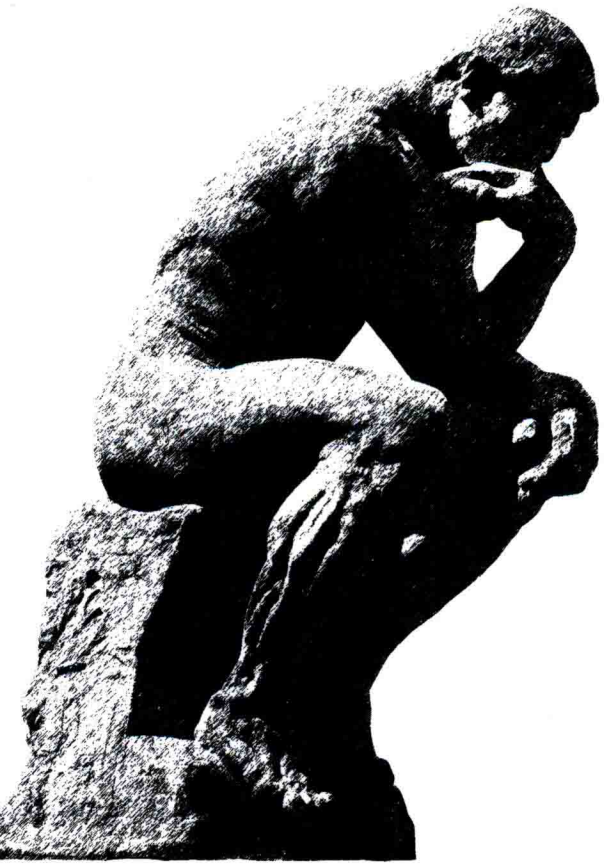
PEARSON

PATTERN HATCHING
DESIGN PATTERNS APPLIED

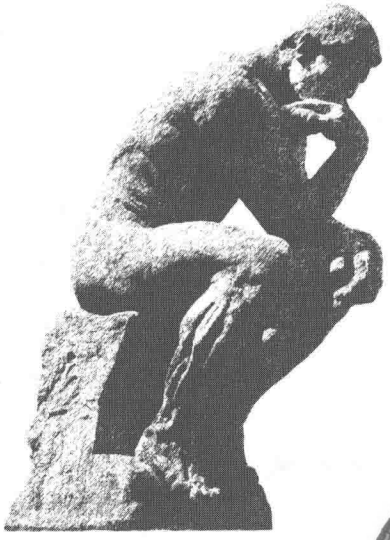
设计模式 沉思录

[美] John Vlissides 著
葛子昂 译



中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

PEARSON



PATTERN HATCHING
DESIGN PATTERNS APPLIED

设计模式沉思录

[美] John Vlissides 著
葛子昂 译

人民邮电出版社
北京

图书在版编目（CIP）数据

设计模式沉思录 /（美）威利斯迪斯(Vlissides,J.）著 ；葛子昂译. -- 2版. -- 北京 ：人民邮电出版社，2015.8
书名原文：Pattern Hatching: Design Patterns Applied
ISBN 978-7-115-36721-1

Ⅰ. ①设… Ⅱ. ①威… ②葛… Ⅲ. ①软件设计 Ⅳ. ①TP311.5

中国版本图书馆CIP数据核字(2015)第154977号

内 容 提 要

本书在GoF的《设计模式》一书的基础上进行了拓展，运用其中的概念，介绍了一些技巧，帮助读者决定在不同的情况下应该使用哪些模式，以及不应该使用哪些模式。本书不仅对已有的一些模式提出新的见解，还让读者见证开发新模式的整个过程。

本书适合使用设计模式的软件开发人员阅读。

◆ 著 ［美］John Vlissides
译 葛子昂
责任编辑 杨海玲
责任印制 张佳莹 焦志炜

◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路 11 号
邮编 100164 电子邮件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
固安县铭成印刷有限公司印刷

◆ 开本：720×960 1/16
印张：10.75
字数：173 千字 2015 年 8 月第 2 版
印数：1－3 000 册 2015 年 8 月河北第 1 次印刷
著作权合同登记号 图字：01-2009-4225 号

定价：35.00 元

读者服务热线：(010)81055410 印装质量热线：(010)81055410
反盗版热线：(010)81055410

版权声明

译者序

我接下本书的翻译工作的时候，正值《Windows 核心编程（第 5 版）》翻译完成，中文版新书上架后没多久。翻译《Windows 核心编程（第 5 版）》耗时 9 个多月，这是与另外两位译者一起合作完成的一本 700 多页的大部头。这项工作让我感觉有些疲惫，原本想好好歇一歇，不料却邂逅本书。

最终，我无法拒绝，接下了本书的翻译工作，主要出于两方面的原因。首先，作为一名设计模式的拥护者和践行者，我非常高兴地看到另一本与设计模式有关的重要著作在问世 10 多年后即将在国内出版，同时也希望自己能够为国内的设计模式社区尽一份绵薄之力。其次，本书的英文原版不到 200 页，在翻译了《.NET 设计规范》和《Windows 核心编程（第 5 版）》两本译作之后，我满怀信心地认为自己应该可以很快完成。

事实很快证明我错误地估计了翻译本书的工作量，而且错得不轻。本书虽然短小，但却当之无愧地是我翻译过的最难的技术书籍，以至于在开工后不久我不得不向出版社告急，要求对原先的翻译进度和出版计划进行调整。这本不足 200 页的书最终花了我将近半年的时间才完成，这远远超出了我的预期。出版社的编辑工作同样反映了本书的难度，译稿在 2009 年上半年就已交付给出版社进行编辑，但最终出版上架却要到 2010 年初，其难度可见一斑。

正是由于这个原因，最初交付的中文版译稿并未能完全达到我希望的流畅程度。对我来说，虽然我尽量追求译文的准确和流畅，但当两者不能兼得时，我会牺牲语言的流畅来保证内容的准确。在此我要感谢本书的编辑陈兴璐小姐和图灵公司的总经理武卫东先生，是他们的编辑和润色使得本书变得更加流畅。即便如此，书中难免还会存在一些生涩的字句和不够流畅的地方，甚至是错误之处。作为译者，我对此负有全部责任。为此我建立了一份网上勘误表，

如果读者发现任何错误，都请通过该网页与我联系，一旦确认我会立即将其更新到勘误表中。勘误表的网址为 www.gesoftfactory.com/ge/ PatternHatching/。

我要感谢我的同事吴宇进、田超和张险峰，是他们在繁忙的工作之余审阅译稿，提出了许多宝贵的意见和建议，从而使得本书的质量更上一层楼。最后，我要感谢我的妻儿，他们的支持和鼓励，是我前进的动力。

本书不仅通过一些通俗易懂的实例对如何运用设计模式进行了深入的讲解，而且还介绍了一些新的设计模式。但与其他的设计模式书籍相比，本书的独特之处在于向读者揭开了模式开发的神秘面纱，让读者了解模式背后鲜为人知的一些故事，并领略其中的苦与乐。我满怀着激动和忐忑之情，将这本设计模式领域的重要著作的中文版呈现给国内的广大读者。希望本书能够帮助你们更好地理解和运用设计模式，甚至有朝一日编写出自己的模式！

葛子昂

2010 年 3 月

序

John 写信告诉我，他打算为 C++ Report 撰写模式专栏，他的这个决定填补了我生命中的一个空白。具体来说，他一年填补了大概 5 个空白——那时我正在撰写一个关于模式的专栏，每两个月一期，Stan Lippman 建议我和 John 轮流撰写。John 主要关注设计模式，而我在专栏中将继续关注更为广泛的主题。我们俩搭档把模式介绍给 C++社区，对此我感到兴奋。不仅如此，我也喜欢 John 介绍这个主题的方式。我在给他的信中写到：

> 孵化（hatching）的比喻不仅讨人喜欢，而且很有道理。我刚刚又阅读了 Alexander 的 *Notes on Systhesis* 第 2 版的前言，显然他认为对那些潜藏在自然中的事物，我们应该去挖掘和发现它们，而不是关注创造它们的“方法”，甚或是从其璀璨的裂缝中窥探它们。

能够与 GoF（模式四人组）之一共同写专栏，我不仅感到高兴，更感到荣幸。如果没有 GoF 的《设计模式》一书，读者也许都不曾听说过模式。通过对模式的介绍，该书成为了这个全新科目的极佳教材。GoF 的 23 个设计模式奠定了一个不大但却非凡的基础，并发展壮大成为我们今天知道的模式社区。而凭借本书读者可以直接深入了解 GoF 作者之一 John 的思考过程，同样对更加广阔的过程有一些间接的了解。

为了总结出一个好模式，突破一些局限在所难免，就像小鸡破壳而出，而 John 的专栏对那些在《设计模式》背后发生的“破壳”的对话进行了探索。例如，John 在类结构不断演变的情况下，对 Visitor 的限制进行了探索。他还谈论了一些模式，比如 GENERATION GAP（见本书的第 3 章）。这些模式未能被收录到《设计模式》中，但它们可能已经足够好了，值得公之于众。读者会发现 GoF 关于 MULTICAST 模式的对话，这段对话让 John 陷入沉思：“一旦了解

了我们在模式开发过程中所经历的混乱，那些认为 GoF 具备非凡能力的人一定会感到震惊。”本书传达了一个重要的事实，它没有出现在更为学术化和更加完善的《设计模式》一书中：模式源自一群认真努力的程序员，虽然他们不可能每次一开始就把事情都做对，但他们努力地让那些重复出现的设计实践变得实用。我认为阅读本书的模式用户将会感谢 GoF 为他们的模式而付出的心血，我还认为阅读本书的模式编写者在今后发掘和编写模式时会比以前更加谦逊和勤勉。

乱中求序是自然科学的主旋律，那么，我们不应该认为设计的科学会有任何的不同之处。模式是人们在工作中共同发现一些构成人们高品质生活的因素，并将它们加以记录的整个过程。这不可避免是个有机过程。贯穿本书，读者将得以洞察各模式背后的有机过程，得以了解普通（但非常有经验而且非常尽职的）软件开发人员在努力形成自己对设计的理解时的思考过程。《设计模式》是对他们集体理解的提炼，而本书是对产生理解的过程的提炼，我们不应该低估它在解释 GoF 模式方面所带来的价值。请允许我引用一封我在 1997 年晚些时候收到的来自 Richard Helm 的信，我相信它进一步证实了这一点。

> GoF 的设计模式只解决了微观架构（micro-architecture）。你仍然必须把宏观架构（macro-architecture）设计好：分层、分布、功能隔离……。而且就像 Cope 说的，你仍然必须把纳米架构（nano-architecture）设计好：封装、Liskov……。在所有这些中的某个地方，你也许会用到一个模式，也许用不上。即使用上了，也可能和某本书中的介绍和描述很不一样。

这本书将帮助你理解如何将《设计模式》——其实是任何关于设计模式的书籍——当作一本珍贵的指南，而不是当作一些累赘的规定。它可以帮助你在更广阔的面向对象设计的基本原则下，将设计模式运用到合适的地方。它道出了虽然不正式但却严格的标准和紧张的迭代过程，《设计模式》中的 23 个模式正是基于这样的标准，经历了这样的迭代过程产生的。知道有这样的过程，以及这样的过程如何发生，让人感到释然，因为它把模式带回到更加讲究实用的日常工作中。我认为这将有助于读者认识到必须根据手头的问题来对模式进行调整，有助于读者加入自己的思考而不仅仅是盲目地遵循“某本书中说过的”

教条。我不认为计算机科学家会喜欢这本书，但现实的程序员会反复品味，获得共鸣，并高度欣赏它。

James O. Coplien
朗讯科技公司
贝尔实验室

前　言

我永远不会忘记 1994 年秋天的那个下午。那天我收到一封来自 Stan Lippman（时任 C++ Report 杂志的主编）的电子邮件，他邀我为该杂志撰写一个专栏，该专栏每两个月一期。

我们算得上是老相识了，早在他参观 Watson 实验室的时候我们就认识了。那一次我们简单地聊了他在开发工具方面所做的工作，以及 GoF 在模式方面所做的工作。与那时大多数人不一样的是，Stan 熟悉模式的概念——他接连阅读了《设计模式》的一些预览本，并说过一些令人鼓舞的话。尽管如此，我们的谈话很快就转移到了写作上。随着谈话的进行，我记得自己愈加炫耀起来，仿佛我已经不是自己了。而 Stan，这位知名的专栏作家，是两本非常成功的图书（还有一本即将出版）的作者，却称自己的写作只是业余水平。我不清楚我们的谈话是否让他感到愉快，还是在他的下一个约会之前他一直都在耐着性子和我谈话。（此后我认识到，如果还有什么能胜过 Stan 的忍耐力，那就是他的真诚！）

几个月后我收到他的电子邮件，心潮起伏，此前的歉疚感就不值一提了。想象着自己为全球的读者定期撰写专栏，这既让我兴奋，又让我恐惧。写了几次之后我是否还能继续？人们是否在乎我写些什么？我应该写些什么？我写的东西对别人是否有帮助？

我在恐惧中沉溺了将近一小时。然后我想起我父亲的一些告诫：局促不安只能使人无所作为。只要关注最基本的东西，其他东西会随之而来的。“只管去做”（Just do it），他说这句话比耐克要早得多。

于是我就接受了。

选择专栏主题非常容易。那时我深陷于模式的研究中已有三年了。我们最近刚完成《设计模式》，但我们都知道它远远没有说完这个话题。专栏会是一个很好的论坛，可以对《设计模式》一书进行解释，可以对它进行扩展，还可以在新问题出现时展开讨论。如果说专栏有助于《设计模式》图书的销售，那也无妨，只要它立场公正，不乱吹嘘。

现在，我的“模式孵化”专栏已经连载了 10 多篇文章了，回过头看，我的恐惧是没有依据的。我从来没有因为要找东西写而为难，而且写作时我乐在其中。我还从世界各地收到了大量令人鼓舞的反馈，包括一些人要求阅读过去的专栏，而且这样的要求一再出现。后来我想到了把我的专栏，以及其他一些尚未发表的关于模式的材料，汇编在一起提供给大家。

本书就是要达到这个目的。读者将在书中找到我前三年专栏写作生涯中的思考和想法，其中包括发表在 C++ Report 和 Object Magazine 中的所有文章，加上一些零碎的新见解。我按照逻辑的顺序来组织内容，而不是通过时间顺序来组织内容，其目的是为了使所有的内容能够像书本一样连贯。这样的组织比我想象的要容易一些，因为许多文章既是这个系列的一部分，又是那个系列的一部分，当然这仍然需要耗费大量的精力。我衷心地希望读者能够喜欢最终的结果。

致谢

一如既往，我要感谢许多人为我提供各种各样的帮助。首先最重要的是我的 GoF 成员——Erich Gamma、Richard Helm 以及 Ralph Johnson。他们每一个人都在不同的时刻为我提供了宝贵的反馈，这些反馈汇集在一起使本书成为了一本更加不同（当然是更好）的图书。我们几人互补性强，如同天成，遇见他们是我三生有幸，我由衷地感谢他们。

然而，同样的帮助也来自其他人。还有许多人花时间研读草稿，为的是找出不合逻辑的论述、不当的言辞，以及大家都再熟悉不过的笔误。他们是 Bruce Anderson、Bard Bloom、Frank Buschmann、Jim Coplien、Rey Crisostomo、Wim De Pauw、Kirk Knoernschild、John Lakos、Doug Lea、Bob Martin、Dirk Riehle 以及 Doug Schmidt。特别感谢 Jim，他是我在 C++ Report 的拍挡，不仅因为他为本书作序，更因为他是如此多才多艺，总是激励我奋进。

接下来要感谢的完全是一些陌生人，他们给我发电子邮件问我问题，提出意见，纠正我的错误，并给我以善意的责备。有许多这样的人，在这里我只列出了本书引用了他们的话的人，或者他们的意见与本书直接相关的人：Mark Betz、Laurion Burchall、Chris Clark、Richard Gyger、Michael Hittesdorf、Michael McCosker、Scott Meyers、Tim Peierls、Paul Pelletier、Ranjiv Sharma、David Van Camp、Gerolf Wendland 和 Barbara Zino。虽然很多人我没有提到名字，请相信我同样感谢你们的反馈。

最后，我要感谢两个家庭，一个是我自己的家人，另一个是我亲如一家的同事，你们对我的支持我无以言表。我欠你们的太多了。

J.V.
vlis@watson.ibm.com
1998 年 1 月于纽约州霍索恩市

目 录

第1章

介绍

在阅读本书之前，如果读者还没有听说过一本名叫《设计模式》（Design Patterns: Elements of Reusable Object-Oriented Software [GoF95]）的书，那么现在正好可以去找一本来读。如果读者听说过该书，甚或自己还有一本但却从来没有实际研读过，那么现在也正好应该好好研读一下。

如果你仍然在继续往下阅读，那么我会假设你不是上述两种人。这意味着你对模式有大致的了解，特别是对 23 个设计模式有一定的了解。你至少需要具备这样的条件才能够从本书受益，这是因为它对《设计模式》中的材料进行了扩充、更新和改进。如果不熟悉所谓的 GoF 模式——也就是刚才提到的那 23 个设计模式，那么读者将会很难理解书中的见解。事实上，在阅读这本书的时候，最好备一本《设计模式》在身边以便随时查阅。我还要假设你熟悉 C++，这么假设应该很合理，因为我们在《设计模式》一书中也做了同样的假设。

现在让我们来实际检验一下，看你是否能用不到 25 个字来描述 COMPOSITE 模式的意图。你可以考虑一分钟。

※　※　※

在描述 COMPOSITE 模式的意图时，你是不是句斟字酌？如果确实如此，好，没问题。其实你大可不必把这个小测验看得太严重了，不妨放松一点。如果你知道该模式的意图，但只是无法准确描述出来，那么不必担心——本书同样适合你。

但如果你的头脑中完全是一片空白，那么很不幸，我的开场白显然没有起作用。我建议你放下本书，拿起一本《设计模式》，从第 163 页[①]开始阅读，直到读完“实现”那一节。对该书第 xv 页中列出的其他模式，也采取同样步骤。如此一来，你就可以对背景知识有足够的了解，从而使得本书对你有用。

你可能会想，为什么本书会取名为 *Pattern Hatching*？我最初选择这个名字，是因为计算机科学中有类似的概念。（此外，与模式有关的好书名都已经被用了。）但此后，我逐渐认识到它非常好地表达了我写作的意图。*Hatching* 之意并非创造，而是在现有的基础上进行拓展。用于此处非常贴切：如果把《设计模式》看成一盒鸡蛋，那么许多新生命将会在这里破壳而出[②]。

请相信，我并不仅仅是效仿《设计模式》。我的目的是在它的基础上进行拓展，运用其中的概念，并使这些概念对读者更加有用。本书介绍了一些技巧，来帮助我们决定在不同的情况下，应该使用哪些模式以及不应该使用哪些模式。本书不仅对我们已有的一些模式提出了新的见解，还可以让读者见证我们开发新模式的整个过程。本书还提供了大量的设计示例，其中一些是久经考验的，另一些是试验性质的，或者说是“半成品”，还有一些则完全是主观推测——很可能是纸上谈兵的设计，根本经不起实践的检验，但它们也可能会蕴含未来健壮设计的种子。

我衷心地希望本书能够加深你对设计的感悟，提高你对模式的认识，并开阔你软件开发的视野。这些都是我在使用模式时曾有过的经历，希望它们也能成为你自己的宝贵经历。

1.1　对模式的十大误解

这些日子，模式引起了大家强烈的兴趣，同时还伴随着一些迷惑、诧异和误解。这在一定程度上体现了主流软件开发人员认为这个领域有多么新，虽然从严格意义上说，它并不是一个新领域。这个领域的快速发展，也造成了一些空白。作为模式的倡导者，我们对此负有一定的责任：我们虽然一直努力让大

① 指此书英文版原书页码。——编者注

② 我相信不会有比这个比喻更贴切的了。

家理解和接受模式（[BMR+96、Coplien96、CS95、GoF95、MRB98 和 VCK96]），但是工作并不彻底。

为此，我感觉自己有义务来纠正那些对模式比较明显的误解，这些误解我常常耳闻，甚至可以自成模式了。我甚至还开玩笑地采用模式的形式来表述它们……直到那一刻我幡然醒悟：将任何事物都归纳为模式，这种行为本身就是对模式的一种误解！无论如何，请记住我并不是代表模式社区在发言。虽然我认为大多数模式专家都会同意这些是对模式最常见的误解，但就如何消除这些误解而言，他们的意见可能会与我的相左。

这些年人们对模式众说纷纭，令我反复思考，众多误解不过分为三类：一类有关模式是什么，一类有关模式能够做什么，还有一类有关一直以来在推动模式的社区。我所列举的“十大”误解都可以被归到这三类中。因此，我会将它们分门别类。首先来看看关于模式是什么的误解。

误解 1：“模式就是在一种场合下对某个问题的一个解决方案。”

这是 Christopher Alexander 的定义[AIS+77]，因此把它算作一种误解可能会显得有些离经叛道。但下面这个反例应该能够显露出它的不足。

问题：如何在过期之前兑现中的彩票？

场合：离最后期限只有一小时，一条狗把彩票吃了。

解决方案：剖开狗的肚子，取出彩票，然后飞奔到最近的兑现点。

虽然这是在一种场合下对一个问题的解决方案，但它并不是一个模式。那它缺少了什么呢？至少需要三样东西。

（1）再现（recurrence），这使得该解决方案不仅与当前场合下的问题有关，而且与当前场合之外的问题也有关。

（2）教学（teaching），这将教会我们去理解怎样对解决方案加以完善，从而适应问题的变体。（对实际使用的模式来说，与教学有关的大部分内容都包含在对问题的描述、对解决方案的描述以及应用模式后得到的结果中。）

（3）一个用来指代模式的名字。

诚然，一个令所有人都满意的定义是很难找到的，从“pattern-discussion”