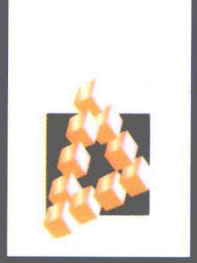
TURING 图灵程序设计丛书



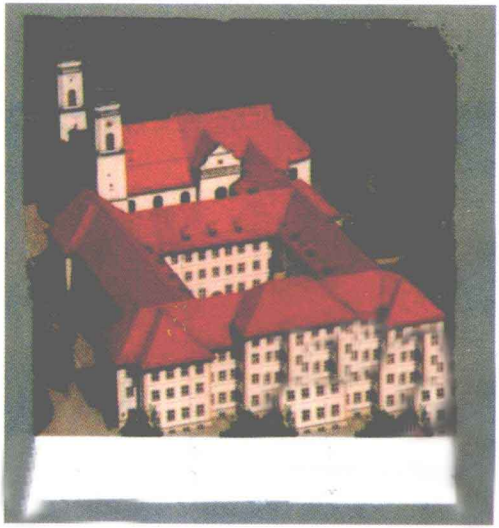
WILEY

Pattern-Oriented Software Architecture Volume 5

On Patterns and Pattern Languages

面向模式的软件架构

模式与模式语言



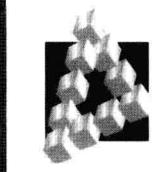
卷5

[德] Frank Buschmann
[英] Kevlin Henney 著
[美] Douglas C. Schmidt

肖鹏 等译

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

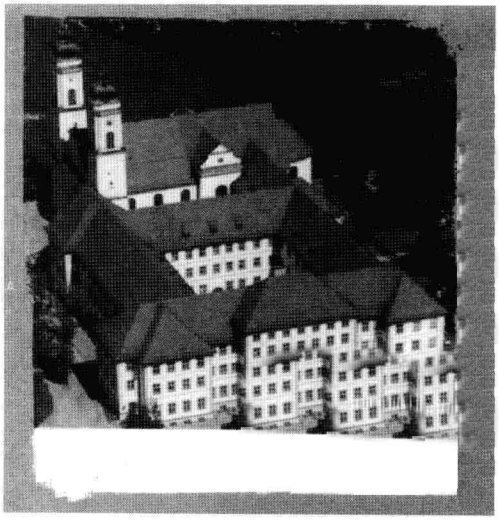


Pattern-Oriented Software Architecture Volume 5

On Patterns and Pattern Languages

面向模式的软件架构

模式与模式语言



卷5

[德] Frank Buschmann
[英] Kevlin Henney 著
[美] Douglas C. Schmidt

肖鹏 等译

人民邮电出版社
北京

图书在版编目（CIP）数据

面向模式的软件架构. 第5卷, 模式与模式语言 /
(德) 布施曼 (Buschmann,F.) , (英) 亨尼 (Henney,K.),
(美) 施密特 (Schmidt,D.C.) 著 ; 肖鹏等译. -- 北
京 : 人民邮电出版社, 2011.9
(图灵程序设计丛书)
书名原文: Pattern-Oriented Software
Architecture Volume 5: On Patterns and Pattern
Languages
ISBN 978-7-115-26173-1

Ⅰ. ①面… Ⅱ. ①布… ②亨… ③施… ④肖… Ⅲ.
①软件设计②面向对象语言－程序设计 Ⅳ. ①
TP311.5②TP312

中国版本图书馆CIP数据核字(2011)第161944号

内 容 提 要

本书共分3部分，首先介绍了单个模式，详细阐述了过去累积的关于如何描述和应用模式的诸多见解，接着探究了模式之间的关系，从组织的角度说明了各个模式的领域，最后介绍了如何将模式和模式语言相结合。

本书适合软件架构师和开发人员阅读。

图灵程序设计丛书

面向模式的软件架构 卷5：模式与模式语言

◆ 著 [德] Frank Buschmann [英] Kevlin Henney
[美] Douglas C. Schmidt
译 肖 鹏 等
责任编辑 王军花

◆ 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 电子邮件 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：17.75
字数：430千字 2011年9月第1版
印数：1－3 500册 2011年9月北京第1次印刷

著作权合同登记号 图字：01-2008-0487号

ISBN 978-7-115-26173-1

定价：59.00元

读者服务热线：(010)51095186转604 印装质量热线：(010)67129223
反盗版热线：(010)67171154

版 权 声 明

谨以此书献给Anna、Bebé和Martina。

——Frank Buschmann

谨以此书献给Carolyn、Stefan和Yannick。

——Kevlin Henney

谨以此书献给Lori和Bronson，也献给我的爸爸和妈妈。

——Douglas C. Schmidt

谨以此书献给John。

——Frank Buschmann、Kevlin Henney和Douglas C. Schmidt

译 者 序

本书是《面向模式的软件架构》系列（*Pattern-Oriented Software Architecture*）的收官之作，其重点不在于具体的模式或者模式语言，而在于模式之间的关系和表现。作者深厚的功力和丰富的经验确保本书值得每个关注架构和设计的人收藏和阅读。

对于我而言，除了对模式和模式语言有了更加深刻的理解之外，我从这部书中学到的更有价值的是作者分析问题的思路。作者反复在模式、模式组合、模式序列、模式语言等各个不同层次上采用类似的分析方式展现问题的本质；他抽丝剥茧，以过程的方式来分析问题；他还在不同的地方使用了框架的方法来辅助分析问题。凡此种种，对于我日常的开发和咨询工作都带来了莫大的裨益。

没有做过大部头翻译的同学大概很难体会翻译的痛苦。这本中文版能够呈现在你的面前，其背后无数个周末和假期，我的大部分业余时间或者贡献给了这本书，或者是在本书尚未译完的愧疚阴影中度过的。感谢我的家人在本书（包括卷4和卷5）翻译过程中给我的支持和理解。当你看到本书的时候，我的孩子已经出生了，在他（她）出生前陪他（她）的时间太少了，希望以后可以补回来。

我非常荣幸能够得到这本书的翻译机会。这样一部经典著作对我来说挑战大于机会，我的整个过程都是小心翼翼、如履薄冰。特别感谢本书的审校，限于译者的水平，审校想必更加痛苦。

从卷4翻译开始到现在已经将近4年了，正是在这期间我加入了ThoughtWorks。这是一个神奇的公司，这里有很多人让我受益匪浅。特别感谢我的同事荣浩、李彦辉、乔梁、张凯峰、韩楷，他们或者帮我翻译了其中的部分章节，或者在审校中帮我指正错误，或者跟我讨论其中的技术细节。

本书第2、3、9、10章由李锐翻译，第4章由乔梁翻译，第5章由李彦辉翻译，第14章由郭晓刚翻译，其余部分由肖鹏翻译，全书由肖鹏通稿。由于译者水平有限，翻译错误或者风格不合口味在所难免，在此我谨代表本书的译者对给你造成的阅读上的不便表示歉意。

你的任何意见既可以发表到我的博客（http://xiaopeng.me），也可以通过邮件直接发给我，我的邮件地址是——eagle.xiao@gmail.com。

感谢并期待你的批评和指正！

Richard P. Gabriel序

“软件模式已大大改变了我们设计的方式……”POSA5在其开头“关于本书”中如此写道——这是它的自白抑或可以看做它的判词。然而，在我们设计的过程中到底发生了什么？设计是关乎问题还是关乎美？对驱动力的化解是否在解决问题的同时也向设计中注入了美的元素？抑或原始材料只有经过（模式）语言加工之后，美才会随着理想解决方案一起浮现？有人曾经告诉我，如果一个建筑的入口显而易见，那么它就不值得造访。

在《牛津英语词典（第2版）》中，对“设计”（design）是这样解释的：

> 1. 一项智力计划。
>
> 1.a. 为了后续行动在头脑中形成的计划或者方案；有关某个即将实施的想法的基本概念；方案。

Guy Steele（盖·斯蒂尔）1998年在OOPSLA上的讲话“Growing a Language”中指出：

> 设计是关于如何构建某个事物的规划。设计是尚未在真实世界中构建之前在人的头脑中构建一个事物——或者，进一步说，就是计划如何构建真实的事物。

我曾经写道：

> 设计是构建之前所做的思考。

Carliss Baldwin和Kim B. Clark是这样定义设计的：

> 设计是一系列指示，它所基于的是将资源转变为有用、有价值的事物的知识。
>
> ——*Between “Knowledge” and “the Economy”: Notes on the Scientific Study of Designs*

以上这些定义和描述无不是围绕着先于构建时刻的想法、计划、知识、问题、价值和意图、线索而展开的。听上去合情合理，这也是我们希望设计所具备的特点——可靠。正因为它是由值得信赖的、接受过良好教育的人想出来的，因而能够给人安全感，不用担心它会左右摇晃甚至坍塌。但是，让我们回过头来考虑一下“我们”是谁。

几千年来，人们在“设计和建造”城市的过程中一直会使用各种工具和概念，比如Christopher Alexander的模式语言。这些作品宏大而复杂。对于像雅典、罗马、柏林、伦敦、伊斯坦布尔，甚至纽约、旧金山和波士顿这样的城市，我们似乎很难说清楚计划所在，因为它实际上发生在数年、数十年、数百年，乃至上千年的时间里。因此，本序言第一句话里所说的“我们”必然是指更接近现代的我们——软件设计者，这些人倾向于上述定义中那些更为保守或者不太现实的想法。即

使在我自己给出的那句定义里面——我尽量给设计、构建和反思留下了融合的空间，这样，即使对构建城市也同样适用——我们很难对构建之前的思考视而不见。

设计的广阔内涵以及含义上的细微差别让人不禁想起Sussex大学Adrian Thompson所定义的一系列设计的特征。我把这些特征称为设计元启发式理论，即有关设计的种种思维方式，比如如何使用模式和模式语言。设计元启发式理论有3种形式。先看第一种。

逆模型易处理：如果对于系统而言，存在易处理的“逆模型”，那么总有办法提前制定一系列变量以获得期望的结果。

这里，设计就是一个计划，或者说蓝图，逆模型是这样的：一旦待设计的对象确定了，总可以找到一个方法来确定如何构建出它来。这是预先设计的基本条件，适用于之前至少构建过一次的情况，通常都构建过很多次了。Alexander了解这种风格的工作方式：模式和模式语言的思想来源于工作在行业一线的人们（比如瑞士建谷仓的人自然知道瑞士谷仓相关的模式和模式语言），这是他们从小就耳濡目染、潜移默化的结果。从这个意义上讲，逆模型很容易确定：我们头脑中对未来谷仓的形象跟隔壁家的谷仓没有什么两样，构建步骤也大同小异。模式语言对于设计师和建造者具有（暗含的或者显式的）指导作用，而那些必不可少的适当调整正是常见的构建过程。这就引出了Thompson的第二个设计元启发式理论。

逆模型不易处理，但是正向模型易处理：这意味着我们可以根据目标值预测各种变化的影响，但是系统本身并不可逆，所以我们也就不可能提前确定一系列变量以获得期望的结果。这暗示我们采用迭代的方式工作，根据正向模型精心挑选的变量按顺序使用。这种迭代式的“设计—测试”的方式是最常见的、传统的工作方式。

设计一点、构建一点、反馈一点——这就是启发式的含义。我认为，这也是敏捷方法论、演进式设计/编程以及Alexander的“基本过程”（Fundamental Process）背后的本质。下面的内容摘自敏捷宣言。

> 欢迎需求的变化，即使在开发末期。敏捷流程通过驾驭变化保持客户的竞争优势。
> 频繁地交付可运行的软件，时间间隔从一两周到一两个月不等，越短越好。
> ……
> 最好的架构、需求和设计都源自自组织的团队。
>
> ——http://agilemanifesto.org/principles.html

Alexander在《秩序的本性》一书中用“基本过程”来描述设计和构建。在这部4卷的鸿篇巨著中，他讨论了如何将关注点组织成整体。下面是第(1)步、第(2)步、第(4)步、第(6)步和第(8)步。

(1) 在整个流程中——不论是构思、设计、制作、维护还是修复阶段——我们必须始终对于要做什么在头脑中有一个整体的认识。我们关注整体，理解它，努力探寻其深层结构。

(2) 我们不断地问自己接下来应该做什么对于整体最有好处。

(4) 在我们对新的居住中心做强化的时候，我们做事的方式跟创建或者强化那些更大的居住中心时的做事方式是一样的。

(6) 我们不断地检查所做的是否真正提高了整体的生活和感觉。如果整体的感觉未得到任何深化，我们刚做的事情就没有意义，应该退回去。否则，我们继续。

(8) 如果没有进一步可做的事情来深化整体的感觉，我们就停下来。

不论是Alexander还是软件开发人员，对于模式语言都是这样用的。这种用法只是说明第一种设计元启发式理论只适用于设计（大部分）是重复的情况。这时候，我们总是能够发现一些模型用来模拟“基本过程”中所描述的“设计一点/构建一点”的模式。或者真有一个模型——物理、数学、科学——可以主宰整个设计，并假装我们真能把设计给“试出来”，或者我们自欺欺人地认为“基本过程”中的大步骤（版本1、2、3）每个都可以看做是独立的设计。之所以说自欺欺人，是因为我们不愿意将视角拉远一点。Alexander的“基本过程”对于各种敏捷方法也是适用的，虽然有人可能认为我更应该提一下重构。

你手中拿的这本书（不知道你是否已经开始阅读）总体上是讲模式语言的。构建之前的思考可以用第(1)步、第(2)步和第(4)步来描述。在这种情况下，设计其实是为了解决问题。正如Jim Coplien所说：

设计是一系列有意识的活动，从发现问题到解决问题。所谓问题，是指目标状态和当前状态的差别。

这种说法似乎遗漏了（对某些人来说）很重要的两个方面：美和新。一个非常好的葡萄酒杯既实用又美观，它是由精致的、具有异国情调的材料制成的，它的形状能够恰到好处地衬托起杯中的美酒，其优雅的线条也令人赏心悦目——每一种美酒都有一种与之相配的酒杯，凝聚其芳香，提高其口感，握起来手感舒适。说到“新”（newness），我恨不得用“新颖”（novelty）这样的词语，但是“新颖”往往意味着知识或者学术上的创新，似乎不能切中要害。设计并不是发明一种全新的东西，而是当你看到一种以前没有见过的东西的时候，它带给你的是快乐和惊奇。我想到了1936年推出的Cord 810汽车。

Cord 810的车身是由设计师Gordon M. Buehrig领导的包括年轻的Vince Gardner在内的团队完成的。新车在1936年11月的纽约车展上引起了轰动。人们挤在Cord 810的周围，甚至有人站到别的车的保险杠上，只为一睹Cord 810的芳容。订单纷至沓来……

——维基百科

人们以前从没见过这样的汽车——前面棺形的车鼻，隐藏的头灯，（每个汽缸一个）铬合金的排气筒弯曲着伸出了两边，前面流畅的曲线——这简直不是设计师的作品，而是时尚的造型师的杰作。在外观、工程设计和用户体验方面最值得注意的有两点，一个是内陷的头灯，另一个是置于发动机前方的变速器，它使得乘客厢相对于当时的其他汽车来说更加平坦而且也更低。有人说，Cord 810的设计算得上是20世纪最独特的设计了。1996年，*American Heritage*杂志宣布Cord 810轿车是“最漂亮的美国车”。

举个例子罢了。

我觉得不能说Cord是为了解决某个问题而得到的产物：肯定有别的什么在里面。Rebecca

Rikner具体指明了这个“别的什么”。

> 设计不是关于如何解决问题的。设计关乎如何发现美。

大家可能认为有关Thompson的启发式理论，我已经说完了，其实不然，还有第三种情况。而第三种情况才是他研究的主题。

> **正向模型和逆向模型均不易处理**：既不能找到一个规律说明哪些变化有利于改善目标值，也无法预期变化会对目标值产生怎样的影响。离开了进化，一切都将消失。
>
> ——*Notes on Design Through Artificial Evolution: Opportunities and Algorithms*

进化！

Thompson曾经设计过一个模拟电路，使用遗传算法通过现场可编程门阵列（FPGA）来区分1kHz和10kHz的方波[①]。人们将其结果表述为“可能是迄今报道过的最离奇、最神秘、最突破常规的无约束演进电路”，这主要是因为最终研究人员也没搞清楚这个电路是怎么工作的。下面的评价，大家不必关心其细节，只需看看人们表达怀疑的方式就行了。

> 然而不知何故，在脉冲结束之后200ns之内，该电路就能“知道”它花了多长时间，尽管这期间它是完全没有响应的。
>
> 这确实难以置信，所以我们不得不通过各种独立的观察来佐证这一发现，所有的观察都表明在脉冲期间电路确实是没有响应的。

对于遗传算法（genetic algorithm）、遗传编程（genetic programming）、神经演化（neuro-evolution）、神经网络（neural net）以及统计/随机方法（statistical/aleatoric method）的研究均向人们展示了令人震惊的结果——有些结果甚至难以理解，比如由美国航天局（NASA）用人工进化方法设计并测试的天线，其大小如婴儿手掌，各个部件弯曲成一种奇怪的隐对称（cryptosymmetry）结构，就像黑色的灌木丛刚刚经历了一场冻霜。它看起来不像是人设计出来的东西，但是它在某些方面确实表现出更优秀的特性，所以从解决问题的角度来说，它算是一个好的设计。

但是我们有没有感觉到那是一个好的设计呢？我们应该怎样利用这类演化呢？人工演化是怎么工作的？简而言之，人工复制就是将来自于双亲的特征结合在一起。由此产生的特征还会发生少量的随机突变，所以包含了原始双亲大部分特征的结果，还要经过适应性测试。假设这些特征跟设计的某些方面相关，我们可以简单地说，它采用了某个看起来不错的解决方案中的一些设计元素，将其与另一个也还不错的解决方案中的设计元素结合在一起，然后再加入一些变化，看它是否可行。除了有一些随机的成分外，整体上听起来并不随意。

从我来说，这样“设计”出来的东西都有有意思的特征，问题是这种方式算不算是设计的一种形式。有人会说，不是由人类设计者作出的设计就不算设计，Coplien描述中的“有意识”和牛

① 见Adrian Thompson等人著的*Explorations in Design Space: Unconventional Electronics Design Through Artificial Evolution*。

津词典定义中的“头脑”是设计的基本特征。那么，“基于知识的指令”体现在哪里呢？如果把前述的FPGA鉴频器和诡异天线交给Pierre Menard[①]照搬一遍，质疑的人说不定就承认它们是设计了。或者可以进一步说，只有给出了人类可理解的解释，我们才承认它是设计。

这让我想到了定向发育（canalization[②]）的概念，它的描述是这样的。

> 如果绑定发育过程以产生一个特定的终止状态，而不论初始状态和发育过程中环境发生什么变化，那么其结果就是定向的。
>
> ——（改写自）Andre Ariew，*Innateness is Canalization: In Defense of a Developmental Account of Innateness*

某物一旦落入某条河流的支流，遵循一定的过程，它终将进入那条河流，我们将这样的产出物称为定向的产出。“定向”（canalized）来自于“运河”（canal）这个词——一旦进入了运河，便不可逃脱。复杂科学中的奇异吸引子（strange attractor）即属于定向的。人工演化并不关心设计或者至少是它的产出物是美丽、优雅、容易理解、成本低廉、可维护、可扩展、有序、光亮、简单，还是多余、丑陋等，它所关注的只有一点——是否符合功能要求。其他有关人的因素不会对定向设计产生任何影响，尽管人类的设计中总会多多少少考虑这些东西。

从另一个方向说，这个天线的设计有些诡异，因为它不像是人设计出来的东西。我们还没有能力穷尽整个设计空间，所以只能关注对我们来说看起来更靠谱的那部分。

本书从模式和模式语言的角度来谈设计。在阅读的过程中，先要把思绪集中在设计理念上，或许之后会发散到更远的地方。放任自己去感受设计的华美和真实。这3种元启发式设计理论或许构成了一个连续的统一体，如果确实如此，也将存在某种设计使人变成非人类（non-human），甚至后人类（post-human）。

模式语言包含了大量的设计空间。它们构成了软件设计和构建的工具箱的一部分。它们帮助人们作出更好的设计。对模式语言应用得如何反映了设计做得怎么样。设计，设计，设计，设计。

Richard P. Gabriel

---

① 出自博尔赫斯的短篇小说*Pierre Menard, Author of The Quixote*。在故事中，一位20世纪的作家［Pierre Menard（皮埃尔·梅纳尔）］立意重写塞万提斯的《堂吉诃德》中的部分篇章，而且要用跟塞万提斯的原文一模一样的文字，但这绝不是照抄，而是不谋而合。请看摘录：

将塞万提斯和梅纳尔对比一下就明白了。塞万提斯这样写道（第一部分第9章）：

……真理，是历史所孕育，是时间的敌人，是事迹的堆积，是往昔的见证，是当今的圭臬，是未来的辅佐。

“外行天才”塞万提斯17世纪写下的这一番排比，无非是对历史的虚辞赞美。

而梅纳尔却是这样写道：

……真理，是历史所孕育，是时间的敌人，是事迹的堆积，是往昔的见证，是当今的圭臬，是未来的辅佐。

历史孕育了真理：这个想法实在是令人惊奇。

作为与威廉·詹姆斯同时代的作家，梅纳尔不把历史看做事实的反映，而是将其视作事实的来源。历史的真相对他来说，不是发生过什么，而是我们认为发生过什么。最后一句话——当今的圭臬，未来的辅佐——更是具有赤裸裸的实用主义色彩。

其风格上的对比也非常鲜明。梅纳尔对于古体风格毕竟有点陌生，他这样写总有做作之嫌。塞万提斯就没有这种毛病，他对当时通行的西班牙语驾轻就熟。

② 生物/心理学名词，也译为限向发育、限向发展等。——译者注

Wayne Cool序

遥想当年，我遇到一群人，他们肩扛来自一个建筑狂人的异端思想的沉重大旗，奔波在编程革命的路上。那个狂人自己一心要通过重现过去的辉煌——不过这是指设计感（sense[①]）的辉煌，而非设计本身——来振兴未来。这伙牛人离群索居，与那些正统的大师不同，他们从不现身于主流的聚会，他们有自己的研讨会。他们退隐于世外桃源，那些地方虽然鲜有人知，但却值得一访[②]。他们并不像那个疯子一样整天把美挂在嘴边，他们撸起胳膊、挽起袖子，拿起螺钉、螺母，将实践与程序、实用性与严谨性紧紧拧在一起。他们的工作面向一线工作者，又未弃理论于不顾。

我随着PLoP会议转战南北，流连于各地的咖啡厅、酒吧和星巴克，见证了Beck的宣讲，见证了研讨会的结束，见证了阿勒顿的暴风雨，见证了伊尔塞旭日初升，见证了维肯贝里骑马之旅的尘埃落定，同时也见证了相关书的编写、审校、付印和赞美。在这中间，我看到了人们对于设计和人造物结构的兴趣慢慢增长，字里行间，人们似乎希望创建一个由抽象和想法构成的世界，而又希望这个世界像由木头、油漆、石头和金属构成的世界那样触手可及。

这就是我要谈的：软件模式将表象的世界（其内部机制隐藏在纷繁与复杂之中）转换成一个简单而透明的世界，隐藏的机制成为了精工细作的目标，那其实就是一种期望，期望把事情做得更好，但是只是为了做好而做好，不图回报，图的只是运用技能时的自豪感，而靠的是敏锐的洞察力和深入的思考。对于这些人来说，手艺隐藏于某种用户界面之下算不得什么——因为总有一天会有人拨开乌云，让其重见天日。这些表象尽管有意义，但是他们并不会把它隐藏起来。

艺术。手艺。工程。科学。这正是设计模式的灵感源泉。艺术和科学是故事，手艺和工程是行动。

手艺是介于艺术和科学中间的形式，艺术和手艺同工程和科学之间又是对立的。艺术是独特的、首要的，是对天赋与欲望浓缩的产物。手艺则是可靠的质量的产物。工匠[③]可能会失意，但是很少失败。手工艺品是由人采用某些材料制成的。工程是指人通过某些设施的帮助可靠而高效地生产产品。科学的产物通常在工程中采用。如果一本书叫做“什么什么的艺术”，它通常记录了这种东西的制作人的秘密。“什么什么的手艺”讲述的则是一个一个作出这些东西来的艰辛和汗水。“什么什么的工程”讨论的则是规则和长远计划，其产品通常是面向企业客户的。“什么什么的科学”是指像数学这样的书。

然而，科学和手艺的角色被人们误读了。很多人坚定地认为科学必然走在手艺和工程的前面，任何构建过程首先需要抽象的认知。比如，要设计一个蒸汽机，你必须懂得（正确的）热力学原

① 或者用敏捷阵营常用的那个词——“味道”（scent）。

② 向Bill Knott致歉。

③ 这回我用了个漂亮的词语craftman。

理。然而，在发明蒸汽机的时代，科学家们坚持“热质论”，即热的本质是一种称为“热质”（卡路里，caloric）的“微妙的流体”。整个宇宙中这种物质的数量是恒定的，它从热的物体传递给冷的物体。讲得不错。热质论能够解释很多现象，但是建造第一个蒸汽发动机的人根本就不知道这个理论，甚至都不关心什么理论。在使用锅炉的过程中，他们注意到了体积、压强和温度之间的关系，于是他们建造了蒸汽发动机。说不定科学家们正是通过对蒸汽发动机的观察才发展出了“现代”热力学理论。

工匠了解自己用的东西。他们会观察。领悟力和系统的思考使他们能够将自己的理解表达出来。成熟的手艺催使技术的发展。当然也包括科学。在科学和工程之间存在着反馈环，但是很难讲谁占有主导地位。

这都是有关构建的事情，这就要求要有计划和执行。计划是先于构建的思考。执行则是理解计划并生产出所要的东西。从20世纪早期开始，管理科学就强调要将计划和执行分开。Frederick Winslow Taylor是这样说的：

> 尽可能将脑力劳动从工作现场剔除，集中到计划和规划部门。
>
> ——科学管理原则（*Principles of Scientific Management*）

真贴心！

计划和执行在艺术领域几乎融为一体，而在工程领域则分得很远，在手艺方面则介于二者之间。科学则属于思想和思路领域。

将计划与执行相分离，并不是为了效率。也不是为了得到最大的价值。也许能，也许不能。计划需要思考，思考也不是免费的。如果你能做很少的思考，而构建出大量的产品，你多半能挣到钱。流程代替了执行过程中的思考，培训代替了教育——Dancing Links和X[①]算法就是这样解决数独问题的。有了这两个算法，解决数独问题只需要一台计算机，不需要人。将计划与执行相分离，考察的是成本。

这也不是为了质量。质量最好的汽车绝对不会是机器人生产线造出来的。成本。

如果成本是采用工程的动力，那么模式则是将我们拉回到手艺和艺术的力量。

模式。匠艺：我们认为它来自人们建成这个世界的纹理之中。在这个世界里，质量和对细节的注意体现在事物的表面之上，体现在恰当的缝隙之间，体现在巧妙而自然的连接之中。在这个世界里，天赋总是伴随着知识和智力。我们来看一下George Sturt在*The Wheelwright's Shop*[②]中对如何制造轮辋（木质车轮边缘部分）是怎样描述的。

---

① Dancing Links是由Donald Knuth提出的高效实现X算法的技术。X算法是一个递归的、非确定的、深度优先的暴力算法，用来找到完全覆盖问题的所有解决方案。

② 在1923年首次出版。

然而，此时谈论过多的细节是徒劳的。通过一个看似简单的流程就把这些简单的装置组装到了一起，但材料带来的变化却是无穷无尽的。当两个轮辋完成的时候，它们有多大的相似性呢？车匠必须让它们相似。他把两块完全不同的轮辋墩子做成了相似的轮辋，因为从来不存在两个一样的墩子。这儿有几个结，那儿有几道裂纹[①]、几块伤疤[②]，高低不平[③]的边缘，过厚或者过薄，种种情况总是带来新的可能或否定原有的解决方案，频频挑战工人的创造力。他没有带锯可用（目前，1923年），只能用最原始的方式来处理眼前的一切。木材不是机器面前无辜的牺牲品，而是让懂得迁就它的人体会到它特有的品性。

——*The Wheelwright's Shop*

你可以像木工那样感受木头。

模式是那些对代码有系统认识的人总结出来的，他们深入理解代码，并长期以作出正确的设计和编码为己任，他们不相信计划可以与执行分开。正是由于这类人的存在，科学才得以发展。这不仅仅是贩卖抽象的理念，这是思考。艺术和工艺的进步并非来自于模式，而是来自于以模式为工具的人。需要抵制的是模式的自动化运行，这会将计划和执行隔裂开来。

计算机科学家站在黑板前写写擦擦、左顾右盼、高谈阔论。模式爱好者关注于手法，躬身于代码之中，埋头于鼠标键盘之间。他们工作的原材料也是：这儿有几个结，那儿有几道裂纹、几块伤疤，高低不平的边缘，过厚或者过薄，种种情况总是带来新的可能或者否定原有的解决方案。他们几乎将编程看做一项体力劳动，袖子卷得高高的，露出结实的臂膀，炯炯有神的眼睛背后是他们闪光的思想指导着双手。

◆ ◆ ◆

模式社区发起伊始，我便在其中了。我站在山头；我躲在林间；我奔跑于海滩；我漫步于胡德山；我驾着小艇，穿过绳网，倾听着日落的歌声；我蒸着桑拿，大汗淋漓；我骑着大马，像野人一样飞驰；我和牛仔们一起分享牛排和玉米。但是，我却未曾贡献只言片语，人们的想法和著作中连个标点符号都没有我的份。然而，我相信你们知道我是怎么想的，对不对？

Wayne Cool于威尼斯海滩

① 由风霜造成的木材中的裂纹。

② 树木早年间遭受的损伤，导致树干中生出新的树皮。

③ 从不太方正的原木上切下来的板子边缘过于锐利或者不均匀，或者在试图切成方形的过程中也可能出现这种情况。

关 于 本 书

软件模式已大大改变了我们设计、实现和思考计算系统的方式。模式给我们提供了用于描述架构愿景的词汇，以及清晰明了、切中要点的典型设计和详细实现示例。使用组成模式描述软件，可以使我们使用更少的词汇进行更为有效和清晰的沟通。

自20世纪90年代中期以来，很多软件系统（包括Java及C#编程语言和库的主要部分）都是在使用模式进行开发。以前模式有时会被选择性地用于解决特定的挑战和问题，或者从基线架构的定义到细粒度细节的实现来支持软件系统的构建。今天，模式的使用则成为软件专业人员的至爱。

过去的15年里，有大批的文献从软件开发的方方面面阐述了已知的模式，内容涉及组织和流程、应用和技术领域以及最佳编程实践。这些文献为软件工程师提供了具体的实践指南，并日益影响学生的教育。每年都有关于更多模式的书和会议纪要发表，以模式的形式从深度和广度上扩展软件开发知识。

同样，我们关于如何应用模式方面的知识和经验在稳步增长，模式概念本身相关的知识亦是如此：内在的属性，不同的风格以及与其他技术的关系。自20世纪90年代中期以来，尽管在软件模式领域中概念性的知识飞速增加，记录和重构具体模式的出版物的数量也在不断增长，而关于模式及模式概念的出版物却只在少数领域稍有更新。《面向模式的软件架构（卷1）》[POSA1]、《设计模式》[GoF95]以及《软件模式》白皮书[Cope96]中关于软件模式的介绍仍是模式概念最相关的来源。另外，最近才有出版物开始明确讨论和阐述模式序列[CoHa04][PCW05][Hen05b]。

总而言之，在模式概念方面没有紧跟技术发展的著作出现。此外，关于模式的概念性基础知识的最新发展也仅停留在模式社区少数专家和思想先驱们的头脑中。本书的目的在于挖掘并阐述模式相关的知识，以满足广大软件开发社区的需求。同时本书是“面向模式的软件架构”系列丛书的第5卷，也是丛书的最后一卷。

本书将呈现、讨论、比较和关联众多模式概念中已知的特性和应用：单个模式、模式互补、模式复合、模式故事、模式序列和模式语言。对于每个概念特性，我们将研究其基础属性和高级属性，探究模式社区广为接受的见解，以及目前仍在探讨和争论的观点。我们还将讨论模式与软件开发中广泛使用的其他技术如何相互作用和支撑。简而言之，我们纵览了软件模式知识和实践当前发展的最新动态。

尽管如此，读者需要注意到我们从总体上对模式概念本身进行了宽泛的阐述和探讨，我们用来举例说明或激发模式不同方面的各具体实例，主要将重点放在软件设计模式上，而不是诸如组织模式、配置管理模式和特定的应用领域模式之类的其他模式上。上述（自我）限制的原因有两个：首先，大多数已经文档化的软件模式都是软件设计模式，因此我们有丰富的材料用作举例；其次，软件模式的最大用户群是架构师和开发者，因此将重点放在软件设计模式上使我们能够使

用这个群体中最熟知的实用的例子，来解释模式背后的“理论”。

目标读者

本书的主要读者是对模式的概念性基础原理感兴趣的软件专业人员。我们的首要目标是帮助上述软件专业人员，从深度和广度上完善他们对模式概念的认识和理解，从而使他们知道模式能够给他们的项目带来什么，以及如何对他们的项目有所贡献。其他的目标是帮助软件专业人员避免常见的模式方面的错误概念，并且将具体的模式有效地应用到日常的软件开发工作中。

本书同样适合在软件工程、编程语言、运行时环境和工具方面有扎实基本功的本科生或研究生。对于这类读者，本书能够帮助他们了解什么是模式，以及模式如何对设计和实现高质量的软件有所帮助。

结构和内容

本书分为3部分，每部分又由几章内容来展开阐述其观点和内容。

第0章揭示了模式概念定义的来龙去脉，探讨了软件社区中是如何衍生出这些模式的定义的，以及如何理解这些定义。我们的分析表明调整和改进模式的相关概念，对于深入理解模式和防止在软件项目中错误使用模式大有裨益。作为介绍性章节，本章详尽阐述和讨论了模式相关概念的调整和改进，给读者提供了一个更为完整、一致的模式概念视图，为本书后续的3个主要部分奠定基础。

第一部分讲解了单个模式的用法，从整体上剖析了过去10多年中我们收集的关于模式的观点。这些观点作为现有模式定义的补充，有助于我们从更深层次理解模式。

第二部分将讲解的重点从单个模式转移到模式之间的关系：有时几个模式组合在一起成为另外一个模式的替代或补充，有时这些模式紧密组合成为一个模式集合。除了常见的关于模式集合的观点之外，该部分还探讨了如何将模式组织为模式序列，使用一个接一个的模式形成模式描述流，从而在设计流程中动态有效地使用模式。

第三部分在前两部分的概念和结论的基础上介绍了模式语言。在为特定技术或应用领域设计和实现软件时，与单个模式和模式序列相比，模式语言更加从整体上使用模式。为达到该目的，模式语言针对相关领域中的每个问题选择多个模式，将它们组织在一起，形成一个高效的领域相关软件开发流程。

第14章撷取了前面3部分讨论的模式相关概念，得出如下结论：尽管模式相关技术为其他软件技术提供支撑，但模式的主要受众还是人。

第15章回顾了在2004年“面向模式的软件架构”系列丛书的第3卷中关于模式未来发展趋势的预测，讨论了过去3年里模式真正的发展方向，分析了模式及模式社区的现状。基于对模式以往发展历史的回顾，我们重新修订了关于模式及模式语言未来发展趋势的预测。

本书是我们计划出版的POSA系列丛书的最后一卷，至少目前是如此。第16章概括了过去超过15年的模式相关工作和经验，检查了在这期间我们所著的5卷POSA系列丛书。

本书最后总结了我们所讨论过的模式概念。最后简短概括了本书使用的所有模式，列出了所有对完成本书有所贡献的参考文献。

毫无疑问，我们难免遗漏个别的模式概念的属性和定义，新的模式概念也会随着对模式及其在实际软件开发中应用的理解不断深入而出现。如果你对本书的内容和写作风格的提高有任何评论、建设性意见或建议，请发电子邮件到siemens-patterns@cs.uiuc.edu。在模式主页http://hillside.net/patterns/可以找到模式相关的注册指南。该链接同样提供了关于模式方方面面的重要信息来源，如已出版和即将出版的书、模式相关的会议及论文等。

致谢

非常感谢在完成本书的过程中支持我们的诸位，他们有的分享他们的知识，有的帮忙审校本书的各部分草稿并提供有用的反馈。

首先，谨以此书献给John Vlissides，以表达我们的谢意。John是软件模式领域中最为才华横溢的大师之一，他是软件模式领域开辟新天地的思想领导人和探路者，同时也是《设计模式：可复用面向对象软件的基础》[GoF95]一书的合著者，以及当今众多世界级知名软件模式专家的顾问。他的著作给予我们灵感以及模式概念的基础，深深地影响我们，并极大地帮助我们在本书中详细讲解和讨论模式的相关概念。

审校奖的桂冠颁发给Wayne Cool、Richard P. Gabriel、Michael Kircher、James Noble和Linda Rising，他们仔细地审校了本书的所有内容，从正确性、完整性和一致性上保证了本书的质量。他们的反馈极大地提高了本书内容的质量。Wayne Cool还贡献了本书深入探讨的诸多观点和思路。

另外，本书的部分内容在四届EuroPLoP模式会议上进行了演讲，并呈送给几位模式专家。Alan O'Callaghan、Lise Hvatum、Allan Kelly、Doug Lea、Klaus Marquardt、Tim O'Reilly、Michael Stal、Simon St. Laurent、Steve Vinoski、Markus Völter、Uwe Zdun和Liping Zhao给予我们大量的反馈，使我们更正了大大小小的关于模式概念诸多定义和表述的错误。

非常感谢Mai Skou Nielsen，允许我们在本书中使用她作品中的照片。

特别感谢Lothar Borrmann和Reinhold Achatz在管理上给予的支持，以及德国慕尼黑Corporate Technology of Siemens AG软件工程实验室给予的帮助。

尤其感谢我们的编辑Sally Tickner、前任编辑Gaynor Redvers-Mutton以及John Wiley & Sons的其他人员。没有他们，就没有本书的出版。在参加2002年EuroPLoP会议时，Gaynor于某个晴朗的夜晚说服我们著本书，并陪伴我们度过了写作过程的前两年。在接下来的两年里，Sally以极大的耐心陪伴我们完成了本书的手稿。同样要感谢来自WordMongers公司的文字编辑Steve Rickaby，他为本书的内容增色不少。Steve以他的建议和支持，陪伴我们完成了这5卷POSA书。

最后，感谢我们的家人，感谢他们在本书写作过程中的耐心和支持！