普通高等教育软件工程专业“十二五”规划教材

UML系统建模基础教程

李占波　薛均晓　主编

科学出版社

普通高等教育软件工程专业“十二五”规划教材

UML 系统建模基础教程

主　编　李占波　薛均晓

副主编　刘小燕

科学出版社

北　京

内 容 简 介

本书围绕统一建模语言 UML(Unified Modeling Language)和建模软件 Rational Rose 两大知识模块，介绍了 UML 的基础知识、UML 所包含的各种图形，以及 Rational Rose 软件的使用方法等内容。书中列举了大量实例，并在每章节提供了一定数量的习题，便于读者理解和掌握相关知识。

本书是一本理论知识和实际案例紧密结合的 UML 系统建模实用教程，既可以作为高等院校软件工程相关专业的教材，也可以作为软件开发人员的系统分析与设计参考用书。

图书在版编目(CIP)数据

UML 系统建模基础教程 / 李占波，薛均晓主编. —北京：科学出版社，2013

普通高等教育软件工程专业“十二五”规则教材

ISBN 978-7-03-037570-4

I. ①U… II. ①李… ②薛… III. ①面向对象语言—程序设计—教材 IV. ①TP312

中国版本图书馆 CIP 数据核字(2013)第 110019 号

责任编辑：于海云 / 责任校对：宣 慧

责任印制：闫 磊 / 封面设计：迷底书装

科 学 出 版 社出版

北京东黄城根北街 16 号

邮政编码: 100717

http://www.sciencep.com

骏杰印刷厂 印刷

科学出版社发行 各地新华书店经销

*

2013 年 8 月第 一 版 开本：787×1092 1/16

2013 年 8 月第一次印刷 印张：13 1/2

字数：355 000

定价：29.00 元

(如有印装质量问题，我社负责调换)

普通高等教育软件工程专业“十二五”规划教材

编 委 会

《UML 系统建模基础教程》编委会

主　编　李占波　薛均晓

副主编　刘小燕

编　委　李占波　薛均晓　刘小燕　李庆宾　张宏涛　陈永霞

前　言

统一建模语言 UML(Unified Modeling Language)是一种用于描述、构造软件系统以及商业建模的语言，综合了在大型、复杂系统的建模领域得到认可的优秀的软件工程方法。近 10 年来，在世界范围内，UML 是面向对象技术领域内占主导地位的标准建模语言。

UML 是一种定义良好、易于表达、功能强大且普遍适用的建模语言。它融入了软件工程领域的新思想、新方法和新技术。它的作用域不限于支持面向对象的分析与设计，还支持从需求分析开始的软件开发的全过程。

本书坚持“理论够用，突出实践”的原则，书中简略介绍了面向对象的基本概念、UML 的基础知识、UML 所包含的各种图形，以及软件开发过程的概念等必要基础知识；重点讲解 UML 的使用方法。全书由一个具体案例贯穿始终，并由案例引入相关的操作和模型创建过程。同时，书中在讲解相关概念时，列举了大量实例，利用这些实例，可以帮助读者更快地掌握 UML 的基本元素和建模技巧，也能让读者学会通过 Rational Rose 开发 UML 的方法。为了帮助读者巩固所学知识，在每章节后还提供了习题。

本书分为 12 章，由李占波(郑州大学)、薛均晓(郑州大学)主编，编写人员有薛均晓、刘小燕(河南理工大学)、张宏涛(郑州大学)、陈永霞(郑州大学)和李庆宾(郑州航空工业管理学院)。李占波教授对本书的定位和总体规划提出了指导性建议，薛均晓编写了第 1 章、第 4 章、第 5 章和第 10 章，刘小燕编写了第 6 章、第 7 章、第 8 章、第 9 章和第 12 章，张宏涛编写了第 2 章，陈永霞编写了第 11 章，李庆宾编写了第 3 章，最后由薛均晓负责全书的统稿工作。

本书在编写过程中得到了科学出版社和编者所在学校的大力支持和帮助，在此表示最诚挚的感谢。由于编者水平有限，编写时间仓促，书中难免有疏漏和欠妥之处，恳请专家和广大读者批评指正。

编　者

2013 年 5 月

目　录

第1章　面向对象概述

面向对象是一种系统开发方法。在面向对象编程中，数据被封装(或绑定)到使用它们的函数中，形成一个整体称为对象，对象之间通过消息相互联系。面向对象建模与设计是使用现实世界的概念模型思考问题的一种方法。对于理解问题、与应用领域专家交流、建模企业级应用、编写文档、设计程序和数据库，面向对象模型都非常有用。

1.1　面向对象的含义

在现实世界中，一个复杂的事物往往是由许多部分组成的。例如，一辆汽车是由发动机、底盘、车身和车轮等部件组成的。当人们生产汽车时，分别设计和制造发动机、底盘、车身和车轮等，最后把它们组装在一起。组装时，各部分之间有一定联系，以便协同工作。

面向对象系统开发的思路和人们在现实世界中处理问题的思路是相似的，是基于现实世界设计与开发软件系统的方式。面向对象技术以对象为基础，使用对象抽象现实世界中的事物，以消息来驱动对象执行处理。与面向过程的系统开发不同，面对对象技术不需要一开始就使用一个主函数概括整个系统的功能，而是从问题域的各个事物入手，逐步构建整个系统。

在程序结构上，常用下面的公式表述面向过程的结构化程序：

面向过程程序=算法+数据结构

算法决定了程序的流程以及函数间的调用关系，也就是函数之间的相互依赖关系。算法和数据结构二者相互独立，分开设计。在实际问题中，有时数据是全局的，很容易超出权限范围修改数据，这意味着对数据的访问是不能控制的，也是不能预测的，如多个函数访问相同的全局数据，因为不能控制访问数据的权限，程序的测试和调试就变得非常困难。另外，面向过程的程序中主函数依赖子函数，子函数又依赖更小的子函数。这种自顶向下的模式使得程序的核心逻辑依赖外延的细节，一个小小的改动有可能带来连锁反应，进而引发依赖关系的一系列变动，这也是过程化程序设计不能很好地处理需求变化，代码重用性差的原因。在实践中，人们慢慢意识到算法和数据是密不可分的，通过使用对象将数据和函数封装(或绑定)在一起，程序中的操作通过对象之间的消息传递机制实现，就可以解决上述问题。于是就形成了面向对象的程序设计：

面向对象程序=(对象+对象+…)+消息

面向对象程序设计的任务包括两个方面：一是决定把哪些数据和函数封装在一起，形成对象；二是考虑怎样向对象传递消息，以完成所需任务。各个对象的操作完成了，系统的整体任务也就完成了。

1.1.1　对象

对象(Object)是面向对象的基本构造单元，是一些变量和方法的集合，用于模拟现实世界中的一些事物模型，如一台计算机、一个人、一本书等；也可以模拟一些虚拟的东西，如一个学号、一个编号、一个院系等。

事实上，对象是对问题域中某些事物的抽象。显然，任何事物都具有两方面特征：一是该事物的静态特征，如某个人的姓名、年龄、联系方式等，这种静态特征通常称为属性；二是该事物的动态特征，如兴趣爱好、学习、上课、体育锻炼等，这种动态特征称为操作。因此，面向对象技术中任何一个对象都应当具有两个基本要素，即属性和操作。一个对象往往是由一组属性和一组操作构成的。

1.1.2 类

为了表示一组事物的本质，人们往往采用抽象方法将众多事物归纳、划分成一些类。例如，常用的名词“人”就是一种抽象表示。因为现实世界只有具体的人，如王安、李晓、张明等。把所有国籍为中国的人归纳为一个整体，称为“中国人”，也是一种抽象。抽象的过程是将有关事物的共性进行归纳、集中的过程。依据抽象的原则进行分类，即忽略事物的非本质特征，只注意那些与当前目标有关的本质特征，从而找出事物的共性，把具有共同性质的事物划分为一类，所得出的抽象概念称为类。

在面向对象的方法中，类的定义如下。

类是具有相同属性和操作的一组对象的集合，它为属于该类的全部对象提供了统一的抽象描述。

事实上，类与对象的关系如同模具和铸件之间的关系。类是对象的抽象定义，或者说是对象的模板；对象是类的实例，或者说是类的具体表现形式。

1.1.3 消息

在面向对象系统中，要实现对象之间的通信和任务传递，采用的方法是消息传递。由于在面向对象系统中，各个对象各司其职，相互独立，要使得对象不是孤立存在的，就需要通过消息传递来使它们之间发生相互作用。通过对象之间互发消息，响应消息，协同工作，进而实现系统的各种服务功能。

消息通常由消息的发送对象、消息的接收对象、消息传递方式、消息内容(参数)、消息的返回等五部分内容组成。消息只告诉接收对象需要完成什么操作，但并不指示接收对象如何完成操作。消息接收者接收能识别的消息，并独立决定采用什么方法完成所需的操作。

1.1.4 封装

封装是指把一个对象的部分属性和功能对外界屏蔽，也就是说从外界是看不到，甚至是不可知的。例如，计算机里面有各种电子元件和电路板，但这些在外面是看不到的，在计算机的外面仅保留用户需要用到的各种按键，即计算机的外部接口。人们使用计算机的时候也不必了解计算机内部的结构和工作原理，只需要知道按键的作用，通过各个外部的按键让计算机执行相应的操作即可。

封装是一种防止相互干扰的方式。所谓封装，包含两层含义：一是将有关数据和操作封装在一个对象中，形成一个整体，各个对象之间相互对立，互不侵扰；二是将对象中某些属性和操作设置为私有，对外界隐蔽，同时保留少量接口，以便与外界联系，接收外界的消息。这样做既有利于数据安全，防止无关人员修改数据，又可以大大降低人们操作对象的复杂度。使用对象的人完全可以不必知道对象的内部细节，只需了解其外部功能即可自如操作对象。例如，如果要从朋友那里借钱，我们不会自己去翻钱包，也不关心朋友的钱是如何赚来的，

只关心朋友是否借，能借多少。

1.1.5 继承

继承是指子类可以自动拥有其父类的全部属性和操作。继承可以指定类从父类中获取特性，同时添加自己的独特特性。例如，已经建立了一个“员工”类，又想另外建立一个“财务人员”类和“销售人员”类，而后两个类与“员工”类的内容基本相同，只是在“员工”类的基础上增加一些属性和操作，显然，不必重新设计一个新类，只需在“员工”类的基础上添加部分新内容。继承简化了对现实世界的描述工作，大大提高了软件的重用性。

1.1.6 多态

同一条消息被不同的对象接收到时可能产生完全不同行为，这就是多态性。多态性支持“同一接口，多种方法”的面向对象原则，使高层代码只写一次而在低层可以多次复用。

实际上，在现实生活中可以看到许多多态性的例子。例如，学校发布一条消息：8 月 25 日新学期开学。不同的对象接收到该条消息后会作出不同的反应：学生要准备好开学上课的必需物品，教师要备好课，教室管理人员要打扫干净教室，准备好教学设施和仪器，宿舍管理人员要整理好宿舍等。显然，对于同一条消息，不同的接收对象作出了不同的反应，这就是多态性。可以设想，如果没有多态性，那么学校就要分别给学生、教师、教室管理人员和宿舍管理人员等许多不同对象分别发开学通知，分别告知需要做的具体工作，显然这是一件非常复杂的事情。有了多态性，学校在发消息时，不必逐一考虑各种类型人员的特点，而不断发送各种消息，只需要发送一条消息，各种类型人员就可以根据学校事先安排的工作机制有条不紊地工作。

从编程角度看，多态提升了代码的可扩展性。编程人员利用多态性，可以在少量修改甚至不修改原有代码的基础上，轻松加入新的功能，使代码更加健壮，易于维护。

1.2 面向对象的有效性

1.2.1 面向过程方法的困难

面向过程方法认为客观世界是由一个个相互关联的小系统组成的，各个小系统依据严密的逻辑组成，环环相扣，井然有序。面向过程方法还认为每个小系统都有着明确的开始和明确的结束，开始和结束之间有着严谨的因果关系。只要将这个小系统中的每个步骤和影响这个小系统走向的所有因素都分析出来，就能完全定义这个系统的行为。所以如果要分析问题，并用计算机来解决问题，首要的工作是将过程描绘出来，把因果关系都定义出来；再通过结构化的设计方法，将过程进行细化，形成可以控制的、范围较小的部分。通常，面向过程的分析方法是找到过程的起点，然后顺藤摸瓜，分析每个部分，直至到达过程的终点。这个过程中的每一部分都是过程链上不可分割的一环。事实上，面向过程方法是一种“自顶向下，逐步细分”的解决问题方法。

在面向过程方法中，计算机解决问题的过程中每一步都会产生、修改或读取一部分数据。每个环节完成后，数据将顺着过程链传递到下一部分。当需要的最终结果在数据中反映出来，即达到预期状态的时候，过程就结束了。显然，数据对于问题的解决至关重要。为了更好地

管理数据，不至于让系统运行紊乱，人们通过定义主键、外键等手段将数据之间的关系描绘出来，结构化地组织它们。然而随着需求越来越复杂，系统越来越庞大，功能点越来越多，一份数据经常被多个过程共享，这些过程对同一份数据的创建和读取要求越来越复杂和多样，经常出现相矛盾的数据需求，因此分析和设计也变得越来越困难。同时，这种步步分析的过程分析方法要求过程中的每一步都是预设好的，有着严谨的因果关系。然而，客观世界从来都不是一成不变的，尤其到了信息时代，外部世界无时无刻不在发生着变化，系统所依赖的因果关系变得越来越脆弱。显然，客观世界的复杂性和频繁变革已经不是面向过程可以轻易应付得了。

1.2.2 面向对象方法的有效性

不同于面向过程方法，面向对象程序设计是一种自下而上的程序设计方法，往往从问题的一部分着手，一点一点地构建出整个程序。面向对象设计以数据为中心，类作为表现数据的工具，成为划分程序的基本单位。面向对象是把构成问题事务分解成各个对象，建立对象不是为了完成一个步骤，而是为了描叙某个事物在整个解决问题的步骤中的行为。

使用计算机解决问题首先要对现实世界进行抽象描述，现实世界中的情况非常复杂，由独立的事物组成，每个事物按照最简单的独立方式对应事件，但是在众多大型事物即刻交互的时候，很难进行预测。这种任务使用面向过程方法难以编程。因为使用面向过程方法基于如下假设：程序结构控制执行流程。所以对于使用过程程序处理上述任务，必须有独立的例程测试或响应为数众多的变化条件。该问题的解决方案是按照问题自身的相似方式构建程序，作为独立的软件事物集合，每个元素都是待抽象的现实世界系统的一个对象。这样就解决了应用域模型和软件域模型之间的冲突，从而将劣势转换为优势。

另一方面，图形用户界面(GUI)在 20 世纪 80 年代和 90 年代的快速普及对现代开发方法提出了特殊的困难。GUI 引入了之前将仿真编程引入主流商业应用中遇到的问题。因为展示给 GUI 用户的是计算机显示屏上高度视觉化的界面，一次性提供很多选项操作，每一种选项都可以通过单击鼠标实现。通过下拉菜单、列表框和其他对话框技术，很多其他的选项也可以通过两次或三次单击鼠标实现。界面开发者自然会探索新技术带来的机遇。结果是，现在系统设计者几乎不可能预测用户通过系统界面执行的任何可能的任务。这意味着计算机应用程序现在很难以过程方式进行设计或控制。面向对象为设计软件提供了自然而然的方法，软件的每一个组件都提供了明确的服务，而这些服务可以被系统的其他部分通过任务序列或控制流独立使用。

在面向对象系统中，信息隐藏表明类有两种定义。外在地，类可以根据接口定义。其他的对象(以及程序员)只需要知道类对象能提供的服务，以及用于请求服务的签名即可。内在地，类可以根据知道的和完成的事情定义，但是只有类的对象(以及程序员)需要知道内部定义的详情。遵循以下思路：面向对象的系统可以被构建，以便每一部分的实现基本上独立于其他部分的实现。这就是模块化的思想，并且有助于解决信息系统开发中最棘手的一些问题。在第 2 章，介绍这些问题包括在开发过程期间和之后需求的改变。模块化的方法可以提供多种方式解决这些问题。按照模块化方式构建的系统易于维护，子系统的改变很可能会对系统的其他部分产生不可预测的影响。基于同样的原因，更新模块化的系统也更加简单。只要替换之前的模块，根据规范采用新的模块，就不会对其他模块产生影响。构建可靠的系统更加容易。子系统可以被单独完整地测试多次，而在之后系统集成的时候，需要解决的问题就会

少得多。模块化系统可以被开发为小型可管理的增量。假定每一个增量被设计用来提供有用且前后一致的功能包，就可以依次进行部署。

1.3 面向对象项目开发

1.3.1 历史回顾

针对日趋复杂的软件需求的挑战，软件业界发展出了面向对象的软件开发模式。目前作为针对“软件危机”的最佳对策，面向对象技术已经引起人们的普遍关注。最初被多数人看成只是一种不切实际的方法和满足一时好奇心的研究，现在得到了人们近乎狂热的追求。许多编程语言都推出了支持面向对象的新版本。大量的面向对象的开发方法被提出来。关于面向对象的会议、学术研讨班和课程极受欢迎。无数专业的学术期刊都为这一话题开辟了专门的版面。一些软件开发合同甚至也指明了必须使用面向对象的技术和语言。面向对象的软件开发对于20世纪90年代，就像结构化的软件开发对于20世纪70年代那样让人着迷，而且面向对象的发展势头还在日益加速。

如“对象”和“对象的属性”这样的概念，可以追溯到1950年初。它们首先出现在关于人工智能的早期著作中。然而，面向对象的实际发展却始于1966年。当时Kisten Nygaard和Ole-Johan Dahl开发了具有更高级抽象机制的Simula语言。Simula提供了比子程序更高一级的抽象和封装；为仿真一个实际问题，引入了数据抽象和类的概念。大约在同一时期，Alan Kay正在尤他大学的一台个人计算机上努力工作，他希望能实现图形化和模拟仿真。尽管由于软硬件的限制，Kay的尝试没有成功，但他的这些想法并没有丢失。20世纪70年代初，他加入了Palo Alto研究中心(PARC)，再次将这些想法付诸实践。

在PARC，他所在的研究小组坚信计算机技术是改善人与人、人与机器之间通信渠道的关键。在这信念的支持下，并吸取了Simula的类的概念，他们开发出Smalltalk语言；1972年PARC发布了Smalltalk的第一个版本。大约在此时，“面向对象”这一术语正式确定。Smalltalk被认为是第一个真正面向对象的语言。Smalltalk的目标是使软件设计能够以尽可能自动化的单元进行。在Smalltalk中一切都是对象，即某个类的实例。最初的Smalltalk世界中，对象与名词紧紧相连。Smalltalk还支持一个高度交互式的开发环境和原型方法。这一原创性的工作开始并未发表，只是被当成带浓厚试验性质的学术兴趣而已。

Smalltalk-80是PARC的一系列Smalltalk版本的总结，发布于1981年。1981年8月的*BYTE*期刊公布了Smalltalk开发组的重要结果。在该期刊的封面上，一个热气球正从一个孤岛上冉冉升起，标志着PARC的面向对象思想的启航，该是向软件开发界公开发表的时候了。起初，影响只是渐进式的，但很快就跃升到火爆的程度。热气球确实启航了，而且影响深远。早期Smalltalk关于开发环境的研究导致了后来的一系列进展：窗口(window)、图标(icon)、鼠标(mouse)和下拉式窗口环境。Smalltalk语言还影响了20世纪80年代早期和中期的面向对象语言，如Object-C(1986年)、C++(1986年)、Self(1987年)、Eiffl(1987年)、Flavors(1986年)。面向对象的应用领域也被进一步拓宽。对象不仅与名词相连，还包括事件和过程。1980年Grady Booch首先提出面向对象设计(OOD)的概念。然后其他人紧随其后，面向对象分析的技术开始公开发表。1985年，第一个商用面向对象数据库问世。20世纪90年代，面向对象的分析、测试、度量和管理等研究都得到了长足发展。目前，对象技术的前沿课题包括设