PEARSON
Addison Wesley

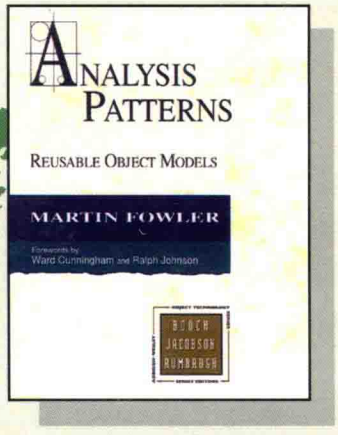
典藏原版书苑

Analysis Patterns
Reusable Object Models

[英] Martin Fowler 著
陈 师 注释

分析模式
——可复用的对象模型

注释版



人民邮电出版社
POSTS & TELECOM PRESS

典藏原版书苑

分析模式——可复用的对象模型

（注释版）

[英] Martin Fowler 著

陈师 注释

人民邮电出版社

北京

图书在版编目（CIP）数据

分析模式——可复用的对象模型（注释版）/（英）福勒（Fowler, M.）著；陈师注释. —北京：人民邮电出版社，2007.11
（典藏原版书苑）
ISBN 978-7-115-15619-8

Ⅰ. 分… Ⅱ. ①福…②陈… Ⅲ. 面向对象语言—程序设计—英文 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2007）第 114916 号

版 权 声 明

典藏原版书苑

分析模式——可复用的对象模型（注释版）

◆ 著 ［英］Martin Fowler
注 释 陈 师
责任编辑 刘映欣

◆ 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
北京铭成印刷有限公司印刷
新华书店总店北京发行所经销

◆ 开本：800×1000 1/16
印张：25
字数：549 千字 2007 年 11 月第 1 版
印数：1－3 000 册 2007 年 11 月北京第 1 次印刷
著作权合同登记号 图字：01-2006-3666 号

ISBN 978-7-115-15619-8/TP

定价：79.00 元

读者服务热线：(010)67132705 印装质量热线：(010)67129223

内容提要

本书讲述各种分析模式和支持模式，专注于面向对象分析与设计的结果——模型本身，给出了来自金融贸易、测量、财务以及组织关系等多个领域内的一系列模式。书中每个模式都包含了设计背后的原理、使用的规则以及实现的技巧，给出的例子包含了有用模型的细节，并透析了用于提高分析、建模和实现的重用技巧。

本书的注释内容并不是对原文的简单摘译，注释者从中文理解的角度对复杂的内容进行翻译，站在专业人员的角度对术语进行解释，并对书中关键内容作了相关的扩展，对领域概念进行了阐述。

出版说明

正如软件工程名家 Roger S. Pressman 所言，“软件工程从少数拥护者所实践的朦胧的思想，演化成一个正式的工程学科。今天，它已经被承认为一个值得认真地研究、细心地学习和热烈地争论的主题。”

在计算机专业书籍中，软件工程领域的许多经典可能既是最易阅读的，又是最需要花费时间去领悟的。它既不富含烦琐的形式化推导，亦无特定的运行环境，更不充斥确定的源代码，读者可以一目十行地顺畅浏览软件工程书籍中大段的文字描述，不会因为时而出现的代码片段或公式阻碍阅读速度，然而读毕掩卷，有时一片茫然，犹如入宝山而空手归，有时满怀感想，却一时理不出头绪。此刻，如果能在书边看到行家的注解或点评，无论是旁敲侧击，还是当头棒喝，常能有拨云见日、豁然开朗之效。中国古典文化有注疏传统，将这类述而不作的经典诠释方式结合到新兴的技术学科中，未尝不是一种有益的探索。

由于软件工程著作之精妙之处常在概念和方法，故而多半都是自然语言的文字论述，而并不以程序语言为载体，这使得对软件工程著作的翻译相比其他领域更富挑战性。有如文学作品的翻译，往往难以顾全“信、达、雅”，软件工程经典的译者，也很少能有自信声明自己的译文在保持技术准确的同时流畅地传递了原文的语言个性——很多思维的微妙之处，都在原著文字的起承转合和字里行间。为原文提供中文注释，在技术上的启发意义之外也能扫清一些语言和文化差异造成的障碍，让更多力所能及的读者都能方便地直接欣赏原文，窥经典之全豹。

另一方面，正如 Roger S. Pressman 说：“软件工程将发生变化——对此我们可以肯定。” Frederick P. Brooks 在《人月神话》中所倡导的“唯一不变的是变化本身”，人们只有在变化中才能体味永恒。在当前的实践和技术氛围中咀嚼软件工程的理论，这样才是这一学科真正的活力源泉。注释者结合软件工程近年发展趋势，重访当年的经典，令读者能在软件技术发展的脉络中领会经典的精粹。借用 UMLChina 团队的口号“软件以用为本”，经典终究是为实践服务的，而我们出注释版图书最终的目的也是希望通过国内相关领域专业人士的注释，令英文原版符合国内读者的阅读需求，为原文增添技术上的辅助、语言上的答疑，以及抛砖引玉，激发读者的思辨。

注释者序

正如 Ralph Johnson 所说，模式是对客观规则的总结，而不是凭空创造的产物。本书的作者作为一名信息系统对象建模的顾问，凭借其深厚的项目建模经验，总结出一系列具有普遍适用性的领域模型，并称之为分析模式。作者的知识背景决定了该书不但包含大量的面向对象分析与设计的内容，还具有多种业务领域的概念。因此本书所面对的读者主要是：对象分析与设计人员和领域知识专家。针对这两大读者群，对本书作了以下三个方面的注释。

首先是对书中不易理解的内容作阐释。书中大量运用了复杂的从句句型，或包含作者自创的专业术语，增加了阅读的难度。注释者从中文理解的角度对复杂的内容进行翻译，并站在专业人员的角度对术语进行解释。

其次是对书中关键内容作相关的扩展。作者对某些问题提出自己的解决方案，注释者也根据自己的专业经验提出更多可能的选项，增强了读者的想像空间。书中某些内容的阅读前提是假设读者具有相关专业基础，例如，虽然本书的主题分析模式与设计模式没有直接联系，书中却提到多种设计模式的概念。注释者对这些背景知识作了一定程度的描述，更易于那些恰巧不具备这些专业基础的读者进行理解。

最后是对领域概念进行解释。书中内容涵盖了金融、医疗、财务等多个业务领域，无论是对象分析设计人员还是某个领域的专业人员都无法具备如此多的行业背景，而理解领域概念又是理解领域相关模型的前提条件，因此注释者对那些不太普及的领域知识进行了解释。

对于英文专业书籍，国内有的读者喜欢阅读中文译本，他们认为用自己熟悉的语言更容易吸收和理解书中的内容，同时却担心中文版的“保真度”。具有一定英文阅读能力的读者更喜欢阅读英文原版，不但可以享受“原汁原味”的快感，更能 read between lines，体会书中深层次的内涵，然而由于与作者语言文化以及知识背景的差异，导致了内容无法很好地理解等问题，类似的问题在阅读时经常困扰着他们。注译版以英文原版为主体，以中文注释内容为辅助，希望能最大程度的减小读者的阅读难度。

本书中有多种不同风格的注释内容，其中一些以小标题加以提示。本书的注释约定包括以下几类。

中文目录及每一章的中心思想

在原书目录之后给出了全书的中文目录，并提炼了每一章的中心思想，给出该章的整体线索，以便在读者阅读时有清晰的阅读思路。

核心理念

提炼出技术核心点和阅读关键点，对其进行详细说明和相关扩展，从而为读者在实际工作中遇到的问题提供更多的解决策略和更大的发挥空间。

术语解读

注释时从专业人员的角度对书中专业术语的出处、意义以及用法进行阐述和解释。

背景知识

书中内容往往涵盖多个业务领域，注释时对不太普及的领域知识以及书中提及的国内读者不熟悉的艺术隐喻、历史典故以及国外的谚语进行解释说明。

触类旁通

原书中的内容对读者的启迪，以及对当今技术发展的现实意义。

当然，注释内容不可能解答所有读者的疑惑，甚至还可能在一定程度上与读者的理解产生冲突。如有不妥之处，敬请批评指正。

陈　师

国内最早的“责任驱动设计（RDD）”理念推广者
在《计算机工程与设计》、《中国工业年鉴 2005》、“第六届 Web-age 信息管理国际论坛 2005”等计算机期刊或国际论坛上发表过多篇相关文章

Foreword

When the "Gang of Four" was writing *Design Patterns*, we knew that there were lots of software patterns other than object-oriented design patterns. By the time we were through with the book, we had seen distributed programming patterns, user interface patterns, and even patterns of organizing software development groups. However, we hadn't seen any patterns that were clearly object-oriented analysis patterns. Peter Coad's patterns were the closest, but they were a lot like our patterns and it seemed to us that pure analysis patterns should differ more.

I found what I was looking for when I read a draft of Martin Fowler's book, *Analysis Patterns*. Its patterns contain a lot of domain knowledge yet can be used in all kinds of business software. Like the design patterns, they are abstract enough to help your software ride over the bumps of requirement changes but concrete enough to be understandable. They are not the most obvious solutions to modeling problems, yet they rang true to me. I had seen many of these solutions before, and they had worked.

I'm a designer more than a modeler, and I don't have a lot of experience in most of the domains that Martin Fowler describes. Though I felt the patterns were good, I couldn't have a lot of confidence in my feelings. Since I read the book, I have been trying out the patterns on projects and using them in teaching. They work! My confidence grew further when I ran across David Hay's book, *Data Model Patterns*, and realized that, despite their different backgrounds and vocabularies, they saw many of the same patterns. Patterns are supposed to describe reality, not invent a new one, and Martin Fowler accurately described the patterns in object-oriented models of business software. You can have confidence in the patterns he described.

This is not a book of principles that you must learn to apply before they can help you, though Martin describes many modeling principles. It is not a book that you have to read through and practice before it can do you any good. It is a book full of practical patterns that you can use right away. Look for the chapters that match the kind of problem you are working on now, and you will find lots of ideas that will help you. You can read the book chapter by chapter, and each chapter will give you new ideas.

To make the most of this book, you need to know two things. First, many of the patterns are more powerful than they might appear at first. Patterns like Accountability can be applied in nearly any project. Don't read only the chapters that obviously apply to your project, but learn as many patterns as you

can, and try them out to see whether they apply. Second, make sure your coworkers read the book. One of the biggest advantages of patterns is that they help us communicate better. You will find that your team meetings will run more smoothly when you have a common vocabulary. This book will make documentation more consistent and easier to understand. Plus, it will make your coworkers better analysts, and it is more fun to work with people who do a good job!

— Ralph Johnson

Foreword

When I look at a software development project, I look for experience. Does the development team have experience doing relevant work? Can they apply their experience to the objects they build? Unfortunately, the answer to these questions is often no.

A growing number of us in the object-oriented development community feel we have misplaced our collective attention for some time. We no longer need to focus on tools, techniques, notations or even code. We already have in our hands the machinery to build great programs. When we fail, we fail because we lack experience.

Martin Fowler has found a way to give us what we need: experience in book form.

He has done for domain objects what Eric Gamma et al. did for implementation objects in their landmark work *Design Patterns: Elements of Reusable Object-Oriented Software*. Martin uses the familiar terminology of our nascent community but in a different way. He uses the word *pattern*, for example, not because he's duplicating or extending Gamma's book (or any of the other new titles bursting onto the market). He calls his written form of experience patterns simply because that is what they are. In his work as a consultant in object modeling information systems, he repeatedly found solutions to recurring problems, and discovered the pattern form in the process.

Martin Fowler easily could have written a book on object-oriented analysis. Luckily, he didn't. Instead we have a book cataloging the result of analysis. Each chapter reports the conclusion of his (and his colleagues') analytic efforts applied to common business problems. The domains addressed vary from medical record keeping to financial derivative trading, with several stops in between. Which chapters apply to you? Amazingly, they all do. Martin places each problem in a context and then offers a solution for that context. You will see familiar aspects in every context. You will recognize the problems. You will appreciate the results. And there it is: experience.

Finally, Martin writes in a personal style, relaying his thoughts and judgments. We feel his respect for his clients and colleagues from whom, he admits, most insights arise. We watch him keep his distance from the vagaries of implementation while still preserving implementability—a tightrope walk that defies direct explanation. As we see into the mind of an expert analyst, we gain a lesson in the how-to of analysis that adds to our own store of experience.

—Ward Cunningham
Cunningham & Cunningham, Inc.

前　言

以前，要寻找一本关于面向对象分析与设计的专业书籍，简直让人踏破铁鞋。而现在，这类书籍如此之多，却又让人眼花缭乱。大多数该类的书籍着重于通过一些简单的例子来介绍某种建模的形式语言，或者推荐某种简便的建模流。本书却不然，其重心不在于流程——建模的方式，而在于流程的结果——模型本身。

作为一名专业的信息系统领域的对象模型顾问，客户通常会要求我对他们的技术人员进行建模方面的培训或项目指导。我的知识大多来自对建模技巧的领悟和运用，更重要的是，我在创立模型过程中所积累的经验使我能够快速定位一些重复出现的问题。事实上我经常发现大多数项目所面临的问题都是我以前所遇见过的，这样我丰富的经验使我毫不费力地重用以前所创立的模型，并加以改进使之更适应于当前项目。

近年来，越来越多的人意识到了这种现象。人们也意识到一些典型的方法论书籍，虽然有其价值，但尚处于初级阶段，它们所介绍的流程还是与具体项目相关，抽象度不高。在这种情况下，模式运动慢慢开展起来。大量具有不同背景、持有不同观点的人们聚在一起，然而他们却具有一个共同的目标，即在软件系统内推广可重用的模式。

由于模式社区成员的多样化，对于“模式”一词，我们很难给予一个统一的定义。不可否认，当遇到某种模式的时候，我们能够识别它，然而却很难给它下一个准确的定义，而我的定义是：模式是一套在某种现实背景下有用的理念，而且该理念很有可能在其他相似背景下也有效。

该定义并不严谨，是因为我不想加入过多的修饰，而让“模式”失去本色。一种模式会有多种形式，每种形式都为该模式增加了独有的特色。（1.2 节就讨论了模式世界当前的状态，以及该书在模式世界应处的地位。）

该书主要讨论的是项目分析过程中的模式——一种反映业务流程概念结构的模式，而不是软件实现的模式。很多章节讨论了多种不同业务领域中的模式。很难将这些模式归类到传统的基础行业（制造业、金融业、医疗保健业等）当中，因为很多模式通常可以适用于多个行业。这些模式的重要之处在于它们有助于我们更好地理解人们认知世界的方式。

我们建立相应的计算机系统模型来反映人们认知的世界，甚至改进人们认知的世界——这就是所谓的业务重组（BPR）的来由。

概念模式无法孤立存在，概念模型仅仅对知道如何实现它们的设计工程师才有价值。在本书中，我将介绍了一些能够将概念模型转化为软件系统的模式，并讨论如何使这些软件适应大型信息系统的构架，当然还将讨论一些模式实现过程中的实用技巧。

一开始我就非常想阅读有关该方面内容的书籍，因此我撰写此书。模型设计师在新的领域中开展工作的时候，能够在本书中找到一些灵感。模式所包含的是模型、模型背后的原理以及应用这些模型的时机。有了这些信息，模型设计师就能够应用相应的模型处理特定的问题。

该书中的模式同样有助于重新省视模型，看看有什么遗留的问题，或推荐一些替代方案进行改进。重新省度一个项目时，我通常将其与以前应用过的模式进行比较。我发现应用模式使得我能够更加容易地应用过去的成功经验，事实上这本身也是一种工作模式，其所揭露的建模问题远远超过一本简单的书所能够涵盖的内容。通过讨论为什么要对事物进行建模，我们更能够理解如何改进模型，即使我们并不会直接使用到这些模式。

本书的结构

本书分为两大部分。第一个部分介绍各个分析模式，这些模式都来自概念业务模型。这些模式都从贸易、度量、审计以及组织关系等领域中抽象出来。这些模式是概念化的，因为其所代表的是人们看待业务的方式，而不是计算机系统的设计方式。第一部分的章节讨论了多种可相互替换的模式，并讨论了每种选项的优缺点。尽管每种可选模式对于相应领域来讲都是有利用价值的，但最基础的模式往往还可应用于其他多种领域。

本书的第二部分着力于讨论支撑模式，这些内容更有助于读者运用分析模式。支撑模式所描述的是分析模式如何适应一个大型信息系统的构架，如何将概念模型转换为软件接口及实现，以及一些高级模型结构与基础架构之间的关系。

我需要特定的符号和形式语言来描述这些模式。在附录中对我所用的符号的意义进行了解释。在描述过程中，我喜欢混合使用各种描述技术，然而附录并不是这些技术的教程，而是一个为读者阅读本书提供一个纲领，让读者有更新的认识，同时它还给出了如何寻找这些技术的相关教程。

本书的两个部分都包含多个章节。分析模式部分中的每个章节都包含了一些与某种目标领域相关的模式，而这些目标领域也是衍生至某些具体的项目。这种结构关系说明模式其实是源于某种现实背景的。模式将出现在各小节中，我并不像其他模式作者（见 1.2.2 节）那样为模式使用正式的标题。我以合理且贴近原始项目的形式来描述每种模式，当然其中需要加上些许抽象。我将用一些实例来说明模式是怎样运用于起始领域的，以及对如何将该模式应用于其他领域提出相关建议。模式的难点在于如何将其抽象并应用于其他领

域，我的原则是将选择的权力留给读者（见 1.2.3 节）。

本书更像是目录，而不是一本必须逐页阅读的书。在撰写过程中，我尽量使得每个章节的内容相互独立。（然而要做到完全独立也不太可能，总有些章节的内容以其他章节的内容为基础，我会在章节介绍中进行说明。）每个章节都有一个章节介绍，用以概括本章内容所涵盖的目标领域、总结本章中的模式，以及衍生模式的原始项目。

如何阅读本书

我建议读者先通读第一章，然后阅读每一章的章节介绍，继而根据个人喜好选择任意章节进行精读。若读者对于我的建模方式或使用的符号和概念不了解，可以参考附录 A。附录 B 中的“模式表”是对书中所有模式的简单介绍，它有助于读者浏览所有模式，或寻找特定的模式。需要强调的一点是，尽管书中每个模式都源自特定的领域，但是其绝对可以应用于该领域之外。因此，有些章节可能与读者感兴趣的领域看似并无相关，但是我鼓励读者阅读这些章节。例如，为医疗保健领域而设计的观察和度量模式被证明在企业金融分析领域中同样有效。

本书的读者对象

本书适用于一定范围的读者，不同的读者从中得到的体会各自不同，因此在阅读之前也要做不同的准备。

我所期待的是该书所面对的主要读者应是计算机系统面向对象（OO）的分析师和设计师，特别是专注于分析领域的分析师。这些读者已经应用过一些面向对象分析设计（OOAD）的方法。如果是 OOAD 领域的新人，建议先阅读一本关于 OOAD 的书籍。必须强调，书中的模式是非常概念化的（贴近于自然现实），而我所应用的建模方式也非常之概念化。这与那些描述基于软件实现的建模方法的书籍在风格上有所不同。

该书还面向一批数量较小但却非常重要的读者——建模工程中的领域专家。这些读者无需具有计算机知识，然而却不能不了解概念建模。我之所以应用概念模型的一个最主要的原因是，这样更易于被领域专家所接受。很多建模项目是为计算机系统开发或业务重组（BPR）服务的。我教过很多专业人士（包括医生、金融业务员、会计、护士、人力资源主管等）这种建模方式，发现计算机背景对于理解概念建模并无影响。本书介绍的业务模型模式就同时包含了业务建模知识和计算机系统分析知识（见 1.4 节），因此面向这方面内容的读者有必要先参加一个着重于概念方面的面向对象分析课程（Odell 的书[1]在该方面颇具价值）。

我相信很多程序员也会阅读本书，尽管一些程序员也许会因为该书缺乏实用代码或过于概念化而忽略它。对于程序员读者，建议仔细阅读第 14 章，并做些相关笔记，这将有

助于他们了解概念模型与软件实现之间的关系。

这是一本面向对象的书籍，我坚信面向对象方法是软件开发的最佳选择。尽管书中的模型主要是概念模型，而应用概念（或逻辑）模型是很多数据建模师的一贯传统。数据建模师会发现书中的很多模式都非常有用，特别是当他们能够用更先进的语义技术时更是如此。模型的面向对象特征揭露了面向对象方法与传统方法的区别。建议这些读者将此书与一本着重介绍概念建模以及面向对象与语义数据建模之间关系的面向对象分析书籍结合起来阅读。

管理者们会发现，在启动一个开发项目时，该书同样有效。以模式来启动项目有助于明确项目目标，而且可以利用模式搭建的底盘开展项目计划。

该书的目标人群没有包括学生，此书更多的是为那些专业软件工程师所撰写。尽管如此，我还是希望一些学生读者能够浏览此书。当我刚开始学习分析与设计时，发现很难找到来自学校以外的好的学习实例。就像研读好的代码有助于提高编程技巧一样，研读好的模型同样有助于分析和设计的学习。

一本“活”的书

我所认识的每一位作者曾经经历过同样的挫败感：一本书一旦出版，其内容就无法再改变。书将这些固定内容传播给读者，即便作者对书中的内容有更新的见解却也无能为力。我深深理解不断的学习过程也是人的理念不断改变的过程，我同样希望读者也能够意识到这种变化。

为此，我在网站 http://www.aw.com/cp/fowler.html 上将实时更新本书的内容，使其更具有生命力。现阶段，我还无法确定该网站的具体内容，不过以下内容是值得期待的：

- 不断学习书中模式所得到的任何新的观点；
- 解答读者有关本书的疑问；
- 关于模式的任何有益的评论；
- 新的分析模式；
- 当有一套统一的建模符号产生时，我将用这种统一符号重新绘制书中的图，并将它们在该网站上更新。

致谢

任何作者在图书的编撰过程中都得到过他人的帮助，而对于本书尤其如此，因为书中的多数模式都是在我的客户、同事以及朋友的协助下创建的。我要借此向他们表示真诚的感谢。

首先我要感谢 Jim Odell，我事业上的良师益友。他向我传授了大量信息系统的开发知识，并且是我不断获得灵感、建议以及幽默感的源泉。可以说，没有他的帮助，本书也不会面世。

伦敦 Coopers & Lybrand 团队在早期给予我很大的支持，在 Smithfield 我们一起度过了很多不眠之夜。

John Edwards 在我形成概念建模的思路以及意识到其在软件开发领域中的角色的初期，对我影响很深，同时也是他向我介绍了包括 Christopher Alexander 在内的很多名家的经典思想。

是 John Hope 让我领悟到领域优先于技术的思维方式，并在我事业的几个重要时期提出了非常有益的建议。

Tom Cairns 和 Mark Thursz 是伦敦圣玛利亚医院的两位医生，他们曾和我一起开发医疗保健模型，而这些模型正是组成第 2 章、第 3 章以及第 8 章的基础内容。他们证明了一个毫无计算机背景的人同样能够成为顶级的概念建模师。Mark 还自愿为我们提供了多个著名的医疗保健案例。

医疗保健项目还得到了许多来自圣玛利亚医院、儿童医院、圣托马斯医院以及威尔士大学的软件和医务专业人员的帮助。在来自儿童医院的 Anne Casey 护士和分析员 Hazim Timimi 的帮助下，Cosmos 模型最终能够建立起来，而 Gerry Gold 是该项目的发起者和推动者。

Brad Kain 对于我形成重用以及组件的思想影响颇大，同时他还承担了一个重要的任务：向我介绍 Boston 的美丽夜景。

将医疗保健模型应用于企业金融领域（在第 4 章中有详细描述）是我的一次成功经验，它证明了分析模式具有跨领域性。感谢 Vivek Salgar 以及 Lynne Halpin 和 Craig Lockwood 领导的来自施乐公司的团队，正是他们用 C++实现了我们概念模型的原型。

感谢来自花旗银行（Citibank）的 David Creager、Steve Shepherd 及其领导的团队，我和他们一起开发的模型组成了第 9~11 章中的金融模式。他们还从原始的医疗保健项目中提取了大量构架思想，这些思想构成了本书的第 12 章。他们还教会了我许多狂热的城市生活方式。

Fred Peel 是我在花期银行工作时的主管，他的管理风格令我感到非常的舒适。而来自 Valbecc 的 Daniel Poon 和 Hazim Timimi 的帮助让我把模糊的思想转化为细节的规范。

第 6 章中的审计模式经过一个较长的酝酿期。Tom Daly、Peter Swettenham、Tom Hadfield 及其各自领导的团队开发了大量的模型，正是在此基础上才有了本书的审计模式。Rich Garzaniti 帮助我将审计模式中的专业术语分类。Kent Beck 为我的 Smalltalk 程序做了大量的工作。

本书的第 14 章是在 James Odell 的帮助下完成的，对以上给予过我帮助的人们表示衷

心的感谢。

对于模式社区而言，我仅仅是一个后辈，直到本书快要完成的时候我才对模式有了更深层次的理解。模式社区是一个开放而友好的组织，并对我的工作给予了很多的激励。Kent Beck、Ward Cunningham 以及 Jim Coplein 鼓励我加入社区，并将自己的想法发展为模式。Ralph Johnson 更是为本书的第一版草稿提出了宝贵的建议。

本书的很多审阅者都给出了一流的建议，在此我要特别感谢：Dave Collins、Ward Cunningham（Cunningham 本人及其 Cunningham 有限公司）、Henry A. Etlinger（罗彻斯特理工学院计算机科学系）、Donald G. Firesmith（Knowledge System 公司）、Erich Gamma、Adele Goldberg、Tom Hadfield（TesserAct Technology 公司）、Lynne Halpin（网景信息）、Brian Henderson-Sellers、Neil Hunt（Pure Software）、Ralph E. Johnson（伊利诺伊大学香槟分校）、Jean-Pierre Kuilboer（马萨诸赛大学波士顿分校）、Patrick D. Logan（Intel 公司）、James Odell、Charles Richter（Objective Engineering 公司）、Douglas C. Schmidt（华盛顿大学）以及 Dan Tasker。特别要指出的是 Don Firesmith 负责收集了以上审阅者所提出的需要改进的问题。

这是我的第一本书，我要特别感谢 Addison-Wesley 的工作人员帮助我完成了出版流程。Carter Shanklin 在 Angela Buenning 的协助下负责组织了高质量的审阅团队。Teri Hyde 在进度紧张的情况下协调了本书的出版任务 。Barbara Conway 对不恰当内容的斧正最终确保了本书的质量。

参考文献

1．Martin，J.，J. Odell Object-Oriented Methods: A Foundation Englewood Cliffs NJ：Prentice-Hall，1995

Contents