TURING 图灵程序设计丛书 程序员修炼系列

PRENTICE HALL

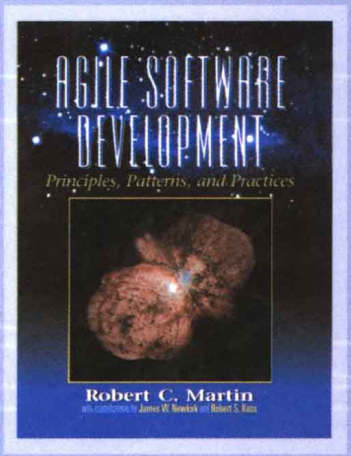
Agile Software Development
Principles, Patterns, and Practices

敏捷软件开发

原则、模式与实践 英文注释版

2003年度Jolt大奖得主

[美] Robert C. Martin 著



- 软件开发的不朽经典
- 生动阐述面向对象原则、敏捷实践、UML和模式
- 大量Java和C++实战示例，让你亲历现场
- 丰富的词汇和背景注释，助你轻松读经典

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书 程序员修炼系列

Agile Software Development Principles, Patterns, and Practices

敏捷软件开发

原则、模式与实践 英文注释版

[美] Robert C. Martin 著

人民邮电出版社
北 京

图书在版编目（CIP）数据

敏捷软件开发：原则、模式与实践（英文注释版）/（美）马丁（Martin, R.C.）著. —北京：人民邮电出版社，2008.1
（图灵程序设计丛书）
ISBN 978-7-115-16439-1

Ⅰ. 敏… Ⅱ. 马… Ⅲ.软件开发—英文 Ⅳ.TP311.52

中国版本图书馆 CIP 数据核字（2007）第 091531 号

内 容 提 要

本书中，享誉全球的软件开发专家和软件工程大师Robert C. Martin深入而生动地使用真实案例讲解了面向对象基本原则、重要的设计模式、UML和敏捷实践等程序员必备的知识。

本书于2003年荣获第13届Jolt大奖，是C++和Java程序员提高自身水平的绝佳教材，也适于用作高校计算机、软件工程专业相关课程的教材或参考书。

图灵程序设计丛书

敏捷软件开发：原则、模式与实践（英文注释版）

◆ 著 [美] Robert C. Martin
责任编辑 傅志红 王慧敏

◆ 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
北京鸿佳印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：787×1092 1/16
印张：34.75
字数：667 千字 2008 年 1 月第 1 版
印数：1－3 000 册 2008 年 1 月北京第 1 次印刷
著作权合同登记号 图字：01-2006-7039 号

ISBN 978-7-115-16439-1/TP

定价：69.00 元

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

出 版 说 明

经过半年多的努力，“图灵程序设计丛书”的新成员——几本英文注释版图书终于与大家见面了。

本次出版的《重构》、《企业应用架构模式》、《敏捷软件开发》（Java 版和 C#版）和《程序员修炼之道》五部著作都是软件行业公认的为数不多的真正的经典之作，如果排除特定于具体语言、工具和技术的图书，它们可以毫无疑问地入选所有软件开发人员必读技术书目的前十名（至于其他候选者，脑中闪过《设计模式》、《人月神话》、《计算机程序设计艺术》、《编程珠玑》、《代码大全》……）。

这其中，《**重构**》曾经与《设计模式》、《反模式》（中文版即将由人民邮电出版社出版）、《解析极限编程——拥抱变化》并称为软件工程四大名著。Martin Fowler 虽然不是重构技术的创造者，但是由于 Kent Beck、John Brant、William Opdyke 和 Don Roberts 等先驱的襄助，本书也成为实至名归的开创性著作。由于近年来工具支持的加强，重构已经越来越成为开发人员日常不可或缺的技术。《**企业应用架构模式**》曾经荣获 2003 年 Software Development 杂志读者选择大奖和 Jolt 生产效率奖，是 Martin Fowler 的另一部名著，某种意义上也是真正奠定他在技术界中地位的一部重要著作。如果说《设计模式》是程序设计层次的圣经，《面向模式的软件架构》是架构设计层次的圣经的话，此书则当之无愧地可以称为企业级应用的圣经。企业级开发已成为主流，其中的困难与挑战是显而易见的，而这一领域的图书一直不多，所以此书更加弥足珍贵。Fowler 的这两部著作都是对业界多年积累的经验的总结，实战性非常强，不仅使你知其所以然，更让你知道如何实现、各种实现方式的适用场合、实现中需要注意哪些问题，等等。书总体上采用教程+参考的形式，教程部分只有 100 页左右，实际阅读时，先精读此部分，参考部分可以通过边干边学方式学习，精华尽在此中。

熟悉最近大红大紫的 Ruby on Rails 的读者，一定知道 Andrew Hunt 和 David Thomas 以及他们创办的 The Pragmatic Programmers 公司，他们撰写和出版的 *Programming Ruby* 和 *Agile Web Development with Rails* 两本书是 RoR 关键性的推动力量。而这个如今大名鼎鼎的公司的名字就出自《**程序员修炼之道**》一书（英文版原名即 *The Pragmatic Programmer*）。此书从各个方面来看都令人称奇：它当然是一本技术图书，但是字里行间弥漫着的浓浓的人文气息、哲理性和幽默感，又时常让我们产生疑惑；它的篇幅不大（只有 300 来页），然而几乎涉及了软件开发和程序员生涯的一切——责任心、学习方法、思考方法、沟通、设计原则、版本控制、代码生成、按契约设计、调试、测试、估算、并发、重构、需求、文档、各种工具……甚至包括如何发邮件、如何在 Windows 上使用 Unix 命令；内容层次跨度也极大，既可以高到团队组建，也可以深至代码级的具体细节，甚至还有不少编程习题！阅读此书的时候，常常会好奇，作者是何方神圣，能够将如此之多之丰富的内涵如此完美地熔于一炉。

《**敏捷软件开发**》则是另一位业界大师 Robert Martin（人称 Bob 大叔）的代表作品，Java 版

（也含有不少C++代码）荣获2003年Jolt大奖，去年年中又出版了C#版，由Robert与他的儿子Micah合作，主要改动除了将代码换成C#之外，还增加了UML建模、MVC模式等方面的内容，脉络更加清晰。此书以“敏捷”命名，其实有很大的误导性，它的副标题“原则、模式和实践”才算名副其实，估计这也是原版C#版书名改为*Agile Principles, Patterns, and Practices in C#*的原因。原因很简单，此书只用了100页左右的篇幅概述敏捷开发方法，主体部分主要是对面向对象诸原则和GoF模式的阐释，其实是一部非常优秀的面向对象设计、实现和设计模式图书（C#版还是很好的UML教程），语言通俗有趣，而且实战性很强。书中的许多绝妙插图，使我们在会意一笑中，更深地理解一些原本艰涩的技术概念。如果你阅读《设计模式》（机械工业出版社）一书感觉有困难，或者在阅读《设计模式解析》（人民邮电出版社）之后需要在实际开发环境中进一步领悟模式，那么本书将是绝佳选择。

经典之所以成为经典，既在于它们不仅是业界大师的作品，凝聚了软件开发社区集体多年摸索而获得的宝贵经验，拥有不因时光流逝而磨灭的价值；更因为它们很难一遍就领会其所有意蕴和精华，需要反复阅读，而且往往能够常读常新——某种意义上，你的阅读所得与你的经历是直接相关的，在不同阶段会有不同感受和收获，这也是前人说“少不读水浒”和“少不读杜甫”的原因。与此同时，经典的翻译往往难以尽善尽美，事实上这几部著作的中译本都多多少少存在各种问题，有的问题还非常严重。美国大诗人Robert Frost曾经说过，“Poetry is what gets lost in translation（诗歌就是在翻译中失去的东西）”，我们在长期的技术图书翻译和编审工作中也充分体会到，经典原著的许多关键的意境、暗示往往是难以用另外一种文字来表达的。有好的译本，当然能够起到事半功倍的作用，但是即便如此，直接或者对照着阅读原著仍然是有所裨益甚至必不可少的。而且，掌握英语，具备良好的英文技术文献阅读能力，也是今天平坦世界（读过《世界是平的》吗？）中一名专业软件开发人员必备的素养，而阅读名著原版，绝对是一种一举两得的英语进阶方式。

我们之所以在这几部著作大多出版多年（最早的原版初版于1999年），已经有了翻译版，甚至此前曾经还有过影印版行世的情况下，仍然下决心再次出版英文版，主要原因正是它们的经典性和长效性。事实上，这些书原版到现在确实依旧长销不衰，我们手上拿到的原版样书，无不已经是最近的印刷，印次达到10几次甚至20几次；在Amazon等主要的网络书店上，它们也仍然位居销售榜前列，让许多热门的新书后辈也难以望其项背；在巴诺和鲍德斯这样的大型连锁书店，你会看到这些著作依旧被摆放在最显著的位置，心中涌起一种感动。反观国内市场，它们的英文版几乎都已经绝版，更有中文版也绝版的奇怪现象。我们希望这批经典的出版，能够打破这一怪圈。

本次出版的英文注释版，除了按原版版权方的要求翻译了前言、序等文字，并原汁原味地保留了原书的正文部分之外，还在多位业界专家的大力支持下，利用页边和页脚的空白增加了一些注释，算是一种新的尝试，力求为读者能够提供更多的价值。

增加的注释主要是以下几种情况：

1. 直接注释单词。其目的就是方便读者，能够少查字典。事实上，我们在阅读（以及翻译）英语技术文献时，所遇到的直接困难并不是技术上的，而更多来自英语本身。因此我们所注释的词语或者语句，许多并不是术语，而是通用的单词或者习语。在注释时，我们并非简单地按字典的常见义项给出，而是充分参考原著的上下文，给出特定语境下的对应词。

2. 章节标题给出翻译，目录也改为双语对应。

3. 提示主题。其作用类似于一些图书页边的 recap（扼要重述或重点提示），有助于读者“在脑中读出目录”（这是 Ruby 专家 Obi Fernandez 对阅读技术图书的建议）来。

4. 知识更新和扩展。软件研发技术的发展日新月异，时光毕竟会在这些经典身上留下一些痕迹，我们力争根据注释时的最新技术进展为读者提供新的信息。这个工作并不容易，像《重构》这样作者一直维护着网站（即 refactoring.com）的实在不多，《企业应用架构模式》和《程序员修炼之道》还算有勘误，而《敏捷软件开发》的网页上只有书的一些摘录、评论和到 Amazon 的链接，连勘误都没有。好在，经典毕竟是经典，这方面需要做的工作总量倒是不大。

按我们最初的设计，如果能在注释版中增加一些导读性、读书心得式的文字，应该会更加完美，但是非常遗憾，由于工作量大、难以找到合适的人选、尺度很难把握等等原因，这次虽然做出了尝试，但并不令人满意。我们计划在图灵网站（www.turingbook.com）上提供类似的信息，构建一个“一起读经典”的社区阅读环境。分享是软件开发的原动力，我们期盼得到有更多的读者和业界专家能够以各种方式加入进来。

这种注释版是一种尝试，由于注释者水平和个人喜好各异，各本书的尺度也会有所差异，由于能力和时间问题，肯定会有各种各样的疏漏和讹误，欢迎大家批评指正。我们的报错信箱是：errata@turingbook.com。同时也可以在图灵网站各本书的配套网页上提交勘误。

图灵编辑部

2007年9月

序

写这篇序时，我刚刚交付了Eclipse开源项目的一个主要版本。我仍然处在恢复阶段，思维还有些模糊。但是有一件事情我却比以往更加清楚，那就是：交付产品的关键因素是人，而不是过程。我们成功的诀窍很简单：和那些全心致力于交付软件的人一起工作，使用适合于自己团队的轻量过程进行开发，并且不断调整。

看看我们团队中的开发人员，他们都将编程视为开发活动的中心。他们不仅编写代码，还努力参悟代码，以保持对系统的理解。使用代码验证设计，从中得到的反馈对于增强对设计的信心至关重要。同时，我们的开发人员理解模式、重构、测试、增量交付、频繁构建和其他一些XP（极限编程）最佳实践的重要性。这些实践改变了我们对开发方法的看法。

对于那些具有高技术风险以及需求经常变化的项目来说，熟练地掌握这种开发方式是取得成功的先决条件。虽然敏捷开发不注重形式和文档，但是非常强调重要的日常开发实践。让这些实践付诸实施，正是本书的中心内容。

Robert是久居面向对象社区的一位活跃分子，对于C++实践、设计模式以及面向对象设计的一般原则贡献颇多，同时他很早就是一位XP和敏捷方法的积极提倡者。本书就以他的众多贡献为基础，全面讲述了敏捷开发实践。这真是一项了不起的成就。不仅如此，Robert在说明每个问题时，还使用了案例和大量的代码，这与敏捷实践完全相符。他实际上是在通过实际编程来阐述编程和设计。

本书中充满了对于软件开发的真知灼见。不管你是想成为一位敏捷（agile）开发人员，还是想进一步提高自己的技能，它都同样有用。我对本书期盼已久，它没有令我失望。

Erich Gamma[①]
IBM公司杰出工程师

① Erich Gamma是面向对象技术大师，《设计模式》一书的第一作者。他与Kent Beck合作开发了测试框架JUnit。在IBM，他领导了Eclipse平台的开发。目前，他正在领导团队协作平台项目Jazz的开发。——注者注

前　言

可是Bob，你说过去年就能写完这本书的。

——Claudia Frers, , 1999年 *UML World* 大会

敏捷开发（Agile Development）就是指能够在需求迅速变化的情况下快速开发软件。为了达到这种敏捷性，我们需要使用一些实践提供必要的准则和反馈，需要使用一些设计原则使我们的软件灵活而且可维护，还需要理解一些已经被证明在特定问题中可以权衡这些原则的设计模式。本书试图将所有这3个概念融汇起来，使它们成为有机的整体。

本书首先描述了这些原则、模式以及实践，然后通过许多案例来演示如何应用它们。更重要的是，案例给出的并不是最终的结果，而是设计的过程。你会看到设计者犯错；你会看到他们是如何找到错误并最终改正；你会看到他们对问题的苦思冥想，面对一些需要权衡和含糊的问题的疑惑与探索。是的，你会看到设计的真正历程。

尽在细节中

本书包含了许多Java和C++代码，希望你能够仔细地阅读它们，因为在很大程度上，代码正是本书的精髓。代码是本书所讲内容的实际体现。

本书采用重复讲解的方式，由一系列不同规模的案例研究组成。有些案例非常小，有些案例则需要用几章来描述。每个案例研究之前都有一些预备材料。例如，在The Payroll案例研究之前，就有一些章节描述在该案例研究中将用到的面向对象设计原则和模式。

本书首先讨论了开发实践和过程，其中穿插了许多小的案例研究以及示例。然后，我们转移到设计和设计原则的主题上，接着是一些设计模式、更多管理包的设计原则以及更多的模式。所有这些主题都附有案例。

因此，请准备好学习一些代码和UML（统一建模语言）图。你将要学习的图书技术性非常强，其中要教授的知识就像恶魔一样，尽在细节中[①]。

本书简史

6年多前，我写了一本名为*Designing Object-Oriented C++ Application using the Booch Method*的书。它曾是我的代表作，其效果和销量都让我非常高兴。

这本书开始时是要作为*Designing*一书的第2版，但是结果却并非如此。书中所保留的原书内容非常少，只有3章多的内容，而这些章也进行了重大的修改，但书的意图、精神以及许多知识是相同的。自*Desinging*出版6年以来，在软件设计方面我又学到了非常多的知识，这些将会在本书中表现出来。

时间过得多么快啊！*Designing*一书刚好在因特网大爆炸之前出版。从那时起，我们使用的各种术语数量已经翻了一倍，诸如Design Patterns、Java、EJB、RMI、J2EE、XML、XSLT、HTML、ASP、JSP、Servlets、Application Servers、ZOPE、SOAP、C#、.NET，等等。必须承认，要使本书的内容跟得上最新技术潮流非常困难。

与Booch的关系

1997年，Booch与我联系，让我帮他撰写其非常成功的*Object-Oriented Analysis and Design with Applications*一书的第3版。以前，我和Grady在一些项目中有过合作，并且是他的许多作品（包括UML）的热心读者和参与者。因此，我高兴地接受了，并邀请我的好朋友Jim Newkirk来帮助完成这项工作。

在接下来的两年中，我和Jim为Booch的书撰写了许多章节。当然，这些成果意味着我不可能在这本书中按照我本来想的那样投入同样多的努力，但是我觉得Booch的书值得我这样做。另外，当时这本书完全只是*Designing*的第2版，并且我的心思也不在其上。如果我要讲些东西的话，我想讲些新的并且是不同的东西。

不幸的是，Booch著作的这个版本始终没有完成。在正常情况下已经很难抽出空写书了，在浮躁的.com泡沫期间，这就更加不可能了。Grady也更加忙于Rational以及一些像Catapulse这样的新企业的事务，因此这项工作就停止了。最后，我问Grady和Addison-Wesley是否可以把我和Jim撰写的那些章节包含在本书中，他们很有风度地同意了。于是，一些案例研究和UML的章节就由此而来。

极限编程的影响

1998年后期，XP崭露头角，它有力地冲击了我们所信奉的关于软件开发的观念。我们是应该在编写任何代码前先创建许多UML图？还是应该不使用任何种类的UML图而仅仅编写大量代码？我们是应该编写大量描述我们设计的叙述性文档？还是应该努力使代码具有自释义能力以及表达力，使辅助性的文档不再必要了？我们应该结对编程吗？我们应该在编写产品代码前先编写测试吗？我们应该怎么做呢？

---

① 原文为 The devils are in the details，谚语，相当于韩非子所说的“千里之堤，溃于蚁穴”，比喻细节之重要。——注者注

这场变革来得正是时候。在20世纪90年代的中后期，Object Mentor公司在面向对象（OO）设计以及项目管理问题上帮助了许多公司。我们帮助这些公司完成项目，在此过程中，我们慢慢地向这些公司灌输自己的一些观点和做法。遗憾的是，这些观点和做法没有被记录下来，它们只是我们对客户的口述。

到了1998年，我认识到需要把我们的过程和实践写下来，这样就可以更好地把它们传达给我们的客户。于是，我在*C++ Report*[①]上撰写了许多关于过程的论文，但这些文章都没有达到目的。它们提供了丰富的信息并且在某些情况下也很引人入胜，但是它们不是对我们在项目中实际应用的实践和看法的整理，而是对影响我数十年的价值观念的一种不经意的放弃。Kent Beck向我指出了这一点。

与Kent Beck的关系

1998年末，当我正为整理Object-Mentor过程烦恼时，我偶然看到了Kent在极限编程（eXtreme Programming，XP）方面的一些文字。这些文字散布在Ward Cunningham的wiki[②]中，并且和其他一些人的文字混合在一起。尽管如此，通过努力和勤奋，我还是抓住了Kent所谈论的要点。这激起了我极大的兴趣，但是仍有一些疑虑。XP中的某些东西和我的开发过程观念完全吻合，但是其他一些东西，比如缺乏明确的设计阶段，却令我迷惑不解。

我和Kent来自完全不同的软件环境。他是一个知名的Smalltalk顾问，而我却是一个知名的C++顾问。这两个领域之间很难相互交流。这之间几乎有一个库恩式的（Kuhnian）[③]范型（paradigm）隔阂。

在其他情况下，我绝不会邀请Kent为*C++ Report*撰写论文。但是我们关于过程认识上的一致填补了语言上的隔阂。1999年2月，我在慕尼黑的OOP会议上遇到了Kent。他在进行关于XP的讲演，而我在进行面向对象设计原则的讲演，我们的讲演场所正好面对面。由于无法听到他的讲演，我就在午餐时找到了Kent。我们谈论了XP，我邀请他为*C++ Report*撰写一篇论文。这是一篇很棒的论文，其中描述了Kent和一位同事在一小时左右的现场系统开发中所进行的彻底的设计改变。

在接下来的几个月中，我逐渐去除了自己对XP的担心。我最大的担心在于所采用的过程中没有一个明显的预先设计阶段，我对这一点有些犹豫。我不是一直在教导我的客户以及整个行业，设计非常重要，应该投入时间吗？

最后我认识到，实际上我自己也并不真正需要这样一个阶段。甚至在我所撰写的所有关于设计、Booch图和UML图的论文以及图书中，总是把代码作为验证这些图是否有意义的一种方式。在我所有的客户咨询中，我会先花费1～2个小时帮助他们绘制一些图，然后会使用代码来指导他

---

① 这些论文可以在 http://www.object.mentor.com “Resources” 中 “Published Articles” 部分找到。共有 4 篇。前 3 篇名为 “Iterative and Incremental Development I, II, III”。最后一篇名为：“C.O.D.E. Culled Object Development procEss”。

② http://c2.com/cgi/wiki。这个网站中包含数量众多的涉及各种各样主题的论文。其作者的数目成百上千。据说，只有 Ward Cunningham 才能仅用几行 Perl 就掀起一场变革。

③ 写于 1995～2001 年之间的任何可信的著作中肯定使用了术语 “Kuhnian”。它指的是 *The Structure of Scientific Revolutions* 一书，作者为 Thomas S. Kuhn，由芝加哥大学出版社出版于 1962 年。[库恩为美国科学哲学家，他提出的范型转换（paradigm shift）理论，影响颇大。1995~2001 年应指因特网大发展时期。——编者注]

们考查这些图。我开始明白，虽然XP关于设计的措词有点陌生（在库恩式[①]的意义上），但是这些措词背后的实践对我来说却很熟悉。

我关于XP的另一个担心相对比较容易解决。我私底下实际上一直是一个结对程序员。XP使我可以光明正大地和同伴沉醉于一起编程的快乐之中。重构、持续集成以及现场客户对我来说都非常易于接受。它们都非常接近于我先前对客户建议的工作方式。

有一个XP实践对我来说是新的发现。当你第一次听到测试优先设计时会觉得它似乎很平常。它只是要在编写任何产品代码前先编写测试用例。编写的所有产品代码都是为了让失败的测试用例通过。对于这种方式编写代码所带来的意义深远的结果，我始料未及。这个实践完全改变了我编写软件的方法，并把它变得更好了。在本书中，你可以看到这个改变。本书中有些代码编写于1999年之前，这些代码都没有测试用例。但是，所有编写于1999年之后的代码都带有测试用例，并且测试用例一般都首先出现。我确信你会注意到它们之间的差别。

于是，到1999年秋天，我确信Object Mentor应该采用XP作为自己的过程，并且我应该放弃编写自己过程的愿望。Kent在表达XP的实践和过程方面已经做了一项卓越的工作，相比起来我自己那些不充分的尝试就显得苍白无力了。

本书组织

本节由6大部分组成，其后跟有一些附录。

- Section 1：Agile Development。本部分描述了敏捷开发的概念。它首先介绍了敏捷联盟宣言，然后概述了极限编程（XP），接着讨论了许多阐明个别极限编程实践的小案例——特别是那些影响我们设计和编写代码方式的实践。
- Section 2：Agile Design。本部分中的章节谈论面向对象软件设计。前一章中提出了问题：什么是设计？它讨论了管理复杂性的问题及其技术。最后，阐述了面向对象类设计的一些原则。
- Section 3：The Payroll Case Study。这是本书中最大也是最完整的研究案例。它描述了一个简单的批量处理薪酬支付系统的面向对象设计和C++实现。本部分的前几章描述了该案例研究所用到的一些设计模式，最后两章包含了完整的案例研究。
- Section 4：Packaging the Payroll System。本部分开始描述了面向对象包设计的一些原则，接着通过增量地打包前一部分中的类来继续阐明这些原则。
- Section 5：The Weather Station Case Study。本部分中包含了一个起初打算用于Booch的书的案例。The Weather Station案例研究描述了一个做出一项重要商务决策的公司，并且阐明了Java开发团队对此是如何做出反应的。同样，本部分开始时描述了一些将会用到的设计模式，最后描述了设计和实现。
- Section 6：The ETS Case Study。本部分描述了我参与的一个实际项目。这个项目自1999年起就已经产品化。它是一个自动考试系统，用来对美国注册建筑师委员会的注册考试进行答题和评分。

① 如果你在一篇文章中提到两次Kuhn，就会得到更多的信任。

- UML Notation附录。前两个附录包含了几个用来描述UML表示法的简单案例研究。
- 其他附录

如何使用本书

如果你是开发人员……

请从头至尾地阅读本书。本书主要是写给开发人员的，它包含以敏捷方式开发软件所需要的信息。从头至尾阅读可以首先学习实践，接着是原则，然后是模式，最后是把它们全部联系起来的案例研究。把所有这些知识整合起来会帮助你完成项目。

如果你是管理人员或者业务分析师……

请阅读Section 1“Agile Development”。这部分中的章节深入讨论了敏捷原则和实践。内容涉及需求、计划、测试、重构以及编程。它会给你一些有关如何构建团队以及管理项目的指导，帮助你完成项目。

如果你想学习UML……

请首先阅读附录A“UML Notation I: The CGI Example”；接着阅读附录B“UML Notation II: The STATMUX”；然后阅读Section 3“The Payroll Case Study”。这种阅读方法在UML语法和使用方面会给你提供一个好的基础，它同时也会帮助你在UML和像Java或者C++这样的编程语言之间进行转换。

如果你想学习设计模式……

要想找到一个特定的模式，可以使用书后的“List of Design Patterns”，找到你感兴趣的模式。

要想在总体上学习模式，请阅读Section 2“Agile Design”学习设计原则，然后阅读Section 3“The Payroll Case Study”、Section 4“Packaging the Payroll System”、Section 5“The Weather Station Case Study”以及Section 6“The ETS Case Study”。这些部分定义了所有的模式，并且展示了如何在通常情形中使用它们。

如果你想学习面向对象设计原则……

请阅读Section 2“Agile Design”、Section 3“The Payroll Case Study”以及Section 4“Packaging the Payroll System”。这些章节将会描述面向对象设计的原则，并且展示如何使用这些原则。

如果你想学习敏捷开发方法……

请阅读Section 1“Agile Development”。这部分描述了敏捷开发，内容涉及需求、计划、测试、重构以及编程。

如果你只想笑一笑……

请阅读Appendix C“A Satire of Two Companies”。

致 谢

衷心感谢以下人士：

Lowell Lindstrom、Brian Button、Erik Meade、Mike Hill、Michael Feathers, Jim Newkirk、Micah Martin、Angelique Thouvenin Martin、Susan Rosso、Talisha Jefferson、Ron Jeffries、Kent Beck、Jeff Langr、David Farber、Bob Koss、James Grenning、Lance Welter、Pascal Roy、Martin Fowler、John Goodsen、Alan Apt、Paul Hodgetts、Phil Markgraf、Pete McBreen、H. S. Lahman、Dave Harris、James Kanze、Mark Webster、Chris Biegay、Alan Francis、Fran Daniele、Patrick Lindner、Jake Warde、Amy Todd、Laura Steele、William Pietr、Camille、Trentacoste、Vince O□rien 、Gregory Dulles、Lynda Castillo、Craig Larman、Tim Ottinger、Chris Lopez、Phil Goodwin、Charles Toland、Robert Evans、John Roth、Debbie Utley、John Brewer、Russ Ruter、David Vydra、Ian Smith、Eric Evans、硅谷Patterns Group中的每一个人，Pete Brittingham、Graham Perkins、Philp以及Richard MacDonald。

本书的审稿人为：

Pete McBreen / McBreen Consulting	Bjarne Stroustrup / AT & T Research
Stephen J. Mellor / Projtech.com	Micah Martin / Object Mentor Inc.
Brian Button / Object Mentor Inc.	James Grenning / Object Mentor Inc.

非常感谢Grady Booch和Paul Recker允许我在本书中包含原本用于Grady的*Object Oriented Analysis and Design with Applications*第3版中的那些章节。

特别感谢Jack Reeves，他慷慨地允许我全文引用他的论文“什么是设计”（What Is Software Design？）。还要特别感谢Erich Gamma为本书做序。Erich希望这次的字体会好一些！

每章开头处美妙、偶尔还有些炫目的插图是Jennifer Kohnke绘制的。散布在章节中间的装饰插图是Angela Dawn Marin Brooks的可爱作品，她是我的女儿，也是我生活中的快乐之一。

资 源

本书中的所有源代码都可以从www.objectmentor.com/PPP下载。

Contents