高等学校计算机应用规划教材

UML 系统建模基础教程

(第 3 版)

胡荷芬　曹德胜　主　编

陈如意　夏雪星　赵　鑫　副主编

清华大学出版社

北　京

内 容 简 介

本书详细介绍了 UML 系统建模的思想和具体方法，内容包括面向对象设计、UML 通用知识点概述、Rational 统一过程、Rational Rose 的安装和操作、使用 Rose 设计 UML、用例图、类图与对象图、序列图、协作图、活动图、包图、构件图和部署图、状态图，最后以典型案例详解 UML 各种技术的综合应用。

本书采用理论结合案例的方法进行讲解，理论讲述清晰，技术讲解细致，案例丰富。在讲述 UML 案例时，结合了使用比较广泛的 UML 开发工具 Rational Rose。除第 14、15 章以外，每章最后还提供了习题，附录还提供了 6 个课程实验，以供读者更好地了解和掌握 UML 技术。

本书可作为高等学校计算机及相关专业课程的教材，也可作为 UML 初学者和网站开发人员的参考书。

图书在版编目(CIP)数据

UML 系统建模基础教程 / 胡荷芬，曹德胜主编. —3 版. —北京：清华大学出版社，2021.1
高等学校计算机应用规划教材
ISBN 978-7-302-56012-8

Ⅰ. ①U… Ⅱ. ①胡… ②曹… Ⅲ. ①面向对象语言－程序设计－高等学校－教材
Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字(2020)第 121742 号

责任编辑：刘金喜
封面设计：高娟妮
版式设计：妙思品位
责任校对：成凤进
责任印制：吴佳雯

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：北京嘉实印刷有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张：**20.5　　**字　　数：**564 千字
版　　次：2010 年 5 月第 1 版　　2021 年 1 月第 3 版　　**印　　次：**2021 年 1 月第 1 次印刷
定　　价：59.00 元

产品编号：084412-01

前　言

UML(Unified Modeling Language，统一建模语言)是当前比较流行的一种建模语言，可以用于创建各种类型的项目需求、设计及上线文档。Rational Rose 是目前最受业界瞩目的可视化软件开发工具之一，通过 Rational Rose 能用一种统一的方式设计各种项目的 UML 图。

UML 的设计动机是让开发者用清晰和统一的方式完成项目的前期需求和设计文档，而这些需求和设计文档能够让项目的开发变得更加便捷和清晰。随着 UML 建模语言的逐渐深入，其已经获得了广泛的认同，目前已经成为主流项目需求和分析的建模语言。

本书之所以选择 Rational Rose 作为开发 UML 的工具，是因为它不仅提供了绘制所有 UML 图的功能，还完全支持“双向工程”，实现代码和模型的相互转化。

本书包含了 UML 的基础知识、基本元素及使用方法，在讲述 UML 的使用过程中结合了 Rational Rose，以便大家能从中感受到利用 Rational Rose 开发 UML 的便捷性和高效性。同时，在讲述 UML 的元素时，结合了大量的实战案例，并且为了提高学习效率，在除了第 14、15 章以外的每个章节后面还提供了一定数量的习题。

本书共分为 15 章和 1 个附录。书中各章的安排遵循从简单到复杂、由浅入深的思路。由于是基于实际项目，所以本书能让读者更快地掌握 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，是 UML 初学者必备的书籍。

1. 本书内容

第 1 章：面向对象设计。介绍了面向对象思想的基本概念、面向对象的三大要素、面向对象与项目设计和用面向对象思想建立系统模型的方法。

第 2 章：UML 通用知识点概述。介绍了常用的 UML 元素、UML 的通用机制和 UML 的扩展机制。

第 3 章：Rational 统一过程。介绍了统一过程的含义、结构，配置和实现 Rational 统一过程的方法。

第 4 章：Rational Rose 的安装和操作。介绍了 Rational Rose 的安装和操作方法及 Rational Rose 的操作技巧。

第 5 章：使用 Rose 设计 UML。介绍了 Rational Rose 的四种视图模型和 Rational Rose 生成代码的方式。

第 6 章：用例图。介绍了用例图的概念和构成要素、用例的重要元素、用例之间的各种重要关系和使用 Rose 创建用例图的步骤。

第 7 章：类图与对象图。介绍了类图和对象图的基本概念，然后介绍了使用 Rose 创建类图的方式，随后介绍了对象图及用 Rose 创建对象图的方式及案例分析。

第 8 章：序列图。介绍了序列图的基本概念、序列图的组成、序列图中项目的相关概念、使用 Rose 创建序列图的方式及使用 Rose 在实际项目中创建序列图的具体案例。

第 9 章：协作图。介绍了协作图的基本概念、组成协作图的元素、使用 Rose 创建协作图的方式及使用 Rose 在实际项目中创建协作图的具体案例。

第 10 章：活动图。介绍了活动图的基本概念、活动图的组成、使用 Rose 创建活动图的方式及使用 Rose 在实际项目中创建活动图的具体案例。

第 11 章：包图。介绍了包图的基本概念、使用 Rose 创建包图的方式及使用 Rose 在实际项目中创建包图的具体案例。

第 12 章：构件图和部署图。介绍了构件图与部署图的基本概念、使用 Rose 创建构件图和部署图的方式及使用 Rose 在实际项目中创建构件图和部署图的具体案例。

第 13 章：状态图。介绍了状态图的基本概念、构成状态图的元素、状态的组成、使用 Rose 创建状态图的方式及使用 Rose 在实际项目中创建状态图的具体案例。

第 14 章和第 15 章：从需求分析讲起，分别通过网上选课系统、教务管理系统，介绍了创建系统用例图模型、静态模型、动态模型和部署模型的方式。

附录一共提供了 6 个完整的课程实验，课程实验可作为课程结束时课程设计使用，有助于学生从整体上把握系统建模的技术和方法，方便老师课堂教学。

2. 本书特点

(1) 从入门到精通。本书遵循由浅入深、循序渐进的方式，按照知识点的梯度逐渐深入，这样编写的目的是让大家能快速地学习和掌握 UML 技术。

(2) 基于实战案例教学。本书的 UML 相关知识点都配套了实际的案例，能让读者了解到现实项目中 UML 的具体应用。

(3) 面向 Rational Rose。目前有很多种 UML 的开发工具，但 Rational Rose 在业内使用比较广泛，通过学习本书，能让读者了解到 Rational Rose 的常规用法。

(4) 习题配套。为了让读者快速掌握 UML 技术，除第 14、15 章外，每章后面都提供了相关的填空题、选择题和上机题，附录提供了 6 个完整的课程实验。

3. 学时安排

本课程总学时为 42 学时，各章学时分配见下表(供参考)。

学时分配建议表

课 程 内 容	学 时 数		
	合　计	讲　授	实　验
第 1 章　面向对象设计	1	1	
第 2 章　UML 通用知识点概述	2	2	
第 3 章　Rational 统一过程	2	2	
第 4 章　Rational Rose 的安装和操作	3	2	1
第 5 章　使用 Rose 设计 UML	2	2	

(续表)

课 程 内 容	学 时 数		
	合 计	讲 授	实 验
第 6 章 用例图	3	2	1
第 7 章 类图与对象图	3	2	1
第 8 章 序列图	3	2	1
第 9 章 协作图	2	1	1
第 10 章 活动图	3	2	1
第 11 章 包图	2	1	1
第 12 章 构件图和部署图	3	2	1
第 13 章 状态图	3	2	1
第 14 章 网上选课系统	2	1	1
第 15 章 教务管理系统	2	1	1
附录 课程实验	6	2	4
合计	42	27	15

本书不仅可以作为高等学校计算机及相关专业的 UML 课程教材，也可作为自学者及网站开发人员的参考书。

本书免费提供 PPT 教学课件、案例源文件和习题答案，读者可通过扫描下方二维码下载。

资源下载

本书由胡荷芬、曹德胜任主编，陈如意、夏雪星、赵鑫任副主编。参与本书编写工作的还有贾云禄、王坚宁、王魁、许小荣等，在此，编者对他们表示衷心的感谢。

在本书的编写过程中，借鉴了许多现行教材的宝贵经验，在此，谨向这些作者表示诚挚的感谢。由于时间仓促，加之编者水平有限，书中难免有错误或不足之处。敬请广大读者批评指正。

服务邮箱：476371891@qq.com。

编 者

2020 年 1 月

目　录

第 1 章

面向对象设计

面向对象技术现在已经逐渐取代了传统的技术，成为当今计算机软件工程学中的主要开发技术。随着面向对象技术的不断发展，越来越多的软件开发人员加入了它的阵营之中。面向对象技术之所以会被广大的软件开发人员青睐，是由于它作为一种先进的设计和构造软件的技术，使计算机以更符合人的思维方式去解决一系列的编程问题。使用面向对象技术编写的程序极大地提高了代码复用程度和可扩展性，使得编程效率也得到了极大的提高，同时减少了软件维护的代价。

面向对象技术发展的重大成果之一就是出现了统一建模语言(Unified Modeling Language，UML)。UML 是面向对象技术领域内占主导地位的标准建模语言，它统一了过去相互独立的数十种面向对象的建模语言共同存在的局面，通过统一语义和符号表示，系统地对软件工程进行描述和构造，形成了一个统一的、公共的、具有广泛适用性的建模语言。

1.1 面向对象思想的基本概念

面向对象和过去的软件开发技术完全不同，是一种全新的软件开发技术。面向对象的概念从问世到现在，已经发展成为一种非常成熟的编程思想，并且成为软件开发领域的主流技术。面向对象的程序设计(Object Oriented Programming，OOP)旨在创建软件重用代码，具备更好的模拟现实世界环境的能力，这使它被公认为是自上而下编程的最佳选择。它通过给程序中加入扩展语句，把函数“封装”到编程所必需的“对象”中。面向对象的编程语言使得复杂的工作条理清晰，编写容易。说它是一场革命，不是对对象本身而言，而是对它处理工作的能力而言。

1.1.1 面向对象的含义

面向对象技术是一种以对象为基础，以事件或消息来驱动对象执行处理的程序设计技术。从程序设计方法上来讲，面向对象设计是一种自下而上的程序设计方法，它不像面向过程程序设计那样一开始就需要使用一个主函数来概括出整个程序，而是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，使用类作为表现数据的工具，类是划分程序的基本单位，而函数在面向对象设计中成了类的接口。以

数据为中心而不是以功能为中心来描述系统，相对来讲，能使程序更具稳定性。面向对象设计将数据和对数据的操作封装到一起，作为一个整体进行处理，并且采用数据抽象和信息隐藏技术，最终将其抽象成一种新的数据类型——类。

类与类之间的联系及类的重用出现了类的继承、多态等特性。类的集成度越高，越适合大型应用程序的开发。另外，面向对象程序的控制流程运行时是由事件进行驱动的，而不再由预定的顺序进行执行。事件驱动程序的执行围绕消息的产生与处理，靠消息的循环机制来实现。更重要的是，在实际的编程过程中，我们可以利用不断成熟的各种框架，如.NET中的.NET Framework等，迅速地将程序构建起来。面向对象的程序设计方法还能够使程序的结构清晰简单，大大提高了代码的重用性，有效减少了程序的维护量，提高软件的开发效率。

在结构上，面向对象程序设计和结构化程序设计也有很大的不同。结构化程序设计应该确定的是程序的流程，以及函数间的调用关系，即函数间的依赖关系是什么。一个主函数依赖于其子函数，这些子函数又依赖于更小的子函数，而在程序中，越小的函数处理的往往越是细节的实现，这些具体的实现又常常变化。这样的结果，就使程序的核心逻辑依赖于外延的细节，程序中本来应该是比较稳定的核心逻辑，也因为依赖于易变化的部分，而变得不稳定起来，一个细节上的小小改动，也有可能在依赖关系上引发一系列变动。可以说这种依赖关系也是过程式设计不能很好地处理变化的原因之一。而一个合理的依赖关系，应该是倒过来的，由细节的实现依赖于核心逻辑才对。而面向对象程序设计是由类的定义和类的使用两部分组成的，主程序中定义数个对象并规定它们之间消息传递的方式，程序中的一切操作都是通过面向对象的发送消息机制来实现的。对象接收到消息后，启动消息处理函数完成相应的操作。

这里以常见的学生管理系统为例，我们使用结构化程序设计方法的时候，首先在主函数中确定学生管理要做哪些事情，并分别使用函数将这些事情表示出来，使用一个分支选择程序进行选择，然后再将这些函数进行细化实现，确定调用的流程等。而使用面向对象技术来实现学生管理系统，对于该系统中的学生，则先要定义学生的主要属性，如学号、院系等，要对学生做什么操作，如查询学生信息、修改学生信息等，并且把这些当成一个整体进行对待，形成一个类，即学生类。使用这个类，我们可以创建不同的学生实例，也就是创建许多具体的学生模型，每个学生拥有不同的学号，一些学生会在不同的院系。学生类中的数据和操作都是给应用程序共享的，我们可以在学生类的基础上派生出中文系学生类、计算机系学生类、金融系学生类等，这样就可以实现代码的重用。

1.1.2 对象

对象(Object)是面向对象(Object Oriented，OO)系统的基本构造块，是一些相关变量和方法的软件集。对象经常用于建立现实世界中一些对象的模型。对象是理解面向对象技术的关键。

万物皆是对象。我们可以看一看现实生活中的对象，如手机、计算机和打印机等。对象可以是某种可为人感知的事物，也可以是思维、感觉或动作所能作用的物质或精神体。

“某种可为人感知的事物”所指的便是我们熟悉的“对象”，它是可以看到和感知到的“东西”，而且可以占据一定事物的空间。我们以学生管理系统为例，理解学生管理系统中围绕学生管理这个概念中应该有哪些物理对象。

围绕学生管理概念可以列举如下一些物理对象。

- 被管理的信息所属的对象学生。
- 对学生信息进行管理的管理员。
- 对学生信息有权进行查询的校方人员。
- 管理信息的计算机及需要在计算机中存储的学生信息。

我们可以在学生管理系统中找到很多对象，但是它们并不都是所要创建的学生管理系统必需的，该内容将在后面使用用例进行需求分析的章节中进行详细的讲解。

“思维、感觉或动作所能作用的物质或精神体”就是我们所说的“概念性对象”，以学生管理系统为例，可以列举出如下一些概念性对象。

- 学生所在的院系。
- 学生的学号。
- 学生的班级。
- 学生的成绩。

我们可以在学生管理系统中列举出很多这样的概念性对象，这些对象是人们不能看到的、听到的，但是在描述抽象模型和物理对象时，仍然起着很重要的作用。

软件对象是一种将状态和行为有机地结合起来而形成的软件构造模型，它可以用来描述现实世界中的一个对象，也就是说，软件对象实际上就是现实世界对象的模型，它有状态和行为。一个软件对象可以利用一个或多个变量来标识它的状态，变量是由用户标识符来命名的数据项。软件对象可以利用它的方法来执行它的行为，方法是与对象相关联的函数(子程序)。

我们可以利用软件对象来代表现实世界中的对象。例如，用一个飞行试驾程序来代表现实世界中正在飞行的飞机，或者用机床数控程序来代表现实世界中运行的机床。同样也可以使用软件对象来表示抽象的概念。例如，单击按钮事件就是一个用在 GUI 窗口系统的公共对象，它可以代表用户单击程序界面中确定按钮的动作。

1.1.3　类

类(Class)是具有相同属性和操作的一组对象的组合，也就是说，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。例如，名为“学生”的类被用于描述为被学生管理系统管理的学生对象。

类的定义包含以下要素。

- 定义该类对象的数据结构(属性的名称和类型)。

- 类的对象在系统中所需要执行的各种操作，如对数据库的操作。

类是对象集合的再抽象，其与对象的关系如同一个模具和使用这个模具浇注出来的铸件一样，是创建软件对象的模板——一种模型。类给出了属于该类的全部对象的抽象定义，而对象是符合这种定义的一个实体。类有以下两个用途。

- 在内存中开辟一个数据区，存储新对象的属性。
- 把一系列行为和对象关联起来。

一个对象又被称作类的一个实例，也称为实体化(Instantiation)。术语“实体化”是指对象在类声明的基础上创建的过程。例如，如果我们声明了一个“学生”类，则可以在这个基础上创建“一个姓名叫李刚的学生”对象。

类的确定和划分没有一个统一的标准和方法，基本上依赖于设计人员的经验、技巧及对实际项目中问题的把握。通常的标准是“寻求共性、抓住特性”，即在一个大的系统环境中，寻求事物的共性，将具有共性的事物用一个类进行表述，在用具体的程序实现时，具体到某一个对象，并抓住对象的特性。确定一个类的步骤通常包含以下几个方面。

(1) 确定系统的范围，如学生管理系统，需要确定一下与学生管理相关的内容。

(2) 在系统范围内寻找对象，该对象通常具有一个和多个类似的事物。例如，在学生管理系统中，某院系有一个名叫李刚的学生，而另一个院系的名叫王芳的人是和李刚类似的，都是学生。

(3) 将对象抽象成为一个类，按照上面类的定义，确定类的数据和操作。

在面向对象程序设计中，类和对象的确定非常重要，是软件开发的第一步，软件开发中类和对象的确定直接影响软件的质量。如果划分得当，对于软件的维护与扩充及体现软件的重用性方面，都非常重要。

1.1.4 消息与事件

当使用某一个系统时，单击后，通常会显示相应的信息。以学生管理系统为例，单击“学生管理系统”界面某菜单时，会显示出当前的操作人所需要的信息。当前程序的运行过程如下。

(1) “学生管理系统”界面的某一个菜单项发送鼠标单击事件给相应的对象一个消息。

(2) 对象接收到消息后有所反应，把操作者需要的信息显示在界面。

(3) 界面将相关信息显示出来，完成任务。

可以看得出，在这个过程中，我们首先要触发一个事件，然后发送消息，那么消息是什么呢？所谓消息(Message)，是指描述事件发生的信息，是对象间相互联系和相互作用的方式。一个消息主要由五部分组成：消息的发送对象、消息的接收对象、消息的传递方式、消息的内容(参数)、消息的返回。传入消息内容的目的有两个，一个是让接收请求的对象获取执行任务的相关信息，另一个是行为指令。

那么什么是事件呢？所谓事件，通常是指一种由系统预先定义而由用户或系统发出的动作。事件作用于对象，对象识别事件并做出相应的反应。与对象的方法集可以无限扩展不同，事件的集合通常是固定的，用户不能随便定义新的事件。但是现代高级语言中可以通过一些其他技术在类中加入事件。我们通常所熟悉的一些事件，如 Click，单击对象时发生的事件；Load，当界面被加载到内存中时发生的事件等。

对象通过对外提供的方法在系统中发挥自己的作用，当系统中的其他对象请求这个对象执行某个方法时，就向该对象发送一个消息，对象响应这个请求，完成指定的操作。程序的执行取决于事件发生的顺序，由顺序产生的消息来驱动程序的执行，而不必预先确定消息产生的顺序。

1.2 面向对象的三大要素

封装、继承、多态是面向对象程序的三大特征，这些特征保证了程序的安全性、可靠性、可重用性和易维护性。随着技术的发展，把这些思想用于硬件、数据库、人工智能技术、分布式计算、网络、操作系统等领域，越来越显示出其优越性。

1.2.1 封装

封装(Encapsulation)就是把对象的状态和行为绑到一起的机制，使对象形成一个独立的整体，并且尽可能地隐藏对象的内部细节。封装有两个含义：一是把对象的全部状态和行为结合在一起，形成一个不可分割的整体。对象的私有属性只能够由对象的行为来修改和读取。二是尽可能隐蔽对象的内部细节，与外界的联系只能够通过外部接口来实现。

封装的信息屏蔽作用反映了事物的相对独立性，我们可以只关心它对外所提供的接口，即能够提供什么样的服务，而不用去关注其内部的细节问题。例如，使用手机时，我们关注的通常是这个手机能实现什么功能，而不太会去关心这个手机是怎么一步步被制造出来的。

封装的结果使对象以外的部分不能随意更改对象的内部属性或状态，如果要更改对象内部的属性或状态，则需要通过公共访问控制器进行。通过公共访问控制器来限制对象的私有属性，有以下好处。

- 避免对封装数据的未授权访问。
- 当对象为维护一些信息，并且这些信息比较重要，不能够随便向外界传递的时候，只需要将这些信息属性设置为私有即可。
- 帮助保护数据的完整性。
- 当对象的属性设置为公共访问的时候，代码可以不经过对象所属类希望遵循的业务流程而去修改对象的值，对象很容易失去对其数据的控制。我们可以通过访问

控制器来修改私有属性的值，并且在赋值或取值的时候检查属性值的正确与否。

- 当类的私有方法必须修改时，限制了对整个应用程序的影响。

当对象采用一个公共的属性来暴露时，修改该公共属性的名称、程序都需要修改这个公共属性被调用的地方。但是，通过私有的方式就能够将程序的影响范围缩小到一个类中。

例如，房子就是一个类的实例，室内的装饰和摆设只能被室内的居住者欣赏和使用，如果没有四面墙的遮挡，室内的所有活动在外人面前将一览无遗。由于有了封装，房屋内的所有摆设都可以被随意改变且不影响他人，然而，如果没有门窗，即使它的空间再宽阔，也没有实用的价值。房屋的门窗，就是封装对象暴露在外的属性和方法，专供人进出，以及空气流通和带来阳光。

但是在实际项目中，如果一味地强调封装，对象的任何属性都不允许外部直接读取，反而会增加许多无意义的操作，为编程增加负担。为避免这一点，在语言的具体使用过程中，应该根据需要和具体情况，来决定对象属性的可见性。

1.2.2 继承

对于客观事物的认知，既要看到其共性，也要看到其特性。如果只考虑事物的共性，不考虑事物的特性，就不能反映出客观世界中事物之间的层次关系，从而不能完整、正确地对客观世界进行抽象的描述。如果运用抽象的原则就是舍弃对象的特性，提取其共性，从而得到适合一个对象集的类，那么在这个类的基础上，再重新考虑抽象过程中被舍弃的对象的特性，则可以形成一个新的类，这个类具有前一个类的全部特征，是前一个类的子集，从而形成一种层次结构，即继承结构。例如，动物可以分为哺乳动物、爬行动物、两栖动物和鸟类等，我们通过抽象的方式实现一个动物类以后，可以通过继承的方式分别实现哺乳动物、爬行动物、两栖动物、鸟类等类，并且这些类包含动物的特性。动物类继承结构示例如图 1-1 所示。

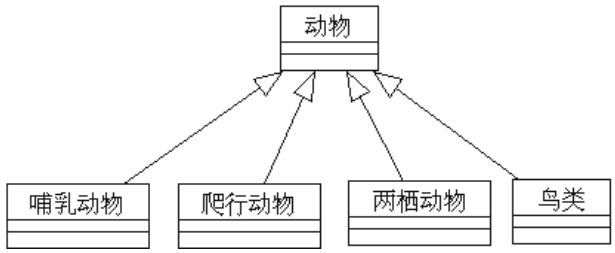

图 1-1　动物类继承结构示例

继承(Inheritance)是一种连接类与类之间的层次模型，其是指特殊类的对象拥有其一般类的属性和行为。继承意味着“自动地拥有”，即在特殊类中不必重新对已经在一般类中定义过的属性和行为进行定义，而是自动地、隐含地拥有其一般类的属性和行为。继承对类的重用性提供了一种明确表述共性的方法，即一个特殊类既有自己定义的属性和

行为，又有继承下来的属性和行为。尽管继承下来的属性和行为在特殊类中是隐式的，但无论在概念上还是在实际效果上，都是这个类的属性和行为。继承是传递的，当这个特殊类被它更下层的特殊类继承的时候，它继承来的及自己定义的属性和行为又被下一层的特殊类继承下去。我们有时把一般类称为基类，把特殊类称为派生类。

继承在面向对象软件开发过程中，有其强有力和独特的一面，通过继承可以实现以下几点。

- 使派生类能够比不使用继承直接进行描述的类更加简洁。派生类只需要描述与基类不相同的、特殊的地方，并添加到类中继承即可。如果不使用继承而去直接描述，则需要将基类的属性和行为全部进行描述一遍。
- 能够重用和扩展现有类库资源。当我们使用已经封装好的类库时，如果需要对某个类进行扩展，则通过继承的方式很容易实现，而不需要再重新编写，并且扩展一个类的时候并不需要其源代码。
- 使软件易于维护和修改。当要修改或增加某一属性或行为时，只需要在相应的类中进行改动，而它派生的所有类全都自动地、隐含地做了相应的修改。

在软件开发过程中，继承性实现了软件模块的可重用性、独立性，缩短了开发的周期，提高了软件的开发效率，同时使软件易于维护和修改。继承是对客观世界的直接反映，通过类的继承，能够实现对问题深入抽象的描述，也反映出人类认知问题的发展过程。

1.2.3　多态

多态是指两个或多个属于不同类的对象对于同一个消息或方法调用所做出不同响应的能力。面向对象设计也借鉴了客观世界的多态性，体现在不同的对象可以根据相同的消息产生各自不同的动作。例如，我们在“动物”基类中定义了“进食”这个行为，派生类“猫”和“狗”都继承了动物类的进食行为，但其进食的事物却不一定相同，猫喜欢吃鱼，而狗喜欢啃骨头。这样一个进食的消息发出以后，猫类和狗类的对象接收到这个消息后各自执行不同的进食行为，如图 1-2 所示就是多态性示例。

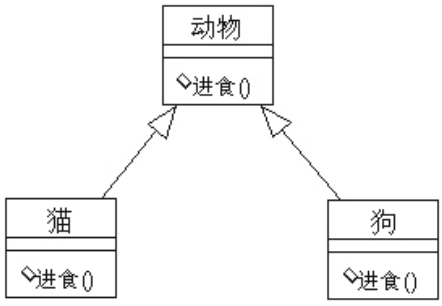

图 1-2　多态性示例

具体到面向对象程序设计来讲，多态性(Polymorphism)是指在两个或多个属于不同类的对象中，同一函数名对应多个具有相似功能的不同函数，可以使用相同的调用方式来调用这些具有不同功能的同名函数。

继承性和多态性的结合可以生成一系列虽类似但独一无二的对象。由于继承性，这些对象共享许多相似的特征；由于多态性，针对相同的消息，不同的对象可以有独特的表现方式，实现个性化的设计。

上述面向对象技术的几个特征的运用，对提高软件的开发效率起着非常重要的作用，通过编写可重用代码、编写可维护代码、修改代码模块、共享代码等方法可以充分发挥其优势。

1.3 面向对象与项目设计

面向对象设计是把分析阶段得到的需求转变成符合成本和质量要求的抽象的系统实现方案的过程。从面向对象分析到面向对象设计，是一个逐渐扩充模型的过程。

1.3.1 用面向对象的方法分析项目需求

面向对象分析的目的是认知客观世界的系统并对系统进行建模。因此，就需要在面向对象分析过程中，根据客观世界的具体实例来构造问题域中准确、具体、严密的分析模型。构造分析模型的用途有 3 个：①用来明确问题域的需求；②为用户和开发人员提供明确需求；③为用户和开发人员提供一个协商的基础，作为后继的设计和实现的框架。需求分析的结果应以文档的形式存在。

如图 1-3 所示就是面向对象的分析过程。

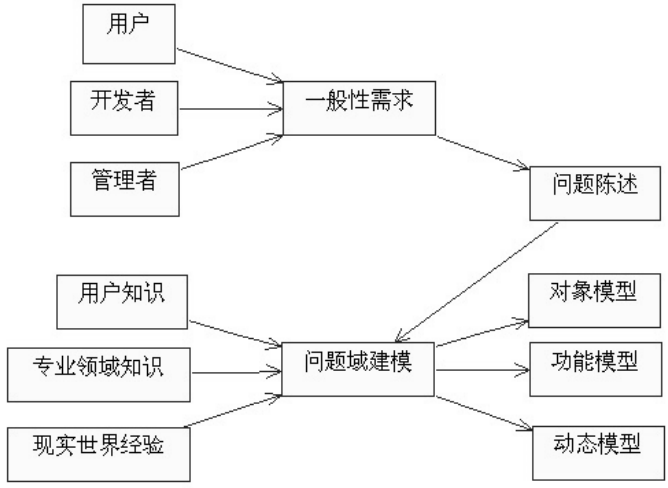

图 1-3　面向对象的分析过程

1. 获取需求内容陈述

系统分析的第一步就是获取需求内容陈述。分析者必须同用户一起工作来提炼这些需求，以及了解清楚用户的真实意图是什么，其中的过程涉及对需求的分析及关联信息的查找。

以“学生管理系统”为例，需求内容陈述如下：在学生管理信息系统中，要为每个学生建立个人信息档案，包括学号、姓名、性别、年龄、入学时间、专业、联系电话和家庭住址等个人信息。系统管理员登录系统并通过身份验证后，可以执行对学生个人信息档案的添加、删除、修改和查询操作。教师登录系统并通过验证后，可以录入自己负责的科目、学生的考试成绩，也可查询学生的个人信息和成绩。学生登录系统并通过身份验证后可以查询自己的个人所有信息。校长登录系统并通过身份验证后可以查询学校所有学生的全部信息。系统管理员要完成系统维护工作，包括日志、管理员权限、学生信息、数据库的维护等工作。

2. 建立系统的对象模型结构

系统分析的第二步就是建立系统的对象模型结构。

要建立系统的对象模型结构首先要标识和关联类，因为类的确定及关联影响整个系统的结构和解决问题的方法；其次是增加类的属性，进一步描述类和关联的基本网络，在这个过程中我们可以使用继承、包等来组织类；最后是将操作增加到类中作为构造动态模型和功能模型的副产品。下面就分别进行介绍。

1) 标识和确定类

构造对象模型的第一步是标出来自问题域的相关对象类，这些对象类包括物理实体和概念的描述。所有类在应用中都应当是有意义的，在问题陈述中，并非所有类都是明显给出的，有些是隐含在问题域或一般知识中的。通常来说，一个确定类的过程包括：从需求说明中选取相关的名词确定一些类，然后对这些类进行分析；过滤掉不符合条件的类。如图 1-4 所示是一个确定类的过程。

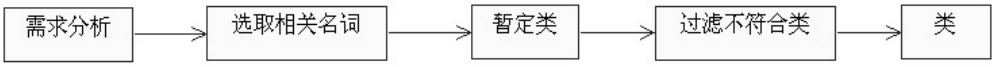

图 1-4　确定类的过程

我们查找问题陈述中的所有名词，产生如下的暂定类。

软件	学生管理系统	系统管理员	学生
老师	个人信息	管理员权限	学生的考试成绩
学生的班级	日志	数据库维护	校长

接下来我们根据下列标准，去掉一些不必要的类和不正确的类。

- 消除冗余类。如果存在两个类表述了同一个信息，则保留最富有描述能力且与系统紧密相关的类。例如，“个人信息”和“学生”就是重复的描述，因为“学生”最富有描述性，所以保留它。
- 删除与系统不相干的类。与问题没有关系或根本无关的类，在类的确定中应当删除。
- 删除模糊类。类必须是明确的，而有些暂定类中边界的定义模糊或范围太广，如“软件”就是模糊类，就这个系统而言，它是指“学生管理系统”。
- 删除属性。某些名词描述的是其他对象的属性，应当把这些类从暂定类中删除。但是如果某一个名词的独立性很重要，则应该把它归属到类，而不把它作为属性。例如，“学生的考试成绩”和“学生的班级”属于学生信息的属性，应当删除。
- 删除操作。如果问题陈述中的名词中有动作含义的名词，则含有这样描述的操作的名词不是类，需删除。例如，“数据库维护”属于操作而不是类。

在图书管理信息系统中，根据上面的标准，把“软件”“个人信息”“学生的考试成绩”“学生的班级”“数据库维护”等类删除。

2) 准备数据字典

为所有建模实体准备一个数据字典来进行描述。数据字典应当准确描述各个类的精确含义及当前问题中类的范围，包括对类的成员、用法方面的假设或限制等。例如，学生的信息应当包括姓名、学号、年龄、性别、入学时间等。

3) 确定关联

关联是指两个或多个类之间的相互依赖。一种依赖表示一种关联，可用各种方式来实现关联。关联常用描述性动词或动词词组来表示，其中有物理位置的表示、传导的动作、通信、所有者关系、条件的满足等。从问题陈述中抽取所有可能的关联表述，把它们记下来，但不要过早地细化这些表述。

系统中所有可能的关联，大多数是直接抽取问题中的动词词组而得到的。在陈述中，有些动词词组表述的关联是不明显的。最后，还有一些关联与客观世界或人的假设有关，必须同用户一起核实这种关联，因为这种关联在问题陈述中找不到。

学生管理信息系统问题陈述中的关联如下。

- 每个学生建立一个个人档案。
- 只有系统管理员有权对学生的信息进行添加、删除、修改。
- 学生包括各个专业的学生。
- 一个学生有多个科目的成绩。
- 一个班级有多个学生。
- 系统管理员完成系统维护工作。
- 维护包括日志、管理员权限、学生信息、数据库的维护等工作。
- 系统提供个人信息的安全保证。

使用下列标准删除不必要和不正确的关联。

- 如果某个类被删除，那么与它有关的关联也必须删除，或者用其他类来进行重新表述。
- 不相干的关联或实现阶段的关联应当被删除。删除所有问题域之外的关联或涉及实现结构中的关联。
- 某些动作应当被删除。关联应该描述应用域的结构性质而不是瞬时事件，因此对一些瞬时事件的描述也应当被删除。
- 派生关联应当被删除。省略可以用其他关联来定义的关联，因为这种关联是冗余的。

4) 确定属性

属性是个体对象的性质，通常用修饰性的名词词组来表示。形容词常表示具体的可枚举的属性值，属性不可能在问题陈述中完全表述出来，必须借助于应用域的知识及对客观世界的知识才可以找到它们。我们只需考虑与具体应用直接相关的属性，不要考虑超出问题范围的属性，首先找出重要属性，避免只用于实现的属性，要为各个属性取一个有意义的名字。按下列标准删除不必要的和不正确的属性。

- 可以作为对象的属性。若实体的独立存在比它的值重要，那么这个实体不是属性而是对象。例如，在邮政目录中，“城市”可以看作一个属性，然而在人口普查中，“城市”则被看作对象。在具体应用中，具有自身性质的实体一定是对象。
- 对象的限定词。若属性值取决于某种具体上下文，则可考虑把该属性重新表述为一个限定词。
- 对象的名称。名称常作为限定词而不是对象的属性，当名称不依赖于上下文关系时，即为一个对象属性，尤其是它不唯一时。
- 对象的标识符。在考虑对象模糊性时，引入对象标识符来表示。在对象模型中不列出这些对象标识符(它是隐含在对象模型中的)，只列出存在于应用域的属性。
- 对象的内部值。若属性描述了对外不透明的对象的内部状态，则应从对象模型中删除该属性。
- 细化的细节。忽略不能对大多数操作有影响的属性。

5) 使用继承来细化类

使用继承来共享公共属性，以此对类进行组织，一般可以使用下列两种方式进行。

- 自底向上通过把现有类的共同性质一般化为父类，寻找具有相似属性、关系或操作的类来发现继承。例如，“博士生”和“本科生”是类似的，可以一般化为“大学生”。这些一般化结果常常是基于客观世界边界的现有分类，只要可能，尽量使用现有概念。
- 自顶向下将现有的类细化为更具体的子类。具体化可以从应用域中明显看出来，在应用域中各枚举情况是最常见的具体化的来源。例如，按钮可以有普通按钮、单选按钮、多选按钮等，这就可以把按钮类具体细化为各种具体按钮的子类。当同一关联名出现多次且意义也相同时，应尽量具体化为相关联的类。在类层次中，

可以为具体的类分配属性和关联。各属性和关联都应分配给最一般的适合的类，有时也加上一些修正。应用域中各枚举情况是最常见的具体化的来源。

6) 完善对象模型

对象建模不可能一次就能保证模型是完全正确的，软件开发的整个过程就是一个不断完善的过程。模型的不同组成部分大多是在不同的阶段完成的，如果发现模型的缺陷，就必须返回前期阶段去修改，而且有些细化工作是在动态模型和功能模型完成之后才开始进行的。

下面是几种可能丢失对象的情况及解决办法。

- 若同一类中存在毫无关系的属性和操作，则分解该类，使各部分相互关联。
- 若一般化体系不清楚，则可以分离扮演两种角色的类。
- 若存在无目标类的操作，则找出并加上失去目标的类。
- 若存在名称及目的相同的冗余关联，则通过一般化创建丢失的父类，把关联组织在一起。

对于多余类还需要进行查找，可删除缺少属性、操作和关联的类。对于丢失的关联的查找，如果丢失了操作的访问路径，则可加入新的关联以回答查询。

3. 建立对象的动态模型

进行分析的第三步是建立对象的动态模型。建立对象的动态模型的过程一般包含下列几个步骤。

1) 准备脚本

动态分析从寻找事件开始，然后确定各对象的可能事件顺序。在分析阶段不考虑算法的执行，算法是实现模型的一部分。

2) 确定事件

确定所有外部事件。事件包括所有来自或发往用户的信息、外部设备的信号、输入、转换和动作，可以发现正常事件，但不能遗漏条件和异常事件。

3) 准备事件跟踪表

把脚本表示成一个事件跟踪表，即不同对象之间的事件排序表，对象为表中的列，给每个对象分配一个独立的列。

4) 构造状态图

对各对象类建立状态图，反映对象接收和发送的事件，每个事件跟踪都对应于状态图中的一条路径。

4. 建立系统功能模型

进行分析的第四步是建立对象的功能模型。功能模型是用来说明值是如何计算的，标明值与值之间的依赖关系及相关的功能。数据流图有助于表示功能依赖关系，其的处理在状态图的活动和动作中进行标识，其中的数据流对应于对象图中的对象或属性。

1) 确定输入值、输出值

先列出输入、输出值。输入、输出值是系统与外界之间的事件的参数。

2) 建立数据流图

数据流图说明输出值是怎样从输入值得来的，数据流图通常按层次组织。

5. 确定类的操作

在建立对象模型时，确定了类、关联、结构和属性，还没有确定操作。只有建立了动态模型和功能模型之后，才可能最后确定类的操作。

1.3.2　用面向对象的方法设计系统

前面已提到过，面向对象设计是把分析阶段得到的需求转变成符合成本和质量要求的抽象的系统实现方案的过程。从面向对象分析到面向对象设计是一个逐渐扩充模型的过程。

1. 面向对象设计的准则

面向对象设计的准则包括模块化、抽象、信息隐藏、低耦合和高内聚等，下面我们对这些特征进行一一介绍。

1) 模块化

面向对象开发方法很自然地支持了把系统分解成模块的设计原则：对象就是模块。它是把数据结构和操作这些数据的方法紧密地结合在一起所构成的模块。类的设计要很好地支持模块化这一准则，这样能使系统有更好的维护性。

2) 抽象

面向对象方法不仅支持对过程进行抽象，而且支持对数据进行抽象。抽象方法的好坏及抽象的层次都对系统的设计有很大的影响。

3) 信息隐藏

在面向对象方法中，信息隐藏是通过对象的封装性来进行实现的。对象暴露接口的多少及接口的好坏都对系统设计有很大的影响。

4) 低耦合

在面向对象方法中，对象是最基本的模块，因此，耦合主要是指不同对象之间相互关联的紧密程度。低耦合是设计的一个重要标准，因为这有助于使系统中某一部分的变化对其他部分的影响降到最低限度。低耦合的程序有助于类的维护，也是衡量类质量的一个很重要的指标。

5) 高内聚

在面向对象方法中，高内聚也是必须满足的条件。高内聚是指在一个对象类中应尽量多地汇集逻辑上相关的计算资源。如果一个模块只负责一件事情，则说明这个模块有

很高的内聚度；如果一个模块负责了很多相关的事情，则说明这个模块的内聚度很低。内聚度高的模块通常容易理解，很容易被复用、扩展和维护。较低的耦合度和较高的内聚度，也即我们常说的“低耦合、高内聚”，是所有优秀软件的共同特征。

2. 面向对象设计的启发规则

在面向对象设计中，可以通过使用一些实用的规则来指导我们进行面向对象的设计。通常这些面向对象设计的启发规则包含以下的内容。

1) 设计的结果应清晰、易懂

使设计结果清晰、易懂、易读是提高软件可维护性和可重用性的重要措施。显然，人们不会重用那些他们不理解的设计。

要使设计的结果清晰、易懂，一般要做到以下几个方面。

- 用词一致。用词不一致会产生理解不一致，增加理解的负担。
- 使用已经存在的函数或方法。已经存在的函数或方法有助于减少函数或方法的数量。
- 减少消息模式的数量。减少消息模式的数量会减少很多不必要的记忆。
- 避免模糊的定义。模糊的定义会给设计和阅读带来麻烦。

2) 一般到具体结构的深度应适当

通常来说，从一般到具体的抽象过程，抽象得越深，对于程序的可移植性也就越好，但是抽象层次过多会给编写和维护带来很大的麻烦。一般来讲，适度的抽象能够更好地提高软件的开发效率和维护工作，系统分析员可根据具体的情况进行抽象。

3) 尽量设计小而简的类

系统设计应当尽量设计小而简的类，这样便于开发和管理程序。为了保持类的设计简单，通常应注意以下几点。

- 类中避免包含过多的属性。
- 每一个类应当有自己明确的定义。
- 尽量简化对象之间的合作关系。
- 对外不要提供太多的操作。

4) 使用简单的消息协议

简单的消息协议有助于帮助记忆和测试，一般来讲，消息中参数的个数不要超过 3 个。

5) 使用简单的函数或方法

通常来讲，面向对象设计出来的类中的函数或方法要尽可能的小，有一些书上建议一个函数或方法一般有三至五行源程序即可，可以用仅含一个动词和一个宾语的简单句子来描述它的功能。

6) 把设计变动减至最小

通常，设计的质量越高，设计结果保持不变的时间也越长，即使出现必须修改设计的情况，也应该使修改的范围尽可能小。提高设计质量是系统设计工作的一大挑战。

3. 系统设计

系统设计是问题求解及建立解答的高级策略。系统必须制定解决问题的基本方法，系统的高层结构形式包括子系统的分解、系统的固有并发性、子系统如何分配给硬软件、数据存储管理、资源协调、软件控制实现、定义人机交互接口等。

系统设计一般是先从高层入手，然后细化。系统设计要决定整个结构及风格，这种结构为后面设计阶段更详细的设计策略提供了基础。下面介绍整个系统设计的一般步骤。

1) 分解系统

系统中主要的组成部分称为子系统，子系统既不是一个对象也不是一个功能，而是类、关联、操作、事件和约束的集合。

2) 确定并发性

分析模型、现实世界及硬件中的很多对象均是并发的。

3) 处理器及任务分配

系统必须分配给各并发子系统单个的硬件单元，要么是一个一般的处理器，要么是一个具体的功能单元。

4) 数据存储管理

系统中的内部数据和外部数据的存储管理是一项重要的任务。通常各数据存储可以将数据结构、文件、数据库组合在一起，不同数据的存储要在费用、访问时间、容量及可靠性之间做出折中考虑。

5) 全局资源的处理

系统必须确定全局资源，并且制定访问全局资源的策略。

6) 选择软件控制机制

分析模型中所有的交互行为都表示为对象之间的事件。系统设计必须从多种方法中选择某种方法来实现软件的控制。

7) 人机交互接口设计

设计中的大部分工作都与稳定的状态行为有关，但必须考虑用户使用系统的交互接口。

1.4 用面向对象思想建立系统模型

在面向对象的开发和设计中，借鉴了建筑行业中的建模思想。在建筑行业中，建模是一项经过检验并被人们广泛接受的工程技术。人们在建立房屋和大厦等建筑物的时候，首先创建建筑物的模型，以帮助用户得到实际建筑物的整体印象，并且可以通过建立数学模型来分析各种因素对建筑物造成的影响，如建筑物的地面压力、地震等。

面向对象的建模以面向对象开发者的观点创建所需要的系统。事实上，选择创建什么样的模型，对如何解决问题和如何形成解决方案有深远的影响。

1.4.1 瀑布模型

瀑布模型也被称为生存周期模型，其核心思想是按照相应的工序将问题进行简化，将系统功能的实现与系统的设计工作分开，便于项目之间的分工与协作，即采用结构化的分析与设计方法将逻辑实现与物理实现分开。瀑布模型将软件的生命周期划分为软件计划、需求分析和定义、软件设计、软件实现、软件测试、软件运行与维护 6 个阶段，并且规定了它们自上而下的次序，如同瀑布一样下落，每一个阶段都是依次衔接的。采用瀑布模型的软件开发过程如图 1-5 所示。

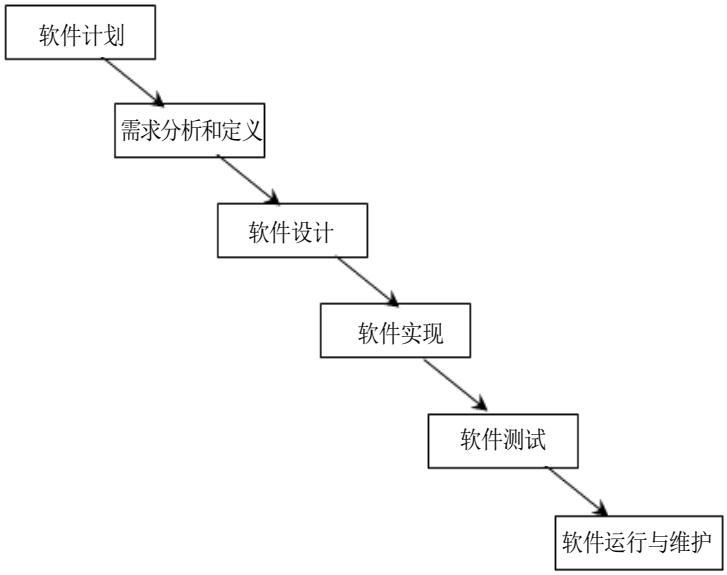

图 1-5　采用瀑布模型的软件开发过程

瀑布模型是最早出现的软件开发模型，在软件工程中占有重要的地位，它提供了软件开发的基本框架。瀑布模型的软件开发过程是，从上一项活动接收该项活动的工作对象作为输入，利用这一输入实施该项活动应完成的内容，给出该项活动的工作成果，并作为输出传给下一项活动。同时评审该项活动的实施，若确认，则继续下一项活动；否则返回到前面，甚至更前面的活动。

瀑布模型为项目提供了按阶段划分的检查点，这样有利于软件开发过程中人员的组织及管理。瀑布模型让我们在当前一阶段完成后，才去关注后续阶段，这样有利于开发大型的项目。然而软件开发的实践表明，瀑布模型也存在一定的缺陷，具体如下。

- 只有在项目生命周期的后期才能看到结果。由于开发模型呈线性，所以当开发成果尚未经过测试时，用户是无法看到软件效果的，这样不能在开发过程中及时得

到反馈，增加了项目开发的风险。在软件开发前期未发现的错误传到后面的开发活动中，进而可能会造成整个软件项目开发失败。

- 通过过多的强制完成日期和里程碑来跟踪各个项目阶段。在每个项目的开发阶段，瀑布模型是通过强制固定的完成日期和里程碑进行项目跟踪的，这使得在项目开发过程中缺乏足够的灵活性，特别是对于需求不稳定的项目更加麻烦。
- 在软件需求分析阶段，要完全地确定系统用户的所有需求是一件比较困难的事情，甚至可以说完全确定是不太可能的。

尽管瀑布模型存在一定的缺陷，但是它对很多类型的项目而言依然是有效的，特别是在进行一些大型项目的开发时。如果能够正确使用，则可以节省大量的时间和金钱。对于所开发的项目而言，是否使用这一模型主要取决于能否理解客户的需求及在项目的进程中这些需求的变化程度。对于能够在前期确定需求的项目，瀑布模型还是有一定价值的。

1.4.2　喷泉模型

喷泉模型是一种以对象为驱动、以用户需求为动力的模型，主要用于描述面向对象的软件开发过程。该模型认为软件开发过程中自下而上周期的各阶段是相互重叠和多次反复的，类似一个喷泉，水喷上去又可以落下来。各个开发阶段没有特定的次序要求，可以交互进行，并且可以在某个开发阶段中随时补充其他任何开发阶段中的遗漏。采用喷泉模型的软件开发过程如图 1-6 所示。

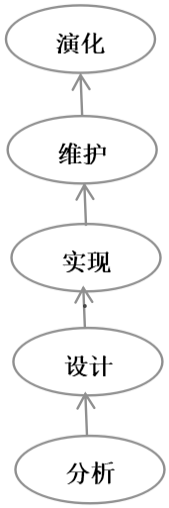

图 1-6　采用喷泉模型的软件开发过程

喷泉模型主要用于面向对象的软件项目，软件的某个部分通常被重复多次，相关对象在每次迭代中随之加入渐进的软件成分。各活动之间无明显边界，例如，设计和实现之间没有明显的边界，这也称为“喷泉模型的无间隙性”。由于对象概念的引入、表达分

析、设计及实现等活动只用对象类和关系，从而可以较容易地实现活动的迭代和无间隙性。

喷泉模型不像瀑布模型需要分析活动结束后才开始设计活动，设计活动结束后才开始编码活动，该模型的各个阶段没有明显的界限，开发人员可以同步进行开发。

喷泉模型的优点是：可以提高软件项目的开发效率，节省开发时间，适应于面向对象的软件开发过程。

喷泉模型的缺点是：由于喷泉模型在各个开发阶段是重叠的，因此在开发过程中需要大量的开发人员，不利于项目的管理。此外这种模型要求严格管理文档，使得审核的难度加大，尤其是面对可能随时加入各种信息、需求与资料的情况。

1.4.3 基于构件的开发模型

基于构件的开发模型是利用模块化方法将整个系统模块化，并在一定构件模型的支持下复用构件库中的一个或多个软件构件，通过组合手段高效率、高质量地构造应用软件系统的过程。基于构件的开发模型融合了螺旋模型的许多特征，本质上是演化形的，开发过程是迭代的。基于构件的开发模型由软件计划、需求分析和定义、软件快速原型、原型评审及软件设计和实现 5 个阶段组成，采用这种开发模型的软件开发过程如图 1-7 所示。

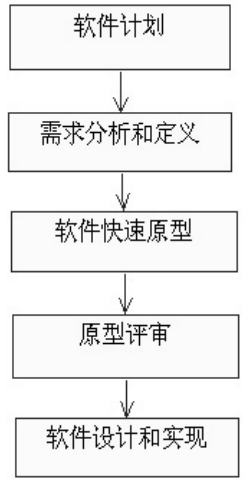

图 1-7　采用基于构件的开发模型的软件开发过程

构件作为重要的软件技术和工具得到了极大的发展，这些新技术和工具有Microsoft的DCOM、Sun的EJB及OMG的CORBA等。基于构件的开发活动从标识候选构件开始，通过搜查已有构件库，确认所需要的构件是否已经存在。如果已经存在，则从构件库中提取出来复用；否则采用面向对象方法开发它。然后通过语法和语义检查后将提取出来的构件通过胶合代码组装到一起以实现系统，这个过程是迭代的。

基于构件的开发方法使软件开发不再是一切从头开始，开发的过程就是构件组装的过程，维护的过程就是构件升级、替换和扩充的过程。

基于构件的开发模型的优点是：构件组装模型使软件可复用，提高了软件开发的效率。构件可由一方定义其规格说明，被另一方实现，然后供给第三方使用。构件组装模型允许多个项目同时开发，降低了费用，提高了可维护性，可实现分步提交软件产品。

基于构件的开发模型的缺点是：由于采用自定义的组装结构标准，缺乏通用的组装结构标准，因而引入了较大的风险，可重用性和软件高效性不易协调，需要精干的、有经验的分析和开发人员。客户的满意度低，并且由于过分依赖于构件，所以构件库的质量影响产品质量。

1.4.4　XP 开发模型

敏捷方法是近几年兴起的一种轻量级的开发方法，它强调适应性而非预测性，强调以人为中心，而不以流程为中心，以及对变化的适应和对人性的关注，其是一个轻载、基于时间、紧凑、并行并基于构件的软件过程。在所有的敏捷方法中，XP(eXtreme Programming)方法是最引人注目的一种轻型开发方法，它规定了一组核心价值和方法，消除了大多数重量型开发过程中的不必要产物，建立了一个渐进型开发过程。XP方法将开发阶段的 4 个活动(分析、设计、编码和测试)混合在一起，在全过程中采用迭代增量开发、反馈修正和反复测试的方法。XP开发模型把软件的生命周期划分为用户场景、体系结构、发布计划、迭代、验证测试和小型发布 6 个阶段，采用这种开发模型的软件开发过程如图 1-8 所示。

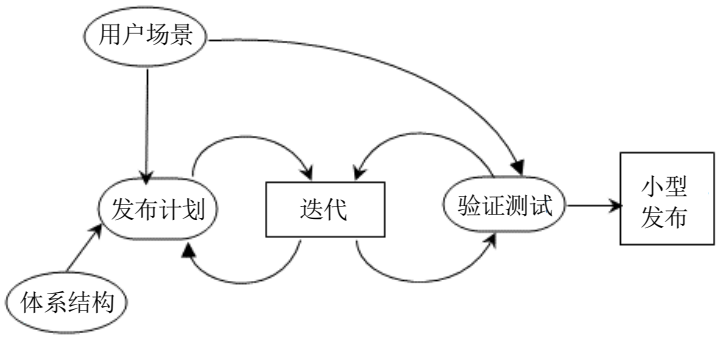

图 1-8　采用 XP 开发模型的软件开发过程

XP开发模型通过对传统软件开发的标准方法进行重新审视，提出了由一组规则组成的一个简便易行的过程。由于这些规则是通过在实践中观察使软件高效或缓慢的因素而得出的，因此它既考虑了保持开发人员的活力和创造性，又考虑了开发过程的有组织、有重点和持续性。XP开发模型是面向客户的开发模型，重点强调用户的满意程度，开发过程中对需求改变的适应能力较强，即使在开发的后期，也可较高程度地适应用户的改变。

XP开发模型与传统模型相比具有很大的不同，其核心思想是交流(Communication)、简单(Simplicity)、反馈(Feedback)和进取(Aggressiveness)。XP开发小组不仅包括开发人员，还包括管理人员和客户。该模型强调小组内成员之间要经常进行交流，在尽量保证质量可以运行的前提下力求过程和代码的简单化；来自客户、开发人员和最终用户的具体反馈意见可以提供更多的机会来调整设计，保证把握正确的开发方向；进取则包含于交流、简单、反馈的原则中。

XP 模型的优点如下。

- 采用简单计划策略，不需要长期计划和复杂模型，开发周期短。
- 在全过程中采用迭代增量开发、反馈修正和反复测试的方法，软件质量有保证。
- 能够适应用户经常变化的需求，提供用户满意的高质量软件。

上面的开发模型或方法或许不能一概而论地说是以面向对象的建模为基础的开发模式，但是在各种开发方法中，都包含了软件的需求分析、软件的设计、软件的开发、软件的测试和软件的部署。在每一个阶段中，都可以借助于面向对象的建模和这些开发模型形成一套适合自己或企业的开发方式，主要体现在如下几个方面。

- 软件的需求分析阶段，对系统将要面临的具体管理问题及用户对系统开发的需求进行调查研究，即首先了解清楚要做什么的问题，然后分析问题的性质，求解问题，在繁杂的问题域中抽象地识别出对象及其行为、结构、属性、方法等。一般称之为面向对象的分析，即 OOA。
- 软件的设计阶段，首先整理问题，对分析的结果做进一步的抽象、归类、整理，然后以范式的形式将它们确定下来。一般称之为面向对象的设计，即 OOD。
- 软件的开发阶段，也即程序实现阶段，用面向对象的程序设计语言将上一步整理的范式直接映射(即直接用程序设计语言来取代)为应用软件。一般称之为面向对象的程序，即 OOP。

开发模式或方法毕竟是方法，如同在冷兵器和火器时代的排兵布阵一样，都有自己的技巧和内容，但是，一个是面向过程，另一个是面向对象的不同而赋予了不同的内容。在这些开发模型中，对于适用 UML 和面向对象开发的代表 Rational 统一过程(Rational Unified Process，RUP)，我们在第 3 章将会详细地讲解。

【本章小结】

在本章中，首先介绍了有关面向对象技术的大体概念，这有助于我们使用面向对象技术实现软件系统的建模工作。其次先后介绍了面向对象的三大基本特征和面向对象分析和设计的一般步骤。最后对软件的开发模式进行简要的介绍。本章节是对UML建模的面向对象的概念等进行全景式的描述，重点是面向对象的特征及面向对象设计的方法。

习题 1

1. 填空题

(1) _____是面向对象技术领域内占主导地位的标准建模语言，它统一了过去相互独立的数十种面向对象的建模语言共同存在的局面，形成了一个统一的、公共的、具有广泛适用性的建模语言。

(2) 类的定义要包含_____、_____和_____要素。

(3) 面向对象程序的三大要素是_____、_____和_____。

(4) 面向对象方法中的_____机制使子类可以自动地拥有(复制)父类全部属性和操作。

(5) 面向对象的系统分析要确立的 3 个系统模型是_____、_____和_____。

2. 选择题

(1) 如果想对一个类的意义进行描述，那么应该采用(　　)。

A. 标记值　　B. 规格描述

C. 注释　　D. 构造型

(2) 建立对象的动态模型的步骤有(　　)。

A. 准备脚本　　B. 确定事件

C. 构造状态图　　D. 准备事件跟踪表

(3) 软件的开发模式有(　　)。

A. 瀑布模型　　B. XP 开发模型

C. 喷泉模型　　D. 构件开发模型

(4) 下列关于类与对象的关系说法正确的是(　　)。

A. 有些对象是不能被抽象成类的

B. 类给出了属于该类的全部对象的抽象定义

C. 类是对象集合的再抽象

D. 类是用来在内存中开辟一个数据区，存储新对象的属性

(5) (　　)模型的缺点是缺乏灵活性，特别是无法解决软件需求不明确或不准确的问题。

A. 瀑布模型　　B. 增量模型

C. 原型模型　　D. 螺旋模型

3. 简答题

(1) 试述对象和类的关系。
(2) 请简要叙述面向对象的概念。
(3) 面向对象设计的原则有哪些？
(4) 软件开发的模式有几种？它们的优缺点各是什么？