
Contents

Part 1 The Unified Process

1	Introduction	3
1.1	Introduction	3
1.2	Why UML and the Unified Process?	5
1.3	Why This Book?	6
1.4	Where to Get More Information	6
1.5	Where to Go Online	7
2	Object-Oriented Analysis and Design	9
2.1	Introduction	9
2.2	Object-Oriented Design Methods	9
2.3	Object-Oriented Analysis	10
2.3.1	Class Responsibility Collaborator (CRC)	10
2.4	The Booch Method	12
2.4.1	The Steps in the Booch Method	12
2.4.2	Strengths and Weaknesses	13
2.5	The Object Modeling Technique	13
2.5.1	The Analysis Phase	14
2.5.2	The Design Phase	14
2.5.3	The Implementation Phase	15
2.5.4	Strengths and Weaknesses	15
2.6	The Objectory Method	15
2.6.1	The Requirements Phase	15
2.6.2	The Analysis Phase	16
2.6.3	The Construction Phase	16
2.6.4	Strengths and Weaknesses	16
2.7	The Fusion Method	16
2.8	The Unified Modeling Language	17
2.9	Summary	18
2.10	References	18

3	An Introduction to the UML and the Unified Process	21
3.1	Introduction	21
3.2	Unified Modeling Language	21
3.2.1	History of the UML	22
3.2.2	Introduction to the UML	24
3.2.3	Models and Diagrams	25
3.3	Analysis of the UML	26
3.4	The Unified Process	26
3.4.1	Overview of the Unified Process	27
3.4.2	Life Cycle Phases	31
3.4.3	Phases, Iterations and Disciplines	32
3.4.4	Disciplines and Activities	35
3.4.5	Applying the Unified Process	35
3.5	The Rational Unified Process	36
3.6	Summary	37
3.7	References	37
3.8	Online References	37
4	Software Architecture and Object-Oriented Design	39
4.1	Software Architecture – the Very Idea	39
4.1.1	Why Have an Architecture?	39
4.1.2	Why We Need an Architecture	41
4.1.3	Architecture Myths	41
4.1.4	Architecture Defined	44
4.1.5	Characteristics of a Good Architecture	45
4.1.6	Layering the Architecture	46
4.1.7	Use Cases and Architecture	46
4.1.8	So What Is an Architecture?	47
4.2	Software Patterns	49
4.2.1	What Are Design Patterns?	49
4.2.2	What They Are Not	49
4.2.3	Architectural Patterns	50
4.3	Constructing the Architecture	50
4.4	Find Architecturally Significant Use Cases	51
4.4.1	Architecturally Significant Use Cases	51
4.4.2	Use Case Description	51
4.5	Identify Key Classes	51
4.6	Breaking the System into Subsystems	52
4.6.1	Breaking the System into Subsystems	53
4.6.2	Identifying Subsystems	54
4.6.3	Assessing the Subsystems	54
4.6.4	Identify Major System Interfaces	55
4.6.5	Layering the Subsystems	55
4.7	Identifying Concurrency and Active Classes	55
4.7.1	Allocating Subsystems to Processors and Tasks	57
4.7.2	Deployment Diagrams	57

4.8	Managing Data Stores	58
4.8.1	Handling Access to Global Resources	59
4.9	Additional Architectural Concerns	59
4.9.1	Choosing the Implementation of Control in Software	59
4.9.2	Identify Generic Design Mechanisms	60
4.9.3	Handling Boundary Conditions	60
4.9.4	Setting Trade-offs Between Competing Resources	60
4.9.5	Specifying Default Policies for the Object Design	61
4.10	Plan Incremental Build of Software	61
4.11	The Online ATM Architecture Design	61
4.11.1	Identifying Architecturally Significant Use Cases	62
4.11.2	Organizing the System into Subsystems	62
4.11.3	Identify Key Classes	62
4.11.4	Identifying Concurrency	62
4.11.5	Allocating Subsystems to Processors	62
4.11.6	Managing Data Stores	62
4.11.7	Handling Access to Global Resources	64
4.11.8	Choosing the Implementation of Control	64
4.11.9	Boundary Conditions	64
4.11.10	Default Policies for the Object Design	64
4.11.11	Implement a Skeleton Architecture	65
4.12	References	65
5	Requirements Discipline: Use Case Analysis	67
5.1	Introduction	67
5.2	Requirements Discipline	67
5.3	Use Case Analysis	68
5.4	The Use Case Model	68
5.5	Use Case Diagrams	69
5.6	Actors	70
5.6.1	Steps in Finding Actors	72
5.7	Use Cases	72
5.7.1	Identifying Use Cases	74
5.7.2	Identifying Use Case Events	75
5.8	Refining Use Case Models	75
5.9	Additional Documents	76
5.10	Interface Descriptions	76
5.11	Online ATM Use Case Analysis	76
5.11.1	Deposit Use Case	78
5.11.2	Withdrawal Use Case	78
5.11.3	Interface Descriptions	78
5.12	Structuring the Use Case Model	81
5.13	Are Use Case Diagrams Useful?	82
5.14	Further Reading	85
5.15	References	85

6	The Analysis Discipline: Finding the Entities	87
6.1	Introduction	87
6.2	Analysis Discipline Activities	89
6.3	The Analysis Model	89
6.3.1	Why Have an Analysis Model?	90
6.3.2	Analysis Model Classes	90
6.3.3	Use Case Realizations	92
6.3.4	Constructing the Analysis Model	94
6.4	Generating Analysis Classes	94
6.4.1	Representing Classes	94
6.4.2	Representing Objects	95
6.4.3	Generating Objects and Classes	96
6.4.4	Identifying Classes for the Online ATM System	98
6.4.5	Rationalizing Classes	98
6.5	Generating Use Case Realizations	100
6.6	Identifying Attributes	100
6.6.1	Identifying Attributes of Objects	101
6.6.2	Identifying Attributes in the Online ATM System	102
6.7	Preparing a Data Dictionary	103
6.7.1	The Online ATM Data Dictionary	103
6.8	Identifying Associations	103
6.8.1	Representing Associations	103
6.8.2	Identifying Associations Between Objects	104
6.8.3	Identifying Associations in the Online ATM System	105
6.9	Identifying Inheritance	106
6.9.1	Representing Inheritance	106
6.9.2	Organizing and Simplifying Analysis Classes Using Inheritance	108
6.9.3	Identifying Inheritance in the Online ATM System	109
6.10	Grouping Analysis Classes into Packages	109
6.10.1	Identifying Analysis Packages	110
6.10.2	Representing Packages	111
6.10.3	Analyzing Analysis Packages	112
6.11	Iterating and Refining the Model	112
6.12	Identify Common Special Requirements	113
7	The Design Discipline: System and Class Design	115
7.1	Introduction	115
7.2	Design Discipline Activities	115
7.3	Class Design Stage	117
7.4	The Design Model	117
7.5	Design Classes	119
7.5.1	Design Class Notation	119
7.6	Identifying and Refining Design Classes	123
7.6.1	Identifying Classes	123
7.6.2	Refining the Set of Classes	124

7.6.3	Identifying Attributes for Design Classes	125
7.6.4	Refining Attributes for Design Classes	125
7.6.5	Representing Operations	126
7.6.6	Describing Operations	126
7.6.7	Identifying Operations	127
7.6.8	Refining Operations	129
7.7	Identifying Operations for the Online ATM System	129
7.7.1	Operations Implied by Events	129
7.7.2	Operations Implied by State Actions and Activities	129
7.7.3	Application or Domain Operations	129
7.7.4	Simplifying Operations	130
7.7.5	The OBA Operations	130
7.8	Analyzing Use Cases	131
7.8.1	Generating Design Classes from Use Cases	131
7.8.2	Design Use Case Realizations	132
7.9	Identifying Dynamic Behaviour	132
7.9.1	Design Collaboration Diagrams	133
7.9.2	Sequence Diagrams	135
7.9.3	Preparation of Sequence Diagrams	135
7.9.4	Dealing with Complexity in Sequence Diagrams	137
7.9.5	Generating a Sequence Diagram	137
7.9.6	Sequence Diagrams for the Online Bank Account System	138
7.9.7	Describing an Object's Behaviour	139
7.10	Statechart Diagrams	139
7.10.1	Start Points	139
7.10.2	Events	140
7.10.3	A Set of Transitions	140
7.10.4	A Set of State Variables	141
7.10.5	A Set of States	141
7.10.6	A Set of Exit Points	142
7.10.7	Building a Statechart Diagram	142
7.11	Associations	146
7.11.1	Representing Associations	146
7.11.2	Identifying Associations	148
7.11.3	Refining Associations	149
7.12	Identifying Interfaces	150
7.13	Identifying Inheritance	151
7.14	Remaining Steps	151
7.14.1	Optimizing the Design	151
7.14.2	Testing Access Paths	151
7.14.3	Implementing Control	152
7.14.4	Adjusting Class Structure	152
7.14.5	Designing Associations	152
7.14.6	Object Representation	152
7.15	Applying the Remaining Steps to OBA	153
7.15.1	Optimizing the Design	153

7.15.2	Implementing Control	153
7.15.3	Adjusting the Class Structure	153
7.15.4	Designing Associations	153
7.15.5	Determining Object Representation	153
7.16	Iterating and Refining the Model	153
7.17	References	154
8	Implementation Phase	155
8.1	Introduction	155
8.2	Implementation Discipline Artefacts	155
8.3	Implementation Discipline Activities	156
8.3.1	Implementing the Skeleton Architecture	156
8.3.2	Define the Implementation Model	157
8.3.3	Implement the System	159
8.3.4	Refining the Implementation Model	160
8.3.5	Integrate the Systems	161
9	The Test Discipline: How It Relates to Use Cases	163
9.1	Introduction	163
9.2	The Purpose of the Discipline	163
9.3	Aims of Discipline	163
9.4	Test Discipline Activities	164
9.4.1	Plan Tests	164
9.4.2	Design Tests	164
9.4.3	Implement Tests	165
9.4.4	Perform Integration Tests	165
9.4.5	Perform System Tests	165
9.4.6	Evaluate Tests	166
9.5	Summary	166
9.6	Reference	166
10	The Four Phases	167
10.1	Introduction	167
10.2	The Unified Process Structure	167
10.3	Relationship Between Phases and Iterations	168
10.3.1	The Four Phases	168
10.4	Effort Versus Phases	171
10.5	Phases and Iterations	172
10.6	Phases and Cycles	173
11	The JDSync Case Study	175
11.1	Introduction	175
11.2	Problem Statement	175
11.3	The Requirements Discipline: Use Case Analysis	175
11.3.1	Actors in JDSync	176
11.3.2	Use Cases	177

11.3.3	The Use Case Diagram	180
11.3.4	Example User Interfaces	181
11.4	The Analysis Discipline	181
11.4.1	Identifying the Analysis Classes	182
11.4.2	Generating the Collaboration Diagrams	184
11.5	The Design Discipline	188
11.5.1	Initial Identification of Classes	188
11.5.2	Sequence Diagrams	190
11.5.3	Identifying Attributes	196
11.5.4	Identifying Operations	197
11.5.5	Describing the Overall Behaviour of an Object	199
11.5.6	Identifying Associations and Inheritance	200
11.5.7	Identifying Interfaces	201
11.5.8	A Complete Design Model	201
11.6	The Implementation Workflow	201
11.6.1	Updating the Class Structure	201
11.7	Summary	206

Part 2 Design Patterns

12	Software Patterns	209
12.1	Introduction	209
12.2	The Motivation Behind Patterns	210
12.3	Documenting Patterns	211
12.4	When to Use Patterns	212
12.5	Strengths and Limitations of Design Patterns	212
12.6	An Example Pattern: Mediator	213
12.7	Summary	218
12.8	Further Reading	218
12.9	References	218
13	Patterns Catalogs	221
13.1	Introduction	221
13.2	GoF Patterns	221
13.3	Creational Patterns	222
13.3.1	Factory Method	222
13.3.2	Abstract Factory	223
13.3.3	Builder	223
13.3.4	Prototype	223
13.3.5	Singleton	223
13.4	Structural Patterns	225
13.4.1	Adapter	225
13.4.2	Bridge	225
13.4.3	Composite	226
13.4.4	Decorator	226
13.4.5	Façade	226

13.4.6	Flyweight	227
13.4.7	Proxy	228
13.5	Behavioural Patterns	228
13.5.1	Chain of Responsibility	228
13.5.2	Command	229
13.5.3	Interpreter	229
13.5.4	Iterator	229
13.5.5	Mediator	229
13.5.6	Memento	231
13.5.7	Observer	231
13.5.8	State	231
13.5.9	Strategy	231
13.5.10	Template Method	232
13.5.11	Visitor	232
13.6	Summary	232
13.7	References	233
14	Applying the Model–View–Controller Pattern	235
14.1	Introduction	235
14.2	What Is the Model–View–Controller Architecture?	235
14.3	What Java Facilities Support the MVC	236
14.3.1	The Delegation Event Model	237
14.4	The MVC in Java	238
14.5	A Simple Calculator Application	240
14.5.1	Swing Component Event Handling	242
14.5.2	Frames, Panels and Layout Managers	243
14.5.3	The Application Code	243
14.6	Discussion	243
14.7	References	244
14.8	Listings	244
15	The Hierarchical MVC	253
15.1	Introduction	253
15.2	Why Isn't This Enough?	253
15.3	The h-MVC	254
15.4	The h-MVC Details	254
15.5	Layered Application	254
15.6	Initialization	257
15.7	Hierarchical Behaviour	259
15.8	The Advantages of the h-MVC	260
15.9	The Disadvantages of the h-MVC	260
15.10	Summary	261
16	The Visitor Framework	263
16.1	Background	263
16.2	The Visitor Pattern	264

16.3	The Visitor Framework	266
16.4	Using the Visitor Framework	266
16.5	A Simple Application	269
16.6	Summary	271
16.7	References	271
16.8	Listings	272
17	The EventManager	281
17.1	Introduction	281
17.2	The Use of Patterns	281
17.3	The Mediator Pattern	283
17.4	The Singleton Pattern	283
17.5	The Design of the EventManager	283
17.5.1	The EventManager	284
17.5.2	The ManagedEvent Class	285
17.5.3	The EventManagerListener Interface	285
17.6	Using the EventManager	286
17.7	The EventManager in a Graphical Client	286
17.8	Reference	286
17.9	Listings	286
18	J2EE Patterns	293
18.1	Introduction	293
18.2	What Are J2EE Design Patterns?	293
18.3	A Catalog of J2EE Patterns	294
18.4	The FrontController Pattern	295
18.4.1	Context	295
18.4.2	Problem	295
18.4.3	Forces	295
18.4.4	Solution	296
18.4.5	Strategies	297
18.4.6	Consequences	297
18.4.7	Related Patterns	297
18.5	The Request–Event–Dispatcher Pattern	298
18.5.1	Context	298
18.5.2	Problem	298
18.5.3	Forces	298
18.5.4	Solution	298
18.5.5	Strategies	300
18.5.6	Consequences	302
18.5.7	Related Patterns	302
18.6	J2EE-based Model–View–Controller	303
18.6.1	Context	303
18.6.2	Problem	303
18.6.3	Forces	303
18.6.4	Solution	303

18.6.5	Strategies	305
18.6.6	Consequences	306
18.6.7	Related Patterns	306
18.7	Summary	307
18.8	Further Reading	307
18.9	References	307
19	The Fault Tracker J2EE Case Study	309
19.1	Introduction	309
19.2	The Fault Tracker Application	309
19.2.1	Requests for Change	310
19.2.2	Problem Reporting	311
19.3	Using the Fault Tracker	313
19.4	The Design of the Fault Tracker	317
19.4.1	What Is the Architecture?	317
19.5	Summary and Conclusions	324

Part 3 The Unified Process in the Real World

20	Are UML Designs Language-Independent?	329
20.1	Introduction	329
20.2	OOD Is Language-Independent – Right?	329
20.3	Making UML Work for You	330
20.4	Questions to Consider	331
20.5	The Java Platform	331
20.6	Classes in the UML	332
20.7	Fields in the UML	332
20.8	Operations in the UML	333
20.9	Constructors	333
20.10	Packages in the UML	334
20.11	UML Interfaces	336
20.12	Templates	336
20.13	Associations	337
20.14	Multiplicity in the UML	339
20.15	Aggregation and Composition	339
20.16	Singleton Objects	340
20.17	Synchronous and Asynchronous Messages	340
20.18	From Code to the UML	341
20.19	Conclusions	342
21	Customizing the Unified Process for Short Time-Scale Projects	343
21.1	Introduction	343
21.2	Particular Problems of Small Projects	344
21.3	The Unified Process as a Framework	345
21.3.1	Experience of Application Domain	346
21.3.2	Experience with Technology	346

21.3.3	Understanding of Requirements	347
21.3.4	The Size of the Task	348
21.3.5	The Nature of the Task	348
21.3.6	Time/Budget/Resources	349
21.3.7	Management Support/Buy-in	350
21.4	Adapting the Unified Process for a Small Project	351
21.4.1	A Typical Short-Term Project	351
21.4.2	Short Time-Scale Project Approach	351
21.5	The Modified Unified Process	352
21.6	Summary	354
21.7	Reference	354
22	Augmenting the Unified Process with Additional Techniques	355
22.1	Introduction	355
22.2	The Unified Process as a Framework	355
22.3	Class Identification	357
22.4	CRC: Class–Responsibility–Collaboration	358
22.5	What Is CRC?	359
22.5.1	The Basic Idea	359
22.5.2	Identifying Classes	360
22.5.3	Identifying Responsibilities	360
22.5.4	Determining Collaborations	360
22.6	Summary	361
22.7	References	361
23	Inheritance Considered Harmful!	363
23.1	Introduction	363
23.2	Inheritance	364
23.2.1	What Is Inheritance?	364
23.2.2	Aims and Benefits of Inheritance	365
23.2.3	The Role of Inheritance	366
23.3	Drawbacks of Inheritance	366
23.3.1	Reduces Comprehensibility of Code	366
23.3.2	Makes Maintenance Harder	369
23.3.3	Makes Further Development Harder	369
23.3.4	Reduces Reliability of Code	371
23.3.5	May Reduce Reuse	372
23.3.6	The Semantically Fragile Base Class Problem	372
23.3.7	Access Modifier Effects on Reuse	372
23.4	Balancing Inheritance and Reuse	374
23.4.1	Categories of Extension	374
23.4.2	Implications for Inheritance	375
23.5	Compositional Reuse	376
23.5.1	Strengths of Compositional Reuse	376
23.5.2	Weaknesses of Compositional Reuse	377
23.6	Promoting Reuse in Object-Oriented Systems	377

23.7	Tool Support	379
23.8	Conclusions	380
23.9	References	380
24	Incremental Software	383
24.1	The Incremental Software Development Process	383
24.2	Incremental Software Development	384
24.2.1	Iterative Versus Waterfall	384
24.2.2	A Spiral Development Process	384
24.2.3	Architecture-Centric	385
24.2.4	Getting Control	386
24.3	Feature-Centric Development	386
24.4	Timeboxing Iterations	387
24.5	Being Adaptive but Managed	387
24.5.1	Planning an Iterative Project	388
24.5.2	Planning an Iteration	388
24.6	Architecture-Centric	390
24.6.1	Why Have an Architecture?	390
24.6.2	Plan Incremental Building of Software	391
24.7	Performance Measurements and Reporting	392
24.7.1	Weekly Progress Meetings	392
24.7.2	Monthly Meetings	392
24.7.3	Progress Reporting	393
24.7.4	Process for Managing Change	393
24.7.5	Sample Timesheets	394
24.8	References	394
25	Agile Modeling	395
25.1	Introduction	395
25.2	Modelling Misconceptions	396
25.3	The Manifesto for Agile Modeling	399
25.4	Agile Modeling	401
25.5	Agile Modeling and the Unified Process	406
25.6	Agile Modelling and Documentation	408
25.7	Tool Misconceptions	409
25.8	Summary	410
25.9	References	410
25.10	Online References	410
Appendix A	UML Notation	411
Index		419

Guide to the Unified Process featuring UML, Java and
Design Patterns

Hunt, J.

2003, XVIII, 424 p., Hardcover

ISBN: 978-1-85233-721-6