第 8 章　面向对象技术

8.1　面向对象技术概述

8.1.1　面向对象方法简介

面向对象的思想提倡运用人类的思维方式，从现实世界中存在的事物出发来构造软件。他建立在“对象”概念的基础上，以对象为中心，以类和继承为构造机制设计和构造相应的系统软件。面向对象方法(Object-Oriented-Menthod)是一种把面向对象的思想应用于软件开发过程中并指导开发活动的系统方法，简称 OO(Object-Oriented)方法。

面向对象方法起源于面向对象的编程语言。20 世纪 60 年代开发的 Simula 语言首次提出了对象的概念，并使用了类，也支持类的继承，它是第一个面向对象过程设计语言。最具有代表性和影响力的面向对象过程设计语言是由美国 Xerox(施乐)公司开发的 Smalltalk 语言，Smalltalk 语言全面实现了面向对象技术的机制，丰富了面向对象的概念，它的发布引起了人们对象面对象概念的广泛关注。随后产生了多种面向对象的程序设计语言，如 C＋＋和 JAVA 等。同事，面向对象的分析和设计方法也被广泛应用于软件开发中。

面向对象方法的优点如下：

1. 与人类习惯的思维方式一致

传统的程序设计技术室面向过程的设计方法，以算法为核心，把数据和过程作为相互独立的部分，数据代表问题空间中的客体，程序代码用于处理数据。这样的设计忽略了数据和操作之间的内在联系，问题空间和解空间并不一致。

面向对象技术以对象为核心，尽可能接近人类习惯的抽象思维方法，并尽量一致的描述问题空间和描述解空间。对象分类，从特殊到一般，建立类等级，获得继承等开发过程，符合人类认识世界，解决问题的过程。

2. 稳定性好

面向对象方法用对象模拟问题域中的实体，以对象间的联系刻画实体的联系。当系统的功能需求变化时不会引起软件结构的整体变化，只需做局部的修改。由于现实世界中的实体是相对稳定的，因而，以对象那个为中心构造的软件系统也比较稳定。

3. 可重用性好

面向对象技术可以重复使用一个对象类。如创建类的实例，直接使用类；又如派生一个

米娜组当前需要的新类，子类可以重用其父类的数据结构和程序代码，方便地进行修改和扩充，而且子类的修改并不影响父类的使用。

4. 较易开发大型软件产品

面向对象技术开发大型软件时，把大型产品看做是一系列相互独立的小产品，降低了开发时技术和管理方面的难度。

5. 可维护性好

由于面向对象的软件稳定性比较好，容易修改，容易理解，易于测试和调试，因而软件的可维护性好。

8.1.2 面向对象的基本概念

1. 对象(object)

在应用领域中有意义的，与所要解决问题有关系的任何事物都可以作为对象，它既可以是具体的物理实体的抽象，可以是人为的概念，也可以是任何有明确边界和意义的东西。(例如，一名学生，一家公司，一本图书，借书，还书等。)总之，对象是问题域中某个实体的抽象，设立某个对象就反映了软件系统保存有关它的信息并具有与它进行交互的能力。

由于客观世界中的实体通常都既具有静态的属性，又具有动态的行为，因此，面向对象方法学中的对象是由描述该对象静态属性的数据与描述该对象动态行为的操作封装在一起构成的统一体。一个对象通常由对象名、属性和操作三个部分组成。

使用对象时，需要注意以下几点：

(1)对象的数据时封装起来的，对数据的处理需要通过特定的操作；

(2)对象之间通过传递消息进行通信，不同的对象独立地处理自身的数据；

(3)要处理对象的内部数据时，外界需要通过接口向对象发送消息，请求执行特定的操作。

2. 类(class)

现实世界中存在的客观事物有些是相似的，例如，张三、李四、王五……虽说每个人职业、性格、爱好、特长等各有不同，但是，他们的基本特征是相似的，都是黄皮肤、黑头发、黑眼睛，于是人们把他们统称为“中国人”。人类习惯于把有相似特征的事物归为一类，分类是人类认识客观世界的基本方法。

在面向对象技术中，“类”就是对具有相同属性和相同操作的一组相似对象的集合。也就是说，类是对具有相同属性和行为的一个或多个对象的描述。

谈到类的概念，就必须知道什么是类的实例。实例是由某个特定的类描述的一个具体的对象。

如图 8-1 所示，“轿车”是一个类，“轿车”类的实例“张三的轿车”、“李四的轿车”都是对象。类是一个支持继承的抽象数据类型，而对象就是类的实例，类和对象的关系类似于程序设计语言中的数据类型和变量之间的关系。

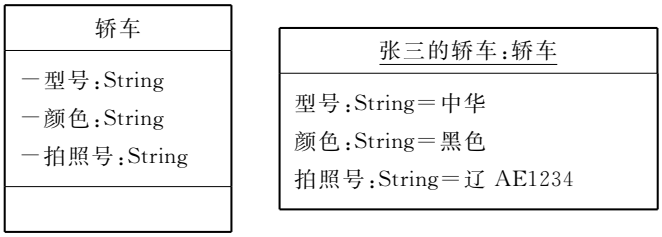


图 8-1　类及实例

3. 消息(Message)

消息，就是要求某个对象执行在定义它的那个类中所定义的某个操作的规格说明。通常一个消息由下述三个部分组成：

(1)对象名：接收消息的对象。

(2)消息名：要求接收对象完成度操作。

(3)参数：纸箱操作时的参数或者操作返回的结果。

例如，MyCircle 是一个半径 4cm、圆心位于(100，200)的 Circle 类的对象，也就是 Circle 类的一个实例，下面是 MyCircle 发送的一条消息：

MyCircle. Show(GREEN)；

其中，MyCircle 是接收消息的对象名称，Show 是要求对象执行的操作名称，圆括号内的 GREEN 是消息的操作参数。当 MyCircle 对象接受到这个消息后，将执行在 Circle 类中所定义的 Show 操作。

4. 方法(Method)

方法，就是对象所能执行的操作，也就是类中所定义的服务。方法描述了对象执行操作的算法和响应消息的方法。在面向对象程序语言中把方法称为成员函数。

5. 属性(Attribute)

属性，就是类中所定义的数据，他是对客观世界实体所具有的性质的抽象。类的每个实例都有自己特有的属性值。在面向对象程序语言中把属性称为数据成员。

6. 封装(Encapsulation)

从字面上解释，所谓封装就是把某个事物包起来，使外界不知道该事物的具体内容。在面向对象的程序中，我们把数据和实现操作的代码集中起来放在对象内部，一个对象好像是一个不透明的黑盒子，表示对象状态的数据和实现各个操作的代码与局部数据都被封装在黑盒子里，从外面是看不见的，更不能从外面直接访问或修改这些数据及代码。

使用一个对象的时候，只需要知道他向外界提供的接口形式而无需知道它的数据结构细节和实现操作的算法。

实现封装性的条件如下：

(1)有一个清晰的边界。所有私有数据和实现操作的代码都被封装在这个边界内，从外面看不见更不能直接访问。

(2)有确定的接口(即协议)。这些接口就是对象可以接收的消息，只能通过向对象发送消息来使用它。

(3)受保护的内部实现。实现对象功能的细节(私有数据和代码)不能再定义该对象的

类的范围外进行访问。

封装性也就是信息隐藏，通过封装把对象的实现细节对外界隐藏起来了。

7. 继承(Inheritance)

一个类可以定义为另一个更一般的类的特殊情况，如“轿车”类是“汽车”类的特殊情况，这时可以称一般类是特殊类的父类，特殊类是一般类的子类。如“汽车”类是“轿车”类的父类，“轿车”类是“汽车”类的子类。同样“汽车”类还可以是“交通工具”类的子类，“交通工具”类是“汽车”类的父类。这样可以形成类的一般--特殊的层次关系，如图 8-2 所示。

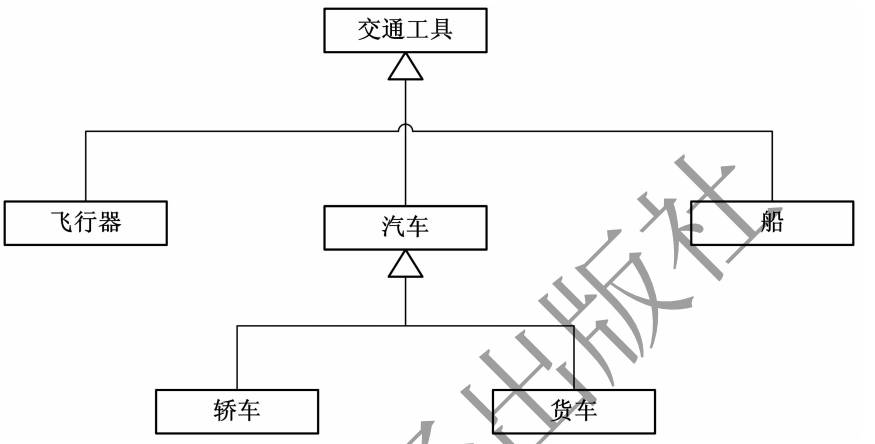


图 8-2 一般——特殊的关系

在这一般——特殊的关系汇总，子类可以继承其父类的所有属性和操作，同事子类中还可以定义自己特有的属性和操作。所有子类的属性和操作是子类中的定义部分和气父类中的定义部分的总和。

继承是类间的一种基本关系，是基于层次关系的不同类共享数据和操作的一种机制。父类中定义了其所有子类的公共属性和操作，在子类中出了定义自己特有的属性和操作外，还可以重新定义父类中操作的实现方法，称为重载(override)。例如，举行时多边形的子类，在多边形类中定义了属性：顶点坐标序列，定义了操作：平移、旋转、显示、计算面积等。在矩形类中，可定义自己的属性长和宽，还可以对操作“计算面积”重新定义。

8. 多态性(Polymorphism)

多态性是指同一个操作作用于不同的对象上可以有不同的解释，并产生不同的执行结果。例如，“画”操作，作用在“矩形”对象上，则在屏幕上画一个矩形，作用在“圆”对象上，则在屏幕上画一个圆。也就是说，相同操作的消息发送给不同对象时，每个对象将根据自己所属类中定义的这个操作去执行，从而产生不同的结果。

9. 重载(Overloading)

有两种重载：函数重载是指同一作用域内的若干个参数特征不同的函数可以使用相同的函数名字；运算符重载是指同一个运算符可以施加于不同类型的操作数上面。当然，当参数特征不同或被操作数的类型不同时，实现函数的算法或运算符的语义是不同的。

重载进一步提高了面向对象系统的灵活性和可读性。

8.2　UML 建模

统一建模语言(unified modeling language,UML)是用一组专用符号描述软件模型的语言,这些符号统一、直观、规范、可以用任何软件开发过程。

8.2.1　UML 简介

UML 是一种标准的图形化(即可视化)建模语言。它适用于系统开发的全过程,用定义完善的符号来图形化地展现一个软件系统。UML 的应用可以贯穿于软件开发周期的每一个阶段,适用于数据建模、业务建模、对象建模和组件建模。UML 作为一种建模语言,并不涉及编程问题,即与语言平台无关,这就使开发人员可以专注于建立软件系统的模型和结构。

现在,UML 已经成为一个事实上的工业化标准。不论在计算机学术界、软件产业界还是在商业界,UML 逐渐成为人们为各种系统建模、描述系统体系结构、商业体系结构和商业过程时用的统一工具,而且在实际过程中人们还在不断扩展它的应用领域。

UML2.0 定义了 13 种图,比 UML1.0 新增了三种。表 8-1 列出了这 13 种图的功能。

表 8-1　　UML2.0 中定义的图型

图名	功能	备注
类图	描述类、类的特性以及类之间的关系	UML1.0 原有
对象图	描述对象的特征以及对象之间的关系	UML1.0 非正式图
复合结构图	描述类的运行时刻的分解	UML2.0 新增
构件图	描述构件的结构与连接	UML1.0 原有
部署图	描述在各个节点上的部署	UML1.0 原有
包图	描述编译时的层次结构	UML1.0 非正式图
用例图	描述用户与系统如何交互	UML1.0 原有
活动图	描述过程行为与并行行为	UML1.0 原有
状态机图	描述事件如何改变对象生命周期	UML1.0 原有
顺序图	描述对象之间的交互,重点在于强调顺序	UML1.0 原有
通信图	描述对象之间的交互,重点在于连接	UML1.0 中的协作图
时间图	描述对象之间的交互,重点在于时间	UML2.0 新增
交互概述图	是一种顺序图与活动图的混合	UML2.0 新增

如图 8-3 所示是从使用角度将 UML2.0 的 13 种图分为结构和行为图,其中共有六种结构图和七种行为图。结构图也称为静态模型图,主要是用来强调系统的建模,它包括类图、对象图、复合结构图、构件图、部署图和包图。行为图也称为动态模型图,主要用来强调系统模型中触发的事件,它包括用例图、交互图、活动图和状态机图。其中交互图用来强调系统

模型中的对象流程，是顺序图、通信图、时间图和交互概述图的统称。

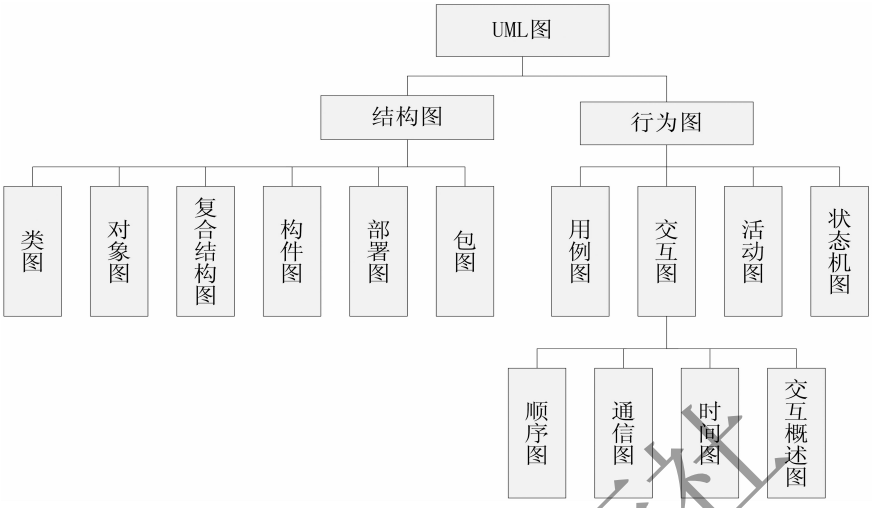


图 8-3　UML2.0 中的图形分类

结构图中比较常用的图有类图和对象图。行为图中比较常用的有用例图、顺序图、状态机图和活动图。

8.2.2　用例图

用例图向外部用户展示了软件系统的行为。

1. 用例模型概述

用例模型描述的外部参与者所理解的系统功能。用例模型用于需求分析阶段，它的建立是系统开发者和用户反复讨论的结构，描述了开发者和用户对需求规格达成的共识。首先，它描述了待开发系统的功能需求：其次，他把系统看作黑盒子，从外部参与者的角度来理解系统；第三，它参与了需求分析之后各阶段的开发工作，不仅在开发过程中保证了系统所有功能的实现，而且被用于验证和检测所开发的系统，从而影响到开发工作的各个阶段和UML 的各个模型。

2. 用例的组成

一个用例由以系统边界、参与者、用例和关系要素构成，下面分别介绍：

(1)系统边界

系统边界是定义由谁或什么(即，参与者)使用系统，系统能够为哪些参与者提供什么特定利益(即，用例)。

系统边界在 UML 中绘制为方框，用系统的名称作为标签，参与者绘制在边界外部，用例绘制在边界内部。如图 8-4 所示，一个棋盘管理系统的边界如下图所示。

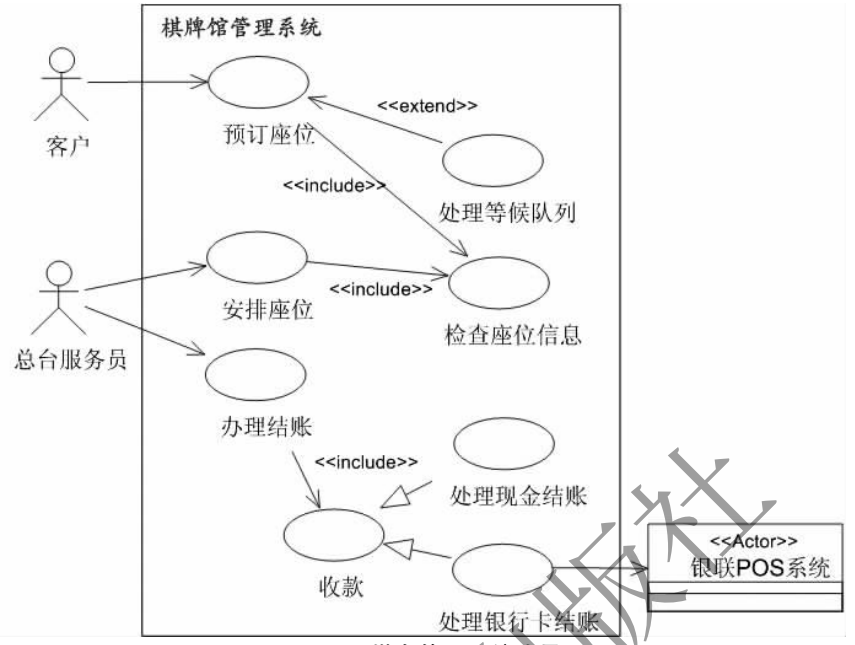


图 8-4　棋盘管理系统边界

(2)参与者(活动者,Actor)

①参与者的表示

参与者是与系统交互的人或物,它代表外部实体,例如用户,硬件设备或与本系统交互的另一个软件系统。使用用例并与系统交互的任何人或物都是参与者。如图 8-5 所示。

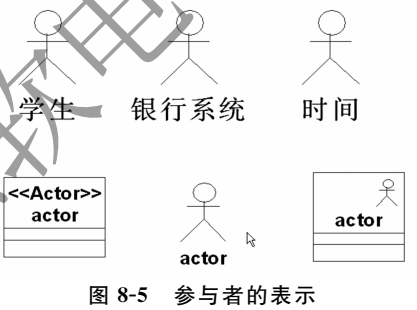


图 8-5　参与者的表示

参与者是一个群体概念,代表的是一类能够使用某个功能的人或物,而不是某个个体。例如,在自动售货机系统中,使用售货功能的人可以是张三,也可以是李四,但是张三或李四这样的个体对象并不能称为参与者。事实上,一个具体的人在系统中可以充当多种不同的角色。例如,某个人既可以为售货机添加物品(执行供货功能),又可以把售货机中的钱取走(执行取货款功能)。

②参与者间的关系

如果一个角色的操作是由另一个角色代理完成的,先建立该角色到另外角色之间的依赖。如下图所示,角色之间的关系如图 8-6 所示。

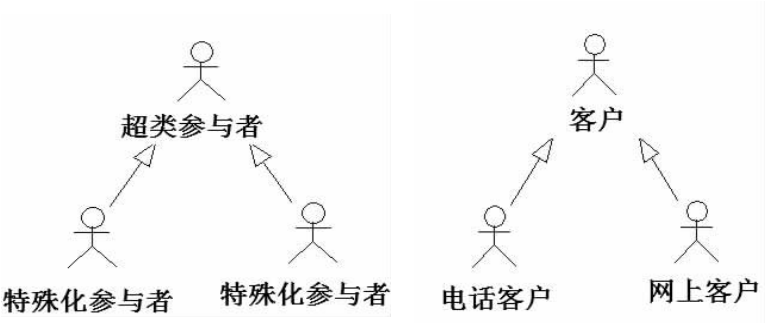


图 8-6 角色之间的关系

③怎样识别参与者

实践表明,参与者对确定用例是非常有用的,面对一个大型、复杂的系统,要列出用例清单往往很困难,这时可以先列出参与者清单,在针对每个参与者列出它的用例。对于参与者通过以下几个问题来确认。

谁向系统提供信息?

谁从系统获取信息?

谁操作系统?

谁维护系统?

系统使用哪些外部资源?

系统是否和已经存在的系统交互?

(3)用例(Use Case)

一个用例实质上是用户与计算机系统之间的一次典型的交互作用,它代表的是系统的一个完整的功能。在 UML 中把用例定义成系统执行的一系列动作,动作的结构能被外部参与者察觉到。

在 UML 用例图中,用例表示为一个椭圆。图 8-7 是自动售货机系统的用例图。其中“售货”、“供货”和“取货款”都是典型的用例。

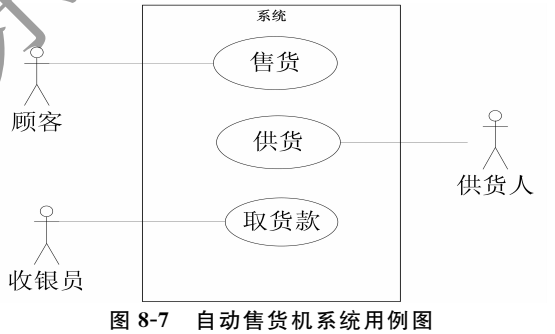


图 8-7 自动售货机系统用例图

概括地说,用例具有以下特点:

①用例代表某些用户可见的功能,实现一个具体的用户目标;

②用例由参与者激活,并提供确切的值给参与者;

③用例可大可小,但它必须是对一个具体的用户目标实现的完整描述。

注意:用例是一个类,它代表一类功能而不是使用该功能的某个具体事例。用例的实例

是系统的一种实际使用方法，通常把用例的实例称为脚本。脚本是系统的一次具体执行过程。例如，在自动售货机系统中，张三投入硬币购买矿泉水，系统收到钱后把矿泉水送出来，这个过程就是一个脚本。

(4)关系(Relationship)

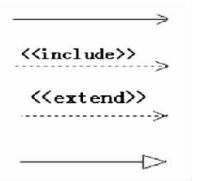
用例图中有以下四种基本关系：

关联(association)

包含(include)

扩展(extend)

泛化(generalization)



①关联关系

描述参与者与用例之间的关系，用单向箭头，表示谁启动用例，每个用例都有角色启动，除包含和扩展用例；客户和转账之间是关联关系，如图 8-8 所示。

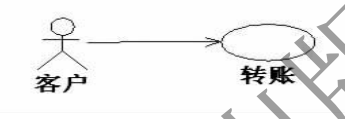


图 8-8　关联关系

②包含关系

是指两个用例之间的关系。其中一个用例(基本用例，base use case)的行为包含了另一个用例(包含用例，inclusion use case)的行为。如图 8-9 所示。

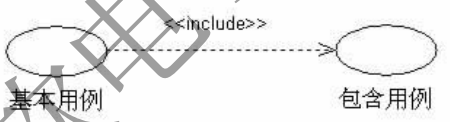


图 8-9　包含关系

执行基用例时，必须执行被包含用例，被包含用例也可单独执行。例如在图 8-10 中，取消订单用例必须先执行查询订单用例，所以这两个用例是包含关系。

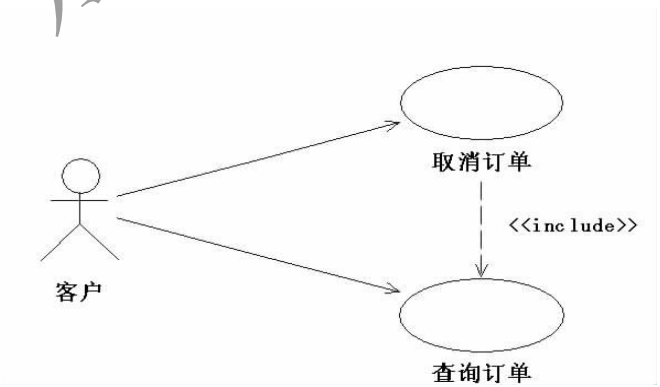


图 8-10　包含关系

如果一个用例的功能太多时，可用包含关系建模成两个或多个小用例。如图 8-11 所示，学生信息管理用例有分解成添加学生记录、删除学生记录和修改学生记录用例。

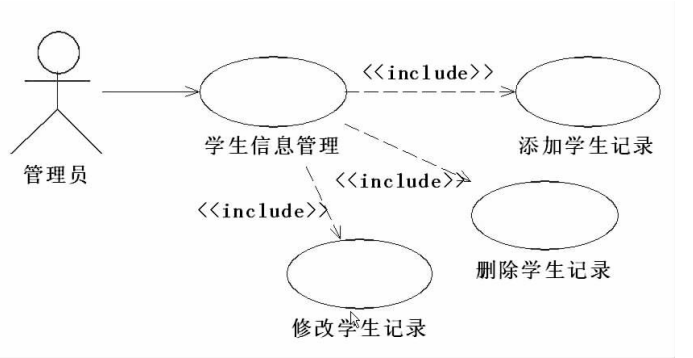


图 8-11 包含关系

③扩展关系

扩展关系是指两个用例之间的关系。一个用例可以被定义为基础用例的增量扩展，称作扩展关系。扩展关系是把新的行为插入到已有用例中的方法。

基础用例即使没有扩展用例也是完整的。一般情况下基础用例的执行不会涉及扩展用例，只有特定条件发生，扩展用例才被执行。如图 8-12 是基本用例和扩展用例的关系。

图 8-12 扩展关系

例：如图 8-12 所示订货购物用例和 VIP 打折用例之间的关系为扩展关系。

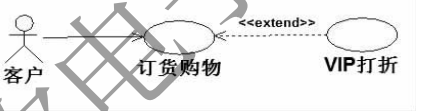


图 8-12 扩展关系示例

④泛化关系

一个用例和几种情形的用例间构成泛化关系。往往父用例表示为抽象用例，任何父用例出现的地方子用例也可出现。如图 8-13 所示。电话预订用例和网上预订用例是父用例预订的具体用例，之间是泛化关系。

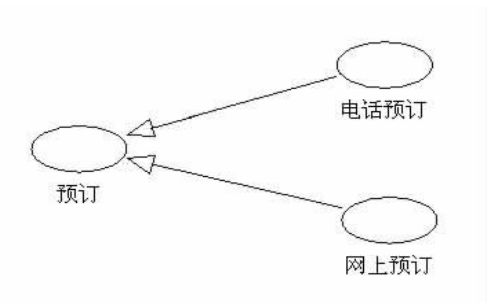


图 8-13 泛化关系

3. 建立用例模型

几乎在任何情况下都需要使用用例，通过用例可以获取用户需求，规划和控制项目。获取用例是需求分析阶段的主要工作之一，而且是首先要做的工作。大部分用例将在项目的

需求分析阶段产生，并且随着开发工作的深入还会发现更多用例，这些新发现的用例都应及时补充进已有的用例集中。用例集中的每个用例都是系统的一个潜在的需求。

(1)发现参与者

为获取用例首先要找出系统的参与者，可以通过请系统的用户回答一下问题的办法来发现参与者：

①谁将使用系统的主要功能？

②谁来维护和管理系统？

③系统控制哪些硬件设备？

④系统需要与哪些其他的系统交换？

(2)获取用例

事实上，从识别参与者起获取用例的过程就已经开始了。一旦识别出了参与者，就可以对每个参与者提出问题以获取用例：

①参与者需要从系统中获得何种功能？

②参与者需要读取、产生、删除、修改、或存储系统中的某种信息吗？

③系统中发生的事件需要通知参与者吗？

④系统需要何种输入/输出？

例：有一业务需求列表如下，要求为其构建一个用例图。

①系统可以供教师使用来为学生记录成绩

②系统根据需要创建报告卡

③系统允许用户浏览记录的成绩

第一步：首先需要询问业务需求的提出者以获取更多的信息。

提问 1：教师可以对已经输入的信息进行更新吗？

回答：可以！

提问 2：谁来创建报告卡，是教师吗？

回答：不！有一位管理人员来做这项工作。

提问 3：报告卡创建后，还可以对它做些什么工作？

回答：在报告卡创建后，管理人员要检查其准确性。当报告卡核准后，教师应该通过计算机分发报告卡。

提问 4：谁需要浏览成绩？

回答：教师和学生。

第二步：确定系统的边界范围，找出系统中的参与者和用例

通过访谈，会得出一个修改过的新的系统需求列表。

①系统可以供教师使用来为学生记录并更新成绩。

②系统根据需求由管理人员创建报告卡，管理人员要检查报告卡的准确性。

③教师需要通过计算机分发报告卡。

④系统允许教师和学生浏览记录的成绩。

由此可得出系统的参与者及用例。

参与者：教师、学生、管理员

用例：记录成绩、更新成绩、生成报告卡、检查报告卡的准确性、分发报告卡、浏览成绩。

第三步：细化每个用例

对“记录成绩”用例进行细化，下面是该用例的主事件流。

①教师确定出要记录哪些学生的成绩。

②系统要确保学生在数据库中。

③教师说明要记录哪项作业的成绩。

④系统开始数据库的一项事务处理。

⑤系统为学生把作业加入数据库。

⑥教师输入学生作业的成绩。

⑦系统核对输入的成绩以确保其属于正确的范围。

⑧系统记录作业的成绩。

⑨系统结束事务的处理。

⑩系统提示教师成绩已经记录。

细化过程中可添加新发现的用例，并根据优先级重新排列。

①登录

②保存成绩(save grades)

③记录成绩 (record grades)

④加载成绩 (load grades)

⑤浏览成绩 (view grades)

⑥更新成绩 (update grades)

⑦生成报告卡(generate grades)

⑧分发报告卡(distribute report cards)

第三步：建立用例模型结构，建立用例图如图 8-14 所示。

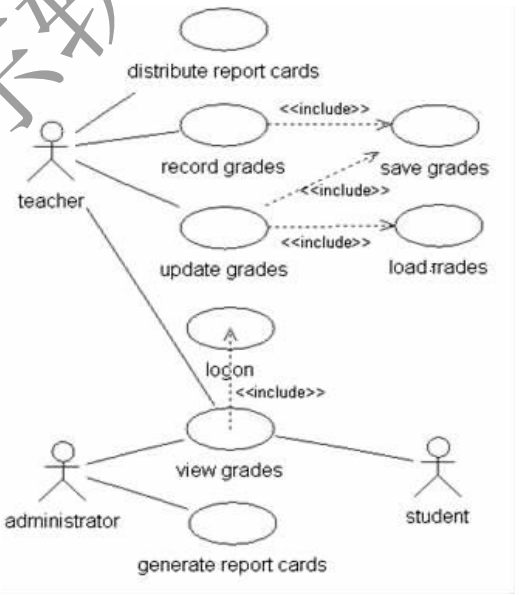


图 8-14 系统用例图

8.2.3　类图

数千年以前，人类就已经开始采用分类的方法有效地简化复杂问题已了解客观世界。在面向对象建模的技术中，使用同样的方法将客观的实体映射为对象，并归纳成一个个类。类、对象和他们之间的关联是面向对象技术中最基本的元素。对于一个想要描述的系统，其类模型和对象模型揭示了系统的结构，指明了一组对象的属性和行为。

1. 类图概述

类图是描述类、协作(类或对象间的协作)接口及其关系的图。是逻辑视图的重要组成部分，用于对系统的静态结构建模，涉及到具体的实现细节。

在系统分析阶段，类图主要用于显示角色和提供系统行为的实体的职责；

在系统设计阶段，类图主要用于捕捉组成系统体系结构的类结构；

在系统编码阶段，根据类图中的类及它们之间的关系实现系统的功能。

2. 类图的组成

类图(Class Diagram)描述类和类之间的静态关系。与数据模型不同，它不仅显示了信息的结构，同时还描述了系统的行为。类图是定义其他图的基础，在类图的基础上，状态图、协作图等进一步描述了系统其他方面的特性。

类描述一类对象的属性和行为。在 UML 中，类的可视化表示为一个划分成三个格子的长方形(下面两个格子可以省略)。其中，上面的格子中包含类名，中间的格子中包含属性，下面的格子中包含类的操作，如图 8-15 所示。

图 8-15　类的图形符号

类图描述了类和类之间的静态关系。定义了类之后，就可以定义类之间的各种关系了。类与类之间通常有关联、泛化(继承)、依赖和细化等关系。

(1)关联关系

关联表示两个类之间存在某种语义上的联系。例如，学生使用计算机，我们就认为在学生和计算机之间存在某种语义上的联系，因此，在类图中应该在学生类和计算机类之间建立关联关系。

①普通关联

普通关联是最常见的关联关系，只要在类与类之间存在链接关系就可以用普通关联表示。普通关联的图示符号是链接两个类表之间的直线，如图 8-16 所示，图 8-16 所示的关联是双向的，可在一个方向上为关联起一个名字(也可不起)。为避免混淆，在名字前面(或后面)加一个表示关联方向的黑三角。

图 8-16　普通关联

如果关联是单向的，则成为导航关联，其符号是用实际箭头链接两个类。仅在箭头所指的方向上才有这种关联关系，如图 8-17 所示，图中只表示某人可以拥有汽车，但汽车被人拥有的情况没有表示出来。

图 8-17　导航关联

在类图中还可以表示关联中的数量关系，即参与关联的对象的个数或数量范围。例如

0..1 表示 0 到 1 个对象

0... * 或 *　　表示 0 到多个对象

1..15 表示 1 到 15 个对象

3 表示 3 个对象

图 8-16 表示一个学生可以使用 1 到多台计算机，一台计算机可被 0 至多个学生使用。

②聚集

剧集也成为集合，是关联的特例。聚集表示的是类与类之间是整个与部分的关系。在陈述需求时使用的“包含”、“组成”、“分为……部分”等字句，往往意味着存在聚集关系。除了一般聚集之外，还有两种特殊的聚集关系，分别是共享聚集和复合聚集。

a. 共享聚集如果在聚集关系中处于部分方的对象可同时参与多个处于整体方对象的构成，则该聚集称为共享聚集。例如，一个社团包含许多学生，每个学生有可以使另一个社团的成员，则社团和学生之间是共享聚集关系，如图 8-18 所示。一般聚集和共享聚集的图示符号，都是在表示关联关系的直线末端紧挨着整体类的地方画一个空心菱形。

图 8-18　共享聚集

b. 复合聚集

如果部分类完全隶属于整体类，部分与整体共存，整体不存在了部分也会随之消失（或失去存在价值了），则该聚集称为复合聚集（简称为组成）。例如，我们在屏幕上打开一个窗口，它由文本框、列表框、按钮和菜单组成，一旦关闭了窗口，各个组成部分也同时消失，窗口和他的组成部分之间就存在着复合聚集关系，如图 8-19 所示。组成关系用实心菱形表示。

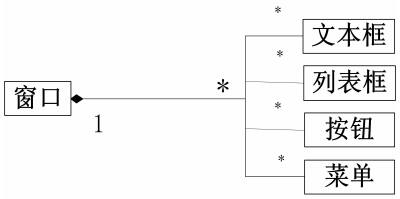


图 8-19　复合聚集

(2)泛化关系

UML 中的泛化关系就是通常所说的继承关系，它是通用元素和具体元素之间的一种分类关系。具体元素完全拥有通用元素的信息，并且还可以附加一些其他信息。如图 8-20 所

示，汽车类包含客车与货车两个子类，它们之间具有泛化关系。

在 UML 中，用一端为空心三角形的连线表示泛化关系，三角形的顶角紧挨着通用元素。

注意，泛化针对类型而不针对实例，一个类可以继承另一个类，但一个对象不能继承另一个对象。

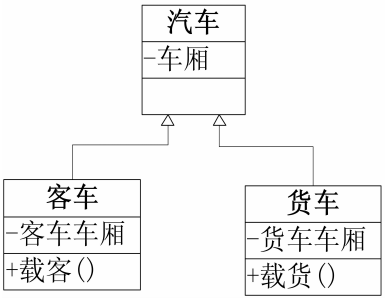


图 8-20　泛化关系

(3)依赖关系

依赖关系描述两个模型元素(类、用例等)之间的语义连接关系；其中一个模型元素是独立的，另一个模型元素不是独立的，它依赖于独立的模型元素，如果独立的模型元素改变了，将影响依赖于它的模型元素。例如，一个类使用另一个类的对象作为操作的参数，一个类用另一个类的对象作为它的数据成员，一个类向另一个类发消息等，这样的两个类之间都存在依赖关系。

在 UML 的类图中，用带箭头的虚线连接有依赖关系的两个类，箭头指向独立的类。在虚线上可以带一个版本的标签，具体说明以来的种类。例如，如图 8-21 表示一个依赖关系，该关系使得 B 类的操作可以使用 A 类中私有的或保护的成员。

图 8-21　依赖关系

3. 建立类图

建立类图主要遵守以下步骤：

(1)分析问题域，确定需求；

(2)寻找类，确定类的含义和职责；

(3)定义类的属性和操作；

(4)确定类之间的关系；

(5)精化类和类间的关系；

(6)绘制类图。

在以上步骤中重点是如何寻找类：

UML 中类有三种主要的版型：边界类、控制类和实体类。

(1)边界类：边界类位于系统与外界的交界处，包括：

①用户界面类，如：窗口、对话框、报表类等

②通讯协议类，如：TCP/IP 的类

③直接与外部设备交互的类

④直接与外部系统交互的类

边界类如图 8-22 所示。

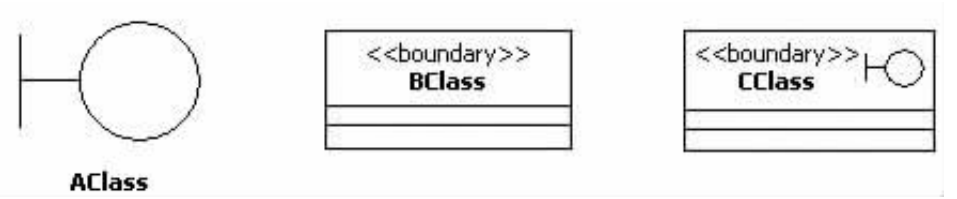


图 8-22 边界类 1

通过用例图可以确定需要的边界类，每个 Actor/User case 对至少需要一个边界类。

边界类一般可以没有属性，只有操作。但并不是每个 Actor/Use case 都需要生成惟一边界类，多个 actor 启动同一 use case 可以使用同一边界类。如图 8-23 所示。

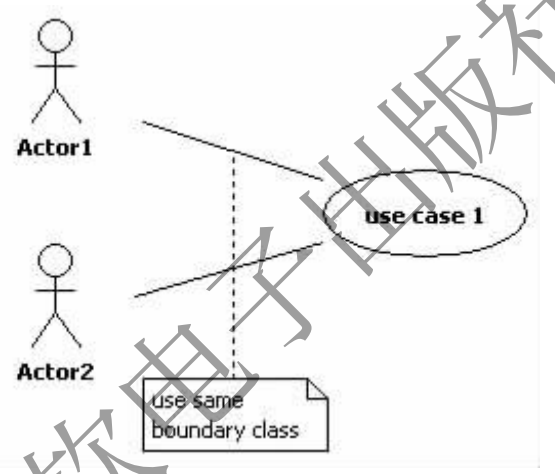


图 8-23 边界类 2

(2)实体类：实体类保存要放进持久存储体(数据库/文件等)的信息。如图 8-24 所示。

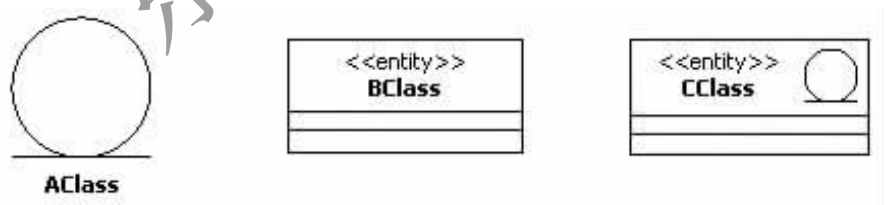


图 8-24 实体类

实体类采用目标领域术语命名。

通常实体类对应数据库中的表，其属性对应表的字段，但实体类与数据库中的表不一定是一一对应关系。

(3)控制类是负责管理或控制其他类工作的类。控制类如图 8-25 所示。

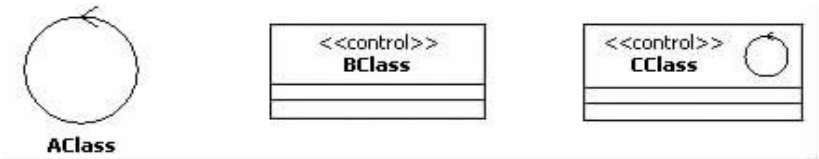


图 8-25　控制类

每个用例通常有一个控制类，控制用例中的事件顺序，控制类也可以在多个用例间共用。控制类较少接收消息，发出较多消息。

实例：通过以下实例来说明如何建立类图。

张明是一个爱书之人，家里各类书籍已过千册，而平时又时常有朋友外借，因此需要一个个人图书管理系统。该系统应该能够将书籍的基本信息按计算机类、非计算机类分别建档，实现按书名、作者、类别、出版社等关键字的组合查询功能。在使用该系统录入新书籍时系统会自动按规则生成书号，可以修改信息，但一经创建就不允许删除。该系统还应该能够对书籍的外借情况进行记录，可对外借情况列表打印。另外，还希望能够对书籍的购买金额、册数按特定时间周期进行统计。

第一步：发现类：下面标注下划线的为已发现的类。

张明是一个爱书之人，家里各类书籍已过千册，而平时又时常有朋友外借，因此需要一个个人图书管理系统。该系统应该能够将书籍的基本信息按计算机类、非计算机类分别建档，实现按书名、作者、类别、出版社等关键字的组合查询功能。在使用该系统录入新书籍时系统会自动按规则生成书号，可以修改信息，但一经创建就不允许删除。该系统还应该能够对书籍的外借情况进行记录，可外借情况列表打印。另外，还希望能够对书籍的购买金额、册数按特定时间周期进行统计。

第二步：筛选修改已选类

“计算机类”、“非计算机类”是该系统中图书的两大分类，因此应该对其建模，并改名为“计算机类书籍”和“非计算机类书籍”，以减少歧义；

“外借情况”则是用来表示一次借阅行为，应该成为一个候选类，多个外借情况将组成“外借情况列表”，而外借情况中一个很重要的角色是“朋友”这一借阅主体。虽然到本系统中并不需要建立“朋友”的资料库，但考虑到可能会需要列出某个朋友的借阅情况，因此还是将其列为候选类。为了能够更好地表述，将“外借情况”改名为“借阅记录”，而将“外借情况列表”改名为“借阅记录列表”。

“购买金额”、“册数”都是统计的结果，都是一个数字，因此不用将其建模，而“特定时限”则是统计的范围，也无需将其建模；不过从这里的分析中，我们可以发现，在该需求描述中隐藏着一个关键类“书籍列表”，也就是执行统计的主体。

第三步：得到候选类

书籍（Book）、计算机类书籍（BookList）、非计算机类书籍（OtherBook）、借阅记录（BorrowRecord）、借阅记录列表（BorrowList）、书籍列表（ItBook）。

在使用“名词动词法”寻找类的时候，很多团队会在此耗费大量的时间，特别是对于中大型项目，这样很容易迷失方向。其实在此主要的目的是对问题领域建立概要的了解，无需太过咬文嚼字。

第四步：关联分析，建模，如图 8-26 所示。

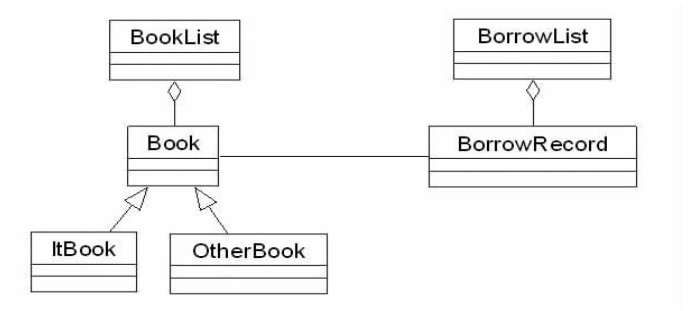


图 8-26　关联分析

第五步：多重性分析，再建模。如图 8-27 所示。

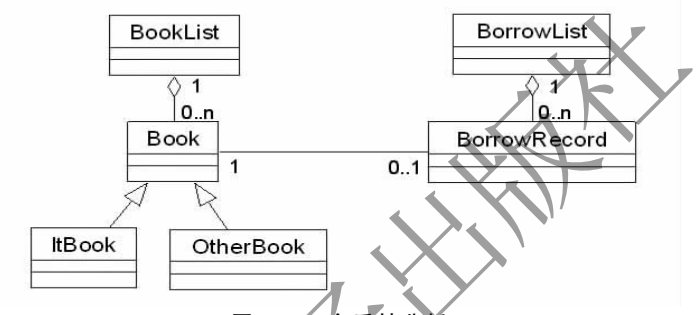


图 8-27　多重性分析

第六步：职责分析

书籍类：从需求描述中，可找到书名、类别、作者、出版社；同时从统计的需要中，可得知“定价”也是一个关键的成员变量。

书籍列表类：书籍列表就是全部的藏书列表，其主要的成员方法是新增、修改、查询（按关键字查询）、统计（按特定时限统计册数与金额）。借阅记录类：借阅人（朋友）、借阅时间。借阅记录列表类：主要职责就是添加记录（借出）、删除记录（归还）以及打印借阅记录 。最终如图 8-28 所示。

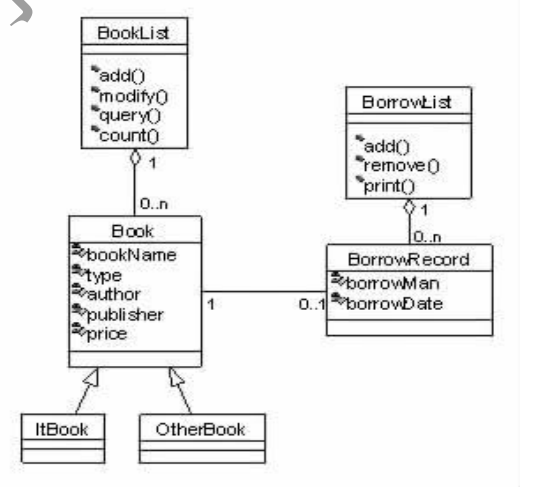


图 8-28　职责分析

8.2.4　对象图

对象是类的实例，对象之间的连接时类之间关联的实例，因此，对象图可以看做是类图的实例。

在 UML 中，对象图与类图具有几乎完全相同的表示形式，主要差别是对象的名字下面要加一条下画线。

对象名有下列三种表示格式：

第一种格式形如对象名:类名，即对象名在前，类名在后，中间用冒号连接。

第二种格式形如:类名，这种格式用于尚未给对象命名的情况，注意，类名前的冒号不能省略。

第三种格式形如对象名，这种格式省略类名。

例如，图 8-16 表示学生类与计算机类之间的关联关系，图 8-29 表示学生类中的对象实例“张林”与计算机类中的对象实例“三机房 8 号机”、“五机房 8 号机”之间的关联关系。

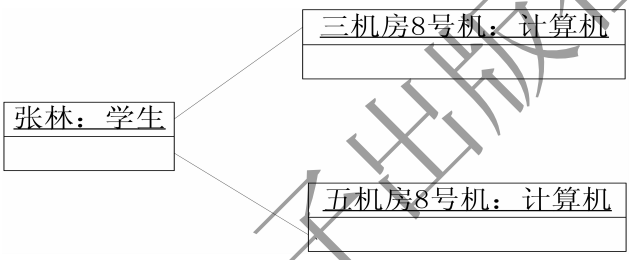


图 8-29　对象图

8.2.5　顺序图

1. 顺序图概述

类图描述了系统中的类以及类间的关系，但是没有详细说明对象的行为，也没有详细说明对象之间如何交互。

顺序图又称序列图，描述对象之间的动态交互关系，着重表现对象间消息传递的时间顺序，是一种详细表示对象之间以及对象与参与者之间行为关系的图，由一组协作的对象(或参与者)以及它们之间可发送的消息组成，强调消息之间的顺序。

2. 顺序图的组成

顺序图由四部分组成:参与者、生命线、激活期与消息。下面分别介绍。

(1)参与者(actor)或者 对象(object)具有以下几个特点：

①参与者和对象按照从左到右的顺序排列；

②一般最多两个参与者，他们分列两端。启动这个用例的参与者往往排在最左边，接收消息的参与者则排在最右端；

③对象从左到右按照重要性排列或按照消息先后顺序排列；

④将对象置于顺序图的顶部意味着在交互开始的时候对象就已经存在了，如果对象的位置不在顶部，那么表示对象是在交互的过程中被创建的。

对象的命名方式有三种:包括对象名和类名;类名(匿名对象);对象名(不关心类)。顺

序表示如图 8-30 所示。

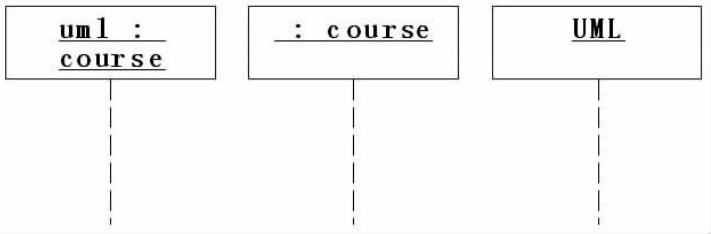


图 8-30 对象的命名

(2)生命线(lifeline)具有以下几个特点：

①每个对象都有自己的生命线，用来表示在该用例中一个对象在一段时间内的存在

②垂直的虚线

③如果对象生命期结束，则用注销符号表示。

生命线表示如图 8-31 所示。

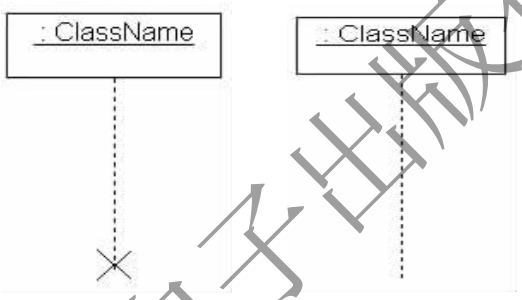


图 8-31 生命线

(3)激活期(activation)/ 控制焦点(focus of control)

①对象在一段时间内获得了焦点，也称激活期

②对象执行某个动作的时期

③空心矩形条

④激活期的长短意味着对象执行某个动作的时间有多长，可以通过约束{10ms}来限制执行时间的长短。

激活期如图 8-32 所示。

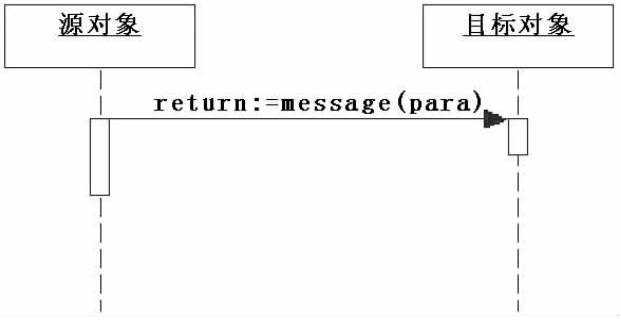


图 8-32 激活期

(4)消息(message)

面向对象方法中，消息是对象间交互信息的主要方式。结构化程序设计中，模块间传递

信息的方式主要是过程(或函数)调用。对象 A 向对象 B 发送消息,可以简单地理解为对象 A 调用对象 B 的一个操作(operation)。

消息具有以下特点:

①顺序图中,尽力保持消息的顺序是从左到右排列的。

②一个顺序图的消息流开始于左上方,消息 2 的位置比消息 1 低,这意味着消息 2 的顺序比消息 1 要迟。

③顺序图中消息编号可显示,也可不显示。

UML1.4 后定义的消息有:调用消息(procedure call)、异步消息(asynchronous)、返回消息 (return)。

UML1.3 前的消息还有:简单消息 (simple)。

Rose 扩充的消息:阻止消息 (balking)、超时消息 (time-out)。

主要的消息介绍如下:

①调用消息:消息的发送者把控制传递给消息的接收者,等待接收者返回或放弃控制,可以表示同步。调用消息必有一个与之配对的返回消息,但是可以不用画出。实心箭头符号。

如类 Driver 向类 Car 传递了 open()消息,如图 8-33 所示。

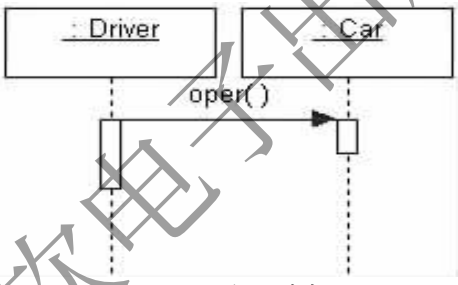


图 8-33　调用消息

②异步消息:消息的发送者把控制传递给消息的接收者,然后继续自己的工作,不等待接收者返回或放弃控制。如图 8-34 所示。

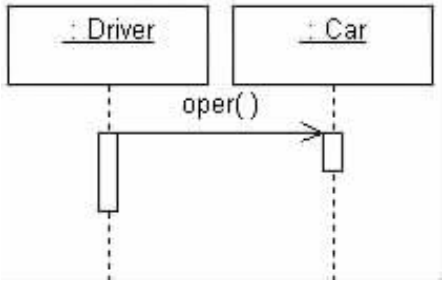


图 8-34　异步消息

③返回消息(Return):表示消息的返回。消息上方放置返回值,也可以不画出,直接隐含。虚线箭头表示。如图 8-35 所示,虚线表示返回消息。

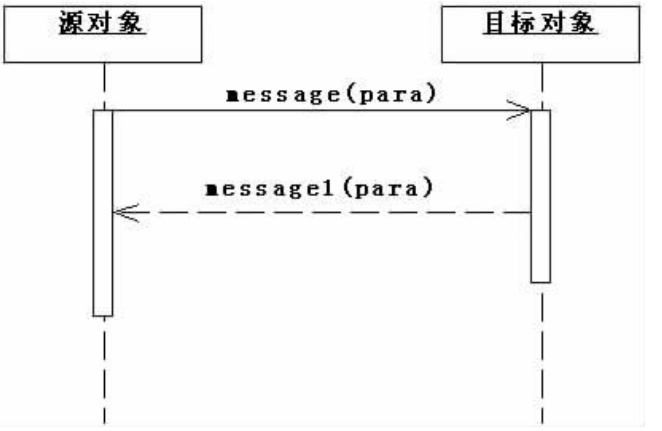


图 8-35 返回消息

④阻止消息：消息的发送者传递消息给接收者，如果接收者无法立即接收，则发送者放弃该消息.

⑤超时消息：消息的发送者发出消息给接收者并按指定时间等待，若接收者无法在指定时间内接收，则发送者放弃该消息.

⑥自调用(Self Call)

某对象自己调用自己的操作，用嵌套的矩形条表示。如图 8-36 所示，对象 Source 调用自己的操作。

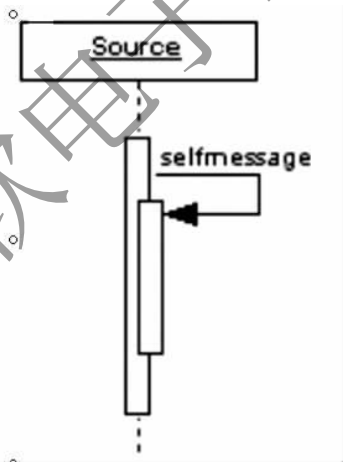


图 8-36 自调用消息

3. 建立顺序图的步骤

①确定交互过程的上下文。

②识别参与交互过程的对象。

③为每个对象设置生命线。

④从引发这个交互过程的初始消息开始，在生命线之间自顶向下依次画出随后的各个消息。

⑤如果需要嵌套或表示消息发生的时间点，使用控制焦点。

⑥如果需要说明时间约束，则在消息旁边加上约束说明。

⑦如果需要，可以为每个消息设置前置条件和后置条件。

下面的实例介绍如何建立顺序图：

需求描述：李小平是一个爱书之人，家里各类书籍已过千册，而平时又时常有朋友外借，因此需要一个个人图书管理系统。该系统应该能够将书籍的基本信息按计算机类、非计算机类分别建档，实现按书名、作者、类别、出版社等关键字的组合查询功能。在使用该系统录入新书籍时系统会自动按规则生成书号，可以修改信息，但一经创建就不允许删除。该系统还应该能够对书籍的外借情况进行记录，可对外借情况列表打印。另外，还希望能够对书籍的购买金额、册数按特定时间周期进行统计。

第一步：寻找分析类

在分析阶段，我们常在用例中寻找三种分析类(边界类、控制类和实体类)。我们通过分析用例描述中的事件流，寻找分析类。

在分析时，还必须综合考虑基本事件流和扩展事件流两个方面。下面以“新增书籍“为例进行分析。其事件流如下所示：

1. 基本事件流

(1)图书管理员向系统发出“新增书籍信息”请求。

(2)系统要求图书管理员选择要新增的书籍是计算机类还是非计算机类。

(3)图书管理员做出选择后，显示相应界面，让图书管理员输入信息，并自动根据书号规则生成书号。

(4)图书管理员输入书籍的相关信息，包括：书名、作者、出版社、ISBN 号、开本、页数、定价、是否有 CD-ROM。

(5)系统确定输入的信息中书名没有重复。

(6)系统将所输入的信息存储建档。

2. 扩展事件流

(1)如果输入的书名有重名现象，则显示出重名的书籍，并要求图书管理员选择修改书名或取消输入。

(2)图书管理员选择取消输入，则结束用例，不做存储建档工作。

(3)图书管理员选择修改书名后，转到 5。

第二步：寻找边界对象

对这个用例而言，参与者“图书管理员”，因此要寻找边界对象只需以“图书管理员”这个参与者为线索，从用例描述中去寻找分析类：

(1)图书管理员向系统发出“新增书籍信息”请求——图书管理员在什么地方向系统发出“新增书籍信息”的请求呢？通常会设计一个主窗口，并在上面摆放一些按钮来实现，因此在此句描述中间，可以发现两个边界类：主窗口、“新增书籍信息”按钮。

(2)系统要求图书管理员选择要新增的书籍是计算机类还是非计算机类——从此句中可以发现一个新的边界类：书籍类别列表框。

(3)图书管理员做出选择后，显示相应界面，让图书管理员输入信息，并自动根据书号规则生成书号——此句中可以发现最为关键的一个边界类——“新书信息录入”窗口以及辅助

的“提交”按钮。

而且，还可以发现新增书籍信息按钮是组成主窗口的一部分。书籍类别列表框、提交按钮则是组成新书信息录入窗口的一部分。根据以上信息，可以先将参与者和边界对象绘制出来，其结果如图 8-37 所示。

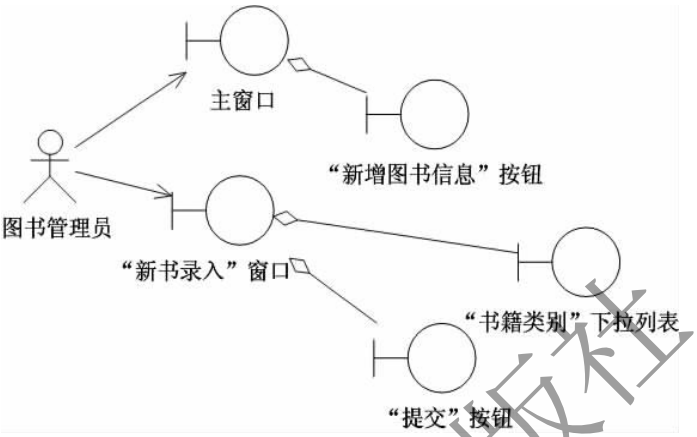


图 8-37　边界类对象分析

第三步：寻找控制对象和实体对象

实体对象通常来源于领域中的类图，也就是描述业务领域的名词或名词短语，通过阅读整个事件流的详细描述，我们得知，实体对象有书籍、计算机书籍、非计算机书籍以及书籍列表 4 个。

在这个例子中，根据扩展事件流的描述，可以在图 8-37 上增加相应的控制对象，得到更进一步的分析图，如下图 8-38 所示。

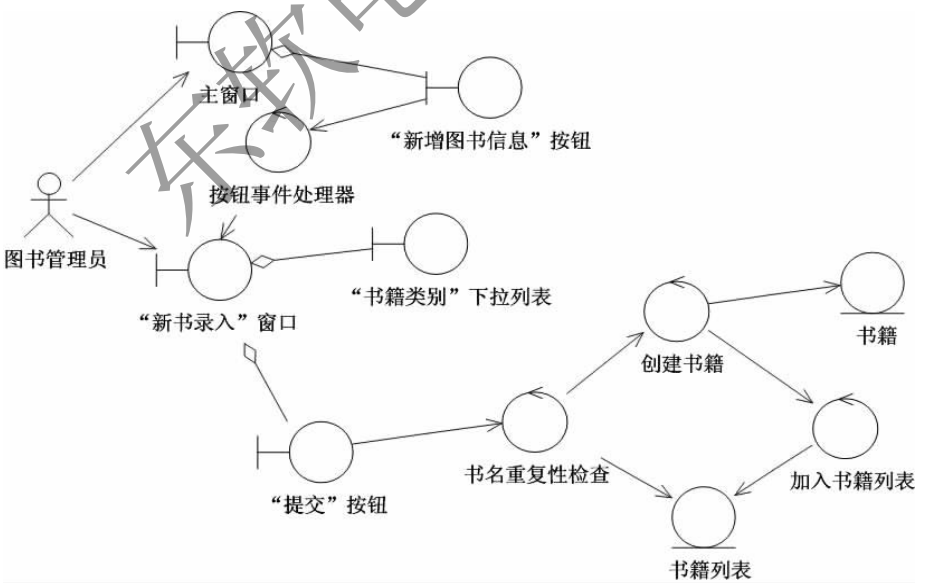


图 8-38　控制对象分析

另外，还有几个关键的事件没有体现到图中：一是基本事件流中的步骤 2、3 要求根据用户选择的类别自动获得书号；二是当书名重复性检查没有通过时（有重名），应返回要求其

重输。

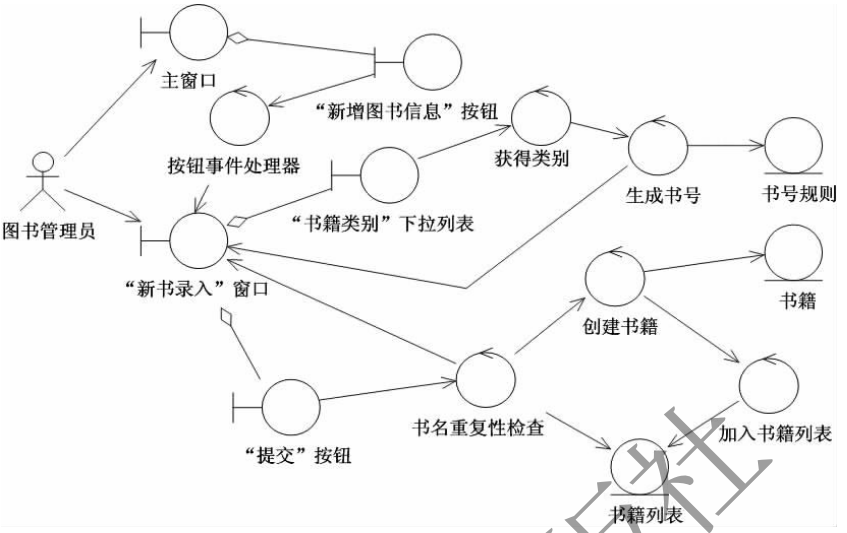


图 8-39　最终分析图

下面是构建顺序图的步骤：

首先，按照从左到右的顺序，依此将参与者、边界对象和实体对象放在最顶部，边界类放在左边，实体类放在右边。注意，在分析模型中可以先不考虑控制类的引用（当然有必要时也可以引入）。最终的完善顺序图如图 8-40 所示。

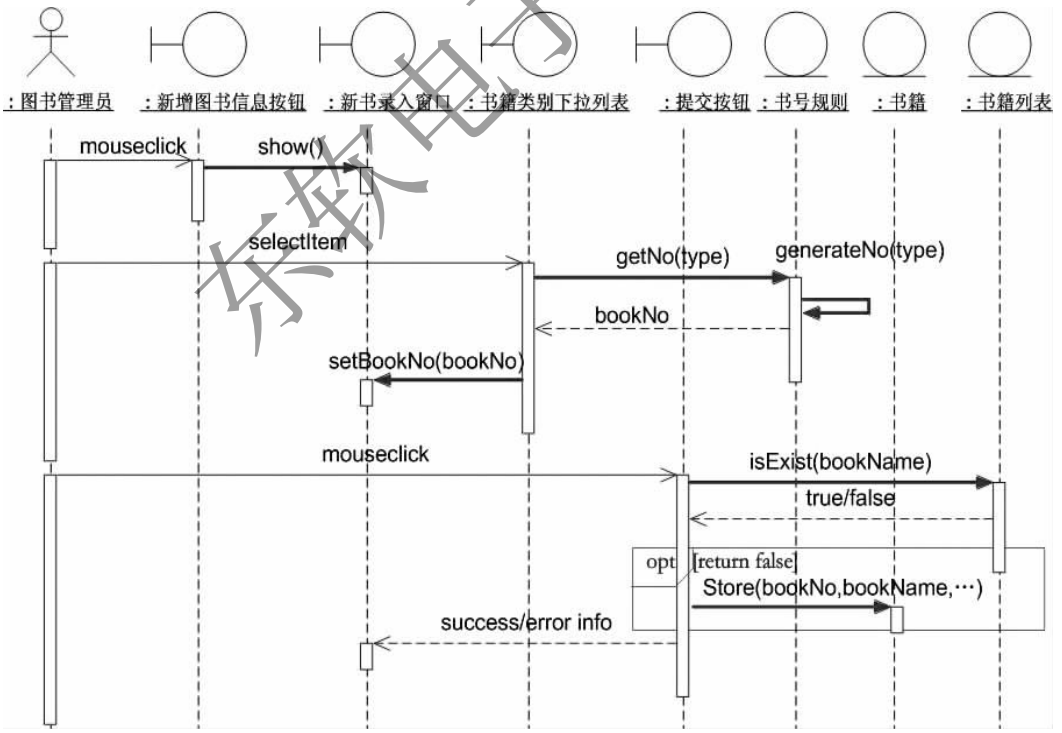


图 8-40　顺序图

浏览顺序图的方法是，从上到下（按时间顺序）查看对象间交换的信息。图 8-41 是顺序

图的一个例子。注意，从 PrinterServer 到 Printer 的消息带有条件，表示当打印机空闲时发送 Print 消息到 Printer，否则无法将 store 消息给 Queue。

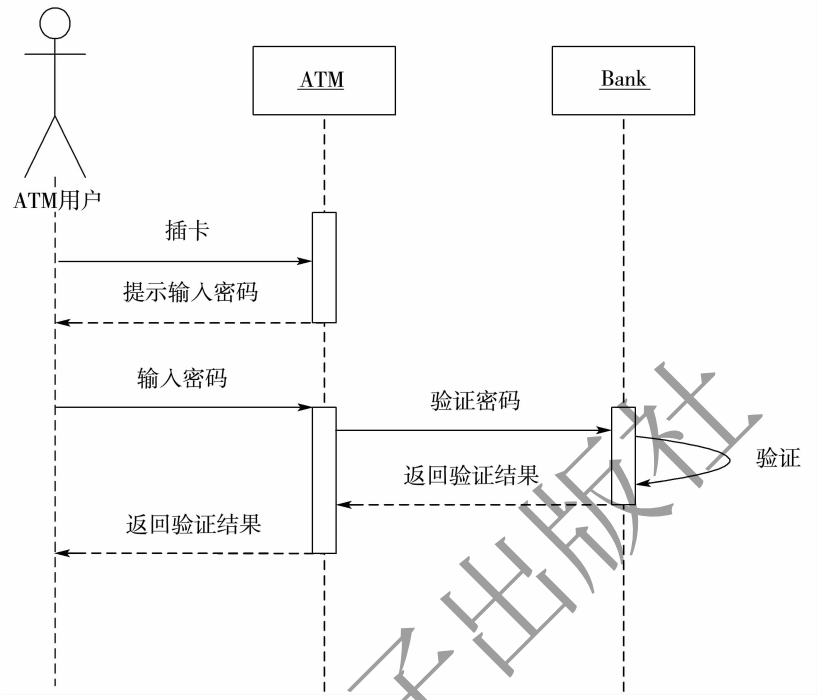


图 8-41 顺序图

一个对象可以通过发送消息来创建或删除另一个对象，当一个对象被删除（或自我删除）时，用一个大“X”来标记。

8.2.6 协作图

1. 协作图概述

协作图用于描述互相合作的对象间的交互关系和链接关系。虽然顺序图和协作图都用来描述对象间的交互关系，但是侧重点不一样。顺序图着重体现交互的时间顺序，协作图则着重体现交互对象间的动态连接关系。

2. 协作图组成

协作图包含了 3 个元素：对象（Object）、链（Link）、消息（Message）。

（1）对象：对象和对象图中的对象一致。

（2）链：协作图中链的符号和对象图中链所用的符号是一样的，即一条连接两个类角色的实线表示。链接用来在协作图中关联对象，链接的目的是让消息在不同系统对象之间传递。没有链接，两个系统对象之间无法彼此交互。

（3）消息：是协作图中对象与对象之间通信的方式。消息在协作图中显示为一个伴随链接或者关联角色的文本字符串，并带有一个箭头来指示消息沿着关系传递的方向。如图 8-42所示。

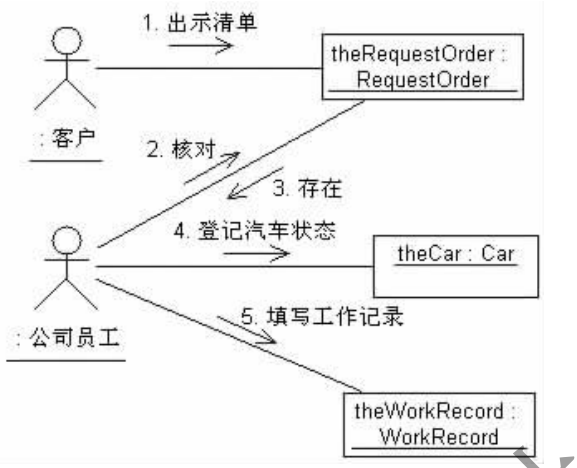


图 8-42　协作图

时序图与协作图都表示对象之间的交互作用，只是它们的侧重点有所不同：时序图描述了交互过程中的时间顺序，但没有明确地表达对象之间的关系。协作图描述了对象之间的关系，但时间顺序必须从顺序号获得。

3. 建立协作图

建立协作图的步骤如下：

(1)确定交互过程的上下文

(2)识别参与交互过程的对象

(3)如果需要，为每个对象设置初始特性

(4)确定对象之间的链，及沿着链的消息

需求分析描述如下：

张老师希望通过系统查询某名学生的学科成绩。

(1)张老师通过用户界面录入学生的学号以及学科科目请求学生信息。

(2)用户界面根据学生的学号向数据库访问层请求学生信息。

(3)数据库访问层根据学生的学号加载学生信息。

(4)数据库访问层根据学生信息和学科科目获取该名学生的分数信息。

(5)数据库访问层将学生信息和分数信息提供给用户界面。

(6)用户界面将学生信息和分数信息显示出来。

第一步：确定协作图的元素如下：教师类(Teacher)、用户界面类(WebInterface)、学生信息类(StudentInfo)、数据库访问类(DataManager)、学生成绩类(grades)。如图 8-43 所示。

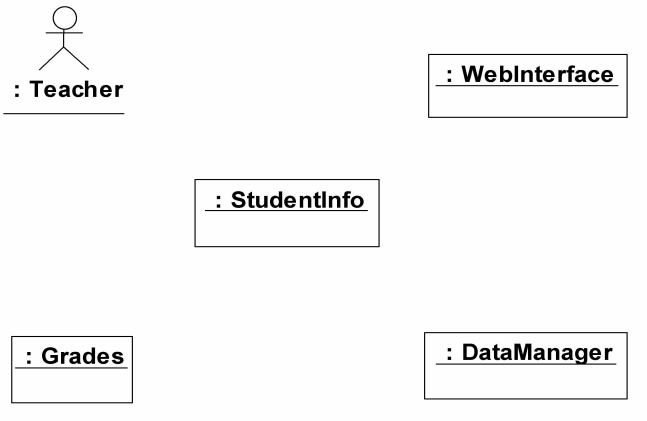


图 8-43 确定协作图元素

第二步:确定元素之间的结构关系

确定这些对象之间的连接关系,使用链与角色将这些对象连接起来。

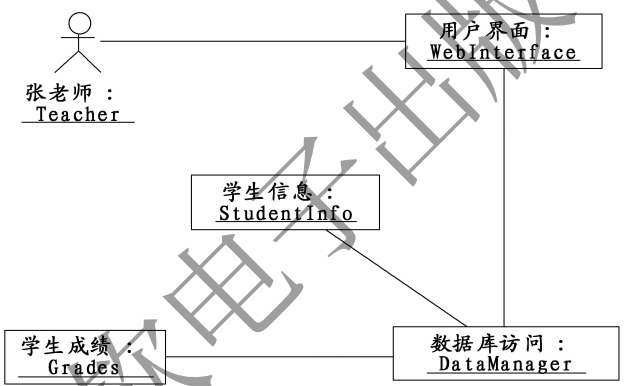


图 8-44 对象之间的关系

第三步:细化协作图

创建协作图的最后一步就是将早期的协作图进行细化。添加的链之间的消息。如图 8-45 所示。

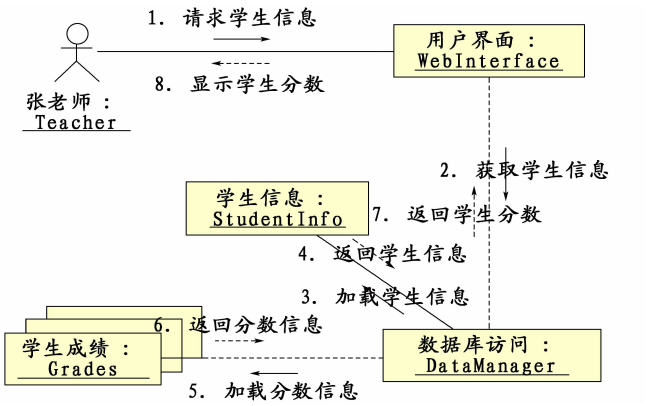


图 8-45 链之间的消息

例：储户在 ATM 机上存钱操作的协作图。

1. 确定系统中的类，如图 8-46 所示。

图 8-46　确定类

2. 确定类之间的关系，如图 8-47 所示。

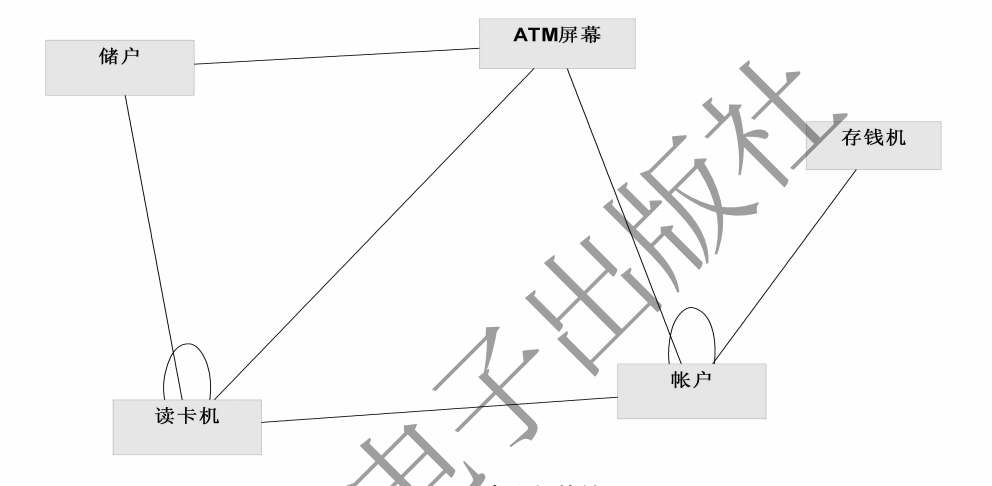


图 8-47　类之间的链

3. 对象实例之间协作关系，如图 8-48 所示。

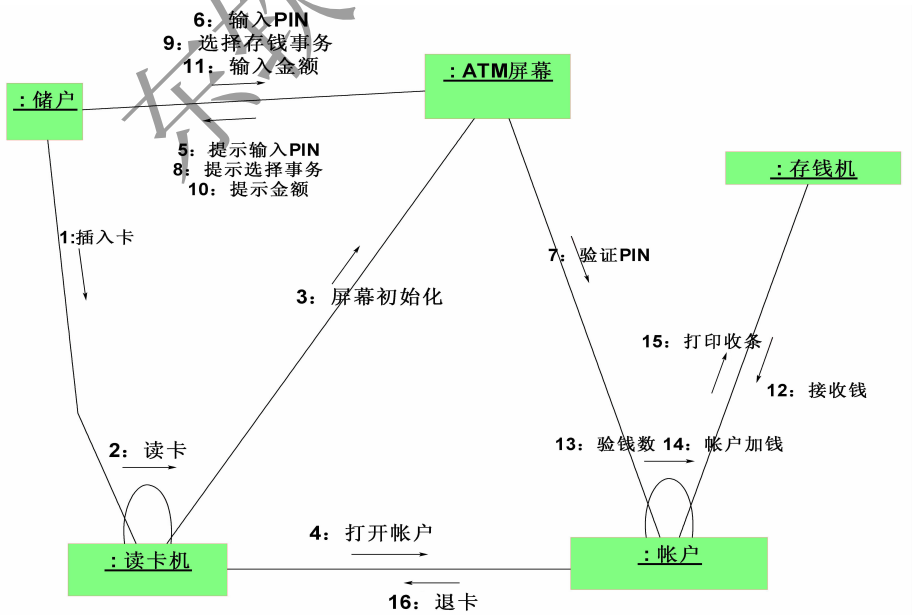


图 8-48　链之间的消息

8.2.7 活动图

1. 活动图概述

活动图(activity diagram)和交互图是 UML 中对系统动态建模的两种主要形式。

交互图(顺序图和协作图)强调的是对象到对象的控制流,而活动图则强调的是从活动到活动的控制流。

活动图用来描述事物或对象的活动变化流程,是一种表述业务过程、工作流的技术。它可以用来对业务过程、工作流建模,也可以对用例实现甚至是程序实现来建模。

活动图描述了从活动到活动的流。从本质上说,是一个流程图,它显示出一个过程的各个步骤。是 UML 中对系统动态方面建模的图之一。

2. 活动图的组成

如图 8-49 所示:活动图由活动、活动流、分支与合并、分叉与汇合、泳道和对象流六要素组成。

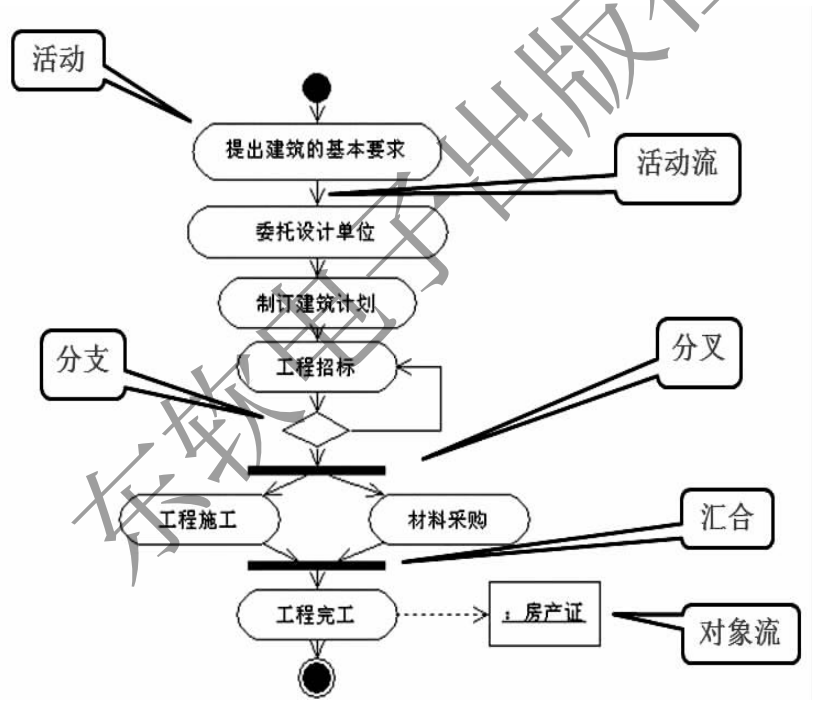


图 8-49 活动图

(1)活动

活动(Action):是活动图主要结点,用两边为弧的条形框表示,中间为活动名。活动分为简单活动和复合活动。简单活动为不能再分解的活动,如图 8-50 所示;复合活动是可以再分解的复杂活动,如图 8-51 所示。

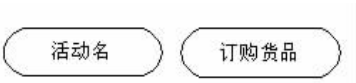


图 8-50　简单活动

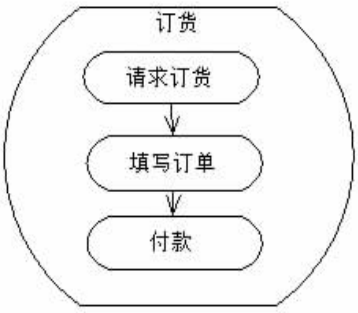


图 8-51　复杂活动

(2)活动流

活动流(Action Flow)：描述活动之间的有向关系，反映一个活动向另外一个活动之间的转移。用带箭头的实线表示，如图 8-52 所示。

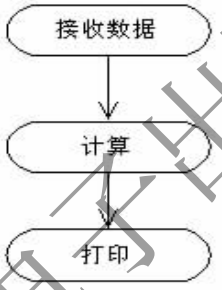


图 8-52　活动流

与状态图不同，活动图的转换一般不需要特定事件的触发，自动转换。

(3)分支与合并

分支：活动流要根据不同的条件决定转换的去向。分支包括一个入转换和多个出转换，出转换之间是互斥的；合并包括多个入转换和一个出转换，如图 8-53 所示。

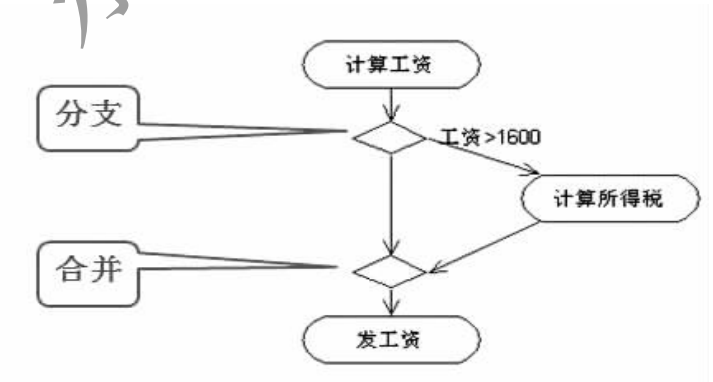


图 8-53　分支与合并

(4)分叉与汇合

分叉与汇合：用来对并发的控制流建模。分叉用于将活动流分为两个或多个并发运行

的分支，如图 8-54 所示。

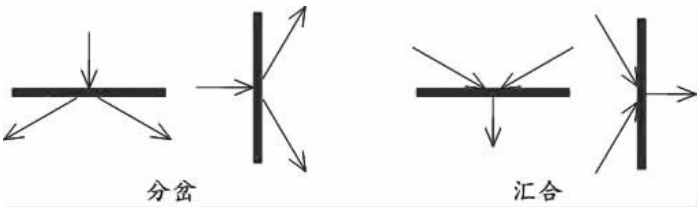


图 8-54　分叉与汇合

如图 8-55 所示，处理订货活动分叉成结账活动和运货活动，之后又汇合成通知客户活动。

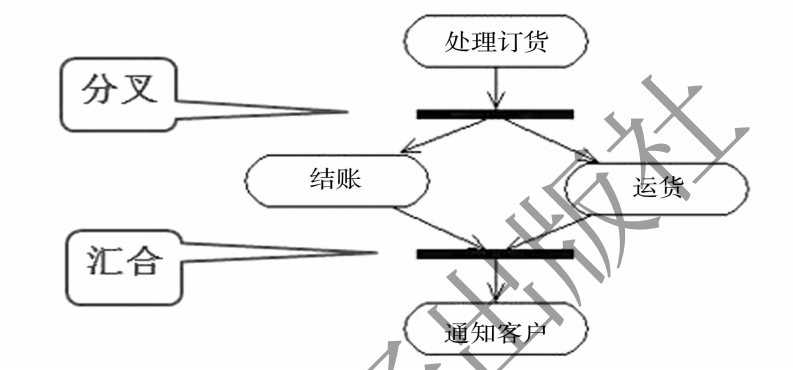


图 8-55　分叉与汇合实例

(5)泳道

泳道(swim lane)：是活动图中的区域划分，每一个泳道代表一个责任区域，指明活动是由谁负责的或发起的。一个泳道中包括一组相关活动。

如图 8-56 所示，有三个泳道，分别由销售、客户服务和财务三个角色发起。

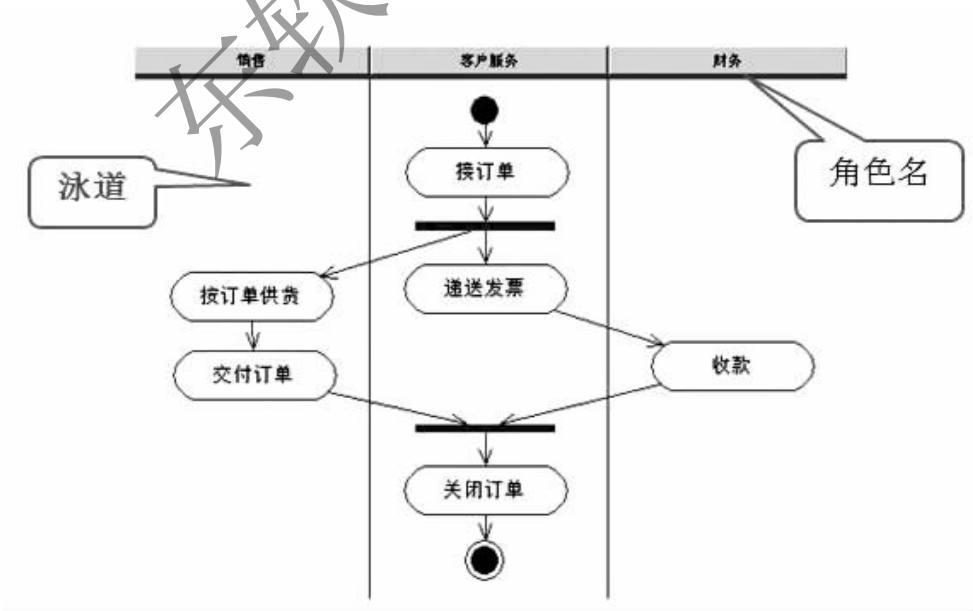


图 8-56　泳道

(6) 对象流

对象流：反映活动与对象之间的依赖关系，表示对象对活动的作用或活动对对象的影响，用依赖关系表示，如图 8-57 所示。

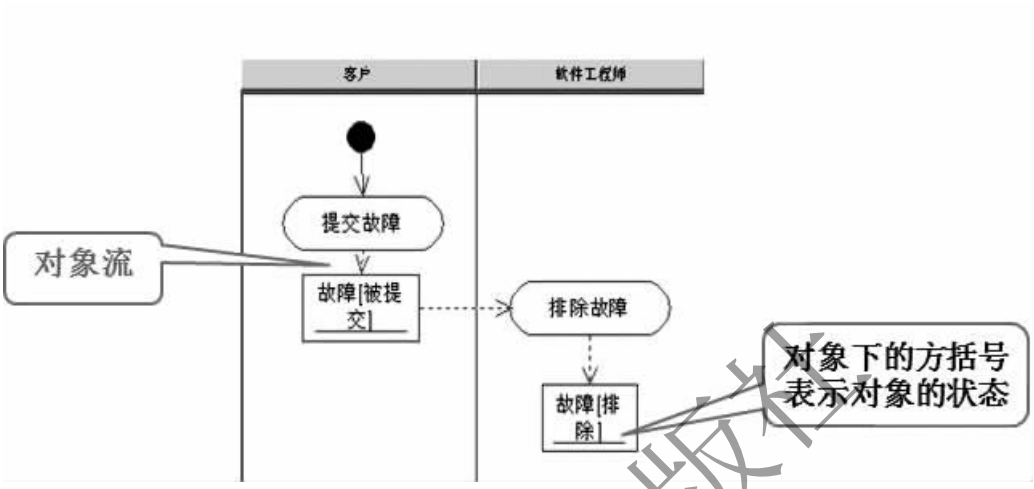


图 8-57　对象流

①如果箭头从活动指向对象，表示活动对对象的创建、修改或撤销等的影响；

②如果箭头从对象指向活动，表示该活动将使用所指向的对象。

3. 建立活动图

下面这些步骤描述了绘制活动图的基本任务，这些任务都以迭代的方式执行。

(1)确定活动图的范围。

(2)增加起点和终点。

(3)添加活动。

(4)添加活动间的转变。

(5)添加决策点。

(6)找出可并行活动之处。

例 1：学生请假流程描述如下：

(1)学生请假须先经班主任同意；

(2)班主任在准假时，如学生请假时间超越审批权限，还要请系办审批，经系办审批后，系办将假条存根留下，事后转班主任存查。

(3)学生请假获准后，应立即报告班长，以便班长向任课教师报告。

过程简化，最终活动图如图 8-58 所示。

例 2：活动图具有广泛地应用，活动图对表示并发行为很有用。在软件建模中，活动图可以用来描述业务流程，如图 8-59 描述了订货处理业务流程。

(2) 描述工程组织过程

各种工程组织过程，管理过程均可以用活动来描述。如图 8-60 所示，描述了建筑工程流程。

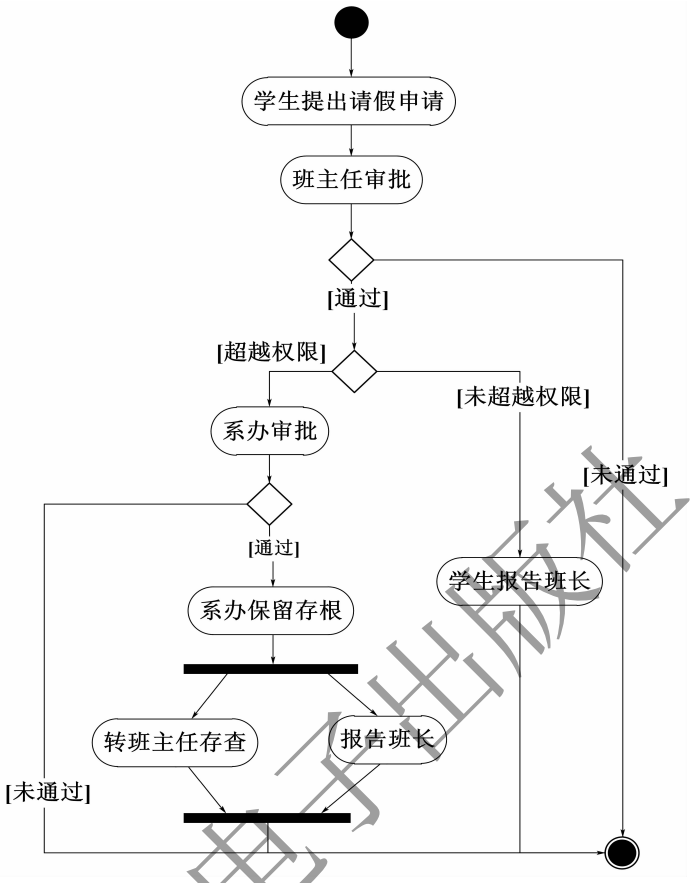


图 8-58 学生请假活动图

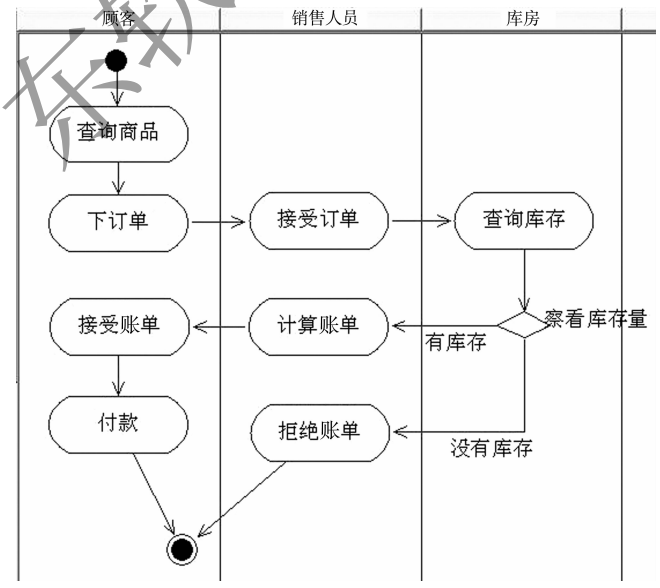


图 8-59 订货处理活动图

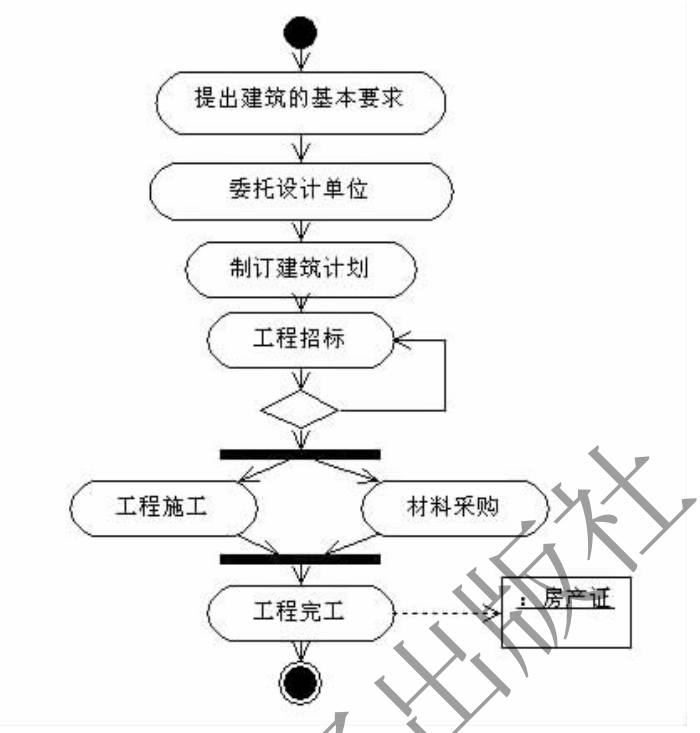


图 8-60　工程活动图

(3) 描述算法流程。如图 8-61 所示描述了算法流程。

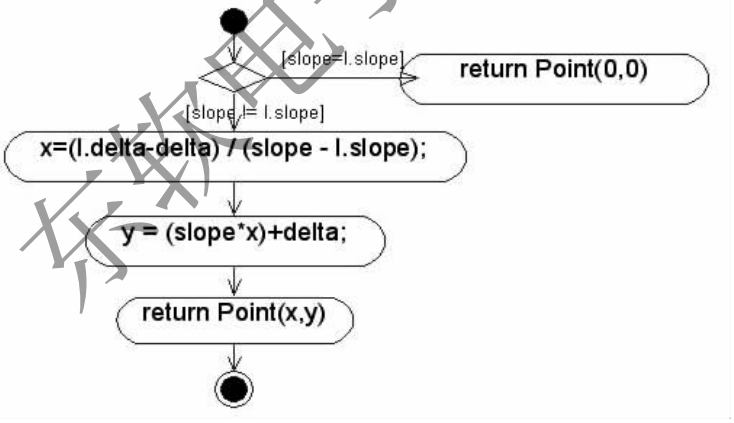


图 8-61　过程活动图

(4)描述软件开发过程。如图 8-62 和 8-63 所示。

4. 活动图和状态图的比较

活动图描述动作及动作之间的关系。活动图示状态机图的一个变种,它的目的与状态机图有些不同,活动图的主要目的是描述动作及动作的结果——对象状态改变。不需指明任何事件,只要动作被执行,活动图中的状态就自动开始转换。如果状态转换的触发事件是内部动作的完成,可以用活动图描述:当状态转换的触发事件是外部事件时,常用状态机图来表示。

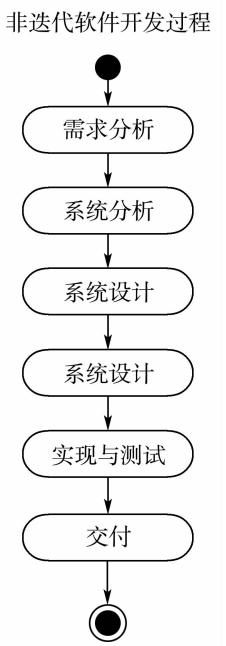


图 8-62 非迭代软件开发过程

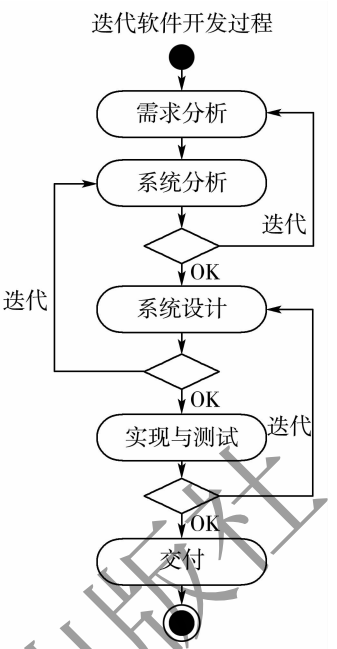


图 8-63 迭代软件开发过程

在活动图中,用例和对象的行为中的各个活动之间通常具有事件顺序。活动图可表示这种顺序,展示出对象执行某种行为时或者在业务过程中所要经历的各个活动和判定点。每个活动用一个圆角矩形表示,判定点用菱形框表示。

描述对象不同:状态图描述对象状态及状态之间的转移,以状态为中心;活动图描述从活动到活动的控制流,以活动为中心。

使用场合不同:状态图描述对象在其生命期中的行为状态变化;活动图描述过程的流程变化。

8.2.8 状态机图

状态机图用来描述一个特定对象的所有可能状态,以及引起状态转换的事件。大多数面向对象技术都用状态机图表示单个对象在其生命期中的行为。一个状态机图包括一系列状态、事件以及状态之间的转移。

1. 状态

所有对象都具有状态,状态是对象执行了一系列活动后的结果。当某个事件发生后,对象的状态将发生变化。在状态机图中定义的状态可能有:初始状态、最终状态、中间状态和复合状态。一个状态机图只能有一个初始状态,而最终状态则可以有多个。

初始状态由一个内部实心的黑圆● 来表示,最终状态由一个内部实心的同心圆● 来表示,

一个中间状态由一个圆角矩形来表示,如图 8-64 所示,其中的状态变量部分和活动表部分可以省略。

如果一个状态可以进一步地细化为多个子状态，我们就称这个状态为复合状态。子状态之间可以有“或”和“与”两种关系。

图 8-64　中间状态的三个组成部分

2. 状态转换

状态机图中两个状态之间带箭头的连线称为状态转换。状态的变迁通常是由事件触发的，此时应在转移上标出触发转移的事件表达式。如果转移上未标明事件，则表示在源状态的内部活动执行完毕后自动触发转移。

一个状态使对象满足一定的条件，状态可以认为是一个类属性值的抽象。

图 8-65 描述了顾客在 ATM 机上进行操作时经历的几种状态以及各种状态之间转换的条件。

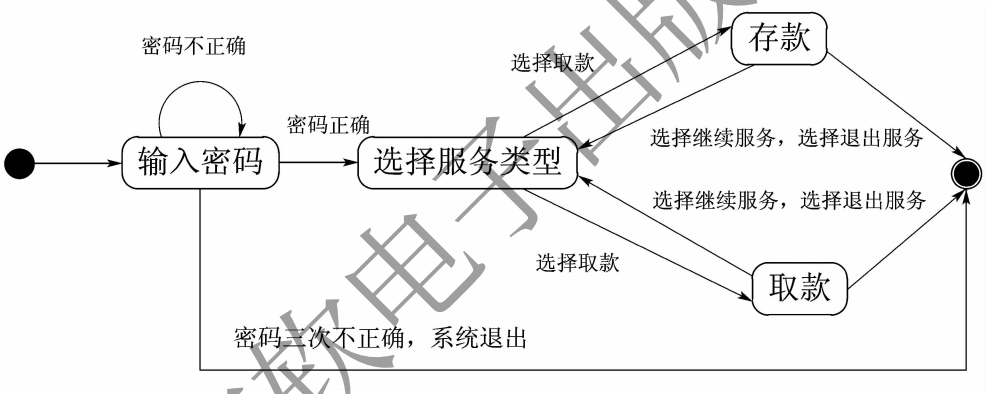


图 8-65　状态机图

8.2.9　包图

包是一种组合机制。把各种各样的模型元素通过内在的语义关系连在一起，形成一个高内聚、低耦合的整体就叫做包。包通常用于对模型的组织管理，因此有时又把包称为子系统。

(1)包的内容

构成包的模型元素称为包的内容，包的内容可以是一个类图也可以是另一个包图。包与包之间不能共用一个相同的模型元素。例如，在图 8-66 中，“系统内部”包由“保险单”包和“客户”包组成，我们把“保险单”包和“客户”包称为“系统内部”包的内容。包的图示符号类似于图示卡片的形状，由两个长方形组成，小长方形位于大长方形的左上角。

当不需要显示包的内容时，包的名字放入主方框内(例如，“保险单”、“Oracle 界面”等)；否则，包的名字放入左上角的小方框内(例如，“系统内部”)，而把包的内容放入主方框内。

(2)包的依赖和继承

包与包之间允许建立依赖、泛化和细化等关系。例如,在图 8-66 中“保险单填写界面”包依赖于“保险单”包,整个“系统内部”包依赖于“数据库界面”包。可以使用继承关系中“一般”和“特殊”的概念来说明通用包和专用包之间的关系。例如,“Oracle 界面”包和“Sybase 界面”包继承“数据库界面”包。通过“数据库界面”包,“系统内部”包既可以使用 Oracle 的界面也可以使用 Sybase 的界面。这是因为和类的继承关系相似,专用包必须与通用包的界面一致。通用包可标记为{abstract},表示该包只是定义了一个界面,具体实现由专用包完成。

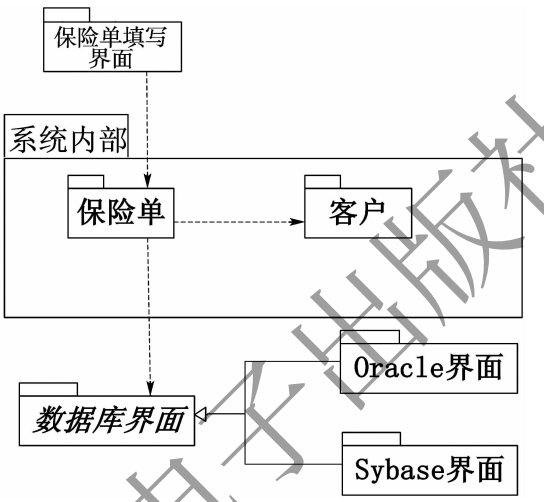


图 8-66 包图

8.2.10 构件图和部署图

构件图和部署图显示系统实现时的一些特性,其中构件图显示代码本身的结构(即静态机构),部署图显示系统运行时刻的实现结构。

1. 构件图

构件图代表的是实现环境中的软件模块。类图和包图对软件的逻辑设计建模,而构件图模拟的是实现视图,是实际的软件模块。类图和包图对软件的逻辑设计建模,而构件图模拟是实现视图,是实际的软件模块。构件图显示软件构件之间的依赖关系。一般来说,软件构建就是一个实际文件,它可以是源代码文件、二进制代码文件或可执行文件等,用来显示编译、链接或执行时构件之间的依赖关系。

在 UML 中,构件的图示符号是左边带有两个小矩形的大长方形。构件间的依赖关系用一条带箭头的虚线表示。可以为一个构件定义其他构件可见的接口,其图示符号从代表构件的大矩形边框画出的一条线,线的另一端为一个小空心圆,接口名写在空心圆附近。图 8-67 是构件图的例子。

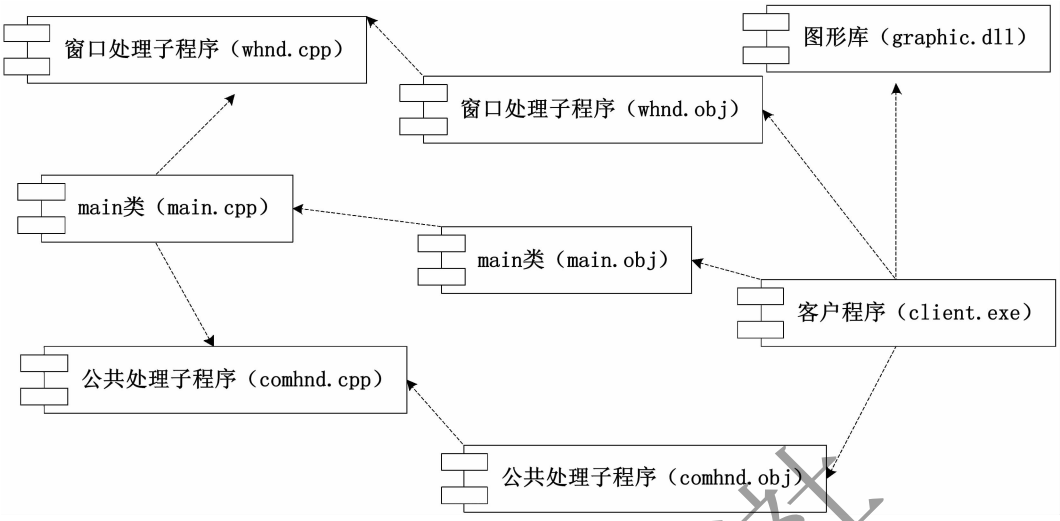


图 8-67　构件图

2. 部署图

部署图也称配置图。部署图描述的是处理器、硬件设备和软件构建在运行时的架构，它显示系统硬件的物理拓扑结构，以及在此结构上实行的软件，也可以显示硬件节点的拓扑结构和通信路径、节点上运行的软件构件、软件构件包含的逻辑单元（对象、类）等。部署图常用于帮助理解分布式系统。

（1）节点和连接

节点代表一个物理设备及在其上运行的软件系统，例如，一台 Unix 主机、一个 PC 终端、一台打印机、一台通信设备等。在图 8-68 中，“客户端 PC”和“保险后台服务器”就是两个节点。在 UML 中，节点用一个立方体来表示，节点名放在立方体的左上角。

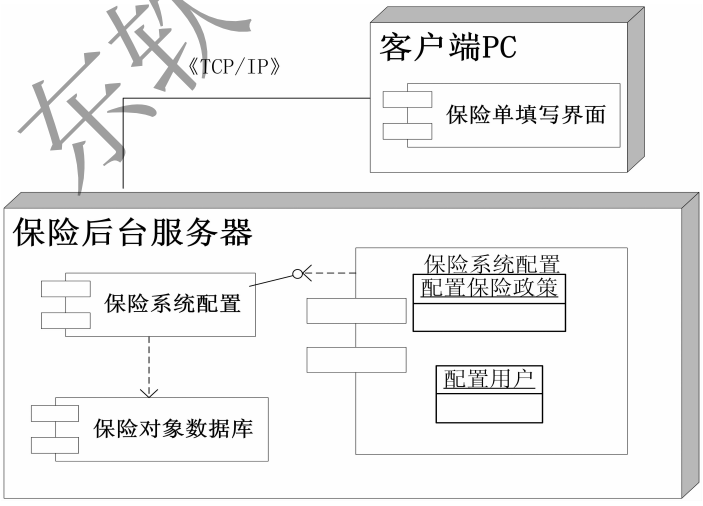


图 8-68　部署图

节点间的连线表示系统之间进行交互的通信线路，在 UML 中称为连接。通信类型则放在连接旁边的“《》”之间，以制定所用的通信协议或网络类型。

(2)构件和接口

在部署图中,构件代表可执行构件的实例,在逻辑上它可以与类图中的包或类对应。因此,部署图显示运行时各个包或类在节点中分布的情况。在图 8-68 种,“保险后台服务器”节点中包含“保险系统”、“保险对象数据库”和“保险系统配置”三个构件。

在面向对象方法中,并不是类和构件等元素的所有属性和操作都对外可见,它们对外提供的可见操作和属性称为接口。借口用一端是小圆圈的直线表示。在图 8-68 中,“保险系统”构件提供了一个名字叫“配置”的接口,图中还显示了构件之前的依赖关系:“保险系统配置”构件通过这个接口依赖“保险系统”构件。

(3)对象

一个面向对象的软件系统中可以运行很多对象,而构件可以看作与包或类对应的物理代码模块,因此,构件中应包含一些运行的对象。图 8-68 中,“保险系统配置”构件包含“配置保险政策”和“配置用户”两个对象。部署图中的对象与对象图中的对象表示方法相同。

8.3 小结

面对对象就是使用对象,类和继承机制来设计和构造相应的软件系统,并且对象之间仅能通过传递消息实现彼此通信。

UML 是一种标准的图形化建模语言,它用若干个视图构造系统的模型,每个视图描述系统的一个方面。

用例模型是描述系统基本功能的工具,它由一个或多个用例图描述。类图描述系统的静态结构,类图由类及它们之间的关系构成,是构建其他 UML 图的基础。所有系统都既有静态结构又有动态行为,动态行描述静态结构内包含的元素是如何交互的。

顺序图描述对象之间的动态交互关系,着重表现对象间消息传递的时间顺序,是一种详细表示对象之间以及对象与参与者之间行为关系的图。

协作图用于描述互相合作的对象间的交互关系和链接关系。虽然顺序图和协作图都用来描述对象间的交互关系,但是侧重点不一样。顺序图着重体系交互的时间顺序,协作图则着重体现交互对象间的动态连接关系。

交互图(顺序图和协作图)强调的是对象到对象的控制流,而活动图则强调的是从活动到活动的控制流。从本质上说,是一个流程图,它显示出一个过程的各个步骤。是 UML 中对系统动态方面建模的图之一。

状态机图用来描述一个特定对象的所有可能状态,以及引起状态转换的事件。大多数面向对象技术都用状态机图表示单个对象在其生命期中的行为。

包是一种组合机制。把各种各样的模型元素通过内在的语义关系连在一起,形成一个高内聚、低耦合的整体就叫做包。包通常用于对模型的组织管理,因此有时又把包称为子系统。

构件图和部署图显示系统实现时的一些特性,其中构件图显示代码本身的结构(即静态机构),部署图显示系统运行时刻的实现结构。

8.4　习题

一、选择题

1. 面向对象程序设计的基本机制是(　　)。

A. 继承　　B. 消息　　C. 方法　　D. 结构

2. 下列属于面向对象的要素有(　　)。

A. 分类性　　B. 抽象　　C. 共享　　D. 封装

3. 下列选项中属于面向对象开发方法的有(　　)。

A. Booch　　B. CAD　　C. Coad　　D. OMT

4. 下列属于 Coad/Yourdon 方法中面向对象的分析模型的层次有(　　)。

A. 主题层　　B. 对象层　　C. 应用层　　D. 接口层

5. 一个类属性依其特征划分,其类型有(　　)。

A. 描述型　　B. 定义型　　C. 派生型　　D. 参考型

6. 在进行面向对象分析时,所采用的模型有(　　)。

A. 对象模型　　B. 动态模型　　C. 静态模型　　D. 功能模型

7. 状态是对象属性的值的一种抽象,它的性质有(　　)。

A. 时间性　　B. 持续性　　C. 有序性　　D. 有穷性

8. 数据流图中的处理必须用对象中的操作来实现,常见的操作有(　　)。

A. 查询　　B. 动作　　C. 活动　　D. 访问

9. 建立继承关系时所采用的方式有(　　)。

A. 自顶向下　　B. 从内到外　　C. 自底向上　　D. 从复杂到简单

10. 对象是人们要研究的任何事物,主要的对象类型有(　　)。

A. 有形实体　　B. 作用　　C. 事件　　D. 性能说明

二、判断题

1. 面向对象的的方法是以类作为最基本的元素,它是分析问题和解决问题的核心。(　　)

2. 类是指具有相同或相似性质对象的抽象,对象是抽象的类,类的具体化就是对象。(　　)

3. 继承性是父类和子类之间共享数据结构和消息的机制,这是类之间的一种关系。(　　)

4. 多态性增强了软件的灵活性和重用性,允许用更为明确、易懂的方式去建立通用软件,多态性和继承性相结合使软件具有更广泛的重用性和可扩充性。(　　)

5. 面向对象分析,就是抽取和整理用户需求并建立问题域精确模型的过程。(　　)

6. 面向对象设计的主要目标是提高生产效率,提高质量和提高可维护性。(　　)

7. 对象模型表示了静态的、结构化的系统数据性质,描述了系统的静态结构,它是从客

观世界实体的对象关系角度来描述，表现了对象的相互关系。 （ ）

8. 面向对象的分析是用面向对象的方法对目标系统的问题域空间进行理解、分析和反映。通过对象层次结构的组织确定解空间中应存在的对象和对象层次结构。 （ ）

9. 类的设计过程包括：确定类，确定关联类，确定属性，识别继承关系。 （ ）

10. 复用也叫重用或再用，面向对象技术中的“类”是比较理想的可重用软构件，它有三种重用方式：实例重用、继承重用、多态重用。 （ ）

11. 主题是一种关于模型的抽象机制，它是面向对象模型的概貌，也是关于某个模型要同时考虑和理解的内容，主题起一种控制作用。 （ ）

12. 面向对象的分析由对象、结构、继承性和基于消息的通信构成。 （ ）

13. 支持继承性是面向对象程序设计语言和传统程序设计语言在语言机制方面的根本区别。 （ ）

14. 面向对象的分析过程主要包括三项内容：理解、表达和验证。 （ ）

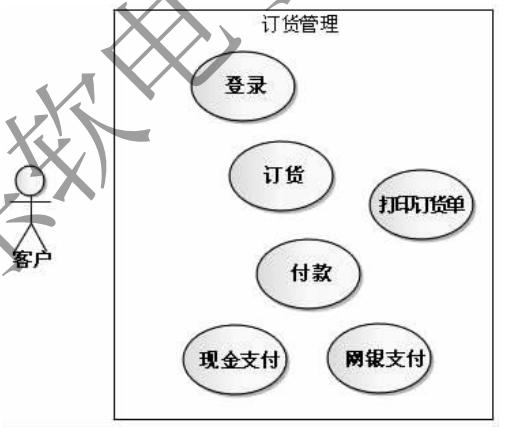
三、简答题

1. 消息传递机制与传统程序设计模式中的过程调用相比，有何本质区别？

2. 比较面向对象方法与结构化方法的特点，说明为什么面向对象方法比结构化方法更加优越。

3. 请详细阐述类图，对象图，用例图，状态机图，顺序图，活动图，通信图，部署图的作用，目的以及他们之间的相互关系。

4. 下面的用例图补充完整。



5. 画用例图

在图书管理系统中，管理员可进行“删除书籍”和“修改书籍信息”操作，但不论进行哪种作，都需要“图书查询”；读者还可以还书，如果所借书籍超期，需要交纳罚金。

6. 画顺序图并将其转化为协作图

用户登录系统，首先要和登录窗口交互，输入用户名和密码。登录窗口负责和服务器交互，将用户输入的用户名和密码发送到服务器，服务器再与数据库交互，以验证用户名和密码的有效性，如果验证成功，则返回 OK，验证失败返回 Error。服务器将通过登录窗口将信息显示给用户。

7. 画活动图

某教学系统操作员登录过程是：启动该系统，系统给出登录窗口，在登录窗口中需要输入用户名和密码，如果用户名或密码有误，则系统提示错误，操作员重新输入，若连续 3 次用户名或密码均没有输入正确，则系统拒绝登录。如果输入正确，则进入系统。用活动图描述操作员的登录过程。

8. 画状态图

首先，线程处于就绪态，当取得 CPU 时间片，进入运行状态；如果正常运行，则直到运行结束；如果运行中 CPU 时间片用完，则返回就绪态；如果运行中不满足所需资源，则进入阻塞状态，当系统满足资源时，重新进入就绪状态。