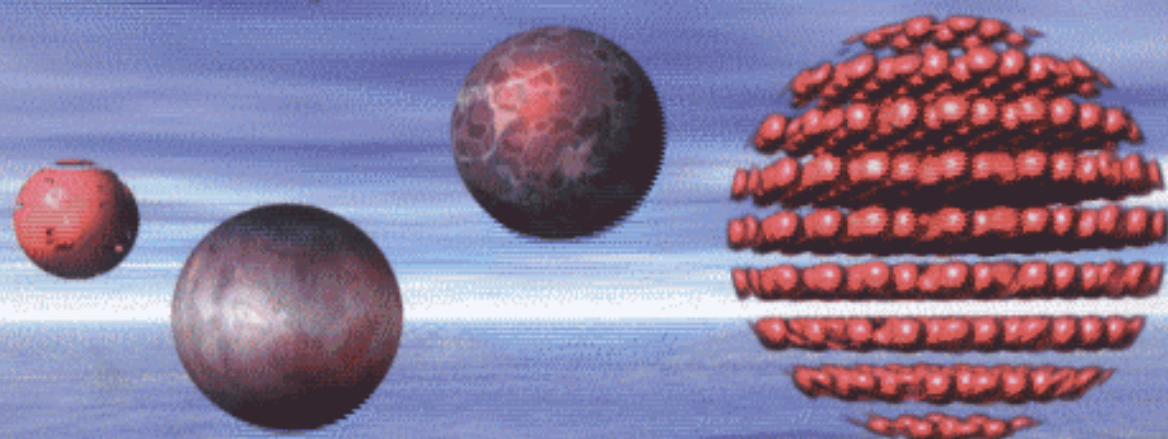


UML *and* C++

A Practical Guide To
Object-Oriented Development

SECOND EDITION



RICHARD C. LEE

WILLIAM M. TEPFENHART

Preface to Second Edition xvii

Preface to First Edition xxvii

1	The Information Management Dilemma	1
	The Problem	1
	Modern Corporations Are Headed Toward Disaster	3
	What Does the Customer Want?	4
	Why Object-Oriented Is Important to Developers	5
	Summary	6
2	Managing Complexity: Analysis and Design	7
	Abstraction Mechanism	9
	Functions	9
	Modules	10
	Abstract Data Types	11
	Classes/Objects	11
	Message Passing	12
	Generalization/Specialization and Polymorphism	12
	Additional Relationships	13
	Associations	13
	Aggregation	15
	Behavior	16
	Static Behavior	16
	Dynamic Behavior	17
	Rules	19
	Complex Systems	20
	Summary	21

3 Object-Oriented Programming 23

What Is Object-Oriented Programming? 24

Not a Silver Bullet 24

An Advanced Paradigm 24

Basic Object-Oriented Programming Concepts 24

Object-Oriented Programming Languages 29

Object-Based Programming 29

Class-Based Programming 29

Object-Oriented Programming 29

Advanced OO Programming 29

Leading-Edge Object-Oriented Programming 30

Why C++? 30

Ways of Organizing Reality 31

Simulation Model of Computation 33

Object-Oriented Way of Organizing Reality 33

Summary 39

4 Bounding the Domain 41

Introduction to Use Cases 42

System 42

Actors 43

Use Cases 44

Use Case Bundles 50

Documenting Use Cases 51

Use Case Diagram 51

Sequence Diagram: Documenting the Details 52

Textual Description 53

Guidelines for Developing Use Cases 54

Avoiding Analysis Paralysis 54

Identifying Actors 55

Identifying High-Level and Essential Use Cases 57

Establishing Use Case Bundles 58

Developing Use Case Details 59

Identifying Supporting Use Cases 61

Developing Boundary Use Cases 62

Contracts 62

Recommended Approach 64

Example 64

Summary 71

5	Finding the Objects	73
	Object-Oriented Analysis: Model of an Application Domain	74
	Building the OO Model	74
	Identification of Objects	75
	Current Techniques	77
	Using the Things to be Modeled	77
	Using the Definitions of Objects and Classes	78
	Using Object Decomposition	78
	Using Generalization	78
	Using Subclasses	79
	Using Object-Oriented Domain Analysis	79
	Reusing an Application Framework	80
	Reusing Class Hierarchies	80
	Reusing Individual Objects and Classes	81
	Using Subassemblies	81
	Using Personal Experience	82
	Traditional Techniques	82
	Using Nouns	82
	Using Traditional Data Flow Diagrams	83
	Using Class-Responsibility-Collaboration (CRC) Cards	84
	Recommended Approaches	86
	Example	88
	Summary	89
6	Identifying Responsibilities	91
	What Is an Object?	91
	What Is an Attribute?	92
	Descriptive Attributes	93
	Naming Attributes	93
	What Is a Service?	93
	What Is a Method?	94
	Identifying Attributes	94
	Specifying Attributes	96
	Identifying Services	97
	Specifying Services	98
	Recommended Approach	99
	Example	100
	Summary	102

7	Specifying Static Behavior	103
	What Is Behavior?	103
	Techniques for Specifying Static Behavior	104
	Techniques for Specifying Control	106
	Techniques for Documenting Control	107
	Activity Diagrams	107
	Collaboration Diagram	108
	Sequence Diagram	109
	Techniques for Documenting Static Behavior	110
	Preconditions and Postconditions	110
	Flowcharting	110
	Data Flow Diagrams	110
	Structured English	111
	Recommended Approach	112
	Example	112
	Summary	116
8	Dynamic Behavior	117
	Introduction	118
	Techniques for Identifying Dynamic Behavior	119
	Common Lifecycle Forms	120
	Models for Capturing Lifecycle	121
	Identifying and Specifying Events	123
	Use Case and Scenario	123
	Sequence Diagram	123
	Example	124
	Specifying Dynamic Behavior	128
	Event List	128
	State Transition Table	130
	Documenting Dynamic Behavior	134
	State Diagrams	134
	Recommended Approach	139
	Summary	140
9	Identifying Relationships	141
	Accessing Another Object's Services	141
	Relationships	142
	Generalization	143

	Identifying and Specifying Generalization/Specialization	145
	Object Aggregation	147
	Classification of Aggregation	148
	Assembly-Parts (Component-Integral Composition)	149
	Material-Object Composition	149
	Portion-Object Composition	150
	Place-Area Composition	151
	Collection-Members Composition	151
	Container-Content (Member-Bunch Composition)	152
	Member-Partnership Composition	152
	Objects and Aggregation Relationships	153
	Links Between Objects	153
	Identifying and Specifying Links and Aggregations	155
	Managing Relationships	156
	Documenting Relationships	158
	Recommended Approach	159
	Example	160
	Summary	161
10	Rules	163
	Introduction	163
	Rules	165
	Identifying Declarative Statements	165
	Specifying and Documenting Rules	166
	Mapping Rules to the Proper OO Concept	168
	Documenting the Rules Using UML	169
	Implementing Rules	170
	Recommended Approach	171
	Summary	171
11	The Model	173
	Concepts	173
	Concepts and Object-Oriented Model	174
	Class	175
	Association	175
	Class Aggregation	176
	Generalization/Specialization	176
	Polymorphism	177
	Instantiation	178

Documenting Concepts Using UML	178
Class Concept	178
Association	179
Class Aggregation	180
Generalization/Specialization	181
Polymorphism	181
Instantiation	181
Refining the Model	182
Subsystems	182
Domain	182
Bridge	183
Organizing Subsystems	184
Horizontal Layers	184
Vertical Partitions	184
Combination	185
Identifying Subsystems	185
Documenting Subsystems	185
Recommended Approach	186
Example	186
Refinement	187
Subsystems	191
Summary	193
12 Design	195
Introduction	195
System Design	196
Subsystems	196
Architectural Frameworks	197
Documenting System Design	200
Detailed Design	201
Class Design	202
Association Design	204
Generalization and Inheritance	205
Delegation	206
Orlando Treaty	207
Multiple Inheritance	208
Documenting Detailed Design	209
Summary	209

13	C++ Fundamentals	211
	History	211
	Programming Elements	212
	Keywords	213
	Identifiers	213
	Literals	214
	Operators	214
	Punctuators	215
	Native Data Types	215
	Basic Data Types	216
	Constant Values	216
	Symbolic Variables	217
	Pointer Types	218
	Constant Types	218
	Reference Types	219
	Enumeration Types	219
	Array Types	220
	Typedef Names	220
	What Is a Statement?	220
	Expressions	221
	Compound Statements	221
	Statement Flow Control	222
	If Statement	222
	For Statement	223
	What Is a Function?	223
	Function Invocation	224
	Function Definition	225
	Function Prototype	226
	Inlining	226
	Storage Class	226
	Auto	226
	Extern	227
	Register	227
	Static	227
	Volatile	228
	Type Conversion	228
	static_cast	228
	const_cast	229
	dynamic_cast	229
	reinterpret_cast	229

Namespace 230
 Recommended Approach 231
 Summary 231

14 Implementing Class 233

Components of a Class 233
 Class Definition 234
 Class Body 235
 Visibility 235
 Data Members 235
 Member Functions 236
 Generalization Using Inheritance 238
 Recommended Approach 238
 Example 239
 Summary 240

15 C++ Libraries 241

C Standard Libraries 242

<cassert> 242
 <cctype> 242
 <cerrno> 242
 <cfloat> 243
 <ciso646> 243
 <climits> 243
 <locale> 243
 <cmath> 243
 <csetjmp> 243
 <csignal> 243
 <cstdarg> 243
 <cstddef> 243
 <cstdio> 244
 <cstdlib> 244
 <cstring> 244
 <ctime> 244
 <wchar> 245
 <wctype> 245

C++ Class Libraries 245

<bits> 245
 <bitstream> 245
 <complex> 245
 <defines> 246

- `<dynarray>` 246
- `<exceptions>` 246
- `<fstream>` 246
- `<iomanip>` 246
- `<ios>` 246
- `<iostream>` 246
- `<istream>` 248
- `<new>` 248
- `<ostream>` 248
- `<ptrdynarray>` 249
- `<sstream>` 249
- `<streambuf>` 249
- `<string>` 249
- `<strstream>` 249
- `<typeinfo>` 250
- `<wstring>` 250

Standard Template Library 250

- `<algorithm>` 251
- `<bitset>` 251
- `<complex>` 251
- `<deque>` 251
- `<functional>` 251
- `<iterator>` 251
- `<list>` 252
- `<map>` 252
- `<memory>` 257
- `<numerics>` 257
- `<queue>` 257
- `<set>` 257
- `<stack>` 260
- `<utility>` 260
- `<valarray>` 260
- `<vector>` 260

Recommended Approach 260

Summary 260

16 Implementing Static Behavior 261

Function Definition 261

- Return Type 262
- Return Statement 263
- Function Argument List 264

Passing Arguments	266
Pass-By-Value	266
Reference or Pointer Argument	267
Return-Type as Reference or Pointer	269
Casting	270
Const and Defaults	271
Const	271
Default Initializers	273
Identifiers Scope	273
Recommended Approach	274
Definition in .h file	274
Definition in .C file	276
Summary	277

17 Implementing Dynamic Behavior 279

Elements of Dynamic Behavior	280
Simple State Diagrams	280
Nested State Diagrams	286
Concurrent State Diagrams	292

Recommended Approach 292

Summary 293

18 Instantiating and Deleting Objects 295

Introduction	295
Constructors	296
Destructors	299
Using Constructors and Destructors Properly	302
Generalization and Constructors	303
Recommended Approach	304
Creating an Object	304
Destroying an Object	304
Coding Guidelines	306
Constructor Coding Guidelines	306
Destructor Coding Guidelines	306

Summary 307

19 Implementing Generalization/Specialization 309

Inheritance	309
Specifying a Derived Class	310
Inheriting from a Derived Class and Implementing Association	312

	Adding Polymorphism	314
	Abstract Class	316
	Multiple Inheritance	318
	Virtual Destructors	323
	Derived Class Visibility	324
	Summary	325
20	Implementing More Relationships	327
	Introduction	327
	Implementing Association	327
	Implementing Attributes of an Association	328
	Implementing Aggregation	329
	Pointers	329
	Arrays	331
	Friends	333
	Static Members	333
	Implementing Association	334
	Binary Association	334
	Many-to-One Association	335
	Many-to-Many Association	337
	Implementing Friends	338
	Class as a Friend	338
	Function as a Friend	339
	Implementing a One-Many Association Using a Friend	340
	Implementing Aggregation	341
	Buried Pointers OO	341
	Embedded Objects	342
	Implementing Static Members	343
	Recommended Approach	345
	Summary	345
21	Introduction to the Case Studies	347
	Case Study 1: Breakout	347
	Requirements	348
	Acquiring Domain Expertise	349
	Expert's Knowledge	350
	Provided Technology Services	351
	Case Study 2: Microwave Oven	353
	Problem Definition	353
	General Description	355

- 22 Case Study: The Breakout Game 357**
- Step 1: Finding the Objects 357
 - Step 2: Identifying Responsibilities 358
 - Looking at Adjectives 358
 - Asking Question 1 359
 - Asking Question 2 366
 - Asking Question 3 370
 - Looking at Services 373
 - Step 3: Specifying Behaviors 376
 - Step 4: Specifying Relationships 386
 - Step 5: Refinement 393
 - Step 6: Design 415
 - Step 7: Implementation 425
 - Implementing Class 425
 - Implementing Static Behavior 439
 - Instantiating Objects 442
 - Implementing Inheritance 446
 - Implementing Relationships 465
- 23 Case Study: Microwave Oven 475**
- Use Cases 476
 - Use Case 1: Cook Without Interruption 476
 - Use Case 2: Cancel Cooking 478
 - Use Case 3: Interrupt Cooking 479
 - Solution 1: The Controller Class Design 480
 - Step 1: Finding the Objects 480
 - Step 2: Identifying Responsibilities 481
 - Step 3: Specifying Behavior 484
 - Step 4: Specifying Relationships 488
 - Step 5: Refinement 489
 - Discussion 489
 - Solution 2: Distributed Responsibility with High Coupling 489
 - Step 1: Identify the Objects 489
 - Step 2: Identifying Responsibilities 489
 - Step 3: Specifying Behavior 491
 - Step 4: Specifying Relationships 495
 - Step 5: Refinement 495
 - Discussion 495

CONTENTS

Solution 3: Distributed Responsibility Using the Observer Mechanism 496

Step 5: Refinement 497

Discussion 509

APPENDIX A Unified Modeling Language 511

Introduction 511

What Is the Unified Modeling Language? 511

What Is Not UML? 512

What Are the Goals of UML? 512

Why Use UML? 512

What Are the Diagrams of UML? 512

What Are the Most Important UML Diagrams? 513

UML Diagrams 514

Use Case Diagram 514

Class Diagram 515

Sequence Diagram 526

Collaboration Diagram 527

Statechart Diagram 529

Activity Diagram 532

Component Diagram 534

Deployment Diagram 535

UML Glossary 536

Bibliography 541

Acknowledgments 545