Mastering UML with Rational Rose 2002

UML与Rational Rose 2002从入门到精通

〔美〕 Wendy Boggs 著
Michael Boggs

邱仲潘 等译

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内 容 提 要

本书深入浅出地介绍了统一建模语言（UML）和Rational Rose软件，通过航空公司与购物推车例子介绍如何用UML和Rose进行项目需求分析、结构规划和生成框架代码，以及如何从现有系统逆向转出工程代码，生成Rose模型。并分章介绍了C++、Java、Visual Basic与CORBA/IDL和XML代码的代码生成与逆向转出工程代码。通过本书学习，项目开发人员可以用这个全新工具紧扣用户需求，方便地开发出符合用户需求的系统或根据用户需求对现有系统进行改造。

本书适合项目开发人员参考，也适合作为大学教材或自学材料。

版权贸易合同登记号：01-2001-5177

图书在版编目（CIP）数据

UML与Rational Rose 2002从入门到精通/（美）伯格斯（Boggs, W.），（美）伯格斯（Boggs, M.）著；邱仲潘等译—北京：电子工业出版社，2002.7

书名原文：Mastering UML with Rational Rose 2002

ISBN 7-5053-7731-0

Ⅰ. U… Ⅱ. ①伯… ②伯… ③邱… 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（2002）第042834号

责任编辑：马振萍 刘 丽

印 刷：北京天竺颖华印刷厂

出版发行：电子工业出版社 http://www.phei.com.cn

北京市海淀区万寿路173信箱 邮编：100036

北京市海淀区翠微东里甲2号 邮编：100036

经 销：各地新华书店

开 本：787×1092 1/16 印张：33.5 字数：860千字

版 次：2002年7月第1版 2002年7月第1次印刷

定 价：50.00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

致　谢

写书是很费劲的事。尽管作者做了一部分工作，但大量工作是由整个小组完成的。感谢参与本书的每一位成员。感谢Sybex公司的Richard Mills与Jordan Gold提供了出版本书的机会；感谢Tom Cirtin提供了出版前的准备材料。感谢Eric Aker进行技术编辑。感谢Sybex公司编辑与制作组的Mae Lum、Donna Crossman、Jill Niles、Christine Detlefs、Kevin Ly与Tony Jonick。感谢操作员Nancy Guenther和校对员Emily Hsuan、Nelson Kim、Yariv Rabinovitch与Nancy Riddiough。没有你们的努力，本书就不可能问世。

译 者 的 话

本书翻译过程中得到了蔡维斌、陈志斌、洪鹰、杨威、周阳生、刘文红、邹能东、彭振庆、黄志坚、李耀平、江文清等同志的大力帮助，刘文琼、温连英、邱冬金、邱燕明等同志完成了本书的录入工作，刘云昌、刘昌和兄弟帮助进行了书稿与打印稿的校对，在此深表感谢。

邱仲潘

前　言

在这个面向对象应用程序开发不断变化的时代，在合理时间内开发和管理高质量应用程序变得越来越困难。为了面对这种挑战，制定出每个公司都能使用的通用对象模型语言，统一建模语言（UML）被及时推出。UML是信息技术行业的蓝图，是详细描述系统结构的方法。利用这个蓝图，我们建立和维护系统越来越容易，保证系统能适应需求的改变。

有许多书籍介绍了应用程序的快速开发过程、面向对象分析与设计、对象模型和UML。本书重点介绍用UML和Rational Rose 2001、2001A与2002进行系统设计。Rose是用UML快速开发应用程序的工具之一。它支持Use Case框图、Activity框图、Sequence框图、Collaboration框图、Statechart框图、Component框图和Deployment框图。通过正向和逆向转出工程代码的特性，可以支持C++、Java、Visual Basic和XML DTD的代码生成和逆向转出工程代码。还有几个面向对象语言的插入件可以进一步扩展Rose的功能。

新内容

近年来，面向对象世界发生了较大的变化。基于Web的系统大量出现，面向对象工具与语言不断革新，开发工作更加全球化。当然，UML也不断演变。UML已经成为一个综合性语言，可以建模系统设计的所有不同方面。可以用UML建模XML、J2EE和各种编程语言中的结构，设计数据库和建模计算机化系统所处的整个行业。

随着UML的演变，Rose也不断演变。Rational Rose 2001、2001A与2002增加了新功能，如支持Web应用程序开发，使设计过程更加顺利。Rational Rose 2001、2001A与2002的新特性包括：

- 业务模型
- Activity框图
- Web模型
- 增加Java与J2EE支持
- 增加EJB支持
- XML DTD支持
- 数据模型
- ANSI C++支持
- SQL Server 2000支持
- Sun Java Server Pages与Microsoft Active Server Pages支持

本书阅读对象

本书适用于UML和Rose的初、中、高级用户，包括编程人员、软件结构师和分析师。编写本书时，我们想回答三个问题：UML每个框图和结构是什么、为什么用每种框图、如何用Rose设计这些框图和结构。

本书包括Rose基础：

- 如何建模企业过程
- 如何创建角色、使用案例和Use Case框图
- 如何创建Sequence和Collaboration框图
- 如何创建类、属性、操作、关系和Class框图
- 如何创建Statechart框图
- 如何创建组件和Component框图
- 如何创建Deployment框图
- 如何用UML与Rose创建系统的完整及详细蓝图
- 如何用Rose 2001、2001A与2002提供的新特性
- 如何用Rose生成C++、Java和Visual Basic代码
- 如何从C++、Java和Visual Basic逆向转出工程代码到Rose中
- 如何生成和逆向转出XML DTD
- 如何用Rose建模数据库结构
- 如何建模Web应用程序

本书不必按顺序阅读。每章都提供了Rational Rose的详细介绍。大多数章后面提供使用UML与Rose的实践练习。

尽管本书介绍了UML和模型元素的基础，但主要介绍Rational Rose支持的部分，而不介绍UML的每个方面。

如果你不熟悉UML与Rose，可以按顺序阅读第1章到第11章并做完所有练习。这些练习介绍了设计样本系统的一个过程。如果你熟悉UML与Rose，则可以把本书作为UML与Rose特定问题的参考手册。

本书组织形式

本书共有19章和一个附录。第1章和第2章概述UML、对象模型过程和Rational Rose工具。第1章介绍UML基础和Rational Rose。我们将介绍不同的UML框图，每种框图的作用及如何建立框图。第2章介绍Rose，介绍其用户界面的组成和Rose提供的功能。

第3章到第11章介绍Rose基础，包括创建与更新框图、增加类与类细节和产生报表。通过创建这几章介绍的不同框图，系统开发小组可以对所建系统的结构和行为有完整的了解。

第12章到第17章介绍Rose的C++、Java、Visual Basic和XML代码生成逆向工程的功能。其中介绍了每个UML如何构筑特定编程语言的映射，列举大量Rose产生的代码例子，并介绍从源代码逆向工程的模型元素。

第18章介绍Rose的Data Modeler特性，这个强大的工具支持建模数据库表格、连接、存储过程、触发器和其他元素。可以从对象模型自动创建数据模型，或从数据模型逆向生成对象模型。Data Modeler还有代码生成特性，可以创建和运行DDL（数据定义语言），用于建立或更新数据库结构。数据库结构也可以逆向工程到Rose模型。

第19章概述Web模型过程，包括建模Web应用程序的不同层次，从Web应用程序模型生成代码和Web应用程序逆向工程。

最后，如果要了解UML的启蒙知识，可以看附录“UML入门”。

本书选配光盘

本书介绍Rose特性时会建立一个样本系统的一些Rose模型。本书选配光盘包括这个系统的样本UML模型和用Rational Rose生成的代码，还有一些样本Rose脚本，是用Rose所带的脚本语言编写的宏。还有Rational website的链接，可以找到Rational伙伴与产品、UML和对象模型的所有信息。

与作者联系

如果有关于Rose和UML的问题，可以与我们联系。Wendy的地址是wboggs@attbi.com，Mike的地址是mboggs@attbi.com，或者访问Sybex站点www.sybex.com。

目 录

第1章 UML简介

业务发展越来越快，更加要求企业在市场中具有竞争力和维持力。在这个电子商务时代，“传统”系统开发方法已经力不从心。系统应以“Internet速度”开发。速度的加快也要求有更加灵活的系统。过去，用户可以向数据处理中心发送请求，两年之后才进行改变。现在，用户向IT部门发送改变请求，要求两周内就完成。六周开发周期、着急的经理、着急的用户和XP（极速编程）的概念都要求：系统应迅速改变。

本章介绍UML（统一建模语言，Unified Modeling Language），UML是最广泛使用的面向对象系统的标准建模方法。本章首先介绍UML的历史与现状，以及面向对象编程概念，然后介绍如何使用UML构造应用程序。

- 了解面向对象机制与可视模型
- 了解图形标注类型
- 了解UML框图类型
- 用可视模型开发软件

面向对象机制简介

在软件工程早期，主流是结构化编程。编程时首先开发标准代码块，进行打印等操作，然后将代码复制和粘贴到编写的每个应用程序中。尽管这样可以减少新应用程序的开发时间，但如果代码块需要改变，则会遇到麻烦，因为开发人员要在复制代码的每个地方进行修改。结构化编程的问题在面向对象编程中得到解决。

在面向对象编程中，开发人员创建的代码块称为对象，然后在各种应用程序中使用这些对象。需要修改其中一个对象时，开发人员只要在一个地方进行改变。各个公司纷纷采用这种新技术，将其集成到现有应用程序中。事实上，大多数当今开发的应用程序都是面向对象的。一些语言，例如Java都需要用面向对象的结构。什么是面向对象呢？

面向对象机制是另一种观察应用程序的方式。利用面向对象方法，把应用程序分成许多小块（或对象），这些对象是相互独立的。然后可以组合这些对象，建立应用程序。可以把它看成砌砖墙。第一步要建立或购买基本对象（各种砖块）。有了这些砖块后，就可以砌出砖墙了。在计算机领域中建立或购买基本对象后，就可以把对象集成起来，创建新的应用程序。

例如，在结构化编程领域中，要创建带列表框的窗体，就要编写大量代码，用来创建窗体的代码，以及创建与填列表框的代码和创建OK按钮的代码，这些代码用来接受列表框中的值。而在面向对象编程中，只要用三个对象（通常是预建对象）：一个窗体、一个列表框和一个OK按钮。老的编码方法是“从头建立，从旧程序中复制可用代码，节省时间”；而新的编程方法是“把一组对象集成起来，然后集中考虑每个特定应用程序的独特之处”。

面向对象机制的一个主要好处是可以一次性地建立组件，然后反复地使用。就像砖块

可以重复用来盖城墙、盖房子、盖太空飞船，面向对象的基本设计和代码可以重复用于会计系统、库存系统或订单处理系统。

这种面向对象机制与传统开发方法有什么不同呢？传统开发方法集中考虑系统要维护的信息。用这种方法时，我们要向用户询问他们需要什么信息，然后设计保存信息的数据库，提供输入信息的屏幕并打印显示信息的报表。换句话说，我们关注信息，而不是关注信息的作用或系统的功能。这种方法是以数据为中心的，多年来用它已建立了成千上万个系统。

以数据为中心的模型适合数据库设计和捕获信息，但用来设计商业应用程序就有问题。一个主要问题是系统要求随着时间不断变化。以数据为中心的系统可以方便地处理数据库的变化，但很难实现商业规则变化和系统功能变化。

面向对象机制的开发正是用来解决这个问题的。利用面向对象机制，我们同时关注信息与功能。因此，我们可开发弹性很大和适应信息与功能变化的系统。

要实现灵活性带来的好处，只能通过设计好的面向对象系统。这就要求了解面向对象的基本原则：包装、继承与多态。

包装

在面向对象系统中，我们将信息与处理信息的功能组合起来，然后将其包装成对象，这称为包装。另一种理解包装的方法就是把应用程序分解成较小的功能组件。例如，我们有与银行账目相关的信息，如账号、结余、客户名、地址、账号类型、利率和开户日期。我们还有银行账目的功能：开户、销户、存款、取款、改变类型、改变客户和改变地址。我们将这些信息与处理信息的功能包装成账目对象。结果，银行系统对账目的任何改变就会在账目对象中实现。它是所有账目信息与功能的集合。

包装的另一好处是将对系统改变的影响限制在对象内。可以把系统看成水，所需的改变看成大石头。一块石头投进水里，会溅起一片水波纹，它们经过湖面、冲向岸边、振荡并与其他水花碰撞。事实上，有些水花会溅到岸边和湖外。换句话说，一石激起千层浪。如今我们要包装湖水，将其分成较小的水体，并围起来。然后，扔出一块石头，仍然水花四溅，但只能限制在围起来的小范围内。因此，通过包装湖水，我们限制了击石引起的波浪，如图1.1。

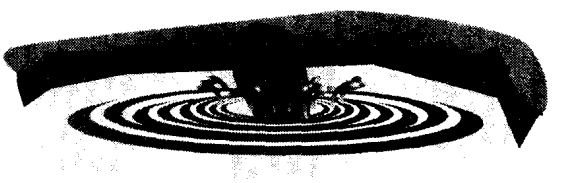

图1.1 包装：湖水模型

下面要把这个包装思想用于银行系统。最近，银行管理层决定，如果客户在银行有信用账号，则可以用信用账号作为支票账号的透支额。在无包装系统中，我们要用类似散弹猎枪的方法进行影响分析。我们并不知道系统中所有取款功能用在哪里，因此要到处寻找。找到之后，我们要根据这个新要求进行改变。如果我们水平很高，则可能发现系统中有80%的取款功能。而利用包装系统，则不必用散弹猎枪方法进行分析。我们只要查看系统模型，寻

找取款功能包装在哪里。找到账目中的取款功能后，只要在对象中一次性地做出所要求的改变，就万事大吉了。从图1.2可以看出，只需改变账目类。

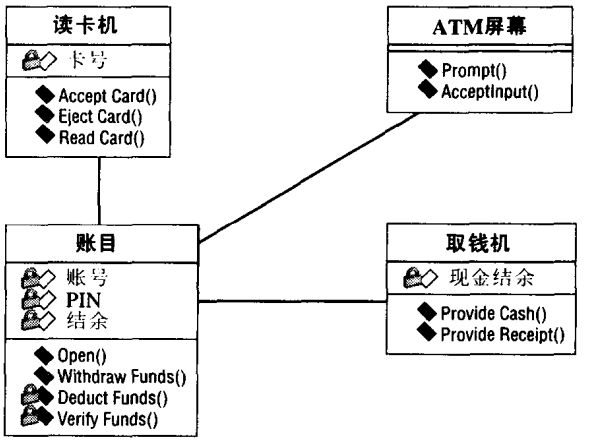

图1.2 包装：银行模型

与包装类似的概念是信息隐藏。信息隐藏就是不向外部显示对象细节的功能。对于一个对象，外部就是对象之外的一切，包括系统其他部分。信息隐藏提供了与包装相同的优势：灵活性。第6章将详细介绍这个概念。

继承

继承是第二个基本面向对象的概念，它与小约翰继承的万贯家财毫无关系，而与你的鼻子像你父亲或母亲的鼻子有关。在面向对象系统中，继承机制可以根据旧对象创建新对象。子对象继承父对象的特性。

自然界中有许多继承的例子。哺乳动物有几百种：狗、猫、人、海豚等等。每种动物都有一些共性和个性，如有毛发、恒温动物、哺乳。用面向对象术语，哺乳动物（mammal对象）有一些共同特性。这个对象是狗、猫、人、海豚等等的父对象。狗对象继承mammal对象的特性，还要有一些狗对象自己的特性，如转圈跑和流口水。面向对象机制借用了自然界中的继承概念，如图1.3，因此可以对系统采用相同的概念。

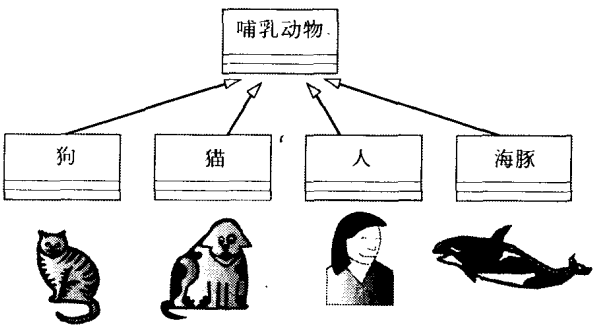

图1.3 继承：自然模型

继承的主要好处之一是易于维护。当产生影响所有哺乳动物的改变时，只要改变父对象，子对象自动继承这个变化。如果所有哺乳动物突然变成冷血动物，只要改变mammal对象，猫、狗、人、海豚和其他子对象自动继承哺乳动物新的冷血特性。

在面向对象系统中，窗口是继承的一个例子。假设一个大系统有125个窗口。一天，客户要求所有窗口显示放弃消息。在没有继承的系统中，我们要吃力地到每个窗口中作出修改。而在面向对象系统中，所有窗口只要从一个父窗口继承。这时只要到父窗口中一次性作出改变即可。所有窗口自动继承这个变化，如图1.4。

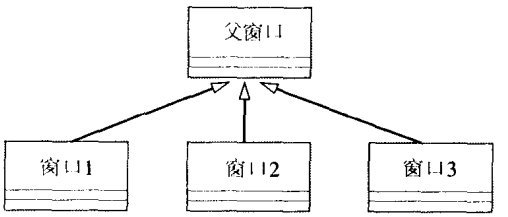

图1.4 继承：窗口模型

在银行系统中，可以对不同类型的账目使用继承。假想银行有四种账目：支票、存款、信用卡和存款证明。这四种账目有一定的相似性。每个账目都有账号、利率、所有者等等。因此可创建父对象account，保存所有账目的共同特性。子对象则在继承特性的基础上增加自己的特性。例如信用账目还有信用额度和最少付款额，存款证明还有到期日期。对父对象的改变影响所有子对象，而对子对象进行改变则不会影响其他子对象和父对象。

多态

面向对象的第三个原则是多态。多态的定义是多种不同形式、阶段或类型发生的事，表示特定功能有多种形式或实现方法。和继承一样，多态在自然界中也有相似的例子。比如让对方说话，人可能说“你好”；狗会汪汪叫；猫会咪咪叫；但也可能就是不理你。

在面向对象系统中，就是特定功能有多种实现方法。例如，我们可能要建立一个绘图系统。用户要画线、圆或者矩形时，系统发出绘图命令。系统中有许多形体，各有不同的绘图功能。因此，用户要画圆时，调用圆对象的绘图命令。利用多态，系统运行时要确定要画的形体类型。而如果没有多态，则绘图功能的代码可能如下所示：

```
Function Shape.drawMe()
{
SWITCH Shape.Type
Case "Circle"
Shape.drawCircle();
Case "Rectangle"
Shape.drawRectangle();
Case "Line"
Shape.drawLine();
END SWITCH
}
```

利用多态，则只要对所画对象调用drawMe()函数，命令如下：

```
Function draw()
{
Shape.drawMe();
}
```

每个形体（线、圆、矩形等等）用drawMe()函数画特定的图。

和面向对象的其他原则一样，多态的好处之一是易于维护。如果应用程序要画一个三角形，在非多态情形中，就要给Shape对象加一个新的drawTriangle()函数，Shape对象的drawMe()函数也要修改成适应新形体的类型。而利用多态，则要创建新的三角形对象，用drawMe()函数绘图。启动绘图操作的draw()函数根本不必改变。

何谓可视化建模

如果要扩建房子，你可能不会买一大堆木材，慢慢钉成所要的样子，而会按照蓝图规划和构造之后再着手扩建。这样的扩建能更持久，而不会因为一场小雨就把一切冲得粉碎。

软件领域中的模型也一样，模型是系统的蓝图。蓝图可以对你的规划进行补充，模型可以帮你规划要建的系统。这就可以保证系统设计良好，要求得到满足，系统能在需求改变时站得住脚。

收集系统需求时，把用户的业务需求映射成开发小组能理解的要求。最终你要利用这些需求产生代码。通过将需求映射为代码，可以保证代码满足这些需求，代码也能方便地回溯成需求。这个过程称为建模。建模过程的结果就是可以跟踪从业务需求、到要求、到模型、到代码的过程及其相反的过程，而不会在这个过程中迷路。

可视化建模将模型中的信息用标准图形元素直观地显示。标准对实现可视化建模的通信功能至关重要。可视化建模的主要目的就是用户、开发人员、分析人员、测试人员、管理人员和其他涉及项目的人员之间的通信。利用非可视信息（文本）也能进行通信，但人类毕竟是视觉动物，通过图形比通过文字更容易理解事情的结构。利用系统的可视化建模，可以在几个层次上显示系统如何工作。我们可以建模用户与系统间的交互，可以建模系统对象间的交互，甚至可以建模系统之间的交互（如果需要）。

建立模型后，可以向所有感兴趣的部门显示这个模型，让他们对模型中的重要信息一目了然。例如，用户可以通过模型直观地看到用户与系统间的交互；分析人员可以看到模型对象间的交互；开发人员可以看到要开发的对象和每个对象的任务；测试人员可以看到对象间的交互并根据这些交互准备测试案例；项目管理人员可以看到整个系统及各部分的交互；而信息总管可以看看高层模型，看看公司的各个系统如何相互交互。总之，可视化建模提供了向各有关部门显示系统计划的强大工具。

图形化标注系统

可视化建模的一个重要问题是用哪种图形标注方法表示系统的各个方面。这个标注方法应能向各有关方面传达意图，否则模型就用处不大。许多人都有自己的可视化建模图注方