TURING 图灵程序设计丛书 程序员修炼系列

PRENTICE HALL

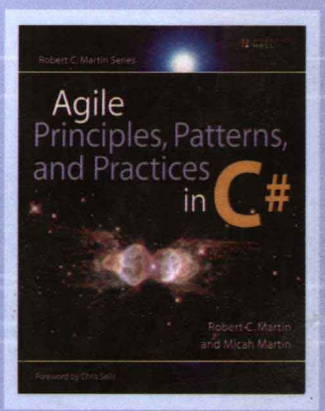
Agile Principles, Patterns, and Practices in C#

敏捷软件开发

原则、模式与实践 C#版·英文注释版

本书Java版曾荣获
2003年度Jolt大奖

[美] Robert C. Martin
Micah 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 Principles, Patterns, and Practices in C#

敏捷软件开发

原则、模式与实践 C#版·英文注释版

[美] Robert C. Martin
Micah 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（C# 版 英文注释版）
/（美）马丁（Martin, R. C.），（美）马丁（Martin, M.）著；
—北京：人民邮电出版社，2008.1
（图灵程序设计丛书）
ISBN 978-7-115-16507-7

Ⅰ.敏… Ⅱ.①马… ②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 101782 号

内 容 提 要

本书旨在指导.NET程序员学习构建软件的最佳实践，从而创建更好的设计并提升.NET应用的质量。书中使用真实案例讲解如何用极限编程来设计、测试、重构和结对编程，包含了极具价值的可重用的C#源代码，还重点讲述了如何使用UML和设计模式解决面向客户系统的问题。

本书适于软件开发和管理人员提高自身水平学习之用，也适于用作高校计算机专业本科生、研究生以及软件学院的软件工程和软件开发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（C#版·英文注释版）

◆ 著 [美] Robert C. Martin Micah Martin
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：48
字数：922 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：01-2007-2670 号

ISBN 978-7-115-16507-7/TP

定价：89.00 元

读者服务热线：(010)88593802 印装质量热线：(010)67129223
反盗版热线：(010)67171154

敏捷软件开发宣言

我们正在通过亲身实践以及帮助他人实践，揭示更好的软件开发方法。通过这项工作，我们认为：

人和交互	重于	过程和工具
可以工作的软件	重于	面面俱到的文档
客户合作	重于	合同谈判
随时应对变化	重于	遵循计划

虽然右项也有其价值，但我们认为左项更加重要。

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler
James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick
Robert C.Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

敏捷宣言遵循的原则

我们遵循以下原则：

- 我们最优先要做的是通过尽早地、持续地交付有价值的软件来满足客户需要。
- 我们欢迎需求的变化，即使到了开发后期。敏捷过程能够驾驭变化，为客户创造竞争优势。
- 经常交付可以工作的软件，从几个星期到几个月，时间间隔越短越好。
- 在整个项目开发期间，业务人员和开发人员必须朝夕工作在一起。
- 围绕斗志高昂的人构建项目。给他们提供所需的环境和支持，并且信任他们能够完成任务。
- 在团队内部，最有效率也最有效果的信息传达方式，就是面对面的交谈。
- 可以工作的软件是进度主要的度量标准。
- 敏捷过程提倡可持续开发。出资人、开发者和用户应该总是保持稳定的开发速度。
- 对卓越技术和良好设计的不断追求有助于提高敏捷性。
- 简单——尽量减少工作量的艺术是至关重要的。
- 最好的构架、需求和设计都源自自我组织的团队。
- 每隔一定时间，团队都要总结如何更有效率，然后相应地调整自己的行为。

版 权 声 明

对本书以及Java版的赞誉

“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大叔当作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

“本书以实例为本，不尚空谈，因此格外真实，摄人心魄。……选材匠心读到，精彩绝伦，特别是还有大量独创性和创新性的技术。”

——孟岩，《程序员》技术主编

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，印次达到十几次甚至二十几次；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“在脑中读出目录来”（这是 Ruby 专家 Obi Fernandez 对阅读技术图书的建议）。

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　一

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进行功能的增加。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[1]，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，这是我最为满意的编码经历之一。

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

1. IDispatch 和 IMoniker 都是微软 COM 模型中的接口名。——注者注

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；然后给出错误和失策的地方，并讲述如何通过应用他所主张的技术来解决这些问题。

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。

Chris Sells[1]

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。现任微软分布式系统组的项目经理。——注者注

序　二[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[2]
IBM公司杰出工程师

---

1. 这是 Erich Gamma 为《敏捷软件开发：原则、模式与实践》一书所写的序。——注者注
2. 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

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。

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*刚好在因特网大爆炸之前出版。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创建许多UML图呢？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？我们应该做什么呢？

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（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？

---

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。它的作者的数目成百上千。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。——注者注]

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精心设计者，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，以至于对许多代码片段无法做出辨别。所有在技术创新上的缺乏，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++作为编程语言进行编写的。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

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