第七章 面向对象分析方法

一、用哲学的观点看面向对象

1、对象论观点：

1>数据和逻辑不是分离的，而是相互依存的。

2>相关的数据和逻辑形成个体，这些个体叫做对象，世界就是由一个个对象组成的。

3>对象具有相对独立性， 对外提供一定的服务。

4>所谓世界的演进，是在某个“初始作用力”作用下，对象间相互调用而完成的交互，在没有初始作用力下，对象保持静止。

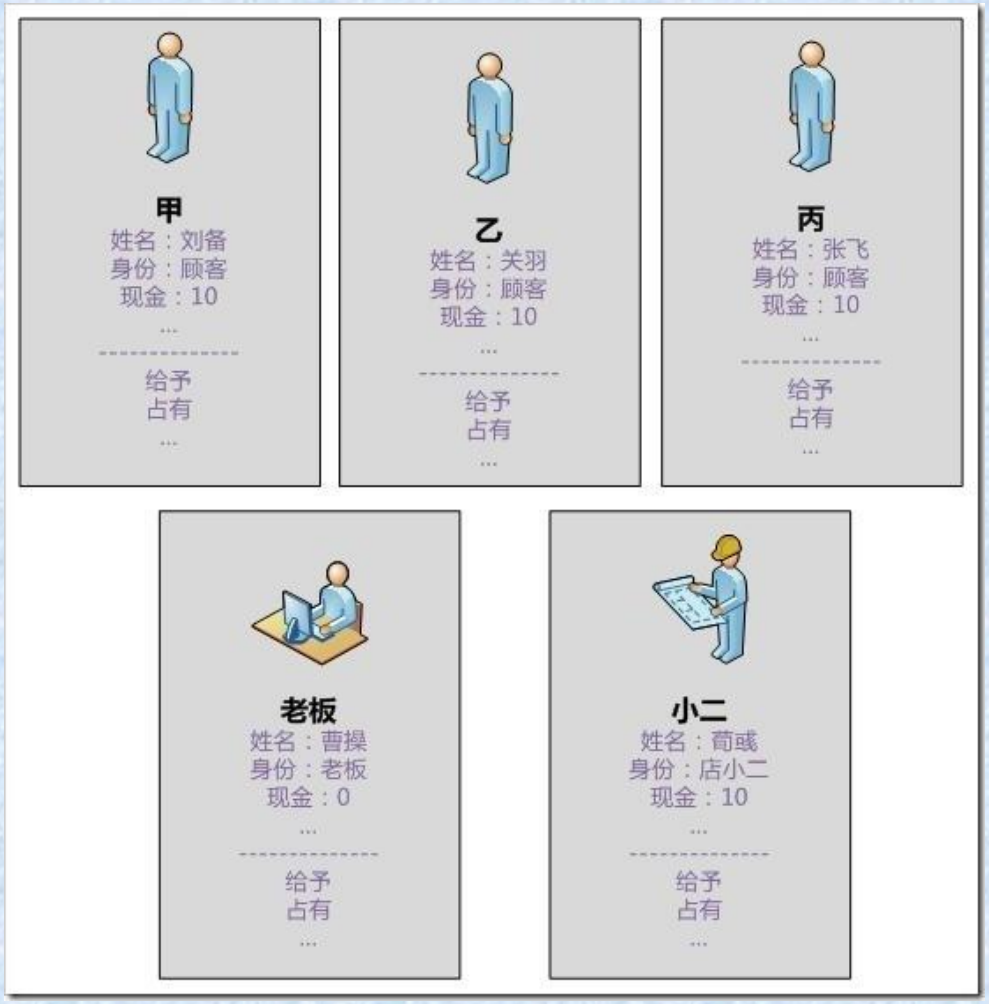
5>交互并不是完全预定义的，不一定有严格的因果关系，对象间交互是偶然的，对象间联系是暂时的。

结论：世界就是由各色对象组成，然后在初始作用力下，对象间的交互 完成了世界的演进。

问题：

有甲、乙、丙三人住店，一间房30。于是每人10元，共计给店老板30元住进一间房。后来店老板发现弄错了，房价应该是25元，于是给小二5元让小二退给 房客。小二黑心，贪污了2元，退给甲乙丙每人1元。这样房客每人付了10-1=9元，三九27，加上小二贪污的2元，共29元，问那1元哪里去了？

如何用对象论的观点解决以上问题？



对象论眼中，有五个基本对象，每个对象有自己的一系列数据和逻辑没错，在对象论眼里，这就是这件事的本质模样，这件事所涉及的东西就是这么几个对象。

本来它们各自独立，老死不相往来。只不过在住店这个外部驱动力下，几个对象偶然、暂时互相联系，利用其他对象提供的公开服务，完成了一些交互。

对应考虑面向对象分析的模型

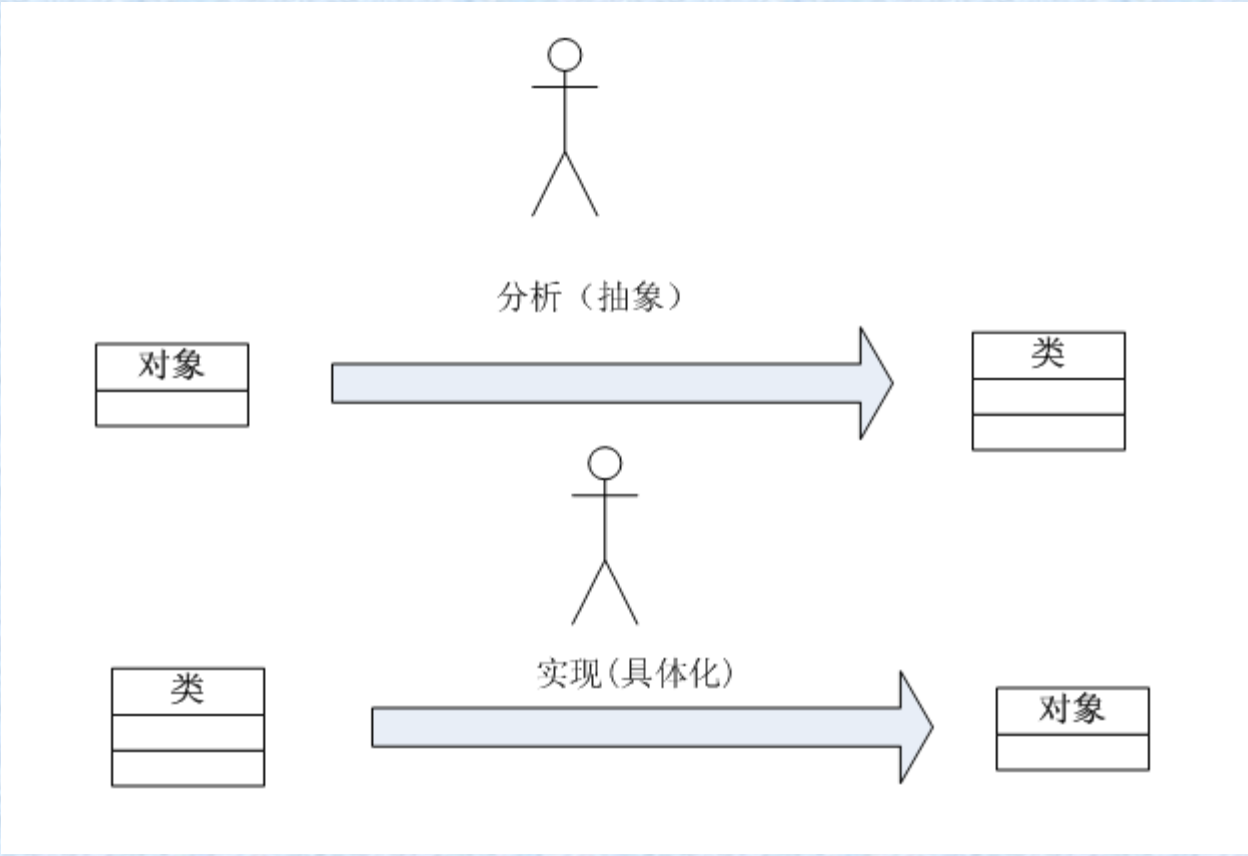
对象论观点

2、抽象

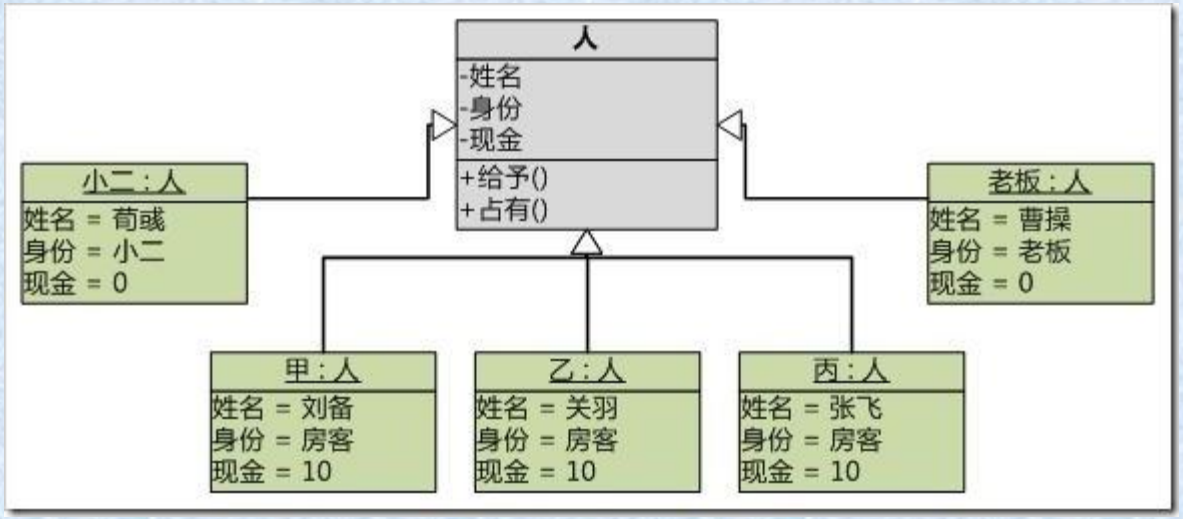
先有对象还是先有类？

从哲学角度说，先有对象，然后才有类，类和对象是“一般和特殊”这一哲学原理在程序世界中的具体体现

类其实是抽象认知能力作用于程序世界的基本元素——对象后所衍生出来的抽象概念，是抽象思维在程序世界中物化后的产物。现实世界中每个对象都 有无数的数据和逻辑，但在具体到程序世界时，我们往往只关心具体场景中相关的数据和逻辑。


分析（抽象）
对象
类
实现(具体化)
类
对象

上面的例子中五个对象很相似，可以看做一类东西，于是，我们给出一个类，叫“人”，并且认为这五个对象都是“人”这个类的具体例子，我们叫其为实例。以后遇到类似的对象，我们都可以知道，这个对象属于“人”类。



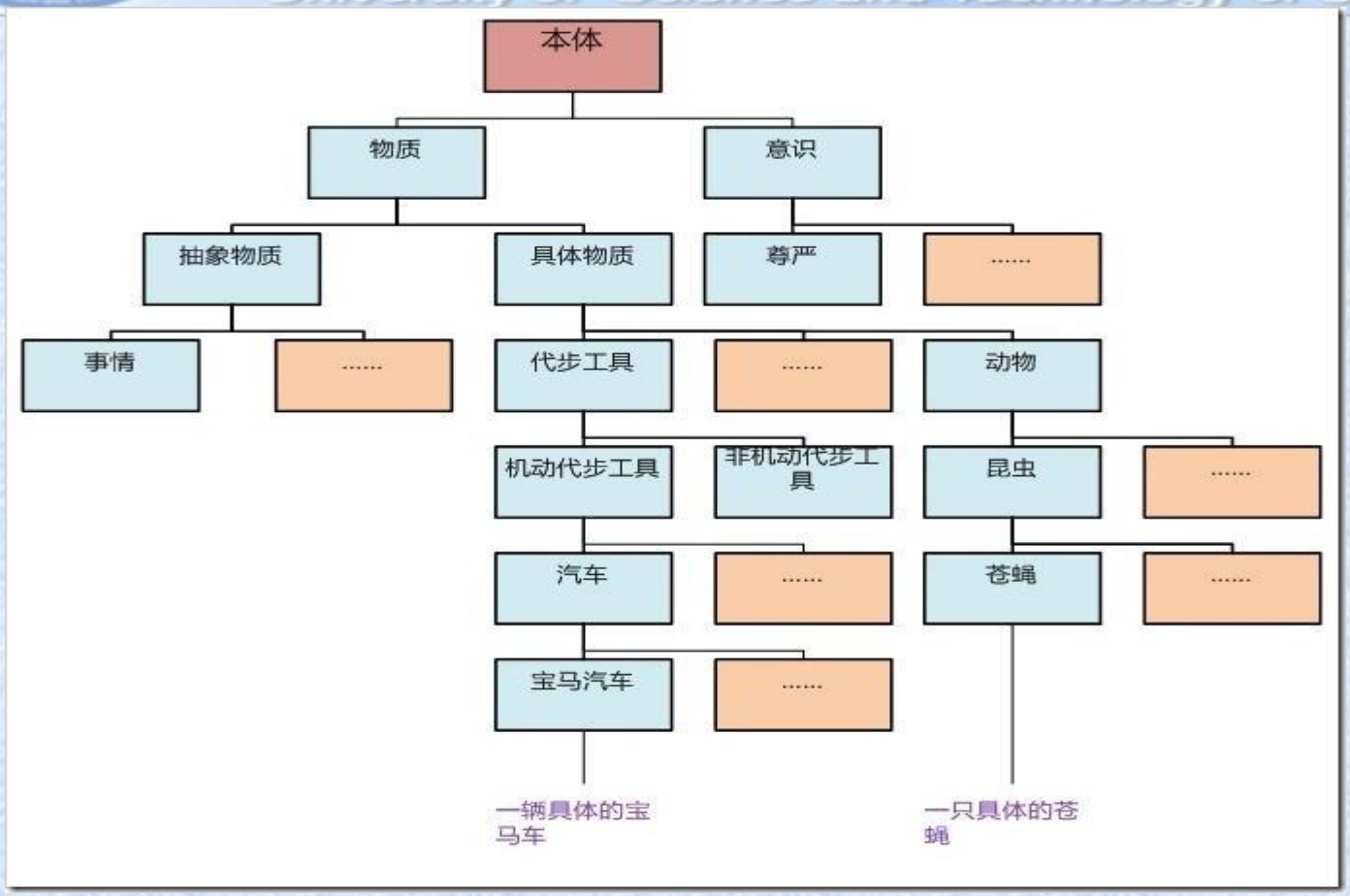
3、层次

“道生一，一生二，二生三，三生万物—老子”

抽象是有层次的

世界可并不是一层对象一层类那么简单，对象抽象出类，在类的基础上可以再进行抽象，抽象出更高层次的类。所以经过抽象的对象论世界，形成了一个树状结构

抽象层次树



本体即万物之源、万物之本，是哲学层面上最高层次的抽象

注意：

抽象层次树理论也是OO中许多内容的理论基础。

Eg:OO中重要的概念——继承和多态，如若探究其哲学本源，就是从这里来的。

- 这是一棵单根树，最顶层“本体”为唯一的根，最下层叶子节点为基本对象。一切中间节点都为类。
- 越往上的类抽象层次越高，具体度越低，其内涵越小，外延越大；越往下的类抽象层次越低，具体度越高，其内涵越大，外延越小

  内涵:指 类对属于自己的对象的说明力度

  外延:指类能包含的具体对象的总和。
- 抽象层次树不是从根部向下长的，而是从叶子节点向上归纳生成的。
- 某一个叶子节点所代表的对象可以归入所有其祖先结点所代表的类
- 直接问两个叶子节点属不属于一个类没有意义，而要指定抽象层次才有意义。例如在较低层，一辆宝马属于汽车，而一只苍蝇属于昆虫，不是一类。但如果指定在较高层比较，两个都属于具体物质，属于一个类。

- 我们定义，如果一个节点CNode非叶子节点也非根节点，那么在哲学意义上，这个节点继承于其父节点PNode，并且说PNode是CNode的泛化。
- 我们定义，如果一个节点CNode非叶子节点也非根节点，如果强行将它看成其任何一个祖先节点ANode，并当做ANode使用，那么在哲学意义上，叫做多态性

4、继承与泛化

抽象层次树上，除根节点和叶子节点外，任一节点CNode非严格继承其所有祖先节点所组成的集合中的任一元 素，而CNode严格继承其父节点PNode。泛化与继承相反。

注意：从哲学和认识论角度来说，是先有对象，然后有类；先有子类，然后有父类，是一种自底向上形成的体系。 而继承一词，就容易让人误解成是先有父类才有子类。所以推荐使用泛化来理解层次树上的关系

5、耦合

“一只蝴蝶在巴西轻拍翅膀，可以导致一个月后德克萨斯州的一场龙卷风——蝴蝶效应”

普遍联系： 世界上各种对象形成了一张复杂的耦合网，正因为有耦合的存在，世界才能演进。正如马克思主义哲学所说：联系是普遍的、客观的。所以， 耦合的存在，有其深刻的哲学意义。

什么是耦合？

可以看成联系，耦合的存在是世界演进的途径，如果没有耦合，世界就变成了“死世界”，无法演进和发展。所以，世界需要耦合 。我们在开发设计中要降低耦合，但并不是要取消耦合

泛化耦合：

由于泛化（继承）关系的存在，在两个有祖孙、父子关系的类间形成的一种逻辑关联。

聚合：

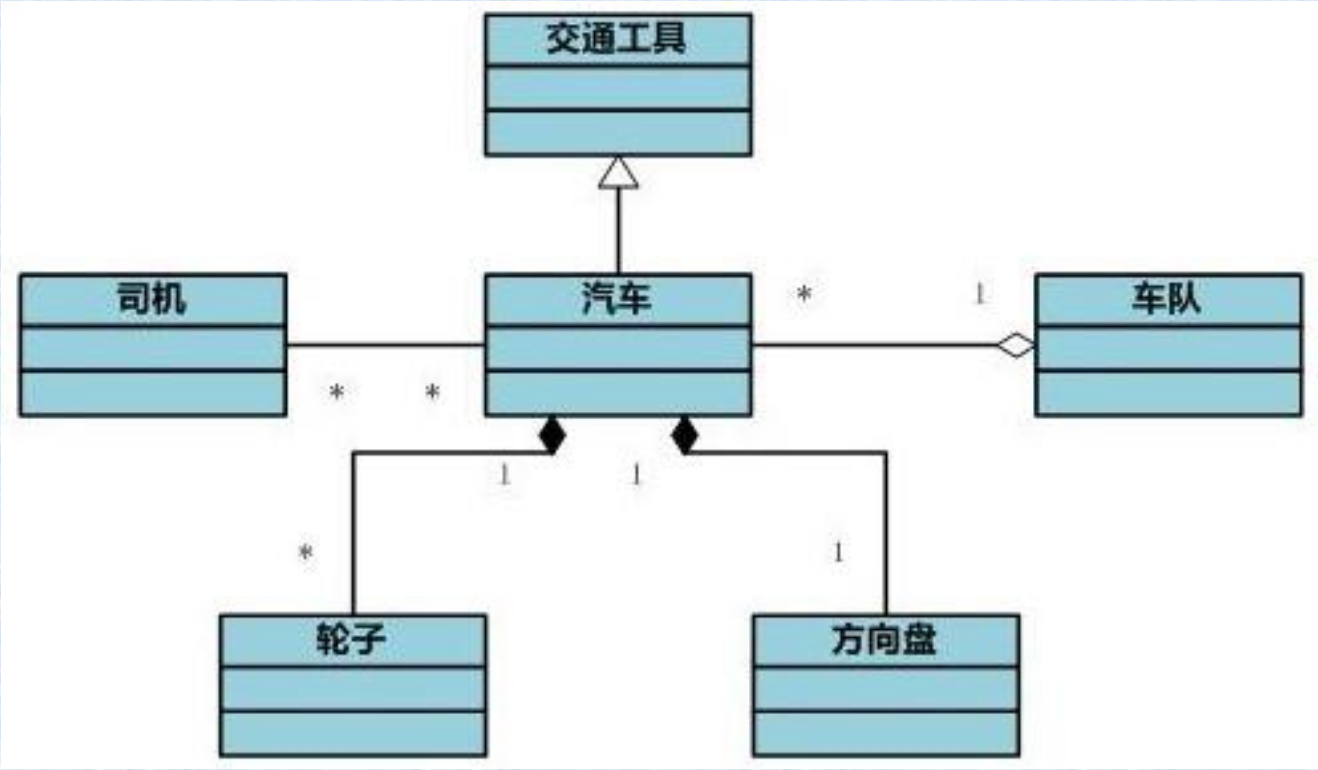
一种弱的拥有关系，体现A对象可以包含B对象，但B对象不是A对象的一部分。

组合：

一种强的拥有关系，体现了严格的部分和整体的关系，部分和整体具有一样的生命周期。

依赖：

由于逻辑上相互协作可能，而形成的一种关系。



其中汽车和交通工具属于泛化耦合，轮子和方向盘组合于汽车，汽车聚合成车队，而汽车和司机具有依赖关系。

二、OO的具体概念

- 客观世界中的应用问题都是由实体及其相互关系构成的。
- 可以将客观世界中与应用问题有关的实体及其属性抽象为问题空间中的对象。
- 为应用问题寻求软件解，是借助于计算机语言对其提供的实体施加某些动作，以动作的结果给出问题的解。
- 面向对象(Object-Oriented,简称OO)的需求分析方法通过提供对象、对象间消息传递等语言机制让分析人员在解空间中直接模拟问题空间中的对象及其行为

在设计和实现中，面向对象的方法是一种运用对象、类、继承、封装、聚合、消息传送、多态性等概念来构造系统的软件开发方法。

面向对象=对象（object）

+类（classification）

+继承（inheritance）

+通信（communication with messages）

可以说，采用这四个概念开发的软件系统是面向对象的。

分析、设计不分开

面向对象方法成为主流开发方法。可以从下列几个方面来分析其原因：

- 从认知学的角度来看，面向对象方法符合人们对客观世界的认识规律。
- 面向对象方法开发的软件系统易于维护，其体系结构易于理解、扩充和修改。
- 面向对象方法中的继承机制有力支持软件的复用。

注意：并没有摒弃面向过程的方法

1. 对象（object）

对象是指一组属性以及这组属性上的专用操作的封装体。

属性（attribute）通常是一些数据，有时它也可以是另一个对象。每个对象都有它自己的属性值，表示该对象的状态。对象中的属性只能通过该对象所提供的操作来存取或修改。

操作（operation）（也称方法或服务）规定了对象的行为，表示对象所能提供的服务。

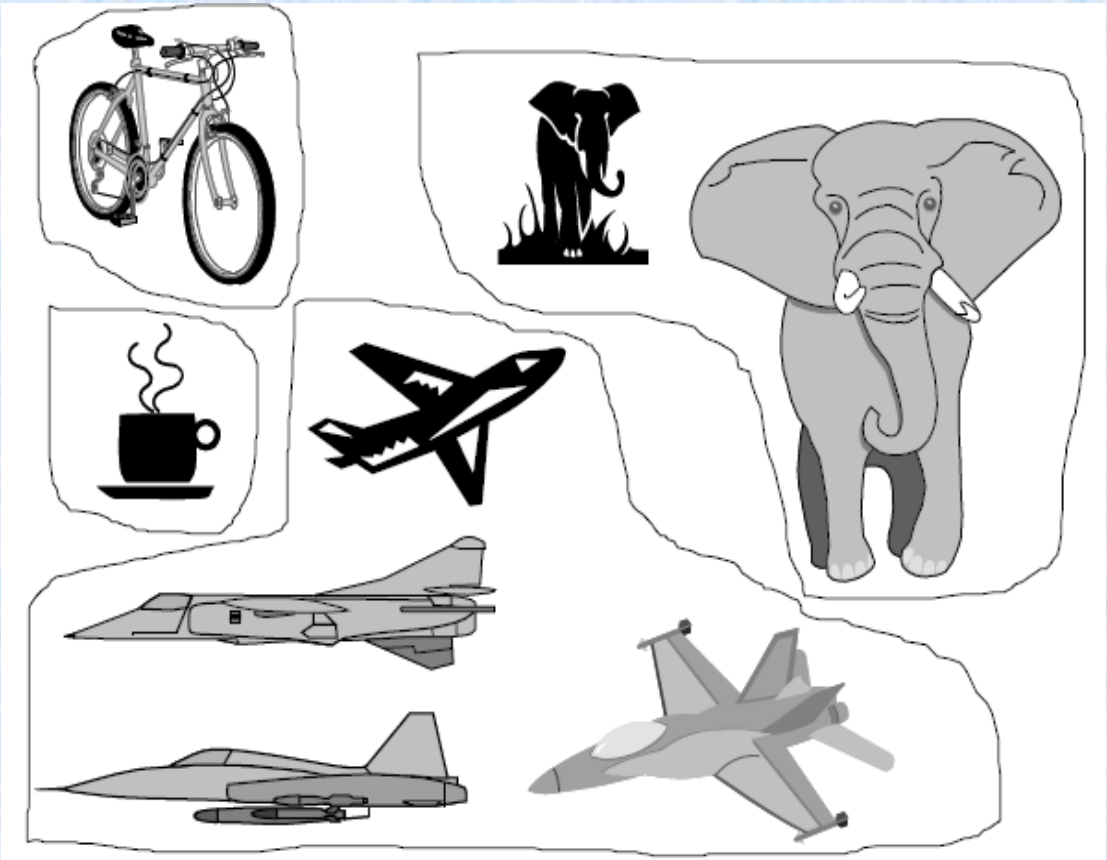
封装（encapsulation）是一种信息隐蔽技术，用户只能看见对象封装界面上的信息，对象的内部实现对用户是隐蔽的。

封装的目的是使对象的使用者和生产者分离，使对象的定义和实现分开。

2. 类（class）

在对系统的分析中可通过抽象可以到类，在设计和实现中，类具体表现为一组具有相同属性和相同操作的对象的集合。一个类中的每个对象都是这个类的一个实例。

在面向对象的实现中，类是创建对象的模板，从同一个类实例化的每个对象都具有相同的结构和行为。



Elephant
Color : text Number of tusks : Integer Location: text Weight: float Height: float
move_to (location) wash (date) feed (amount, date, time)

对象和类的描述

对象和类一般采用“对象图” 和 “类图” 来描述。

类图

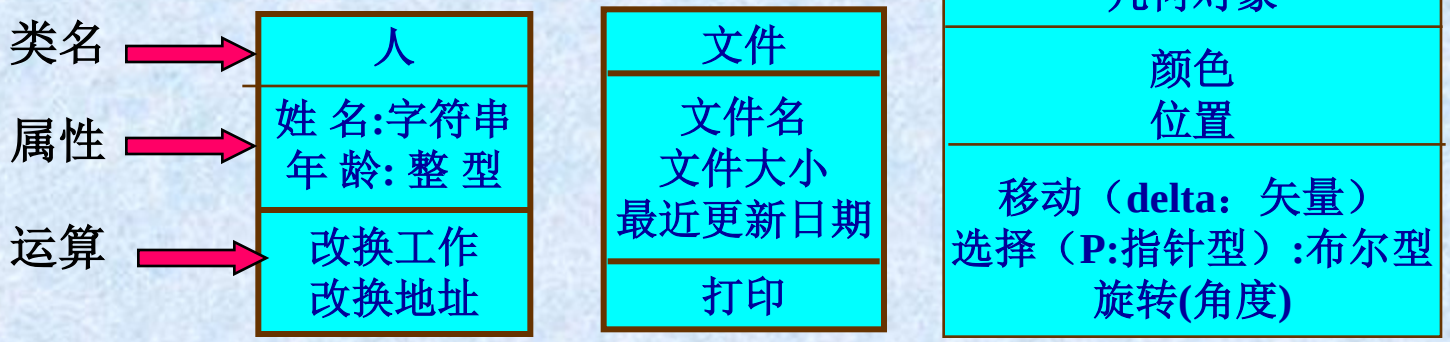
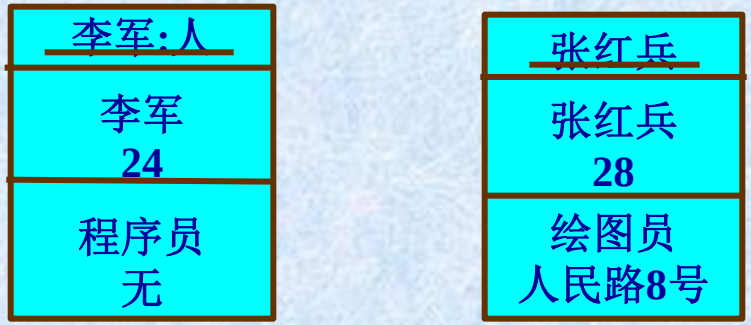
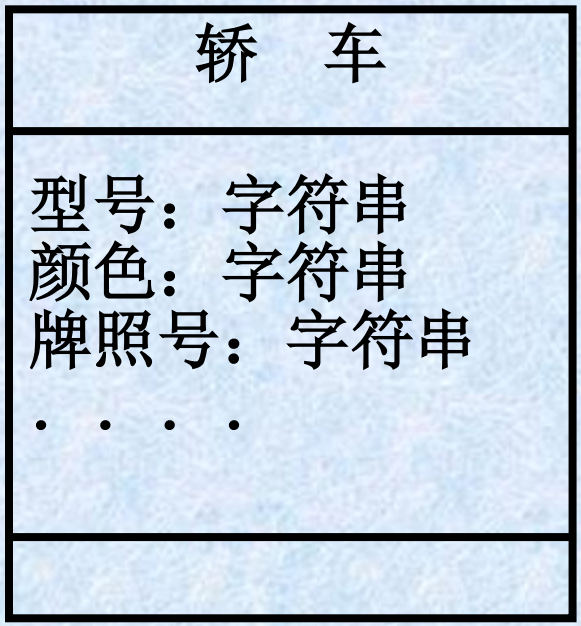


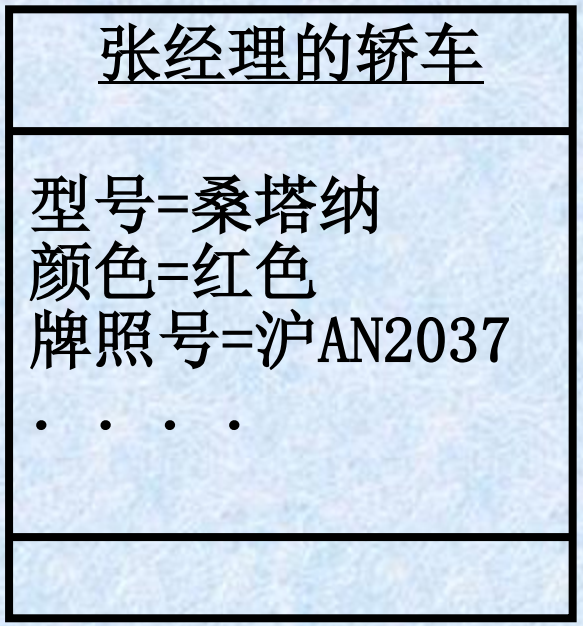
图　对象类的描述

对象图





类



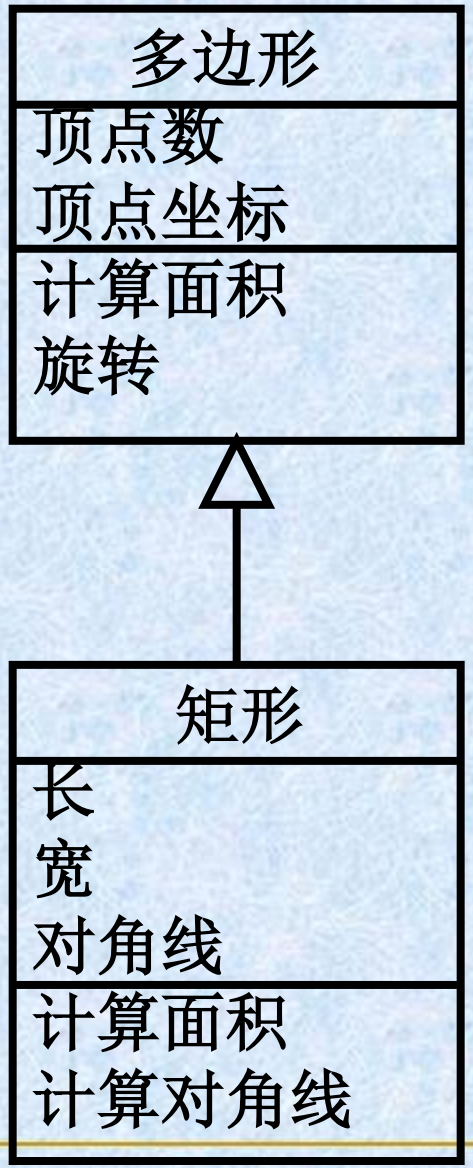
实例对象

3.继承（inheritance）

继承表明了两个类在抽象层次树上的关系。在设计和实现中，通过继承可以实现不同类共享数据和操作。

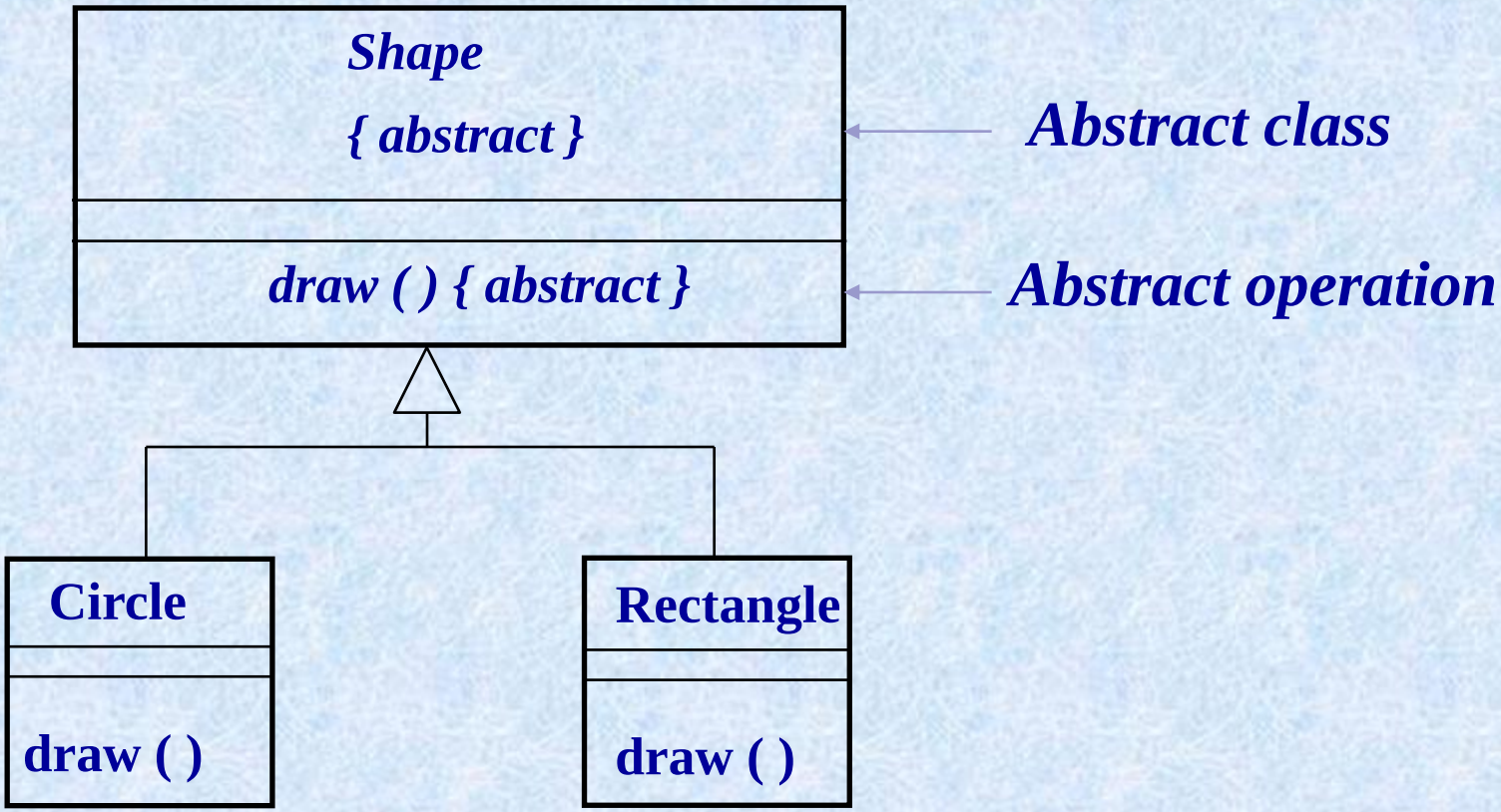
父类中定义了其所有子类的公共属性和操作，在子类中除了定义自己特有的属性和操作外，可以继承其父类（或祖先类）的属性和操作，还可以对父类（或祖先类）中的操作重新定义其实现方法。

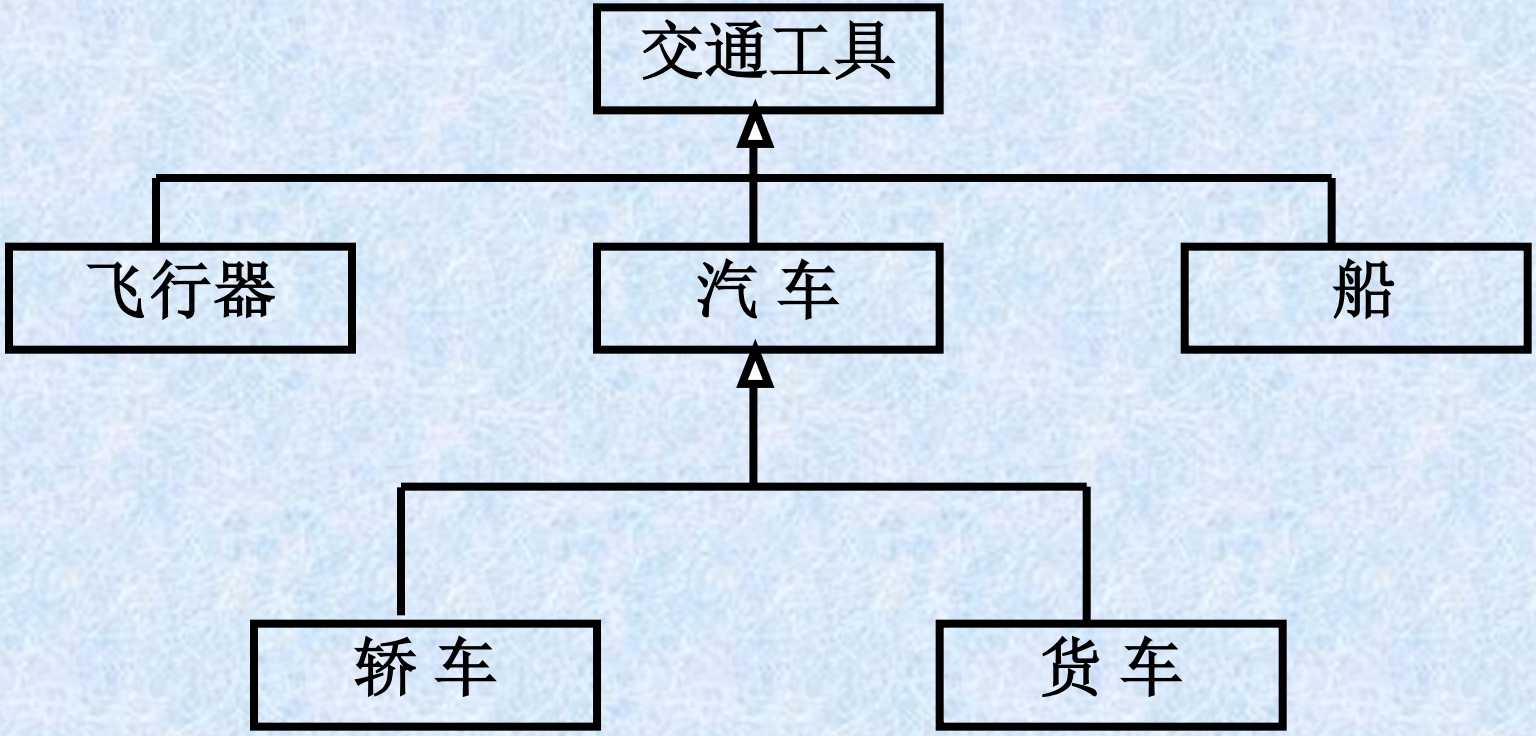
继承可以实现代码的重用。


多边形
顶点数
顶点坐标
计算面积
旋转
矩形
长
宽
对角线
计算面积
计算对角线

抽象类(abstract class)：没有实例的类，它把一些类组织起来，提供一些公共的行为，但并不需要使用这个类的实例，而仅使用其子类的实例。

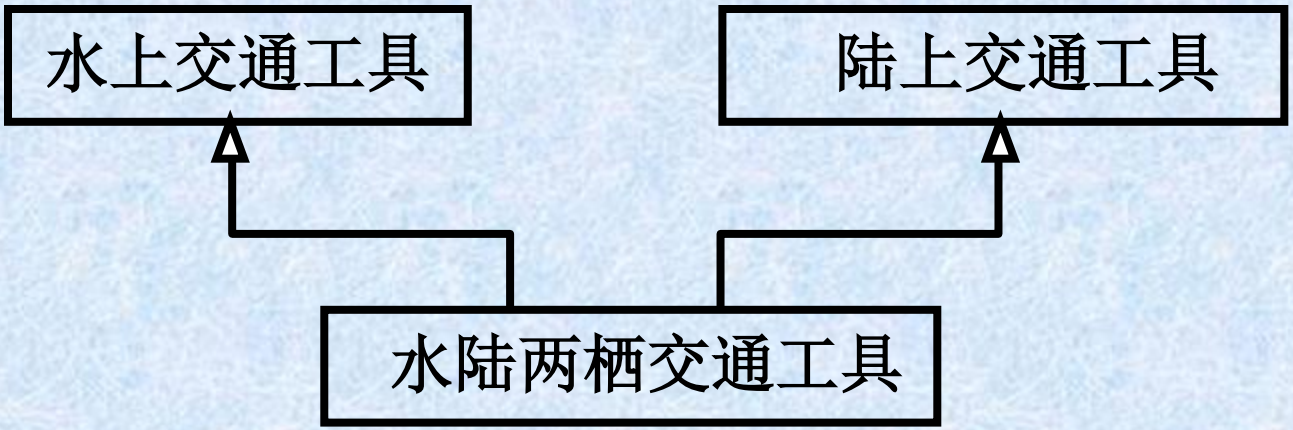
在抽象类中可以定义抽象操作，抽象操作指：只定义这个类的操作接口，不定义它的实现，其实现部分由其子类定义。抽象操作操作名用斜体字表示，也可以在操作特征（signature）后面加上特征字符串{abstract}。


Shape
{ abstract }
draw ( ) { abstract }
Abstract class
Abstract operation
Circle
draw ( )
Rectangle
draw ( )



一般-特殊关系

如果一个子类只有唯一一个父类，这个继承称为单一继承。如果一个子类有一个以上的父类，这种继承称为多重继承。（Java能否多继承？C#呢？）



多重继承

4. 消息，方法，属性

- 消息就是某个操作的规格说明，其组成：
  - 接收消息的对象
  - 消息名（消息选择符）
  - 零个、一个或多个变元

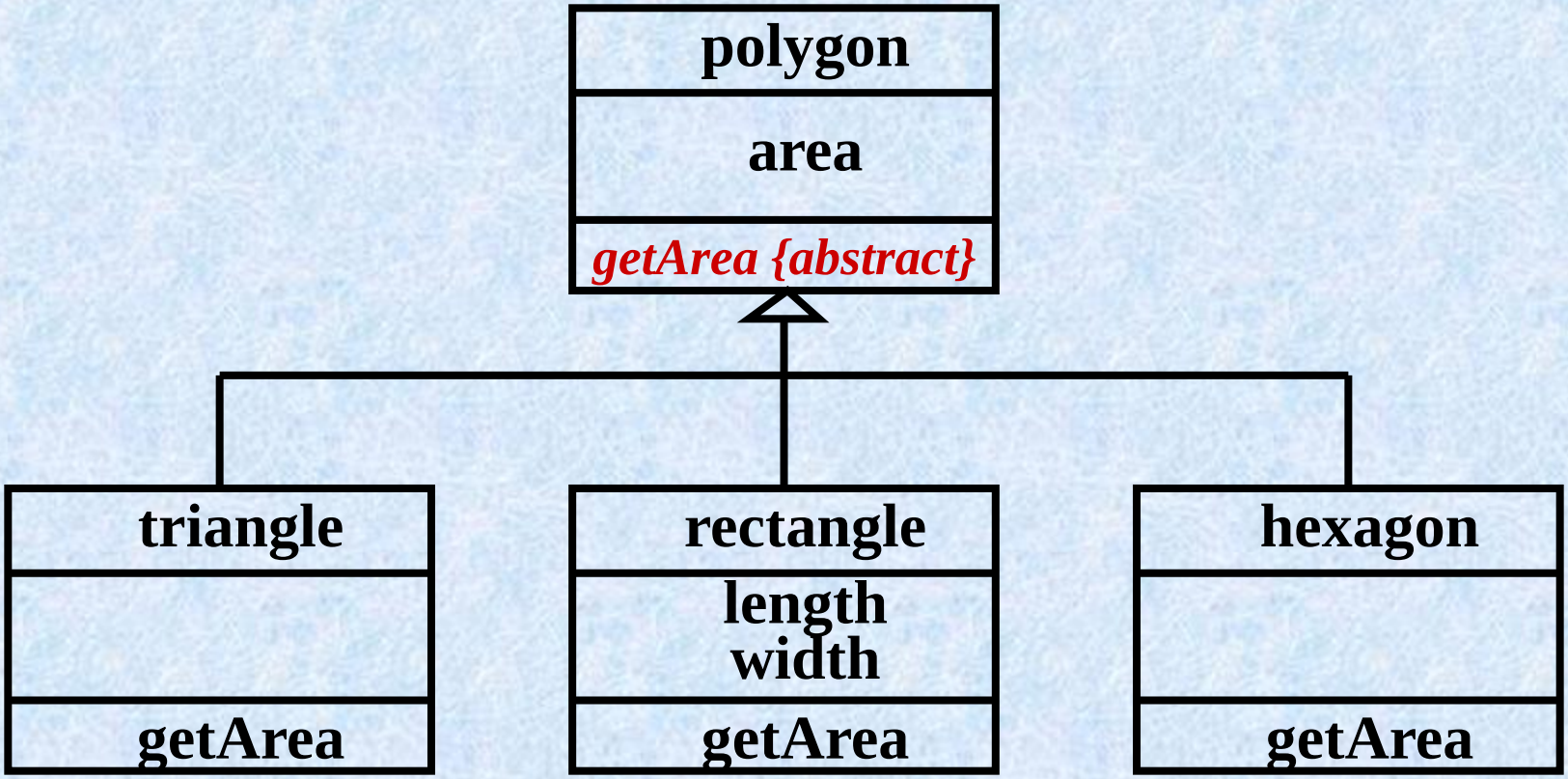
  例如：对于类Circle的一个实例MyCircle，如果使其以绿色在屏幕上显示，MyCircle . Show (GREEN);
- 方法（操作、服务）
  - 对象所能执行的操作，即类中所定义的服务。它是对操作算法和响应消息办法的描述。
  - 在类Circle中给出成员函数Show (int color)的定义。
- 属性是类中所定义的数据，是实体性质的抽象
  - 类实例都有其特有的属性值，如类Circle定义的圆心、半径和颜色。

5. 多态性与动态绑定

多态性（polymorphism）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。

在一般与特殊关系中，子类是父类的一个特例，所以父类对象可以出现的地方，也允许其子类对象出现。因此在运行过程中，当一个对象发送消息请求服务时，要根据接收对象的具体情况将请求的操作与实现的方法进行连接，即动态绑定。


polygon
area
getArea {abstract}
triangle
getArea
rectangle
length
width
getArea
hexagon
getArea

6、永久对象(Persistent object)

所谓永久对象是指生存期可以超越程序的执行时间而长期存在的对象。

目前，大多数OOPL不支持永久对象，如果一个对象要长期保存，必须依靠于文件系统或数据库管理系统实现，程序员需要作对象与文件系统或数据库之间数据格式的转换，以及保存和恢复所需的操作等烦琐的工作。

三、面向对象分析

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。

需求分析工作主要包括：理解、表达、验证。面向对象需求分析过程和模型如下：

- 需求描述
- 静态模型（对象模型）
- 动态模型（交互次序）交互图、状态图
- 功能模型（数据交换）

注意：需求分析的过程和分析模型的内容并没有严格的划分和限定。（有可能和设计模型重叠）

1 需求描述

需求描述的内容包括：问题范围，功能需求，性能需求，应用环境及假设条件等。需求的描述可以通过前面介绍的需求导出阶段获得。

无歧义的描述方法：数学符号描述

如何描述需求？

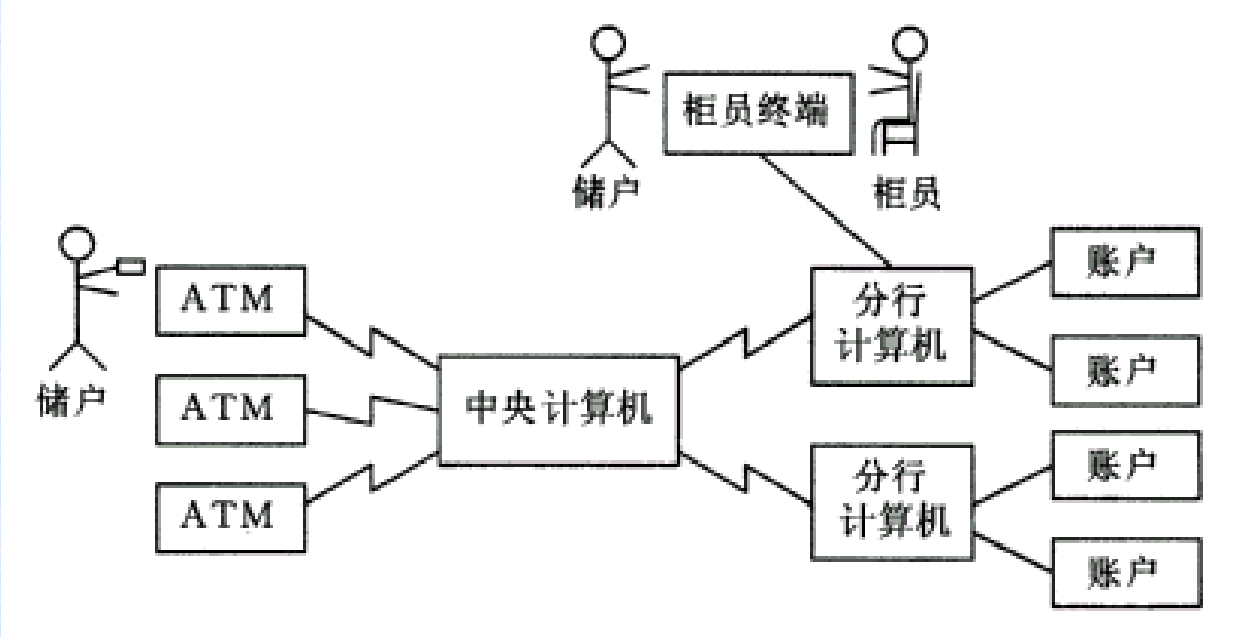
- 图型化方法
- 格式化文档

图形化表示便于理解，尤其在需求描述很复杂的情况下

eg: 用例图，用例模板

举例：银行ATM

需求文字描述



2 静态模型（对象模型）

1>、找出类-对象并进行筛选；
2>、找出对象之间的关联；
3>、划分主题
4>、找出对象的属性；
5>、用继承组织和简化对象类；
6>、迭代并反复提炼模型。

1>找出类-对象并进行筛选

- 找出候选的类—对象

有多种分析方法获得类-对象。eg:语法分析法：

如果我们有一个项目需要做需求分析，这时客户和你交谈的时候因该注意什么呢？

动词　名词

△ 名词通常是我们面向对象中类的表示

△动词通常是我们面向对象中事件，方法，行为的表示

△把握好名词动词，并且把它们合理的抽象出来，在软件开发的前期有着重要作用

例如：比赛的目标是要把篮球投入篮框并且要尽量比对手得更多的分数。每个篮球队由5名队员：两个前锋，两名后卫和一名中锋。

上面一小段文字里面，能够抽象出什么？？？

例如：比赛的目标是要把篮球投入篮框并且要尽量比对手得更多的分数。每个篮球队由5名队员：两个前锋，两名后卫和一名中锋。

红色为名词，蓝色为动词

候选类-对象：篮球、队员、篮筐、球队、分数

ATM例子：

银行、自动取款机(ATM)、系统、中央计算机、分行计算机、柜员终端、网络、总行、分行、软件、成本、市、街道、营业厅、储蓄所、柜员、储户、现金、支票、账户、事务、现金兑换卡、余额、磁卡、分行代码、卡号、用户、副本、信息、密码、类型、取款额、账单以及访问。

注意：

通常，在需求描述中不会一个不漏地写出问题域中所有有关的类–对象，因此，分析员应该根据领域知识或常识进一步把隐含的类–对象提取出来。

例如，在ATM系统的描述中没写“通信链路”和“事务日志”，但是，根据领域知识和常识可以知道，在ATM系统中应该包含这两个实体。

- 筛选出正确的类—对象

筛选时主要依据下列标准，删除不正确或不必要的类—对象。

1）冗余

2）无关

3）笼统

4）属性

5）操作

ATM系统筛选后的类-对象

ATM、中央计算机、分行计算机、柜员终端、总行、分行、柜员、储户、账户、事务和现金兑换卡。

2> 找出对象之间的关联

两个或多个对象之间的相互依赖、相互作用的关系就是关联。分析确定关联，能促使分析员考虑问题域的边缘情况，有助于发现那些尚未被发现的类–对象。

●初步确定关联

找对象之间关系的方法

△动词或动词词组，通常表示关联关系。

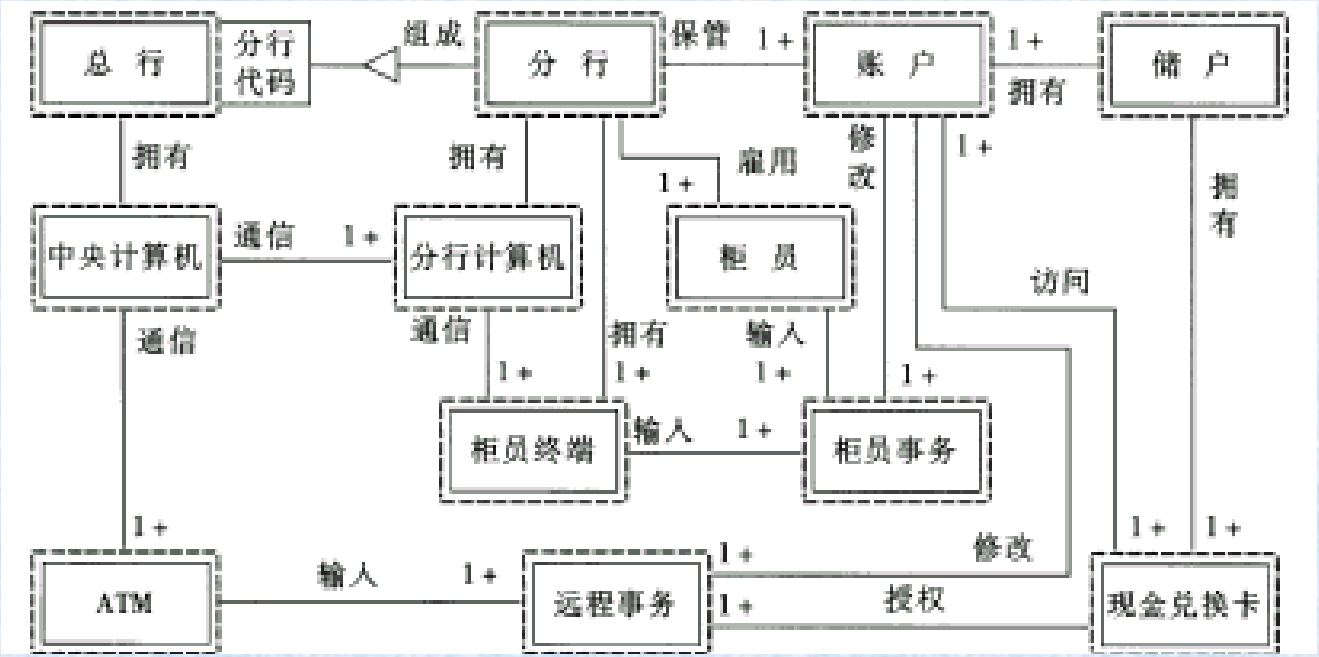
△通过分析需求描述，还能发现一些在陈述中隐含的关联。

△最后，分析员还应该与用户及领域专家讨论问题域实体间的相互依赖、相互作用关系，根据领域知识再进一步补充一些关联。

- 筛选

经初步分析得出的关联只能作为候选的关联，还需经过进一步筛选，以去掉不正确的或不必要的关联。

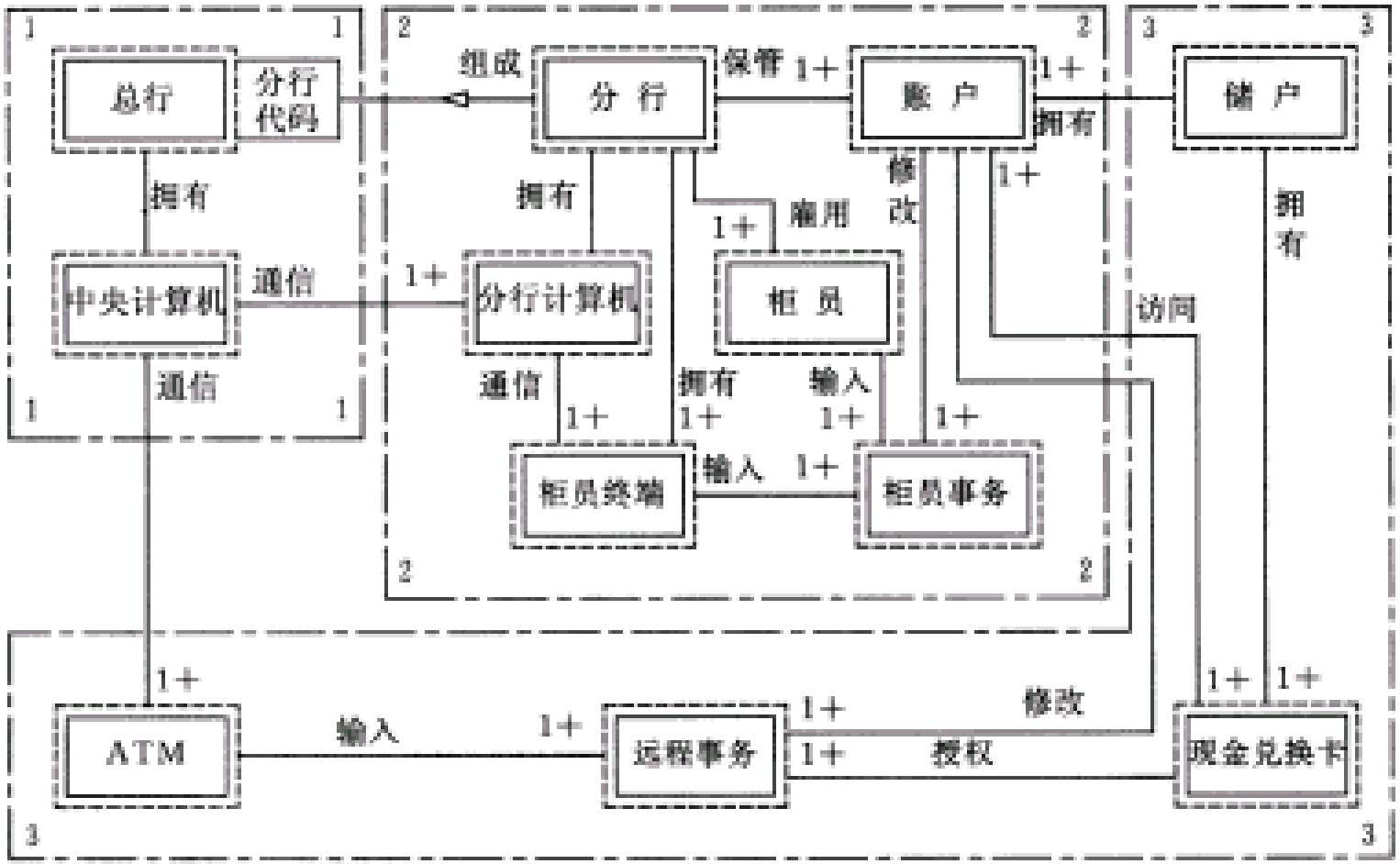
注意：前面提到的4种耦合关系


总 行
分行代码
组成
分 行
保管
1+
账 户
1+
拥有
储 户
拥有
拥有
雇用
1+
修改
1+
拥有
中央计算机
通信
1+
分行计算机
柜 员
访问
通信
通信
拥有
1+
1+
输入
1+
1+
柜员终端
输入
1+
柜员事务
1+
1+
1+
修改
ATM
输入
1+
远程事务
1+
授权
现金兑换卡

ATM系统原始对象图

3>划分主题

在开发大型、复杂系统的过程中，为了降低复杂程度，人们习惯于把系统再进一步划分成几个不同的主题，也就是在概念上把系统包含的内容分解成若干个范畴。



把ATM系统划分成三个主题

4〉找出对象属性

一般说来，确定属性的过程包括分析和选择两个步骤。

分析:

应该仅考虑与具体应用直接相关的属性，不要考虑那些超出所要解决的问题范围的属性。在分析过程中应该首先找出最重要的属性，以后再逐渐把其余属性增添进去。在分析阶段不要考虑那些纯粹用于实现的属性。

选择:

认真考察经初步分析而确定下来的那些属性，从中删掉不正确的或不必要的属性。通常有以下几种常见情况。

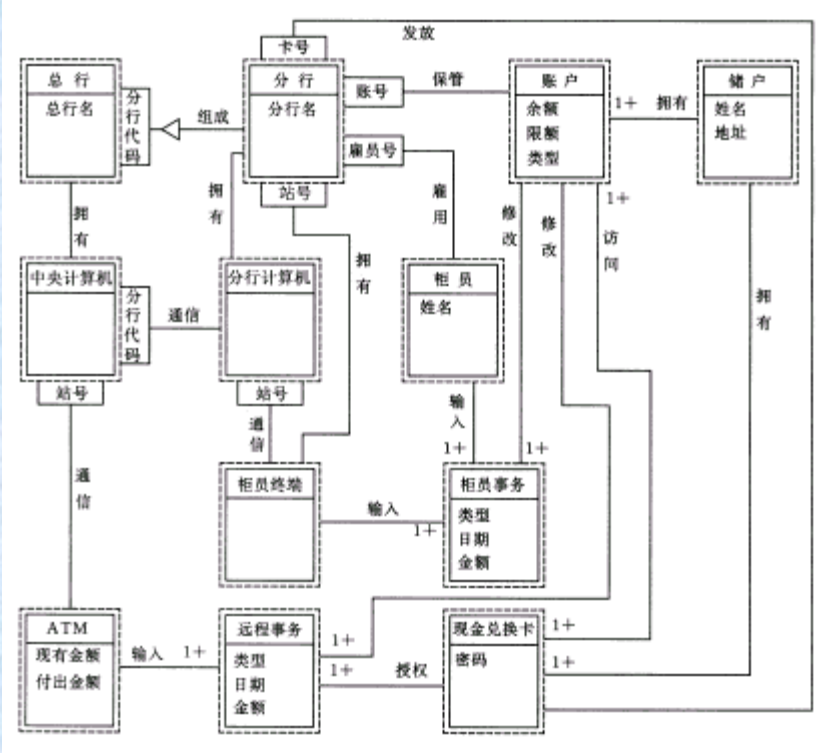
(1) 误把对象当作属性

(2) 把限定误当成属性

(3) 误把内部状态当成了属性

(4) 过于细化

(5) 矛盾的属性



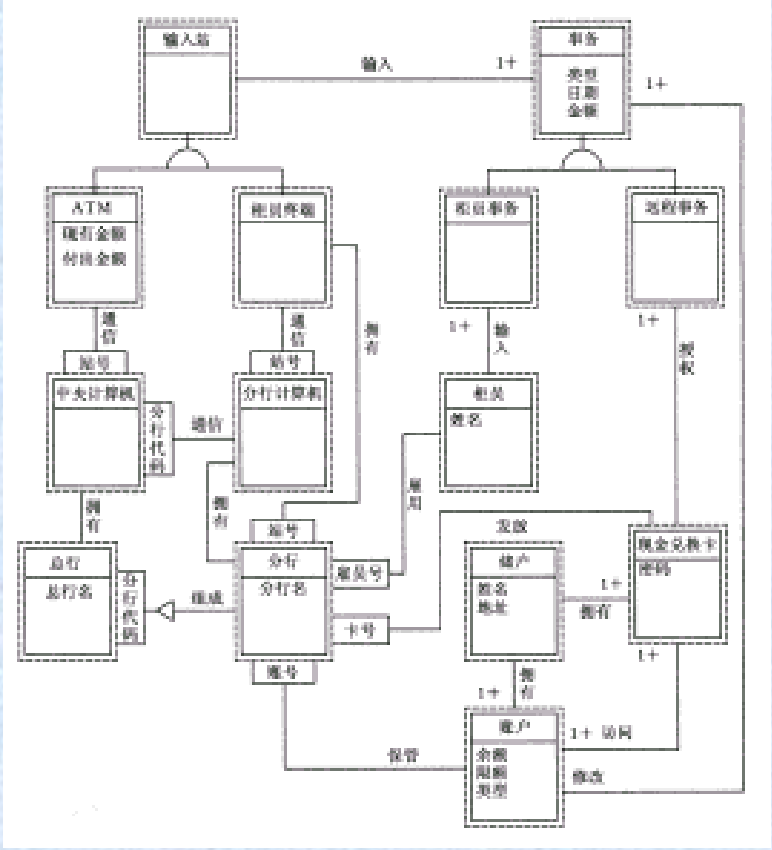
ATM对象模型中的属性

5>通过抽象用继承组织和简化对象类

确定了类中应该定义的属性之后，就可以利用继承机制共享公共性质，并对系统中众多的类加以组织。

一般说来，可以使用两种方式建立继承关系。

- 自底向上：抽象出现有类的共同性质泛化出父类，这个过程实质上模拟了人类归纳思维过程。
- 自顶向下：把现有类细化成更具体的子类，这模拟了人类的演绎思维过程。



带有继承关系的ATM对象模型

6>迭代并反复提炼模型

仅仅经过一次建模过程很难得到完全正确的对象模型。事实上，软件开发过程就是一个多次反复修改、逐步完善的过程。在建模的任何一个步骤中，如果发现了模型的缺陷，都必须返回到前期阶段进行修改。

3 动态模型

动态模型是与时间和变化有关的系统性质。该模型描述了系统的控制结构，它表示了瞬时的、行为化的系统控制性质；它关心的是系统的控制，操作的执行顺序；它从对象的事件和状态的角度出发， 表现了对象的相互行为。

如何建立动态模型？

1>编写脚本；

2>设计用户界面；

3>画事件跟踪图

4>画状态图；

5>审查动态模型。

1>编写脚本

脚本是指系统某一执行期间内出现的一系列事件。脚本范围可以是变化的，它可包括系统中所有事件，也可以只包括被某些对象触发或产生的事件。脚本可以是执行系统的历史记录，也可以是执行系统的模块。

使用电话的脚本：　　呼叫者拿起电话；

响拨号声；

呼叫者拨号；

事件

- 事件的含义

现实世界，各对象之间相互触发，一个触发行为就称作是一个事件。

对事件的响应取决于接受该触发的对象的状态，响应包括状态的改变或形成一个新的触发。

- 事件类

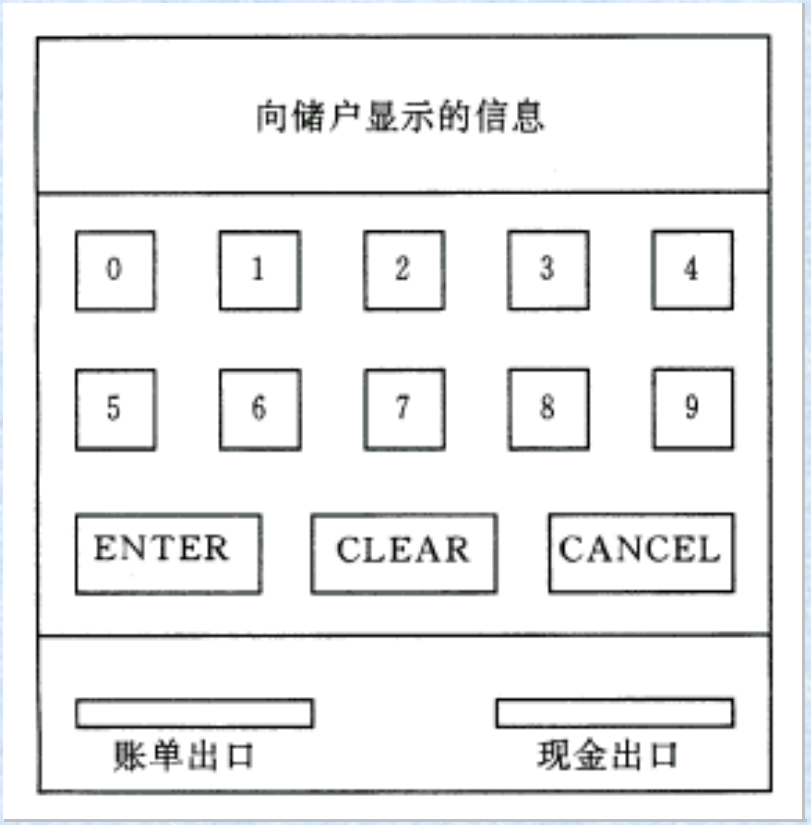
把各个独立事件的共同结构和行为抽象出来，组成事件类， 给每个类命名，这种事件类的结构也是层次的，大多数事件类具有属性，用来表明传递的信息，但有的事件类仅仅是简单的信号。

ATM系统的正常情况脚本。

- ATM请储户插卡；储户插入一张现金兑换卡
- ATM接受该卡并读它上面的分行代码和卡号
- ATM要求储户输入密码；储户输入自己的密码“1234”等数字
- ATM请求总行验证卡号和密码；总行要求“39”号分行核对储户密码，然后通知ATM说这张卡有效
- ATM要求储户选择事务类型(取款、转账、查询等)；储户选择“取款”
- ATM要求储户输入取款额；储户输入“880”
- ATM确认取款额在预先规定的限额内，然后要求总行处理这个事务；总行把请求转给分行，该分行成功地处理完这项事务并返回该账户的新余额[
- ATM吐出现金并请储户拿走这些现象；储户拿走现金
- ATM问储户是否继续这项事务；储户回答“不”
- ATM打印账单，退出现金兑换卡，请储户拿走它们；储户取走账单和卡
- ATM请储户插卡

2>设计用户界面

方便用户理解用户和系统的交互。（界面原型）



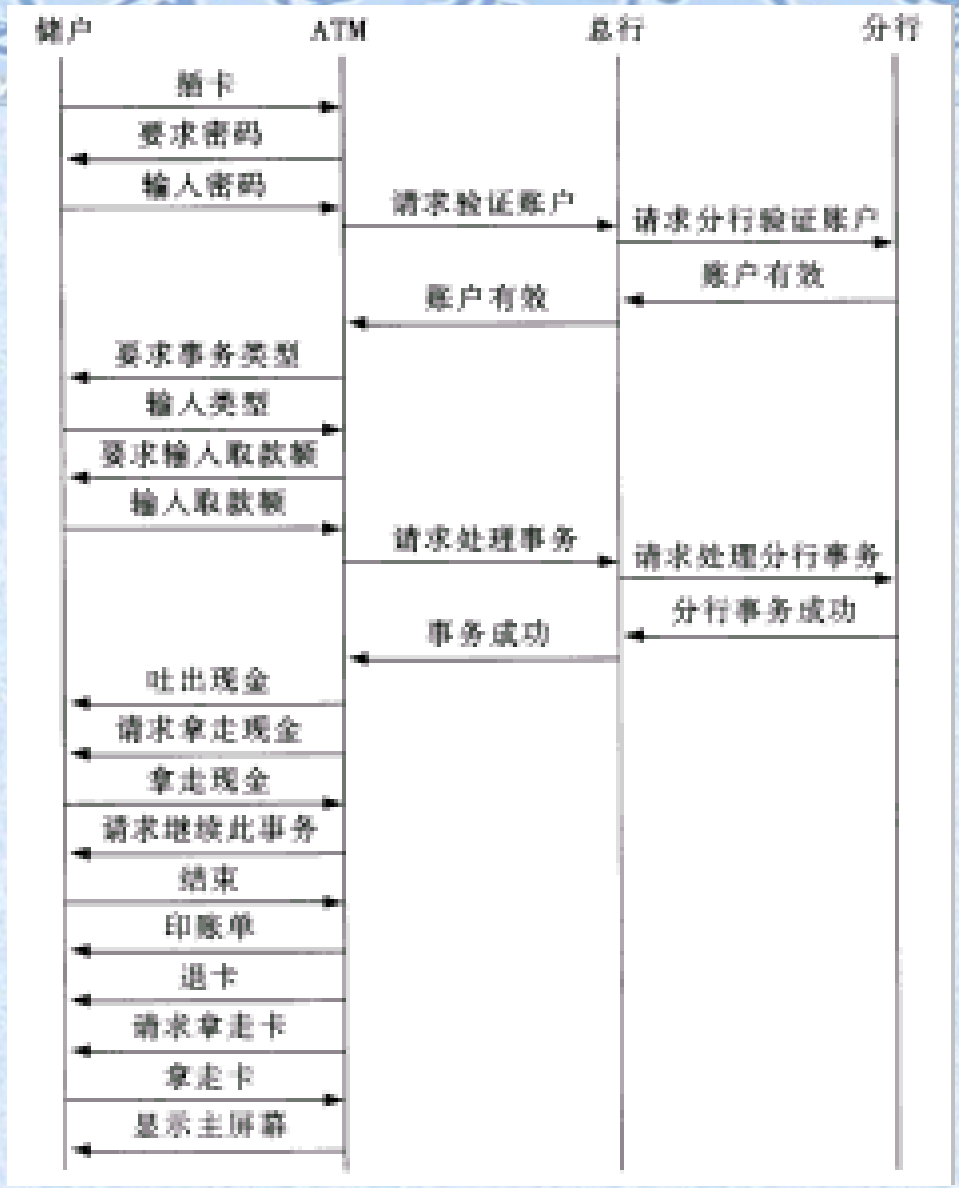
ATM的界面设计

3>画事件跟踪图

为什么要用时间跟踪图？

完整、正确的脚本为建立动态模型奠定了必要的基础。但是，用自然语言书写的脚本往往不够简明，而且有时在阅读时会有二义性。为了有助于建立动态模型，通常在画状态图之前先画出事件跟踪图。为此首先需要进一步明确事件及事件与对象的关系。

用事件跟踪图来表示事件、事件的接收对象和发送对象，接收和发送对象位于垂直线顶端。各事件用水平箭头线表示， 箭头方向是从发送对象指向接收对象，时间从上到下递增。



对应UML的顺序图

ATM系统正常情况脚本的事件跟踪图

4>画状态图

为什么需要状态图？

状态图描绘事件与对象状态的关系。当对象接受了一个事件以后，它的下个状态取决于当前状态及所接受的事件。由事件引起的状态改变称为“转换”。

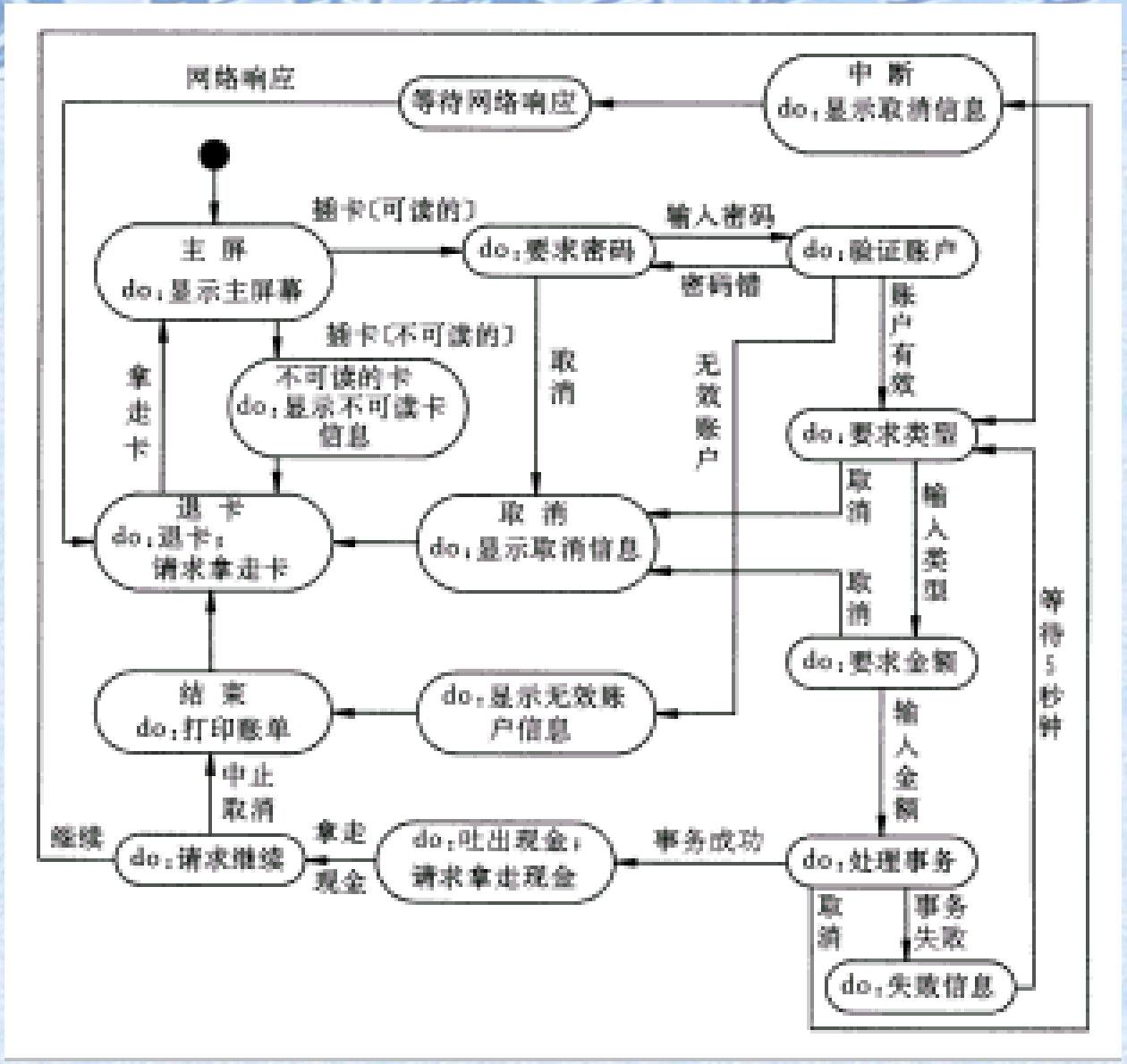
通常，用一张状态图描绘一类对象的行为，它确定了由事件序列引出的状态序列。但是，也不是任何一个类—对象都需要有一张状态图描绘它的行为。系统分析员应该集中精力仅考虑具有重要交互行为的那些类。

状态的性质

（1）时间性：状态与时间间隔有关，事件表示时刻，状态表示时间间隔，同一对象接收两个事件之间是一个状态。对象的状态依赖于接收的事件序列。

（2）持续性： 状态有持续性，它占有一个时间间隔， 状态常与连续的活动有关，状态需要时间才能完成的活动有关。

（3）事件与状态的关系：事件和状态是孪生的，一事件分开两种状态， 一个状态分开两个事件。


网络响应
等待网络响应
中 断
do：显示取消信息
插卡(可读的)
主 屏
do：显示主屏幕
do：要求密码
输入密码
do：验证账户
密码错
插卡(不可读的)
不可读的卡
do：显示不可读卡信息
拿走卡
取消
无效账户
账户有效
do：要求类型
退 卡
do：退卡；请求拿走卡
取 消
do：显示取消信息
取消
取消
输入类型
do：要求金额
结 束
do：打印账单
do：显示无效账户信息
输入金额
中止
取消
继续
do：请求继续
拿走现金
do：吐出现金；请求拿走现金
事务成功
do：处理事务
取消
事务失败
do：失败信息
等待5秒钟

ATM的状态图

5>审查动态模型

通过对动态模型的不断迭代和审查，使得动态模型得到完善

2.4 功能模型

为什么要功能建模？

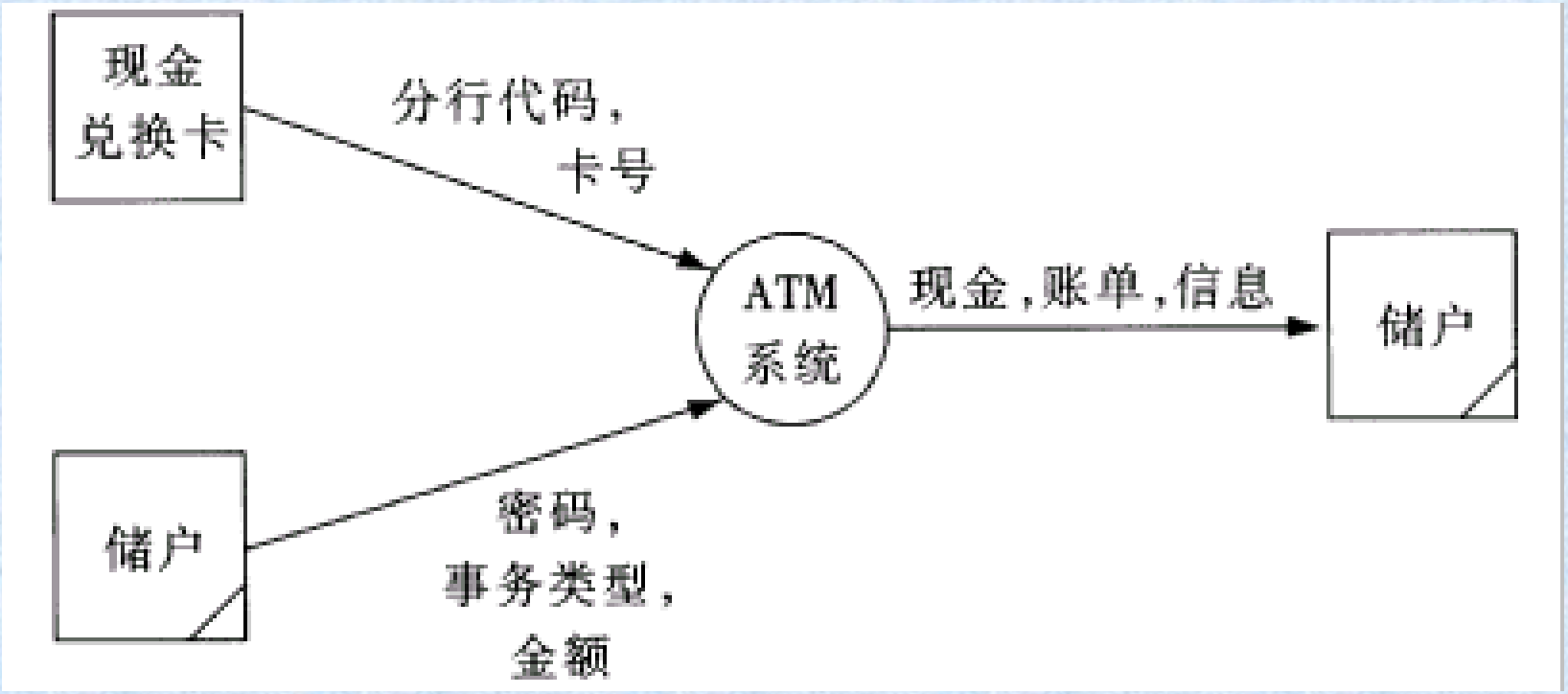
功能模型表明了系统中数据之间的依赖关系，以及有关的数据处理功能，它由一组数据流图组成。

通常在建立了对象模型和动态模型之后再建立功能模型。 根据从功能模型中获得的信息，重新审查静态模型和动态模型，以便进一步完善面向对象分析的结果。

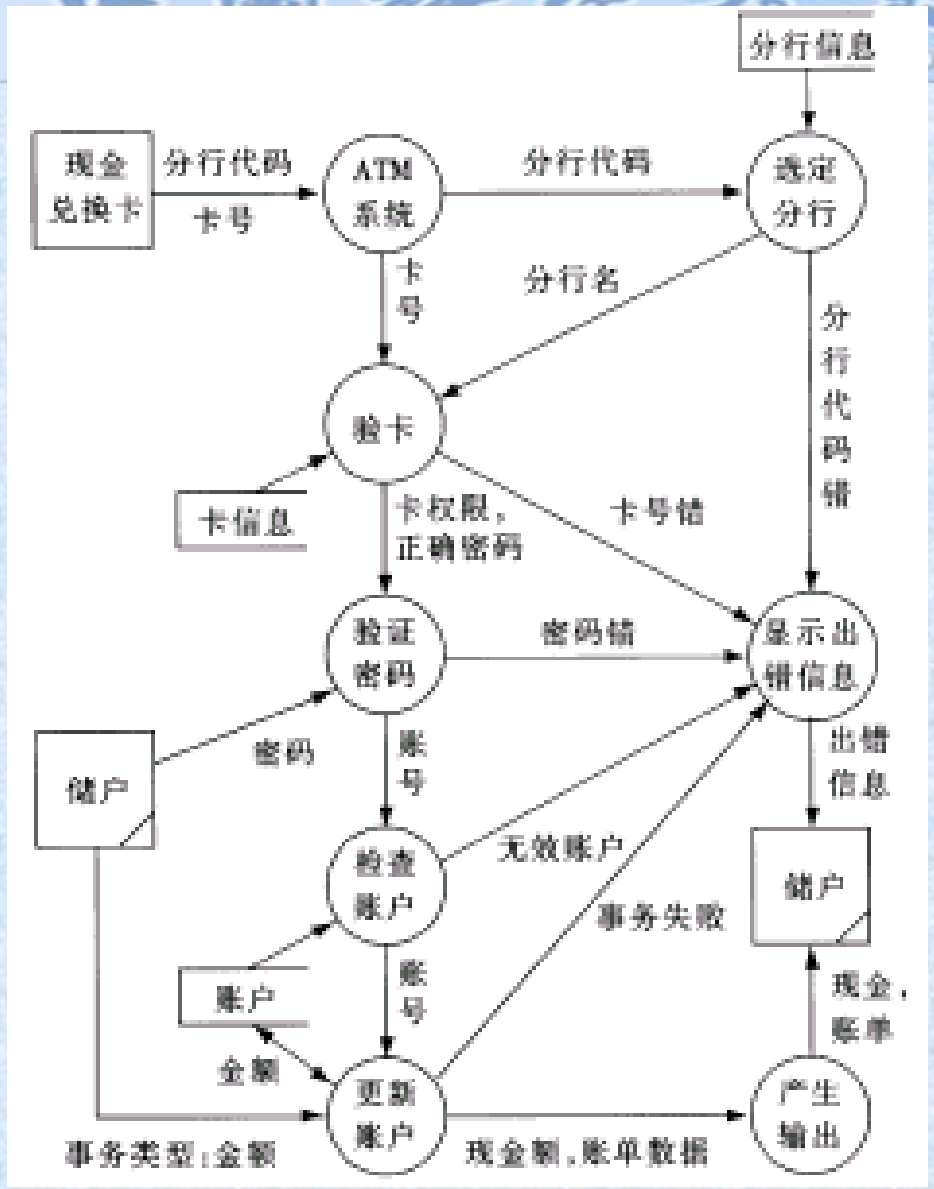
功能模型说明对象模型中操作的含义、动态模型中动作的意义以及对象模型中约束的意义。 一些不存在相互作用的系统，如编译器系统， 它们的动态模型较小，其目的是功能处理，功能模型是这类系统的主要模型。

功能模型由多个数据流图组成。与结构化分析中的DFD相似

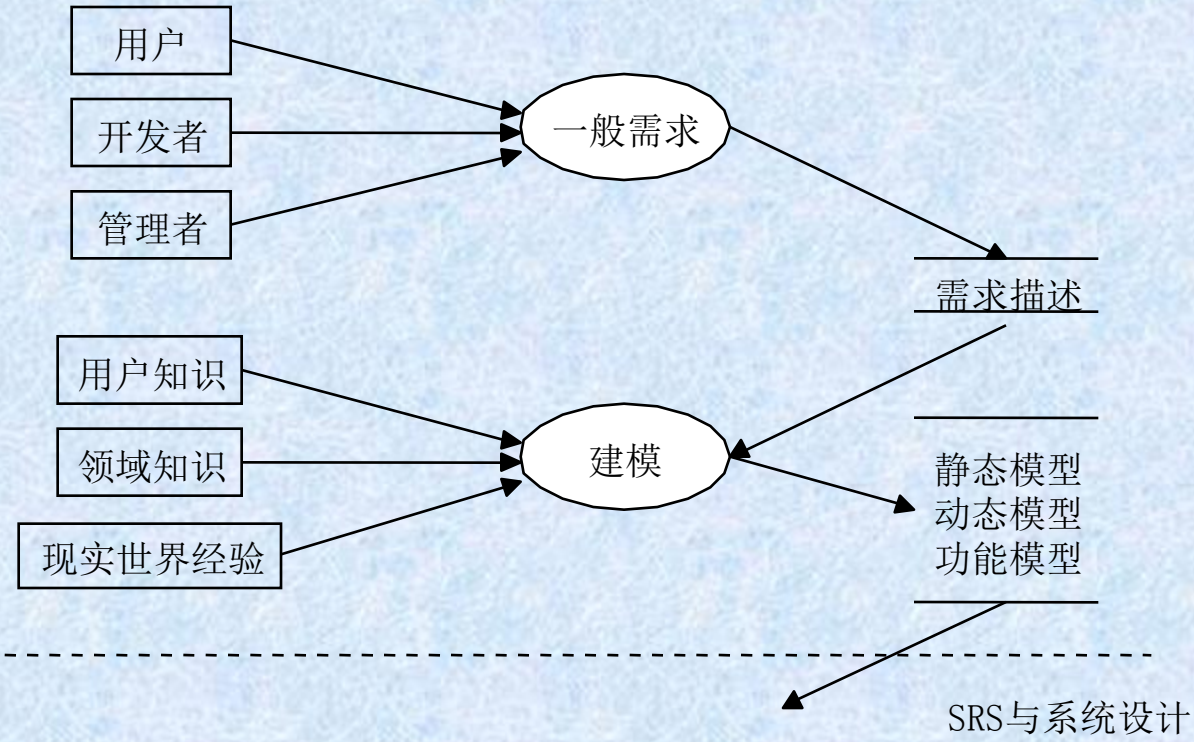
(1) 画出基本系统模型图
(2) 画出功能级数据流图。
(3) 描述处理框功能。



ATM系统的基本系统模



ATM系统的功能级数据流图


用户
开发者
管理者
一般需求
需求描述
用户知识
领域知识
现实世界经验
建模
静态模型
动态模型
功能模型
SRS与系统设计

三、两种方法在几方面的比较

从概念方面看

- 结构化软件是***功能***的集合，通过模块以及模块和模块之间的分层调用关系实现；
- 面向对象软件是***事物***的集合，通过对象以及对象和对象之间的通讯联系实现；

从构成方面看

- 结构化软件＝过程＋数据，以过程为中心；
- 面向对象软件＝（数据＋相应操作）的封装，以对象为中心；

从运行控制方面看

- 结构化软件采用顺序处理方式，由过程驱动控制；
- 面向对象软件采用交互式、并行处理方式，由消息驱动控制；

从开发方面看

- 结构化方法的工作重点是设计；
- 面向对象方法的工作重点是分析；

WHY?

在结构化方法中，分析阶段和设计阶段采用了不相吻合的表达方式，需要把在分析阶段采用的具有网络特征的数据流图转换进行转换。

从应用方面看

- 结构化方法更加适合数据类型比较简单的数值计算和数据统计管理软件的开发；

- 面向对象方法更加适合大型复杂的人机交互式软件和数据统计管理软件的开发；

四、几点注意：

1、面向对象分析方法的出现并不意味结构化分析方法的消亡。

2、分析阶段（分析模型）可能会和设计阶段（设计模型）间有重合

3、对于不同的分析人员，面向对象的分析模型的描述和构成可能会不同

4、分析模型建立需要多次反复的迭代和细化。

5、分析模型的质量直接关系到后续阶段的质量和最终产品的成败