TURING 图灵程序设计丛书 程序员修炼系列

Addison Wesley

Refactoring to Patterns

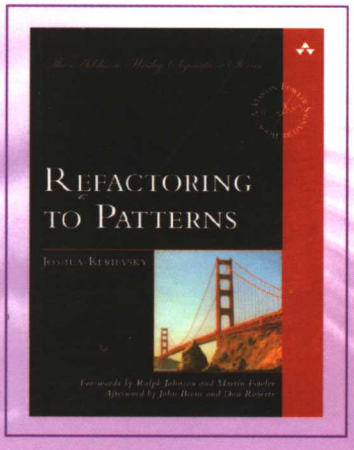
重构与模式

software development 15th annual productivity award

[美] Joshua Kerievsky 著

杨 光 刘基诚 译

- 《设计模式》和《重构》之后又一里程碑式著作
- 凝聚众多业界专家经验与领悟
- 帮你打通重构与模式任督二脉



人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

重构与模式

Refactoring To Patterns

[美] Joshua Kerievsky 著

杨 光 刘基诚 译

人民邮电出版社
POSTS & TELECOM PRESS

图书在版编目（CIP）数据

重构与模式 /（美）科瑞夫斯盖著；杨光，刘基诚译. —北京：人民邮电出版社，2006.12
（图灵程序设计丛书）

ISBN 7-115-15336-1

Ⅰ. 重… Ⅱ. ①科…②杨…③刘… Ⅲ. 程序设计 Ⅳ. TP311.1

中国版本图书馆 CIP 数据核字（2006）第 117727 号

内 容 提 要

本书开创性地深入揭示了重构与模式这两种软件开发关键技术之间的联系，说明了通过重构实现模式改善既有的设计，往往优于在新的设计早期使用模式。本书不仅展示了一种应用模式和重构的创新方法，而且有助于读者结合实战深入理解重构和模式。书中讲述了 27 种重构方式。

本书适于面向对象软件开发人员阅读，也可作为高校计算机专业、软件工程专业师生的参考读物。

图灵程序设计丛书

重构与模式

◆ 著 [美] Joshua Kerievsky
译 杨 光 刘基诚
责任编辑 傅志红

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子函件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京顺义振华印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：800 × 1000 1/16
印张：19.75
字数：465 千字 2006 年 12 月第 1 版
印数：1– 5 000 册 2006 年 12 月北京第 1 次印刷

著作权合同登记号 图字：01-2005-3576 号

ISBN 7-115-15336-1/TP • 5727

定价：45.00 元

读者服务热线：(010)88593802 印装质量热线：(010)67129223

对本书的赞誉

“重构必须付诸实践，而非仅仅作为一种抽象的智力练习，才能体现出其真正价值。模式则记录了具有公认良好属性的程序结构。本书将两者完美地结合起来。如果想真正实践重构，我推荐你阅读本书并活学活用。”

——Kent Beck，三川学院院长，极限编程创始人，模式先驱

“在《设计模式》一书中，我们曾经提到，设计模式是重构的目标。本书终于证实我们所言不虚。除此之外，本书还能够加深读者对设计模式和重构两方面的领悟。”

——Erich Gamma，IBM 公司 Eclipse Java 开发工具负责人，《设计模式》四作者之一，模式先驱

“现在，软件模式和敏捷开发之间的联系终于被人道破。”

——Ward Cunningham，极限编程创始人，模式先驱，Wiki 发明者

“本书展示了一种应用模式的创新方法，将自上而下地使用设计模式与自下而上地揭示迭代式开发和持续重构结合起来。任何职业软件开发人员都应该使用这种方法，去寻找使用模式改进代码的新的可能。”

——Bobby Woolf，IBM 公司 WebSphere 软件服务部门 IT 咨询专家，*Enterprise Integration Patterns* 和 *The Design Patterns Smalltalk Companion* 作者之一

“Joshua Kerievsky 通过一系列独树一帜的设计级重构，将重构提升到全新的层次。本书向开发人员展示了如何对设计进行改进，从而简化日常工作。本书是重构实践的珍贵参考书。”

——Sven Gorts，重构与敏捷开发布道者，比利时 refactoring.be 网站创始人

“本书是对《设计模式》一书的重构，可能意义还不仅限于此。在此之前，设计模式这一主题一直是作为静态和僵化的过程来阐述的，本书则将其看作是动态和灵活的，使模式的学习变成了一种试验、出错然后改正的人性化过程，从中读者能够理解到，优秀的设计并非一蹴而就——它们都经历了艰难和反思。Kerievsky 还重构了阐述方式本身，使其更加清晰，更容易接受。实际上，他解决了我在写作 *Thinking in Patterns* 一书中遇到的许多组织问题。本书透彻地介绍并结合了测试、重构和设计模式诸多方面，字里行间洋溢着叙述的轻松、良好的技术感觉和难得的真知灼见。”

——Bruce Eckel，Mindview 公司总裁，《Java 编程思想》和《C++编程思想》的作者

“我第一次见到 Joshua，就对他在理解、应用和教授设计模式上表现出来的热情留下了深刻印象。伟大的教师都对自己教授的内容和如何与人分享有这样的热情。我想 Joshua 不愧是一个伟大的教师，一个伟大的开发者，我们都从他的深刻洞察中获益良多。”

——Craig Larman，Valtech 首席科学家，《UML 和模式应用》和《敏捷迭代开发》的作者

“本书非常重要，不仅由于它为有条不紊地引入合适的模式以改进代码提供了循序渐进的指导，更重要的是，它教授了设计模式之下的原则。本书对于新入门和专家级的设计人员都同样适用。真是一本伟大的书。”

——Kyle Brown，IBM 公司 WebSphere 软件服务部门，*Enterprise Java™ Programming with IBM® WebSphere®, Second Edition* 作者

“掌握一门手艺不仅仅要获得正确的工具，还需要学会高效地使用工具。本书阐释了如何将工业级的设计工具与艺术家的技巧融于一炉。”

——Russ Rufer，硅谷模式小组

“Joshua 使用模式引导一步步小的重构，从而实现更大的目标，又通过重构将模式引入代码，从而对其做出改进。你将学会如何以渐进方式大大改善既有代码，而不是强制地去适应一个预先设计好的解决方案。随着代码不断改变，你将实现超越，看到更好的设计方案——是的，你将体验到这些。”

——Phil Goodwin，硅谷模式小组

译 者 序

设计模式和重构对我们来说早已不是什么陌生的字眼了。1994 年，GoF 的巨著《设计模式》初次向世人展示了设计模式的魅力。2002 年，Martin Fowler 的《重构：改善既有代码的设计》则刮起了一阵重构的旋风。记得在《重构》刚刚出版的时候，软件开发界和评论界就赞扬它是一本具有与《设计模式》同等高度的图书。我相信本书的每一位读者都和我一样，早已收藏了这两本书，反复阅读，仔细品味，并从中获益匪浅。

设计模式代表了传统软件开发的思想：好的设计会产生好的软件，因此在实际开发之前，值得花时间去做一个全面而细致的设计。而重构则代表了敏捷软件开发的浪潮：软件并不是在一开始就可以设计得完美无缺的，因此可以先进行实际开发，然后通过对代码不断地进行小幅度的修改来改善其设计。这两种方式看似格格不入，但是它们都在本质上有一个相同的思想——设计很重要，只是两者达到良好设计的方法不同。从设计模式和重构第一天与开发人员见面开始，它们就注定是一对休戚相关的兄弟。现在，本书终于为人们架设了一道连接设计模式与重构的桥梁。

这段时间，我总在想这样一个问题：什么是设计模式？每一类编程语言都具有其自身的特性，就面向对象编程语言来说，其特性就是抽象、封装、继承和多态。同时，使用每一类编程语言开发软件时也都有一些设计准则，这些准则保证了软件的质量，即具有良好的设计。而设计模式则是广大软件开发人员总结出的开发经验和技巧，它们利用编程语言的特性，实现这些设计准则。因此，在有经验的软件设计师眼里，没有设计模式，只有设计准则。现在，本书作者告诉我们：重构是实现设计模式的一种手段，设计模式往往也是重构的目的。从某种意义上来说，重构成全了设计模式，而设计模式度量了重构。需要注意的是，所谓“设计模式是重构的目的”，并不是说重构的结果一定是设计模式，有些情况下，重构恰恰是为了避免设计模式的过度使用。这是本书最值得关注的地方。

在准备写这篇译者序的时候，我总是觉得很为难，因为写译者序类似于写读后感，是要道出翻译过程中的特别感受，而我在翻译的过程中并没有什么特别突兀的触动。从本书的第一个重构直到最后一个，一切都显得那么自然；作者给出的每一个建议，每一个告诫，每一次小小的改动，给我的感觉都是水到渠成的。现在想想，其实重构的魅力就在于此，它就是每个软件开发人员自然而然应该做的事情。有句话叫“绚烂之极归于平淡”，用来形容重构，真是再合适不过了。

翻译从来就不是一件轻松的事，加之我完成本书翻译的日子都是在上海炎热的夏天中度过

的。每当我汗流浃背地坐在电脑前，斟酌应该如何表达作者原意的时候，我都会从作者迸发的思维和精巧的话语中感受到一种平淡而又无穷的智慧。有趣的是，我接受本书翻译邀请的那天，Martin Fowler 先生正好到上海做演讲，能在这位《重构》作者的演讲堂上接受到一本重构图书的翻译邀请，真是机缘巧合。

最后，我要谢谢我的父母。是他们给了我一个健康的身体，一个良好的生活环境，更重要的是全心全意的支持。没有他们就没有这一切。爸爸妈妈，我爱你们。

译　者

2006 年 8 月 25 日

Ralph Johnson 序

《设计模式》一书中叙述了使用模式的几种方式。有些人在编写任何代码之前，都要很早地为模式做计划，而有些人在编写了大量代码之后才开始添加模式。第二种使用模式的方式就是重构，因为是要在不增加系统特性或者不改变其外部行为的情况下改变系统的设计。有些人在程序中加入模式，只是因为觉得模式能够使程序更容易修改；更多人这样做只是为了简化目前的设计。如果代码已经编写，这两种情形都是重构，因为前者是通过重构使修改更容易，而后者则是通过重构在修改后进行整理。

虽然模式是在程序中能够看到的东西，但是模式也是一种程序转换。每个模式都可以通过展示模式应用前后程序的变化来进行解释。这是可以将模式看作重构的另一种方式。

不幸的是，许多读者都忽视了设计模式和重构之间的联系。他们认为模式只是关乎设计，与代码无关。我想可能是设计模式这个书名误导了他们，可是《设计模式》一书中到处都是 C++ 代码，这一点应该也说明了模式与设计和代码都密切相关，而且添加模式通常都需要改变代码。

Joshua Kerievsky 恰恰发现了这种联系。我初次遇到他的时候，他刚刚开始组织纽约市的设计模式学习小组。他介绍了通过“前后变化”——用例子说明模式对某个系统的影响来学习模式的想法。在他富于感染力的热情号召下，他离开纽约市之前，小组已经发展到 60 多人，每月聚会数次。他开始通过客户现场培训、自己开班和因特网为各个公司教授模式课程。他甚至还教其他人如何教授模式。

Joshua 继而还成为一位极限编程实践者和教师。因此，由他来写一本书介绍设计模式与极限编程的核心实践之一——重构之间的联系，可以说是再合适不过了。重构与设计模式绝不是没有关系的——相反，它们密切相关。虽然本书中谈到的模式并不是都来自我们的《设计模式》一书，但是都遵循书中的风格。本书说明了怎样让模式帮助我们设计，而又不必进行预先设计。

按本书中所教授的方法进行实践吧，这不仅能够提高你做出优秀设计的能力，也能够提高思考优秀设计的能力。

——Ralph Johnson，伊利诺伊大学厄巴尼-尚佩恩分校教授，《设计模式》四作者之一

Martin Fowler 序

几年来，我参与了敏捷方法尤其是极限编程的宣传和推广活动。在此过程中，人们经常会问到，这些方法与我长期对设计模式的兴趣是怎样和平共处的。事实上，我还曾经听人说，在我鼓励人们重构和演进式设计时，实际上是在放弃自己以前关于分析模式和设计模式的作品中所讲述的观点。

这种说法其实并不确切。看看模式社区的那些主要成员，再看看敏捷方法和极限编程社区的主要成员，有很多人活跃在两个社区。事实上，模式和演进式设计从非常早的时期起就有着密切的关系。

Joshua Kerievsky 正处在两个社区交集的核心。我初次遇到他的时候，他已经在纽约市组织了一个成功的模式学习小组。小组成员互相合作，研究不断涌现的设计模式文献。我很快认识到 Joshua 对设计模式的领悟力可以说是首屈一指的，我从倾听他的谈话中获得了许多真知灼见。所以，他后来成为一名极限编程的先锋，对我来说是意料之中的。他在第一次极限编程会议上关于模式和极限编程的论文是我的最爱之一。

正因为如此，如果说有什么人最适合写模式与重构之间联系的话，那应该非 Joshua 莫属了。这个主题我在《重构》一书中曾经有所涉及，但是并没有深入探讨，因为我想把篇幅集中于基本的重构上。本书极大地扩展了这一主题，非常详细地讨论如何发展出《设计模式》一书[DP]中大多数流行的模式，说明了不需要预先将它们设计到系统中，而是应该随着系统发展而逐步演变出来。

除了通过学习获得的有关这些重构的具体知识以外，本书还讲述了有关模式与重构的一般性知识。许多人都说过，重构是学习模式的一种更佳方式，因为可以在重构的演进步骤中看到问题和解决方案之间的互动。这些重构还进一步证实了一个重要事实：重构其实就是循序渐进地进行较大的修改。

因此我非常高兴能够将本书介绍给大家。我花了很长时间说服 Joshua 写一本书，这些努力最终促成了本书的诞生。我对这一成果非常满意，不知读者以为然否？

——Martin Fowler

前　言

本书主旨

本书讲述的是重构（改善既有代码设计的过程）与模式（针对反复出现的问题的经典解决方案）的结合。本书建议，使用模式来改善既有的设计，要优于在新的设计早期使用模式。这对于已经存在几年和几分钟的代码都同样适用。我们通过一系列低层次的设计转换，也就是重构，来应用模式，改进设计。

本书目的

撰写本书是为了帮助读者：

- 理解如何结合重构和模式；
- 用模式导向的重构（pattern-directed refactoring）改善既有代码的设计；
- 找出需要进行模式导向重构的代码段；
- 了解为什么使用模式来改善既有的设计要优于在新的设计早期使用模式。

为了实现这些目的，本书包含以下特色：

- 一个含有 27 种重构方式的目录；
- 示例以实战代码为基础，没有纯示意性的玩具代码；
- 模式的描述，包括实际的模式示例；
- 一组坏味[1]（也就是问题），表示需要进行模式导向的重构；
- 实现同一模式的不同方式的示例；
- 就什么时候应该通过重构实现模式、趋向模式以及去除模式给出建议。

为了帮助个人和小组学习本书中的 27 种重构，本书给出了学习顺序的建议。

1. 本书中将 smell 译为坏味，这是借用了围棋术语。围棋中说味道不好或者有坏味，通常就是指感觉可能存在潜在的问题。——译者注

读者对象

本书的读者是从事或者有兴趣改善既有代码设计的面向对象程序员。他们中很多人都在使用模式和重构，但是从来没有通过重构来实现模式。还有一些程序员对重构和模式知之甚少，但愿意了解更多相关内容。

本书对新项目开发（从头编写新的系统或者特性）和遗留开发（主要是维护遗留系统）都适用。

所需背景

本书要求读者熟悉紧耦合、松耦合等设计方面的概念，以及继承、多态、封装、组合、接口、抽象类和具体类、抽象方法和静态方法等面向对象方面的概念。

书中示例使用 Java 代码。我发现对于大多数面向对象程序员来说，Java 代码都很容易读懂。我有意识地不使用那些 Java 独有的特性，因此无论你习惯于用 C++、C#、Visual Basic .NET、Python、Ruby、Smalltalk，还是其他面向对象语言编程，都应该能够理解本书中的代码。

本书与 Martin Fowler 的经典著作《重构》[F]息息相关。该书中包含了对许多低层次的重构，例如：

- 提炼方法（Extract Method）
- 提炼接口（Extract Interface）
- 提炼超类（Extract Superclass）
- 提炼子类（Extract Subclass）
- 上移方法（Pull Up Method）
- 搬移方法（Move Method）
- 重命名方法（Rename Method）

《重构》一书中还有一些更复杂的重构，例如：

- 用委托替换继承（Replace Inheritance with Delegation）
- 用多态替换条件语句（Replace Conditional with Polymorphism）
- 用子类替换类型代码（Replace Type Code with Subclasses）

为理解本书中介绍的模式导向的重构，读者无需了解上面列出的所有重构，相反可以跟随阐释这些重构的示例代码进行学习。但是，如果要获取阅读本书的最佳效果，我推荐你同时有一本《重构》在手。该书是无价的重构资源，而且对理解本书很有帮助。

我要讨论的模式来自经典图书《设计模式》[DP]，还有其他作者的作品，包括 Kent Beck、Bobby Woolf 和我本人的作品。我和同事们在实际项目中都实践了重构实现、重构趋向和重构去

除这些模式。通过学习模式导向重构的艺术，你将理解如何重构实现、重构趋向和重构去除本书中没有提到的模式。

阅读本书不必事先成为这些模式的专家，但是对模式有所了解当然会有帮助。为了帮助读者理解所讨论的模式，本书包含了一些简洁的模式总结、模式的 UML 略图和许多示例实现代码。要更详细地理解模式，我推荐你在学习本书的同时，也结合研读所引用的模式文献。

本书使用 UML 2.0 表示法。如果对 UML 不太熟悉的话，不要担心。我也只是知其大略而已。编写本书时，Fowler 的《UML 精粹》[Fowler, UD] 一书常伴我左右，不时查阅。

如何使用本书

要概略地了解本书中的重构，可以从学习每个重构的总结（参见 5.1 节），以及每个重构中“动机”一节的“优点和缺点”开始。

要更深入地理解重构，应该研究每个重构的各个部分，但“做法”一节除外。“做法”一节比较特殊。它的目的是通过建议应该遵循哪些低层次重构，帮助读者实现该重构。理解本书中的重构，并不需要阅读这一节。这一节更可能用作在实际重构时的参考。

本书和《重构》[F] 所讨论的代码坏味（code smell），是识别设计问题和找到有助于解决问题的相关重构的一种有益方式。也可以查看本书和《重构》中的重构列表（按字母顺序排列），找到能够改进设计的重构。

本书记载的是使设计实现、趋向和去除模式的重构。为了帮助你找到着手的方向，3.4 节专门讲述这一主题。本书还有一个表列出了所有模式的名称和可以用于使设计实现、趋向和去除模式的重构。

本书历史

我从 1999 年开始动笔撰写本书。当时，有好几个因素都促使我为模式、重构和极限编程（extreme programming，XP）[Beck, XP]写点什么。首先，我非常吃惊地发现，XP 文献中还没有提及模式。我因此撰写了一篇名为 *Patterns & XP*（模式与 XP）的论文[Kerievsky, PXP]，在该文中我公开地讨论了这一问题，并就如何将软件开发界的这两大主题结合起来提出了一些建议。

其次，我知道 Martin Fowler 在《重构》 [F] 一书中只写到了几个“通过重构实现模式”，而且他明确表示，希望有人在此方面进一步写作。这看上去是一个很值得努力的目标。

最后，在我和同事教授的设计模式研讨班上，我注意到有些学员需要更多指导，才能决定何时应该在设计中实际地应用模式。知道模式是什么是一回事，而真正理解什么时候去如何应用模式，就完全是另一回事了。我认为这些学员需要学习一些实际的案例，在这些案例中，在设计时应用模式能看到实实在在的效果，因此我开始将这种案例汇编成一个提纲。

当我开始撰写本书时，我遵循了 Bruce Eckel[1]的优秀写作传统，将草稿在网上公开，听取人们的意见。网络真是一个好东西。许多人向我发来反馈，有建议，有鼓励，也有感谢。

随着书稿和想法的不断成熟，我开始在许多会议、讲座和 Industrial Logic 公司的“模式与重构”强化研讨班上讲授“通过重构实现模式”的主题。这使我获得了更多的改进建议，而且更多地了解到程序员理解这一主题需要些什么。

渐渐地，我认识到重构是审视模式的最佳方式，而且模式正是一系列低层次重构所能达到的最佳目标。

很幸运，书成稿之后，得到了许多经验丰富的专业人士的审阅，他们提出了很多改进建议。我在致谢中提供有关他们的更多情况。

站在巨人肩上

1995 年的夏天，我走进书店，第一次见到了《设计模式》[DP]一书，并从此与模式结下不解之缘。我希望感谢四位作者 Erich Gamma、Richard Helm（我还未曾谋面）、Ralph Johnson 和 John Vlissides 编写了如此优秀的技术图书。他们在书中所表现出的睿智，使我大大提高了自己的软件设计水平。

大约在 1996 年，我在一次模式会议上遇到了 Martin Fowler，那时他还没有出名。这就是我们长期友谊的开始。如果 Fowler（以及他的合作者 Kent Beck、William Opdyke、John Brant 和 Don Roberts）没有写经典著作《重构》 [F]的话，我真地怀疑自己是否还能写出这本书。与《设计模式》一样，《重构》完全改变了我从事软件设计的方式。

我能够完成本书，全拜《设计模式》和《重构》的作者们的辛勤劳动所赐。对此我感激不尽。

致谢

我是如此幸运，有一位妻子在我写作本书期间全心全意地支持我。Tracy，你是最棒的。我愿与你白头偕老。

我们的两个女儿 Sasha 和 Sophia，都是在我写作本书期间出生的。我要感谢她们在爸爸写作时候表现出的耐心。

在 20 世纪 70 年代，我的父亲 Bruce Kerievsky 将我和哥哥带到工作场所，让我们画那些位于空调房间中的巨大计算机。他还给我们看长长的绿色和白色的计算机清单，上面用巨大的字母写

1. Bruce Eckel 是《Java 编程思想》和《C++编程思想》的作者，他令人吃惊地将全书电子文件公开，结果却取得了巨大成功。——译者注

着我们的名字。这些都激励我进入了这个伟大的行业——谢谢你，父亲。

感谢家人之后，应该是技术方面的致谢了。

John Brant 对本书居功至伟。他和他的同事 Don Roberts 都位居世界上最渊博的重构专家之列。John 审阅了本书手稿的四个版本，提出了很多想法，并鼓励我删去许多比较平淡的内容。他的真知灼见遍及目录中几乎所有重构“做法”部分的字里行间。Don 虽然忙于其他的项目，未能投入更多精力，但是他复查了 John 的反馈意见，非常感谢。我还要感谢两位为本书题跋。

Martin Fowler 在审阅和建议上用力甚勤，包括简化略图和澄清某些技术讨论。他帮助我改正了一些有问题的 UML 图，而且进行了更新以反映 UML 2.0 的变化。我很荣幸 Martin 选中本书作为他主编的签名系列之一，感谢他为本书作序。

Sven Gorts 下载了本书手稿的多个版本，发来数量惊人的经过深思熟虑的意见。他提出了许多有用的想法，使本书的内容、图和代码都有改进。

Somik Raha 在本书内容的提高上帮助很大。他的开源项目 htmlparser，是在他完全掌握模式之前启动的，成了需要“通过重构实现模式”的代码宝库。 Somik 和我结对完成了其中的许多重构。由衷地感谢他的支持、鼓励和建议。

Eric Evans，《领域驱动设计》一书的作者，对本书手稿的早期版本提供了建议。我们在写书的过程中，经常在旧金山附近的咖啡厅会面。在那里我们共同写作、交换电脑，并评论对方的书稿。Eric，感谢你的反馈和友谊。

Chris Lopez，硅谷模式小组（SVPG）的成员，对书的内容、图和代码提出了大量极为详细和有用的建议。同时也感谢硅谷模式小组的其他成员。Chris 对本书的细心审阅大大超出了常规。

Russ Rufer、Tracy Bialik 和硅谷模式小组的其他程序员（包括 Ted Young、Phil Goodwin、Alan Harriman、Charlie Toland、Bob Evans、John Brewer、Jeff Miller、David Vydra、David W. Smith、Patrick Manion、John Wu、Debbie Utley、Carol Thistlethwaite、Ken Scott-Hlebek、Summer Misherghi 和 Siqing Zhang）开了许多会议，审阅本书较早和更成熟的版本。他们提出了大量好的建议，帮助我认识到哪些地方需要澄清、扩充和精简。特别感谢 Russ 为本书安排这么多会议，感谢 Jeff 为讨论录音。

Ralph Johnson 和他领导的 UIUC（伊利诺伊大学厄巴尼-尚佩恩分校）的模式阅读小组对本书手稿的早期版本提出了极为有用的反馈。这些反馈是用 MP3 文件记录下来的。我花了大量时间倾听他们讨论的录音，并采纳了许多建议。我尤其要感谢 Ralph、Brian Foote、Joseph Yoder 和 Brian Marick，感谢他们的关心和建议。我还要感谢小组里的其他人——我还不知道他们的名字。感谢 Ralph 为本书作序。

John Vlissides 以各种形式提供了极为有用的反馈，包括对本书草稿第一版的许多详尽的注释。他对我的工作鼓励有加，对此深表谢意。

Erich Gamma 为本书介绍性的内容以及重构提供了一些很棒的建议。

Kent Beck 审阅了本书中的许多重构，而且还提供了内联 Singleton（6.6 节）重构中的旁注。我非常感谢他在意大利 Alghero 召开 XP2002 会议期间与我结对编程，合作创造了 State 模式的重构。

Ward Cunningham 也提供了内联 Singleton（6.6 节）重构的旁注，并对编排介绍性的内容提供了有益而且关键的建议。

Dirk Baumer（Eclipse 开发自动重构的首席程序员）和 Dmitry Lomov（IntelliJ 开发自动重构的首席程序员）都为本书中的许多重构贡献了真知灼见和建议。

Kyle Brown 审阅了手稿较早的版本，提供了许多很好的意见。

Ken Shirriff 和 John Tangney 对本书手稿的很多版本都提供了大量富于想法的反馈。

Ken Thomases 指出了用类替换类型代码（9.1 节）重构中“做法”的较早版本的一个严重错误。

Robert Hirshfeld 帮助阐明了将装饰功能搬移到 Decorator（7.3 节）重构较早版本中的做法。

Ron Jeffries 在 extremeprogramming@yahoogroups.com 上与我长篇大论地争论，帮助我澄清了本书中的一些内容。他还帮助我“重构”了本书介绍性内容中很难处理的一节中的文字。

Dmitri Kerievsky 帮助我润色了前言中的文字。

以下诸位也不断提供了许多有益的反馈：Gunjan Doshi、Jeff Grigg、Kaoru Hosokawa、Don Hinton、Andrew Swan、Erik Meade、Craig Demyanovich、Dave Hoover、Rob Mee 和 Alex Chaffee。

我还要感谢邮件列表 refactoring@yahoogroups.com 上讨论本书中重构的诸位的反馈。

我要感谢 Industrial Logic 公司各种课程、设计模式研讨班和测试与重构研讨班上的学员，他们也对本书中的重构提供了建议。其中许多人帮助我了解到书中哪些地方不够清楚，哪些地方讲得还不够。

我特别感谢编辑 Paul Petralia，还有他的团队（Lisa Iarkowski、Faye Gemmellaro、John Fuller、Kim Arney Mulcahy、Chrysta Meadowbrooke、Rebecca Rider 和 Richard Evans）。当其他出版社也在争取出版本书时，是 Paul 煞费苦心地说服 Addison-Wesley 争得了出版权。对此我由衷地感谢。我阅读 Addison-Wesley 许多知名著作多年，本书能够成为其中一员我备感荣幸。在本书写作过程中，Paul 成了我的朋友。在他不唠唠叨叨地催促我加紧完稿的时候，我们在一起谈孩子、打网球，度过了许多愉快和轻松的时光。谢谢你，Paul——有你这样的编辑真是幸运。

目　录