软件工程技术丛书

HZ BOOKS

对象技术系列

面向对象软件开发教程

（原书第2版）

The Object Primer

The Application Developer's Guide to Object Orientation and the UML

(Second Edition)

（加） Scott W. Ambler 著　　车皓阳　刘锐　译

机械工业出版社
China Machine Press

软 件 工 程 技 术 丛 书

对象技术系列

面向对象软件开发教程

（原书第2版）

The Object Primer

The Application Developer's Guide to Object Orientation and the (Second Edition)

（加） Scott W. Ambler 著　　车皓阳　刘锐　译

机械工业出版社
China Machine Press

本书是一本优秀的面向对象技术的基础教程，深受全世界专业人士和学生的广泛好评。作者以一种通俗易懂的方式清晰地解释了面向对象的基本概念，使用 UML 技术介绍了面向对象的需求、分析和设计方法。全书共分 11 章，每章专注于一个主题，包括需求收集、需求确认、OO 分析、OO 设计、OO 编程、OO 测试、软件过程等。每章最后都配有复习题，有助于读者自己测试对所学内容的掌握程度。本书作为 OO 入门读物，可用作高校面向对象技术的教材，也广泛适用于 OO 程序员、业务分析员、用户、项目经理、系统设计员等广大读者。

Scott W. Ambler: The Object Primer: The Application Developer' s Guide to Object Orientation and the UML, Second Edition (ISBN 0-521-78519-7).

Original English language edition published by Cambridge University Press.

本书版权登记号：图字：01-2002-1184

图书在版编目（CIP）数据

面向对象软件开发教程（原书第 2 版）/（加拿大）艾姆勒（Ambler, S.W.）著；车皓阳等译. -北京：机械工业出版社，2003.3

（软件工程技术丛书　对象技术系列）

书名原文：The Object Primer: The Application Developer's Guide to Object Orientation and the UML, Second Edition

ISBN 7-111-11678-X

Ⅰ. 面…　Ⅱ. ①艾…②车…　Ⅲ. 面向对象语言，UML-程序设计　Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2003）第 009744 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：刘立卿

北京昌平奔腾印刷厂印刷·新华书店北京发行所发行

2003 年 6 月第 1 版第 1 次印刷

787mm×1092mm　1/16·24.5 印张

印数：0 001-5 000 册

定价：39.00 元

译者序

正是由于本书，我们才得以了解 Scott W. Ambler，他既是一名优秀的开发人员，又是一名高产的作家，同时还是一位咨询公司的总裁。Ambler 对软件工程中诸多领域（例如敏捷建模、极限编程、软件模式、统一过程等）都有涉足，曾出版过多部相关著作，近年来的研究重点放在了统一开发过程上面。作者在本书中以一种详熟、简明的风格带领我们踏进了对象技术的神奇世界。本书不同于一般软件工程类图书，它更侧重于实际应用，书中给出了一些具体应用情景和实用技巧，如果我们能对其多加思考，并能熟练运用，相信会收获不菲。“UML 播种机”（*http://www.umlchina.com*）中有关于 Scott W. Ambler 的访谈录（《非程序员》杂志第 12 期中亦有相关内容），从中读者们可以了解他的兴趣所在，领略他关于“*空手道和太极拳*”的精彩论断。

我们大家都知道，翻译有三个基本原则：“信”、“达”、“雅”。科技书籍的翻译与文学书籍有一些不同之处，它更侧重于达到“信”和“达”两字所确定的目标，而并不过份执着于“雅”字所要求的境界。我们认为，译文与原文一样都是随着时代而变化的，同一段原文在不同时代背景下也应翻译为不同的内容，换句话说，翻译应该具有时代特征，我们认为三字原则中的“信”字应该包含这一特征。例如，Operational Research 为“运筹学”的意思，但在 1956 年时的翻译却是“运用学”，如果现在我们仍然翻译为“运用学”的话，恐怕没有人会弄清楚我们在说什么。因此，在本书中，我们也遵循这一原则进行翻译。

我们把本书的中译本奉献给亲爱的读者，希望读者能从中得到些许收获。由于译者知识水平和眼界所限，缺点和错误在所难免，敬请读者对翻译中的缺点和错误提出批评、指正。

车皓阳 刘锐

2003 年于中科院软件所

OO式作者简介

scottAmbler 是 **OOConsultant** 类的一个实例，这个类为基于 **denverColorado** 的 **roninInternational**（*www.ronin-intl.com*）工作。**roninInternational** 实现 **objectArchitectureConsulting( )**、**componentArchitectureConsulting( )**和 **softwareProcessConsulting( )**等操作。**HumanoidLifeForm** 的所有实例都可以发送电子邮件给他，电子邮件地址是：

scott.ambler@ronin-intl.com

scottAmbler 是一个多才多艺的对象，他可以以多种形式改变自己的类型，以满足 **roninInternational** 客户不同的要求。例如，他常常变为 **OOMentor**、**OOArchitect**、**OODeveloper** 或 **SoftwareProcessMentor** 对象。Scott 自 1991 年以来已经是 **OOConsultant** 的一个实例了。他曾是 **MastersStudent** 对象，从 **universityOfToronto** 类那里接受了一个 **InformationScienceDegree** 的实例。作为 **MastersStudent**，**scottAmbler** 在 OO CASE 领域做了大量工作，并在计算机支持的协同工作（这是关于群件的一种学术名称）中实例化了一个 **ThesisPaper** 对象。他的 **ThesisPaper** 实例响应的惟一消息是 **sitOnShelfAndGatherDust**，**scottAmbler** 发现这虽然令人失望，但又是早就料到的。**scottAmbler** 曾供职于多家机构，包括 **Bank**、**InsuranceCompany**、**TelecommunicationsCompany**、**InternetStartup**、**ServicesOutsourcer**、**MilitaryContractor** 和 **ProductDistributor** 等类的实例。

ScottAmbler 定期为 **softwareDevelopment**（*www. sdmagazine.com*）和 **computingCanada**（*www.plesman.com*）中的对象专栏写作。他的个人网站地址是*http://www.ambysoft.com*，在此他公布了一系列关于对象的白皮书和其他一些软件开发资源。

原　序

最近在网上冲浪时，我看到了一些非常有趣的数字。分析人员指出，30%～40%的软件项目订单都被取消了。软件项目平均费用几乎是最初预计费用的两倍多，只有 15%～20%的软件项目可以按预计时间和预算完成。另外，还有机会成本，它测量那些损失的商业机会，而这些商业机会是：如果项目从一开始就能正常进行的话本可以抓住的机会。机会成本是一个真正的问题，现在已经增长到了相当惊人的程度。

面对这些数据我变得非常迷惑，这就是我们从业的真实情况吗？很明显，现实情况的确如此——这活生生的数字不会说谎。那什么才是产生这种灾难性后果的原因呢？

最显然的一个答案是在项目中缺乏足够的开发人员资源。我们所处的时代是一个开发人员统治一切的时代，社会急需开发人员，但开发人员又十分稀缺。这就造成了 IT 行业中显著的供需不平衡。

然而，这只是冰山一角。还有许多其他因素导致了软件开发的失败。例如差劲的软件开发过程、苛刻的用户以及不恰当的面向对象分析和设计等等。

好的一面是，我们能够通过教育改变这些不利因素。通过了解正确的软件开发，就可以避免那些曾出现在其他工业中的绊脚石。避开这些问题非常重要，尤其当时间与市场的对比是以周而不是以月或年来计算时。

本书是学习面向对象软件开发的最好教程。当阅读本书时，Scott Ambler 将为你解释很多概念和范型，这些内容都可以立即在软件开发中用到。这些概念以一种易于理解、令人愉快的方式进行讲解，使得任何开发者，不管其背景如何，都可以掌握这些概念。

我坚信阅读本书会增加项目成功的机会。在进行软件开发时，本书能使你对自己可接受的内容尽早下定决心，这将有助于增加你作为独立开发人员的市场竞争力。

我希望你们在项目开发过程中工作顺利，让我们一起，共同完善我们自己。

Ed Roman
CEO，The Middleware Company
edro@middleware-company.com

前　言

本书是一本简明易懂的基础性教程，通过应用 UML 技术介绍了面向对象的概念、需求、分析和设计方法。面向对象（OO）是自结构化方法以来对系统开发过程的最重要的变革；在 20 世纪 90 年代，它代替了结构化范型而成为软件开发的主要方法。面向对象常常被用来开发复杂的系统，而了解如何使用面向对象技术进行工作并不很复杂。本书与其他面向对象的入门书籍不同，它是从实际开发人员的角度来阐述问题，只有这些开发人员才经受过学习这种令人兴奋的软件范型所面临的困难。

谁应该阅读本书

如果你是一名大型机上的 COBOL 或 PL/1 程序员的话，同时你又第一次参与 OO 的项目，本书会适合你。如果你是一名业务分析员或用户代表，需要处理 OO 应用的需求文档，那么本书同样也适合你。如果你是一名项目经理，需要在 OO 上加把劲的话，本书正是为你而准备的。如果你是一名系统设计员，你的单位正准备转向对象技术，本书也适合你。如果你是一名第一次上 Java 或 C++课程的学生，这本书也会适于你。但如果你是一名研究人员或你的学术研究兴趣在于软件工程理论的话，很抱歉，这本书还做不到让你满意的程度。

贯穿本书，我大量地使用了“开发人员”这个术语：开发人员是指任何参与软件应用程序开发的人。它包含了程序员、分析员、设计人员、用户代表、数据库管理员、支持工程师，等等。是的，很多人可能不会把用户代表包含于其中，但我的经验是，积极的用户参与往往是软件项目成功的决定性因素。用户可以积极地参与到需求、分析，有时甚至是设计当中，对我而言，显而易见，用户应当被当做开发人员。

为什么需要阅读本书

通过阅读本书，你可以获得对面向对象概念的深刻理解，同时也可以掌握一些基本的面向对象建模技术。这些都是在开发面向对象应用时，尤其是那些基于 C++和 Java 的软件时所必备的基本技能。书中进一步将这些技术放在一个实例的背景当中进行讨论，你将会发现它们确实能够在真实世界中得以应用。

为什么再版

当我 1994 年写出本书的第 1 版时，对象工业界相对来说处于较为混乱的状态。建模标记的“战争”还在不断地蔓延，尽管已经存在七、八个广为人知的建模标记以及 30 多个不是那么流行的符号，但却仍然没有出现明显的优胜者。现在，当我在 1999/2000 年冬天重写本书的时候，统一建模语言已经成为工业标准。1994 年，以用户为中心的设计充其量仅是软件开发环境中一个不太重要的概念；现在以应用为中心的设计（例如本质用况建模）也已成为工业标准。在 1994 年时，许多组织还不太确定是否要采纳对象技术，而在 2000 年，对象和组件技术已经成为大多数组织内部所承认的标准。1994 年，Smalltalk、Eiffel 和 C++还在竞争成为语言的统领者，而到 2000 年，Java 却明显地成为市场中的胜利者，紧随其后的是 C++—— 一般用于性能要求高的应用当中。真是时过境迁哪！

第 2 版中有哪些新内容

第 2 版保持了第 1 版的强势和品质，它简单而易于理解。但是，第 2 版在第 1 版的基础上增加了新的内容，加入了一些以应用为中心的设计技术，例如本质用况建模和基本用户界面建模。还包括了 CRC 建模——极限编程的一项关键技术。分析和设计建模也增加了新内容，包括了基本的 UML 技术以及持久性（数据）建模。同样，也加入了一些介绍如何从设计到编程再到测试等内容的章节，进一步完善本书。最后，像模式和统一过程这样先进的开发技术和过程也被包含进来，以介绍目前的技术发展状况。更重要的是，第 2 版反映了我在过去的 5 年来在各个问题领域帮助其他人了解如何使用对象技术的经验。

目　录

开发人员通常擅长按正确的方式构建系统；
却不擅长构建合适的系统。

第1章 简　介

你将从本章学到什么？

- 什么是面向对象
- 传统开发方法所遇到的困难
- 本书是如何组织的
- 怎样阅读本书

为什么需要阅读本章？

理解为什么需要掌握面向对象技术，理解结构化范型所面临的挑战以及对象范型怎样解决这些问题。

本书描述了面向对象（OO）范型，这种开发策略基于下面这样一种概念：即系统可以由一系列称做对象（object）的可重用组件来构建。与在结构化范型里分离数据和功能不同，对象包含了它们两者。虽然面向对象范型和结构化范型听上去大致相同，但是你将在本书中看到它们实际上是非常不同的。许多经验丰富的开发人员所犯的一种常见错误是他们认为自己一直都在“制造对象”，只因为他们应用了相似的软件工程理论。事实是你必须认识到对象是不同的，这样你才能顺利地开始你的学习经历。

1.1 结构化范型与面向对象范型

结构化范型（structured paradigm）是一种基于如下概念的开发策略，即一个系统应该被划分为两部分：数据（使用数据/持久化模型建模）和功能（使用过程模型建模）。简言之，使用结构化的方法，数据将和设计模型中以及系统实现（也就是程序）中的行为分离。

另一方面，如图 1-1 中所示，面向对象范型背后的主要概念不是把系统定义为两个分离的部分（数据和功能），而是需要把系统定义为一组正在交互的对象。对象可以完成一些事情（也就是说它们有功能），它们也知道一些事情（也就是说它们有数据）。虽然这听起来和结构化范型相似，但事实上却是不同的。

定 义

范型（*paradigm*）：做事情的整体策略或观点。范型是一套特定的思想集。

考虑一下你会如何完成一所大学信息系统的设计。如果采用结构化的方法，你将定义数据库的布局和程序的设计来访问这些数据。数据库中存有关于学生、教授、教室以及课程的信息。程序可以让用户登记学生的选课情况，分派教授授课，安排授课教室，等等。程序可以访问并更新数据库，实际上它可以支持学校的日常运作。

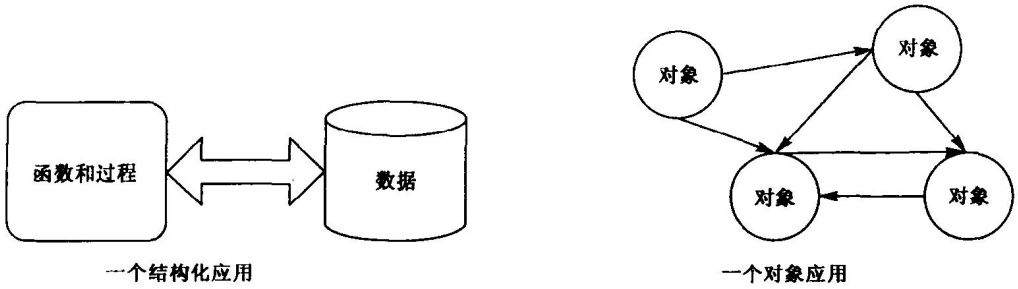

图 1-1 比较结构化范型和面向对象范型

现在让我们以一种面向对象的观点来考虑大学信息系统。在现实世界中，有学生、教授、教室和课程，所有这些事情都将被当成对象来考虑。学生知道一些事情（他们有自己的名字、住址、生日、电话号码等等），可以完成一些事情（登记课程、取消课程以及支付学费）。教授也知道一些事情（他们所教授的课程以及他们的名字），同样也能完成一些事情（输入分数以及提出进度要求）。从系统的观点来看，教室也知道一些事情（它们所处的建筑物以及它们的房间号），也应该能够完成某些事情（例如告知空闲状态以及可以预定它们的时间段）。课程也知道某些事情（它们的名称、描述以及谁将选这门课），也能完成一些事情（例如通知学生选课或通知学生取消选课）。

要实现这样的系统，我们可以定义一组能进行交互的类（类（class）是相似对象的一种通用表示）。例如，我们将使用“课程”、“学生”、“教授”以及“教室”类。这样的类将组成我们的应用，它将包括功能（程序）和数据。

对于个人，OO 是一种全新的思考方式。对于机构，OO 则彻底改变机构内的系统开发文化。

如同你所见到的那样，面向对象方法将使人们对什么是应用产生完全不同的认识。与其说我们拥有一个可以访问数据库的程序，不如说我们拥有一个存在于对象空间中的应用。对象空间（object space）是驻留应用的程序和数据的地方。我将在第 5 章进一步讨论这个概念，现在你只需把对象空间当成虚拟内存就可以了。

定 义

类（*class*）：对象创建（实例化）时所使用的模板。尽管在现实世界中，Doug、Wayne 以及 Bill 都

是“学生对象”，但我们可以把他们模型化为“学生”类。

对象空间（*object space*）：包含所有可访问的永久存储的内存空间，对象存在于其中并可以进行交互。

对象（*object*）：可以是一个人、一个地方、一件物品、一个概念、一个事件、一个屏幕显示或者一个报表。对象知道一些东西（也就是说它们有数据），它们可以完成一些事情（也就是说它们有功能）。

面向对象范型（*object-oriented paradigm*）：一种基于以下概念的开发策略：即从被称为对象的可重用组件来构建系统的。

OO（**面向对象**）：可以和“object-oriented”以及“object orientation”这两个术语交替使用的缩写词。例如，如果我们说 OO 编程，我们实际上是指面向对象（object-oriented）编程。如果我们说这是一本描述 OO 的书，我们其实说的是这本书介绍的是面向对象（object orientation）。

1.2 本书怎样组织

本书涵盖了开始 OO 开发时需要了解的任何事情。

本书涵盖了在现实应用开发过程中证明行之有效的前沿面向对象技术和概念。详细描述了这种称为面向对象的新方法、需求技术（例如用况和 CRC 建模）、OO 概念、使用 UML 建模技术进行 OO 分析和设计、OO 编程、OO 测试以及 OO 软件过程。本书结束时还讨论了如何继续学习过程的问题，包括对通用面向对象技术的描述以及应用软件过程技术的描述。

表 1-1 总结了每一章的内容。图 1-2 展示了本书的组织结构，包括每一章以及各章之间的关系。图的左侧描述了软件过程的基本活动，例如收集需求、面向对象分析与面向对象编程。框间的箭头代表各章之间的关系：你可以看到描述收集需求、确认需求以及面向对象分析的这几章相互之间关系是比较紧密的。第 9 章描述了面向对象测试，及确认分析、设计以及编程效果的测试技术。图 1-2 右侧罗列的章节，提供了理解面向对象范型很关键的一些内容。

定　义

统一建模语言（*Unified Modeling Language，UML*）：用于面向对象软件的一种标准建模语言定义，包括定义一种建模符号及其应用这种符号的语义，就像 OMG（对象管理组）所定义的那样。

全生命周期面向对象测试（*Full Lifecycle Object-Oriented Testing，FLOOT*）：面向对象开发中的一种测试方法学，它提供方法来验证你的应用程序在开发的每个阶段都能正常工作。

表 1-1　每章中包含的内容

章	描　述
第 2 章：一种新的软件范型	讨论了面向对象的优点和缺点，为什么说对象技术将扎下根来，并对软件过程进行了概述
第 3 章：收集需求	描述了需求收集技术，包括用况、变例、CRC 建模、会见、用户界面原型。并包含了如何共同使用这些技术的讨论
第 4 章：确认需求	描述了用况情景测试以及需求走查等需求确认技术
第 5 章：面向对象概念	描述面向对象的基本概念，包括继承、多态、聚合以及封装
第 6 章：面向对象分析	描述了通用的面向对象分析技术，例如顺序图以及类图等。并描述了如何从需求转换为分析以及这些技术如何配合使用
第 7 章：面向对象设计	描述了通用的面向对象设计技术，例如类图、状态图、协作图以及持久化模型。并描述了如何从分析转换为设计以及这些技术如何配合使用
第 8 章：面向对象编程	概述了通用的面向对象编程技巧和技术，讨论了如何在设计和编码之间进行转换
第 9 章：面向对象测试	概述了全生命周期面向对象测试（FLOOT）方法学及技术
第 10 章：面向对象软件过程	概述了面向对象软件过程（OOSP）以及增强的统一过程生命周期
第 11 章：如何继续学习过程	讨论了要继续面向对象学习过程所需要完成的事情，包括对先进的对象技术，以及像 Java、EJB、C++以及基于组件的开发技术的描述

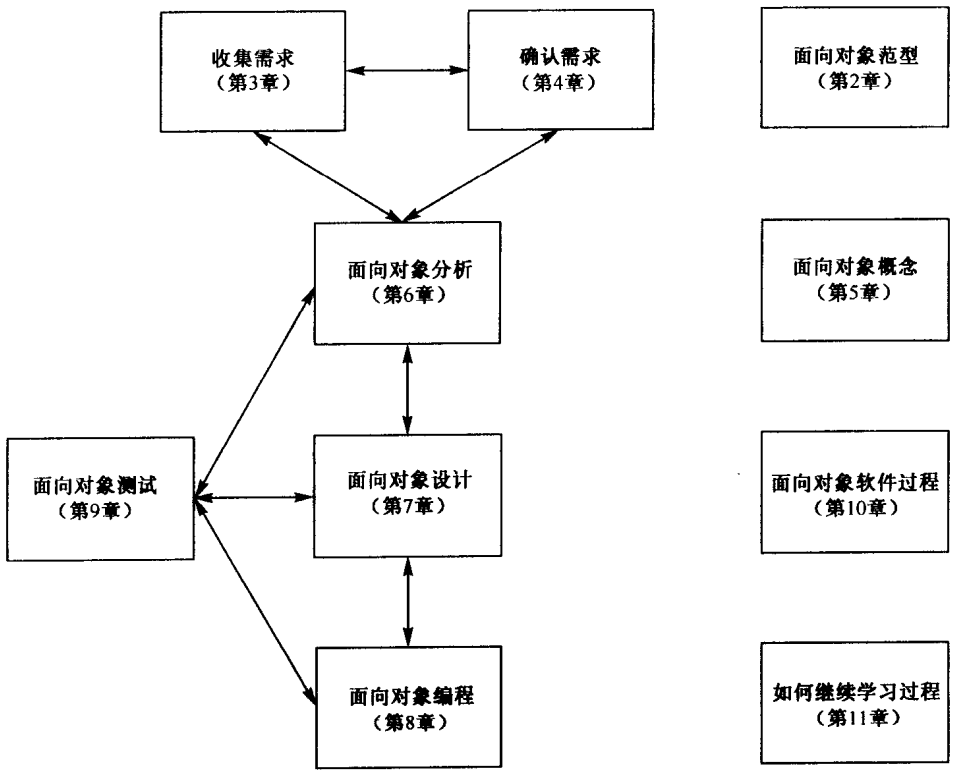

图 1-2　本书的组织结构