Mastering UML with Rational Rose

UML with Rational Rose 从入门到精通

〔美〕 Wendy Boggs 著
Michael Boggs
邱仲潘 等译

- 创建开发项目的UML框图
- 自动生成C++、IDL、JAVA等代码
- 为现有应用程序逆向转出工程代码
- 使用Rational Rose开发Oracle8结构、生成CORBA对象

SYBEX
PHEI
電子工業出版社
Publishing House of Electronics Industry
URL:http://www.phei.com.cn

Mastering UML with Rational Rose

UML with Rational Rose 从入门到精通

〔美〕 Wendy Boggs
Michael Boggs 著

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ATM和订单处理例子介绍如何用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、PowerBuilder和IDL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。

图书在版编目（CIP）数据

UML with Rational Rose从入门到精通/（美）鲍戈斯（Boggs，W.），（美）鲍戈斯（Boggs，M.）著；邱仲潘等译. - 北京：电子工业出版社，2000.3

书名原文：Mastering UML with Rational Rose

ISBN 7-5053-5790-5

Ⅰ. U… Ⅱ. ①鲍… ②鲍… ③邱… Ⅲ. ①UML语言 ②软件工具，Rose Ⅳ. TP312

中国版本图书馆CIP数据核字（2000）第04072号

书　　名：**UML with Rational Rose从入门到精通**

著 作 者：〔美〕Wendy Boggs　Michael Boggs

译　　者：邱仲潘等

责任编辑：陈　宇

印 刷 者：北京天竺颖华印刷厂

装 订 者：三河金马印装有限公司

出版发行：电子工业出版社　URL: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：36.125　字数：920 千字

版　　次：2000年3月第1版　2000年3月第1次印刷

书　　号：ISBN 7-5053-5790-5

TP・3010

定　　价：59.00元

版权贸易合同登记号　图字：01-1999-2980

致　谢

没有各位高人的努力，本书不可能面世。首先感谢Rational公司的开发小组生产了这么优秀的系统设计工具。METEX系统公司的Greg Rusnell提供了本书选配光盘中的材料。感谢Sybex人员的努力，特别是Fred Sloan、Richard Mills、Tracy Brown、Rebecca Rider、Kristen Vanberg-Wolff、Tony Jonick、Jerry Williams、Lisa Reardon和Carrie Bradley。感谢Dan Graydon投入大量时间进行本书的技术审查，Carrie Soltesz进行了编辑审查。最后特别感谢我们的亲人和朋友在此期间的支持。项目完成后，希望能有更多时间与你们相聚。

译 者 序

本书翻译过程中得到了周阳生、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、邓其根等同志完成了本书的录入工作，刘云昌、刘联昌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，我们越来越容易建立和维护系统，保证系统能适应需求的改变。

有许多书籍介绍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 98/98i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、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特性，可以支持C++、Java、Visual Basic和Oracle8的代码产生和逆向转出工程代码。还有PowerBuilder、Forte和其他面向对象语言的插入件，可以进一步扩展Rose的功能。

本书阅读对象

本书适用于UML和Rational Rose的初、中级用户。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。

本书介绍Rose基础：

- 如何生成角色、使用案例和Use Case框图
- 如何生成Sequence和Collaboration框图
- 如何生成类、属性、操作、关系和Class框图
- 如何生成Statechart框图
- 如何生成组件和Component框图
- 如何生成Deployment框图
- 如何用UML与Rose生成系统的完整、详细的蓝图
- 如何用Rose 98i提供的新特性，包括Web Publisher和集成Visual C++
- 如何用Rose生成C++、Java、Visual Basic和PowerBuilder代码
- 如何用Rose产生Oracle8结构
- 如何用Rose产生IDL和DDL
- 如何从C++、Java、Visual Basic和PowerBuilder逆向转出工程代码
- 如何用Rose逆向转出工程Oracle8结构

本书不必按顺序阅读。每章提供Rational Rose一个方面的详细知识。大多数章后面通过练习提供UML与Rose的实践。

如果你不熟悉UML与Rose，可以按顺序阅读第1章到第10章并做完所有练习。这些练习介绍了一个小公司设计订单输入系统的过程。如果你熟悉UML与Rose，则可以用本书作为特定UML与Rose问题的参考手册。

本书组织形式

本书分成下列四个部分。

UML简介

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，介绍其用户界面组成和Rose提供的功能。

Rose基础

第3章到第10章介绍Rose基础，包括生成与更新框图、增加类与类细节和产生报表。

代码生成

第11章到第18章介绍Rose的C++、Java、Visual Basic、PowerBuilder和Oracle8代码生成功能。其中介绍每个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，并介绍如何用Rose直接从模型产生IDL和DDL。

逆向转出工程代码

第19章到第24章介绍Rose的逆向转出工程代码功能。Rose可以从许多不同编程语言逆向转出工程代码。这几章介绍从C++、Java、Visual Basic、PowerBuilder和Oracle8逆向转出工程代码。

关于本书选配的光盘

本书介绍Rose特性时将对一个ATM系统和一个订单处理系统建立一些Rose模型。在本书选配的光盘上，有这两个系统的样本UML模型和可以用Rational Rose产生的代码例子；还有一些样例Rose脚本，是用Rose所带编程语言编写的宏；还有用新的Rose特性Web Publisher产生的HTML页面。从Rational Web站点的链接可以找到Rational的各种产品、UML和对象模型的各种信息。最后，如果用PowerBuilder作为开发工具，则可以用METEX系统公司在本书选配光盘上提供的PowerBuilder插入件产生PowerBuilder代码和逆向转出工程PowerBuilder应用程序。关于本书选配光盘的详细信息，见本书末尾的“本书选配光盘内容”。

与作者联系

尽管我们尽了最大努力，但本书仍然难免会有错误。我们当然希望能找出所有错误，但如果读者发现错误和不一致之处，或其他需要澄清之处，欢迎指出。

Rose和UML的一大妙处是两者都不断改进。但和信息技术行业的其他方面一样，我们很难跟踪所有变化。如果有关于Rose和UML的问题，可以与我们联系。Wendy的地址是wboggs@jps.net，Mike的地址是mboggs@jps.net，或者访问Sybex站点www.sybex.com。

目　录

第1章 UML简介

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

本章介绍UML（统一建模语言，Unified Modeling Language），这是最广泛使用的面向对象系统的建模方法。本章首先介绍可视模型及其如何构造应用程序，然后介绍UML的历史与现状，最后介绍开发过程及模型的作用。

UML由几种不同框图构成，本章简要介绍这些框图，后面的章节还将详细介绍每个框图。

面向对象机制简介

面向对象是软件行业的新术语。各个公司纷纷采用这个新技术，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。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，就可以集成起来，生成新的应用程序。

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。就像砖块可以重复利用盖城墙、盖房子、盖太空飞船，基本面向对象的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要解决这个问题。利用面向对象机制，我们同时关注信息与功能。因此，我们可开发密切关注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（encapsulation）。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，我们有与银行帐目相关的信息，如帐号、结余、客户名、地址、帐号类型、利率和开户日期。我们还有银行帐目的功能：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帐目对象。结果，银行系统对帐目的任何改变就会在帐目对象中实现。它是所有帐目信息与功能的集合。

包装的另一好处是将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，有些水花会溅上岸边和湖外。换句话说，一石击水引起了巨大的波浪。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。

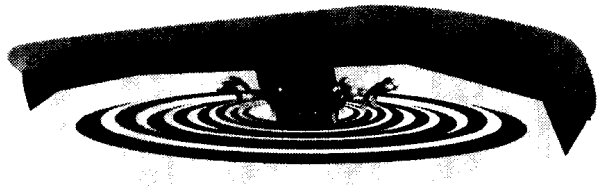

图1.1 包装：湖水模型

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帐号，则可以用信用帐号作为支票帐号的透支额。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。找到帐目中的取款功能后，只要在对象中一次性地作出所要求的改变，就万事大吉了。从图1.2可以看出，只要改变Account类。

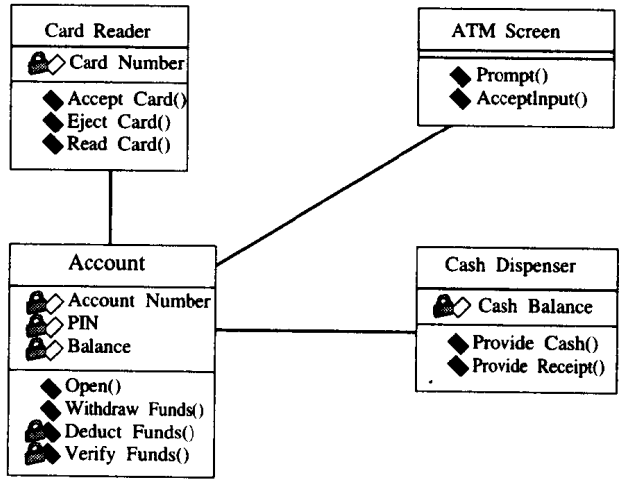

图1.2 包装：银行模型

与包装类似的概念是信息隐藏（information hiding）。信息隐藏就是不向外部显示对象细节。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（inheritance）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的鼻子有关。在面向对象系统中，继承机制可以根据已对象生成新对象。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、热血、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。

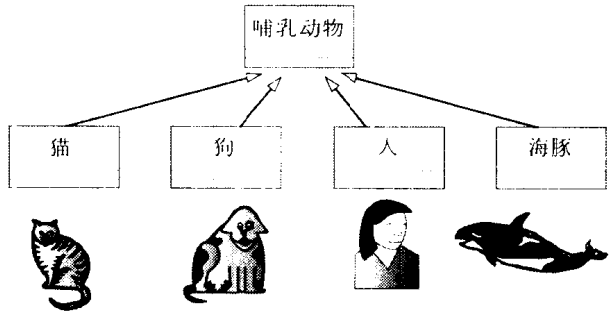

图1.3 继承：自然模型

继承的主要好处之一是易于维护。发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，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改变。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。

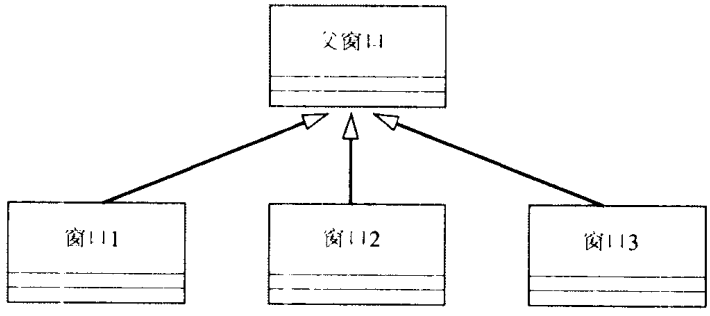

图1.4 继承：窗口模型

在银行系统中，可以对不同类型的帐目使用继承。假想银行有四种帐目：支票、存款、信用卡和存款证明。这四种帐目有一定的相似性。每个帐目都有帐号、利率、所有者等等。因此可生成父对象account，保存所有帐目的共同特性。子对象则在继承特性的基础上增加自己的特性。例如信用帐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（polymorphism）。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，多态也有自然界中的例子。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：

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

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）用自己的drawMe()函数画图。

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，则生成新的三角形对象，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把一切冲得粉碎。

软件中的模型也一样，是系统的蓝图。蓝图可以帮你规划要作的补充，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，系统能在要求改变时站得住脚。

收集系统要求时，把用户的业务需求映射到开发小组能理解的要求。最终你要利用这些要求产生代码。通过将要求映射为代码，可以保证代码满足这些要求，代码也能方便地回溯要求。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，但人类毕竟是百闻不如一见，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方面显示这个模型，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，分析人员可以看到系统对象间的交互，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方面显示系统计划的强大工具。

Booch、OMT与UML

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法，最常用的图注方法有Booch、对象建模技术（OMT）和统一建模语言（UML）。Rational Rose 98i支持这三种方法，但UML是大多数公司采用的标准，是ANSI和OMG等部门采用的标准。

Booch方法是按其发明者Grady Booch命名的，他是Rational软件公司的首席科学家。他著有几本关于可视化建模的需求与好处方面的书籍，并开发了表示模型各个细节的图注方法。例如，图注中把对象画成云图，表示对象几乎无所不包。Booch图注方法还用各种箭头表示对象之间的关系类型。图1.5是用Booch图注方法表示的对象和关系。

OMT（对象建模技术）图注方法来自James Rumbaugh博士，他著有多本关于系统分析与设计的书籍。在《系统分析与设计》一书中，James Rumbaugh博士介绍了模型系统在实际组件（称为对象）中的重要性。他提供的OMT方法有许多追随者，Rational Rose和Select OMT等行业中标准软件建模工具都支持这个方法。OMT使用比Booch更简单的图形表示系统。图1.6是用OMT图注方法表示的对象和关系。

统一建模语言（UML）图注方法是Grady Booch、Dr. James Rumbaugh、Ivar Jacobson、Rebecca Wirfs-Brock、Peter Yourdon和许多其他人员集体智慧的结晶。Jacobson是个学者，他的著作包括在事务包“使用案例”（use cases）中捕获系统要求。第3章将详细介绍使用案