PEARSON
Addison
Wesley

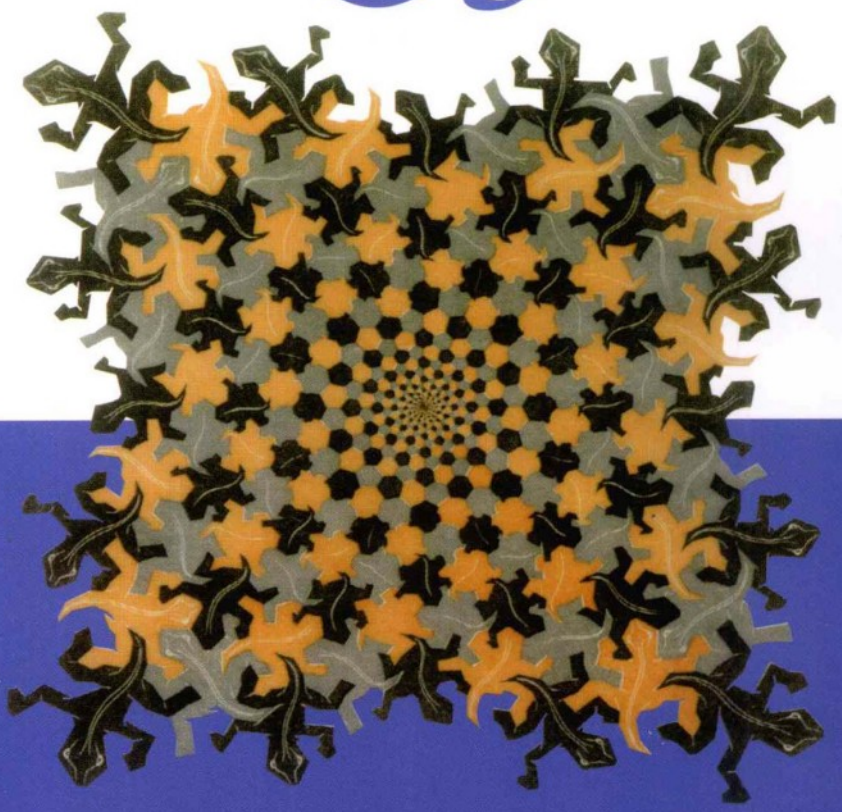
Pattern Languages of Program Design

程序设计模式语言

卷5

Dragos Manolescu
Markus Voelter
James Noble
著

陈宗斌 等译



清华大学出版社
PEARSON Education

Pattern
Languages of
Program Design

程序设计模式语言

卷5

Dragos Manolescu
Markus Voelter
James Noble

著

陈宗斌 等译

清华大学出版社
北京

Simplified Chinese edition copyright © **2009 by PEARSON EDUCATION ASIA LIMITED and TSINGHUA UNIVERSITY PRESS.**
Original English language title from Proprietor's edition of the Work.

Original English language title: Pattern Languages of Program Design 5 **by** Dragos Manolescu, Markus Voelter, James Noble, Copyright© **2009**
EISBN: 0-321-32194-4
All Rights Reserved.

Published by arrangement with the original publisher, Pearson Education, Inc., publishing as Prentice-Hall.

This edition is authorized for sale only in the People's Republic of China (excluding the Special Administrative Region of Hong Kong and Macao).
本书中文简体翻译版由 Pearson Education(培生教育出版集团)授权给清华大学出版社在中国境内(不包括中国香港、澳门特别行政区)出版发行。

本书封面贴有 Pearson Education(培生教育出版集团)激光防伪标签,无标签者不得销售。
版权所有,侵权必究。侵权举报电话: 010-62782989　13701121933

图书在版编目(CIP)数据

程序设计模式语言·卷5/(美)马诺勒斯库(Manolescu,D.)等著;陈宗斌等译.—北京:清华大学出版社,2009.7
书名原文:Pattern Languages of Program Design 5
ISBN 978-7-302-20017-8

Ⅰ.程… Ⅱ.①马… ②陈… 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(2009)第 063354 号

责任编辑: 龙啟铭
责任校对: 徐俊伟
责任印制: 王秀菊

出版发行: 清华大学出版社　　**地　　址:** 北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编:** 100084
社　总　机: 010-62770175　　**邮　购:** 010-62786544
投稿与读者服务: 010-62776969,c-service@tup.tsinghua.edu.cn
质　量　反　馈: 010-62772015,zhiliang@tup.tsinghua.edu.cn
印 刷 者: 北京四季青印刷厂
装 订 者: 三河市兴旺装订有限公司
经　　销: 全国新华书店
开　　本: 185×230　　**印　张:** 28　　**字　数:** 637 千字
版　　次: 2009 年 7 月第 1 版　　**印　次:** 2009 年 7 月第 1 次印刷
印　　数: 1～3000
定　　价: 59.00 元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题,请与清华大学出版社出版部联系调换。
联系电话:010-62770177 转 3103　　产品编号:030630-01

译者序

作为提高软件工程、系统设计与开发效率和质量的一种极其有效的方法，模式语言受到越来越多程序设计人员和相关行业的重视。模式收集提炼了许多优秀软件的设计经验，提供给所有软件工程师。

本系列丛书(《面向对象软件开发》)已经出版了4卷，本书是该丛书的第5卷，集合了来自程序模式语言会议的一流模式。本书的全部内容都来自各届PLoP，很多人为此作出了卓越的贡献。

本书共分6部分19章，各章内容大致如下。

第Ⅰ部分为设计模式。其中，第1章介绍了动态对象模型，解释了如何利用流行的面向对象语言将新的类型动态地引入应用程序，而无需设计新的类；第2章介绍了域管理器，允许应用程序通过编写代码来控制域对象，同时支持多种数据存储或应用程序服务器；第3章介绍了封装上下文，说明如何在不引入全局变量的情况下管理越来越多的调用参数。

第Ⅱ部分为并发、网络与实时模式。其中，第4章介绍了用于高效、可预测及可伸缩的调度组件的模式语言，这些模式为分布式实时中间件和嵌入式中间件提供了组成部分；第5章以汽车制动系统为例，介绍了硬实时系统中用于可靠通信的模式系统；第6章介绍了实时与资源重载语言，其中的模式取自电信运营等级系统。

第Ⅲ部分为分布式系统。其中，第7章介绍了分散锁定；第8章介绍了比较模式，使用专门的值进行简单的身份标识；第9章介绍了发现服务的模式语言，它提供了以实际方式解决公共问题的方法。

第Ⅳ部分为特定于域的模式。其中，第10章介绍了移动与无线资源管理的模式语言(MoRaR)；第11章介绍了Web上的内容转换与生成的模式语言，进一步提供基于Web的应用程序设计的便利性与通用性。

第Ⅴ部分为体系结构模式。其中，第12章介绍了插件模式，涵盖了众多主体软件的技术，包括操作系统、Web浏览器、绘图程序及开发环境；第13章介绍了网格体系结构模式，包括体系结构网格中间件的元素，以及实现和部署它们的指导原则；第14章介绍了组件及语言集成模式，以便为待要解决的问题提供最有利的工具；第15章介绍了成功框架开发的模式，便于程序设计人员同时建立应用程序及框架。

第Ⅵ部分为元模式。其中，第16章介绍了高级模式编写，为提高模式的编写水平提供了很有价值的建议和借鉴；第17章介绍了语言设计者的模式语言，帮助设计人员从有限制的系

统中生成模式语言；第18章介绍了审阅语言，关注、分析并提供对模式的反馈；第19章介绍了草原式住宅模式，通过建筑界著名的草原式住宅向人们展示了如何接近和编写模式。

随着模式由研究领域逐步发展到应用于实际软件开发中，越来越多的开发人员认识到可重用的设计模式有助于更快更经济地开发应用程序。而设计模式是软件开发人员开发软件所使用的模式语言。

希望通过本书的学习，读者能够理解和掌握模式语言的精髓，并逐步应用到程序设计与软件开发中，提高开发的效率与水平。

特别注意：由于有些模式的名称容易与其功能和作用发生混淆，因此为了让读者既能理解模式的含义和作用，又不至于混淆模式名称与其作用，在第9、11、12、14、16、17、18章中，标题中的模式名称采用了中文形式，而正文中的模式名称则采用了英文形式。

参加本书翻译的人员有：陈宗斌、陈红霞、张景友、易小丽、陈婷、管学岗、王新彦、金惠敏、张海峰、徐晔、戴锋、张德福、张士华、张锁玲、杜明宗、高玉琢、王涛、申川、孙玲、李振国、高德杰、宫飞、侯经国、刘淑妮、张春林、李大成、程明、张路红、张淑芝、孙先国、刘冀得、梁永翔、张广东、郁琪琳、邵长凯、蒲书箴、潘曙光、刘瑞东、李军、焦敬俭、赵中元、金鑫、赵宏伟、张宏顺、尹周、王开年、贾震、陆晓萍、金国良、俞群等。

由于时间紧迫，加之译者水平有限，错误在所难免，恳请广大读者批评指正。

此为试读，需要完整PDF请访问：www.ertongbook.com

致　谢

我们要感谢以下自愿帮助我们完成本卷的模式语言专家：
Alejandra Garrido　位于 Urbana 平原的伊利诺伊大学
Andy Longshaw　Blue Skyline 有限公司
Arno Haase　Arno Haase 咨询公司
Arno Schmidmeier　AspectSoft 公司
Brian Foote　Refactory 公司
Brian Marick　Exampler 公司
Charles Weir　Penrillian 公司
David Trowbridge　Microsoft 公司
Diethelm Bienhaus　Cooperative Education North Hesse 大学
Dirk Riehle　Bayave 软件公司
Doug Lea　位于 Oswego 的纽约州立大学
Douglas C. Schmidt　Vanderbilt 大学
Dwight Deugo　Carleton 大学
Erich Gamma　IBM 公司
Eugene Wallingford　北爱荷华大学
Fabio Kon　圣保罗大学
Gregor Hohpe　Google 公司
Jens Coldewey　Coldewey 咨询公司
Jim Coplien　曼彻斯特大学及 DAPCA 公司
Joe Yoder　Refactory 公司
Joel Jones　阿拉巴马大学
John Vlissides　IBM 公司
Kevlin Henney　Curbralan 公司
Klaus Marquardt　Dräger Medical
Kyle Brown　IBM 公司
Neil B. Harrison　Utah Valley 州立学院
Patricia M. Carlin　ThoughtWorks 公司

Peter Sommerlad　HSR Hochschule fur Technik Rapperswil
Ralph Johnson　位于 Urbana 平原的伊利诺伊大学
Richard P. Gabriel　Sun Microsystems 公司
Robert S. Hanmer　Lucent Technologies 公司
Uwe Zdun　维也纳经济与农业大学

此外，Dragos Manolescu 非常感激 John Vlissides 委托他编辑本卷，Markus Voelter 和 James Noble 的加入使本书更完美；Neil Harrison 和 Brian Foote 分享他们编辑本系列丛书第 4 卷的经验；Chris McMahon 对各章摘要的评价；Mara、Nora、Traian 和 Beth 多次在去洛基山的短途旅行中携带 PowerBook 处理本书的原稿。James Noble 感谢 Katherine 和 Amy 让他最终完成本书。

Dragos Manolescu，Lawrence 于美国堪萨斯
Markus Voelter，Heidenheim 于德国
James Noble 于新西兰威灵顿

前　言

On the first day of EuroPLoP, my workshop said to me,
"Delete everything on page three."

很奇怪，在软件设计中采用设计模式至少已有 10 年。Erich Gamma、Richard Helm、Ralph Johnson 和 John VUssides 的《设计模式》一书——软件设计中模式的最初文献——于 1994 年底向可信任的世界（或至少是在 OOPSLA 会议上）发布——虽然对传统的大图书出版商而言，其版权日期是 1995 年。Jim Coplien 的《软件模式》和 1996 年后出版并产生持久影响的《软件体系结构模式（卷 1）》（由 Frank Buschmann、Regine Meunier、Hans Rohner、Peter Sommeriad 和 Michael Stal 编写），设置了模板（或蓝图），即人们会遵循的模式。受到这些大范围成果的启发，许多其他作者也开始研究和编写模式，1994 年以编程模式语言为中心议题的系列会议 PLoP 第一次在伊利诺伊州的 Monticello 举行；紧接着，1995 年在 Kloster Irsee 举行了 EuroPLoP 会议；然后，又进一步扩展到在世界各地举行了一系列相关会议。在这一期间，我们都老了：很多人的头发都少了。

设计模式——所有工作都是同时进行的，随后受到支持或反对——实际上有两个关键问题。首先，要成功地理解和分类，应证明软件的设计。其次，要解释和命名设计，以使它们能够被他人理解、传达和采用。最后，软件设计被称为"模式"：小的、独立的、相互依存的设计片段，可以以各种方式与其他的模式和编程技术集合，生成许多完整的程序设计，就像语言中的文字可以以多种方式结合，产生无穷的句子、段落、小说或书籍的前言。本书是《程序设计的模式语言》系列丛书中的最新一卷，也是模式灵活性、多样性和开放性，以及围绕它们不断发展的社会的最好证明。

On the second day of Europlop, my workshop said to me.
"Too many patterns."

本卷是本系列丛书的第 5 部分，集合了来自程序的模式语言会议的一流模式。这些会议是同类工作会议中独一无二的。他们不是被动地工作、也不会无视用户，相反，这些模式作者积极接受来自同行的反馈。在作者工作的团队中，一组模式作者讨论草案，并提供建设性的批评。

本书的所有内容都在1998～2004年PLoP会议上研讨过。我们从PLoP and ChiliPLoP会议(美国)、Eu-roPLoP and VikingPLoP会议(欧洲)、KoalaPLoP会议(澳大利亚)、SugarLoaf PLoP会议(巴西)和MensorePLoP会议(日本)邀约了一流的建议。在为本次收集添加独特色彩时,为了保持PLoPD丛书此前各卷的传统,编辑团队由3位不同背景的编辑组成(他们来自不同的大洲),他们都是自己所负责模式的作者,也是模式社区论坛的老成员了。

过去和现在的模式作者都十分欢迎PLoPDS呼吁大家积极贡献自己的成果。因此,我们共收到超过70多篇为本卷而作的论文。内容极其广泛,涉及多方面的体系结构、分析和设计模式,例如基于Web的应用程序、安全、中间件、分布式工作流、有限状态机、模式的模式(即元模式)等。为很好地理解和把握这些内容,我们邀请一些经验丰富的模式社区成员帮助我们进行甄选。每篇提交论文的质量都至少由3位评审人员评审,并由编辑最终决定它包含在哪个模式中,也可能或者我们中的两个人(或其他两人,这取决于参照系)在结束该项目后说,"这全是Dragos的过错。"

On the third day of Europlop, my workshop said to me,
"Three known uses."

该如何信任本书中的模式呢?又为什么应该信任它们呢?部分答案就在上文所述的审核过程中:本书中的模式不只是某个程序员持续熬夜和喝了太多咖啡的疯狂想象,它们的确是某个程序员持续熬夜和喝了太多咖啡的疯狂想象——但同时也经过模式专家们的严格审核和编辑。

实际上,本书中的模式已远远超过其倡导者、工作组、研讨会和评审者所提供的内容。我们希望,在你学习这些模式时,能够识别每种模式的3个已知用途:至少有3个文档化实例的描述,这些实例是本书模式作者没有设计或实现的,也尚未在其设计中明确包含这些模式的实例。

很显然,在最初的《设计模式》中,几乎所有模式的3个已知应用是Smalltalk、E T++与交互。这就证明,面向对象设计贯穿整个软件产业——这种进步毫无疑问归功于《设计模式》一书的成功——本书中模式的这3个已知应用更为多样化。Smalltalk和C++图形用户界面框架无疑仍然"潜伏"于表面之下,而本书所介绍的模式更为广泛,包括面向对象系统、中间件、管理、通信、武器系统、战斗机、元模式等,不胜枚举。但原则仍然不变:这些模式是可信赖的,这并不是因为作者或我们编辑人员假设如此,而是因为每种模式至少在3个已知的经过实践获得成功的系统中被证明是可信的。

On the fourth day of Europlop, my workshop said to me,
"Four forces."

设计模式不仅仅是软件设计的描述，那是 UML 类图的工作。Christopher Alexander 开发了描述体系结构设计的模式，他将模式定义为“描述一定背景下问题的解决方案的三部分规则”。然而，软件模式不仅仅是一个描绘恰当的问题和解决方案的说明（解决问题的 3 种已知用法）。

好的模式应当首先使其读者相信它所描述的问题是真实的、有力的，并且值得花费时间和精力去妥善解决的（因此编写模式首先要考虑的是时间和精力）。其次，它要使读者对其解决方案感到惊喜——就是 Ward Cunningham 所说的“啊！的时刻”（Aha! moment）。最后，好的模式还要作出可靠的结论：解决方案为什么能够解决实际问题，以及如何解决。

Alexander 将应用模式时需要考虑的重要因素称为“限制条件”。模式可能会分解限制条件，在其应用程序之后使其不那么重要，或在使用特定的模式表现它们，要求程序员在思考不需要事先考虑它们。好的模式不仅讨论它的优点是什么（它的好处和要解决的限制条件），还要讨论其缺点是什么（其义务和其所要揭示的限制条件）。模式不只提供介绍，还要提供论证、分析和理由，说服编程人员为什么提出的解决方案是可靠的、可重复的和可信的。

On the fifth day of Europlop, my workshop said to me,
“Five good things, four forces, three known uses, too many patterns,
and delete everything on page three.”

100 年前是爱因斯坦“奇迹般的一年”。他发表的几篇论文改变了相对论、量子理论和分子理论。他的论文《On the Electrodynamics of Moving Bodies》（移动物体的电气力学）介绍了狭义相对论（Special Relativity）理论和一种理解空间与时间之间关系的新方法。今天，大多数人都同意以爱因斯坦的相对论代替艾萨克·牛顿的绝对空间和时间理论，并认为这是爱因斯坦天才的表现。然而，单靠天才不足以形成其理论和解决难题。狭义相对论和广义相对论的提出依赖于爱因斯坦的前辈和同辈人的贡献，其中包括 Galileo Galilei（伽利略）、Ernst Mach、Carl Friedrich Gauss（高斯）、Georg Friedrich Bernhard Riemann、Gregorio Ricci-Curbastro、Tullio Levi-Civita、Hermann Minkowski 和 Hendrik Antoon Lorenz。

虽然软件开发与理论物理有很大不同，但它也要解决复杂性问题。从第一款可编程机器的出现至今，我们已经走过了漫长的道路。今天，参与软件开发的人必须在很多不同的情况下处理复杂问题。他们建立基于 Web 的分布式应用程序；编写支持可移植设备的软件；建立可实时接收和处理事件的系统；编写应用程序，让其他人以从未想过的方式加以扩展；还为整个产品线设计软件。他们设计的系统必须能适应广泛的异构环境。就像狭义相对论和广义相对论一样，如果从零开始、不利用其他人开发的思想和技术，很难甚至不可能解决所有这些问题。软件模式（如在本卷和本丛书其他卷说介绍的）代表了这些必须整理和解释关于软件设计的想法和技术的最好尝试。希望你能找到它们的用途。

为了尽量包含为本卷所选的论文，本书分为6部分。第Ⅰ部分着重介绍针对设计对象系统的设计和所包含的模式。随着Internet和嵌入式系统影响的扩大，出现了并发和资源管理问题。第Ⅱ部分主要介绍针对这些问题的模式。第Ⅲ部分从一个应用程序扩展到多个应用程序，并包含分布式系统的模式。第Ⅳ部分是关于特定域的模式，侧重于移动电话和基于Web的应用程序。第Ⅴ部分概括了体系结构，以及解决组合、可扩展和重用问题的模式。第Ⅵ部分为提高模式论文质量和帮助相关作者，提供了元模式的综合资料。

此为试读，需要完整PDF请访问：www.ertongbook.com

导　言

真令人惊讶，从举行第一届名为 PLoP 的模式会议并出版了第 1 卷收集编程模式（绰号 PLoPD）工作成果的书籍至今，已经 10 年了。在这样一个关于软件的最惹眼的新想法也会在短短的几年内迅速过时和被遗忘的世界，软件模式的开发继续蓬勃发展是非常有意义的。然而，有意思的是并没有谁从该技术中获利致富。

即使缺乏推广这些模式的经济刺激，相关的消息仍在传播。虽然没有良好的投资营销，仍有许多天才专业人员志愿用他们的时间和才能来大力发展这些想法。至今，已经在五大洲 9 个国家的 11 不同地区举行过有关模式的会议。

面对 10 年的成功（这在我们的领域是很少见的），我们应该停下来，看看在这一过程中可以找到哪些值得借鉴的东西。或许，有些模式有待就如何开展其他活动加以提炼，或帮助模式社区维持其使命和发展动力。业内已开始接受回溯程序作为一种反映和借鉴已有经验的方法，我认为这是组织未来新内容的完美手段。

回顾建立在审查历史的基础之上，所以可以利用以前 PLoPD 各卷的介绍材料来了解这一长达 10 年之久的过程。从 1995 年出版的第 1 卷编程模式开始，我们就很清楚："成立 PLoP 会创建一种新的文献。创始人……已经意识到，这些规则的发展会受限于……科技出版物的传统：它们更青睐新的、最近的发明或发现，而平常的发现无论多么有用都不会那么受重视。"

因此，建立起了社区模式（Patterns Community），以写明哪些工作可以很好地借鉴从业人员的经验。这是一个很受欢迎的想法，短期内已迅速在世界范围内引起人们发现、编写和出版该领域最优成果的兴趣。

几十名作者开始以各种模式格式和方法介绍他们的最佳工作成果。随着所收集的关于模式的论文不断增加，研究人员逐渐了解以模式的形式表示解决方案所带来的影响。1996 年 PLoPD 第 2 卷出版，其中介绍了这一新的研究领域的发展。

人们关注的重点也从"模式是关于什么的？"转向"是什么使一种模式成为好模式，或者使好的模式更好？"这深刻的变化表明人们对于模式的想法迅速成熟。这是从自我思考向健康活动的转移。现在，许多人认识到，模式的价值不在于其发现和定义，而在于其相关性、质量和影响力。软件中的模式是与其技术本身同样了不起的文献资料。

1998 年，模式继续流行，模式名称成为软件开发人员词汇表的主要组成部分，在这种情况下，本系列的第 3 卷应运而生。从最畅销的《设计模式》书籍中找到模式并在通用软件文献中引用是很常见的，与此同时，许多其他的模式和模式语言悄悄地工作着。本系列第 3 卷前言的

作者对于这一领域的成功十分欣喜，但也警告我们要注意一个新问题："Robert Martin 在芝加哥会议上向相当多的观众介绍了设计模式。他问谁买过《设计模式》一书，约有 80% 的人举手。然后，他让实际上并没有好好阅读这些书的人把手放下，大概一半的人放下手了。接着他又问谁能解释 Visitor（访问者）模式，几乎所有的手都放下了。"

很显然，模式已经声名狼藉，许多程序员都知道这一新技术，但很少有人真正掌握进而很好地利用模式。这是为什么？多数努力工作的软件专业人员为了跟上不切实际的进度表，常常要超负荷工作，为此他们会购买了有关模式的书籍，寄希望于能由此最大程度地改善自己的职业生涯，但却发现其中只有很少的部分可以借鉴利用。我们的专业需要不断发展成熟，以适应继续教育的预期，这里面甚至包括有雇主和管理人员的需求。但是，这一思路主要在有关模式的子系列文献中介绍，所以还要快速回顾一下模式发展的历史。

在 2000 年，关于 PLoPD 第 4 卷的介绍材料表明，对模式的理解和应用经验都已成熟，并重视编写普通模式这一领域的最佳做法。此时，尽管模式作者协会与世界各地的参与者仍旧继续大力发展模式，所有媒体关注的焦点已远离模式。

作为一名被认为过时的提倡模式的先锋，我很困惑为什么相关的讨论社区还在持续发展。然而，这一活动新的领导者和一些新加入的模式作者解释说，"很难说清，但确实有一些这样的文化团体。"

从一开始，模式就接受和支持规范，并鼓励所有模式都遵守这些规范。事实证明，这一方式能够在世界范围内被跨文化的人群所接受，因为它是在于世界各地召开的 PLoPD 会议基础上重建的。我们的原则和实践，有助于个人在安全和有力的环境中，从技术和编写方面开发自己的技巧，PLoPD 第 4 卷介绍的就是这些内容。

"PLoP 是一个与众不同的会议，因为我们着眼于软件开发中人类活动所涉及的人。各次会议都花很大的精力创造一个舒适宜人的环境。它们是唯一安排午睡时间的会议，EuroPLoP 限制参加人数，这并不是具有排他性，而是要确保私密性和每个小组的凝聚力。所有的 PLoP 会议都会提供一些社会活动和服务，帮助与会者建立一个社区。"

经过 5 年的共事，我们已意识到模式社区有一种特殊的相互影响的方式。我们都知道这种方式在起作用，但说不清"它"是什么。那之后又过了 4 年，Jim Coplien 才在其学术论文《Culture of Patterns》（模式文化）中解释了这一神秘现象。

现在可以结束我们对历史的快速回顾了，但回顾不仅是过去事件的故事而已，它还是从以往的事件中总结教训和学习经验的过程。回顾的方法之一就是对精心挑选的问题探索答案。

问题 1：什么做得比较好并且是决不能忘记的？

我们的目标是基于软件领域的工作经验，创建一种新的文献种类。不是因为 Donald Knuth 的经典著作中有那么多众所周知的智慧。PLoPD 系列不仅是很好的模式源，而且许多关于模式影响的书籍已经出版，它们包括如下相关议题：

- 小内存系统

- 实时嵌入式系统
- 分布式系统
- 服务器系统
- 大型系统
- 并发系统
- 需求
- 软件体系结构
- 软件重用
- 重构(refactoring)
- 集成
- 测试
- 配置管理
- 生产线建设
- 团队组织
- 在组织中进行改变
- 企业体系结构
- 特定语言的知识:C++、Smalltalk、Java 等

因此,我们决不能忘记我们的任务是收获、提炼、编写、学习并在我们的职业生涯中很好地利用这些。

问题 2:我们学到了什么?

模式运动的创始人十分喜欢两种简单的做法(实践 practice),只是很少讨论。而我认为这两种做法正是模式社区充满活力和长期存在的原因:①安排审阅者帮助作者将论文精炼为可指导工作的文献;②模式论文在作者工作组的研讨会中加以讨论,而后呈献在被动的读者面前。

指导者帮助作者的方式不同于评审人员评审论文的传统方式,有时很痛苦。作者的工作小组尊重编写工作,他们在小组循环讨论中给予每位作者聆听和宣读自己论文的难得机会。

除了这两种简单的想法外,社区规范还鼓励遵循各种想法,即使那些想法与自己的想法相背。从新手到专家,编写工作中每位读者的意见都具有同等价值。我们建立了一个社区,在这里你可以安全地表达自己的想法,甚至对于你自己不理解的内容也同样可以发表看法。

问题 3:下一次要做的与以往有何不同?

在模式运动的早期,很难找到好的想法并编写出高质量的单一模式。作为专业人员,我们有意识地研究同事的工作,希望能借此找出共同的东西,将其反映到我们自己的工作上,然而我们几乎没有什么技术可以借鉴。

因此，早期成功的模式文献都是单一模式和单一模式的集合。诚然，这些工作对这一专业产生了深刻影响，它们同时还为后来的研究人员指明了方向。就像一个淘金矿工找到了天然金块模式——忽略了模式潜在的更有价值的关联系统，这些可参考 Christopher Alexander 有关“模式语言”的介绍。

单一模式可能会显著改善系统的一个关键部分，或对软件体系结构的某一个小方面产生重要影响，但只有模式语言能够全面处理严重的问题和复杂的系统。正是通过模式语言，该领域的领军人物才能与我们分享他们一生的研究成果。该专业里一些了不起的软件设计师不断建立一个又一个系统——其中的很多人我们甚至不知道他是谁，又怎么能指望向他们学习呢？

我想下一次我要做的与以往不同的是更强有力地呼吁寻找模式语言，同时仍然十分重视已发现的单一模式。我会尽力改正这一点。

当许多同行致力于单一模式的研究时，我悄悄地对模式语言进行实验，发现对模式系统的思考和研究非常有价值。为了从我多年的经验中提炼出最主要的知识，我开始开发一个回顾的模式语言。在工作进行过程中，模式语言不断使我原有的想法产生突破，而且往往在我努力修复一个单一的大模式时，会被引导发现更进一步的知识和多种新的模式。通过这种方法，可以发现和完善一个具有关联概念的复杂系统，而不会被淹没其中。

回顾模式语言完成时，我对工作过程非常满意，但又担心模式语言的形式与大部分读者没有关系。因此，我将我的模式语言“翻译”为更常见的“容易阅读”的书籍《Project Retrospectives》提供给大家。对于熟悉模式语言的读者来说，逆向推理我书中的模式语言是很容易的，但在讨论社区中会无法了解构建模式语言是揭示和组织复杂知识的一种强大方法。

我在探索模式语言的道路上并不孤独——Ward Cunningham 在其 EPISODES 模式语言（PLoPD2）的开发中也开发出了极限编程（eXtreme Programming）。我怀疑近期出版的其他书籍也是建立在模式语言推理的基础上。

虽然在 PLoPD 系列的其他卷中也可以找到语言模式，但正是第 5 卷最终完整集合了模式语言。我们已经开始发现 Christopher Alexander 生活工作中真正的实用工具了。

问题 4：还有何难题？

我们最困惑的是“如何为目前面临的解决方案找到正确的模式。”虽然我个人偏向于模式的语言，但我承认我的同行已开发出大量极有意义的单一模式和模式联合。早期大约有 100 种已发布的模式，我能记住其中的大部分，而且也很容易了解和部署它们。

随着发布的模式不断增减，现在已经有了几千种。我发现我几乎无法在需要时找到理想的模式。

2000 年初，Linda Rising 及时地在她的《The Pattern Almanac》一书中解决了这一难题。Linda 通过艰苦的工作对所有已知发布的模式进行分类并给出参考，她做得很好。我也经常参考她的书。但是，出于很多与该书本身无关的原因，本书没有出版。

尽管如此，Linda 的努力表明图书馆学中的学科分类原则可应用于模式，而且必须如此，

否则越来越多的模式将得不到充分的应用。我认为，这是我们面临的下一个挑战——将原有的大量评论论文整理到文献中。

问题 5：我最大的乐趣是什么？

过去 10 年来，模式研究者们已经建立起真正的伙伴关系——这对于该领域而言十分珍贵。我相信这个团体已经帮助许多有能力但对该领域一无所知的人发展其事业，并在国际上获得好评，通过同样的方式，许多大学生已成为行业领导者。有人可能会争辩说，模式研究所吸引的是那些注定要成为领导者的人，但我宁愿认为，我们在工作中为获得大量知识所养成的良好习惯使我们成为明智和令人尊敬的领导者。

回顾挑战

我们又回到回顾展，这是揭示模式的一种好方法。我把回顾展定义为“将社区集合到一起，共同讲述过去曾发生的事件，以寻求比各人可能已注意到的更丰富的知识。”因此，我们在这里用同一个声音讲话。可能社区最开始的讨论就是如何实现这一点，以及下一步向何处发展。

结论

模式的作者做了重要的工作。我们提炼和记录整理最好的成果，目的是使这一领域发展到一个新的专业水平。虽然我们是每个人单独工作，但会得到同行的帮助和支持，他们是我们的指导者和研讨会的参与者。

我们的工作非常重要；它持续地对软件领域产生深刻影响。在 2005 年，本系列丛书（共出版 4 卷）中的 3 本入围第 15 届 Annual Jolt 图书奖（该奖是关于模式的）的“决赛”。最具声望奖则颁给了《Head First Design Patterns》一书，竞争该奖项的图书还有《Refactoring to Patterns》和《Software Factories：Assembling Applications with Patterns，Models，Frameworks and Tools》。与此对照，回首 10 年间其他新兴思想形成的时间——形式方法论，用例和 UML；权能成熟模型，ISO 标准及其不断的完善；软件生命周期的改进和快速应用程序开发；重用和业务流程再造；进程制和函数指针——有多少相互竞争的想法仍然影响到我们的工作？

积极的读者手中会有一些来自模式讨论社区的成品。本卷增加了社区提供的最佳成果。在过去的几年中，这些文献已经反复编写、推敲、修改和完善。你现在看到的不只是作者的辛勤劳动，其中还包含了指导者的意见、会议参与者的讨论完善，以及令人尊敬的编辑人员的精心挑选，所有这些共同形成了本卷的最终成稿。

但是，如果你不能将本书中的这些知识融入到专业实践中，那么所有这些工作都是徒劳的。我希望你通过本书能找到改善工作的好方法。

Norman L. Kerth
于俄勒冈州波特兰市
2005年7月

参考文献

J. O. Coplien. "The Culture of Patterns." *ComSIS Consortium*, Volume 01, Issue 02 (November 2004).

W. Cunningham. *EPISODES: A Pattern Language of Competitive Development (PLoPD2)*, pp. 371-390. Reading, MA: Addison-Wesley, 1999.

Eliz. Freeman, E. Freeman, B. Bates, and K. Sierra, *Head First Design Patterns*. O'Reilly, 2004.

E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns*. Reading, MA:, Addison-Wesley, 1995.

J. Greenfield, K. Short, S. Cook, S. Kent, and J. Crupi. *Software Factories: Assembling Applications with Patterns, Models, Frameworks and Tools*. Wiley, 2004.

J. Kerievsky. *Refactoring to Patterns*. Boston: Addison-Wesley Professional, 2004.

L. Rising. *The Pattern Almanac 2000*. Boston: Addison-Wesley, 2000.

此为试读，需要完整PDF请访问：www.ertongbook.com