统一建模语言UML与对象工程

朱小栋　编著

科学出版社

统一建模语言 UML 与对象工程

朱小栋　编著

科学出版社

北京

内 容 简 介

统一建模语言(UML)是面向对象的软件系统开发全生命周期的良好建模方法。本书系统地介绍了UML的内容，在各章节知识点中引入大量的场景例子，注重提高读者学习本书的趣味性和教师授课的生动性。

全书分为三篇共17章，第一篇包括第1～3章，阐述对象工程的理念，介绍UML的概念和发展历史，以及类与对象的基本概念；第二篇包括第4～14章，依据软件系统开发的生命周期脉络，详细地介绍各种UML图的功能、语法和创建过程；第三篇包括第15～17章，通过两个典型的软件系统案例综合阐述UML在软件系统建模上的应用。

本书可作为高等院校和各类培训机构相关课程的教材与参考书，也可作为相关研究领域科研工作者的参考书，还可作为各类软件开发从业者的参考书。

图书在版编目(CIP)数据

统一建模语言UML与对象工程/朱小栋编著. —北京：科学出版社，2019.8

ISBN 978-7-03-062107-8

Ⅰ. ①统… Ⅱ. ①朱… Ⅲ. ①面向对象语言–程序设计 Ⅳ. ①TP312.8

中国版本图书馆CIP数据核字(2019)第179179号

责任编辑：余 江 张丽花 霍明亮 / 责任校对：彭珍珍

责任印制：张 伟 / 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

北京虎彩文化传播有限公司 印刷

科学出版社发行 各地新华书店经销

*

2019年8月第 一 版 开本：787×1092 1/16

2019年8月第一次印刷 印张：15 3/4

字数：373 000

定价：69.00元

(如有印装质量问题，我社负责调换)

前　言

2018 年恰好是软件工程这一概念提出 50 周年，在软件工程的发展史册中，学术界为软件工程贡献了许多重要的理论和方法，它们在产业界发挥了重要的现实意义。在面向对象的软件系统开发过程中，“磨刀不误砍柴工”的开发理念经久不衰。如果在需求分析上偷工减料，那么我们会浪费很多时间去设计客户不想要的东西；如果在系统设计上压缩时间，那么我们会编写很多无助于解决客户问题的代码。

在软件的需求分析和系统设计阶段，统一建模语言(UML)是良好的建模方法。本书一直倡导对象工程的理念，倡导用面向对象的思想去思考现实世界，用 UML 来具体地建模软件系统和绘制软件系统，解决现实软件系统的分析与设计问题。

本书的知识内容分为三篇：第一篇是基础篇，阐述对象工程的理念，介绍 UML 的发展历史，阐述类与对象的概念。第二篇是对象工程篇，详细地介绍 UML 2.0 版本的各个 UML 图的功能、语法和应用。第三篇是实践篇，通过具体的软件系统案例综合阐述 UML 在软件系统建模上的应用。

本书的主要特色：

(1)注重内容的新颖性。本书在每个章节的示例中融入了大数据时代的新系统、新平台。读者可以学习如何运用对象工程的方法建模这些新系统。

(2)注重本书的适用范围。本书读者群覆盖广泛，包括高等院校和各类培训机构相关课程的师生、相关研究领域科研工作者以及各类软件开发从业者。如果读者学习过高级编程语言 C++、Java 或者 C#中的一种，那么更容易理解本书中的一些概念和术语。即使读者没有接触过任何编程语言，学习本书时也不用担心，因为本书努力用案例和故事的方式进行阐述，增加内容的生动性，让读者掌握面向对象的思想，并学会用 UML 来建模系统。

(3)注重内容的可应用性。本书的每个章节的知识点都用现实的软件系统作为分析案例，最后的电子商务网站系统和微信系统则是在本书的所有知识点之上的综合案例。读者据此可以综合学习 UML 如何在具体领域中进行系统建模。

本书的出版得到了上海市高水平地方高校创新团队建设项目(USST-SYS-01)和上海理工大学 2017 年度“精品本科”系列教材建设项目的资金支持。本书配有电子课件和习题参考答案，选用本书作为教材的教师可与出版社联系。读者也可以从网站 http://cc.usst.edu.cn 查找作者后获取本书相关电子教学资源。

感谢我的研究生顾骏涛、陈洁、罗长利、崔健和徐怡为本书的资料收集整理所付出的努力。感谢我的家人，他们的支持给予我无穷的动力和思路。

本书内容颇多，难免存在不足之处，恳请读者批评指正。

作者联系邮箱：zhuxd1981@163.com。

作　者

2018 年 12 月

目　录

第一篇　基 础 篇

第二篇　对象工程篇

第三篇 实 践 篇

第一篇　基 础 篇

第 1 章　对象工程的理念

1.1　面向对象的软件开发概述

软件开发是根据用户要求建造出软件系统或者系统中软件部分的过程。回顾软件开发的历史，可以知道它经历了从面向过程的软件开发，到面向对象的软件开发两个阶段。面向对象的高级编程语言的出现和演化，推进了面向对象的软件开发技术不断发展。20 世纪 60 年代的 Simula、70 年代的 Smalltalk 语言是早期的面向对象编程语言，后来出现的 C++、Java、Objective-C、C#等在当今面向对象的编程语言市场中占有一席之地[①]。

程序设计是软件开发的一个子过程。在面向对象的程序设计方法出现之前，传统的程序设计方法大都是面向过程的。面向过程的程序设计结构清晰，它在历史上为缓解软件危机做出了贡献。面向过程的程序设计方法是以功能分析为基础的，它强调自顶向下的功能分解，并或多或少地把功能和数据进行了分离。换言之，当采用这种方法开发软件系统时，不管在模型设计中还是在系统实现中，数据和操作都是分开的。C 语言是面向过程的程序开发语言的代表。对用这种设计方法设计出的系统而言，模块独立性较差，模块之间的耦合度较高，对一个模块的修改可能会造成许多其他模块功能上的改变。因此，系统的理解和维护都存在一定的难度。具体而言，面向过程的程序设计方法存在如下问题。

(1) 软件系统是围绕着如何实现一定的功能来进行的，当功能中的静态和动态行为发生变化需要修改时，修改工作颇为困难。因为这类系统的结构是以上层模块必须掌握和控制下层模块的工作为前提的。当底层模块变动时，常常会迫不得已去改变一系列的上层模块，而这种一系列的修改并不是当时变动底层模块的目的。同样，当需要修改上层模块时，新的上层模块也必须了解它的所有下层，修改这样的上层模块变得更加困难。所以，这种结构已经无法适应迅速变化的技术发展。

(2) 程序员对客观世界的认知，与他们的程序设计之间存在着一道鸿沟。在客观世界中，人们的认知可以是从一组实例抽象出的概念或者模型，也可以是将概念具体化为一组实例对象。但在面向过程的程序设计中，很难通过编程实现这样的认知。

---

① TIOBE. TIOBE Index for May 2018. [2018-05-11]. https://www.tiobe.com/tiobe-index.

(3)面向过程的程序设计导致模块间的控制作用只能通过上下之间的调用关系来进行，这样会造成信息传递路径过长、效率低、易受干扰甚至出现错误。如果允许模块之间为进行控制而直接通信，那么结果是系统总体结构混乱，从而难于维护。所以，这种结构无法适应以控制关系为重要特性的软件系统的要求。

(4)用这种方法开发出来的系统往往难以维护，主要因为所有的函数都必须知道数据结构。许多不同的数据类型的数据结构之间只有细微的差别。在这种情况下，函数中常常充满了条件语句，它们与函数的功能毫无关系，只是因为数据结构的不同而不得不使用它们，结果使程序变得非常难读。

(5)自顶向下功能分解的分析方法大大限制和降低了软件的易复用性，导致对同样对象的大量重复性工作，从而降低了开发人员的生产效率。

面向对象程序设计方法是面向过程程序设计方法强有力的补充。面向对象的程序设计方法是一种新型、实用的程序设计方法，它强调数据抽象、易扩充性和代码复用等软件工程原则，特别有利于大型、复杂软件系统的生成。该方法的主要特征在于支持数据抽象、封装和继承等概念。借助数据抽象和封装，可以抽象地定义对象的外部行为而隐蔽其实现细节，从而达到归约和实现的分离，有利于程序的理解、修改和维护，对系统原型速成和有效实现大有帮助；支持继承则可以在原有代码的基础上构造新的软件模块，从而有利于软件的复用。当采用面向对象的程序设计方法开发系统时，系统实际上是由许多对象构成的集合。

面向对象程序设计语言可以直接、充分地支持面向对象程序设计方法，从而成为软件开发的有力工具。在面向对象程序设计语言中，对象由属性(状态)和相关操作(方法)封装而成。对象的行为通过操作展示，外界不可以直接访问其内部属性，操作的实现对用户透明。消息传递是对象间唯一的交互方式，对象的创建和对象中操作的调用通过消息传递来完成。类是对具有相同内部状态和外部行为对象结构的描述，它定义了表示对象状态的实例变量集和表示对象行为的方法集。类是待创建对象的模板，而对象则是类的实例。子类可以继承父类的实例变量和方法，同时可以定义新的变量和方法。对象的封装性降低了对象之间的耦合度，从而使得程序的理解和修改变得容易。类之间的继承机制使得代码可以复用，易于在现有代码的基础上进行延伸拓展，为系统增添新的功能。

总之，面向对象的概念框架的特征包括：①抽象；②封装性；③模块化；④层次性；⑤并发性；⑥类型化；⑦持久性；⑧易复用性；⑨易扩充性；⑩动态绑定等。

面向对象的软件开发方法涉及从面向对象分析(OOA)→面向对象设计(OOD)→面向对象程序设计或编码(OOP)→面向对象测试(OOT)等一系列特定阶段。面向对象设计方法期望获得一种独立于语言的设计描述，以求达到从客观世界中的事物原型到软件系统间的尽可能的平滑过渡。

面向对象的方法把功能和数据看作高度统一，其优点主要包括以下几点。

(1)它更好地诠释了软件度量中“高内聚，低耦合”的评价准则。

(2)它能较好地处理软件的规模和复杂性不断增加所带来的问题。

(3)它更适合于控制关系复杂的系统。

(4)面向对象系统通过对象间的协作来完成任务，因而更容易管理。

(5)它使用各种直接模仿应用域中实体的抽象和对象，从而使得规约和设计更加完整。

(6)它围绕对象和类进行局部化，从而提高了规约、设计和代码的易扩展性、易维护性

和易复用性。

(7)它简化了开发者的工作，提高了软件和文档的质量。

当开发人员正确地使用了面向对象的开发方法时，就能够有效地降低软件开发的成本，提高系统的易扩展性、易维护性、易复用性和易理解性。

1.2 面向对象的软件建模方法

Booch 是面向对象方法最早的倡导者之一，他提出了面向对象软件工程的概念。1991 年，他将以前所进行的面向 Ada 的工作扩展到整个面向对象设计领域。1993 年，Booch 对其先前的方法做了一些改进，使之适合于系统的设计和构造。Booch 在其 OOAda 中提出了面向对象开发的 4 个模型：逻辑视图、物理视图及其相应的静态和动态语义。对逻辑结构的静态视图，OOAda 提供对象图和类图；对逻辑结构的动态视图，OOAda 提供了状态变迁图和交互图；对于物理结构的静态视图，OOAda 提供了模块图和进程图。

Jacobson 于 1994 年提出了面向对象的软件工程(OOSE)方法，该方法的最大特点是面向用例(use case)。OOSE 是由用例模型、域对象模型、分析模型、设计模型、实现模型和测试模型组成的。其中用例模型贯穿于整个开发过程，它驱动所有其他模型的开发。

Rumbaugh 等提出了 OMT 方法。在 OMT 方法中，系统是通过对象模型、功能模型和动态模型来描述的。其中，对象模型用来描述系统中各对象的静态结构以及它们之间的关系；功能模型描述系统实现什么功能(即捕获系统所执行的计算)，它通过数据流图来描述如何由系统的输入值得到输出值。功能模型只能指出可能的功能计算路径，而不能确定哪一条路径会实际发生。动态模型则描述系统在何时实现其功能(控制流)，每个类的动态部分是由状态图来描述的。

Coad 与 Yourdon 提出了 OOA/OOD 方法。一个 OOA 模型由主题层、类及对象层、结构层、属性层和服务层组成。其中，主题层描述系统的划分；类及对象层描述系统中的类及对象；结构层捕获类和对象之间的继承关系及整体-部分关系；属性层描述对象的属性和类及对象之间的关联关系；服务层描述对象所提供的服务(即方法)和对象之间的消息链接。OOD 模型由人机交互(界面)构件、问题域构件、任务管理构件和数据管理构件组成。

Fusion 方法被认为是“第 2 代”开发方法。它是在 OMT 方法、Objectory 方法、形式化方法、CRC 方法和 Booch 方法的基础上开发的。面向复用/再工程以及基于复用/再工程的需求开发是 Fusion 的一大特点。Fusion 方法中用于描述系统设计和分析的图形符号虽然不多，但比较全面。Fusion 方法将开发过程划分为分析、设计和实现 3 个阶段，每个阶段由若干步骤组成，每一步骤的输出都是下一步骤的输入。

Shlaer-Mellor 方法是由 Shlaer 与 Mellor 两人重新修订其先前开发的面向对象系统分析(OOSA)方法后所得到的一种建模方法。Shlaer-Mellor 方法中的 OOA 使用了信息建模、状态建模和进程建模技术。OOD 中使用类图、类结构图、依赖图和继承图对系统进行描述。

Martin 与 Odell 提出的 OOAD 方法的理论基础是逻辑和集合论。尽管面向对象的分析通常被划分成结构(静态)和行为(动态)两部分，但 Martin 与 Odell 的 OOAD 却试图将它们集成在一起。他们认为面向对象分析的基础应该是系统的行为，系统的结构是通过对系统行为的分析而得到的。

1.3 对象工程的概念

我们首先区分基本概念科学、工程、技术以及语言的内涵。中国计算机软件学先驱徐家福教授对这些基本的概念曾做过精确阐述。

他认为：科学是系统化的知识。科学研究的过程就是发现规律，探求真理的过程。科学既有阐述义又有引申义。科学是没有阶级的，科学要想发挥自己的作用，就必须发挥人的主观能动性。一个人活着是为了在认识自然、改造自然，认识社会、改造社会的过程中，起到一个螺丝钉的作用。科学工作者做学问不能停留在坐而论道、著书立说上，而要学以致用，将所学应用到社会发展中。技术既有领域义又有学科义，学习技术就是要学以致用。工程包含项目义和学科义。工程就是将技术应用到某个项目中。语言的种类包括自然语言和人工语言，语言的构成要素包括语法、语义和语用，语言的风范包括命令式和申述式，并强调了语言和言语的异同之处。语言强调语，重在写；言语强调言，重在说。

在此基础上，给出对象工程的定义：对象工程是指将面向对象的软件建模技术应用到具体的领域中，解决该领域的系统建模问题。统一建模语言 UML 是对象工程的软件建模技术。

对象工程隶属于软件工程的范畴。软件工程首次提出是在 1968 年，目前对它的普遍认知是以系统性的、规范化的、可定量的过程化方法去开发和维护软件。当然，软件工程现在也是计算机科学与技术一级学科下面的二级学科，是一门研究用工程化方法构建和维护有效的、实用的和高质量的软件的学科。

1.4 统一建模语言(UML)简介

1.4.1 模型与建模

模型是现实系统的简化。建模是对现实系统进行适当过滤，用适当的表现规则描绘出简洁的模型。通过模型人们可以了解所研究事物的本质，而且在形式上便于人们对其进行分析和处理。人的认知过程是逐渐提升的，用模型来认知事物是认知过程的高级阶段，我们倡导当代大学生，特别是研究生不断地学习用模型来认知现实软件系统。图 1-1 是两个模型示意图。

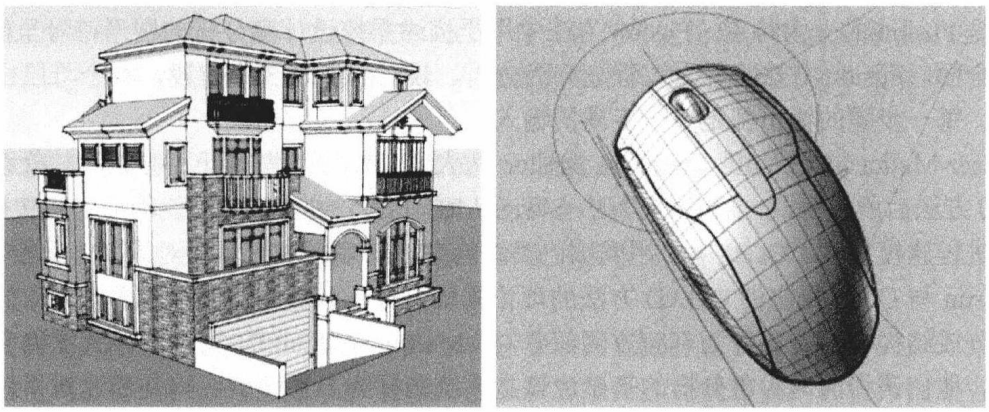

图 1-1 模型示意图

模型是抓住现实系统的主要方面而忽略次要方面的一种抽象，是理解、分析、开发或改造现实系统的一种常用手段。

由于人们对复杂性的认识能力有限，因此系统的设计者在系统设计之初往往无法全面地理解整个系统。此时，人们就需要对系统进行建模。

如同建造一座摩天大楼，我们首先设计并绘制图纸，反复推敲，验证大楼的可靠性。开发一个稳定可靠的软件也是如此，首先建模该软件系统，反复推敲，验证软件的可靠性。

软件设计和建筑设计相似。建模可以使设计者从全局上把握系统及其内部的联系，而不致陷入每个模块的细节之中。模型可使具有复杂关系的信息简单易懂，使人们容易洞察复杂堆砌而成的原始数据背后的规律，并能有效地将系统需求映射到软件结构上。具体而言，模型主要有如下作用。

(1)模型可以促进项目有关人员对系统的理解和交流。模型对问题的理解、项目有关人员(客户、领域专家、分析人员和设计人员等)之间的交流、文档的准备以及程序和数据库的设计等都非常有益。模型可使得人们直接研究一个大型的复杂软件系统。建模能促进人们对需求的理解，从而可得到更清晰的设计，进而得到更易维护的系统。

(2)模型有助于挑选出代价较小的解决方案。当研究一个大型软件系统的模型时，人们可以提出多个实际方案并对它们进行相互比较，然后挑选出一个最好的方案。

(3)模型可以缩短系统的开发周期。模型实质上是过滤掉一些不必要的细节，刻画复杂问题或者结构的必要特性的抽象，它使得问题更容易理解。在有了模型之后，软件系统的开发过程就会变得较快，同时降低系统的开发成本。

1.4.2 UML 的发展历史与应用领域

公认的面向对象建模语言出现于 20 世纪 70 年代中期。从 1989～1994 年，其数量从不到十种增加到了五十多种。在众多的建模语言中，语言的创造者努力地推崇自己的产品，并在实践中不断完善。但是，OO 方法的用户并不了解不同建模语言的优缺点及相互之间的差异，因而很难根据应用特点选择合适的建模语言，于是爆发了一场“方法大战”。在 90 年代中期，一批新方法出现了，其中最引人注目的是 Booch 1993、OOSE 和 OMT-2 等。

概括起来，首先，面对众多的建模语言，用户由于没有能力区别不同语言之间的差别，因此很难找到一种比较适合其应用特点的语言；其次，众多的建模语言实际上各有千秋；最后，虽然不同的建模语言大多类同，但仍存在某些细微的差别，极大地妨碍了用户之间的交流。因此在客观上，极有必要在精心比较不同的建模语言优缺点及总结面向对象技术应用实践的基础上，组织联合设计小组，根据应用需求，取其精华，去其糟粕，求同存异，统一建模语言。

1994 年 10 月，Booch 和 Rumbaugh 开始致力于这一工作。他们首先将 Booch93 和 OMT-2 统一起来，并于 1995 年 10 月发布了第一个公开版本，称为统一方法 UM 0.8(Unified Method)。1995 年秋，OOSE 的创始人 Jacobson 加盟到这一工作。经过 Booch、Rumbaugh 和 Jacobson 三人的共同努力，于 1996 年 6 月和 10 月分别发布了两个新的版本，即 UML 0.9 和 UML 0.91，并将 UM 重新命名为 UML(Unified Modeling Language)。1996 年，一些机构将 UML 作为其商业策略已日趋明显。UML 的开发者得到了来自公众的正面反应，并倡议成立了 UML 成员

协会，以完善、加强和促进 UML 的定义工作。当时的成员有 DEC、HP、I-Logix、 Itellicorp、IBM、ICON Computing、MCI Systemhouse、Microsoft、Oracle、Rational Software、TI 以及 Unisys。这一机构对 UML 1.0(1997 年 1 月)及 UML 1.1(1997 年 11 月)的定义和发布起了重要的促进作用。

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。

首先，UML 融合了 Booch、OMT 和 OOSE 方法中的基本概念，而且这些基本概念与其他面向对象技术中的基本概念大多相同，因而，UML 必然成为这些方法以及其他方法的使用者乐于采用的一种简单一致的建模语言；其次，UML 不仅仅是上述方法的简单汇合，而是在这些方法的基础上广泛征求意见，集众家之长，几经修改而完成的，UML 扩展了现有方法的应用范围；最后，UML 是标准的建模语言，而不是标准的开发过程。尽管 UML 的应用必然以系统的开发过程为背景，但由于不同的组织和不同的应用领域，需要采取不同的开发过程。

作为一种建模语言，UML 的定义包括 UML 语义和 UML 表示法两个部分。

(1) UML 语义。描述基于 UML 的精确元模型定义。元模型为 UML 的所有元素在语法和语义上提供了简单、一致、通用的定义性说明，使开发者能在语义上取得一致，消除了因人而异的最佳表达方法所造成的影响。此外 UML 还支持对元模型的扩展定义。

(2) UML 表示法。定义 UML 符号的表示法，为开发者或开发工具使用这些图形符号和文本语法提供了标准。这些图形符号和文字所表达的是应用级的模型，在语义上它是 UML 元模型的实例。

UML 的目标是以面向对象图的方式来描述任何类型的系统，具有很宽的应用领域。其中最常用的是建立软件系统的模型，但它同样可以用于描述非软件领域的系统，如机械系统、企业机构或业务过程，以及处理复杂数据的信息系统、具有实时要求的工业系统或工业过程等。总之，UML 是一个通用的标准建模语言，可以对任何具有静态结构和动态行为的系统进行建模。此外，UML 适用于系统开发过程中从需求规格描述到系统完成后测试的不同阶段。在需求分析阶段，可以通过用例来捕获用户需求。通过用例建模，描述对系统感兴趣的外部角色及其对系统(用例)的功能要求。分析阶段主要关心问题域中的主要概念(如抽象、类和对象等)和机制，需要识别这些类以及它们相互之间的关系，并用 UML 类图来描述。为实现用例，类之间需要协作，这可以用 UML 动态模型来描述。在分析阶段，只对问题域的对象(现实世界的概念)建模，而不考虑定义软件系统中技术细节的类(如处理用户接口、数据库、通信和并行性等问题的类)。这些技术细节将在设计阶段引入，因此设计阶段为构造阶段提供更详细的规格说明。

编程是一个独立的阶段，其任务是用面向对象编程语言将来自设计阶段的类转换成实际的代码。当用 UML 建立分析和设计模型时，应尽量避免考虑把模型转换成某种特定的编程语言。因为在早期阶段，模型仅仅是理解和分析系统结构的工具，过早考虑编码问题十分不利于建立简单正确的模型。

UML 模型还可作为测试阶段的依据。系统通常需要经过单元测试、集成测试、系统测试和验收测试。不同的测试小组使用不同的 UML 图作为测试依据：单元测试使用类图和类规格说明；集成测试使用部件图和合作图；系统测试使用用例图来验证系统的行为；验收测试

由用户进行，以验证系统测试的结果是否满足在分析阶段确定的需求。

总之，标准建模语言 UML 适用于以面向对象技术来描述任何类型的系统，而且适用于系统开发的不同阶段，从需求规格描述直至系统完成后的测试和维护。

1.4.3 UML 建模工具简介

1. Rational Rose

Rational Rose 是一种基于 UML 的建模工具。在面向对象应用程序开发领域，Rational Rose 是影响其发展的一个重要因素。Rational Rose 自推出以来就受到了业界的瞩目，并一直引领着可视化建模工具的发展。越来越多的软件公司和开发团队开始或者已经采用 Rational Rose，用于大型项目开发的分析、建模与设计等方面。Rational Rose 主要是在开发过程中的各种语义、模块、对象、流程、状态等描述比较好，主要体现在能够从各个方面和角度来分析和设计，使软件的开发蓝图更清晰，内部结构更加明朗，对系统的代码框架生成有很好的支持。

从使用的角度分析，Rational Rose 易于使用，支持使用多种构件和多种语言的复杂系统建模；利用双向工程技术可以实现迭代式开发；团队管理特性支持大型、复杂的项目和队员分散在各个不同地方的大型开发团队。同时，Rational Rose 与微软 Visual Studio 系列工具中 GUI 的完美结合所带来的方便性，使得它成为绝大多数开发人员首选的建模工具；Rational Rose 还是市场上第一个提供 UML 数据建模和 Web 建模支持的工具。此外，Rational Rose 还为其他一些领域提供支持，如用户定制和产品性能改进。

比较经典的 Rational Rose 工具的版本分别是 2003 版和 2007 版(V7.0 版)，目前 IBM 发布的版本包括 2016 版(V8.1 版)。不过，IBM 公司在 Rational Rose 系列产品上的更新动作很慢。其中一个原因是 IBM 公司在 2003 年收购 Rational 之后，发布了全新的 Rational Software Architect 产品(RSA)，它是 IBM 公司开发平台的一部分。这款产品是在 IBM 的 Eclise 基础上建造的。RSA 很好地支持 UML 2.0，相比 Rational Rose 系列，RSA 具有更好的可用性，以及更好地支持逆向过程。

总的来说，IBM 在软件建模方面的持续努力，以及发布系列软件建模工具的目的是支持更新的 UML 版本、更新的编程语言、更新的软件开发方法(如 SOA)、更强大的数据建模等。

2. Enterprise Architect

Enterprise Architect，简称 EA，是澳大利亚 Sparx Systems 公司的旗舰产品。近些年，EA 越来越受到全球系统分析与设计用户的欢迎，有超越 Rational Rose 的势头。EA 覆盖了系统开发的整个周期，除了开发类模型，还包括事务进程分析、使用案例需求、动态模型、组件和布局、系统管理、非功能需求、用户界面设计、测试和维护等。Enterprise Architect 是一个完全的 UML 分析和设计工具，它能完成从需求收集经步骤分析、模型设计到测试和维护的整个软件开发过程。它基于多用户 Windows 平台的图形工具，可以帮助您设计健全可维护的软件。除此，它还包含特性灵活的高品质文档输出，用户指南可以在线获取。

EA 具备源代码的前向和反向工程能力，支持多种通用语言，包括 C++、C#、Java、Delphi、VBNet、Visual Basic 和 PHP，除此，还可以获取免费的 CORBA 和 Python 附加组件。EA 具有令人惊叹的速度，加载超级大的模型只需要几秒钟。通过配备高性能的模型库，EA 可让大型团队分享相同的企业视图。凭借紧密集成的版本控制能力，EA 还可让分布在全世界的团队在共享项目上展开高效的合作。

3. Visio

UML 建模工具 Visio 原来仅仅是一种画图工具，能够用来描述各种图形，如从电路图到房屋结构图，直到 Visio 2000 才开始支持软件分析设计功能，它是目前较好地用图形方式来表达各种商业图形用途的工具(对软件开发中的 UML 支持仅仅是其中很少的一部分)。它跟微软的 Office 产品的能够很好地兼容，能够把图形直接复制或者内嵌到 Word 文档中。Visio 不擅长双向工程，用于软件开发过程的迭代开发有点牵强，这方面更胜一筹的是 Rational Rose 和 EA。

除此之外，UML 的建模工具还有 Visual Paradigm 公司 Visual Paradigm for UML 15.0、Astah 公司的 Astah UML 7.2、MKLab 公司的 StarUML 3，以及它们更新的版本等。在本书中，绘制 UML 图的工具不做限制，读者可以充分地发挥以上这些工具的优势。

1.5 本 章 小 结

本章在面向对象的软件开发与软件建模的概念基础之上，提出对象工程的理念，并倡导使用对象工程这个术语。对象工程隶属于软件工程，是使用面向对象的软件开发技术所进行的软件工程。

统一建模语言 UML 是良好的面向对象的软件建模技术。我们简介 UML 的发展历史，以及 UML 的应用领域。最后对 UML 常用工具 Rational Rose 和 Visio 做了简单的介绍。

习 题

简述题

1. 简述软件开发的两个历史阶段。
2. 简述科学、工程、技术和语言的含义。
3. 简述对象工程的概念。
4. 统一建模语言 UML 的统一、建模和语言的内涵是什么？
5. 面向过程的程序设计方法存在哪些问题?
6. 面向对象的程序设计方法有哪些优点?
7. 简述 UML 的发展历史。
8. 现行的 UML 工具有哪些?

第 2 章　类与对象概述

对象工程学的中心是围绕着对象、类、关系、继承性、封装性和多态性等概念、机制和原理展开的。其中，对象和类是这一方法的核心，关系是连接它们的纽带，封装性增强对象和类的内聚功能，继承性是这一方法的独特贡献，而多态性使得这一方法更加完美。

现实世界纷繁复杂，难以认识和理解。面向对象的思想就是把复杂的事物抽象成类，因此，类是学习对象工程及面向对象建模的核心。类是对象的抽象，对象是类的实例，本章着重介绍类与对象的概念及它们之间的区别联系。掌握好这一章，深刻理解类与对象，有利于后面章节的学习。

2.1　类

2.1.1　类的定义

面向对象思想来源于对现实世界的认知。现实世界缤纷复杂、种类繁多，难以认识和理解。人们学会了把这些错综复杂的事物进行分类，从而使世界变得井井有条。例如，我们将各行各业的人抽象出人的概念，将各式各样的汽车抽象出汽车的概念，将数不胜数的书抽象出书的概念，将五彩斑斓的鲜花抽象出花的概念。人、汽车、书和花都代表着一类事物。每一类事物都有特定的状态，如人的年龄、性别、姓名、职业，汽车的品牌、时速、功率、耗油量、座椅数，书的书名、作者、出版社、ISBN 编号，花的颜色、花瓣形状、花瓣数目等，这些都是在描述事物的状态。每类事物也都有一定的行为，如人可以走、跑、跳，汽车可以启动、行驶、加速、减速、制动，鲜花可以盛开、凋谢。这些不同的状态和行为将各类事物区分开来。

类(class)是具有相同属性和操作的一组对象的组合，即抽象模型中的类描述了一组相似对象的共同特征，为属于该类的全部对象提供了统一的抽象描述。类是对同一组对象的抽象，是人们认识世界的过程中归纳方法的体现。例如，今天见到了某一只狗，明天见到了另一只狗，这样，就逐步形成了对狗这类对象共同特征的认识，如狗是四条腿、爱吃肉的哺乳动物等概念。大多数人一定不会记得见到的第一只狗(对象)是什么样子了，但经过抽象在人脑中形成的狗的概念(类)却能伴随人一生。人类这种抽象能力是认识世界的最重要的特征之一。类并不是描述单个实体的特定行为，而是用来描述同一组中所有对象(一类对象)的公共信息。

2.1.2　类的结构

任何类的结构都由三部分构成：标识、属性和方法。图 2-1 就给出了从具体教师到教师类的抽象，以及教师类的构成。其中，标识给出了类的名称，是对一类对象总的称呼。

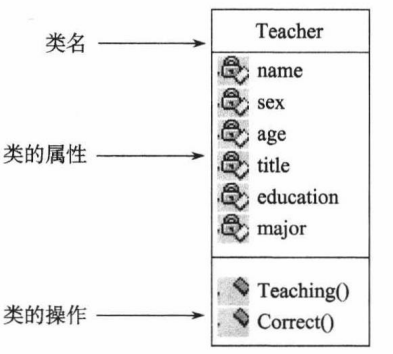

图 2-1　类的结构