第一图书网, tushu007.com

<<敏捷软件开发>>

图书基本信息

书名：<<敏捷软件开发>>

13位ISBN编号：9787115294685

10位ISBN编号：7115294682

出版时间：2013-1

出版时间：人民邮电出版社

作者：Robert C. Martin,Micah Martin

页数：538

字数：936000

译者：邓辉,孙鸣

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：http://www.tushu007.com

内容概要

享誉全球的面向对象技术大师Robert C. Martin在《敏捷软件开发：原则、模式与实践(C#版·修订版)》中深入而生动地使用真实案例讲解了面向对象设计的基本原则、重要的设计模式、UML和敏捷方法。

《敏捷软件开发：原则、模式与实践(C#版·修订版)》Java版曾荣获2003年第13届Jolt大奖，是公认的典著作。
《敏捷软件开发：原则、模式与实践(C#版·修订版)》是C#程序员提升功力的绝佳教程，也可用作高校计算机、软件工程专业本科生、研究生的教材或参考书。

第一图书网, tushu007.com

作者简介

Robert C. Martin（Bob大叔）世界级软件开发大师，著名软件咨询公司Object Mentor公司的创始人和总裁。
曾担任C++ Report杂志主编多年，也是设计模式和敏捷开发运动的主要倡导者之一。

Micah Martin Robert C. Martin之子，也是经验丰富的软件工程师，曾任Object Mentor公司的咨询师，现任8th Light公司总裁。
擅长.NET、面向对象技术、模式和敏捷开发。
他是开源测试工具FitNesse的主要开发者。

第一图书网, tushu007.com

书籍目录

目　录

第一部分　敏捷开发
第1章　敏捷实践　3
1.1　敏捷联盟　4
1.1.1　人和交互重于过程和工具　4
1.1.2　可以工作的软件重于面面俱到的文档　5
1.1.3　客户合作重于合同谈判　5
1.1.4　随时应对变化重于遵循计划　6
1.2　原则　6
1.3　结论　8
1.4　参考文献　8
第2章　极限编程概述　9
2.1　极限编程实践　9
2.1.1　完整团队　9
2.1.2　用户故事　10
2.1.3　短交付周期　10
2.1.4　验收测试　10
2.1.5　结对编程　11
2.1.6　测试驱动开发　11
2.1.7　集体所有　12
2.1.8　持续集成　12
2.1.9　可持续的开发速度　12
2.1.10　开放的工作空间　13
2.1.11　计划游戏　13
2.1.12　简单设计　13
2.1.13　重构　14
2.1.14　隐喻　14
2.2　结论　15
2.3　参考文献　15
第3章　计划　16
3.1　初始探索　17
3.2　发布计划　17
3.3　迭代计划　18
3.4　定义“完成”　18
3.5　任务计划　18
3.6　迭代　19
3.7　跟踪　19
3.8　结论　20
3.9　参考文献　21
第4章　测试　22
4.1　测试驱动开发　22
4.1.1　优先设计测试的例子　23
4.1.2　测试促使模块之间隔离　24
4.1.3　意外获得的解耦合　25

4.2　验收测试　26
4.3　意外获得的构架　27
4.4　结论　27
4.5　参考文献　28
第5章　重构　29
5.1　素数产生程序：一个简单的重构示例　30
5.1.1　单元测试　31
5.1.2　重构　32
5.1.3　最后审视　35
5.2　结论　38
5.3　参考文献　39
第6章　一次编程实践　40
6.1　保龄球比赛　40
6.2　结论　75
第二部分　敏捷设计
第7章　什么是敏捷设计　81
7.1　设计臭味　81
7.1.1　设计臭味——腐化软件的气味　82
7.1.2　僵化性　82
7.1.3　脆弱性　82
7.1.4　顽固性　82
7.1.5　粘滞性　82
7.1.6　不必要的复杂性　83
7.1.7　不必要的重复　83
7.1.8　晦涩性　83
7.2　软件为何会腐化　84
7.3　Copy程序　84
7.3.1　熟悉的场景　84
7.3.2　Copy程序的敏捷设计　87
7.4　结论　88
7.5　参考文献　88
第8章　SRP：单一职责原则　89
8.1　定义职责　90
8.2　分离耦合的职责　91
8.3　持久化　92
8.4　结论　92
8.5　参考文献　92
第9章　OCP：开放-封闭原则　93
9.1　OCP概述　94
9.2　Shape应用程序　95
9.2.1　违反OCP　95
9.2.2　遵循OCP　97
9.2.3　预测变化和“贴切的”结构　98
9.2.4　放置吊钩　99
9.2.5　使用抽象获得显式封闭　99
9.2.6　使用“数据驱动”的方法获取封闭性　100
9.3　结论　101

9.4　参考文献　101
第10章　LSP：Liskov替换原则　102
10.1　违反LSP的情形　103
10.1.1　简单例子　103
10.1.2　更微妙的违反情形　104
10.1.3　实际的例子　108
10.2　用提取公共部分的方法代替继承　111
10.3　启发式规则和习惯用法　113
10.4　结论　114
10.5　参考文献　114
第11章　DIP：依赖倒置原则　115
11.1　层次化　116
11.1.1　倒置的接口所有权　117
11.1.2　依赖于抽象　117
11.2　简单的DIP示例　117
11.3　熔炉示例　119
11.4　结论　121
11.5　参考文献　121
第12章　ISP：接口隔离原则　122
12.1　接口污染　122
12.2　分离客户就是分离接口　123
12.3　类接口与对象接口　124
12.3.1　使用委托分离接口　124
12.3.2　使用多重继承分离接口　125
12.4　ATM用户界面的例子　126
12.5　结论　131
12.6　参考文献　131
第13章　写给C#程序员的UML概述　132
13.1　类图　134
13.2　对象图　135
13.3　顺序图　136
13.4　协作图　136
13.5　状态图　137
13.6　结论　137
13.7　参考文献　137
第14章　使用UML　138
14.1　为什么建模　138
14.1.1　为什么构建软件模型　139
14.1.2　编码前应该构建面面俱到的设计吗　139
14.2　有效使用UML　139
14.2.1　与他人交流　139
14.2.2　脉络图　141
14.2.3　项目结束文档　142
14.2.4　要保留的和要丢弃的　142
14.3　迭代式改进　143
14.3.1　行为优先　143
14.3.2　检查结构　144

14.3.3　想象代码　146
14.3.4　图的演化　147
14.4　何时以及如何绘制图示　147
14.4.1　何时要画图，何时不要画图　147
14.4.2　CASE工具　148
14.4.3　那么，文档呢　149
14.5　结论　149
第15章　状态图　150
15.1　基础知识　150
15.1.1　特定事件　151
15.1.2　超状态　152
15.1.3　初始伪状态和结束伪状态　153
15.2　使用FSM图示　153
15.3　结论　154
第16章　对象图　155
16.1　即时快照　155
16.2　主动对象　156
16.3　结论　159
第17章　用例　160
17.1　编写用例　160
17.1.1　备选流程　161
17.1.2　其他东西呢　161
17.2　用例图　162
17.3　结论　162
17.4　参考文献　162
第18章　顺序图　163
18.1　基础知识　163
18.1.1　对象、生命线、消息及其他　164
18.1.2　创建和析构　164
18.1.3　简单循环　165
18.1.4　时机和场合　166
18.2　高级概念　168
18.2.1　循环和条件　168
18.2.2　耗费时间的消息　169
18.2.3　异步消息　171
18.2.4　多线程　174
18.2.5　主动对象　175
18.2.6　向接口发送消息　175
18.3　结论　175
第19章　类图　177
19.1　基础知识　177
19.1.1　类　177
19.1.2　关联　178
19.1.3　继承　179
19.2　类图示例　180
19.3　细节　181
19.3.1　类衍型　181

19.3.2　抽象类　182
19.3.3　属性　183
19.3.4　聚集　183
19.3.5　组合　184
19.3.6　多重性　185
19.3.7　关联衍型　186
19.3.8　内嵌类　187
19.3.9　关联类　187
19.3.10　关联修饰符　187
19.4　结论　188
19.5　参考文献　188
第20章　咖啡的启示　189
20.1　Mark IV型专用咖啡机　189
20.1.1　规格说明书　190
20.1.2　常见的丑陋方案　192
20.1.3　虚构的抽象　193
20.1.4　改进方案　194
20.1.5　实现抽象模型　198
20.1.6　这个设计的好处　209
20.2　面向对象过度设计　214
20.3　参考文献　214
第三部分　薪水支付案例研究
第21章　COMMAND模式和ACTIVE OBJECT模式：多功能与多任务　219
21.1　简单的Command　220
21.2　事务　221
21.2.1　实体上解耦和时间上解耦　222
21.2.2　时间上解耦　223
21.3　Undo()方法　223
21.4　ACTIVE OBJECT模式　224
21.5　结论　227
21.6　参考文献　228
第22章　TEMPLATE METHOD模式和STRATEGY模式：继承和委托　229
22.1　TEMPLATE METHOD模式　230
22.1.1　滥用模式　232
22.1.2　冒泡排序　232
22.2　STRATEGY模式　235
22.3　结论　239
22.4　参考文献　239
第23章　FACADE模式和MEDIATOR模式　240
23.1　FACADE模式　240
23.2　MEDIATOR模式　241
23.3　结论　243
23.4　参考文献　243
第24章　SINGLETON模式和MONOSTATE模式　244
24.1　SINGLETON模式　245
24.1.1　SINGLETON模式的好处　246
24.1.2　SINGLETON模式的代价　246

24.1.3　运用SINGLETON模式　246
24.2　MONOSTATE模式　247
24.2.1　MONOSTATE模式的好处　249
24.2.2　MONOSTATE模式的代价　249
24.2.3　运用MONOSTATE模式　249
24.3　结论　253
24.4　参考文献　253
第25章　NULL OBJECT模式　254
25.1　描述　254
25.2　结论　256
25.3　参考文献　256
第26章　薪水支付案例研究：第一次迭代开始　257
26.1　初步的规格说明　257
26.2　基于用例分析　258
26.2.1　增加新雇员　259
26.2.2　删除雇员　260
26.2.3　登记考勤卡　260
26.2.4　登记销售凭条　260
26.2.5　登记工会服务费　261
26.2.6　更改雇员明细　261
26.2.7　发薪日　263
26.3　反思：找出底层的抽象　264
26.3.1　雇员支付类别抽象　264
26.3.2　支付时间表抽象　265
26.3.3　支付方式　266
26.3.4　从属关系　266
26.4　结论　266
26.5　参考文献　267
第27章　薪水支付案例研究：实现　268
27.1　事务　268
27.1.1　增加雇员　269
27.1.2　删除雇员　273
27.1.3　考勤卡、销售凭条以及服务费用　274
27.1.4　更改雇员属性　280
27.1.5　犯了什么晕　287
27.1.6　支付雇员薪水　290
27.1.7　支付领月薪的雇员薪水　292
27.1.8　支付钟点工薪水　294
27.2　主程序　302
27.3　数据库　303
27.4　结论　304
27.5　关于本章　304
27.6　参考文献　305
第四部分　打包薪水支付系统
第28章　包和组件的设计原则　308
28.1　包和组件　308
28.2　组件的内聚性原则：粒度　309

28.2.1　重用-发布等价原则　309
28.2.2　共同重用原则　310
28.2.3　共同封闭原则　311
28.2.4　组件内聚性总结　311
28.3　组件的耦合性原则：稳定性　311
28.3.1　无环依赖原则　311
28.3.2　稳定依赖原则　316
28.3.3　稳定抽象原则　319
28.4　结论　322
第29章　FACTORY模式　323
29.1　依赖问题　325
29.2　静态类型与动态类型　326
29.3　可替换的工厂　326
29.4　对测试支架使用对象工厂　327
29.5　工厂的重要性　328
29.6　结论　329
29.7　参考文献　329
第30章　薪水支付案例研究：包分析　330
30.1　组件结构和符号　330
30.2　应用CCP　332
30.3　应用REP　333
30.4　耦合和封装　335
30.5　度量　336
30.6　度量薪水支付应用程序　337
30.6.1　对象工厂　340
30.6.2　重新思考内聚的边界　342
30.7　最终的包结构　342
30.8　结论　345
30.9　参考文献　345
第31章　COMPOSITE模式　346
31.1　组合命令　347
31.2　多重性还是非多重性　348
31.3　结论　348
第32章　OBSERVER——演化至模式　349
32.1　数字时钟　350
32.2　OBSERVER模式　365
32.2.1　模型　365
32.2.2　面向对象设计原则的运用　366
32.3　结论　366
32.4　参考文献　367
第33章　ABSTRACT SERVER模式、ADAPTER模式和BRIDGE模式　368
33.1　ABSTRACT SERVER模式　369
33.2　ADAPTER模式　370
33.2.1　类形式的ADAPTER模式　370
33.2.2　调制解调器问题、适配器以及LSP　370
33.3　BRIDGE模式　374
33.4　结论　375

33.5　参考文献　376
第34章　PROXY模式和GATEWAY模式：管理第三方API　377
34.1　PROXY模式　377
34.1.1　实现PROXY模式　381
34.1.2　小结　391
34.2　数据库、中间件以及其他第三方接口　392
34.3　TABLE DATA GATEWAY　394
34.3.1　测试和内存TDG　399
34.3.2　测试DbGateWay　400
34.4　可以用于数据库的其他模式　403
34.5　结论　404
34.6　参考文献　404
第35章　VISITOR模式　405
35.1　VISITOR模式　406
35.2　ACYCLIC VISITOR模式　409
35.3　DECORATOR模式　418
35.4　EXTENSION OBJECT模式　423
35.5　结论　432
35.6　参考文献　432
第36章　STATE模式　433
36.1　嵌套switch/case语句　434
36.1.1　内部作用域的状态变量　436
36.1.2　测试动作　436
36.1.3　代价和收益　436
36.2　迁移表　437
36.2.1　使用表解释　437
36.2.2　代价和收益　438
36.3　STATE模式　439
36.3.1　STATE模式和STRATEGY模式　441
36.3.2　代价和收益　442
36.4　状态机编译器　442
36.4.1　SMC生成的Turnstile.cs以及其他支持文件　443
36.4.2　代价和收益　448
36.5　状态机应用的场合　448
36.5.1　作为GUI中的高层应用策略　448
36.5.2　GUI交互控制器　450
36.5.3　分布式处理　450
36.6　结论　451
36.7　参考文献　451
第37章　薪水支付案例研究：数据库　452
37.1　构建数据库　452
37.2　一个代码设计缺陷　453
37.3　增加雇员　455
37.4　事务　464
37.5　加载Employee对象　468
37.6　还有什么工作　478
第38章　薪水支付系统用户界面：Model-View-Presenter　479

第一图书网, tushu007.com

38.1　界面　480
38.2　实现　481
38.3　构建窗口　489
38.4　Payroll窗口　495
38.5　真面目　504
38.6　结论　505
38.7　参考文献　505
附录A　双公司记　506
Rufus公司：“ 日落 ”项目　506
Rupert工业公司：“ 朝阳 ”项目　506
附录B　什么是软件　516
索引　524

章节摘录

版权页： 插图： 在探索UML的细节之前，我们应该先讲讲何时以及为何使用它。
UML的误用和滥用已经对软件项目造成了太多的危害。
14.1为什么建模 工程师为何要构建模型？
航天工程师为何要构建飞行器模型？
结构工程师为何要构建桥梁模型？
构建这些模型的目的是什么呢？
这些工程师构建模型的目的是为了知道他们的设计是否可行。
航天工程师构建飞行器模型并把它们放到风洞中是为了看看它们是否能够飞行。
结构工程师构建桥梁模型是为了看看它们是否能够屹立不倒。
建筑师构建建筑模型是为了看看他们的客户是否喜欢这些建筑的样子。
构建模型就是为了弄清楚某些东西是否可行。
这意味着模型必须是可测试的。
如果不能对模型应用一些可测试的标准，那么构建模型是毫无用处的。
如果模型不能被评估，那么这个模型就没有任何价值。
航天工程师为何不是直接构建一架飞机，然后就让它试着飞行？
结构工程师为何不是直接建造一座桥梁，然后看看它是否能够屹立不倒？
答案很简单，飞机和桥 梁要比模型昂贵很多。
当模型比要构建的真实实体便宜得多时，我们就会使用模型来研究设计。
14.1.1为什么构建软件模型 UML图是可测试的吗？
和其表示的软件相比，它创建和测试起来更便宜一些吗？
针对这两个问题，我们无法像航天工程师以及结构工程师那样得出明显的答案。
在测试UML图方面，没有严格的标准。
我们可以查看它，评估它，并应用一些原则和模式，但是评估最终仍是主观的。
绘制UML图确实要比编写软件代价更小一些，但并没有小多少。
事实上，时常会出现更改源代码比更改图示更容易的情况。
那么，在什么情况下使用UML是有意义的呢？
如果UML没有使用价值，那么我就不会编写这几章了。
不过，UML确实很容易被误用。
当我们有一些确定的东西需要测试，并且使用UML要比使用代码测试起来代价更低一些时，就使用UML。
比如，我有一个关于某个设计的想法。
我想知道团队中的其他开发人员是否认为它是一个好的想法。
于是，我就在白板上画一幅UML图，并询问团队成员的意见。

媒体关注与评论

本书是对敏捷编程和敏捷原则最全面和最有价值的介绍……本书绝对是所有.NET程序员必读之作。
—Jesse Liberty，微软资深技术专家我最喜爱的技术作家Robert Martin善于通过实践展示技术，让读者能够以自己喜欢的方式逐步理解……请把Bob大叔当做你在敏捷世界里的导师。
—Chris Sells，.NET资深技术专家，微软“软件传奇人物”前几天，我找到了记有我对Bob大叔第一印象的备忘录。
上面写着“优秀的对象思想”。
你手中的这本书就是能让你受益终生的“优秀的对象思想”。
—Kent Beck，软件开发大师，极限编程之父，设计模式先驱我期待这本书已经很久了，关于如何去掌握我们的行业技能，本书作者有非常丰富的实际经验可以传授。
—Martin Fowler，软件开发大师，《重构》与《企业应用架构模式》作者这本书中充满了对于软件开发的真知灼见。
不管你是想成为一个敏捷开发人员，还是想提高自己的技能，本书都同样有用。
我一直在期盼着这本书，它没有令我失望。
—Erich Gamma，软件开发大师，《设计模式》作者在本书中，Bob Martin向我们展示了他作为开发大师以及教育家的天赋。
书中都是重要的经验，并且阅读起来就是一种享受。
他以其务实的判断力和令人愉悦的风格给我们以启迪。
—Craig Larman，《UML和模式应用》作者这也许是第一部把敏捷方法、模式和最新的软件开发基本原则完美结合在一起的图书。
当Bob Martin发言时，我们最好洗耳恭听。
—John Vlissides，软件开发大师，《设计模式》作者本书作者成功地实现了目标，这要归功于他三十多年的开发经验。
……本书对设计模式的阐述使我难以释卷。
—Diomidis Spinellis，软件开发大师，《高质量程序设计艺术》作者以我之见，本书不愧为最佳面向对象设计图书。
—John Wetherbie，JavaRanch.com

名人推荐

本书是对敏捷编程和敏捷原则最全面和最有价值的介绍……本书绝对是所有.NET程序员必读之作。
——Jesse Liberty，微软资深技术专家 我最喜爱的技术作家Robert Martin善于通过实践展示技术，让读者能够以自己喜欢的方式逐步理解…请把Bob大叔当做你在敏捷世界里的导师。
——Chris Sells，.NET资深技术专家，微软“软件传奇人物”前几天，我找到了记有我对Bob大叔第一印象的备忘录。
上面写着“优秀的对象思想”。
你手中的这本书就是能让你受益终生的“优秀的对象思想”。
——KentBeck，软件开发大师，极限编程之父，设计模式先驱。
我期待这本书已经很久了，关于如何去掌握我们的行业技技能，本书作者有非常丰富的实际经验可以传授。
——Martin Fowler，软件开发大师，《重构》与《企业应用架构模式》作者 本书中充满了对于软件开发的真知灼见。
不管你是想成为一个敏捷开发人员，还是想提高自己的技能，本书都同样有用。
我一直在期盼着这本书，它没有令我失望。
__Erich Gamma，软件开发大师，《设计模式》作者 在本书中，Bob Martin向我们展示了他作为开发大师以及教育家的天赋。
书中都是重要的经验，并且阅读起来就是一种享受。
他以其务实的判断力和令人愉悦的风格给我们以启迪。
——Craig Larman，《UM L和模式应用》作者 这也许是第一部把敏捷方法、模式和最新的软件开发基本原则完美结合在一起的图书。
当Bob Martin发言时，我们最好洗耳恭听。
——John Vlissides，软件开发大师，《设计模式》作者 本书作者成功地实现了目标，这要归功于他三十多年的开发经验。
……本书对设计模式的阐述使我难以释卷。
——Diomidis Spinellis，软件开发大师，《高质量程序设计艺术》作者 以我之见，本书不愧为最佳面向对象设计图书。
——John Wetherbie，JavaRanch.com

第一图书网, tushu007.com

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:http://www.tushu007.com