PROGRAMMER TO PROGRAMMER™

Professional UML with Visual Studio .NET

VS.NET UML 建模高级编程

——应用Visio for Enterprise Architects

(美) Andrew Filev
Tony Loton 等著
冯 丽 秦王玉 译

清华大学出版社

VS.NET UML 建模高级编程

——应用 Visio for Enterprise Architects

(美) Andrew Filev
Tony Loton 等著

冯丽 秦王玉 译

清华大学出版社

北 京

内 容 简 介

为了帮助广大.NET 开发人员高效、快捷地设计和创建大型企业级应用程序，本书全面介绍了 Visio 的图形表示和数据库建模等强大功能，并展示了 Visio 与 Visual Studio .NET 集成的优越性。具体内容包括绘制业务组件，从 UML 模块中生成代码，将 Visual Studio .NET 逆向工程为 UML 模块，利用 UML 和 Visio 对项目进行编档，利用 Visio 图设计分布式应用程序、对象角色建模和数据库设计的双向工程等。

本书适合那些熟悉 UML 基本概念并想了解 UML 和 Visio 在 Visual Studio .NET 应用程序开发过程中的作用和用法的.NET 开发人员。

EISBN:1-86100-795-7
Professional UML with Visual Studio .NET
Andrew Filev, Tony Loton et al.

北京市版权局著作权合同登记号　图字：01-2002-0517

图书在版编目(CIP)数据

VS.NET UML 建模高级编程—— 应用 Visio for Enterprise Architects/(美)凡立弗等著；冯丽，秦王玉译.
—北京：清华大学出版社，2003
书名原文：Professional UML with Visual Studio .NET
ISBN 7-302-06898-4
Ⅰ.V…　Ⅱ.①凡…②冯…③秦…　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 058545 号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦
http://www.tup.com.cn　　**邮　　编**：100084
社 总 机：010-62770175　　**客户服务**：010-62776969

组稿编辑：曹康
文稿编辑：李万红
封面设计：康博
版式设计：康博
印 刷 者：清华大学印刷厂
发 行 者：新华书店总店北京发行所
开　　本：185×260　**印张**：18　**字数**：461 千字
版　　次：2003 年 8 月第 1 版　2003 年 8 月第 1 次印刷
书　　号：ISBN 7-302-06898-4/TP・5108
印　　数：1～4000
定　　价：36.00 元

出版者的话

近年来，国内计算机类图书出版业得到了空前的发展，面向初级用户的应用类软件图书铺天盖地，但是真正有深度和内涵的高端图书不多。已经掌握计算机和网络基础知识的人们，尤其是 IT 专业人士迫切需要“阳春白雪”。IT 图书市场呼唤精品！

为了满足这种市场需求，清华大学出版社从世界出版业知名品牌 Wrox 出版公司引进了受到无数 IT 专业人士青睐，被奉为 IT 出版界经典之作的 Professional 系列丛书。这套讲述最新编程技术与开发环境的高级编程丛书，从头到尾都贯穿了 Wrox 出版公司“由程序员为程序员而著(Programmer to Programmer)”的出版理念，每一本书无不是出自软件大师之手。实际上，Wrox 公司的图书作者都是世界顶级 IT 公司(如 Microsoft，IBM，Oracle 以及 HP 等)的资深程序员，他们的作品既深入研究编程机理，传授最新编程技术，又站在程序员的角度，指导程序员拓展编程思路，学习实用开发技巧，从而风靡世界各地，被 IT 专业人士和程序员视为职业生涯中的必读之作。

为了保证该系列丛书的质量，清华大学出版社迅速组织了一批位于 IT 开发领域前沿的专家学者进行翻译，经过编辑人员的进一步加工整理后，现陆续奉献给广大读者。

读者可以从 www.wrox.com 网站下载所需的源代码并获得相关的技术支持。同时，也欢迎广大读者参与 p2p.wrox.com 网站上的在线讨论，与世界各地的编程人员交流读书感受和编程体验。

前　言

对于许多人来说，Visio for Enterprise Architects 似乎是一种带有神秘面纱的绘图工具。与 Visual Studio .NET Enterprise Architect 相结合，它的潜在性能好像更加明显——从设计到代码、再从代码回到设计，可以为开发人员快速开发应用程序提供极大的好处。之所以说其“神秘”，是因为 Visio 的强大功能可能会使用户感到胆怯，但最为重要的是，其用法的许多方面直接面向那些缺乏经验的软件开发人员。

本书的目的是解决这种问题，在此我们重点强调用来开发.NET 应用程序的 Visio 功能，包括：

- UML 图
- 从 UML 图生成代码
- 对源代码执行逆向工程(reverse engineering)，将其转换为 UML 图
- 数据库建模

顺着这条思路，我们将了解到在软件开发生命周期中 Visio 比较常见的一些应用以及 Visio 的特性，它们是每位 Visio 用户几乎都将遇到的内容。

换而言之，读完本书之后，您即可揭开 Visio for Enterprise Architects 的神秘面纱。

本书主要内容

第 1 章先复习主要的 UML 概念、主要的图类型和这些图在软件开发过程中所扮演的角色。如果您完全不了解 UML，这种常识性的介绍将有助于您了解本书的其他部分。

在第 2 章中，我们先对 Visio 稍加进行研究，大致了解一下 Visio 的一般环境。在触及到本书的主要内容——UML 图之前，先浏览了有助于软件开发的其他 Visio 方面，并尝试让您熟悉 Visio，包括 Visio 的页面、形状和连接线。

在第 3 章，介绍如何使用 Visio 进行对象建模，包括为.NET 应用程序定义数据访问基类、定义业务对象基类、从用例中派生业务类、使用抽象类和具体类以及使用顺序图对对象之间的消息流建模。顺着这条思路，我们将了解到 Visio 的许多 UML 图特性，为继续学习下一章作准备。

Visio for Enterprise Architects 可以从现有的 UML 图生成 C#、Visual Basic .NET 或 C++骨架源代码。此外，Visio 还提供了更多选项，使开发人员可以很好地控制此源代码的实现。在第 4 章，我们了解在 Visio 中如何通过 UML 模型生成代码，并介绍可用于生成代码的各种选项，其中包括使用代码模板指定 Visio 生成的源代码的结构。还将了解 UML 到代码的各种映射以及您将在更复杂的模型中遇到的典型情况。

Visual Studio .NET Enterprise Architect 和 Visio for Enterprise Architects 共同提供用于对现

有的 C#、VB.NET 或 C++.NET 源代码执行逆向工程，进而将其转换到 Visio UML 静态结构模型中的工具。在第 5 章，我们将介绍逆向工程这种功能，并介绍逆向工程为什么有用以及如何在 Visual Studio .NET IDE 中对其.NET 源代码执行逆向工程，研究执行了逆向工程的典型 Visio UML 模型的结构，还介绍诸如一般化(继承)和关联之类的重要构造如何进行代码到 UML 的映射。最后，使用反射对.NET 程序集执行逆向工程，为 UML 图提供.NET Framework 基类模型。

在第 6 章，继续介绍 UML 图、生成代码以及从代码生成更多图，并了解 Visio 和 UML 在整个软件开发生命周期中所扮演的角色。实际上，我们将讨论如何使用 Visio 和 UML 将一个典型开发项目不同阶段的工作编写成文档，在本章结束时，您将进一步了解在进行自己的项目中如何使用 Visio 和 UML。

在第 7 章，将介绍如何利用 Visio 帮助您解决常见的设计问题。设计分布式系统是一个从需求分析到模分解(modular breakdown)再到打包和部署策略的迭代过程。但是，分布式系统与非分布式系统的设计过程不同。在本章，将介绍一个.NET Remoting 示例，即一个 Bank 应用程序。我们先概述.NET Remoting，然后了解如何决定应用程序中的哪些类应该是.NET Remoting 类型，如何决定每个.NET 远程类型的激活模式，如何在 Visio 中用图来表示这一过程，应将哪些代码元素组织到一个组件中，以及如何准备组件图和部署图。

第 8 章继续介绍 Visio 与企业开发人员直接相关的另一方面——数据库建模。我们逐步了解数据库建模和对象角色建模(Object Role Modeling，ORM)，并查看 Visio 的 ORM 源图和实体关系源图。随后了解如何根据这些模型生成数据库模式，通过将数据库逆向工程为 ORM 和 ER 模型来进一步调整设计方案，并用所做的修改对数据库进行更新以产生双向数据库工程。

本书读者对象

本书适用于具备以下条件的.NET 开发人员：

- 熟悉 UML 的基本概念
- 想了解如何有效利用 Visio for Enterprise Architects
- 想了解 UML 和 Visio 通常如何为项目开发提供帮助

使用本书的条件

使用本书必须具备以下条件：

- Visual Studio .NET Enterprise Architect Edition
- Visio for Enterprise Architects

因此，可访问上述每一个环境是使用本书的先决条件。

用户支持

我们一贯重视读者的意见，并想知道每位读者对本书的看法，包括读者喜欢和不喜欢的内

容，以及读者希望我们下一次完善的地方。您可以通过发送电子邮件(地址为feedback@wrox.com)来向我们反馈意见。请确保反馈信息提到本书的书名。

如何下载本书的示例代码

当您访问 Wrox 公司站点(地址为 http://www.wrox.com/)时，通过 Find a Book 工具或书名列表，可以方便地定位需要的书目。然后，单击 Code 列中的 Download 超链接，或者单击本书的详细页面中的 Download　Code 超链接，就可以下载相应的示例代码。

从我们的站点上下载的文件都是使用 WinZip 压缩过的文档。保存文件到本地磁盘上的文件夹中后，需要使用一个解压缩程序(例如 WinZip 或 PKUnzip)来解压缩文件。在解压缩文件时，通常将代码解压缩到每一章所在的文件夹中。在解压缩的过程中，应确保解压缩程序(如 WinZip、PKUnzip)被设置为使用原有文件夹名。

勘误表

我们已经尽最大努力确保本书中的文本和代码没有错误，但是错误仍然在所难免。如果您发现本书存在错误，例如拼写错误或不正确的代码段，请反馈信息给我们，我们将不胜感激。勘误表的发送可以节约其他读者学习本书的时间，而且能够帮助我们提供更高质量的信息。您的反馈信息将被检查，如果正确，将被粘贴到本书的勘误页面上，或者在本书的后续版本中使用。

要在我们的站点上找到勘误表，请访问 http://www.wrox.com/，并通过 Advanced Search 或者书名列表轻松定位本书页面。然后，单击 Book Errata 超链接即可，该链接位于本书的详细信息页面中的封面图解下面。

E-Mail 支持

如果您希望直接向详细了解本书的专家咨询本书中的问题，可以发送电子邮件到 support@wrox.com，要求在邮件的主题栏中带上本书的书名和 ISBN 的后 4 位数字。一封典型的电子邮件应包括下面的内容：

- 在主题栏中必须有本书的书名、ISBN 的后 4 位数字和问题所在的页码。
- 邮件正文中应包括读者的名字、联系信息和问题。

我们仅仅需要有用的详细资料，因此将不返回您的无用邮件，以便可节约您和我们的时间。当您发送一个电子邮件信息时，它将经过下面一系列支持：

- 用户支持：首先，您的信息将被递送到我们的用户支持人员手中，并由他们阅读。对于一些被频繁提到的问题将被归档，并将立即回答有关本书或者 Web 站点的任何常见问题。
- 编辑支持：接着，一些有深度的问题将被送到对本书负责的技术编辑手中，他们在程序设计语言或者特定的产品上有着丰富的经验，能够回答相关主题的详细技术问题。
- 作者支持：最后，如果编辑不能回答您的问题(这种情况很少发生)，他们将请求本书的作者。我们将尽量保护作者免受干扰，以便不影响其写作。然而，我们也非常高兴转寄给他们一些特殊的问题。所有 Wrox 公司的作者都为他们的书提供技术支持。作为回

应，他们将发送电子邮件给用户和编辑，进而使所有的读者受益。

Wrox 公司的支持过程仅仅对那些与我们出版的书目内容直接相关的问题提供支持，对于超出常规书目支持的问题，您可以从 http://p2p.wrox.com/论坛的公共列表中获得支持信息。

p2p.wrox.com 站点

为了便于作者和其他人讨论，特将讨论内容加入到 P2P 站点的邮件列表中，而且我们独特的系统将 programmer to programmer(由程序员为程序员而著)的编程理念与邮件列表、论坛、新闻组以及所有其他服务内容(一对一的邮件支持系统除外)相联系。如果您向 P2P 发送一个问题，应该相信它一定会被登录邮件列表的 Wrox 公司作者和其他相关专家所检查到。无论您是在阅读本书，还是在开发自己的应用程序，都可以在 p2p.wrox.com 站点中找到许多对自己有所帮助的邮件列表。特列适合本书的邮件列表是 vs-dotnet 和 UML 列表。

按照下面的步骤可以预订一个邮件列表：

(1) 登录 http://p2p.wrox.com/站点。

(2) 从左边的主菜单栏选择一个适当的类别。

(3) 单击希望加入的邮件列表。

(4) 按照说明订阅并填写自己的邮件地址和密码。

(5) 回复您收到的确认邮件。

(6) 使用预订管理程序加入更多的邮件列表并设置自己的邮件首选项。

本系统提供最佳支持的原因

您可以加入整个邮件列表，也可以只接收每周的邮件摘要。如果您没有时间和工具来接收邮件列表，可以直接查找我们的在线文档。独特的 Lyris 系统可以将一些没有用的垃圾邮件删除，并保护您的电子邮件地址不被侵扰。当存在加入和离开列表、以及任何有关列表的其他常见问题时，请发送邮件到 listsupport@p2p.wrox.com。

目　录

第 1 章　UML 概述 …… 1
1.1　UML 的定义 …… 1
1.2　端到端 UML 建模 …… 4
1.2.1　UML 基本表示法和核心概念 …… 4
1.2.2　进行 UML 填充 …… 13
1.2.3　UML 建模工具 …… 15
1.3　与过程有关的基础知识 …… 16
1.3.1　RUP …… 16
1.3.2　Microsoft Solutions Framework …… 18
1.4　小结 …… 19
1.4.1　建模概述 …… 19
1.4.2　过程概述 …… 20

第 2 章　Visio 快速预览 …… 21
2.1　Visio 背景知识 …… 21
2.2　开始学习 Visio——简单图 …… 21
2.3　普通的 Visio 软件图 …… 30
2.3.1　创建 COM and OLE 图 …… 31
2.3.2　创建 Data Flow 图 …… 35
2.3.3　创建 Enterprise Application 图 …… 37
2.3.4　Windows Interface Diagram …… 38
2.3.5　创建数据库模型图 …… 47
2.4　小结 …… 51

第 3 章　绘制业务对象 …… 52
3.1　业务对象的定义 …… 52
3.1.1　数据建模与对象建模的对比 …… 52
3.1.2　对属性和行为建模 …… 53
3.1.3　构建整体式应用程序 …… 54
3.1.4　构建基于组件的应用程序 …… 54
3.2　使用业务对象的好处 …… 54
3.2.1　灵活性——编写一次即可随处重用 …… 54
3.2.2　数据访问的灵活性——编写一次，更改一次 …… 55
3.2.3　标准化应用程序逻辑——编写一次即在很长时间内不需更改 …… 55

3.2.4 代码位置——编写并查找它 ····· 55
3.2.5 设计复杂的软件 ····· 56
3.3 设计基于组件的应用程序 ····· 56
3.4 业务类和数据访问基类 ····· 56
3.4.1 创建命名空间数据包 ····· 57
3.4.2 创建抽象的数据访问类 ····· 58
3.4.3 创建类(静态结构)图 ····· 59
3.4.4 向类中添加操作 ····· 60
3.4.5 指定操作参数 ····· 64
3.4.6 将操作标记为抽象 ····· 66
3.4.7 向模型中添加.NET 基类 ····· 67
3.4.8 添加 SaveDataSet 操作 ····· 68
3.4.9 创建具体子类 ····· 69
3.4.10 创建业务对象基类 ····· 71
3.5 简单图书馆系统的用例 ····· 79
3.6 对 Check Out Media 用例建模 ····· 80
3.7 从用例中派生类 ····· 81
3.8 创建顺序图 ····· 83
3.8.1 改变绘图页的方向 ····· 83
3.8.2 向顺序图中添加用例文本 ····· 84
3.8.3 添加角色和 UI 占位符 ····· 85
3.8.4 在对象之间添加消息 ····· 88
3.8.5 创建业务对象类 ····· 91
3.8.6 向顺序图中添加 Borrower 对象 ····· 92
3.8.7 向 Borrower 对象中添加消息调用 ····· 93
3.8.8 调整激活形状的大小 ····· 94
3.8.9 检索已借出的媒体 ····· 95
3.8.10 计算滞纳金 ····· 96
3.8.11 显示借方信息 ····· 97
3.8.12 借出媒体 ····· 99
3.8.13 调整顺序图 ····· 100
3.9 小结 ····· 102

第 4 章 通过 Visio 模型生成代码 ····· 103
4.1 代码生成概述 ····· 103
4.2 Visio 中的代码生成 ····· 104
4.3 生成代码 ····· 116
4.3.1 检查错误 ····· 120

4.3.2 用不同语言生成代码 ……120
4.4 代码模板 ……122
4.4.1 使用模板 ……122
4.4.2 XML 注释和代码模板 ……129
4.5 增强模型 ……134
4.6 小结 ……138

第 5 章 逆向工程 ……140
5.1 使用逆向工程的原因 ……140
5.2 从源代码执行逆向工程 ……140
5.2.1 逆向工程快速启动 ……141
5.2.2 逆向工程的关键功能和限制 ……143
5.2.3 逆向工程示例 ……146
5.3 代码到 UML 的映射示例 ……149
5.3.1 一般化 ……149
5.3.2 关联和属性 ……151
5.3.3 操作和特性 ……153
5.3.4 基本类型和值类型 ……155
5.4 无源代码的逆向工程 ……156
5.4.1 运行 RE.NET Lite 逆向工程 ……157
5.4.2 RE.NET Lite 内部结构 ……161
5.4.3 RE.NET Lite 的限制 ……166
5.5 小结 ……167

第 6 章 对项目编档 ……168
6.1 典型的软件开发生命周期 ……168
6.2 UML 和 Visio 在项目中所扮演的角色 ……171
6.2.1 作为文档的 UML ……171
6.2.2 需求开发文档 ……172
6.3 体系结构文档 ……177
6.3.1 使用类图 ……177
6.3.2 使用活动图 ……178
6.3.3 使用组件图 ……180
6.4 详细的设计文档 ……181
6.4.1 使用详细的类图 ……181
6.4.2 使用顺序图 ……182
6.5 编码和实施文档 ……184
6.5.1 编码的需求文档 ……184
6.5.2 编码的体系结构文档 ……184

6.6 测试和质量保证文档 …… 184
6.7 在整个项目中使用 Visio 报表 …… 185
6.7.1 静态结构图报表 …… 188
6.7.2 部署图报表 …… 191
6.7.3 组件报表 …… 192
6.8 小结 …… 193

第 7 章 分布式系统设计 …… 194
7.1 .NET 中基于对象的分布式系统 …… 194
7.2 用于分布式系统的.NET 基础结构 …… 197
7.3 ASP.NET 与.NET Remoting 的比较 …… 200
7.4 Visio 中的准备工作 …… 201
7.5 用于.NET 分布式系统的自定义 UML 类别模板 …… 203
7.6 打包和部署 Bank 应用程序 …… 206
7.7 小结 …… 217

第 8 章 利用 Visio for Enterprise Architects 对数据库建模 …… 218
8.1 设计过程概述 …… 218
8.1.1 数据库建模 …… 218
8.1.2 对象角色建模 …… 219
8.1.3 ORM 的定义 …… 220
8.1.4 Visio 数据项目 …… 224
8.1.5 第 2 步——绘制事实类型 …… 226
8.1.6 CSDP 的第 3 步 …… 231
8.1.7 约束 …… 232
8.1.8 CSDP 的第 4 至第 7 步 …… 236
8.1.9 创建概念、逻辑和物理数据库 …… 241
8.1.10 构建逻辑模型 …… 247
8.2 数据库的逆向工程 …… 257
8.2.1 ER 图的逆向工程 …… 257
8.2.2 ORM 图的逆向工程 …… 266
8.3 小结 …… 269

第1章 UML 概 述

本章的目的是希望您能够通过了解主要的 UML 概念、主要图类型以及那些图在软件开发过程中所起的作用，学会如何设置环境。如果您对 UML 非常陌生，本章将通过非常实用的介绍，帮助您在继续阅读之前对本书的其余部分有一个整体的了解。如果您对 UML 非常了解，本章可作为一份手头参考资料，您可能会从中发现一些您还未了解到的宝贵信息。

总之，我们开始学习的出发点大致相同：都对 UML 表示法(notation)感兴趣，都对相关软件的开发过程有所了解，都倾向于使用.NET 和 Visio for Enterprise Architects 工具。

最后一点非常重要，这也正是编写本书的目的所在。近几年，大部分 UML 文献都着眼于 Java 开发以及像 Rational Rose 这样的建模工具的使用。在本书中，在演示迄今为止已经发展成熟且与 Visual Studio .NET Enterprise Architect 捆绑在一起的 Visio 建模工具时，我们将应用.NET 开发观点。

下面将对统一建模语言(Unified Modeling Language，UML)进行介绍，或者根据您在此方面的背景回顾 UML 的相关内容。

1.1 UML 的定义

谈到 UML 时，我们首先需要建立一个重要的概念：

注意：

统一建模语言是一种表示法；它是可以安排用于描述软件系统设计的图和图元素的集合。UML 既不是过程，也不是由表示法和过程组成的方法。

从理论上来讲，您可以根据要选择的任一过程所指示的步骤来应用该表示法的各个方面，其中所选择的过程包括传统的瀑布法(waterfall)、极限编程(eXtreme Programming)和 RAD(快速应用程序开发)，但人们已经专门开发了多种过程来补充 UML 表示法。您将在本章后面部分阅读到更多有关增补过程的信息。

1. 使用 UML 的原因

这一特殊问题中隐含了一个比较普通的问题，即“为什么要使用形式分析和设计表示法——UML 或其他表示法”，接下来我们先通过一个类比来解释这一问题。

假定您要在一条小溪上架一座桥。您一般会横跨小溪的两岸搭起一块木板，也许您所能做的也只有这些。即使木板承受不了您的重量，最坏的情况也只是沾湿您的双脚而已。

现在假定您要在比较窄的河面上架一座桥。您需要事先做一个计划，估计一下需要用哪些材料(木材、砖块或金属)以及各需要多少。如果需要帮助，应该让帮助您的人知道您要建一座

什么样的桥。

最后，假定您要在非常宽的河面上架一座桥。同样，需要事先作一个计划，与更多的小组成员交流意见。这可能是一项商业提议，将来需要通过行人交费的方式来收回成本，所以您需要与有关权威机构取得联系，按照其认可的健康安全要求加以设计和实施。另外，还需要留下充足的文档资料，以备后人对桥梁的结构进行维护，以使其经久耐用。

在软件环境中，这将意味着随项目规模以及复杂程度的提高，尤其是涉及到的人员的增多，形式设计将越来越重要。基于以上类比和丰富的项目经验，我们可以得出结论，那就是模型设计表示法在以下方面非常重要：

- 根据具体应用设计蓝图
- 估计和规划时间及用料
- 小组之间以及小组内部进行合作和沟通
- 编写项目文档

当然，我们大概都遇到过这样的项目，事先所作的模型设计量很小或者根本没有进行模型设计(与上面列出的前三点相对应)；实际上，除了我们提及的项目之外还有更多项目也是如此。即使在那些情况下，经证实在对最终结果编写文档(上面列出的最后一点)时，UML 表示法的价值仍是无法估量的。虽然这里并不推荐您这样做，但是如果对 UML 的要求仅限于编写最终文档，那么您可以重点阅读第 5 章中有关“逆向工程”的讨论。

回答了上面提出的一般性问题之后，接下来我们重新讨论特殊问题——为什么使用 UML？

UML 已被纳入行业标准，这意味着很容易就能够找到其他了解 UML 的人。上面列出的沟通和文档这两点非常重要。另外，如果您或小组中的其他任何人都不了解 UML，可以查找相关的培训课程或类似于本书的相关资料。

UML 发展简史

我们将统一建模语言作为本书的起点，在前一节已经讨论了它的“语言”(即“表示法”)方面，下一节将研究“建模”方面的问题，这里将涉及到“统一”这一词。在出现 UML 之前是什么样的？UML 的前身是什么？它们又是如何实现统一的？了解 UML 的发展简史后，您就会对这些问题得出明确的答案。

最初，虽然有许多面向对象的“方法”，但是以下三种方法最基本：

- Grady Booch 设计的 Booch 方法
- Jim Rumbaugh 设计的对象建模技术(Object Modeling Technique，OMT)
- Ivar Jacobson 设计的面向对象的软件工程(Object Oriented Software Engineering，也称为 Objectory)

这三种方法中有许多共同的想法，只是对于这些想法所采用的是不同的表示法。您也许还记得在 OMT 类图中各个类是以矩形框表示的，而在 Booch 方法中则是以固定格式的云状图来表示的。另外，每种方法在面向对象的软件开发方面的侧重点不同。例如，Jacobson 所介绍的用例(use case)想法是其他方法都没有提及的。

说明：

简单地说，用例是系统为角色(如用户)所提供的某种功能。例如，字处理应用程序中的一

个用例可能是“运行拼写检查程序”。

将这三种方法统一就是把各种方法中的精华结合起来，对于共同概念采用相同的表示法(UML)，最终形成分析和设计方面的行业标准表示法。如果与您打交道的人中有人打算实现对象建模，那么他们非常幸运，因为他们可以使用 UML。

这种统一过程又是如何及时完成的呢？请查看下面的重大日期：

- 1994 年度 OOPSLA —— Jim Rumbaugh 离开 General Electric(通用电子)加盟到 Grady Booch 的 Rational Software 公司，从而将他们的方法融合起来，并确立了行业标准。
- 1995 年度 OOPSLA —— Booch 和 Rumbaugh 共同出版了统一方法(Unified Method)0.8 版本。Rational Software 收购了 Objectory，于是 Ivar Jacobson 也加盟到该公司。
- 1997 年 1 月 —— Booch、Rumbaugh 和 Jacobson(三个盟友)通过 Rational 联合发表了 UML 1.0 版本的有关提议。
- 1997 年 9 月 —— UML 1.1 版本被对象管理组(Object Management Group，OMG)采纳。

注意：

“对象管理组”以前因 CORBA 标准而出名，现在是一家非盈利性组织 —— 其下有许多成员公司。它鼓励且支持在行业中采用对象技术，并对其加以标准化。欲了解有关 OMG 的更多信息，可访问http://www.omg.org站点。

如果认为“统一建模语言”仅仅是这三位创始人(三位盟友)的成果，那是不正确的，下面我们查看一下相关的记录。UML 中的某些概念建立在其他人早期工作的基础之上 —— 例如，David Harel 在 Statechart 图方面的研究；后来取得的一些进展则来自于 OMG 的其他成员机构的努力；如对象约束语言(Object Constraint Language，OCL)就是由 IBM 设计的。

注意：

OCL 的问世，使我们能够以某种比英语更易理解的语言形式向 UML 模型中添加其他规则。例如，语句“Person.Employer=Person.Manager.Employer”可能比英文“a person and their manager must both work for the same company.”的表达更清楚、更易懂。欲了解有关 OCL 的更多信息，可访问http://www-3.ibm.com/software/ad/library/standards/ocl.html。

编写本书时，UML 规范的版本是 1.4 版，OMG 成员在 2001 年年中就开始着手于 UML 2.0 版本的重大升级。对于规范而言，建模工具 —— 包括 Visio for Enterprise Architects —— 始终滞后于它们所支持的 UML，但这并不是一个大问题，因为下一节要讨论的核心概念现在还并不是十分的成熟和稳定。

编写本书时，本章中所使用的 Visio for Enterprise Architects 版本至少支持到 UML 1.2 版本，支持情况由 Microsoft Visio Help 中的 about error checking in the UML model(关于 UML 模型中的错误检查)一节来确定：

“语义错误检查将基于 UML 1.2 规范中的一系列完善规则自动进行，在检查过程中会及时发现 UML 模型元素中设计方面的错误，并提请您注意。”

1.2 端到端 UML 建模

了解 UML 的用途以及它的发展简史之后，我们再来了解一下它本身的表示法。我们不可能通过一章内容对表示法加以完整阐述，所以对那些更深入的描述，建议您阅读以下相关资料。

- Pierre-Alain Muller 编写的《Instant UML》(Wrox Press，ISBN 1-86100-087-1)。
- Grady Booch、James Rumbaugh 和 Ivar Jacobson 编写的《The Unified Modeling Language User Guide》(Addison Wesley，ISBN 0-201-57168-4)。
- Martin Fowler 和 Kendall Scott 编写的《UML Distilled》(Addison Wesley，ISBN 0-201-65783-X)。

本章所要讲述的是基本表示法和核心概念，掌握了这些基本知识后您就可以继续阅读本书的其余章节。

本书的另一个目的是对众多 UML 课程和图书中常见的一个问题加以阐述。该问题是各种图都以孤立的形式存在，而我们无法从这种形式中获取关于各种图之间关系的任何明确指示。更为糟糕的是不同的图经常是以不同的示例来演示的，而那些示例都是实际要构建的系统所涉及不到的。比如用状态图描述汽车变速箱，而用顺序图描述酒店电梯的运转情况就属于这种情形。

所以在下一节中将只使用一个示例，即 Order Processing 系统，即使您不打算构建这样一个系统，也会从中获得相关信息，从而独立于该示例来构建自己想要的系统。

1.2.1 UML 基本表示法和核心概念

现在我们将始终按照从活动图到部署图这一顺序依次介绍 UML 图，该顺序如下：

- 活动图(Activity Diagram)
- 用例图(Use Case Diagram)
- 顺序图和协作图(Sequence and Collaboration Diagram)
- 状态图(Statechart Diagram)
- 静态结构图(Static Structure Diagram)
- 组件图(Component Diagram)
- 部署图(Deployment Diagram)

每个图都以浅灰色标记，其中显示了 UML 元素的指定名称——这些图的正式说法就是 UML 元模型(metamodel)。

说明：

UML 元模型本身是一个 UML 模型，用来定义构建其他 UML 模型所应遵循的规则。如果您在自己的一个模型中表述“银行与一个或多个账户相关联”，那么在元模型中将表述“某个类可能与其他类相关联”，这种表述体现的是一种更一般的关系。

总的来说，模型元素都会用 Visio EA 术语加以标记，从而避免使用该工具时造成混乱。从历史的角度来说，在其他建模工具中，可能会涉及到另外一种 UML 术语。在本章的结尾部分列出了可供选择的术语。

在本章后面将会看到您所遵循的软件开发过程可能被描述为用例驱动的开发过程，这很明显地说明了我们将从用例图开始进行开发。但对于某些业务过程(business process)来说，这些用例无疑将适用于整个过程，因为有可能模型的建立已由业务分析师事先完成。所以我们将以某个业务过程为起点，并以此为工具，通过演示最适当的图(活动图)来达到我们的目的。

1. 活动图

活动图在 UML 中最接近于流程图，并且它也最接近于业务过程图。图 1-1 是一个活动图样本，其中重要的 UML 元素都已标记出来，下面是对那些元素的描述。

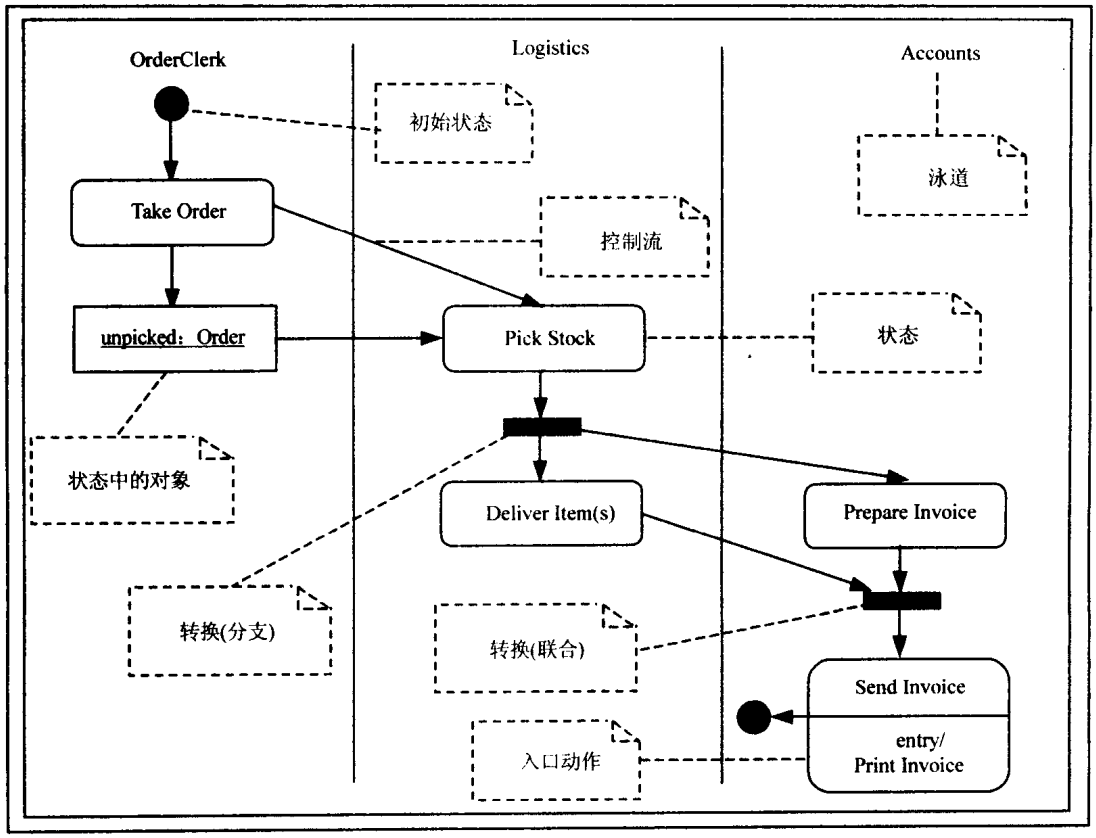

图　1-1

- 初始状态(Initial state)是图开始的位置。
- 控制流(Control flow)显示从一个活动转移到另一个活动的转移控制。
- 状态(State)代表个人或小组完成某项工作的时间段。
- 转换(分支)显示要开始两个或多个并行活动的点。
- 转换(联合)显示两个或多个并行活动必须同步或聚合的点。
- 泳道(Swim lane)允许分配到一组中的某个人或某个小组完成所有活动。
- 入口动作(Entry action)显示活动开始时必须执行哪些操作。
- 状态中的对象(Object in state)显示某一活动中产生或消失的对象，其产生或消失(对象流)由虚线表示。