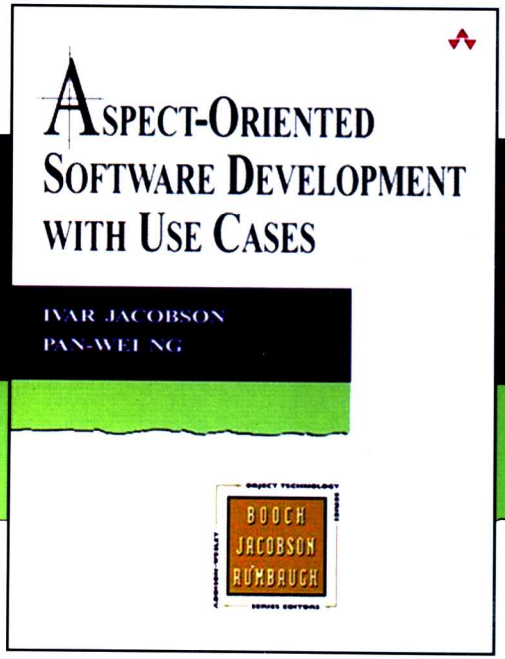
Broadview
WWW.BROADVIEW.COM.CN

AOSD中文版

——基于用例的面向方面软件开发



Ivar Jacobson, Pan-Wei Ng 著
徐锋 译
Ivar Jacobson Software China 审校

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

AOSD 中文版

——基于用例的面向方面软件开发——

Aspect-Oriented Software Development with Use Cases

[美] Ivar Jacobson 著
Pan-Wei Ng
徐 锋 译
Ivar Jacobson Software China 审校

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

内 容 简 介

本书系统阐述了面向方面软件开发（AOSD）方法，AOSD 的目标是通过使系统的功能需求、非功能需求、平台特性等诸多不同的关注点相互独立，实现更好 的模块化，来构建出易于理解、易于扩展、高复用性、高质量的软件系统。AOSD 将用例技术和面向方面技术有机结合在一起，为软件开发提供了一个切实可行的最佳实践集。本书还系统阐述了用例技术、AOP（面向方面编程）技术的特点和使用方法，以及实现用例与 AOP 结合使用的用例模块、用例切片等技术。并且通过一个现实世界中常见的“酒店管理系统”来展示了如何在项目实践中高效地应用 AOSD 方法。作者见解独到、精辟，不仅阐述了理论知识，还详尽说明了如何在项目的不同阶段中应用 AOSD 技术。这本书对于项目经理、系统分析员、系统设计师及广大开发人员，都具有很高的实用价值。

版权贸易合同登记号：图字：01-2005-2189

图书在版编目（CIP）数据

AOSD 中文版：基于用例的面向方面软件开发 /（美）雅各布森（Jacobson,I.），（美）黄邦伟著；徐锋译.
北京：电子工业出版社，2005.10
书名原文：Aspect-Oriented Software Development with Use Cases
ISBN 7-121-01831-4

Ⅰ.A… Ⅱ.①雅…②黄…③徐… Ⅲ.软件开发 Ⅳ.TP311.52

中国版本图书馆 CIP 数据核字（2005）第 116812 号

责任编辑：周 筠 熊妍妍
印 刷：北京智力达印刷有限公司
出版发行：电子工业出版社
北京市海淀区万寿路 173 信箱 邮编 100036
经 销：各地新华书店
开 本：787×980 1/16 印张：28.75 字数：500 千字
印 次：2005 年 10 月第 1 次印刷
定 价：49.00 元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系。联系电话：（010）68279077。质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

《AOSD 中文版——基于用例的面向方面软件开发》

专家委员会

伊万·雅各布森（Ivar Jacobson）**博士**：被公认是深刻影响或改变了整个软件工业开发模式的几位世界级大师之一，是软件方法论的一面“旗帜”。是组件和组件架构、用例、现代业务工程以及 Rational 统一过程等业界主流方法/技术的创始人，UML 建模语言创始人之一。

黄邦伟（Pan-Wei Ng）**博士**：雅各布森咨询公司（Ivar Jacobson Consulting）创始人之一。曾经是瑞理（Rational）公司全球仅有的数名培训大师（Training Master）之一。

马杰明（James J. Majure）：雅各布森咨询公司（Ivar Jacobson Consulting）资深咨询顾问，参与并领导美国银行(US Bank) RUP 导入项目。曾在美国富国银行（Wells Fargo）等公司的多个大型项目中担任系统架构师。

蒋胜：雅各布森软件（北京）有限公司执行董事、总经理。曾在美国 Microsoft 公司和中兴通讯公司工作多年。

冯晨华：雅各布森软件（北京）有限公司高级咨询顾问。致力于现代软件工程和软件过程的研究，已为数十家软件企业成功提供了软件工程和过程改进的咨询和培训服务。

徐锋：中国系统分析员顾问团华东首席顾问，《程序员》专栏作者。致力于系统分析与设计、需求工程、软件过程改进等领域的研究。

潘加宇：UMLChina 首席专家，潜心研究和实践 UML/UP 相关技术的应用。已上门为超过 80 家软件组织提供技术指导和训练服务。

以上专家共同保证该书的质量。

对《基于用例的面向方面软件开发》一书的赞许

“这是一种通过面向方面技术来改进用例建模的，令人神清气爽的新方法。”

——Ramnivas Ladddad

《AspectJ in Action》作者

“自 1980 年以来，用例技术已经成为将用户带入软件设计的一种主要方法，但将用例转换成为软件还是一门艺术，因为用户通常对代码边界并不关注。而现在面向方面编程（AOP）可以直接在代码中直接表示横切关注点，这也就是意味着用例的开发人员可以在用例中一步步地组织横切关注点，然后将其在不同的模块中编程实现。如果这些方法应用于你的设计和开发实践中，将会使得软件质量有巨大的提升，不管是针对开发者还是用户。

——Wes Isberg

AspectJ 开发团队成员

“本书不仅提供了面向方面软件开发的观点和例子，还告诉你如何将其应用于实际的软件开发项目中。”

——Michael Ward

ThoughtWorks 公司

“没有一个系统是从理想的地平线开始设计的；每个系统都是在日积月累中，以功能之上叠加功能的形式组成。传统的设计技术无法良好地解决这个问题，最终只能是以完整性降低作为结果。现在，我们第一次拥有一组有助于行为合成的技术，它使得系统不是按照程层次性的功能模块来定义，而是通过合成（composition）的方法。本书是现代方法的一个重要进步，它必将对软件工程下一个十年的发展方向产生深远影响，就像先前“面向对象软件工程”的影响一样。”

——Kurt Bittner

IBM 公司

“用例是捕获系统需求、并以用户为中心的视角来驱动系统开发及测试过程的优秀方法。本书提供了一个清晰阐述用例驱动开发的详尽指令，告诉你如何从最初的需求建模开始，设计及实现系统。它对使用面向方面设计及编程技术实现用例模型，提供了一组简单但很丰富的方针集。这些对于研究者和实践者都是珍贵的资源。”

——Dr Awais Rashid

英国 Lancaster 大学教授，《面向方面的数据库系统》的作者

“AOSD 是一种有助于程序员开发出更好系统的重要技术。遗憾的是，AOSD 并未清晰地指出如何集成到项目生命周期中去。本书打破了这一障碍，提供一个用 AOSD 的具体例子，覆盖了从需求分析到测试的全过程。”

——Charles B.Haley

英国 Open 大学研究院成员

中文版序

我们总在说软件系统变得越来越复杂，如今这几乎已经是陈词滥调，但却仍然值得重申。为了解决复杂性，我们需要有更好的方法来将系统分解成一个个小的部分，并每次解决其中的一个部分。这正是模块性技术所涵盖的——每个模块执行一组定义清晰、范围适当的职责集合。多年以来，软件开发社群已经试验了各种不同的模块性技术，并形成了对象和组件技术。面向方面技术是一种全新的模块性构造技术，旨在克服组件和对象技术的局限，因此也可能使系统拥有更好的可维护性和可扩展性。面向方面技术背后的思想同用例有很多相似之处，这意味着如果你已经完全理解用例驱动开发，就可以将面向方面应用于项目当中。

虽然本书的主题是面向方面，但我们在这里所提倡的概念并不局限于采用面向方面编程实现的软件系统。诸如最小化设计、基础结构用例等概念，以及我们在这里所介绍的其他许多技术，也都是普遍适用的。事实上，我们已经成功地帮助许多客户采用“面向方面的思维方式”将用例驱动方法运用于他们的系统当中——即便是面向对象系统。这些系统包括了从基于 Web 的业务交易系统到实时电信系统等等。此外，我们还发现，许多读者通过阅读我们的书，对于关注点分离和模块性等概念有了全新的理解，而这些正是实现优秀软件系统的关键因素。

为广大软件开发人员培育良好的开发实践是我们的一贯使命和目标，撰写本书仅仅是其中的一个“方面”。中国拥有强大的软件产业，并将会更加强大。中国的软件开发人员总是渴望学习和尝试新的概念。欢迎你们学习和尝试，并与我们一起共同成长。

但我们能够并且希望做得更多。为此我们在中国设立了分公司，为专业开发人员提供培训，为项目团队提供辅导，并全方位地改善软件开发组织的能力。我们诚挚邀请您参加我们的面向方面以及其他的软件开发最佳实践的课程。

Ivar Jacobson
Pan-wei Ng
2005 年 10 月

中文版序

We have always said that software systems are becoming increasingly more complex. This is almost a cliché now, but it is still worth repeating. To overcome the complexity, we need better ways to break the system into smaller parts and solving them one at a time. This is what modularity is all about – each module performing a well defined and appropriately scoped set of responsibilities. Over the years, the software development community has experimented with different kinds of modularity giving rise to objects and components. Aspects are a new kind of modularity construct to overcome the limitations of components and objects and thereby provide opportunities for better maintainability and extensibility. The idea behind aspects bears much resemblance to the idea of use cases, which means that if you understood what use case driven development is all about, you are ready to apply aspects in your project.

Although the subject is on aspects, the concepts which we advocate here are not limited to systems implemented using aspect oriented programming. Concepts such as minimal design, infrastructure use cases, etc. and many more which we describe here are general applicable. In fact, we have successfully helped a number of our clients to apply "aspect thinking" using the use case driven approach in their systems – even object oriented ones. These systems ranges from web enabled business transaction systems to real time telecommunication systems. Moreover, what we also discovered is that those who read our book have a renewed understanding of modularity, which is a key factor to good software systems.

It has been our mission and goal to nurture good software development practices amongst software professionals. Writing this book is just one "aspect" of it. China is and will become an even stronger software development industry. Chinese professionals are eager to learn and try new concepts. You are welcome to do so and to grow together with us.

But we can and want to do more. We have established a company in China to train software professionals, to mentor project teams and to improve the overall capability of software development organizations. We invite you to our courses on aspect orientation and other development best practices.

Ivar Jacobson

Pan-wei Ng

October, 2005

译者序

2000 年以前，笔者在软件需求实践中，一直对如何有效地标识、组织、管理用户的需求感到十分的困惑。纷繁复杂而且拖沓冗长的“软件需求规格说明书”似乎一直没有起到它应该起的作用，如何才能够使得用户和开发团队之间建立更好的需求沟通呢？Ivar Jacobson 先生提出的“用例驱动方法”让我找到了解决的方法。它使我们站在“用户的视角”来观察“将要开发的系统”，通过对零散的软件需求进行合并，抽象出参与系统的不同参与者（Actor），将一系列的使用场景进行抽象形成“用例”，从而清晰地勾勒出系统的框架模型。这样总结出来的需求，往往能够与用户产生共鸣，让笔者在实践中也获益匪浅。

然而不久，第二个困惑又摆在了眼前，如何在“用例模型”（分析模型）的基础上进行设计呢？在从“用例描述”到“顺序图”、“活动图”的转换中，一直感到力不从心。所幸的是，Ivar 先生提出的“Robustness 分析方法”[1]又让我走出了这块新的沼泽。通过控制类、边界类，以及简明、随意的 Robustness 图，使这种转换变得 streamline（流线型）起来[2]。遗憾的是，UML 对 Robustness 图的舍弃，不知对多少实践者带来了这种困难。

但当我对“用例驱动方法”应用得越来越自如的时候，又遇到了新的困惑。在实际的系统分析和设计实践中，我突然发现类、组件与用例之间的对应关系是交错的。也就是一个用例可能会涉及多个类或组件，而一个类或组件也可能参与了多个用例。这种交错与缠绕一度让我一度感到心力交瘁。我们刚刚通过用例实现的“松耦合”设计，却又在具体到类的层面、实现的环节再次“耦合”在一起了。怎么办？如何解决？虽然从《软件复用：结构、过程和组织》中能够领会到一些解决的方法，但是总感觉在实践应用中还是有很大的局限性。

正是在这个时候，我从亚马逊网站上看到了本书。基于自己对 AOP 的了解，一看书名就让我感到无比的兴奋，心里念叨着“找到答案了，找到了！”。当找到一些更详细的资料和部分章节后，我确认了自己的想法，因此毫不犹豫地向 CSDN 的熊妍妍推荐了本书。在她的大力支持下，我终于看到了本书的全貌，答案浮出了水面：

[1]更多的细节内容可参阅 Ivar 先生的里程碑式著作《面向对象软件工程：一种用例驱动的方法》。
[2]笔者的实践心得已整理为《实战 OO》系列文章，连载于《程序员》杂志 2004.4~11 月刊。

- AOP 为你提供了一种手段，可以**将横切关注点的实现代码分离**，并模块化成为“方面”。面向方面提供了一种组合机制，使得在编译时甚至是运行时，再将横切行为组合到预期的操作和类中成为可能。而在操作和类的源代码中则可以摆脱横切关注点，从而使程序更易于理解和维护。

- 为了推进 AOP 的发展，本书作者提出了 AOSD，其目标主要围绕着如何使整个系统更好地模块化。它包括使功能性需求、非功能需求、平台特性等许多不同的关注点更好地模块化，从而使它们之间相互独立。**保持所有的关注点相互独立**，将使你构建的系统具有更易于理解的结构，并且更易于配置和扩展，以满足涉众各种衍生的需求。

- 如何进行 AOSD？如何识别方面？何时用类而非方面？如何详细说明方面？这需要一个明确的系统化方法来帮助你进行 AOSD。而事实上，已经有一种成熟的系统化方法。它就是**用例驱动方法**。它提供了一种明确的、聚焦于实现涉众关注点并给最终用户传递价值的开发方法。

哈，和我的预想一样。用例驱动方法与 AOP 的结合，必将会引发一场软件开发范型的革命。而通过阅读本书，你可以清除用例驱动方法的认识误区，了解方面技术的基本概念，掌握应用用例（application use case）、基础结构用例（infrastructure use case）及用例模块等新手段的应用，了解如何结合二者实现弹性的架构设计。而且本书还为你指出了一条实践 AOSD 的通途，可以说这是一本从实践中来，又回到实践中去的好书。相信阅读过本书的读者一定也会和笔者有相同的感受，一定不要错过这本经典的好书。

在此，我必须感谢参与了本书部分章节初译工作的周松奕（第 11~14 章）和吴兰陟（第 15~16 章）两位同仁；感谢 CSDN 的熊妍妍、博文视点的责任编辑陈兴璐，感谢她们辛劳细致的工作；同时也对参与了审校工作的 Ivar Jacobson Software China 公司的专家们、UMLChina 的潘加宇表示衷心的感谢，他们的工作使本书的翻译质量得到了很大的提高和保证。最后要感谢我的父母和妻子许高芳对我多年来的默默支持与鼓励。

鉴于笔者水平有限，因此在此必须感谢每一位读者，希望能够得到您的反馈与批评。欢迎来信（xf@csai.cn）与我一起交流与本书相关，与用例分析技术、需求工程、系统分析以及设计等软件开发相关的话题。

徐 锋

2005 年 10 月于厦门紫荆园

Preface

序

什么是面向方面编程

拿起这本书的一般都是软件开发社群中的人：测试人员、开发人员、项目领导者、项目经理、架构师、分析员，或者是许许多多其他与开发相关的某种角色。同时，一般也都是众多想改进自己开发方法的一员。希望自己开发的系统更易于维护，更具有扩展性，可复用性更好，如果是一位项目领导者，肯定还希望自己的团队更高效。然而，这些目标并不容易达到。

软件开发为什么这么难？其中一个理由是有太多的事情需要密切关注。从人本的角度，你不得不关注时间、预算、资源、技能等问题。作为团队的一员，经常会接到比自己所能承担的还多的任务。如果你有两个上司，而且每人都期望获得你100%的支持，那么你就必须付出200%的努力。作为一名开发人员，你必须理解应用、领域知识，以及平台的特性。当设计系统时，必须处理和平衡许多困难的关注点：系统如何达到其预期的功能，如何达到性能和可靠性的要求，如何处理平台的特性等。你可能会在编写类、操

作的代码时，发现程序必须实现许多功能，导致产出意大利面条式的代码，这说明设计存在不足。因此，必须对设计进行改进，使其更加模块化，更好地对关注点进行分离。正如团队中每个成员都必须明确地聚焦于自己的工作一样，每一个组件、每一个类，以及每一个操作都必须聚焦于其特定的目标。

但现有的技术会使你受到限制。无论你做什么，都会发现系统中很多地方需要完成日志、验证、留存、除错、跟踪、分布性、错误处理等方面任务的代码段。有时，操作或对象中的很大部分均不是在实现预期达到的功能。面向方面编程（AOP）将这种冗余称为“横切关注点”，这是因为这些代码段存在于许多操作和类中，也就是横切于操作和类中。横切关注点并不仅限于诸如验证、留存等这样的技术关注点。它还包括系统和应用功能，你会发现功能性需求的改变也经常会影响很多类。

AOP 为你提供了一个手段，可以将横切关注点的实现代码分离出来，并模块化成为“方面”。面向方面提供了一种组合机制，使得在编译时甚至是运行时，再将横切行为组合到预期的操作和类中成为可能。而在操作和类的源代码中则可以摆脱横切关注点，从而使得程序更易于理解和维护。

什么是面向方面软件开发

要推进 AOP 的发展，就需要一个贯穿从需求到分析和设计、实现和测试全过程的面向方面的软件开发的整体方法。这就是面向方面软件开发（AOSD）。

AOSD 的目标主要围绕着如何使整个系统更好地模块化，它包括使功能性需求、非功能需求、平台特性等许多不同的关注点更好地模块化，从而使它们之间相互独立。保

持所有的关注点相互独立，将使你构建的系统具有更易于理解的结构，并且更易于配置和扩展，以满足各种衍生的需求。

AOSD 并不仅仅是 AOP。它包括帮助你实现更好的模块化的一系列技术。这些技术包括面向对象、基于组件的开发，设计模式，诸如 J2EE 和.NET 之类的面向对象的框架。AOSD 并不是一种与现有技术竞争的技术，而是基于这些技术的改进提高。

基于用例的 AOSD

如何进行 AOSD？如何识别方面？什么时候用类而非方面？如何详细说明方面？这需要一个合理的和系统化的方法来帮助你进行 AOSD。开发社群正在呼唤这种软件开发的系统化方法的出现。

事实上，已经有了一种成熟的系统化方法。它就是用例驱动方法。通过聚焦于实现涉众关注和为用户传递价值，用例驱动方法提供了一种合理的开发方法。

大家都知道面向方面主要是有助于在编码阶段实现横切关注点的模块化，但我们还需要在开发的更早阶段甚至是在需求阶段实现横切关注点的模块化。而用例则是实现这一目标的卓越技术。用例就是横切关注点，用例的实现通常与多个类相关。事实上，你可以使用用例来为大多数横切关注点建模，在本书中我们就将展现如何建模。

面向方面的本质理念与用例驱动开发是相似的。这就意味着，从使用用例表示的涉众关注的需求，到使用方面来实现它们之间是可以无缝转化的。

简要地说，你可以从以下几个方面来基于用例进行 AOSD：首先使用用例为横切关注点建模。然后按照在类之上进行叠加（overlay）的方式来设计用例，这种叠加称为用

例切片和用例模块。接着使用方面技术来组合用例切片和用例模块以构建成整个系统的完整模型。

接下来，我们用家庭装修作为隐喻，来进一步解释这一概念。假设你有一座新房，但它还是一座空房，没有家具、没有灯、没有电话线、没有线缆、没有煤气，并且没有Internet！每一个缺少的设施或服务显然都是一个不同的关注点，很显然需要请不同的专业人员来安装这些设施及服务。这些设施和服务都是横切关注点，它们与不同的房间（或者称为对象）都有关联。这与用例很类似。为了决定如何着手其工作，每个领域的专业人员都必须设计一个平面图，通常是基于建筑平面图来设计的。建筑平面图展示出了房间和墙的位置。电工会复印一张建筑平面图，然后在上面绘制出准备安装的线缆；水管工人则绘制出如何布置水管，等等。不同的专业人员将独立工作，但他们的工作都是基于相同的建筑平面图。所有的工作就是完成这些图纸的叠加。

如果不同的专业人员都在幻灯片上绘制其图纸，则这叠加图可以在幻灯机上融合在一起。这种叠加与用例切片和用例模块十分类似。只要这些叠加图都是基于同样尺寸的建筑平面图，就可以在屏幕上看到所有工作完成时的完整景象。如果你需要修改Internet线路的位置，只需修改相应的叠加图，其图纸就会更新到融合的模型中。当项目有了其它叠加图，就可以得到房间的新的影像。往图片中增加更多的叠加图，或交换叠加图都是很容易的事。能够得到一致的影像主要得益于统一的尺寸标准。这描述的是建筑领域的工作情形。

使用用例切片和用例模块进行系统开发，就能够对横切关注点进行清晰的分离。你可以演化和扩展它们。这就将使得每个切片更易于复用。这也就能够生成可靠的切片，因为它们与其它切片之间并不会相互干扰。使用这一方法，将获得更好的可维护性、更好的扩展性，以及更高的性能。

开发社群可以从基于用例的 AOSD 中获得更多的利益。我们相信在用例驱动的开发方法中引入面向方面技术，其促进作用将是十分显著的，这是因为该方法作为驱动系统开发、测试和交付的手段，已经被广泛地接受了。在开发社群中可以很方便地获取大量与用例驱动相关的文献。很多专业人士、甚至公司一直在指导和推广用例驱动技术的应用方法。不管是大型的还是小型的项目团队都有应用这一方法的成功经验。因此，基于 AOSD 的用例驱动方法是很具有吸引力的。

本书是什么

本书系统化地介绍了如何基于用例进行 AOSD。其中覆盖了需求、分析、设计、实现和测试。并展示了如何使用 UML 对横向关注点和方面进行建模，以及如何基于用例和方面构建弹性的体系结构。还突出了在应用 AOSD 时必须注意的关键变革以及范型转变。同时还指出，如何才能够使你的项目最快地获得 AOSD 的好处。

我们还展示了如何在混合了诸如 J2EE 这样的面向对象框架、面向对象设计模式、AOP 等技术的环境中进行 AOSD，因为我们认为这是在实践中对你的挑战。同时还说明了如何将方面和用例分析映射到不同的设计和实现技术中去。

在本书中花费了大量的篇幅来描述如何基于用例和方面构建一个稳固的、能够弹性地应对变化的体系结构。

也许你们中的一部分人对 Ivar Jacobson 早期的作品比较熟悉，如《面向对象软件开发：一种用例驱动的方法》（Addison-Wesley，1992）和《统一软件开发过程》（Addison-Wesley，1999）。本书与这些书有十分紧密的关联。

可能一些读过面向方面相关书籍的人想知道如何完整地在软件开发中使用它。这本

书就是为此准备的。面向方面的新手也将从中学习到其主要的原则和应用方法。如果你熟悉用例驱动的方法，那肯定会欣然接受面向方面实现技术的好处。本书将帮助大家从更大的范围中认识方面技术，不仅仅是 AOP，也是 AOSD。

在本书中，我们将用一个简单的酒店管理系统为例，你会随着本书的进程不断地了解它。通过这个简单例子，会使围绕面向方面和用例的讨论更加具体化。

本书不是什么

本书不是一本编程方面的书籍，它不涉及现在可用的 AOP 语言以及面向方面的框架的技术细节。与这些细节相关的知识，请参考相应的指南和教程。本书是关于面向方面软件开发的（不仅仅是编程）。本书的重点在于关注从需求到代码的软件开发方法，协调地、迭代式地应用一系列技术来帮助你成功地开发软件系统。

本书并不打算成为一个面向方面设计的指南，也不想讨论所有可以想到的横切关注点（同步、事务处理、缓存机制等）。不过我们相信本书的知识广度，将足以针对你可能在实践中遇到的所有不同类型的横切关注点提供解决的原则和基础。

阅读本书所需的基础知识

要理解本书的内容有以下必备的基础知识，你至少需要对统一建模语言（UML）有一定了解。本书包括了一些 UML 的解释，但很大程度上希望你对其基础知识有一定程度的掌握。我们假定你知道什么是类，并且能够读懂用例图、类图、顺序图和协作图。顺便提一下，在 UML 2.0 中后两种图称为交互图和通信图。

如果你曾经在项目中使用过用例驱动开发方法，而没有面向方面的背景，那么一定会从本书中获得收益。我们特意在本书中花了一些时间来确保你能掌握 AOP 的基础知识。

如果你对面向方面技术十分熟悉，但对用例驱动开发了解很少，也不必忧虑。我们也为你考虑了。在第 2 篇中则是专注于用例和用例实现的。

本书中展现了一些 AspectJ 的代码实例，以让你对我们建议的支持 AOSD 的 UML 扩展有一个具体的印象。AspectJ 是一种支持 AOP 的 Java 扩展语言。因而有 Java 的基础知识对理解本书是很有帮助的。但是，我们要强调这不是一本 AOP 的书。本书并不打算教你 AspectJ 完整的语言。

由于要展示如何在混合的环境中应用 AOSD，以及如何处理不同的平台特性，因此需要使用一些平台来使得讨论更加具体化。为了这个目的，我们选择了 J2EE，因此一些 J2EE 的知识对理解本书也很有用。如果曾经听说过 servlet 和 EJB，就说明你有足够的背景知识。如果了解 J2EE 核心模式，那就更好。

好了，欢迎你阅读本书。

如何阅读本书

我们将本书组织为五个部分：

第 1 篇：用例和方面的案例

第 1 篇主要是对“序”的扩展。它的目的是帮助你理解什么是基于用例的 AOSD。将侧重于对等横切关注点和扩展横切关注点的案例，并说明如何用方面来解决它们。另外将通过一些简单的代码实例来介绍 AspectJ，这是当前应用最广泛的 AOP 技术。同时还提供了用例驱动开发的概述，说明了现在的用例技术是什么，用例如何实现，如何映

射到类；以及我们所期望的能够与方面结合的未来用例技术，即用例切片和用例模块。

第 2 篇：基于用例捕获关注点并建模

无论你是否熟悉用例，都应该阅读第 2 篇。它将对用例技术进行概括性总结，并澄清一些普遍的误解。第 2 篇还将对用例建模技术进行补充，使其与面向方面分析与设计无缝结合。特别地，我们还将展示如何在用例中对连接点集（joint point set）进行建模。并将以一个详尽的基于用例对功能性和非功能性的横切关注点进行建模的实例作为第 2 篇的结束。它们将使用多种不同的应用用例（application use case）和基础结构用例（infrastructure use case）进行建模。同时本书还将演示如何一直将这些不同的用例应用到实现阶段。

第 3 篇：基于用例模块分离关注点

第 3 篇将深入说明用例切片和用例模块的概念。用例切片能够帮助你保持用例特性与设计模型的分离。它通过曾经提到的叠加来使横切关注点相互分离。本书将展示如何使用 UML 对用例切片和方面进行建模，以及针对 AOP 的 UML 符号扩展。我们会在第四篇使用这些符号及其背后的概念，这些符号均总结于附录 B 中。

第 4 篇：基于用例和方面构建体系结构

决定项目成功与否的关键在于其体系结构。第 4 篇展示了如何一步步获得一个良好的体系结构。良好的体系结构要完成的事之一就是保持不同的关注点相互分离。要使特定用例（use-case-specific）与通用用例（use-case-generic）相分离，特定应用与通用应用相分离，平台特性与平台无关性分离，等等。这种分离不仅使你的系统更加易于理解和