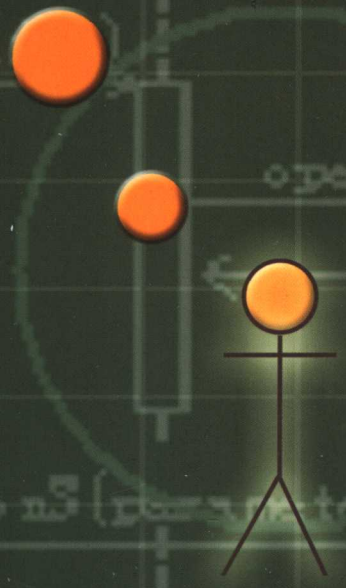
UML建模实例详解

范晓平 编著



清华大学出版社

UML 建模实例详解

范晓平　编著

清华大学出版社
北　京

内 容 简 介

本书介绍了运用面向对象方法及UML(统一建模语言)分析与设计一个办公自动化系统实例的全过程。该系统功能包括发文办理、收文办理、会议管理、档案管理、公告管理、个人助理和系统管理。书中结合该系统实例介绍了设计、建模方法,并提供了完整的系统模型。

本书实用性强,特别适合UML初、中级学习者,可作为计算机项目管理人员、软件开发人员的参考用书,也可以作为高等院校以及相关培训课程的教材。

图书在版编目(CIP)数据

UML建模实例详解/范晓平编著. —北京:清华大学出版社,2005.10
ISBN 7-302-10821-8

Ⅰ.U… Ⅱ.范… 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(2005)第033449号

出 版 者:清华大学出版社
http://www.tup.com.cn
社 总 机:010-62770175
地 址:北京清华大学学研大厦
邮 编:100084
客户服务:010-62776969
责任编辑:付弘宇
封面设计:严高峰
印 刷 者:北京市通州大中印刷厂
装 订 者:三河市化甲屯小学装订二厂
发 行 者:新华书店总店北京发行所
开 本:185×260 **印张**:18.25 **字数**:450千字
版 次:2005年10月第1版 2005年10月第1次印刷
书 号:ISBN 7-302-10821-8/TP·7199
印 数:1~3000
定 价:25.00元

前　言

作为一种建模语言，UML 与其他任何语言一样，是由一套符号组成的。

UML 来源于实践，但不等同于实践，它是对实践活动的提炼和抽象。掌握 UML，说难也不难。正如学习自然语言，如果在母语环境中学外语，并非人人可以学会。许多人花了几年时间，记了许多语法和单词，还是不能用外语与他人交流，可见其难。但是，如果在母语环境中学母语，任何人都可以学会，一个 3 岁小孩就能初步掌握本族的基本语言，可见其不难。

学习 UML 也一样，如果仅从概念到概念，即使能将 UML 硬塞进脑子，也不过是一些抽象的符号，难以用它来自如地建模。但是，如果置身于一个大型系统的建模活动之中，亲身利用 UML 来逐步构造系统模型，将会发现，UML 原来并不难。模型完成之后，留在脑子里的不是一堆符号，而是一个整体的、鲜活的 UML。

本书就是模拟这样一个环境，使读者亲身体验一个办公自动化系统的建模过程，从而在应用中掌握 UML。

书中引用的系统实例的原型取自作者已完成的一个软件项目。为了使实例更具一般性，作者在编写本书的过程中对原型进行了修改，删除了其中的个性部分，补充了共性部分，经过精心调整使之成为了本书中的实例。

全书分为两部分，共 13 章。第 1 部分包括第 1、2 章，简要介绍了面向对象方法和 UML 的基本知识。这部分内容是为办公自动化系统实例建模的必备知识，熟悉这些内容的读者阅读时可以跳过此部分。第 2 部分包括第 3～13 章，第 3 章是对办公自动化系统实例的概括性说明，第 4～13 章详细介绍办公自动化系统实例的建模过程。建模过程规划为 10 个活动：设计用例模型、设计实体类模型、设计接口类模型、设计接口控制类模型、设计用例控制类模型、设计系统类模型、设计窗口结构、设计用户接口原型、设计动态模型和设计数据模型。每一个活动的内容单独构成一章，其中包括活动的方法、活动的过程与活动的产品。这部分内容是全书的重点。

每章的末尾都附有小结。小结概括了本章内容，提示了本章要点。同时，小结还对本章使用的建模方法从横向上与传统的方法进行比较，对本章设计的模型从纵向上与前面已建立的模型进行关联。通过比较，可以加深理解；通过关联，可以获得整体印象。

本书的附录部分介绍了实现模型的建模过程。其中第 2 部分中构建的模型是系统的分析模型和设计模型，它们是系统的逻辑模型。附录中构建的实现模型是系统的物理模型，实现模型是设计模型的自然延伸，是拟建系统的代码框架。本书主要介绍系统的分析模型和设计模型的构建过程，从全书的结构考虑，将实现模型的构建过程放到附录中介绍。

值得一提的是，UML 模型设计是一个迭代过程，要不断循环往复才能完成。某一特定阶段能够获得的信息通常是片面的，模型要随着设计活动的进展做出适当的调整，书中提供的模型只是系统分析和设计阶段的结果。而且，对一个拟建系统而言，即使是最后完成的结果也会因人而异，不同的设计者往往会有不同的结果。设计方案并无“标准答案”，本书中给出的方案只是一孔之见，仅供参考。

本书结构清晰，语言流畅，实用性强。如果您是一位 IT 项目管理人员或软件开发人员，本书可作为您的参考用书；如果您是一位对 UML 有初步了解或完全不了解的初、中级学习者，本书更适合您阅读。本书还可作为高等院校的软件工程、UML、系统分析与设计等相关课程的教材或参考书。

将 UML 及其应用写得通俗易懂而又不失严谨，是本书追求的目标。但是，将这二者完美结合并非易事。限于编者的水平，书中疏漏在所难免，愿与读者共同探讨、进步。

编　者

2005 年 3 月

目　录

第 1 部分　面向对象与 UML

第 2 部分　办公自动化系统建模过程

第1部分　面向对象与UML

工欲善其事，必先利其器。

——《论语·卫灵公》

第1章　理解面向对象

面向对象方法正在逐渐取代传统的方法，日益成为当前软件工程领域的主流方法。

本章简要介绍面向对象的方法，包括面向对象的基本观点和主要概念。这些内容是阅读第2部分各章节的预备知识。

1.1　面向对象的基本观点

什么是“面向对象”？不同的人有不同的看法。软件工程专家Coad和Yourdon指出，“面向对象＝对象＋类＋继承＋通信”。按照这个定义，如果一个计算机软件系统在开发时应用了这四个概念，就可以认为该软件系统是面向对象的。

面向对象内容丰富，归纳起来有以下一些基本观点。

- 客观世界是由对象组成的，任何客观事物都是对象，复杂的对象可以由简单的对象组成。
- 具有相同数据和相同操作的对象可以归并为一个类，对象是类的一个实例。从一个类可以产生多个对象。
- 类可以派生出子类，子类继承父类的全部特性（数据和操作），还可以有自己的新特性。子类与父类形成类的层次结构。
- 对象之间通过消息传递相互联系。类具有封装性，它的数据与操作等对于外界是不可见的，外界只能通过消息请求进行某些操作，获得所需要的服务。

1.2　面向对象的主要概念

为进一步理解面向对象的内涵，下面介绍面向对象的主要概念。

1.2.1　对象

对象（object）从不同的角度看有不同的含义。

从所面对的问题域来说，对象指的是现实世界中的一个事物，是建立在系统模型中的、与目标有关的、有待抽象的事物，它有自己的静态特征和动态特征。对象可以是具体的有形的物体，如人、公文等；也可以是无形的事物或概念，如生产计划、待办事项等；还可以是一种社会活动，如会议等。

从系统的建模和实现而言，对象描述客观事物的一个实体，是构成系统的基本单元，它由一组属性和操作组成。对象的属性是描述对象静态特征的数据项。对象的操作是对象的动态特征的体现，它通常是一个可执行语句或过程，对属性进行操作，从而实现某种服务。

1.2.2 类

类(class)是一组具有相同属性和相同操作的对象的集合。类是对象的抽象，它给出了属于该类的全部的抽象定义，包括类的属性、操作和其他性质。一个具体的对象只是类的一个实例(instance)。

类就像一个对象模板(template)，用它可以产生许多个对象，这些对象具有不同的属性值。类代表的是一个抽象的概念或事物，在客观世界中实际存在的是类的实例，即对象。例如，在办公自动化系统中，“发文”是一个类，指以本机关名义制发的公文。“发文”类的属性有“文号”、“文件名称”、“正文内容”、“起草人”、“收件单位”等，“发文”类的操作可以定义为“草拟”、“修改”、“删除”等。一份具体的“发文”是一个对象，也是“发文”类的一个实例。

类体现了人们认识事物的基本思维方法——分类(classification)，即把具有相同属性和操作的对象划分为一类，用类作为它们的抽象描述。在面向对象的系统分析和设计中，并不需要对每个对象进行说明，而是着重描述代表一批对象的共性的类。

1.2.3 封装

封装(encapsulation)是面向对象方法的一个重要原则。封装是指把对象的属性和操作结合在一起，构成一个独立的对象。它的内部信息对外界是隐蔽的，不允许外界直接存取对象的属性，而只能通过有限的接口与对象发生关系。对于对象的外界而言，只需要知道对象所表现的外部行为，不必了解对象行为的内部实现细节。

封装体现了面向对象方法的“信息隐蔽与局部化”原则。

1.2.4 继承

继承(inheritance)是指子类(派生类、特化类)可以自动拥有父类(基类、泛化类、超类)的全部属性与操作。

在对现实世界建立系统模型时，可以根据事物的共性抽象出一些基本的类(父类)，在此基础上再根据事物的个性抽象出新的类。新的类既有父类的全部属性与操作，又有自己独特的属性或操作。这些新的类就成为父类的子类，或者称为特化类或派生类。父类也称作基类、泛化类或超类。父类与子类的关系是一般与特殊的关系。

通过继承机制，子类可以自动拥有(隐含复制)父类的全部属性与操作。继承机制简化了对现实世界的认识和描述。在定义子类时，不必重复定义那些已在父类中定义过的属性和操作，只要声明自己是某个父类的子类，把精力集中在定义本子类所特有的属性和操作上即可。继承机制提高了软件的可复用性。

例如，类“公文”是一个父类，“发文”、“收文”是它的两个子类。在子类“发文”中，不但拥有“公文”类的全部属性，如“文号”、“文件名称”、“正文内容”等，而且还有自己的属性“签发意见”、“收件单位”等。在子类“收文”中，不但拥有“公文”类的全部属性，而且还有自己的属性“来文单位”、“承办意见”等。子类“发文”与“收文”除都可以使用“公文”类中的操作外，还可以有自己的特定的操作。

继承具有传递性。如果一个类 A 有子类 B，而子类 B 又有其子类 C，那么子类 C 能够继承类 A 的属性和操作。因此，一个类的对象除了具有该类的全部特性外，还具有该类上层的全部父类的一切特性。

在某些情况下，一个类可能需要同时使用两个以上的父类的属性和操作，或者一个类包含在两个以上的父类的交集之中，则该类将从两个以上的父类中继承属性和操作，这称为多继承(multiple inheritance)。

1.2.5　消息

消息(message)是指对象之间在交互中所传送的通信信息。

一般情况下，一个对象向另一个对象发送消息请求某项服务时，接收消息的对象响应该消息，激发所要求的服务操作，并把操作的结果返回给请求服务的对象。

一个消息应当包含以下信息：消息名、接收消息的对象的标识、服务标识、输入信息、回答信息。

面向对象技术的封装机制使对象各自独立、各司其职。对象之间通过消息互相联系，互发消息，响应消息，协同工作，实现系统的各种服务功能。

对于一个顺序系统，不存在并发执行的多个任务，其操作是顺序执行的。在顺序系统中，每个消息都是向对象发出的一个服务请求，它将引起接收消息的对象的一个操作的执行。除了主动对象外，一般的对象都只有接收了消息后才能触发所要求操作的执行，而消息的发送对象则要等待消息的接收对象执行完所要求的操作并返回执行结果后，才继续执行在该消息之后的操作。也就是说，操作是串行的。

对于一个并发系统，情况要复杂得多。在并发系统中，多个控制线程(thread of control)并发执行。消息是在控制线程之间传送的通信信息，它的用途可以是向接收者发出一个服务请求、提交一些数据、发布一个事件信息、传递一个同步控制信息等。

1.2.6　结构与连接

世间万物都不是孤立的、互不相关的，而是彼此之间存在着各种各样的联系。同样，表达客观事物的对象之间也存在着联系。对象之间常见的联系有：分类关系(一般与特殊的

关系)、组成关系(部分与整体的关系)、对象属性之间的静态联系、对象行为的动态联系等。在面向对象方法中通常采用以下的结构和连接来表现这些联系。

1. 分类结构

分类(classification)是对象抽象的基础,分类结构表现的是事物的一般与特殊的关系。在面向对象的术语中常把一般与特殊的关系称为泛化(generalization)与特化(specialization)关系。

例如,类"公文"是"发文"类和"收文"类的泛化,即父类,代表一般性事物;而"发文"类和"收文"类是"公文"类的特化,即子类,代表特殊性事物。子类既继承父类的全部特性,又有自己特有的特性。

一个类是其父类的子类,本身又可以有一个或多个子类。例如,"公文"类可以有父类"文书",而"公文"类本身又有子类"发文"和"收文"。这样形成一个类的层次结构,它是一个树型结构,其中位于最顶层的类为根,它没有父类;位于最底层的类为叶,它没有子类。

如果在分类结构中存在多继承,那么将形成一个类的网状结构。

2. 组装结构

组装结构(composition structure)表示类之间的组成关系,即部分与整体的关系。组装结构体现了面向对象方法的聚合(aggregation)原则。

例如,某局由办公室、业务处、综合处、资料室等类组成,则"办公室"、"业务处"、"综合处"、"资料室"类与"局"类之间就是部分与整体的关系。

一个类可以是另一个类的组成部分,同时本身又可以代表其他类的整体。例如,"业务处"类还可以由"处长"、"职工"等类组成,"业务处"类既是"局"类的组成部分,其本身又是代表"处长"、"职工"等类的整体。因此,组装结构可以表示出复杂的对象层次结构。

3. 实例连接

实例连接(instance connection)表现了对象之间的静态联系,它通过对象的属性来表现对象之间的依赖关系。在面向对象中,常把对象之间的实例连接称为链接(link),而把存在实例连接的类之间的联系称为关联(association)。

实例连接可以有多种类型:一对一的连接、一对多的连接、多对多的连接、零对一的连接、零对多的连接等。多重性说明了在关联中一个类的对象可以对应另一个类的多少个对象。

4. 消息连接

消息连接(message connection)是对象之间的通信联系。当一个对象需要另一个对象的服务时,便向它发出请求服务的消息,接收消息的对象响应消息,触发所要求的服务操作。对象之间的这种联系称为消息连接,它表现了对象行为的动态联系。

消息连接是具有交互的对象之间的一种基本的联系。如果两个对象之间不存在交互行为,那么也没有消息连接。

1.2.7　多态性

多态性(polymorphism)一词源于希腊语。在面向对象中,多态性是指在父类中定义的属性和操作被其子类继承后,可以具有不同的数据类型或表现出不同的行为。例如,父类"汽车"定义了操作"驾驶汽车",其子类"卡车"或"客车"继承其操作"驾驶汽车"以后,变成"驾驶卡车"或"驾驶客车"。"驾驶卡车"、"驾驶客车"都是"驾驶汽车",只不过作用的对象不同,其具体的动作也不同。

在一般与特殊关系的类层次结构中,利用多态性可以使不同层次的类共享一个方法(操作)的名称,而各自有不同的实现。当一个对象接收到一个请求进行某项服务的消息时,将根据该对象所属的类,动态地选用在该类中定义的操作。也就是说,同样的消息可以传给父类对象也可以发送给其子类对象,父类对象和子类对象接到消息后,再根据各自所属的类动态地选用在该类中定义的实现算法。

子类继承父类的属性或操作的名称,而根据子类对象的特性修改属性的数据类型或操作的内容,这称为重载(overloading)。重载是实现多态性的方法之一。

多态性给设计者在结构方面提供了灵活性,减少了信息的冗余,提高了软件的可重用性和可扩充性。

1.3　小　　结

接受面向对象方法,关键要将思维方式从传统的面向过程转变为面向对象。用面向过程的方法看待一个实体世界时,充斥于头脑的往往是一个个功能或过程,而不是一个个实体对象。而用面向对象的方法看待世界时,世界都是由实体构成的。我们生活在一个实体世界中,理应用面向对象的方法看待世界。

应用面向对象方法的最大好处是软件的"复用"。对象的封装和信息隐蔽等特性,既在理论上给出了"复用"的方法,也在程序设计语言和工具方面为"复用"提供了有力的支持。

第2章 认识UML

UML是国际上先进的用于软件分析与设计的统一建模语言，在全世界得到了广泛的支持和应用，已经成为事实上的工业标准。

本章简要介绍UML的基本内容，包括UML概况、模型元素、扩展机制、图和视图。与第1章一样，这些内容也是阅读第2部分各章节的预备知识。

2.1 UML概况

本节概要介绍UML，简述UML的由来、组成及功能，从概貌上认识UML。

2.1.1 UML的由来

UML(Unified Modeling Language)译为统一建模语言。

UML由面向对象方法领域的三位著名学者James Rumbaugh、Grady Booch和Ivar Jacobson提出，并结合其他众多的优秀的软件方法和思想演变而成。UML于1997年被国际对象管理组织(Object Management Group，OMG)接受，发布了UML的标准版。如今，UML已成为公认的最好的分析和设计面向对象软件的标准建模语言。

2.1.2 UML的组成

众所周知，一种自然语言一般由字、词、句、段、篇等部分组成。由字组成词，由词组成句，由句组成段，由段组成篇。

类似地，UML由模型元素、扩展机制、图及视图等部分构成。由模型元素或扩展机制构成图，由图构成视图。

模型元素是构成图的最基本的元素。一个模型元素可以用于多个不同的图中，无论怎样使用，它总是具有相同的含义和相同的符号表示。

扩展机制用于扩展一个模型元素。模型元素就像UML的基本词汇，如果基本词汇不够用，用户可以扩展需要的词汇。在图中可以像使用模型元素一样使用扩展机制。

图由模型元素和扩展机制构成。UML定义了9种不同的图。9种图分为两类，一类是静态图，包括用例图、类图、对象图、组件图和配置图；另一类是动态图，包括序列图、协作图、状态图和活动图。

视图由一个或多个图构成。

在机械制图中，为了表示一个机件体的外部形状和内部构造，需要主视图、俯视图和侧视图。这3个视图是分别从机件体的前面、上面和左侧看过去的投影。UML的视图是从

系统不同的角度对系统的描述，它用来表示系统的各个方面，这些方面从不同的目的出发，为系统建立多个模型。所有模型都反映同一个系统，并具有一致性。如果要为系统建立完整的模型图，只需定义一定数量的视图就可以了。

UML 包括 5 种不同的视图：用例视图(use case view)、设计视图(design view)、过程视图(process view)、实现视图(implementation view)和配置视图(deployment view)。

2.1.3　UML 的功能

从普遍意义上说，UML 是一种语言。语言的基本含义是一套按照特定规则和模式组成的符号系统，能被熟悉该符号系统的人或物使用。自然语言用于熟悉该语言的人群之间的交流，编程语言用于编程人员与计算机的交流。机械制图也是一种语言，它用于工程技术人员与工人之间的交流。UML 作为一种建模语言，则用于系统开发人员之间、开发人员与用户之间的交流。

具体地说，UML 有如下功能。

1. 为软件系统的产出建立可视化模型

UML 符号具有良好的语义，不会引起歧义；UML 为系统提供了图形化的可视模型，使系统的结构变得直观、易于理解；用 UML 为软件系统建立的模型不但有利于交流，还有利于软件维护。

模型是什么？模型是对现实的简化和抽象。对于一个软件系统，模型就是开发人员为系统设计的一组视图。这组视图不仅描述了用户需要的功能，还描述了怎样去实现这些功能。

2. 规约软件系统的产出

UML 定义了在开发软件系统过程中需要做的所有重要的分析、设计和实现决策的规格说明，使建立的模型准确、无歧义并且完整。

3. 构造软件系统的产出

UML 不是可视化的编程语言，但它的模型可以直接对应到各种各样的编程语言。例如，可以由 UML 的模型生成 Java、C++ 、Visual Basic 等语言的代码，甚至还可以生成关系数据库中的表。

从 UML 模型生成编程语言代码的过程称为前向工程(forward engineering)，从编程语言代码生成 UML 模型的过程称为逆向工程(reverse engineering)。

4. 为软件系统的产出建立文档

UML 可以为系统的体系结构及其所有细节建立文档。

2.2 模型元素

模型元素包括活动者、用例、类、对象、消息、接口、包、组件、状态、活动和结点等。除此以外，模型元素与模型元素之间的连接关系也是模型元素，常见的关系有关联、泛化和依赖。

UML 为每一个模型元素规定了独特的图形表示符号，称为图标(icon)。UML 提供了一个标准的、统一的建模符号体系。

下面对模型元素的图标逐一进行介绍。

2.2.1 活动者(actor)

活动者代表与系统交互的人、硬件设备或另一个系统。尽管可以在模型中使用活动者，但活动者不是软件系统的组成部分，活动者存在于系统的外部。活动者的图标如图 2.1 所示。

2.2.2 用例(use case)

在 UML 中，用例规定了系统或部分系统的行为，它描述了系统所执行的动作序列集，并为执行者产生一个可供观察的结果。用例表示了系统的功能，一个用例是系统功能的一个通用描述，系统的用例构成了系统的所有使用功能。

可以将用例应用到整个系统，也可以将用例应用到系统的一部分，如子系统等。一个系统通常需要多个用例来描述系统需求。

用例的图标如图 2.2 所示。

2.2.3 类(class)

类是面向对象系统中最基本的组成元素，用来表示系统中需要处理的事物。类用长方形表示，长方形分成上、中、下 3 个区域，分别标识类的名字、属性和操作，如图 2.3 所示。

图 2.1 活动者的图标

图 2.2 用例的图标

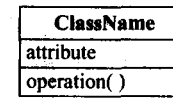


图 2.3 类的图标

类的名字用黑体字书写在长方形的最上面。类的名字最好能够反映类所代表的问题域中的概念，含义要清楚准确，不能含混不清。

1. 类的属性

类的属性书写在类名字的下面。类的属性描述该类的对象所具有的特征。描述属性的语法格式如下，其中放在“[]”中的部分是可选的。

[可见性] 属性名[：类型名][=初始值]

不同属性具有不同的可见性，利用可见性可以控制外部事物对类的属性的操作方式。属性的可见性通常分为 3 种，用以下可见性标记表示。

＋　公有(public)

－　私有(private)

＃　保护(protected)

公有属性能够被系统中的任何其他操作访问；私有属性仅在类内部可见，只有类内部的操作才能访问，同时该属性也不能被其子类访问；保护属性可以被本类或子类的对象访问。

属性与变量一样，也有其类型。属性的类型可以是程序语言能够提供的任何一种数据类型。

属性的标记如图 2.4 所示。

2. 类的操作

类的操作部分书写在类图中长方形的底部。操作用于存取或改变该类的属性值，或执行某个动作，操作描述了该类能做些什么工作。操作通常又称为函数，它是类的一个组成部分，只能作用于该类的对象。属性是类需要处理的数据，操作则描述了处理数据的具体方法。类将属性与操作封装起来，形成一个完整的整体。

描述类的操作的语法格式为：

[可见性] 操作名[(参数表)][：返回值类型] [{属性字符串}]

操作的可见性分为公有和私有两种，其含义与表示方法与属性的公有和私有可见性相同，如图 2.5 所示。

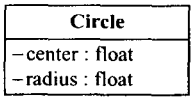


图 2.4　属性的标记

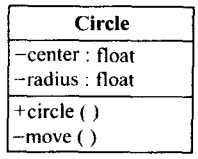


图 2.5　带有属性和操作的类

3. 类的类型

类可以分为 3 种类型：实体类、接口类和控制类。

实体类代表了应用领域的核心内容，例如发文、收文。实体类的作用是用来持久地保存应用程序的实体，同时提供驱动应用程序中大多数的交互服务。可以将实体类与 ER 图中的实体对比理解，但不要将两者相混淆。ER 图中的实体只表示系统中的持久的元素。实体类除表示系统中的持久的元素外，还具有行为特性——操作。实体类是 ER 图中的实体提供的内容的超集。实体类最后要映射到数据库的数据表。