TURING 图灵程序设计丛书

PRENTICE HALL

Agile Principles, Patterns, and Practices in C#

敏捷软件开发
原则、模式与实践（C#版）

[美] Robert C. Martin
Micah Martin 著

邓辉 孙鸣 译

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

Agile Principles, Patterns, and Practices in C#

敏捷软件开发
原则、模式与实践（C#版）

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 : C#版 / (美) 马丁 (Martin,R.C.) , (美) 马丁 (Martin,M.) 著 ; 邓辉, 孙鸣译. -- 2版. -- 北京 : 人民邮电出版社, 2010.12
(图灵程序设计丛书)
书名原文: Agile Principles,Patterns, and Practices in C#
ISBN 978-7-115-23997-6

Ⅰ. ①敏… Ⅱ. ①马… ②马… ③邓… ④孙… Ⅲ. ①软件开发②C语言－程序设计 Ⅳ. ①TP311.52②TP312

中国版本图书馆CIP数据核字(2010)第189042号

内 容 提 要

本书中，享誉全球的面向对象技术大师Robert C. Martin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。

本书Java版曾荣获2003年第13届Jolt大奖，是公认的经典著作。本书是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。

图灵程序设计丛书

敏捷软件开发：原则、模式与实践（C# 版）

◆ 著 [美] Robert C. Martin Micah Martin
译 邓 辉 孙 鸣
责任编辑 杨海玲
执行编辑 李 静

◆ 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
北京艺辉印刷有限公司印刷

◆ 开本：800×1000 1/16
印张：35
字数：1010千字 2010年12月第2版
印数：7 001－10 500册 2010年12月北京第1次印刷
著作权合同登记号 图字：01-2007-1482号
ISBN 978-7-115-23997-6

定价：79.00元

读者服务热线：(010)51095186 印装质量热线：(010)67129223
反盗版热线：(010)67171154

版 权 声 明

业界大师鼎力推荐

“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，*Programming C#* 作者

“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

译 者 序

2002 年 10 月，“Bob 大叔”（Robert C. Martin）终于推出了软件开发社团期待已久的 *Agile Software Development, Principles, Patterns, and Practices* 一书。该书以真实案例为基础，通过真实开发场景再现的方式对软件开发中涉及的各种知识及其有效的运用方法进行了讲解。这种做法得到了广大软件从业人员的一致认可。该书一出版就好评如潮，并毫无争议地获得了第 13 届软件开发图书类的 Jolt 大奖。

次年，“Bob 大叔”又推出了另外一本书 *UML for Java Programmers*。该书秉承了上一本书的讲解风格，不过其重点在于 UML。“Bob 大叔”在书中介绍了一些常用的 UML 特性；更为重要的是，他把重点放在了如何在真实的项目开发中，以注重实效的态度来使用 UML。对于那些习惯于动辄画数十页精美的 UML 图，并把这些 UML 图当成真正的软件设计的架构师们来说，该书无疑是对他们的当头棒喝。

应该说，这两本书中所教授的内容和思维方法是与具体编程语言无关的，但是许多软件开发人员还是很希望这些知识能够基于自己特定的语言和开发平台进行讲解。作为一名资深的软件咨询大师，“Bob 大叔”当然很清楚这一点。为了让.NET 开发社团也能够像 Java 社团那样，学习到这些可以改善软件开发状况，并让程序员感受到开发乐趣的敏捷开发和敏捷设计的权威知识，“Bob 大叔”于 2006 年 7 月推出了一本新书 *Agile Principles, Patterns, and Practices in C#*，也就是读者正在阅读的这本书。按照“Bob 大叔”的说法，这本书是他前两本书的合订本。

在本书中，“Bob 大叔”去除了前两本书中的重复内容，并把它们有机地融合在一起。此外，对于书中的案例也做了相应的调整，去掉了不为大多数开发人员熟悉的气象站案例以及略显仓促的 ETS 案例。“Bob 大叔”对具有典型代表性的薪水支付应用案例进行了增强，使其贯穿全书，并增加了关于数据库和 MVP 模式两章内容使得该案例更加完整。这种做法使得本书读起来更加顺畅。读者花一本书的价格买到“Bob 大叔”的两本经典著作，从某种意义上来讲可以看作是“Bob 大叔”对.NET 社团作出的补偿。☺

能够在 4 年后再次翻译“Bob 大叔”的这本新书，对我个人而言，既是一种荣幸，同时也是对这几年敏捷开发实践的一次难得的反思、总结以及再学习的机会。曾经有读者认为本书中讲的东西太多、太杂，很多内容完全可以独立成书，放在一起显得比较散乱。我不认同这种观点。敏捷开发的核心就是以最低的成本，最快速地为客户提供价值。书中所讲述的过程方法、实践、设计原则、模式以及思考方式看似独立，其实都围绕在这个核心周围，并以相互支援的方式为达成这个核心目标服务。为了能够快速提供价值，我们应用短迭代、快速交付的开发方法；为了保证这些价值是客户真正需要的，

我们和客户紧密合作并应用反馈驱动的方法；为了能够降低软件的演化和维护成本，我们应用好的设计原则和模式；为了降低设计成本，我们采用测试驱动、随时重构、演化设计的方法……如果能够以这个核心为主线去理解和学习书中教授的内容，效果应该会更好一些。我这几年的实践经历也证实了这一点。

软件开发应该是一项充满快乐和激情的工作，衷心希望本书能够帮助国内.NET社团的程序员朋友体会到这种快乐和激情。

邓 辉

Chris Sells 序[1]

在我的职业编程生涯中，所做的第一份工作是为一个 bug 数据库增加功能。这些功能主要是为明尼苏达大学农场校区的植物病理系服务的，因此这里的“bug”指的是真正的 bug，比如蚜虫、蝗虫以及毛虫。数据库的代码原来是由一位昆虫学家编写的，他所掌握的 dBase 知识仅仅能够编写出一种类型的表格，然后就在应用的其余部分到处复制。在我增加功能时，我把功能尽可能地集中在一起，这样就可以在一个地方修正代码的 bug，并且也可以在一个地方增加功能。这项工作花费了我整整一个夏天，最终的功能是原来的两倍，但是代码规模却只有原来的一半。

许多年后，我和我的一位朋友因手边没有什么急迫的工作要做，所以决定一起编一些程序（编写的要么是 IDispatch，要么是 IMoniker[2]，当时我们认为这两个东西都很重要）。我先编写一会儿代码，而他在边上观看，并告诉我哪里写错了。接着，他掌控键盘，而我在旁边提建议，然后他把键盘的控制权又交给我。就这样持续了几个小时，这是我最为满意的一段编码经历。

之后不久，我的朋友就聘用我来担当他的公司新成立的软件部门的首席架构师。作为架构工作的一部分，我经常会为一些还没有存在的对象（我假想它们已经存在）编写客户代码，并把代码移交给工程师，由他们来继续实现直到客户程序能够工作。

我猜我对敏捷开发方法各个方面的实践体验并非个例。总的来说，我在敏捷方法（比如重构、结对编程以及测试驱动开发）方面的实践是成功的，虽然我对自己所做的还没有非常清楚的认识。当然，在这之前我是可以获取一些敏捷开发方面的资料的，但是正如我不愿意从《国家地理杂志》的过期刊物中学习如何邀请女孩跳舞一样，我更希望敏捷技术能够适合于我的特定情况，也就是.NET。和那些费尽心思去学习学生中间的流行语的中学老师一样，Robert 使用.NET（即使他很清楚地指出，.NET 在很多方面并不比 Java 优秀）在讲述我使用的语言，但他知道内容本身要比传达介质更重要。

除了.NET 外，我还喜欢在尝试新东西时，能够循序渐进、逐步深入，不至于感到恐惧，但是又可以真正理解所有重要的东西。而这正是 Bob 大叔（Robert Martin）在本书中所做的。他的介绍性章节讲述了敏捷运动的基础知识，但没有急于向读者提及 SCRUM、极限编程以及任何其他的敏捷方法，从而使读者能够以一种自己喜欢的方式来逐步进行理解。更好的是（也是 Robert 写作风格中我最喜欢的部分），他是通过实践来展示这些技术的：提出一个问题，分析它，就像发生在真实环境中一样；展现出错误和失策的地方以及如何通过应用他所主张的技术来解决这些问题。

---

① Chris Sells 是世界知名的.NET 技术专家。曾被微软授予“软件传奇人物”（Software Legend）称号。代表著作有《Windows Forms 程序设计》（人民邮电出版社，2004）等。——编者注

② IDispatch 和 IMoniker 都是微软 COM 模型中的接口名。——编者注

我不知道 Robert 在本书中描述的情况在现实中是否存在。我只是在自己的经历中曾经隐约有过这样的体验。但是，可以肯定的是，所有比较“酷”的年轻人都在这样做。请把 Bob 大叔当作自己在敏捷世界中的优秀导师，他的唯一目标就是当你想去体验时，能够让你做好，并且保证每个人都享受其中的乐趣。

Erich Gamma 序[①]

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认识。

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。

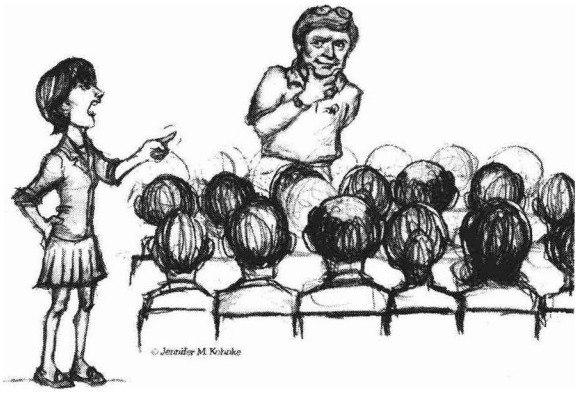
Robert 是面向对象社区的一位活跃分子，对于 C++开发、设计模式以及面向对象设计的一般原则贡献颇多，同时他很早就是一位 XP 和敏捷方法的积极提倡者。本书就以他的众多贡献为基础，全面讲述了敏捷开发实践。这真是一项了不起的成就。不仅如此，Robert 在说明每个问题时，还使用了案例和大量的代码，这与敏捷实践完全相符。他实际上是在通过实际编程来阐述编程和设计。

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。

---

① Erich Gamma 是 IBM 公司杰出工程师，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。此外，他还领导团队协助开发了平台项目 Jazz。——编者注

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

Bob 的导言

离Claudia说出这句合情合理的抱怨已经7年了，不过我觉得我已经做出了补偿。在这几年里，我出版了3本书，对于一个同时经营着一家咨询公司，并且还得进行大量的代码编写、培训、指导、演讲的工作，以及撰写文章、专栏和博客的人来讲，要每隔一年出一本书是一项很大的挑战，更不要说还得养活并陪伴一个大家庭了。但是，我喜欢这样。

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保持灵活且可维护，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误；你会看到他们如何找到错误并最终改正；你会看到他们对问题苦思冥想，面对一些难以权衡的含糊问题的疑惑与探索。是的，你会看到设计的真正历程。

Micah 的导言

2005年初，我参与到一个小的开发团队中，该团队正准备用C#开发一个.NET应用程序。使用敏捷

开发实践是团队的强制性规定，这也是我参与其中的原因之一。虽然我以前曾经使用过C#，但是我的大部分编程经验都是基于Java和C++的。我认为.NET不会有什么不同，结果也表明确实如此。

项目开始两个月后，我们进行了第一次发布。这是一次部分发布，其中只包含了所有计划特性中的一部分，但却是完全可用的，并且也确实被投入使用。仅仅两个月公司就得到了我们开发的软件带来的好处。管理层非常兴奋，他们要求雇佣更多的人，这样就可以启动更多的项目。

我投身到敏捷社区已经有几年了，我认识很多可以帮助我们的敏捷开发者。我通知了他们每一个人，请求他们加入到我们中来。结果，没有一个敏捷朋友加入我们的团队。为什么？也许最主要的原因是我们是基于.NET进行开发的。

几乎所有敏捷开发者都具有Java、C++或者Smalltalk方面的背景。几乎从来没有听说过有敏捷.NET程序员。也许，当我说我们正在使用.NET进行敏捷软件开发时，我的那些朋友根本就没当回事，也许他们想避免和.NET有什么瓜葛。这是一个严重的问题。我已经不止一次看到这种情况了。

我讲过许多为期一周的关于各种软件主题的课程，有机会见到来自世界各地的具有广泛代表性的开发者。我曾经指导过的很多学生都是.NET程序员，也有很多是Java和C++程序员。恕我直言：在我的经历中，.NET程序员常常要比Java和C++程序员差一些。显然，也并非总是如此。但是，通过在课堂中的再三观察，我只能得出这样的结论：在敏捷软件实践、设计模式、设计原则等方面，.NET程序员往往要弱一些。在我的课堂上，.NET程序员常常从来没有听说过这些基本概念。*必须改变这种情况。*

本书的另一版本，由我父亲Robert C. Martin撰写的*Agile Software Development: Principles, Patterns, and Practices*在2002年末出版，并赢得了2003年的Jolt大奖。那是一本很好的书，得到了许多开发者的赞扬。遗憾的是，它对.NET社区几乎没有提供什么帮助。尽管书中的内容同样适用于.NET，但是几乎没有.NET程序员读过它。

我希望这本.NET版本能够充当.NET社区和其他开发者社区之间的桥梁。我希望程序员能够阅读它并看到更好的构建软件的方法。我希望他们开始使用更好的软件实践、创建更好的设计并提升.NET应用的质量标准。我希望.NET程序员可以和其他程序员一样好。我希望.NET程序员能够在软件社区中获得新的地位，这样Java程序员就会以加入.NET团队为荣。

在完成本书的整个过程中，对于是否把我的名字放在一本与.NET有关的图书的封面上，我有过多次思想斗争。我曾问自己是否要把名字和.NET联系在一起，并承担可能由此带来的所有负面后果，但现在我不再迟疑了。我是一名.NET程序员。不！是一名敏捷的.NET程序员。我以此为荣。

关于本书

本书简史

20世纪90年代初，我（Bob）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。

这本书最初想作为*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章内容，即使这3章也进行了大量的修改，但书的意图、精神以及许多知识是相同的。自*Desinging*出版10年以来，在软件设计和开发方面我又学到了非常多的知识，这些将在本书中表现出来。

十年过去了！*Designing*刚好在因特网大发展之前出版。从那时起，我们使用的缩略词的数量已经翻了一倍，诸如EJB、RMI、J2EE、XML、XSLT、HTML、ASP、JSP、ZOPE、SOAP、C#、.NET以及设计模式、Java、Servelet和应用服务器。我要告诉你，要使这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，我和他在一些项目中有过合作，并且是他的许多作品（包括UML）的热心读者和参编者。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，这些工作意味着我不可能按照我本来想的那样投入大量精力写作本书，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来写书了，在浮躁的.com泡沫时期，就更加不可能了。Grady忙于Rational以及Catapulse等新公司的事务。因此这项工作就停止了。最后，我问Grady和Addison-Wesley公司是否可以把我和Jim撰写的那些章节包含在本书中，他们很慷慨地同意了。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，极限编程（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画许多UML图呢？还是应该不使用任何UML图而仅仅编写大量代码？我们是应该撰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，免除撰写辅助性文档的必要？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？我们应该做什么呢？

这场变革来得正是时候。在20世纪90年代中后期，Object Mentor公司在面向对象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*上撰写了许多关于过程的论文[①]，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，但是它们没有系统地反映我们在项目中实际应用的实践和看法，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（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]范型隔阂。

---

① 这些论文可以在http://www.object.mentor.com的*publications*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。这个网站中包含有数量众多的涉及各种各样主题的论文，论文作者成百上千。人们都说，只有Ward Cunningham才能使用几行Perl煽动一场社会革命。

③ 写于1995~2001年的任何可信的学术作品中肯定使用了术语*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。[库恩是美国著名科学史家和哲学家，在其代表作《科学革命的结构》一书中提出了“范型转换”（paradigm shift）理论。——编者注]

如果不是看到他的观点，我绝无可能邀请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，我们的会场所正好面对面。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*撰写了一篇论文。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，我逐渐消除了自己对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，我对此有些犹豫。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项目中，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熟悉。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驱动开发[①]（TDD）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。

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。

.NET

在几个大公司之间一直在进行着一场战争。这些公司在为了赢得你的忠诚而战。这些公司相信，如果它们拥有了语言，那么它们将拥有程序员以及雇佣这些程序员的公司。

首先打响这场战争的是Java。Java是第一个由大公司创造的用来赢得程序员的编程语言。结果取得了极大的成功。Java确实深深地扎根于软件开发社团中，并成为现代多层IT应用开发的事实标准。

对此的还击之一来自IBM。IBM通过Eclipse开发环境占领了大部分的Java市场。另外一个对此的重大阻击来自微软的一些追求完美的设计者，他们给我们提供了通用的.NET平台和特定的C#语言。

令人惊异的是，很难对Java和C#做出区分。这两个语言在语义上是等同的，并且语法也非常相似，以至于许多代码片段都可以写得很相似。所有在技术创新上的不足，微软都通过其在能力上的卓越表现进行了超额的补偿，并赶超上来，赢得胜利。

本书的第一版是使用Java和C++作为编程语言编写的。本书使用了C#和.NET平台。不要把这看作是对某一方的支持。我们不会在这场战争中拥护某一方。事实上，我认为当几年后一种更好的语言出现并占领了参与交战的公司花费巨大代价赢得的程序员意向份额时，这场战争就会自行结束。

本书的.NET版本只是为了能够影响到.NET程序员。虽然本书中的原则、模式和实践与语言无关，但是案例研究却与语言相关。正如.NET程序员更加乐于阅读.NET案例研究一样，Java程序员则更加乐

① Kent Beck, Test-Driven Development by Example, Addison-Wesley, 2003。

于阅读Java示例。

尽在细节中

本书包含了许多.NET代码。希望你能够仔细阅读它们，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，其中讲述了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研究。

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]。

本书的内容结构

本书由四个部分和两个附录组成。

第一部分：敏捷开发。本部分描述了敏捷开发的概念。首先介绍了敏捷联盟宣言，然后提供了对极限编程（XP）的概述，接着讨论了许多阐明个别极限编程实践的小案例，特别是那些影响设计和编写代码方式的实践。

第二部分：敏捷设计。本部分中的各章谈论了面向对象软件设计：什么是面向对象软件设计，管理复杂性的问题以及技术，面向对象类设计的一些原则。本部分最后几章讲述UML实用子集。

第三部分：薪水支付案例研究。它描述了一个简单的批量处理薪水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会用到的一些设计模式。最后一章包含了完整的案例研究，这也是本书中最大和最完整的一个案例。

第四部分：打包薪水支付系统。本部分开始描述面向对象包设计的一些原则。接着，通过增量地打包上一部分中的类来继续阐明这些原则。本部分最后讲述薪水支付应用的数据库和UI设计。

接下来是两个附录：附录A，“双公司记”；附录B，Jack Reeves的文章“什么是软件”。

如何使用本书

如果你是一名开发人员，请从头至尾阅读本书。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。

如果你是一名管理人员或者业务分析师，请阅读第一部分“敏捷开发”。第1章～第6章提供了对敏捷原则和实践的深入讨论。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，请首先阅读第13章~第19章。然后，阅读第三部分“薪水支付案例研究”的所有章节。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，同时也会帮助你在UML

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。——编者注

和C#语言之间进行转换。

如果你想学习设计模式，请先阅读第二部分“敏捷设计”学习设计原则，然后阅读第三部分“薪水支付案例研究”、第四部分“打包薪水支付系统”。这几部分定义了所有的模式，并且展示了如何在典型的情形中使用它们。

如果你想学习面向对象设计原则，请阅读第二部分“敏捷设计”、第三部分“薪水支付案例研究”以及第四部分“打包薪水支付系统”。这些章节将会描述面向对象设计的原则，并且向你展示如何使用这些原则。

如果你想学习敏捷开发方法，请阅读第一部分“敏捷开发”。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，请阅读附录A“双公司记”。

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 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、Jessica D'Amico、Chris Guzikowski、Paul Petralia、Michelle Housley、David Chelimsky、Paul Pagel、Tim Ottinger、Christoffer Hedgate以及Neil Roodyn。

非常感谢Grady Booch和Paul Becker允许我在本书中使用原本用于Grady的*Object-Oriented Analysis and Design with Applications*第3版中的章节。特别感谢Jack Reeves，他慷慨地允许我全文引用他的论文“什么是软件设计”（What Is Software Design？）。

每章开头处美妙的、偶尔还有些炫目的插图是Jennifer Kohnke和我的女儿Angela Brooks绘制的。

目　录

第一部分　敏捷开发