《基元设计模式（英文版）》

图书基本信息

书名：《基元设计模式（英文版）》

13位ISBN编号：9787121211911

10位ISBN编号：7121211912

出版时间：2013-10

出版社：电子工业出版社

作者：【美】史密斯（Smith,J.M.）

页数：364

版权说明：本站所提供下载的PDF图书仅提供预览和简介以及在线试读，请支持正版图书。

更多资源请访问：www.tushu111.com

《基元设计模式（英文版）》

内容概要

即使是经验丰富的软件专业人士，也会发现要为其企业找到能带来实质价值的模式应用方式殊非易事。本书首次以全面的方法论介绍基元设计模式，给出标准的命名和描述，阐述它们的重要性，帮助人们比较和选用，充分利用模式的真正力量，将它们转化成实际的、更加简洁直接的软件实现，并得到非常不错的效果。
对于开发工程师、设计师、架构师和分析师，本书都能提供有价值的指导，帮助他们在大多数语言、环境和问题领域使用模式。

《基元设计模式（英文版）》

作者简介

Jason McC. Smith，2005毕业于北卡罗莱纳州立大学教堂山分校，获博士学位。该校也是基元设计模式的诞生地，当时是作为模式查询和识别系统（System for Pattern Query and Recognition，SPQR）项目的组成部分。Smith博士因其在校的研究项目而荣获两项美国国家专利，一项与SPQR所采用的技术相关，另一项则来自FaceTop分布式文档协作系统。
此前，Smith博士在物理模拟工程和咨询业界工作过多年，取得了华盛顿州立大学的物理学和数学学士学位。值得一提的项目包括声纳和海洋环境模拟、电子工程模拟、商用和军用飞机飞行模拟，以及实时图形训练系统等。
在IBM沃森研究中心的四年时间内，Smith博士得到了一个机会，将从SPQR和EDP目录中获得的经验加以组织，并应用到大量的软件实体中去，包括遗留系统和现代系统。
Smith博士现在供职于华盛顿州柯克兰市的The Software Revolution公司，任资深研究科学家。在那里，他持续地精化EDP目录，并寻找各种方法来强化公司在自动控制现代化以及遗留系统改造方面的业务目标。

《基元设计模式（英文版）》

书籍目录

Figures xi
Tables xv
Listings xvii
Foreword xix
Preface xxi
Acknowledgments xxiii
About the Author xxv
1 Introduction to Design Patterns 1
1.1 Tribal Musings 5
1.2 Art or Science? 9
1.2.1 Viewing Patterns as Rote 9
1.2.2 Language-Dependent Views 10
1.2.3 From Myth to Science 12
2 Elemental Design Patterns 13
2.1 Background 14
2.2 The Where, the Why, the How 17
2.2.1 Decomposition of Decorator 18
2.2.2 Down the Rabbit Hole 21
2.2.3 Context 30
2.2.4 The Design Space 33
2.3 Core EDPs 42
2.4 Conclusion 44
3 Pattern Instance Notation 45
3.1 Basics 45
3.2 The PINbox 49
3.2.1 Collapsed PINbox 49
3.2.2 Standard PINbox 51
3.2.3 Expanded PINbox 55
3.2.4 Stacked PINboxes and Multiplicity 56
3.2.5 Peeling and Coalescing 62
3.3 Conclusion 65
4 Working with EDPs 67
4.1 Composition of Patterns 68
4.1.1 Isotopes 72
4.2 Recreating Decorator 77
4.3 Refactoring 91
4.4 The Big Picture 101
4.5 Why You May Want to Read the Appendix 105
4.6 Advanced Topics 108
4.6.1 Focused Documentation and Training 108
4.6.2 Metrics 109
4.6.3 Procedural Analysis 112
4.7 Conclusion 112
5 EDP Catalog 115
Create Object 117
Retrieve 126
Inheritance 130

Abstract Interface 140
Delegation 145
Redirection 151
Conglomeration 159
Recursion 165
Revert Method 172
Extend Method 181
Delegated Conglomeration 187
Redirected Recursion 193
Trusted Delegation 200
Trusted Redirection 209
Deputized Delegation 216
Deputized Redirection 222
6 Intermediate Pattern Compositions 229
Fulfill Method 231
Retrieve New 235
Retrieve Shared 240
Objectifier 244
Object Recursion 251
7 Gang of Four Pattern Compositions 259
7.1 Creational Patterns 260
7.1.1 Abstract Factory 260
7.1.2 Factory Method 263
7.2 Structural Patterns 265
7.2.1 Decorator 265
7.2.2 Proxy 269
7.3 Behavioral Patterns 273
7.3.1 Chain of Responsibility 273
7.3.2 Template Method 275
7.4 Conclusion 279
A ρ -Calculus 281
A.1 Reliance Operators 282
A.2 Transitivity and Isotopes 285
A.3 Similarity 286
A.4 EDP Formalisms 287
A.5 Composition and Reduction Rules 291
A.6 Pattern Instance Notation and Roles 293
A.7 EDP Definitions 295
A.7.1 Create Object 295
A.7.2 Retrieve 296
A.7.3 Inheritance 298
A.7.4 Abstract Interface 298
A.7.5 Delegation 299
A.7.6 Redirection 300
A.7.7 Conglomeration 300
A.7.8 Recursion 301
A.7.9 Revert Method 301
A.7.10 Extend Method 302
A.7.11 Delegated Conglomeration 303

A.7.12 Redirected Recursion 303
A.7.13 Trusted Delegation 304
A.7.14 Trusted Redirection 305
A.7.15 Deputized Delegation 306
A.7.16 Deputized Redirection 307
A.8 Intermediate Pattern Definitions 308
A.8.1 Fulfill Method 308
A.8.2 Retrieve New 309
A.8.3 Retrieve Shared 310
A.8.4 Objectifier 311
A.8.5 Object Recursion 312
A.9 Gang of Four Pattern Definitions 313
A.9.1 Abstract Factory 313
A.9.2 Factory Method 314
A.9.3 Decorator 316
A.9.4 Proxy 317
A.9.5 Chain of Responsibility 318
A.9.6 Template Method 319
Bibliography 321
Index 325
Figures
2.1 Decorator ’ s usual example UML. 19
2.2 Objectifier as UML. 20
2.3 Object Recursion as UML. 20
2.4 A simple method call as UML. 23
2.5 The parts of a method call. 31
2.6 A simple design space. 34
2.7 A simple design space with EDPs. 35
2.8 Our first four EDPs. 39
2.9 The design space extended to three dimensions. 39
2.10 The design space with method similarity fixed to similar. 40
2.11 Recursion Example UML. 42
2.12 Deputized Redirection example UML. 42
3.1 UML collaboration diagram. 47
3.2 Strategy as pattern:role tags in UML. 48
3.3 Huge UML of a not-so-huge system. 48
3.4 Multiple instances of Strategy as pattern:role tags in UML. 49
3.5 Collapsed PINbox. 50
3.6 Collapsed PINbox as annotation. 50
3.7 Singleton and Abstract Factory in class diagram. 50
3.8 Template Method in sequence diagram. 51
3.9 Standard PINbox. 51
3.10 PIN used with UML class diagram. 52
3.11 PIN used with UML sequence diagram. 53
3.12 Standard PIN role connections. 54
3.13 Blank expanded PIN instance. 55
3.14 Expanded PIN instance. 56
3.15 Expanded PIN instance using UML. 57
3.16 A need for multiple related PINboxes. 59

3.17 Stacked PINbox. 60
3.18 Multiple Strategy instances as PINboxes. 61
3.19 Showing the interaction between multiple Strategy PINboxes. 62
3.20 Abstract Factory as part of a larger UML diagram. 63
3.21 Abstract Factory subsumed within the expanded PINbox. 64
3.22 Coalesced PINbox. 65
4.1 Abstract Interface and Inheritance EDPs as UML. 68
4.2 Internal definition of Fulfill Method as UML. 69
4.3 Fulfill Method as simple connected PINboxes. 69
4.4 Fulfill Method as expanded PINbox. 69
4.5 Fulfill Method as standard PINbox. 70
4.6 Flipping our EDPs in Fulfill Method—oops. 71
4.7 Flipped EDPs as PINboxes. 72
4.8 Alternative classes that can fulfill an Abstract Interface EDP. 75
4.9 Alternative structures that can fulfill an Inheritance EDP. 76
4.10 Decorator ’ s usual example UML. 78
4.11 Fulfill Method definition as annotated UML. 79
4.12 Objectifier UML annotated with PIN. 80
4.13 Objectifier and Trusted Redirection. 81
4.14 Object Recursion annotated with PIN. 82
4.15 Object Recursion as just PIN. 83
4.16 Object Recursion and Extend Method. 84
4.17 Decorator annotated with PIN. 85
4.18 Decorator as PIN. 86
4.19 Decorator instance as a PINbox. 86
4.20 Expanding Decorator: one level. 87
4.21 Expanding Decorator: two levels. 88
4.22 Expanding Decorator: three levels. 89
4.23 Expanding Decorator: four levels. 90
4.24 Delegation before Rename Method refactoring. 93
4.25 Delegation after Rename Method refactoring—Redirection. 94
4.26 Delegation before Move Method refactoring. 95
4.27 The design space with method similarity fixed to dissimilar. 96
4.28 Delegation after Move Method refactoring: boring case. 97
4.29 Delegation after Move Method refactoring: into same type. 97
4.30 Delegation after Move Method refactoring: Delegated Conglomeration. . 97
4.31 Delegation after Move Method refactoring: Conglomeration. 98
4.32 Delegation after Move Method refactoring: Trusted Delegation. 98
4.33 Delegation after Move Method refactoring: Revert Method. 99
4.34 Delegation after Move Method refactoring: Deputized Delegation. 99
4.35 Summarizing refactoring effects so far. 100
4.36 Implicit used-by relationships among the EDPs and selected other patterns. 102
4.37 The full method-call EDP design space: dissimilar method. 103
4.38 The full method-call EDP design space: similar method. 104
4.39 Method-call EDP refactoring relations. 106
5.1 Polymorphic approach 173
5.2 Subclassing approach 175
5.3 UI class cluster showing an instance of Trusted Delegation. 203
5.4 UI class cluster showing an instance of Trusted Redirection. 211

5.5 UI class cluster showing an instance of Deputized Delegation. 218
5.6 UI class cluster showing an instance of Deputized Redirection. 225
7.1 Abstract Factory subsumed within the expanded PINbox. 261
7.2 Reducing the diagram to just one instance of Abstract Factory. 262
7.3 Simplifying Figure 7.2. 264
7.4 Abstract Factory as PIN only. 265
7.5 Factory Method subsumed within the expanded PINbox. 266
7.6 Factory Method as PIN only. 267
7.7 Decorator subsumed with the expanded PINbox. 268
7.8 Decorator as PIN only. 269
7.9 Decorator expanded three levels deep and flattened. 270
7.10 Proxy subsumed with the expanded PINbox. 271
7.11 Proxy as PIN only. 272
7.12 Proxy PIN reorganized to better match Decorator. 272
7.13 Chain of Responsibility subsumed within the expanded PINbox. 274
7.14 Chain of Responsibility as PIN only. 275
7.15 Template Method subsumed within the expanded PINbox. 276
7.16 Template Method reduced to a single instance. 277
7.17 Template Method as PIN only. 278
7.18 Template Method PIN reorganized to better match Decorator. 279
7.19 Factory Method redefined using Template Method. 279
A.1 The full method call EDP design space: similar method 288
A.2 The full method call EDP design space: dissimilar method 289
A.3 Standard PINbox 294
A.4 Expanded PIN instance 294
Tables
2.1 Pattern pieces sorted into three categories of a pattern definition 22
2.2 All interactions between entities of object-oriented programming 28
2.3 Nonscoping interactions between entities of object-oriented programming 28
A.1 All interactions between entities of object-oriented programming 283
A.2 Nonscoping interactions between entities of object-oriented
programming 284
Listings
2.1 A simple method call as pseudocode. 24
2.2 Fields within classes, instances, and namespaces, as defined and used in C++. 26
2.3 A Java class, and one possible equivalent object and type. 27
2.4 Typing as context. 29
2.5 A method call chain as pseudocode. 30
2.6 Simple method call for Figure 2.5. 31
2.7 Example of a Recursion method call in Java. 36
2.8 Example of a Delegation method call in C++. 36
2.9 Example of a Redirection method call in Objective-C. 37
2.10 Example of a Conglomeration method call in Java. 38
5.1 Uninitialized data. 118
5.2 Fixed default values. 120
5.3 Dynamic initialization 121
5.4 Create Object Implementation. 125
5.5 Retrieve with an update. 126
5.6 Retrieve in a temporary variable. 127

5.7 Basic inheritance example in Objective-C. 131
5.8 Overriding an implementation. 132
5.9 Implementation assumption mismatch. 133
5.10 Obvious fix—but likely not feasible. 134
5.11 Fixing a bug while leaving old code in place. 134
5.12 Using Redirection to hide part of an interface. 137
5.13 Animals almost all move but in very different ways. 141
5.14 CEO delegates out responsibilities. 145
5.15 Tom paints the fence with help. 152
5.16 Prep work and cleanup are important. 153
5.17 Prep work and cleanup are decomposable. 160
5.18 Instance swapping for protocol fallback in C++. 173
5.19 Auto fallback/forward using Revert Method. 176
5.20 Using Redirection in Python to add behavior. 182
5.21 Using Extend Method to add behavior. 182
5.22 Inviting friends naively in Java. 187
5.23 A slightly better approach for inviting friends. 188
5.24 Delegated Conglomeration in Java. 188
5.25 Traditional iteration and invocation in C. 193
5.26 Object-oriented iteration and invocation in C++. 193
5.27 Basic Redirected Recursion in C++. 194
5.28 Paratroopers implementing Redirected Recursion. 195
5.29 UI widgets demonstrating Trusted Delegation in C++. 201
5.30 Event handler in C++ showing Trusted Redirection. 210
5.31 UI widgets demonstrating Deputized Delegation in C++. 216
5.32 UI widgets demonstrating Deputized Redirection in C++. 222
6.1 Conditionals to select behavior. 246
6.2 Using Objectifier to select behavior. 247
A.1 Simple code example 287

《基元设计模式（英文版）》

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:www.tushu111.com