软 件 工 程 技 术 丛 书

HZ BOOKS

设计系列

美 国
“软件开发”杂志
效率大奖获奖作品

SOFTWARE Development
PRODUCTIVITY AWARD 1996

面向模式的软件体系结构
卷1：模式系统

Pattern-Oriented Software Architecture
Volume 1: A System of Patterns

Frank Buschmann　Regine Meunier
（德）Hans Rohnert　Peter Sommerlad　著　贲可荣　郭福亮　赵皑　等译
Michael Stal

WILEY

机械工业出版社
China Machine Press

软 件 工 程 技 术 丛 书

设计系列

Frank Buschmann　Regine Meunier
(德)　Hans Rohnert　Peter Sommerlad　著
Michael Stal

贲可荣　郭福亮　赵皑　等译

面向模式的软件体系结构
卷1：模式系统

Pattern-Oriented Software Architecture
Volume 1: A System of Patterns

机械工业出版社
China Machine Press

本书系统地介绍了模式的概念并讨论了模式描述的原则；给出了来自不同应用领域的8个软件体系结构模式；还给出了8个设计模式，这些模式分别针对在定义了软件系统的整体结构之后遇到的典型问题；对惯用法模式进行了阐述；论述了将模式组织成模式系统的重要性；讨论了模式如何嵌入到软件体系结构中，特别是对软件体系结构及其基本原则的理解。此外，本书还介绍了模式历史、相关工作和整个模式团体，并对模式的未来发展作了展望。

本书力图做到既是教材又是参考指南，帮助软件开发人员以一种新的方式考虑软件体系结构，并提供了一些技术来解决特定的再现设计问题，对软件开发的初学者和专家都有帮助。把本书用作软件工程课程的指南，可以给学生提供大型软件设计的完整的新观点。把本书用作参考手册，可以提供全面的技术并随查随用。本书包含了涉及模式实际应用的许多指导原则和约束限制。本书可作为计算机专业高年级本科生、研究生的软件工程教材或参考书，亦可作为软件开发人员的参考手册。

Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad and Michael Stal: Pattern-Oriented Software Architecture, Volume 1: A System of Patterns (ISBN: 0-471-95869-7).

Authorized translation from the English language edition published by John Wiley & Sons, Inc.

本书版权登记号：图字：01-2002-0816

图书在版编目（CIP）数据

面向模式的软件体系结构 卷1：模式系统 / 布希曼（Buschmann, F.）等著；贲可荣等译. -北京：机械工业出版社，2003.1

(软件工程技术丛书-设计系列)

书名原文：Pattern-Oriented Software Architecture, Volume 1: A System of Patterns

ISBN 7-111-11182-6

I. 面 … II. ① 布… ② 贲… III. 数据结构 IV. TP311.12

中国版本图书馆CIP 数据核字（2002）第088419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：姚 蕾

北京第二外国语学院印刷厂印刷 · 新华书店北京发行所发行

2003年1月第1版第1次印刷

787mm × 1092mm 1/16 · 19印张

印数：0 001-5 000册

定价：45.00元

译者序

软件体系结构的模式描述了在特定设计语境中出现的设计问题的特殊重现，并为这种方案提供了一个被证明良好的一般计划。方案计划通过描述它的关键组件、它们的责任和相互关系以及它们结合的方式来指定。

模式有助于利用资深软件工程师的经验来构建软件。模式总结在软件开发中现存的、被证明良好的经验，再用来促进好的设计实践。每个模式处理一个软件系统的设计或实现中一种特定类型的重复出现的问题。模式可以用来构建具有特定属性的软件体系结构。

保存模式文档已被证明是很好的设计经验，它们提炼并提供一种方式来重用从实践经验中获得的设计知识。模式为设计原则提供一种公共的词汇和理解。它们提高了对设计问题及其解决方案的讨论效率。

模式是软件体系结构文档化的一种方式。当设计一个软件系统时，它们可以进行形象描述。模式支持采用定义属性来构造软件。模式提供一个功能行为的基本框架，有助于实现应用程序的功能。模式还清楚描述了软件系统的非功能需求，如可修改性、可靠性、可测试性和可重用性。模式有助于建立一个复杂的不同种类的软件体系结构。每个模式提供组件的预定义集、它们的作用以及它们之间的关系。

一个模式提供了一个问题族的一般解决方案的计划，而不是可以使用的预制模块。模式有助于相似单元的创建。模式有助于解决问题，但它不能提供彻底的方案。

模式有助于管理软件复杂度。每个模式描述一种已被证明的方法以处理指出的问题：所需组件的种类、它们的作用、要隐藏的细节、必须看到的抽象以及每一个是如何工作的，等等。

本书对软件开发的初学者和专家都有帮助。本书可以帮助初学者在适当规模的项目上像专家一样工作，而不需要积累多年的经验。本书可以支持专家去设计具备已定义属性的大型复杂软件，并可以促使他们学习别的专家的经验。

本书力图做到既是教材又是参考指南，帮助软件开发人员以一种新的方式考虑软件体系结构，并提供一些技术来解决特定的再现设计问题。把本书作为软件工程课程的指南，可以给学生提供大型软件设计的完整的新观点。把本书作为参考手册，可以提供全面的技术并随查随用。本书包含了涉及模式实际应用的许多指导原则和约束限制。

本书可作为计算机专业高年级本科生、研究生的软件工程教材或参考书，亦可作为软件开发人员的参考手册。

本书第3、4章由郭福亮翻译，第5~8章由赵皑翻译，其余章节由贲可荣翻译。全书由贲可

荣审校。俞立军、孙宁、王淑雪、吴铁洲、周娟、石国景等参与了本书的翻译工作，在此表示感谢。

由于各种原因，译稿难免存在错误和疏漏。欢迎读者批评指正。

译 者

2002年8月8日

译者介绍

贲可荣　男，1963年8月生，江苏省海安县人。现任海军工程大学教授。

1983年在苏州大学数学系获理学学士学位，1986年在南京大学数学系获理学硕士学位。1986年8月至1990年3月在海军工程大学计算机系任教。1994年6月在国防科技大学获工学博士学位。博士期间，由国防科技大学计算机学院陈火旺院士指导，主修计算机软件。1994年12月任海军工程大学计算机系副教授。1995年起担任计算机应用技术专业硕士生导师、海军工程大学学位评定委员会委员。经教育部批准，2000年3月至2001年3月任武汉大学软件工程国家重点实验室访问学者；2000年被教育部确定为海军首批十名骨干教师之一。2000年12月晋升为教授。

主要译著有《净室软件工程——技术与过程》(电子工业出版社，2001年6月)，《能力成熟度模型(CMM)：软件过程改进指南》(电子工业出版社，2001年7月)，《基于项目的软件工程——面向对象研究方法》(机械工业出版社，2002年6月)，审校《实用软件测试指南》(电子工业出版社，2003年1月)，参加《计算机科学技术百科全书》(清华大学出版社，1998)的编写。先后承担国家自然科学基金项目、国家863项目和军队科研等五个项目。在“中国科学”、“软件学报”、“计算机科学”等刊物和会议发表论文50余篇。成果获军队科技进步奖。主要研究方向：软件可靠性、软件质量保证技术、形式化方法等。兼任中国计算机学会计算机理论专业委员会委员、中国造船工程学会电子技术学术委员会委员，《海军工程大学学报》、《舰船电子工程》等刊物编委。

郭福亮　男，1963年1月生，河北省滦南县人。现任海军工程大学教授。

1982年在海军工程学院数学专业获理学学士学位，1987年在北京航空学院计算机系软件工程研究生班毕业。1987年8月至1996年8月在海军工程学院计算机系任教。2001年1月在华中科技大学计算机学院获硕士学位。2001年9月起在华中科技大学攻读博士学位。由华中科技大学计算机学院卢炎生教授指导，主修计算机软件与理论。1996年12月任海军工程学院华中地区军队院校计算机技术服务中心副教授。1998年起担任计算机应用技术专业硕士生导师。2001年12月晋升为教授。

曾主编过《微型计算机维护维修实用技术》(华中理工大学出版社，1997年11月)，主要译著有《自己动手装网络Linux》(华中理工大学出版社，1998年12月)。在“中国科技文库”、“计算机应用研究”、“图像工程”、“军事”等刊物和会议发表论文20余篇。先后承担多项军队科研项目，其中两项成果获军队科技进步奖。主要研究方向：软件可靠性，数据挖掘技术等。

赵皑　女，1973年1月生，河北张家口人。现任海军工程大学讲师。

1994年在国防科技大学计算机系获工学学士学位，1997年在国防科技大学计算机系获工学硕士学位。1997年3月至今在海军工程大学任教。主要研究方向：计算机网络及应用、多媒体技术。

主要译著有《JAVA 2 类库》（机械工业出版社，2001年1月）。

前　言

本书讨论了软件体系结构的模式，简称模式。近年来，模式已引起广泛关注，讨论模式的形式包括专题讨论会、研究班、电子论坛、期刊论文、专著等。模式团体甚至开始举行它们自己的会议。所有这些有关模式的热情讨论看起来像是达到了“对象波”的顶峰。

为什么关于模式如此令人激动呢？很可能是因为它们在收集熟练的设计者和软件工程师的经验方面起到了“根基”的作用。这样的专家对许多再现设计问题已有了解决方案。模式以易于获得的方式和所希望的良好书写的格式捕获了这些已证实的解决方案。

我们希望本书对软件开发的初学者和专家都有帮助。本书可以帮助初学者在适当规模的项目上像专家一样工作，而不需要积累多年的经验。本书可以支持专家去设计具备已定义属性的大型复杂软件，并可以促使他们学习别的专家的经验。本书会帮助这两种人找到对一个特定设计问题的充分考验的解决方案。

本书力图做到既是教材又是参考指南，帮助软件开发人员以一种新的方式考虑软件体系结构，并提供一些技术来解决特定的再现设计问题。把本书用作软件工程课程的指南，可以给学生提供大型软件设计的完整的新观点。把本书用作参考手册，可以提供全面的技术并随查随用。本书包含了涉及模式实际应用的许多指导原则和约束限制。

以规范形式记录设计知识可以追溯到Christopher Alexander[⊖]。他在建筑学领域率先提出模式概念。他的著作《*The Timeless Way of Building*》说明了如何将模式应用到房屋构建以及邻近地区和整个城市的规划之中。他的工作的基础主题是居住场所的设计，不仅要注重功能性和流行趋势，而且要讲究舒适感和欣慰感。良好设计的建筑可以展现清晰可辨而又难以描述和量化的内在特质。简而言之，这样的建筑拥有“没有名称的质量”。

把这种方法改用到软件工程中的早期试验主要偏重于Alexander的风格。最近软件界经历了寻求更适合软件设计的风格形态的试验。对模式尝试了几种不同的描述形式，但至今没有达成一致。

尽管我们把相当多的精力用于寻找一种描述模式的好方法，但是开发“模式风格”理论并不是本书的主要目标。这当然不是开始模式工作的原动力。在1991年，我们以一种直截了当的方式记录了第一个模式。随着模式系统文档的逐步增长，很快就发现单个模式不能单独存在。模式揭示了丰富的相互关系集。这是出版本书的驱动因素之一，而不是在某个时候把模式文档化并作为系列论文出版。采用图书这种方法的缺点在于，在成书之前需要一个较长的酝酿思考阶段。尽管模式成名已经几十年了，但是花费如此之长的时间来提出一个好的模式描述仍然让我们吃惊。

⊖ Christopher Alexander是一位开业建筑师和城市规划师，同时他也是美国加州大学伯克利分校的建筑学教授以及环境建筑中心的主任。他创建了建筑学的理论，基于模式的构造和使用来进行构建和规划。他的模式理论本身、他的理论实践以及对这种途径的批评和讨论已由牛津大学出版社以系列丛书的形式出版。

其他四位作者经历了同样的现象。1994年秋天，Erich Gamma、Richard Helm，Ralph Johnson和John Vlissides[⊖]出版了重要著作《设计模式——可复用面向对象软件的基础》。尽管设计模式的思想不再新颖，但是“GoF”（简称为“四人帮”）的著作仍然给出了对良好描述的面向对象程序设计模式的最初分类。

我们的方法和“四人帮”的方法略有差别，尽管有许多相似之处并有部分重叠。“GoF”的著作集中于设计层模式，而我们的模式跨越几个抽象层次。范围从高层体系结构模式经设计模式到低层惯用法。我们也关注面向对象以外的问题，并试图结合对模式描述技术的最新认识。我们的整体目标是使用模式来在软件体系结构的更广泛的语境中提供帮助。我们把这种方式叫做面向模式的软件体系结构。我们谈到了模式系统，模式不仅仅被收集在不同种类的容器中，而且也根据相应的标准分组。“GoF”的著作通过把模式划分为“创建型”、“结构型”和“行为型”等开始了分类工作。我们下一步试图根据已获得的更好标准（如交互式和适应性系统、工作的组织、通信和访问控制等）来对模式分组。

我们想要鼓励我们模式系统的用户与其同事共享我们的研究成果。模式共享建立了设计问题的通用词汇表。它使得正在成长的模式团体的成员以更有效的方式来确认、命名和讨论问题及其解决方案。在系统设计中达到“提速”是在工作中使用模式的一个更加重要的理由。

我们的模式系统并不完全。如此多的模式用一本书是不可能记录下来的。随着技术的演化，新的模式也跟着演化。我们希望读者能针对你们的特定需要对我们的模式系统进行扩展、修改和剪裁。缺少的模式应该添加，不需要的模式可以忽略，其他的模式可以更改。

如果你对本书风格和内容方面的改进有任何评论、批评或建议，请大胆提出。我们也欢迎用我们描述的模式来报告经验。你可以写信并通过John Wiley & Sons.出版公司转交给我们，或发电子邮件到patterns@mchp.siemens.de。

我们对于本书包括的绝大多数模式的最初版本已经在因特网上进行了讨论。我们的动机不是为了获得免费广告或赠送模式。相反，我们想支持出版业的一个新趋势，即在刊印之前尽早将材料展示给团体成员，这样对每一个部门都有益。我们欣赏这些经验并感谢所有的参与者。但是这并不意味着我们关闭了本书的公开电子讨论。邮件列表仍然存在并仍然欢迎读者参与。预订指南可以在模式主页上找到。它的URL是：

http://www.hillside.net/patterns/

这个URL也是所有模式方面（比如已出版的和待出版的著作、模式会议、模式论文等）最重要的信息资源。

本书结构

第1章系统地介绍了模式的概念并讨论了模式描述的原则。第2章~第4章给出了我们的模式目录的内容。

⊖ 他们四人常被称为“四人帮”，或简称为“GoF”（Gang-of-Four）。——编辑注

体系结构模式是最高层模式。它们旨在提供整个系统体系结构的构架。第2章给出了来自不同应用领域的8个体系结构模式。

第3章给出了8个设计模式，这些模式分别针对在定义软件系统的整体结构之后遇到的典型问题。例如，我们的设计模式处理以下情况：构建控制复杂性的组件，分布组件间工作量，组织组件间通信。

第4章是模式内容的第三部分也是最后一部分。它处理惯用法，一种与语言相关的模式。但是本书主要谈到其他人的工作而不是对我们自己的惯用法进行文档化，并且仅仅提供了一个惯用法作为一个具体实例。本书没有对我们自己的惯用法进行描述，理由很简单——已经有了很多诸如C++和Smalltalk语言的惯用法可供使用。我们不是仅对这些模式进行改述，我们要有选择地引用原始资源。

在第5章中，我们证实了把模式组织成模式系统是很重要的。这样的系统在如下几个方面对模式的作者和使用者都有帮助：为手头的情形寻找正确的模式，填充模式汇集中的缺口，理解模式之间的关系并演化模式系统。

在第6章中，我们讨论了如何将模式嵌入到软件体系结构中。特别是我们讨论了对于软件体系结构及其基本原则的理解，我们展示了模式是如何支持这些原则的。

第7章讨论模式历史、相关工作和整个模式团体。结束本书前，第8章给出了对于模式未来发展的展望。

本书以附录结尾，附录包括符号表、常用词汇表、参考文献和模式索引。

致谢

我们向所有以不同方式对本书的形成起到过帮助作用的人士表示感谢，不仅因为这是一个惯例，而且更是因为我们真心诚意希望这样做。

我们感谢Joelle Coutaz、Wilhelm Gruber、Claus Jäkel、Doug Lea、Oscar Nierstrasz、Laurence Nigay、Frances Paulisch、Wolfgang Pree、Uwe Steinmüller、John Vlissides和Walter Zimmer，他们对我们工作的早期版本进行了讨论和修订。Ralph Johnson和他在Urbana-Champaign伊利诺斯大学的体系结构读书小组的成员，其中包括John Brant、Michael Chung、Brian Foote、Don Roberts和Joseph Yoder，他们仔细评审了我们的绝大部分模式描述。他们为我们提供了许多评论以及改进建议。我们也感谢山腰组（Hillside Group）的支持和鼓励。

我们在每个模式描述最后以单独一节感谢对这个模式的改进提供过帮助的人们。

特别感谢James Coplien、Joseph Davison、Neil Harrison和Douglas Schmidt。他们详细评审了我们的材料，帮助我们设计和润色本书的最终内容。

我们的夏季学生Marina Seidl和Martin Botzler和我们一起经历了一些早期试验。也特别感谢Franz Kapsner和Hartmut Raffler在管理上的支持，感谢德国慕尼黑西门子公司合作研究和开发部软件工程实验室的支持。

Francis Glassborow和Steve Rickaby力求提高我们有限的英语写作能力，并帮助消除了最差的“德语习语”——这不是一项容易的任务。

最后，我们感谢编辑Gaynor Redvers-Mutton以及John Wiley & Sons出版公司内所有使本书按时出版的人士。

读者指南

本书的编排使读者能从头至尾地阅读。万一你想在本书中选择你自己的阅读路线，可以参考下面给出的提示。

第1章“模式”，给出了软件体系结构的全面解释。一切都建立在这些讨论之上，所以你应该首先阅读这一章。阅读单个模式的顺序取决于你。想掌握某个特定模式的关键思想，你只要阅读它的语境、问题和解决方案等小节。广泛的交叉引用会指导你理解模式间的相互关系。

如果模式对你而言是全新的知识，我们建议你首先阅读基本和简单的模式，即那些易于理解并在许多良好结构的软件系统中出现的模式。例子包括：

- 管道和过滤器体系结构模式
- 代理设计模式
- 转发器-接收器设计模式

你也可以利用本书寻找你当前工程中遇到的设计问题的解决方案。利用第5章“模式系统”中的模式综述作为你研究的指南，再查看那些你选为潜在解决方案的模式的详细描述。

其他章节——第6章“模式和软件体系结构”、第7章“模式团体”和第8章“模式将走向何方”——可以以任何顺序阅读，尽管书中给定的顺序是我们认为最适合读者阅读的。

索引中的页码为英文原书面码，与书中边栏的页码一致。

目　录

第 1 章
模　式

……很久以前，飘荡在广漠无际的太空中的一组随意的原子受到严重伤害，并以特别不同的模式将它们粘附在一起。这些模式迅速学会复制自身（这是模式如此不凡的部分原因）且继续对它们飘过的每个行星造成巨大的麻烦。这是生命在宇宙中的开始……

Douglas Adams，《*The Hitchhiker's Guide to the Galaxy*》

模式有助于我们利用熟练的软件工程师的集体经验来构建软件。它们捕捉软件开发中现存的、充分考验的经验，再用于促进好的设计实践。每个模式处理一个软件系统的设计或实现中一种特殊的重复出现的问题。模式可以用来构建具有特定属性的软件体系结构。

本章对软件体系结构的模式是什么，以及它们如何帮我们构建软件等问题给出了一个全面深入的说明。

1.1　什么是模式

当专家求解一个特殊问题时，他们一般不会发明一种和已有解决方案完全不同的方案来处理这个问题。他们往往想起已解决过的相似问题，并重用其解法的精华来解决新问题。这种“专家行为”，即同时考虑一对问题-求解方案，这一点在很多不同领域中都是共同的，比如建筑学[Ale79]、经济学[Etz64]和软件工程[BJ94]领域就是如此。这是解决任何一类问题或社会交互的一种自然手段[NS72]。

下面是取自建筑学领域的一对问题-解决方案的一个很好的直观例子：

例子　窗户位置[AIS77]

每个人都喜爱靠窗的座位、凸窗、有着低窗台的大窗户、舒适的椅子放在它们边上……。一个房间如果没有一个这样的位置，你自然不会觉得舒适或完美……。

如果房间没有“位置”适宜的窗户，房里面的人会受到以下两种强制条件的折磨：

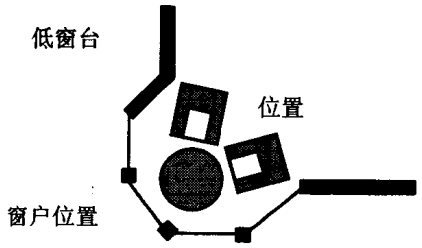
(1) 他想舒适地坐下来。

(2) 他想朝向光线。

很明显，如果让你感到舒适的位置——房间中你最想坐的位置——距离窗户较远，你会无法克服这种冲突。

因此，如果你每天在那个房间都要呆上一段时间，那么至少让这个房间的一个窗户处于一个适宜的“窗户位置”。

从特定的一对问题-解决方案中进行抽象并提炼出公共要素可以形成模式：“这些成对的问题-解决方案倾向于形成相似的问题-解决方案的系列，且每一个系列体现一种在问题和解决方



案中的模式”[Joh94]。建筑师Christopher Alexander在《建筑的永恒方法》(The Timeless Way of Building)[Ale79]一书中，定义了如下的术语“模式”(*pattem*)：

> 每个模式是一条由三部分组成的规则，它表示了一个特定环境、一个问题和一个解决方案之间的关系。
>
> 作为建筑领域的一个元素，每个模式是一个特定环境，在该环境中重复出现的一个特定强制条件体系(system of forces)以及给这些强制条件自求解提供一个特定的空间配置之间的相互关系。
>
> 作为语言的一个元素，一个模式是一个说明，它说明了如何使用这个空间配置，不断求解给定的强制条件体系，只要环境是与它相关的。
>
> 简单地说，模式是在同一时间里发生在世界上的一件事物和如何创建这个事物以及我们何时必须创建它的规则。它既是一个过程，又是一个事物；既是一个活生生事物的描述，又是产生那个事物的过程描述。

我们在软件体系结构中也发现了许多模式。软件工程中的专家从实践经验中获知这些模式并在开发具有特定属性的应用中遵循这些模式。专家们使用模式有效而出色地解决设计问题。在详细讨论这些内容之前，让我们看一个著名的例子：

例子　模型-视图-控制器(Model-View-Controller, MVC)(参见2.4.1节)

在开发人机界面软件时考虑这种模式。

用户界面容易改变需求。例如，当扩展一个应用程序的功能时，必须修改菜单以访问新功能，而且用户界面必须适合特定的客户。一个系统可能经常以不同的“视觉和感觉”标准运行在另一个平台上。甚至升级一个新的窗口系统版本就意味着需要更改你的代码。总之，经久耐用的系统的用户界面是一个我们不断变动的目标。

如果用户界面和功能核心紧紧交织在一起，则建立一个具有所要求灵活性的系统是昂贵的而且易于出错。这导致开发和维护几个实质上不同的软件系统，一个系统针对一个用户界面的实现。确保改动随后散布到许多模块中。总之，当开发这样一个交互软件系统时，你必须要考虑两方面的因素：

- 用户界面应该是易于改变的，在运行期间也是可能改变的。
- 用户界面的修改或移植应该不影响应用程序的功能核心的代码。

为解决这个问题，把一个交互应用程序划分成三个部分：处理、输出和输入：

- 模型(*model*)组件封装核心数据和功能。模型独立于特定的输出表示或输入行为。
- 视图(*View*)组件给用户显示信息。视图获得来自模型的待展示数据。一个模型可以有多

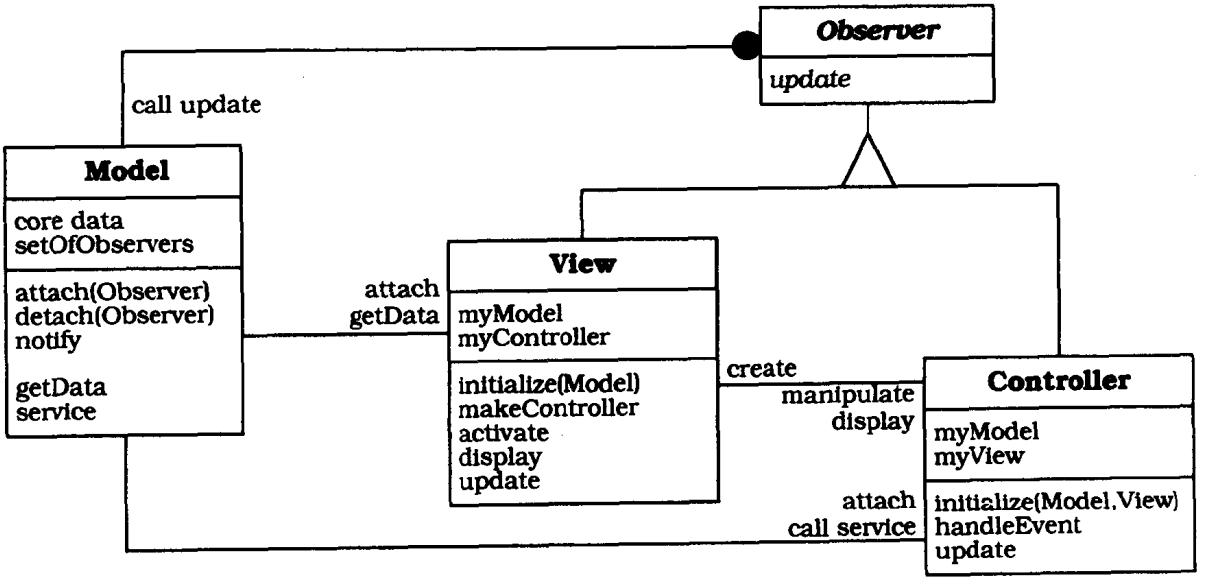
个视图。

- 每个视图都有一个相关的控制器（*cantroller*）组件。控制器接受输入，通常就是像鼠标移动、鼠标点击或键盘的输入等事件。事件被转换为服务请求，而服务请求又被传送给模型或视图。用户仅仅通过控制器与系统交互。

将模型与视图及控制器组件分离允许同一模型有多个视图。如果用户通过一个视图的控制器改动了模型，则所有其他依赖于这个数据的视图应该反映这个改动。为达到这一目的，数据一旦改动，模型就通知所有视图。视图依次从模型获得新数据并更新显示信息。

这种解决方案可以改动应用程序的一个子系统而不对其他子系统产生重要影响。例如，把一个非图形化用户界面改为图形化用户界面不需要修改模型子系统。你可以添加一个对新的输入设备的支持而不需要影响信息显示或功能核心。该软件的所有版本可以处理同一模型独立于特定“视觉和感觉”的子系统。 4

下面的OMT（对象模型技术）类图[⊖]描述了这种解决方案：



❑

从下面这个介绍性的例子中，我们可以导出软件体系结构模式的几个属性[⊖]：

一个模式关注一个在特定设计环境中出现的重现设计问题，并为它提供一个解决方案。在我们的例子中，问题是支持用户界面的可变性。开发人机交互软件系统时，这个问题就会出现。你可以通过严格区分职责来解决这个问题：将应用程序的核心功能从其用户界面中分离出来。

各种模式用文档记录下现存的经过充分考验的设计经验。它们不是人工发明或创造的。它们“提炼并提供一种手段来重用从有经验的实践者获得的设计知识”[GHJV93]。那些熟悉足够 5

⊖ 对于分析和设计方法对象建模技术（OMT）及其表示法的小结，参见本书第8章之后的“表示法”。详细描述可见[RBPEL91]。

⊖ 如果不另外声明，我们把术语“模式”（*pattern*）和“软件体系结构模式”（*pattern for software architecture*）作为同义词使用。