TURING 图灵程序设计丛书

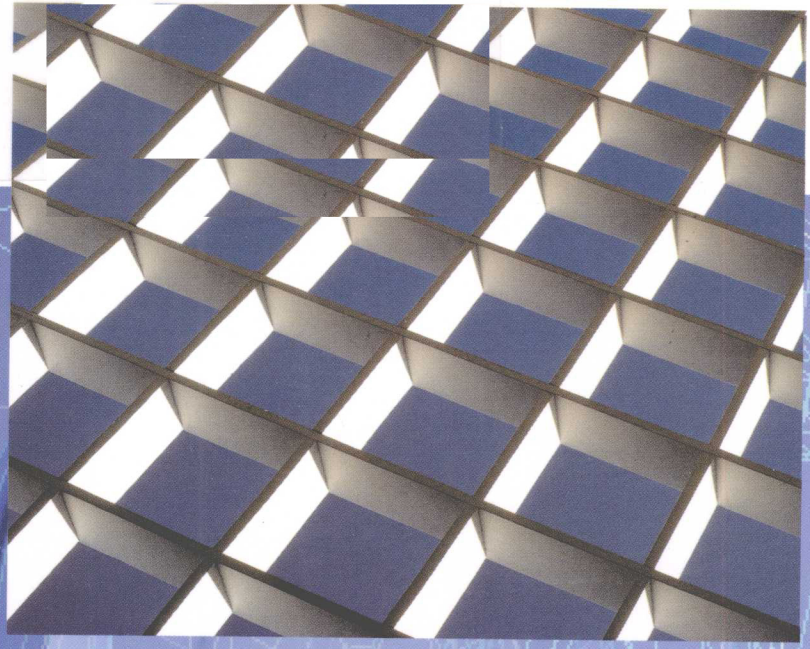
Addison Wesley

Design Patterns Explained

A New Perspective on Object-Oriented Design

设计模式解析（第2版）

[美] Alan Shalloway
James R. Trott 著
徐言声 译



- 简明易读、注重实用的设计模式最佳入门图书
- 不仅讲述模式本身，更提示模式背后的思想
- 凝聚业界专家自身学习和教学经验

人民邮电出版社
POSTS & TELECOM PRESS

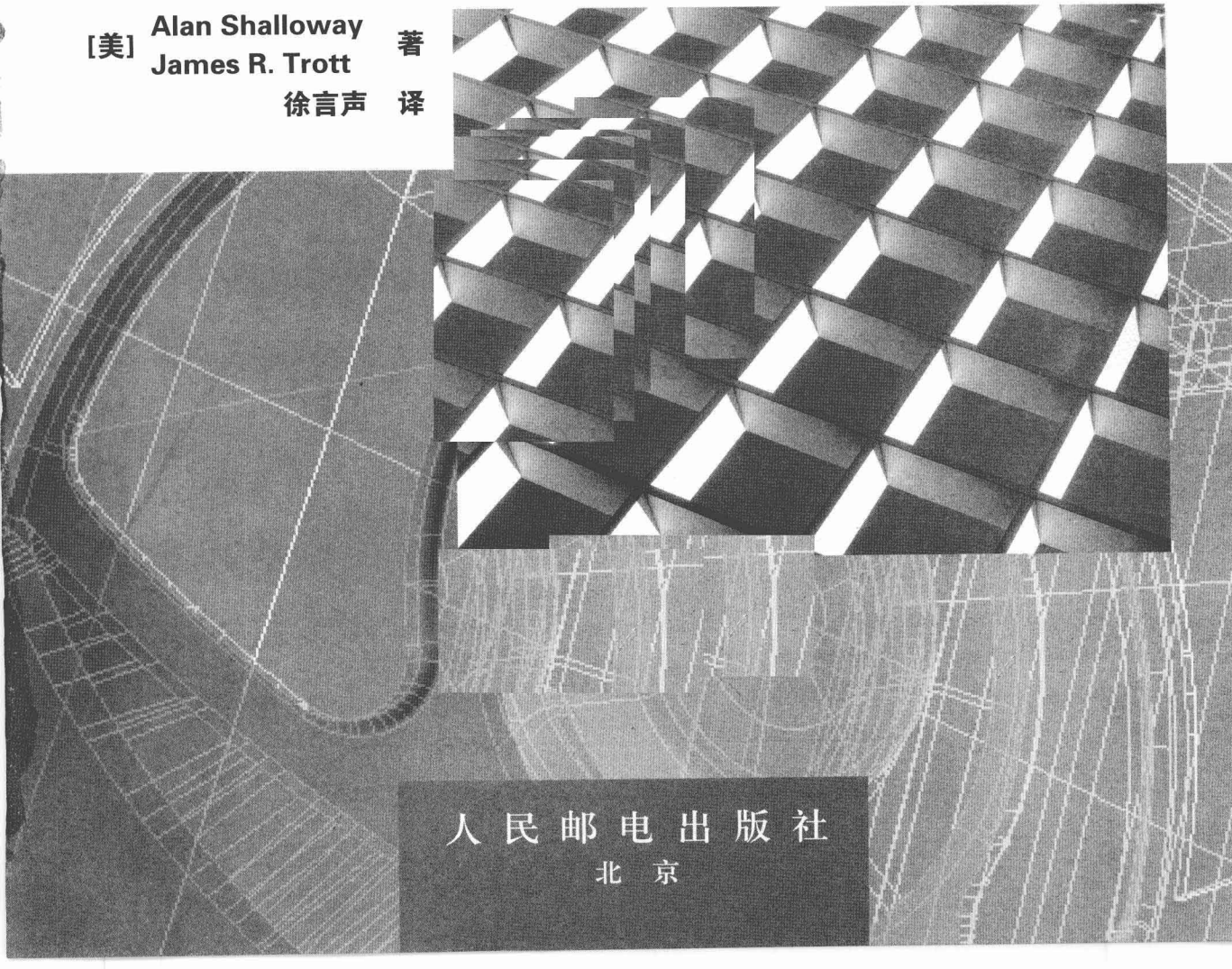
TURING 图灵程序设计丛书

Design Patterns Explained

A New Perspective on Object-Oriented Design

设计模式解析（第2版）

[美] Alan Shalloway
James R. Trott 著
徐言声 译



人民邮电出版社
北京

图书在版编目（CIP）数据

设计模式解析 / （美）沙洛维（Shalloway,A.），（美）特罗特（Trott,J.R.）著；徐言声译. -- 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：Design Patterns Explained: A New Perspective on Object-Oriented Design
ISBN 978-7-115-24098-9

Ⅰ. ①设… Ⅱ. ①沙… ②特… ③徐… Ⅲ. ①软件设计 Ⅳ. ①TP311.5

中国版本图书馆CIP数据核字(2010)第209224号

内 容 提 要

本书以作者自身学习、使用模式和多年来为软件开发人员（包括面向对象技术老兵和新手）讲授模式的经验为基础撰写而成。首先概述了模式的基础知识，以及面向对象分析和设计在当代软件开发中的重要性，随后使用易懂的示例代码阐明了12个最常用的模式，包括它们的基础概念、优点、权衡取舍、实现技术以及需要避免的缺陷，使读者能够理解模式背后的基本原则和动机，理解为什么它们会这样运作。

本书适合软件开发专业人士，以及计算机专业、软件工程专业的高校师生阅读，也可作为面向对象分析与设计课程的参考教材。

图灵程序设计丛书

设计模式解析（第2版）

◆ 著　[美] Alan Shalloway
译　徐言声
责任编辑　傅志红
执行编辑　丁晓昀

◆ 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
中国铁道出版社印刷厂印刷

◆ 开本：800×1000　1/16
印张：19.5
字数：444千字　2010年12月第2版
印数：14 501－17 500册　2010年12月北京第1次印刷

著作权合同登记号　图字：01-2005-3572号

ISBN 978-7-115-24098-9

定价：55.00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

献给 Leigh、Bryan、Lisa、Michael 和 Steven，
为了他们的爱、支持、
鼓励和牺牲精神。
——Alan Shalloway

献给 Jill、Erika、Lorien、Mikaela 和 Geneva，
你们是我生命花园中的玫瑰。
唯上天得荣耀。
——James R. Trott

前　言

如果我已经有了第 1 版，还需要买第 2 版吗？

回答当然是“需要”！原因如下。

自完成第 1 版的写作之后，我们对设计模式又有了大量更深入的理解，包括以下一些方面。

- 如何使用共性和可变性分析来设计应用程序的架构。
- 设计模式与极限编程（eXtreme programming，XP）和敏捷开发的关系，以及设计模式如何有助于二者的实施。
- 为什么测试是高质量编程的一个优先原则。
- 为什么使用工厂（factory）实例化和管理对象至关重要。
- 对帮助学生理解如何用模式思考而言，哪些模式是必不可少的。

本书探讨了所有这些主题。我们进一步深化和澄清了前一版阐述过的主题，并增加了一些非常有用的新内容，包括：

- 第15章，共性和可变性分析；
- 第20章，来自设计模式的教益：各种工厂模式；
- 第22章，Object Pool模式（《设计模式》一书中没有讨论的模式）；
- 第24章，工厂模式的总结。

我们改变了一些模式的阐述顺序，据参加该课程的学生反映，这样的顺序更有助于掌握模式背后的思想。

所有章节的内容都进行了少量修改，综合了过去三年来从许多读者那里收到的各种反馈意见。而且，为了帮助学生学习，我们为每一章都编写了复习题（答案在本书配套网站可以找到）。

我们可以非常坦率地说，本书无疑是少数值得拥有的第 2 版，即使读者已经购买了第 1 版。

非常乐于倾听您宝贵的建议。

——Alan 和 Jim

设计模式和面向对象程序设计曾经做出过这样的承诺：要简化软件设计人员和开发人员的工作。技术媒体甚至大众媒体每天都在传播相关的术语。然而，要学习这两种技术，熟练掌握它们并且知其所以然，可能非常困难。

你使用某种面向对象或者基于对象的语言可能已经多年，可你是否知道，对象真正的威力并不在于继承，而是来自封装行为。你可能对设计模式很好奇，而且感觉相关的著作都有些太过深奥和夸张。倘若如此，本书正适合你。

本书是以作者多年来为许多软件开发人员（包括面向对象技术老兵和新手）讲授模式的经验为基础撰写而成的。我们相信，而且我们的经验也已经证明，如果能够理解模式背后的基本原则和动机，理解它们为什么会这样运作，那么你的学习曲线将不可思议地缩短。而且在我们对设计模式的讨论中，可以懂得真正的面向对象思维定式，这正是登堂入室的必由之路。

通过阅读本书，读者能够完整地理解 12 个核心设计模式和 1 个分析模式，也将了解到设计模式并不是孤立存在的，多个设计模式协同工作才能帮助我们创建更加“健壮”的应用程序。你还可以获得阅读其他设计模式文献所需的足够基础知识，如果愿意，可能还能够自己发现新的模式呢。最重要的是，你将为创建灵活、完善而且更易维护的软件做好准备。

虽然这里所讲授的 12 个模式并没有涵盖所有应该学会的模式，但是理解了这 12 个模式，就能够举一反三，更加容易地自学其他模式。我们没有讨论入门所需之外的更多模式，而是讲述了更加有用的与模式相关的若干问题。

从面向对象到模式再到真正的面向对象

从很多方面来看，本书实际上是在复述我自己学习设计模式的经历。我是在学习模式本身之后，再学习模式背后的思想。然后，又将这种理解扩展到分析和测试领域，也扩展到学习模式与敏捷编程方法的关系中。本书第 2 版中包含了许多第 1 版出版后我的一些领悟。在学习设计模式之前，我自认为已经是一个很不错的面向对象分析和设计专家了，我曾经设计和实现了几个针对许多不同行业的非常出色的项目。我使用 C++并且正在学习 Java，代码中的对象可以说是中规中矩[①]、封装严密，而且还能为继承层次结构设计出优秀的数据抽象。我想自己应该已经得面向对象之道了。

现在回想起来，我发现自己那时虽然总是遵循大多数专家的建议行事，但是并没有真正理解面向对象设计的全部威力。直到开始学习设计模式，我的面向对象设计能力才得以拓展和加强。即使还没有直接使用模式，理解设计模式也已经使我成为更加出色的设计人员。

我开始学习设计模式是在 1996 年。那时我还是美国西北部一家大型航天公司的一名 C++和

① 原文为 well-formed，这个术语许多文献译为“格式/形式良好的”、“良构”等等，但根据 C++标准 1.4.1 的定义：“按照语法规则、可诊断语义规则和定义一次规则构造的 C++程序”，似乎按照数学界的习惯译为“合式的”或“合乎规则的”更好。此处因为并不强调技术语义，所以随文就译了。——译者注

面向对象设计讲师。有几个人要求我领导一个设计模式学习小组，正是在那里我遇到了本书的另一位作者 Jim Trott。学习小组中发生的几件事情很有意思。一开始，我就对设计模式着了迷。我很喜欢能够将自己的设计与其他经验更多的人的设计进行比较。然后，我就发现自己并没有发挥“按接口设计”（designing to interface）的全部威力，而且并不总是关注是否存在“一个对象在还不知道另一个对象的类型时就使用这个对象”的情况。我还注意到，刚刚从事面向对象设计的人（一般总是认为这时就学习设计模式过早）从学习小组所得的获益，居然同专家差不多。设计模式展示了优秀的面向对象设计实例，而且阐明了基本的面向对象设计原则，这些有助于初学者更快设计出成熟的方案。到整个学习结束时，我已经完全相信：设计模式是面向对象设计发明以来软件设计领域出现的最伟大的东西。

可是，在我审视当时自己的工作时，却看到代码中并没有使用任何设计模式。或者说，至少还没有有意识地使用任何一个模式。后来，随着对模式学习的深入，我发现自己开始在代码中使用许多设计模式了，不再仅仅是一个好的编程匠。当然，现在我对模式的理解更加深入，使用起它们来也更加得心应手了。

我当时只是觉得自己可能对设计模式还了解得不够，应该学习更多。那时我只了解其中 6 个模式。然后，我突然顿悟。当时我是一个项目的面向对象设计顾问，而且应要求为项目做高层设计。这个项目的负责人极为聪明，但在面向对象设计方面却是一个新手。

问题本身并不怎么难，但是对代码维护性的要求很高。我花两分钟查看了一下问题，然后就照本宣科地按照通常使用的数据抽象方法提出了一个设计。很不幸，我自己也清楚这不会是什么好的设计。只用数据抽象使我无功而返，必须另寻良策。

两个小时之后，我用尽了自己知道的所有设计技术，已经黔驴技穷，情况却毫无好转的迹象。我的设计没有本质上的变化。最让人感到灰心的是，我知道肯定有更好的设计方案，只是我想不出来。更具讽刺意味的是，我还知道这个问题里藏着 4 个设计模式，可是我不知道怎么使用。于是，我——一个自封的面向对象设计专家，被一个简单问题生生噎住了！

我感到失望之极，只好休息一下，出去走走，清醒清醒头脑。我告诉自己，至少 10 分钟之内不要再想这个问题了。可是，才过 30 秒钟，我就忍不住又思考起来！这次我突生灵感，刹那间，自己的设计模式观改变了：不能将模式作为一个单独的东西使用，应该把它们结合起来。

模式应该相互配合，共同解决问题。

这话以前我也听说过，但是当时并没有真正理解。因为软件中的模式最初以设计模式为名引入，我想当然地认为它们主要都是关于设计的，并一直囿于这样的想法而碌碌无为。我曾认为，在设计领域，模式就是类之间合乎规则的关系。后来读到 Christopher Alexander 的奇书 *The Timeless Way of Building*（牛津大学出版社，1979）[①]，我才知道所有层次——分析、设计和实现都存在模式。Alexander 曾阐述了使用模式有助于理解问题域（甚至有助于描述它），而不仅仅是用来在理

① 中文版《建筑的永恒之道》，由知识产权出版社出版。——译者注

解问题域之后完成设计。

我错就错在试图在问题域中创建类，然后再将它们结合起来形成一个系统，这正是 Alexander 所说的非常糟糕的做法。我从没有自问过这些类是否正确，因为它们看上去很好，显而易见，类从一开始分析就马上浮现于我的脑海，而且它们正是按教科书一直教的那样应该在系统描述中寻找的那些“名词”。但是试图将它们组合起来时却遇到了重重困难。

当我再回到办公室，在设计模式和 Alexander 方法的指导下重新创建类时，仅仅几分钟时间，一个大为改观的解决方案就展现出来。这真是一个好设计，我们最后根据它实现了产品。我真的很兴奋——兴奋于自己设计了如此优秀的方案，也兴奋于设计模式的威力。从那时起，我开始将设计模式融入开发工作和教学中。

我发现，不熟悉面向对象设计的程序员也可以学习设计模式，而且，通过学习设计模式，他们能够掌握基本的面向对象设计技术。对我而言就是如此，对我教授的学生而言亦然。

想象一下我是多么惊讶吧！我曾经读过的设计模式图书，曾经与之交谈过的设计模式专家都说，在开始设计模式研究之前，需要扎扎实实地打好面向对象设计基础。可是，我自己的亲身经验却表明，在学习面向对象设计的同时学习设计模式的学生，比仅仅学习面向对象设计的学生，进步更快，他们掌握设计模式的速度甚至与有经验的面向对象老手一样。

我开始使用设计模式作为自己教学的基础，并且将自己的课程称为“面向模式的设计：设计模式从分析到实现”。

我希望我的学生能理解这些模式，继而可以发现使用一种探索式的方法是有助于促进这种理解的最佳方式。例如，我发现，先提出问题，然后通过大多数模式中都适用的一些指导性原则和策略，帮助学生尝试为此问题设计出解决方案，这样阐述 Bridge（桥接）模式更好。在实际探索中，学生找到了解决之道——其实就是 Bridge 模式，并牢牢地记在心中。

设计模式与敏捷方法/极限编程

设计模式之下所隐藏的指导性原则和策略现在对我而言已经非常清楚了。《设计模式》一书中肯定提到了这些内容，但是讲得太简洁，以至于我第一次阅读时完全没有体会到其价值。我相信《设计模式》一书实际上是以 Smalltalk 社区为目标读者而写的[①]，这些原则在 Smalltalk 社区可以说是根深蒂固，所以不需要太多背景。但是因为自己对面向对象范型的理解很有限，我理解这些原则花了很长时间。直到我将《设计模式》四位作者的工作与 Alexander 的工作、Jim Coplien 关于共性和可变性分析的工作、Martin Fowler 关于方法学和分析模式的工作结合起来之后，这些原则对我而言才变得足够清晰，我甚至能够向其他人讲解。这对我自己的教学

① 此说法不确切，设计模式的起源与量子力学殊途同归的群英会非常相似，得多人之力，而众多先驱中只有 Kent Beck、Ward Cunningham 和 Ralph Johnson 等是纯 Smalltalk 社区背景的，Eric Gamma、Jim Coplien、John Vlissides 等则来自 C++社区。事实上，《设计模式》一书本身是用 C++作为描述语言的。——译者注

生涯也很有帮助——我再也不能像自己工作时那样容易地想当然，而又能侥幸无事了。

自本书第1版出版以来，我一直在进行大量的敏捷开发实践，具有了较多的极限编程实践、测试驱动开发（TDD）和Scrum的经验。刚开始，在结合设计模式和极限编程、测试驱动开发时还是经历了一段困难时期。但是，我很快认识到它们都非常重要，而且植根于一些相同的原则（虽然设计方法并不相同）。事实上，在敏捷软件开发训练课上，我明确说明，如果正确使用设计模式将为引入敏捷开发打下很好的基础。

贯穿本书始终，我讨论了许多设计模式与敏捷管理和编程实践的关联。如果读者对极限编程、测试驱动开发或者Scrum不熟悉，可以不用太在意这些论述。但是，如果真的如此，我建议你下一步就去读一本有关的著作。

我发现无论何种情况下，都可以用这些指导性原则和策略来“推导”出几个设计模式。这里“推导出设计模式”的意思是，如果看到可能用设计模式解决的问题，就可以使用从模式中学到的这些指导性原则和策略，得到以模式表达的解决方案。我明确地告诉学生们，我们并不是用这种方法真正得出设计模式；相反，我只是要说明一种可能的思考过程，最终成为设计模式的那些解决方案的提出者使用的也是这样的过程。

我解释这些数量不多但很强大的原则和策略的能力与日俱增。随之而来的，是我发现自己解释《设计模式》一书中的模式时，它们更加有用了。事实上，我在设计模式课上用这些原则和策略能够阐述几乎所有的模式。

我还发现，无论是否使用设计模式，自己在设计中都在使用这些原则。我对此并不感到惊讶。如果使用这些原则和策略能够得到后来才发现等效于设计模式的设计，那么就说明这些原则和策略已经给了我一种自己做出优秀设计的方法（因为根据定义，模式代表着优秀设计）。有了这些技术难道还会只是因为不知道对应模式（可能已知也可能还没有发现）的名字而得出较差的设计吗？

这些认识帮助我很好地磨砺了自己的培训过程（和现在的写作）。我现在的教学工作已经有了好几个层次。教授面向对象分析和设计基础时，我教设计模式，并用它们作为优秀面向对象分析和设计的例子。此外，通过设计模式来教授面向对象概念，学生们还能更好地理解面向对象原则。而且教授指导性原则和策略，使学生们可以自己创建出质量能够与模式媲美的设计。

在此讲述这些，是因为本书正是沿袭了我所教授课程的模式，几乎所有的材料就是我们目前的课程之一——“设计模式、测试驱动开发或者敏捷开发最佳实践”[①]的内容。

通过阅读本书，你将学习到这些模式。但尤其重要的是，你将学到模式为何有效和如何协同工作，以及模式背后的原则和策略，这将有助于充分利用你自身的经验。当本书提出一个问题时，如果你能够联想到一个曾经碰到的类似问题，将极其有益。本书并没有讲述什么新知识或者如何

① 参见本书英文版配套网站 http://www.netobjectives.com/dpexplained，还有更多有关课程的信息。

应用新模式，而是提供了一种考虑面向对象软件开发的新视角。我希望你的自身经验能够与设计模式的原则结合，成为学习过程中强大的助力。

Alan Shalloway
2000 年 12 月第 1 版
2004 年 10 月第 2 版

从人工智能到模式再到真正的面向对象

我的设计模式历程与 Alan 的设计模式历程可以说是殊途同归，具有下述同样的结论。

- 基于模式的分析能够使我们成为更高效也更有效的分析人员，因为它使我们能够更加抽象地处理模型，因为它代表了许多其他分析人员的集体经验。
- 模式能够帮助人们学习面向对象的原理，并有助于解释我们处理对象的方式。

我的职业生涯开始于人工智能领域，工作是创建基于规则的专家系统。这其中涉及倾听专家们的讲述，为其决策过程建立模型，然后将这些模型编码成基于知识的系统中的规则。在构建这些系统时，我发现了一些反复出现的主题。对于一些相同类型的问题，专家们总是用类似的方法去解决。例如，诊断设备问题的专家往往首先寻找简单、快速的解决方案，然后再系统化一些，通过系统分析将问题分解为多个子问题。在系统诊断时，他们也往往在进行其他形式的测试之前，先尝试成本低或者能排除较大范围问题的测试。其实无论诊断的是计算机中的还是某个油田设备的问题，这种方法都适用。

要换在今天，我会把这些反复出现的主题称为“模式”。设计新的专家系统时，我很自然地也开始寻找这些反复出现的主题了。对于模式思想，我当时是非常开放而且心存好感的，虽然还不知道它们是什么。

后来到了 1994 年，我发现欧洲的研究者们已经整理了这些专家行为的模式，放入名为“知识分析和设计支持”（简称 KADS）的软件包中。Karen Garder 博士——一位才华横溢的分析师、建模专家、顾问，一个天才，开始在美国将 KADS 应用于她的工作中。她扩展了欧洲研究者的工作，将 KADS 应用于面向对象系统。她使我的眼界大开，看到了软件界正在形成的一个全新领域——基于模式的分析和设计，这很大程度上都源自 Christopher Alexander 的工作。Karen Garder 的书 *Cognitive Patterns*（剑桥大学出版社，1998 年）叙述了这些思想。

突然之间我有了一个为专家行为建模的框架，能够不至于过早地陷入复杂和异常情况。借助这种框架完成接下来的三个项目时，我花费的时间缩短了，重复工作减少了，而最终用户的满意度却提高了，原因如下。

- 我能更快地设计模型，因为模式已经事先告知会出现什么情况。它们能够告诉我基本的对象是什么，什么应该特别注意。
- 我与专家沟通的成效大增，因为对于处理细节和异常情况我们有了结构化更强的方法。

- 通过模式，能够针对系统设计更好的最终用户培训方案，因为模式已经预先告知系统最重要的特性。

最后一点意义重大。模式之所以能够帮助最终用户理解系统，是因为它们提供了系统的来龙去脉，说明了为什么我们会这样设计。我们可以用模式描述系统的指导性原则和策略。我们可以用模式开发最佳范例，从而帮助最终用户理解系统。

我被这一切吸引住了。

因此，当我就职的公司组织了一个设计模式学习小组时，我当然积极参与。于是我遇到了本书的另一位作者 Alan Shalloway，他作为一位面向对象设计师和顾问也殊途同归地在工作中体会到了类似的境界。于是也就有了本书的诞生。

自完成第 1 版以来，我进一步领悟到这种分析方法能够多么深刻地增进我们的理解。我参与了许多不同类型的项目，其中许多甚至与软件开发无关。我看到许多一起协作、相互交换知识、交换思想、生活在不同地方的人所组成的各种系统。模式和面向对象的原则依然有助于我。和计算机系统中一样，减少工作系统之间的依赖性也能获得很好的效果。

我衷心希望本书中讲述的原则能够对各位读者成为更有效而且更高效的分析人员有所帮助。

James R.Trott
2000 年 12 月第 1 版
2004 年 12 月第 2 版

本书约定

在本书写作过程中，我们选用了一些特定的风格和版式约定。读者可能对其中一些不太习惯。因此我们对如此选择的原因作以下说明。

第一人称

本书是两位作者的合作作品。为了找到阐述这些概念的最佳方式，我们经常争论并且不断做出改进。Alan 在他的课程中曾使用这些方式试讲，然后我们又共同进行了改进。本书主体部分中我们选择使用第一人称单数，因为这能够如我们希望的那样，以更生动和自然的方式娓娓道来。

便于浏览

我们试图使本书易于浏览，读者能够在不全部阅读正文的情况下获得要点，也可以迅速找到需要的信息。我们大量使用了表格和带有项目符号的列表并在页边加上了总结相应段落的文字。讨论每个模式时，提供了模式关键特性总结表。我们相信这些措施能够使本书的可读性大大提高。

示例代码

本书讲述的是分析和设计，而不是具体的代码实现，其目的是帮助读者在面向对象开发中积累的真知灼见和最佳实践经验的基础上，思考如何做出优秀的设计，就像用设计模式所表示的那样。所有程序员都会遇到的挑战之一就是：不能过早开始进行实现，需要三思而后行。既然如此，本书有意地尽量避免过多讨论实现。我们的示例代码可能看上去有些分量不足而且不够完整。比如说，代码都没有提供错误检查。这是因为使用它们只是为了说明概念。然而，本书的配套网站 http://www.netobjectives.com/dpexplained 中存放了更完整的示例代码[①]，书中的代码是从中摘出来的。C++语言和 C#语言的示例代码也可以在这个网站中找到。

策略和原则

本书是一本介绍性读物，有助于你快速学习设计模式，并从中理解启发了设计模式的原则和策略。阅读本书之后，可以继续阅读更学术化的模式图书或参考书。本书最后一章将列出许多可能对你有用的参考书。

展现广度，形成认识

本书努力使读者能够对设计模式有所认识，书中将展现模式领域的广度，但不会很深入地讲述任何一个模式（参见上一点）。

如果带一个人去美国旅游两个星期，你会带他去哪里呢？也许是几个主要景点，可以帮助他了解一些建筑、风土人情、城市及城市之间的广袤原野的风光、高速公路和咖啡厅，但是不可能向他展示一切。为了丰富他的知识，可以选择性地给他播放一些其他景点和城市的宣传片，使他有所认识。以后他就可以自己计划未来的旅游行程。我们试图让你对设计模式有一个基本认识，因而与此类似，本书也将带你参观设计模式中的主要景点，然后使你对其他方面也有所认识，这样读者就可以自己计划进一步的设计模式学习旅程了。

C#开发人员如何阅读 Java 代码

本书中的所有代码都是用 Java 语言编写的。如果你没有使用 Java 的经验，但是能够阅读 C#代码，需要了解以下几点：

Java 使用 extends 和 implements 分别表示扩展另一个类或者实现一个接口的类，而不是像在 C#中那样两种情况都用冒号（:）。

因此，在 Java 程序中，我们会看到：

① 本书的示例代码也可从图灵网站下载。——编辑注

```
public class NewClass extends BaseClass
```

或者

```
public class NewClass implements AnInterface
```

而在C#程序中，看到的则是：

```
public class NewClass : BaseClass
```

或者

```
public class NewClass : AnInterface
```

Java中的所有方法都是虚拟的，因此不用指定方法是new还是overridden。Java中没有这样的关键字，所有子类方法都会改写它们从基类继承的方法。虽然还有其他区别，但是我们的代码示例中不会出现。

C++开发人员如何阅读 Java 代码

C++开发人员阅读 Java 要困难一些，但是也不算太难。最明显的区别是 Java 没有头文件。但是阅读组合了头文件和代码的 Java 式文件，其实非常直观，无需解释。除了上述与 C#的区别之外，Java 是不在栈中存储对象的。Java 在堆存储区中存储对象，栈中只存储保存着对象引用（指针）的变量。所有对象都必须用 new 关键字创建。

因此，在 Java 程序中，我们会看到：

```
MyClass anObject= new MyClass();
anObject.someMethod();
```

而在 C++中，看到的则是：

```
MyClass *anObject= new MyClass();
anObject->someMethod();
```

因此如果在每个引用对象的变量名声明中添加一个星号（*），然后将句点（.）转换为连字符加右尖括号（->），Java 代码看上去就像 C++代码了。

反馈

设计模式仍然在发展当中，它们其实就是发现最佳实践以及面向对象基本原则的专业人员之间的行话。

因此，我们非常重视你对本书的反馈意见：

- 我们什么地方做得比较好，什么地方做得还不够？
- 有没有需要改正的错误？
- 有没有什么地方写得不够清晰？

请访问Net Objectives公司为本书英文版设立的配套网站，网址是http://www.netobjectives.com/dpexplained[①]。在这个网站上，能够找到我们的最新研究成果，以及与本书和一般性软件开发问题相关的讨论组。请在讨论组中发表改正意见、评价、真知灼见和心得体会。

第 2 版的新内容

第 2 版相对第 1 版而言有许多变化和改进。它反映了我们过去几年使用和教授设计模式中的收获，也加入了从读者那里收集到的无私和非常有价值的反馈意见。

主要的变化有下面几个方面。

- 重新组织了章节顺序（例如，将对 Strategy 模式的叙述提前）。
- 扩展了对共性和可变性分析（CVA）的讨论。
- 将极限编程和设计模式结合起来。
- 所有示例现在都是完整的可执行代码，而不再是示意性的或者片段代码了。配套网站还有 C#和 C++语言示例代码。
- 增加了对为什么将工厂用作实例化器/管理器的解释，这部分内容极为有用。
- 新增一个《设计模式》书中没有讲到的模式：Object Pool。
- 讨论了模式的缺陷，包括关于“将模式作为辅助思考指导”的警告。模式并不是真理！
- 在语法和风格方面也进行了大量小的订正。

致谢

几乎每本书前言的最后都会有致谢，感谢帮助该书出版的人。直到自己写书，我们才如此真切地体会到其中缘由。出一本书的确是集体劳动的结晶。我们要感谢的人，可以列出一个长长的清单。

以下诸位对我们的帮助尤其重要。

- Addison-Wesley 公司的 Debbie Lafferty，她永不疲倦对我们进行鼓励和督促。
- 我们的同事 Scott Bain，他耐心地审阅了本书，并提供了许多真知灼见。他和 Alan 在 Net Objectives 公司的合作对于本书第 2 版中新增的许多内部都很有启发性。
- 我们的审稿团队：James Huddleston、Steve Metsker 和 Clifton Nock。

① 本书中的相关资源请到图灵网站（www.turingbook.com）下载。——编者注

- 特别要提到 Leigh 和 Jill——她们是极富耐心的妻子，容忍而且鼓励我们完成本书，实现梦想。

审阅本书第 1 版和第 2 版多个草稿版本的人很多，他们提供了许多极好的评论。我们尤其要提到 Brian Henderson、Bruce Trask、Greg Frank 和 Sudharsan Varadharajan，他们无私而且耐心地让我们分享了他们自己的所知所想。

Alan 要特别感谢——

- 我以前的几个学生，他们对我的影响之大，可能他们自己都永远不会知道。在我对是否在课程中引入新想法犹豫再三，感觉似乎应该墨守成规的时候，是他们在我刚开始讲述课程时对新概念的热情，鼓励我在课程中越来越多地加入了自己的想法。感谢 Lance Young、Peter Shirley、John Terrell 和 Karen Allen。他们的行动对我来说是一种不断的提醒，说明鼓励的作用是多么深远。
- John Vlissides，感谢他富于思想的意见和启发性的询问。
- 第 2 版还要加上对另一位 Net Objectives 公司的同事 Dan Rawsthorne 博士的感谢。他从事敏捷开发的方法对我影响很大。对另一位同事 Jeff McKenna 的支持和肯定我也感激不尽。我还要感谢 Kim Aue，我们 Net Objectives 公司的“大总管”，她在各方面的支持对我帮助极大。
- 我要特别感谢 Martin Fowler、Ken Schwaber、Ward Cunningham、Robert Martin、Ron Jeffries、Kent Beck 和 Bruce Eckel 就本书相关问题与我的交谈（有时候是通过电子邮件），当然，这并不是意味着本书内容他们都同意或者认可。

Jim 要特别感谢——

- Karen Gardner 博士，人类思维模式顾问和良师。
- Marel Norwood 博士和 Arthur Murphy，我在 KADS 和基于模式分析方面的最初合作者。
- Brad VanBeek，他为我提供了在本学科中发展的空间。
- Alex Sidey，他指导我掌握技术写作的规律和诀窍。
- Sharon 和 Bob Foote 博士，现在任教于西点军校，使我具备了对人永不知足的好奇心和持久的兴趣。他们的爱和鼓励已经成为模式永存我心，无论是作为一个人、一位父亲和丈夫，还是作为一位分析师。
- 千禧救助与开发服务组织（`www.mrds.org`）的 Bill Koops 和 Lindy Backues，他们帮助我看到，基于模式的方法甚至能够用于救助贫穷和边缘化的人。他们真是好伙伴，好导师。

目　录

第一部分　面向对象软件开发简介

第二部分　传统面向对象设计的局限

第三部分　设 计 模 式