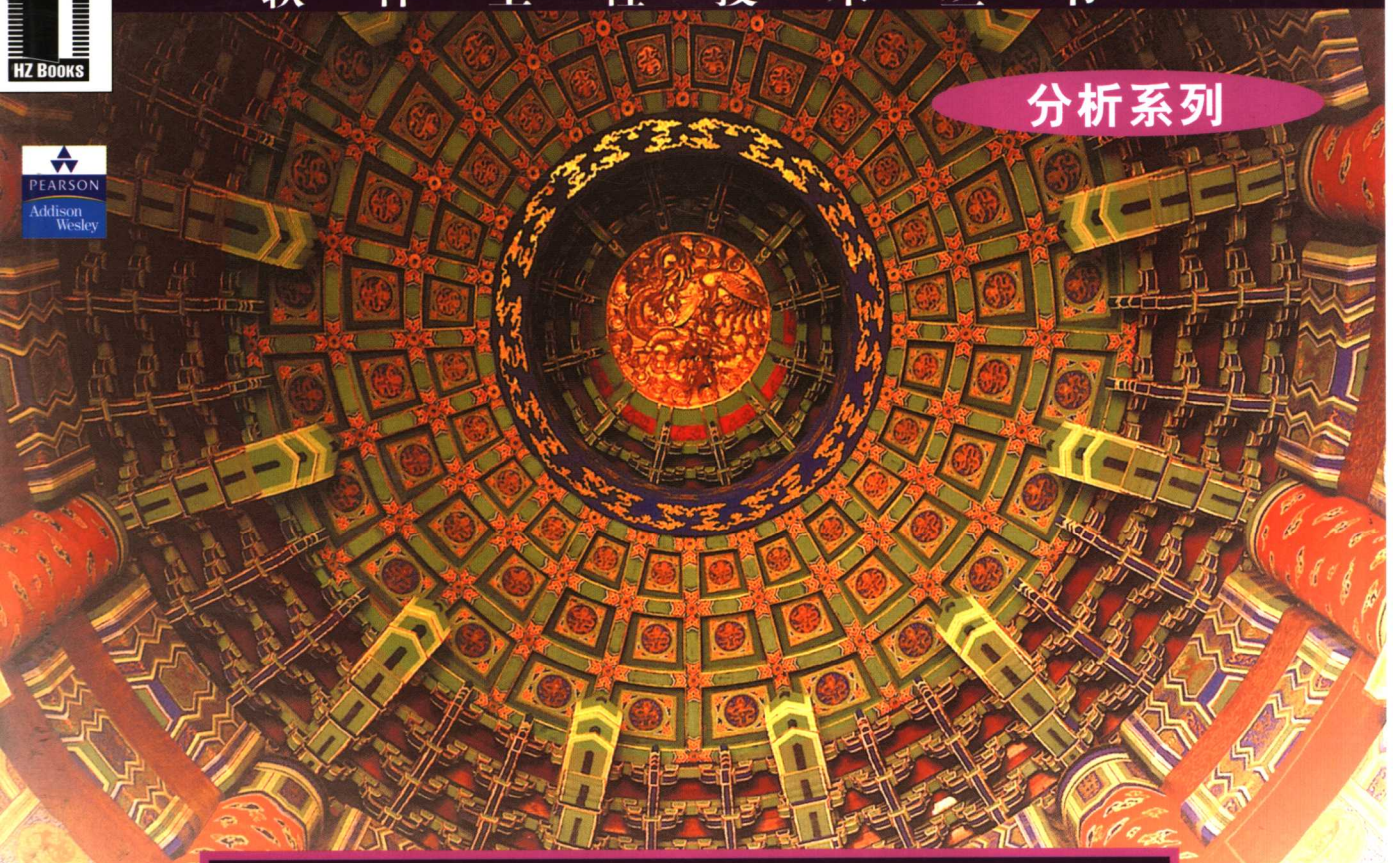
软 件 工 程 技 术 丛 书

HZ BOOKS

PEARSON
Addison Wesley

分析系列



# 分析模式
## 可复用的对象模型

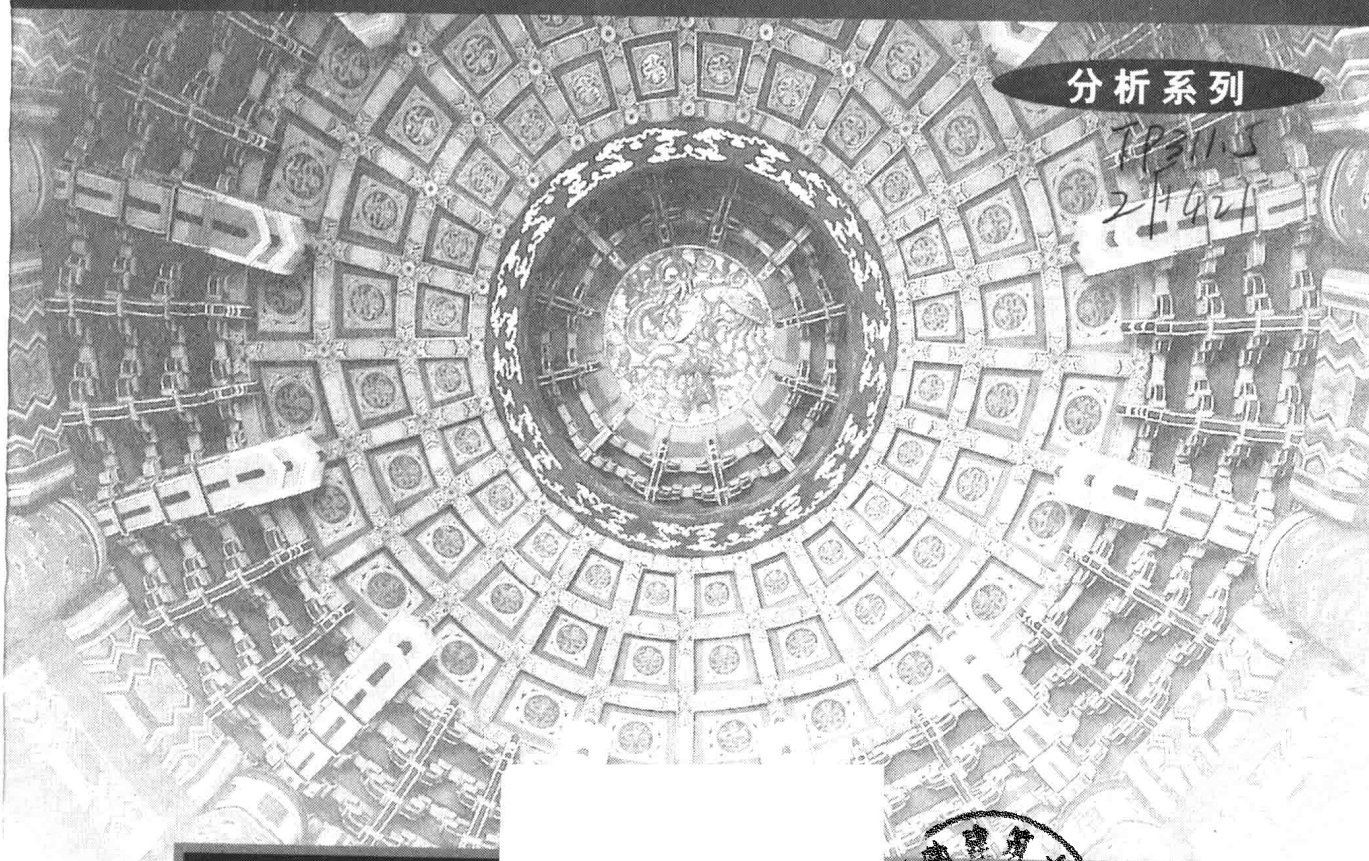
# Analysis Patterns
## Reusable Object Models

（英） Martin Fowler 著
樊东平 张路 等译

机械工业出版社
China Machine Press

软 件 工 程 技 术 丛 书

分析系列



# 分析模式

## 可复用的对象模型

Analysis Patterns

Reusable Object Models

（英） Martin Fowler 著

樊东平 张路 等译

机械工业出版社
China Machine Press

本书的作者Martin Fowler是国际著名的OO专家，敏捷开发方法的创始人之一，现为ThoughtWorks公司的首席科学家，本书是作者的代表作之一，深受业界专业人士和广大读者的好评，经久不衰。

本书讲述各种分析模式（即来自概念性业务模型的模式）和支持模式（即讲述如何使用分析模式的辅助性模式），把论述重点放在介绍面向对象分析和设计的最终结果——即模型本身。作者透过平实朴素的语言，将自己丰富的对象建模经验与读者分享，使读者可以马上采纳这些经验性模式。

本书适合的读者范围非常广：面向对象的计算机分析人员和设计人员（尤其是那些参与系统分析的人员）、数据建模人员、编程人员以及专业的软件工程师都可以从本书中获得宝贵的知识和经验。



**本书版权登记号：图字：01-2001-4784**

**图书在版编目（CIP）数据**

分析模式：可复用的对象模型 /（英）福勒（Fowler, M.）著；樊东平等译. -北京：机械工业出版社，2004.1

（软件工程技术丛书　分析系列）

书名原文：Analysis Patterns: Reusable Object Models

ISBN 7-111-13301-3

Ⅰ.分…　Ⅱ.①福…　②樊…　Ⅲ.面向对象语言－程序设计　Ⅳ.TP312

中国版本图书馆CIP数据核字（2003）第100324号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）

责任编辑：温莉芳　姚　蕾

北京昌平奔腾印刷厂印刷·新华书店北京发行所发行

2004年1月第1版第1次印刷

787mm×1092mm 1/16·21印张

印数：0 001-5 000册

定价：40元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：（010）68326294

# Ralph Johnson序

当我们“四人帮”在写《设计模式》(*Design Patterns*)一书时，就知道除了面向对象设计模式外，还存在大量的软件模式。在阅读本书以前，我们已经了解了分布式编程模式、用户界面模式，甚至还见过如何组建软件开发小组的模式。然而，我们还没有见过能够清楚地表述面向对象分析的任何模式。Peter Coad的模式最接近于分析模式，可是这些模式却跟我们的设计模式非常相似，而我们认为纯粹的分析模式应该不是这样。

当我在阅读Martin Fowler的这本《分析模式》的手稿时，我找到了自己一直想要的东西。书中所提到的分析模式虽然包含了大量的领域知识，但适用于所有的业务软件。像设计模式一样，这些分析模式很抽象，足以帮助你的软件适应需求的变化；同时它们又非常具体化，很容易理解。它们不是最显而易见的建模问题解决方案，然而我认为它们是正确的。我以前曾经见过许多这样的解决方案，它们都发挥了很好的作用。

我首先是一个设计人员，其次才是一个建模人员，而且我对Martin Fowler所描述的领域也没有多少经验。起初，虽然我感觉这些模式非常好，但对于这种自我感觉缺乏足够的信心。自从阅读了本书以后，我就在项目开发和教学实践中试用了这些模式，它们果真发挥了作用！当我偶然看到David Hay的《数据模型模式》(*Data Model Patterns*)一书时，就更增强了信心，并且意识到，虽然Martin Fowler和David Hay具有不同的背景且所用的词汇不同，但是他们所总结的模式却非常相似。模式应该是总结已存在的客观事物，而不是凭空创造一个新事物。Martin Fowler准确地阐述了业务软件面向对象模型中的各种模式。他所描述的这些模式是值得信赖的。

虽然Martin Fowler在本书中阐述了许多建模原则，但你并不需要再花时间去学习如何应用这些原则。它也不是一本必须先加以通读然后才能给你的实践带来裨益的书。它是一本充满可以马上采纳的经验性模式的书。找到你当前所要解决问题的对应章节，将发现其中的许多思想会给你带来帮助。也可以逐章阅读本书，每一章都会给出新的启发。

为了更好地使用本书，需要明白两件事情。首先，大部分模式的实际效果都比看起来要好。像责任这样的模式可以用在几乎所有项目中。不要只阅读那些跟你的项目明显相关的章节，应该了解尽可能多的模式，并试试看它们是否适用。第二，确保你的同事也阅读过本书。模式的最大好处之一就是它们可以帮助我们更好地交流。你会发现当你与你的同事拥有共同的词汇后，小组会议会开得更加顺畅。本书可以帮助你们把文档组织得更加连贯一致，且更易于理解。此外，本书也会把你的同事培养成更优秀的分析人员，而同优秀人员共事本身就是一件非常愉快的事情！

——Ralph Johnson[⊖]

⊖ Ralph Johnson是《设计模式》的四位作者之一。《设计模式》的中文版和英文影印版已由机械工业出版社出版。——编辑注

# Ward Cunningham序

当我面对一个软件开发项目时，首先看重的是经验。软件开发小组成员是否拥有相关的工作经验？他们能否将自身的经验有效地应用到他们所创建的对象中？不幸的是，答案往往是否定的。

越来越多的像我们这样从事面向对象开发的人员都感觉到：已经有一段时间我们把集体的注意力放错了地方。事实上，我们已不再需要把注意力放在诸如工具、技术、图符甚至是代码上，我们已经掌握了建造复杂程序的机制。如果我们失败了，那仅仅是因为缺乏经验。

Martin Fowler发现了一种给予我们所需东西的途径：那就是以书本的形式总结并介绍经验。

他研究了领域对象，就像Eric Gamma等人在他们的代表作《设计模式》中研究了实现对象一样。Martin Fowler使用了与其相似的术语，但是方法却不一样。比如，他使用“模式”一词，并不是在抄袭或者扩充Eric Gamma的著作（或当前市面上突然出现的那些新书）。他把经验的书面形式称为“模式”，只不过是因为那就是它们本来的名称。作为一个信息系统对象建模方面的顾问，他在工作中再三地发现反复出现的问题的解决方案，并发现在该过程中的模式的组成形式。

Martin Fowler本来可以很容易地写一本有关面向对象分析的书。但是，他并没有那样去做。其结果是我们拥有了一本对分析结果进行分类的书。书中每一章都汇集了他（以及他的同事）对通常性业务问题的分析工作的结论。涉及的领域很广，从医疗记录的保存到金融衍生交易，中间还经历了几个阶段。哪一章适于你呢？令人惊讶的是，所有章节都很有用。Martin Fowler将每个问题置于特定的上下文环境中，然后给出相应的解决方案。你会在每种上下文环境中看出相似的地方，你也将识别出问题之所在。更重要的是，你将获得意外的收获，那就是：经验！

最后，Martin Fowler采用一种较为个性化的书写方式，来表达他的想法和见解。我们可以感受到他对客户和同事的尊重，他承认绝大部分灵感都是受他们（客户和同事）启发的结果。我们察觉他保持了与实现的反复无常的行为的距离，同时兼顾了可实现性，而且拿捏得恰到好处。其实，当我们考察一个分析专家的想法时，我们主要向专家学习的是如何进行分析的思路和方法，以便进一步丰富我们自己的经验。

——Ward Cunningham
Cunningham & Cunningham, Inc.

# 前　言

不久前，还没有任何有关面向对象分析和设计的书籍。而现在，却有如此之多的书籍，以致于任何一个专业人员都无法全部涉猎。其中大部分书籍都专注于：传授一种图符表示法，提出一个简单的建模过程，并用几个简单的示例来加以说明。本书是一本与它们完全不同的书。它并不把重点放在过程——即如何建模，而是把重点放在过程的结果——即模型本身。

我是一个信息系统对象建模方面的顾问。客户常常聘请我训练他们的员工如何建模和为他们的项目提供指导。我的大部分技能来自对建模技术以及如何运用这些技术的了解。然而，更重要的是我的实际经验，这些经验是在建造许多模型和经常分析重复出现问题的过程中积累起来的。我经常发现项目在很多方面会遇到以前我曾面对的同样问题。这些经验使得我可以重用以前所建造的模型，我只需要对这些模型加以改进，使之适应新的需求。

在过去的几年里，越来越多的人已经意识到这一现象，并且认识到那些通常介绍方法论的书籍虽然很有价值，但都只提出了学习过程的第一步，而这个学习过程还必须捕获要被建模的实际事物本身。这种认识逐渐发展成为“模式”运动，在这一运动中汇集了各种各样的人，他们有着不同的兴趣和观点，却抱着共同的目标，即传播有用的软件系统模式。

由于这个模式群体构成的多样性，我们很难给“模式”一个准确的定义。我们中的所有人都相信，一旦我们看到一个模式，就能辨别出它；我们认为我们在大多数情况下是一致的，但我们无法给出一个简单的定义。我对模式的定义是：**模式是一种问题解决思路，它已经适用于一个实践环境，并且也可能适用于其它环境。**

我喜欢给出一个宽松的定义，因为我希望能尽可能地接近模式研究的初衷，而不需要增加太多限制性的内容。模式可以有多种形式，而每一种形式增加了对于该模式有用的特性（1.2节讨论有关模式研究的现状以及本书所处的地位）。

本书讨论的是分析方面的模式，这些模式反映的是业务过程的概念架构，而不是实际的软件实现。绝大部分章节讨论不同业务领域的模式。这些模式很难按照传统的行业（如制造、金融、医疗保健等）进行分类，因为它们通常可用在多个领域。这些模式非常重要，因为它们可以帮助我们了解人们对世界的认识。基于这样的认识去设计计算机系统并确实去改变这种认识是非常有价值的，而认识中需要改变的地方正是需要进行业务过程重组（BPR）的地方。

然而，概念模式不可能孤立存在。对于软件工程师来讲，只有在他们明白如何实现概念模型时，这些概念模型才有用。本书介绍了一些可用于将概念模型转化成软件实现的模式，并且讨论了在一个大型信息系统中这些软件实现是如何适应系统构架的，另外还讨论了使用这些模式的具体实现技巧。

我写本书是因为它也正是我在开始时想要阅读的书。建模人员会从本书中找到可以帮助他们如何在新领域中大展拳脚的基本思路。这些模式包括：有用的模型、设计背后的论证以及适用范围。**拥有这些信息，建模人员就可以为特定的问题改造现有的模型。**

本书中的模式也可用于评审已有的模型——看看其中哪些可以省略，并给出一些有助于模型改进的可选方案。当我在评审一个项目时，通常会将所看到的东西同我在以前工作中所总结出的模式加以比较。我发现了解我的书中包含的模式有助于我更容易地应用自己过去的经验。像这样的一些模式还揭示了在简单的教科书中没有谈及的建模问题。通过探讨为什么要按照我们的方法去进行建模，我们在即使没有直接使用这些模式的情况下也可以更深刻地认识到如何改进我们的建模方法。

## 本书结构

本书分为两个部分。第一部分介绍各种分析模式，即来自概念业务模型的模式。它们提供来自贸易、测量、财务以及组织关系等多个问题域的关键抽象。这些模式都是概念性的，因为它们表征了人们考虑业务的方式，而不是设计计算机系统的方式。该部分的各章重点阐述了可用的可选模式以及这些模式的优缺点。对于同一领域的建模人员来说，每种模式都是明显有用的，但是其中的基本模式通常在别的领域也能够发挥作用。

第二部分重点介绍支持模式，这类模式帮助我们使用分析模式。支持模式展示：分析模式如何适合一个信息系统构架，概念模型的构造如何演变成为软件接口和实现，以及那些特定的高级建模构造如何与更简单的结构关联。

为了描述这些模式，需要借助于一种新的图符表示法。附录简要地讨论了这种图符表示法，并提供了其中各种符号表示的含义。我没有采用单一的方法，而喜欢综合采用来自不同方法的多种技术。附录并不是一个技术指南，但它可以提供一个大纲，使你能回想起技术内容。此外，附录还告诉你应该到何处去查找我所采用技术的指南。

书中每个部分都划分成若干章节，有关分析模式的各个章节阐述了相关的模式，这些相关模式同属一个概念较为松散的主题空间，并受到产生它们的项目的影响。这种组织形式反映了这样一个事实：任何一种模式都必须来自于一个实际的环境。每个模式都由一章中的单独小节加以阐述。我没有采用其他一些模式作者所用的任何正式标题（参见1.2.2节），而是采取尽可能具体而又尽量合理地接近原始项目形式的方式去描述模式。我还增加了一些例子来说明如何将模式应用于其原有领域以及如何将模式应用于其它领域。有关模式研究的最困难的一件事是如何对它们加以抽象，并用于其它领域；我所采取的策略是把这一问题留给读者自己去思考和解决（参见1.2.3节）。

本书更像是一本目录册，而不是一本从头读到尾的书。我尽量按照可单独阅读的方式撰写书中的每一章。（但是，这一点并非总是能做到的。在阅读一章时如果需要先了解其他章的内容，我将在引言中加以说明。）每一章都有引言，引言部分概要地说明该章所针对的问题域，并概要地描述该章所包含的相关模式，此外还会谈到这些模式来自什么项目。

## 如何阅读本书

建议先通读第1章，然后阅读每一章的引言部分，接下来就可以自由地按照自己喜欢的顺序去钻研各章。如果你还不熟悉我所采用的建模方法、表示法或者概念，请参见附录A。附录B

“模式表”简要地总结各模式的概况，因此你可以在再次阅读本书时利用它来协助钻研本书的内容或查找所需的模式。需要特别强调的是本书中的模式也可用于其它领域，不局限于模式所产生的领域，因此我鼓励你浏览一下你可能认为不属于自己兴趣范围之内的章节。比如，我发现为医疗保健行业而设计的观察和测量模型在企业金融分析中就被证明是非常有用的。

## 本书的读者

虽然不同的读者会从本书学到不同的东西并且可能需要一些不同的准备知识，但本书适合的读者范围非常广。

我认为最大的读者群是面向对象（OO）计算机系统的分析人员和设计人员，尤其是那些主要从事系统分析的人员。这一类读者应该或多或少地使用过某种OO分析和设计方法。本书并不提供任何有关面向对象分析与设计的内容，因此如果你对这一领域还很陌生，建议你先阅读有关的书籍。必须强调的是，本书中的模式本质上是一种概念，而我是使用一种完全概念化的方法建模，这使得本书在表述风格上与使用更趋向基于实现的方法建模的书籍有某些区别。

另外一群读者虽然数量很少，但却非常重要，他们主要是那些在建模项目中担当领域专家的人员。这些读者不需要计算机方面的专门知识，但需要了解概念建模。我在本书中使用概念模型的主要原因之一就是要使书中内容更适合于这一类读者。这里的建模项目可能是指针对计算机系统开发或业务过程重组（BPR）的分析。我曾经教过许多专业人士（包括医生、金融交易人员、会计人员、护士以及薪资监管人员等）进行这种类型的建模，并且发现软件背景知识对于概念建模来讲可有可无。本书的业务模型模式恰好介于计算机系统分析和业务建模之间（参见1.4节）。任何这一类型的读者最好都应学习着重于概念性内容的面向对象分析课程。（Odell的著作[1]在这方面特别有价值。）

虽然一些编程人员会对缺少代码和倾向于概念性的描述存在异议，但我希望大多数编程人员能够研读本书的内容。对于这些读者，我建议你们特别注意第14章，该章阐述了概念模型与最后得到的软件之间的关系。

这是一本面向对象方面的书，我坚信面向对象方法是开发软件的上好途径。然而，本书中的这些模型主要是概念模型，而大量数据建模人员已有很悠久的使用概念（或逻辑）模型的传统。数据建模人员会发现很多模式非常有用，尤其是在他们使用了更先进的语义技术的情况下。模型的面向对象特征将揭示面向对象方法与传统方法的许多不同之处。我非常希望这一类型的读者能将本书与其它着重于概念性建模的面向对象分析书籍相结合，并能将面向对象与语义数据建模联系起来。

管理人员会发现可以将本书作为开发活动的一个起点。从模式开始着手开发工作可以帮助我们搞清楚开发工作的目标，而项目计划的制定也可以利用模式所提供的广泛基础。

我并没有把本书的读者定位为普通学生，而主要是针对专业的软件工程师。但我希望普通学生也能关注本书。在学习分析和设计时，我曾感到非常困难，因为只有很少的好例子可以参考，而这些实例又来自大学校园以外的世界。正如看好的代码可以教给你大量的编程技巧一样，看好的模型也可以教给你大量的分析和设计技巧。

## 一本动态更新的书

我所知道的每位作者都曾有过这样的挫折感：一本书一旦出版，书中的内容就很难进行改变。书在读者之间传播，但是作者却没有什么途径可以传达可能出现的变化。随着我不断地学习新的东西，肯定会改变某些想法。我希望这些改变能够传递给读者。

在我的个人主页(http://martinfowler.com/)上会提供更多的资料，使本书可以保持动态更新。因此请随时关注该站点的动态，并通过它让我了解如何改进和发扬本书的思想。

## 致谢

任何作者都会感激那些给过其帮助的人。而本书的编写过程尤其如此，因为我所写的大多数的模式是在我的客户、同事和朋友的协助下创建的。在此我向他们表示最衷心的感谢。

首先也是最需要感谢的是Jim Odell，他是我事业上的良师益友。他传授给我大量有关信息系统开发的知识，并且是我不断获得灵感、有益建议以及强烈幽默感的源泉。可以这么讲，如果没有他的支持，就不可能有本书的面世。

其他需要要感谢的人是：

在伦敦Coopers & Lybrand的开发小组。他们在早期阶段给予我很大的帮助，并帮助我在Smithfield度过了很多个夜晚。

John Edwards。他协助我形成了有关概念性建模及其在软件开发中的作用的早期思想，他还向我介绍了许多有趣的想法，其中包括Christopher Alexander的想法。

John Hope。他建议我首先要考虑领域，再考虑技术，并在几个关键点上为我的工作提出了很好的建议。

伦敦圣·玛丽医院的两位医生Tom Cairns、Mark Thursz。他们和我一起开发了组成第2、3、8章的基本内容的医疗保健模型；他们的工作证明了计算机背景知识并不是一流的概念性建模人员所必须具备的。其中，Mark精通医药术语，他欣然为我们提供了各种医疗保健案例。

医疗保健项目还得到来自圣·玛丽医院、儿童医院（HSC）、圣·托马斯医院以及威尔斯大学的软件专业人员和医疗保健专家的帮助。儿童医院的护士Anne Casey和系统分析员Hazim Timimi协助将最终的Cosmos模型汇集在一起。Gerry Gold发起并推动了该项目。

Brad Kain。他深深地影响了我，使我考虑到构件和重用。

对我来讲，将医疗保健模型应用到第4章的公司财务模型中的实践证明了分析模式可以在不同领域中使用。在此特别感谢Vivek Salgar以及施乐公司中领导MBFW小组的Lynne Halpin和Craig Lockwood，是他们用C++把我们的概念性想法变成了客观实现。

David Creager、Steve Shepherd以及他们在Citibank（都市银行）工作的小组。他们和我一起开发了在第9到11章中刻画金融模式时用到的模型。他们还从原始的医疗保健工作中进一步提出了第12章的大量构架性思想，并向我讲述了伦敦城许多精神病人的生活。

Fred Peel，他发起并负责维护我在Citibank中的工作，并且由于他的强力推动，解除了我的后顾之忧。来自Valbecc的Daniel Poon和Hazim Timimi，他们将我的模糊的思想变成了详细的规格说明。

第6章中的账务模式经过长期的酝酿。其间，Tom Daly、Peter Swettenham、Tom Hadfield以及他们各自小组开发的模型引发了本书中账务模式的诞生。Rich Garzaniti把我的会计学术语挑选了出来。而Kent Beck在完善我的Smalltalk程序方面做了大量工作。

James Odell，他协助我编写了第14章。

我是模式研究团体的后来者，在写完本书的大部分章节之后，我才逐步较好地了解了模式这一概念。正是这一非常开放和友好的团体所做的大量工作激励了我进行研究工作。Kent Beck、Ward Cunningham以及Jim Coplein鼓励我加入这一团体，将一些想法总结成为模式。Ralph Johnson特别为本书的第一稿提出了中肯的意见。

还有许多给予我很好建议的审阅者，他们是：Dave Collins、Ward Cunningham（Cunningham & Cunningham公司）、Henry A. Etlinger（RIT计算机科学系）、Donald G. Firesmith（Knowledge Systems公司）、Erich Gamma、Adele Goldberg、Tom Hadfield（TesserAct Technology）、Lynne Halpin（Nescape Communications）、Brian Henderson-Sellers、Neil Hunt（Pure Software）、Ralph E. Johnson（伊利诺伊大学厄巴纳－尚佩恩分校）、Jean-Pierre Kuilboer（波士顿马萨诸塞大学）、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。还需要特别指出的是，Don Firesmith负责收集了以上审阅者提出的需要修改的问题。

## 参考文献

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

# 目　录

# 第 1 章

# 绪　论

## 1.1 概念模型

绝大多数的对象建模书籍都在讨论分析和设计问题，然而对于分析和设计之间的界限却一直没有形成一致的看法。在对象开发过程中一个很重要的原则就是：要设计软件，使得软件的结构反映问题的结构。遵循该原则的一个结果是：从分析和设计所得到的模型最终都存在既定的某种相似性，使得大多数人觉得它们之间没有多大区别。

我坚信分析和设计之间仍然存在着不同之处，这些不同之处正日益成为一个研究重点。在进行分析时你尽力想理解问题的本质；在我看来，这并不仅仅是用用况 [8] 列出需求清单那么简单的事情。在系统开发中，用况（use-case）即使不是最基本的也是很有价值的一个部分，但捕获它们并不意味着分析的结束。分析还包括透过表面需求进行深入分析，以便获得一个有关该问题本质内容的智力模型。

设想某人打算开发一个模拟斯诺克台球游戏的软件。该问题可以用用况来描述表面的特征："玩家击中白球，使得白球按照一定的速度前进，并按照特定的角度撞击红球，红球被撞击后按照特定的方向前进一定的距离。"你可以记录下成百上千这样的事件，并测量球的速度、角度和行进距离。但仅凭这些样例还不足以支持编写一个好的模拟软件。为了使工作做得更好，你需要透过表面去了解运动背后蕴含的规律，其中涉及到质量、速度、动量等概念。了解这些规律就会很容易地明白如何构造这样的软件。



斯诺克台球问题并不难解决，因为其中的规律已经众所周知，并且已有相当长的历史。但在很多企业里，相关的规律并不易于让人了解，我们必须努力去揭示它们。为此我们创建了概念模型——一种允许我们了解并简化问题的智力模型。某些类型的概念模型是软件开发的必要组成部分，甚至最不愿受控制的黑客也要建立一些概念模型。不同之处在于：建立概念模型到底是它自身的一个过程还是整个软件设计过程的一个方面。

需要特别提醒的是，概念模型是一种人工制品。开发人员用来创建类似于斯诺克台球模拟程序这类事物的运动规律在现实世界中并不存在；它们是由人类创造的，表征了现实世界的某种模型。从工程角度来讲，这些模型非常有效，因为它们使我们能够更深刻地理解现实世界所发生的事情。另外，开发人员可以使用一个或更多的模型；例如，对于斯诺克台球模拟，就可以采用牛顿模型或者爱因斯坦模型。你可能认为爱因斯坦模型会更准确，因为它考虑了由于球的运动速度的变化而导致的质量的变化，所以更加精确。然而，开发人员却基本上都倾向于用牛顿模型，因为球速（相对于光速）如此之慢，在模拟时完全可以忽略质量变化，如果考虑过多，反而会使问题复杂化。这说明一个重要的原则：不存在正确或者错误的模型，只存在对手头的工作更适用的模型。

**建模原则：**模型无好坏对错之分，只有好用或者不好用之分。

模型的选择会影响最终产生的系统的灵活性和可重用性。你可能认为开发者应该使用爱因斯坦模型，因为这样可以使得最终的软件具有很强的灵活性，足以处理包括原子碰撞在内的多个问题。但这样去做会是一件很危险的事情。过多地考虑系统的灵活性将使软件系统过分复杂，对于工程实施来讲，这并不是一件好事。工程实施要求在建造及维护费用与系统功能之间取得平衡。要建造满足某个目标的软件系统，就必须开发满足实际需要的概念模型。你需要的是所能获得的最简单的模型。请不要增加任何不太可能会用到的灵活性特征。

最简单的模型不一定是马上就能想到的模型。寻找简单的解决方案需要花费大量的时间和精力，也有可能会遇到挫折与失败。人们对简单模型的通常反应是："哦，对呀，那是显而易见的"，并且会问自己"为什么会花那么长的时间才得到它"。但无论怎样，简单模型是值得花时间和精力的，不仅是因为它们可以使得建造工作更加容易，更重要的是它们能够使得最终产品在将来更加易于维护和扩展。这也是为什么大家都愿意用功能相同但却更加简单的软件的原因。



如何表示一个概念模型？对多数人来讲，概念模型是用他们的软件语言来表示的。使用语言的好处在于：你可以执行一个模型以检验其正确性并进行进一步探讨。这是一个不小的优点；我就经常使用Smalltalk语言来进行概念建模。另一个好处是：你最终会将模型转变成编程语言，因此用目标语言进行建模可以减少转换工作量。（有一些工具可以解释或者编译分析和设计模型，因此可以减少转换过程中可能出现的问题。）

使用一种语言的危险之处在于：很容易陷于该语言的使用问题之中，而忽略所要探究的问题。（这种使用语言的危险对于高级语言（比如

Smalltalk）来讲问题会小些，我就认识几个使用该语言进行建模的有才能的概念建模人员）。用编程语言进行建模还存在这样的危险：使得模型更偏向于那种语言。这种情况下所建造出来的模型可能使用了一些该语言独有（其它语言不具备）的特性。当然，这并不意味着概念模型不能转换成其它语言，但这可能会使转换过程复杂化。

要避免这些问题，许多人在概念建模时使用了分析和设计技术。这些技术可以帮助人们把注意力放在概念性问题而不是软件设计问题上，并且这些技术也易于领域专家学会并传授给他人。分析和设计技术使用了表达能力更强的图形。它们可能具有非常严格的语法要求，但并非必须如此。要将相关技术设计成可运行的，当然必须要有严格的语法，但当分析方法是与一种编程语言一起使用的时候，这些分析和设计技术就不需要那么严格了。

我使用分析和设计技术的主要原因之一是想让领域专家也能参与其中。让领域专家参与概念建模是很基本的要求。我相信有效的模型只能由那些真正了解该领域的员工（在该领域专职工作的员工）来创建，而不是由软件开发人员来创建（无论这些开发人员在该领域已经工作了多长时间）。如果领域专家要参与概念建模工作，就必须对他们进行培训。我曾经向客户服务主管、医生、护士、金融交易人员以及公司财务分析师传授OO分析和设计技术。我发现IT背景知识对于建模技能既无多大帮助，也无多大妨碍。我所认识的最优秀的建模人员是伦敦一所医院的一位医生。作为一名专业的系统分析师和建模人员，我可以为开发过程带来一些有价值的技能：我能提供一套严谨的方法，知道如何使用各种技术，并且我的外行似的观点可以挑战任何公认的至理名言。然而，所有这些并不足够。虽然我在医疗保健计算领域做了大量的工作，但对于医疗保健行业的了解程度将永远比不上任何一位医生或者护士。专家知识是建立一个好的分析模型的关键。



我们努力使分析技术独立于软件技术。比较理想的情况是概念建模技术完全独立于软件技术，就像运动定律一样。这种独立性可以使技术不会妨碍对问题的理解，并使得最终的模型能够适用于所有类型的软件技术。但在实践过程中，这种理想情况并不会发生。我曾竭尽所能地想开发非常概念性的只关注问题本身的模型，然而我使用的是面向对象的技术，因此，仍然只能反映一种软件设计方法的理念。通过将本书的模型同David Hay[7]的模型进行比较，会很强烈地感受到软件技术是如何影响概念建模的。我们都尽力想建造概念模型，然而我们的结果却有很大的不同，因为他使用的是关系技术而我使用的是面向对象技术。这是软件特性的必然结果。建造软件就是在建造虚拟的机器。我们用来建造软件的语言既能控制

物理的机器，还能表达问题的要求。语言发生变化的一个原因就是我们找到了表达问题的要求的更好方式。这些语言上的变化也会因此影响到我们建造概念模型的方法。尽管还有一些棘手的领域（参见第14章），但总的来说，将产生的模型转换成面向对象的软件并不太困难。

然而，在这里确实需要提醒的是：概念模型更贴近于软件的接口，而不是软件的实现。面向对象软件的一个重要特征就是将接口与实现分离开来。但在实际的开发过程中很容易忘记这种区分，因为通常的语言在这两者当中没有显式的区别。软件构件的接口（构件类型）与它的实现（构件类）之间的区分是非常重要的。在“四人帮”的著作《设计模式》[6]中许多重要的基于委托（delegation-based）的模式都依赖于这种区分。在实现这些模型时，一定要牢记这种区分。

**建模原则：**概念模型是与接口（类型）而不是与实现（类）相关联的。

## 1.2 模式世界

最近两年，模式已经成为对象研究团体里最热门的话题之一。它们正迅速成为一种时尚，并引发了一大批人的注意和广泛的宣传。我们也看到，在对象研究团体内部存在一些有关什么值得研究的争论，其中也包括有关模式的确切定义是什么的争议。的确很难找到一个有关模式的通用定义。

开展模式运动有着不同的根源。最近几年，越来越多的人感觉到软件界不擅长描述并传播好的设计实践经验。方法论虽然有很多种，但是它们只是在定义用于描述设计的语言而不是在描述具体的设计。当前仍然缺乏可用于培训和启发的、阐述基于经验的有用设计的技术性论文。正如Ralph Johnson和Ward Cunningham所讲的：“尽管有了最新的技术，项目仍可能因为缺乏一般的解决方案而失败”[4]。

模式也从几个不同的起点发展而来。Kent Beck和Ward Cunningham是Smalltalk语言的两位倡导者，他们偶然间接触到了Christopher Alexander（他创立并发展了建筑学方面的模式理论与模式集合）的思想。Bruce Anderson，他在20世纪90年代早期领导了OOPSLA上的一个系列讨论，为软件构架设计师编制了一本手册。Jim Coplien，他在自己的C++著作[3]中阐述了一些可用在C++中的惯用法。大量的这类人员聚在一起成立了Hillside Group（山边小组），以便进一步研究这些思想。

“四人帮”的著作《设计模式》[6]的出版以及Hillside Group在1994年发起的关于PLoP（编程模式语言）的会议[4]触发了模式运动的开展与广泛普及。

我与这个不断发展的团体曾少有联系。长期以来，我一直渴望能够拥