CHAPTER

第5章 面向对象开发技术

面向对象技术主要强调在软件研发过程中面向客观现实世界或问题域中的事物，采用人类在认识客观世界的过程中普遍习惯运用的思维方法，更加直观、自然地描述客观世界中的有关事物，成为计算机界关注的重点和软件开发方法的主流。面向对象(Object Oriented，OO)是软件开发方法。面向对象的概念和应用已超越了程序设计和软件开发，扩展到如数据库系统、交互式界面、应用结构、应用平台、分布式系统、网络管理结构、CAD技术、人工智能等领域，其发展前景更为广阔。

教学目标

- 掌握面向对象及其方法的有关概念和特点。
- 理解面向对象软件的主要开发任务及过程。
- 熟悉面向对象分析(OOA)和面向对象设计(OOD)方法。
- 掌握一种面向对象分析和设计的方法的实际应用。

5.1 面向对象的相关概念

【案例 5-1】 面向对象是当前计算机界关心的重点，它是20世纪90年代软件开发方法的主流。面向对象的概念和应用已超越了程序设计和软件开发，扩展到很宽的范围，如数据库系统、交互式界面、应用结构、应用平台、分布式系统、网络管理结构、CAD技术、人工智能等领域。

从世界观的角度可以认为：①面向对象的基本哲学是认为世界是由各种各样具有自己的运动规律和内部状态的对象所组成的；②不同对象之间的相互作用和通信构成了完整的现实世界。因此，人们应当按照现实世界这个本来面貌来理解世界，直接通过对象及其相互关系来反映世界。这样建立起来的系统才能符合现实世界的本来面目。

从方法学的角度可以认为：①面向对象的方法是面向对象的世界观在开发方法中的直接运用；②它强调系统的结构应该直接与现实世界的结构相对应，应该围绕现实世界中的对象来构造系统，而不是围绕功能来构造系统。

5.1.1 对象与类

学习和掌握面向对象的开发技术,需要理解面向对象的相关概念。

1. 对象及其三要素

对象(Object)是描述客观事物的一个抽象(实体),是构成系统的基本单位。面向对象方法学以对象分解代替了传统方法的功能分解。面向对象的软件系统由对象组成,复杂的对象由简单的对象组合而成。**对象**具有**三要素**:对象标识、属性和服务。其中,**对象标识**(Object Identifier)为对象的名字,用于唯一地识别系统内部对象,在定义或使用对象时指定。**属性**(Attribute)也称为状态(State)或数据(Data),**用于**描述对象的静态特征。在某些 OOP 语言中,属性通常称为成员变量(Member Variable)或简称变量(Variable)。**服务**(Service)也称为操作(Operation)、行为(Behavior)或方法(Method)等,用于描述对象的动态特征,在某些 OOP 语言中,服务通常称为成员函数(Member Function)或简称函数(Function)。

2. 封装

封装(Encapsulation)有两层含义:一是对象是其全部属性和全部服务紧密结合而形成的一个不可分割的整体;二是对象如同一个密封的"黑盒子",表示对象状态的数据和实现操作的代码都被封装在其中,封装是对象的一个重要特性。在面向对象的系统中,对象是一个封装数据属性和操作行为的实体。使用某一对象时,只需知道其向外界提供的接口形式,无须知道其数据结构细节和实现操作的算法。

对象有两个视图,分别表现在分析设计和实现方面。在分析设计方面,对象表示一种概念,将有关现实世界的实体模型化。实现方面,一个对象表示在应用程序中所出现实体的实际数据结构。两个视图可将说明与实现分离,对数据结构和相关操作实现进行封装。

3. 类和实例

类(Class)也称为对象类(Object Class),是对具有相同属性和服务的一组对象的抽象定义。类与对象是抽象描述与具体实例的关系,一个具体的对象称为类的一个实例(Instance)。具有相同特征和行为的对象集合就是类。对象的状态包含在实例的属性中。

【案例 5-2】 "张三轿车"等具体对象可得到"轿车"类,而这些具体的对象就是该类的实例,如图 5-1 所示。

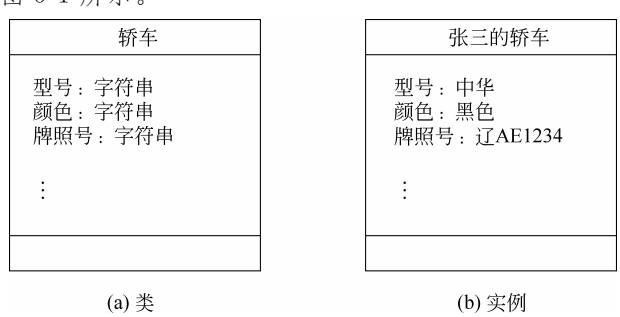

图 5-1 类与实例

类定义了各个实例所共有的结构，类的每个实例都可使用类中定义的操作。实例的当前状态是由实例所执行的操作定义的。通常类可视为一个抽象数据类型（ADT）的实现，更重要的是将类看作是表示某种概念的一个模型。实际上，类是单个的语义单元，可以很自然地管理系统中的对象，匹配数据定义及操作。类加入操作给通常的记录赋予语义，可提供各种级别的可访问性。

5.1.2 继承及多态性

1. 继承

继承（Inheritance）是父类和子类之间共享数据结构和方法的一种机制，是以现存的定义的内容为基础，建立新定义内容的技术，是类之间的一种关系。继承有两种：一是单重继承，指子类只继承一个父类的数据结构和方法；二是多重继承，指子类继承了多个父类的数据结构和方法。继承性通常表示父类与子类的关系，如图 5-2 所示。子类的公共属性和操作归属于父类，并为每个子类共享，子类继承了父类的特性。

图 5-3 是继承性描述的一种图示方法。通过继承关系还可构成层次关系，单重继承构成的类之间的层次关系为一树状，若将所有无子类的类都看成还有一个公共子类，多重继承构成的类之间的关系为一个网格，而且继承关系可传递。

图 5-2 继承性

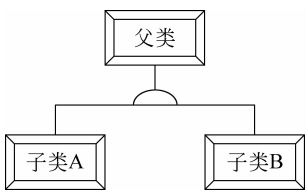

图 5-3 继承性描述

建立继承结构的优点有 3 个：一是易编程、易理解且代码短，结构清晰；二是易修改，共同部分只在一处修改即可；三是易增加新类，只须描述不同部分。

2. 多态性和动态绑定

多态性（Polymorphism）是指多种类型的对象在相同的操作或函数、过程中取得不同结果的特性。利用多态技术时，用户可发送一个通用的消息，而实现的细节则由接受对象自行决定，这样同一消息就可调用不同的方法。多态性不仅增加了面向对象软件的灵活性，进一步减少了信息冗余，而且显著提高了软件的可重用性和可扩充性。多态有多种不同形式，其中参数多态和包含多态统称为通用多态，过载多态和强制多态则统称为特定多态。

动态绑定（Dynamic-binding）是多态性的基石之一。将函数调用与目标代码块的连接延迟到运行时进行，只有发送消息时才与接收消息实例的一个操作绑定。它与多态性可使建立的系统更灵活易于扩充。

5.1.3 消息与方法

1. 消息与消息通信

消息(Message)是向对象发出的服务请求,包含信息为:提供服务的对象标识、消息名、输入信息和回答信息。对象与传统的数据有本质区别,不是被动地等待外界对其进行操作,而是进行处理的主体,必须对其发消息请求以执行其某个操作,处理其私有数据,而不能从外界直接对其私有数据进行操作。

消息通信(Communication with messages)与对象的封装原则密切相关。封装使对象成为各司其职、互不干扰的独立单位;消息通信则为其提供唯一合法的动态联系途径,使其行为可以互相配合,构成一个有机的系统。

2. 方法

方法(Method) 指在对象内的操作。数据描述对象的状态,操作可操纵私有数据,改变对象的状态。当其他对象向该对象发出消息并响应时,其操作才得以实现。方法是类中操作的实现过程,一个方法包括方法名、参数及方法体。方法描述了类与对象的行为,每个对象都封装了数据和算法两个方面,数据由一组属性表示,而算法即是当一个对象接收到一条消息后,它所包含的方法决定对象如何动作。通常是在某种编程语言下实施的运算。

讨论思考

(1) 什么是对象及其三要素?什么是类及实例?

(2) 怎样理解继承及多态性?举例说明。

(3) 举例说明消息与方法及其之间的关系。

5.2 面向对象方法概述

5.2.1 面向对象方法的概念

从 20 世纪 70 年代末开始,传统的软件工程方法对克服“软件危机”,促进软件产业的发展起到重要作用。面向过程和面向数据的开发方法、软件项目的工程化管理、软件工具及开发环境,以及软件质量保障体系都极大地促进了软件工程技术及应用。

随着软件形式化方法及新型软件的开发,传统的软件工程方法的局限性逐渐显现;由于软件本质上是信息处理系统,传统的软件工程方法是面向过程的,将数据和处理过程分离,增加了软件开发的难度。其求解过程是先对应用领域(问题空间)进行分析,建立起问题空间的逻辑模型,再通过一系列复杂的转换和算法,构造软件系统获取解空间。由于问题空间与解空间的模型及描述方式不同,转换过程复杂,特别更难适应复杂系统和普遍存在的需求变化。而且,难以支持软件复用技术。

为了进一步提高软件研发效率、系统稳定性、可维护性和可重用性,20 世纪 80 年代末,出现了面向对象方法。**面向对象的方法**将软件系统看作一系列离散的解空间对象的

集合，并使问题空间的对象与解空间对象尽量一致，这些解空间对象相互之间发送消息相互作用，从而获得问题空间的解。因此，问题空间与解空间的结构、描述的模型一致，减少了软件系统开发的复杂度，使系统易于理解和维护。

面向对象方法(Object-Oriented Method，OOM)是面向对象技术和方法在软件工程中的全面运用，包括面向对象分析(Object-Oriented Analysis，OOA)、面向对象设计(Object- Oriented Design，OOD)、面向对象编程(Object-Oriented Program，OOP)、面向对象测试(OOT)和面向对象维护等方法，在此主要概述前两部分。

Coad 和 Yourdon 对**面向对象的定义**：面向对象＝对象＋类＋继承＋消息通信，具有这 4 个概念的软件开发方法称为**面向对象方法 OOM**。面向对象方法学的出发点和基本原则，使软件开发的方法和过程尽可能接近人类认识现实世界解决问题的方式方法和思维方式。只有同时使用对象、类、继承与消息通信，才能体现面向对象的特征和方法。

5.2.2 面向对象方法的特点

面向对象的开发方法 OOSD(Object-Oriented Software Development)的基本思想是尽可能按照人类认识世界的方法和思维方式分析和解决问题，可提供更加清晰的需求分析和设计，是指导软件开发的系统方法。OOSD 贯穿于整个软件生命期，其中面向对象的分析与设计是面向对象开发的关键。OOM 具有 4 个**主要特点**。

(1) 符合人类分析解决问题的习惯思维方式。传统软件开发方法对各阶段进行综合考虑面向过程，以算法为核心，将数据和过程相互独立，程序用于处理这些数据。以计算机的观点将数据和代码分离，忽视了数据和操作之间的内在联系，使以此方法设计的软件的解空间与问题空间不一致。而 OOM 以对象为核心，强调模拟现实世界中的概念而非算法，尽量用符合人类认识世界的思维方式渐进地分析解决问题，使问题空间与解空间一致，利于对开发过程各阶段综合考虑，有效地降低开发复杂度，提高软件质量。

(2) 各阶段所使用的技术方法具有高度连续性。传统的软件开发过程用瀑布模型描述，其主要缺点是将充满回溯的软件开发过程硬性地分割为几个阶段，而且各阶段所使用的模型、描述方法不相同。而 OOM 使用喷泉模型作为其工作模型，软件生存期各阶段无明显界限，开发过程回溯重叠，用相同的描述方法和模型保持连续。

(3) 开发阶段有机集成有利系统稳定。将 OOA、OOD、OOP 有机集成，始终围绕着建立问题领域的对象(类)模型进行开发过程，而各阶段解决的问题又各有侧重。由于构造软件系统以对象为中心，而不是基于对系统功能分解，当功能需求改变时不会引起其结构变化，使其具有稳定性和可适应性。

(4) 重用性好。利用复用技术构造新软件具有很大灵活性，由于对象所具有的封装性和信息隐蔽，使对象的内部实现与外界隔离，具有较强独立性，所以，对象类提供了较理想的可重用软件成分，而其继承机制使得 OO 技术实现可重用性更方便、自然和准确。

5.2.3 面向对象开发过程及范型

1. 面向对象开发过程

OOM 不仅是一些具体的软件开发技术与策略，而且是一整套处理软件系统与现实

世界的关系并进行系统构造的软件方法学。面向对象软件的开发过程与其他方法不同，从问题论域开始，历经从问题提出到解决的一系列过程。**开发过程中的步骤**如下。

(1) 分析阶段。分析阶段包括两个步骤：论域分析和应用分析。标识问题论域中的抽象，在分析时找到特定对象，基于对象的公共特性将其组合成集合，标识出对此问题的一个抽象，并标识抽象之间的关系，建立对象之间的消息连接。

① 论域分析。主要开发问题论域模型。在应用分析前进行论域分析，了解问题前对问题集思广益，考察问题论域内的较宽范围，分析覆盖范围应比直接要解决问题更广泛。

② 应用分析。应用(或系统)分析细化在论域分析阶段所开发的信息，并将注意力集中在当前要解决的问题。通过论域分析，分析人员可具有较宽论域知识，有助于更好抽象。

(2) 高层设计。在 OOD 中，软件体系结构设计与类设计常为同样过程，但还应将体系结构设计与类设计分开。在高层设计阶段，设计应用系统的顶层视图。如同开发一代表系统的类，通过建立该类的一个实例并发送给它一个消息以完成系统的“执行”。

(3) 开发类。主要依据高层设计所标识的对各类的要求和类的规格说明，进行类开发。由于一个应用系统通常是一个类的继承层次，对这些类的开发是最基本的设计活动。

(4) 建立实例。建立各对象的实例，实现问题的解决方案。

(5) 组装测试。在按照类与类之间的关系组装一个完整的应用系统的过程中进行测试。各类的封装和类测试的完备性可减少组装测试所需成本。

(6) 维护。维护的要求将影响应用和各类。继承关系可支持对现有应用的扩充，或加入新的行为，或改变某些行为的工作方式。

2. 面向对象的软件开发范型

(1) 改进了传统软件开发方法。在控制问题求解的规模和复杂度，提高软件系统的易理解性等方面结构化方法起到重要作用，但是，这种方法却很难实现软件重用，导致软件生产效率低下，质量难以保证且难以维护。

(2) 面向对象的软件开发方法按照同传统软件开发一样的步骤，同样要经历分析、设计、编码实现和测试的生命周期。在软件开发的生命周期中的每个阶段中都运用了面向对象的思想。面向对象技术导致了软件构件可以方便地被复用，尤其是基于程序构件的复用，通过利用组装可重用的构件快速地开发新软件系统。

(3) 大部分面向对象软件开发模型都包括以下内容。

① 分析用户的需求，提炼对象。

② 将现实中问题领域的对象抽象成计算机软件中的对象。

③ 分析并描述对象之间的关系。

④ 根据用户的需求，不断地修改并完善。

5.2.4 面向对象开发方法

1. OOSE 方法

面向对象软件工程(OOSE)方法是 1992 年 I. Jacobson 在其出版的专著《面向对象的

软件工程》中提出的。OOSE 方法采用五类模型建立目标系统，将面向对象的思想应用于软件工程中。这 5 类**模型**如下。

(1) 需求模型(Requirements Model，RM)。主要用于获取用户的需求、识别对象，主要的描述手段有用例图(Use case)、问题域对象模型及用户界面。

(2) 分析模型(Analysis Model，AM)。主要用于定义系统的基本结构。通过将 RM 中的对象，分别识别到 AM 中的实体对象、界面对象和控制对象三类对象中。每类对象都有各自的任务、目标并模拟系统的某个方面。

实体对象由使用事件确定，模拟在系统中需要长期保存并加以处理的信息，通常与现实生活中的一些概念符合。界面对象的任务是提供用户与系统之间的双向通信，在使用事件中所指定的所有功能都直接依赖于系统环境，它们都放在界面对象中。而控制对象的典型作用是将另外一些对象组合形成一个事件。

(3) 设计模型(Design Model，DM)。AM 只注重系统的逻辑构造，而 DM 需要考虑具体的运行环境，将在分析模型中的对象定义为模块。

(4) 实现模型(Implementation Model，IM)，即用面向对象语言 OOL 来实现。

(5) 测试模型(Testing Model，TM)。测试的重要依据是 RM 和 AM，测试的方法与技术同第 7 章软件测试介绍的类似，而底层是对类(对象)的测试。TM 实际上是一个测试报告。

OOSE 的开发活动主要分为分析、构造和测试 **3 个过程**，如图 5-4 所示。其中分析过程分为需求分析(Requirements analysis)和健壮分析(Robustness analysis)两个子过程，分析活动分别产生需求模型和分析模型。构造活动包括设计(Design)和实现(Implementation)两个子过程，分别产生设计模型和实现模型。测试过程包括单元测试(Unit testing)、集成测试(Integration testing)和系统测试(System testing)3 个过程，共同产生测试模型。

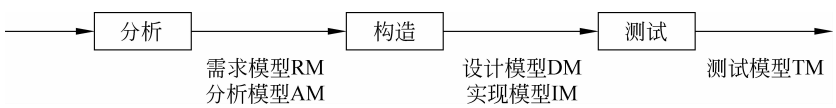

图 5-4　OOSE 的开发活动

用例(Use case)是 OOSE 中的重要概念，在开发各种模型时，用例是贯穿 OOSE 活动的核心，描述了系统的需求及功能。用例实际上是从使用者的角度来确定系统的功能，描述系统用户(也称为使用者)对于系统的使用情况。所以，应先分析确定系统的使用者，然后进一步考虑使用者的主要任务及使用的方式，识别所使用的事件，即用例。如图 5-5 所示，使用者以“人形”表示，“椭圆”表示用例，“大的矩形框”表示系统的边界。用“箭头线”连接使用者和用例或用例之间关系，表示使用者驱动事件的完成。用例之间通常有“使用”和“扩展”两种关系。

用例定义需求，提供了很好的需求分析策略和描述手段，弥补了以前的面向对象需求中的缺陷。另一重大贡献是定义了交互图，交互图对一组相互协作的对象，在完成一个 Use case 时执行的操作及它们之间传递的消息和时间顺序做了更精确的描述。

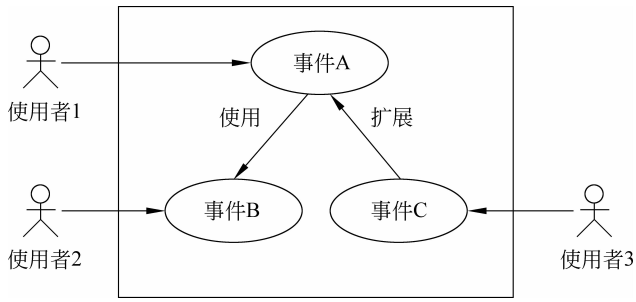

图 5-5　用例图

2. 常见的面向对象开发方法

目前,面向对象开发方法的研究已日趋成熟,已有很多面向对象产品问世。其开发方法有 Coad 方法、Booch 方法、OMT 方法和 UML 语言等。

(1) Booch 方法。Booch 最先描述了面向对象的软件开发方法的基础问题,指出面向对象开发是一种根本不同于传统的功能分解的设计方法。面向对象的软件分解更接近人对客观事务的理解,而功能分解只通过问题空间的转换来获得。

(2) Coad 方法。1989 年 Coad 和 Yourdon 提出了面向对象开发方法。该方法的主要优点是通过多年来大系统开发的经验与面向对象概念的有机结合,在对象、结构、属性和操作的认定方面,提出了一套系统的原则。该方法完成了从需求角度进一步进行类和类层次结构的认定。尽管 Coad 方法没有引入类和类层次结构的术语,但事实上已经在分类结构、属性、操作、消息关联等概念中体现了类和类层次结构的特征。

(3) OMT 方法。对象建模技术(Object Modeling Technique,OMT)是美国通用电气公司提出的一套系统开发技术。它以面向对象的思想为基础,通过对问题进行抽象,构造出一组相关的模型,从而能够全面地捕捉问题空间的信息。该方法是一种新兴的面向对象的开发方法,开发工作的基础是对真实世界的对象建模,然后围绕这些对象使用分析模型来进行独立于语言的设计,面向对象的建模和设计促进了对需求的理解,有利于开发得更清晰、更容易维护的软件系统。该方法为大多数应用领域的软件开发提供了一种实际的、高效的保证,努力寻求一种问题求解的实际方法。

(4) UML 语言。1995 年至 1997 年软件工程领域取得重大进展,其成果超过软件工程领域过去十多年的总和,最重要的成果之一是统一建模语言(Unified Modeling Language,UML)的出现。UML 成为 OO 技术领域内占主导地位的标准建模语言,是一种定义良好、易于表达、功能强大且普遍适用的建模技术和方法,融入了软件工程领域的新思想、新方法和新技术。其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全过程。不仅统一了 Booch 方法、OMT 方法、OOSE 方法的表示方法,而且对其作了进一步的发展,最终统一为大众接受的标准建模语言。将在后续内容中进一步概述。

讨论思考

(1) 面向对象包括哪些主要概念? 它们的具体含义是什么?

(2) 面向对象具有哪些特征?

(3) 面向对象的软件开发过程是怎样的?

5.3 面向对象分析

面向对象分析(OOA)的**目标**是获取用户需求并建立一系列问题域的精确模型,描述满足用户需要的软件。OOA 所建立的模型应表示出系统的数据、功能和行为 3 方面的基本特征。先要进行调研分析,在理解需求的基础上建立并验证模型。对复杂问题的建模,需要反复迭代构造模型,先构造子集、后构造整体模型。

5.3.1 面向对象分析的任务

1. 面向对象分析的原则

OOA 阶段是获取和描述用户需求并建立问题域对象模型的过程。分析系统中所含的所有对象及其相互间的关系,为建立分析模型,**OOA** 应遵照 5 个**基本原则**:一是建立信息域模型;二是描述功能;三是表达行为;四是划分功能、数据、行为模型,揭示更多的细节;五是以早期模型描述问题实质,以后期模型给出实现细节,这是 OOA 的基础。

2. 面向对象分析的任务

OOA 的**关键**是定义所有与待解决问题相关的类,包括类的操作和属性、类与类之间的关系以及它们表现出的行为,主要完成 6 项任务。

(1) 全面深入调研分析,掌握用户各项业务需求细节及来龙去脉。

(2) 准确标识类,包括定义其属性和操作。

(3) 认真分析定义类的层次关系。

(4) 明确表达对象与对象之间的关系(对象的连接)。

(5) 具体确定模型化对象的行为。

(6) 建立系统模型。反复运用前面的过程,通过上述分析,建立系统的 3 种模型:描述系统数据结构的对象模型,描述系统控制结构的动态模型,描述系统功能的功能模型。主要从不同侧面描述或表示系统的内容,以及相互影响、相互制约、有机地结合,全面表达对目标系统的需求。

5.3.2 面向对象分析的过程

OOA 是利用面向对象的概念和方法为软件需求建造模型,使用户需求逐步精确化、一致化、完全化的分析过程,也是提取需求的过程,**主要包括**理解、表达和验证 3 个**过程**。通常,由于现实世界中的问题较为复杂,分析过程中的交流又具有随意性和非形式化等特点,软件需求规格说明的正确性、完整性和有效性需要进一步验证,以便及时进行修正。

OOA 中建造的模型主要有对象模型、动态模型和功能模型 3 种。其关键是识别出问题域中的对象,在分析其之间相互关系基础上,建立问题域的简洁、精确和可理解的模型。对象模型常由 5 个层次组成:类与对象层、属性层、服务层、结构层和主题层,其层次对应

着 OOA 过程中建立对象模型的五项主要活动：发现对象、定义类、定义属性、定义服务、设计结构。面向对象分析过程如图 5-6 所示。

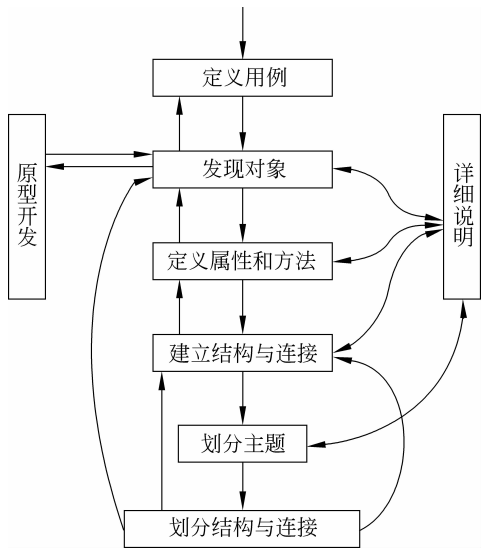

图 5-6　面向对象分析过程

5.3.3　对象建模技术

为了减少问题的复杂性，可利用模型将知识规范地进行表示。模型由一组图示符号和组织这些符号的规则组成，用于定义和描述问题域中的术语和概念。

对象建模技术(Object Modeling Technique，OMT)主要用于 OOA、系统设计和对象级设计。可将分析时获取的需求信息构建在对象模型、功能模型和动态模型三类模型中。各模型分别侧重系统的一个方面，从不同角度构成了对系统的完整描述，解决了对象模型定义“对谁做”，状态模型定义“何时做”，功能模型定义“做什么”的问题。

1. 建立对象模型

对象模型是 OOA 最关键的模型之一，主要描述系统中对象的静态结构、对象之间的关系、对象的属性和操作。利用包含对象和类的关系图表示，通过表示静态的、结构上的、系统的“数据”特征，为动态模型和功能模型提供基本框架。

建立对象模型时，首先确定对象和类，然后分析对象的类及其相互之间关系。对象类与对象间的关系可分为 3 种：一般-特殊(继承或归纳)关系、聚集(组合)关系和关联关系。对象模型用类符号、类实例符号、类的继承关系、聚集关系和关联等表示。有些对象具有主动服务功能，称为主动对象。对复杂系统，可划分主题画出主题图，有助于对问题的理解。

对象模型描述系统的静态结构包括：类和对象，它们的属性和操作，以及它们之间的关系。构造对象模型的目的是找出与应用程序密切相关的概念。