

# 第10章 面向对象分析

## 目录

- 10.1 面向对象分析的基本过程
- 10.A UML简介
- 10.B 建立功能模型（用例模型）
- 10.3 建立对象模型
- 10.4 建立动态模型

# 10.1 面向对象分析的基本过程

- 获取需求
- 整理需求
- 建立模型
  - **功能模型**：指明系统应“做什么”，由用例图表示。
  - **对象模型**：描述静态结构, 定义做事情实体，类图和对象图表示。
  - **动态模型**：描述交互过程, 由状态图和顺序图表示。
- 书写需求规格说明书
- 复审

# 10.A UML简介

- UML : Unified Modeling Language 统一建模语言
- 90年代中期，出现了一批面向对象建模的图形化方法，最著名的是以下三人的方法：
  - Grady Booch , Jim Rumbaugh , Ivar Jacobson
- 此三人在Rational公司（ 2002年被IBM收购 ）工作时统一了方法，建立了UML。



The three amigos:

Grady Booch   Jim Rumbaugh   Ivar Jacobson

# 常用UML图

- 用例图：从用户角度描述系统功能。
- 类图：描述系统中类的静态结构。
- 对象图：系统中的多个对象在某一时刻的状态。
- 状态图：是描述状态到状态控制流，常用于动态特性建模
- 活动图：描述了业务实现用例的工作流程
- 顺序图：对象之间的动态合作关系，强调对象发送消息的顺序，同时显示对象之间的交互
- 部署图：定义系统中软硬件的物理体系结构
- 包图：对构成系统的模型元素进行分组整理的图

# 10.B 建立功能模型（用例模型）

- UML提供的**用例图**是进行需求分析和建立功能模型的有力工具。
- 以用例图建立起来的系统模型称为用例模型，它描述的是外部行为者所理解的系统功能。

## UML用例图

- 用例图的组成元素
  - 用例(use case)
  - 行为者(actor)
  - 行为者与用例之间的关系、用例与用例之间的关系

# 用例与行为者

- 用例(use case)
  - 行为者感受到的一个完整的功能。
  - 用例是类，代表一类功能，用例的实例称为脚本（scenario）。
- 行为者(actor)
  - 与系统交互的人或其他系统。它代表一种角色。

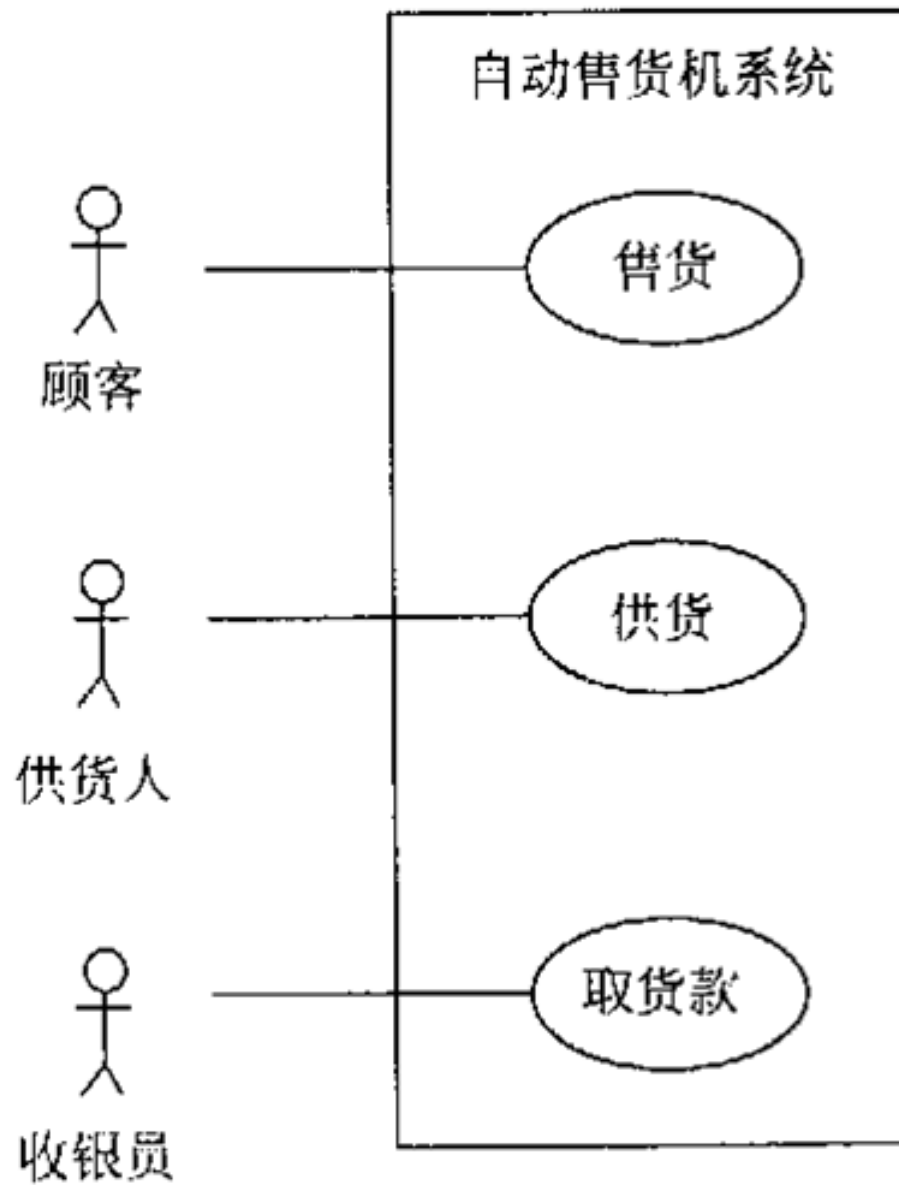


图 9.17 自动售货机系统用例图

# 用例之间的关系

- 扩展关系
  - 向一个用例中添加一些动作后构成另一个用例
  - 描述一般行为的异常情况
- 使用关系（包含关系）
  - 一个用例使用另一个用例
  - 两个或多个用例中出现重复描述时可采用使用关系

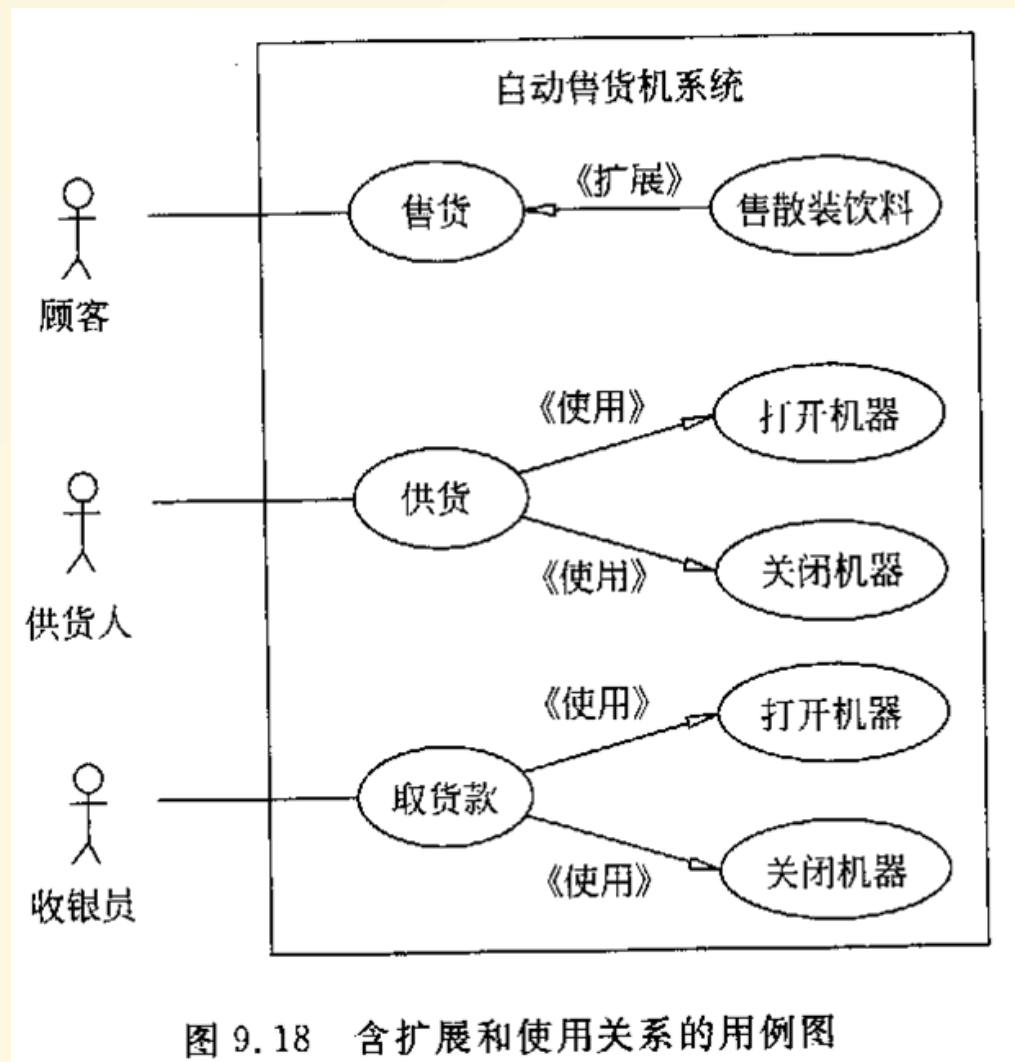


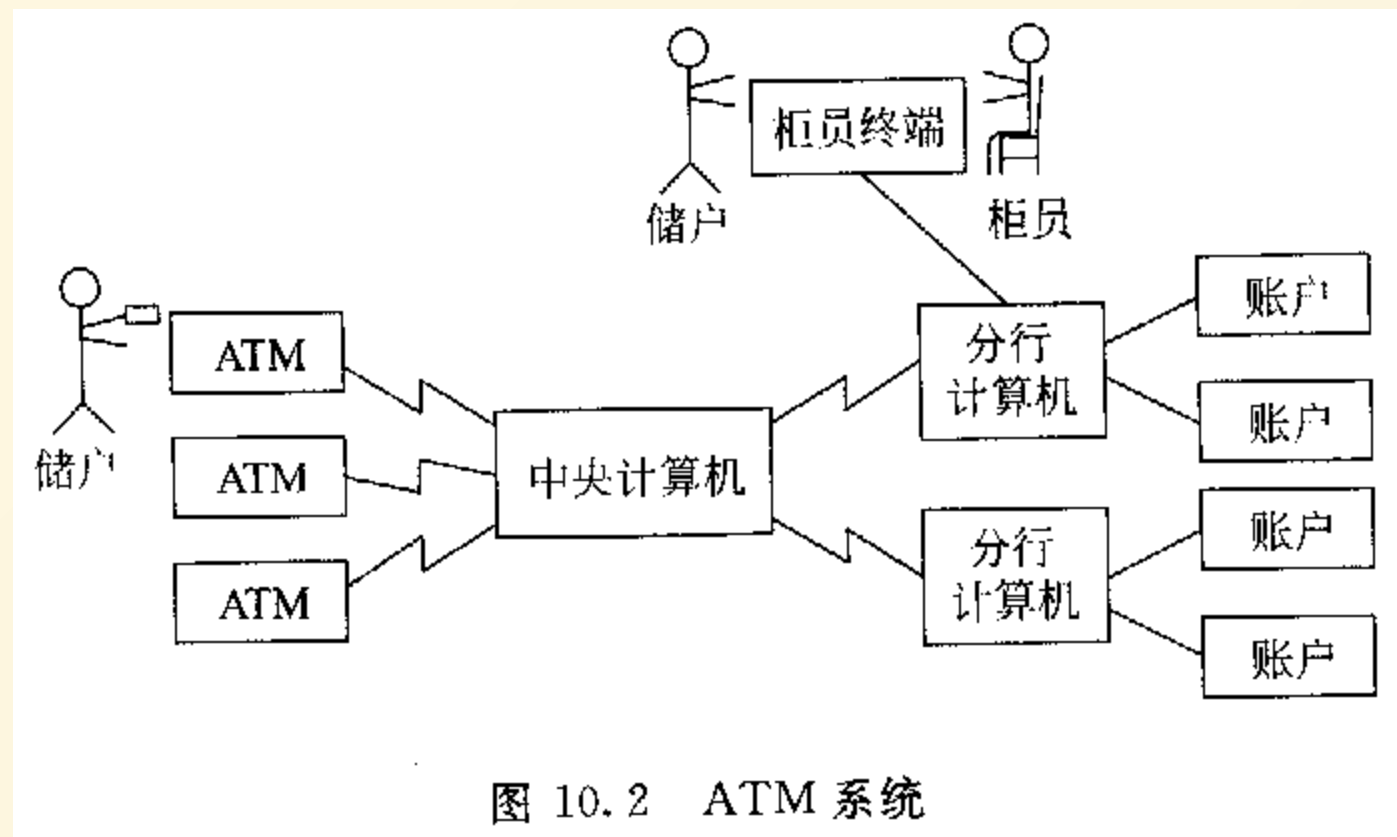
图 9.18 含扩展和使用关系的用例图

# 建立用例模型的步骤：

1. 识别外部行为者；
2. 识别用例；
3. 建立用例图；
4. 补充用例描述：为建立对象模型和动态模型打下基础。



# 举例：自动取款机（ATM）系统

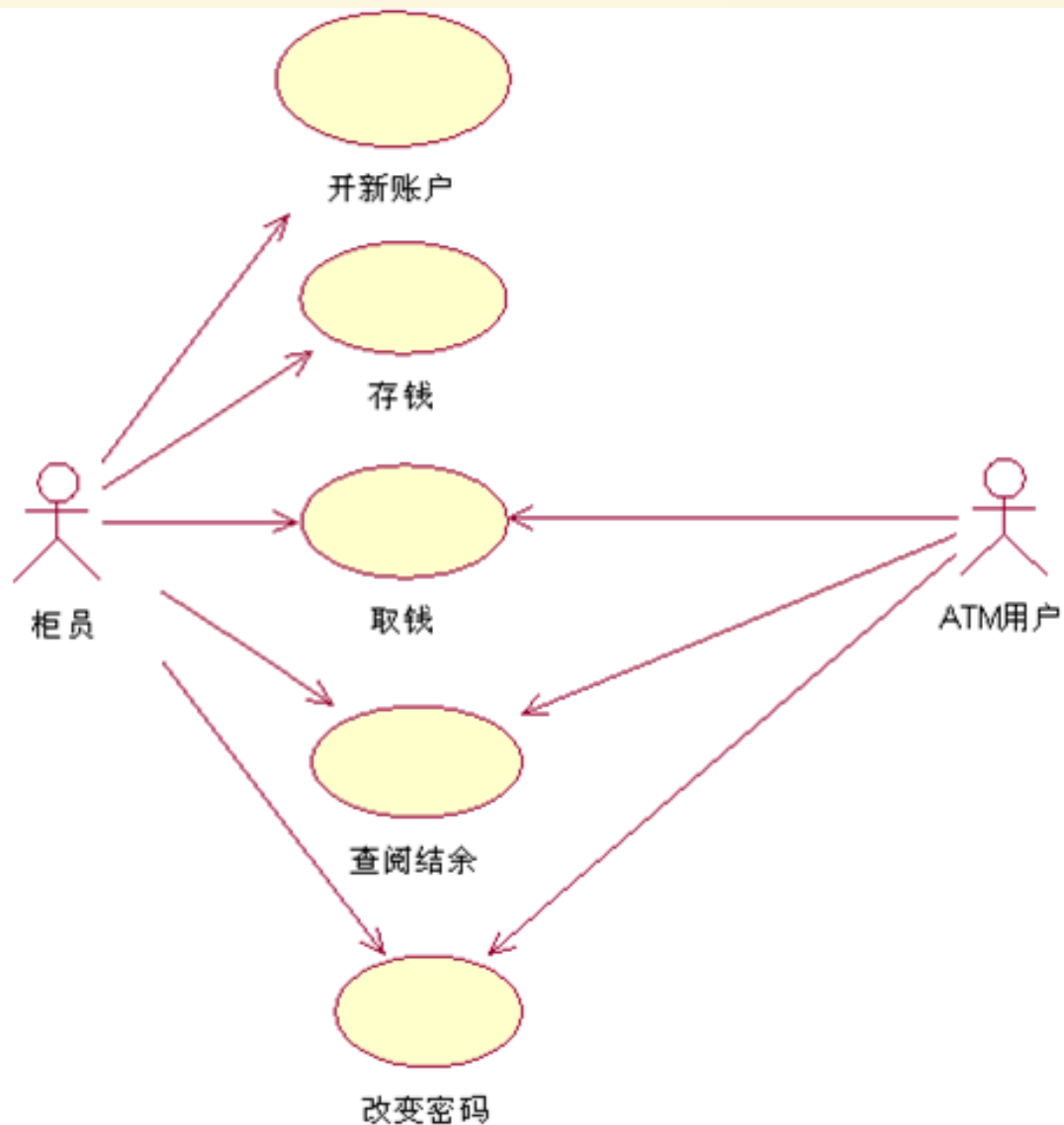


需求陈述见教材P234-235

# ATM系统用例图

注：

1. 本系统虽名为“ATM系统”，实际范围超过ATM机。
2. 功能模型=用例图 + 每一个用例的描述。



# 用例描述举例：存钱

用例	存钱
行为者	柜员
前提	用户账户已存在。
目标	用户账户余额得到更新。
正常流程	<ol style="list-style-type: none"><li>1. 柜员输入账户信息。</li><li>2. 分行计算机对账户信息进行验证并返回验证结果。</li><li>3. 柜员输入存入的金额数目。</li><li>4. 分行计算机更新账户余额，并返回更新结果。</li></ol>
异常情况	<ol style="list-style-type: none"><li>2a. 分行计算机返回账户信息验证错误。</li></ol>

## 10.3 建立对象模型

- 对象模型描述类及相互关系，表达目标系统的静态结构。
- 一般用UML中的**类图**表示。

### UML类图

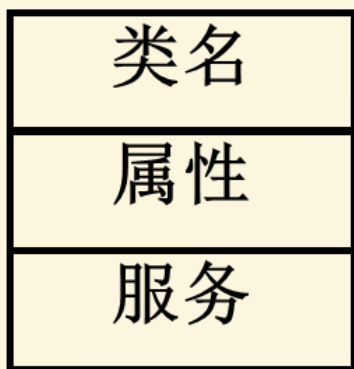


图9.5 表示类的图形符号

# 类的定义

- 类名
- 属性：`可见性 属性名：类型名 = 初值 {性质串}`
  - 可见性：公有（public）/+、私有（private）/-、保护（protected）/#
  - {性质串}：可能枚举值或其它性质，如{只读}
- 服务（操作）：`可见性 操作名（参数表）：返回值类型 {性质串}`
  - 参数的语法：`参数名：类型名 = 默认值`

Student
- student id : String - name : String - gender : int - date of birth : Date
+ takeClass() : void + doHomework() : void + haveRest() : void

# 类与类之间的关系

- 关联
- 聚集
  - 共享聚集
  - 组合聚集
- 泛化（继承）

# 关联



图9.6 普通关联示例

重数（multiplicity）的表示方法：

0..1 表示 0到1个对象；

0..\* 或\* 表示 0到多个对象；

1+ 或1..\* 表示 1到多个对象；

1..15 表示 1到15个对象；

3 表示 3个对象。

# 关联的角色

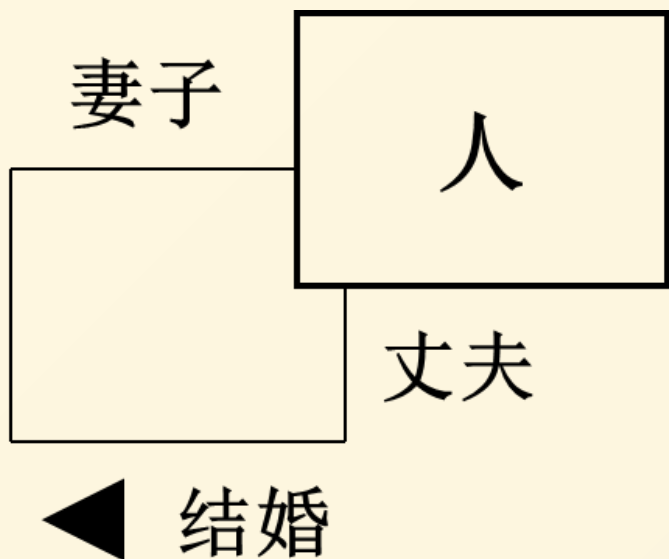


图9.7 关联的角色

上图是一个递归关联的例子。这种情况下，标明角色名有助于理解类图。



# 限定关联

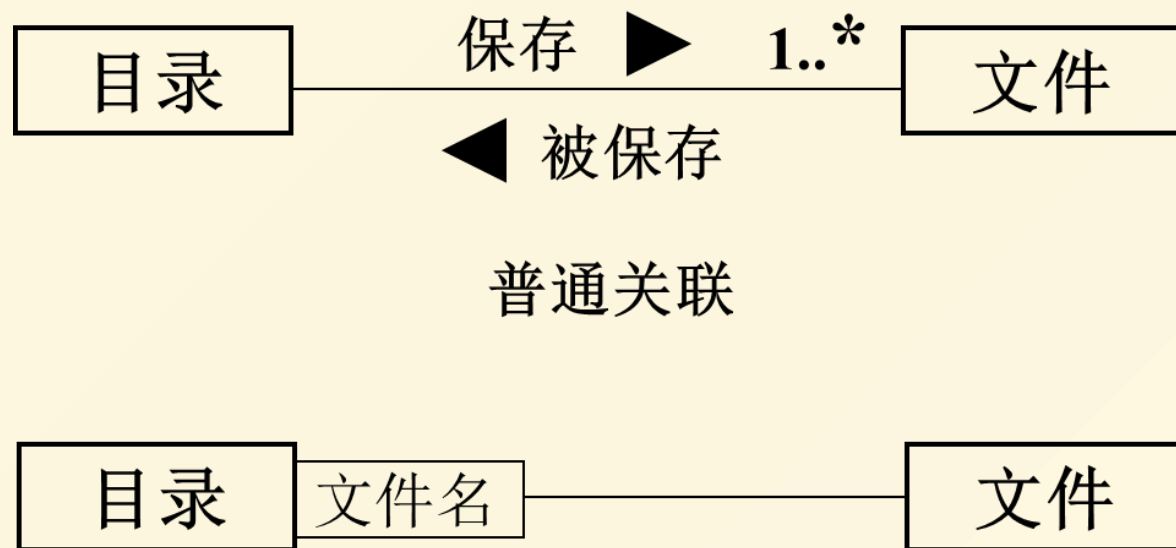


图9.8 限定关联

利用限定词把一对多关系简化成了一对一关系。

# 关联类

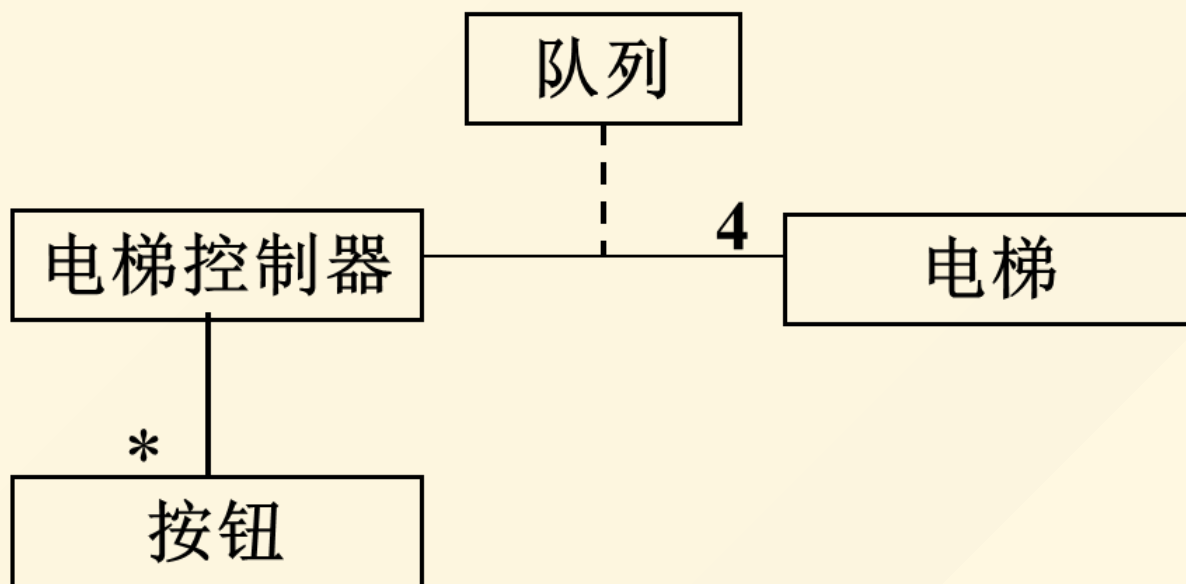


图9.9 关联类示例

控制器对象和电梯对象之间的连接，对应着一个队列（对象），它存储着控制器和电梯内部按钮的请求信息。

# 共享聚集

聚集表示类与类之间是整体与部分的关系。

共享聚集：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。

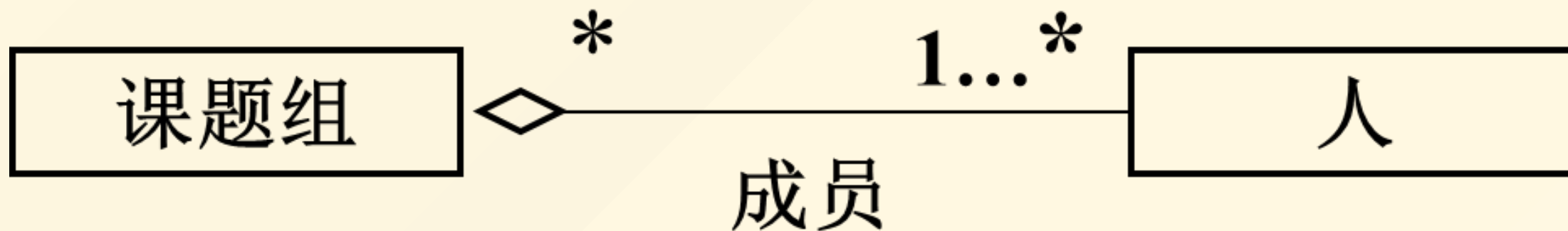


图9.10 共享聚集示例

# 组合聚集

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随之消失。

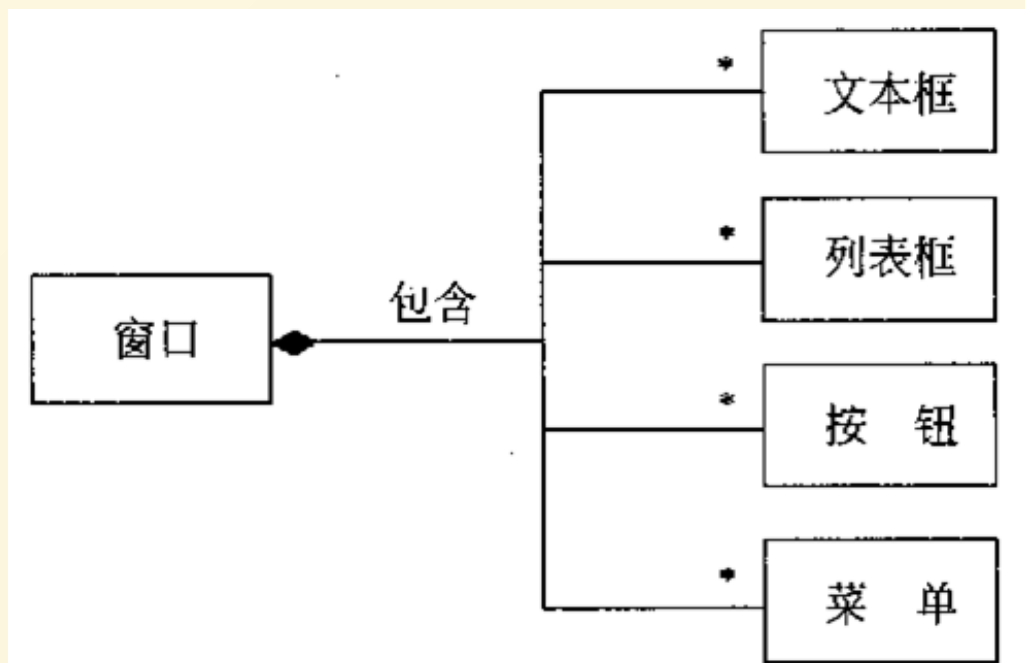
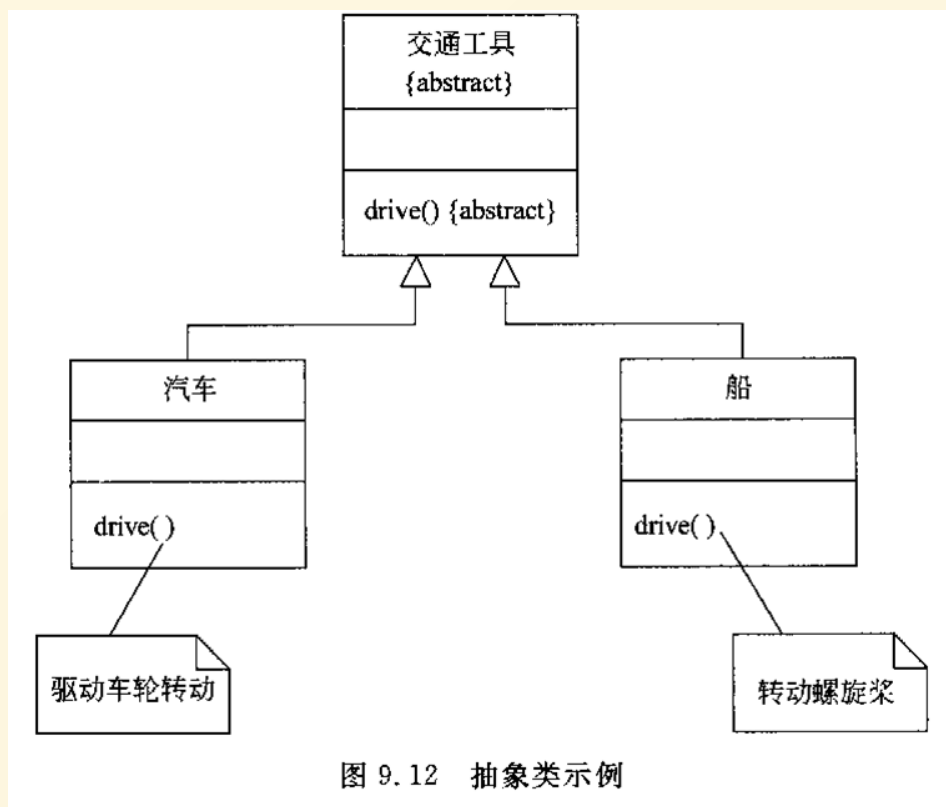


图 9.11 组合聚集示例

# 泛化（继承）



抽象类：描述子类的公共属性和行为，不能生成具体对象的类。

# 类图举例

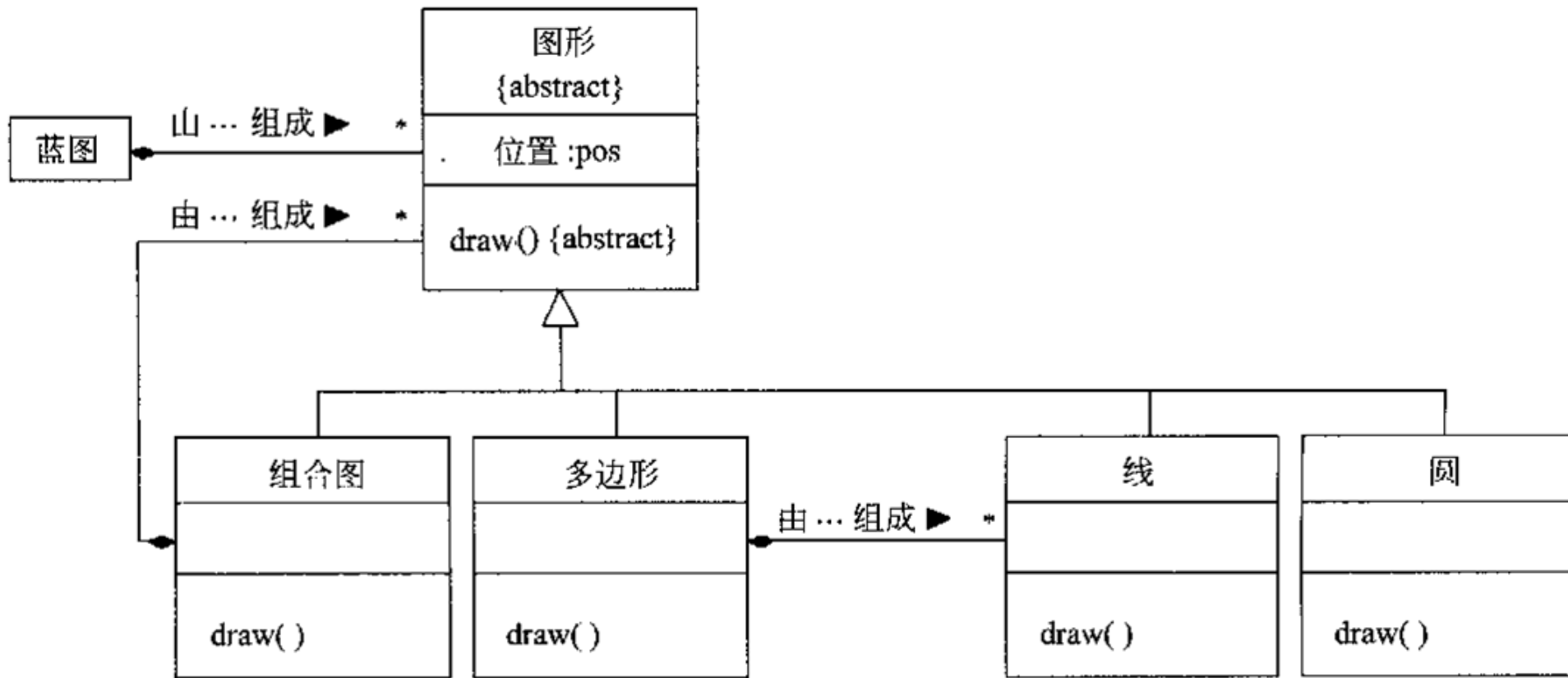


图 9.13 复杂类图示例

# 建立对象模型的步骤

1. 确定类与对象；
2. 确定类的关联；
3. 划分主题；
4. 确定属性；
5. 识别继承；
6. 反复修改。

# 1. 确定类与对象

## 找出候选的类与对象

- 1 ) 可感知的物理实体，如汽车、书、房屋；
- 2 ) 人或组织的角色，如雇员、雇主、柜员；
- 3 ) 应该记忆的事件，如演出、访问、事故；
- 4 ) 两个或多个对象的相互作用，如购买；
- 5 ) 需要说明的概念，如政策、法律。

非正式分析方法：用自然语言书写需求陈述，把陈述中的名词作为类与对象的候选者，从形容词中考虑属性，把动词作为服务（操作）的候选者。



## ATM系统：类与对象的候选者

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、帐户、事务、现金兑换卡、余额、磁卡、分行代码、卡号、用户、信息、密码、类型、取款额、帐单、访问等。

# 筛选出正确的类与对象

主要依据以下标准：

- 1 ) 冗余：如“储户”与“用户”、“磁卡”与“现金兑换卡”；
- 2 ) 无关：如“成本”、“街道”、“营业厅”、“储蓄所”；
- 3 ) 笼统：如“银行”、“网络”、“系统”、“软件”、“信息”；
- 4 ) 属性：如“余额”、“分行代码”、“卡号”、“密码”、“类型”；
- 5 ) 操作：如“访问”。

## ATM系统：筛选后的类与对象

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

## 2. 确定关联

### 初步确定关联

- 1 ) 直接提取动词短语得出的关联

总行 拥有 ATM

储户 拥有 帐户

分行计算机 维护 帐户

.....

- 2 ) 需求陈述中隐含的关联

分行 组成 总行

分行 保管 帐户

.....

- 3 ) 根据问题域知识得出的关联

现金兑换卡 访问 帐户

分行 雇用 柜员

.....

# 筛选

根据下述标准删除候选关联：

- 1 ) 已删去的类之间的关联。如“系统”、“成本”等类已经删除，所以也应该删除的关联：  
系统 维护 事务日志  
分行 分摊 软件成本
- 2 ) 与问题无关的或应在实现阶段考虑的关联。如应删除的关联：  
系统 处理 并发访问
- 3 ) 瞬时事件。关联不应是一个瞬时事件，如应该删除：  
ATM 读取 现金兑换卡
- 4 ) 三元关联。三个或三个以上对象之间的关联，可分解为二元关联：  
“柜员输入针对帐户的事务”可分解为：  
“柜员 输入 事务”、“事务 修改 帐户”
- 5 ) 派生关联。去掉可以用其它关联定义的冗余关联，如“分行计算机 维护 帐户”，可用“分行 保管 帐户”、“事务 修改 帐户”代替。

# 进一步完善

- 1 ) 正名  
选择含义更明确的名字。
- 2 ) 分解  
必要时分解类与对象，如“事务”分解为：  
“远程事务”、“柜员事务”。
- 3 ) 补充  
发现遗漏的关联及时补上。
- 4 ) 标明重数

# ATM系统原始类图

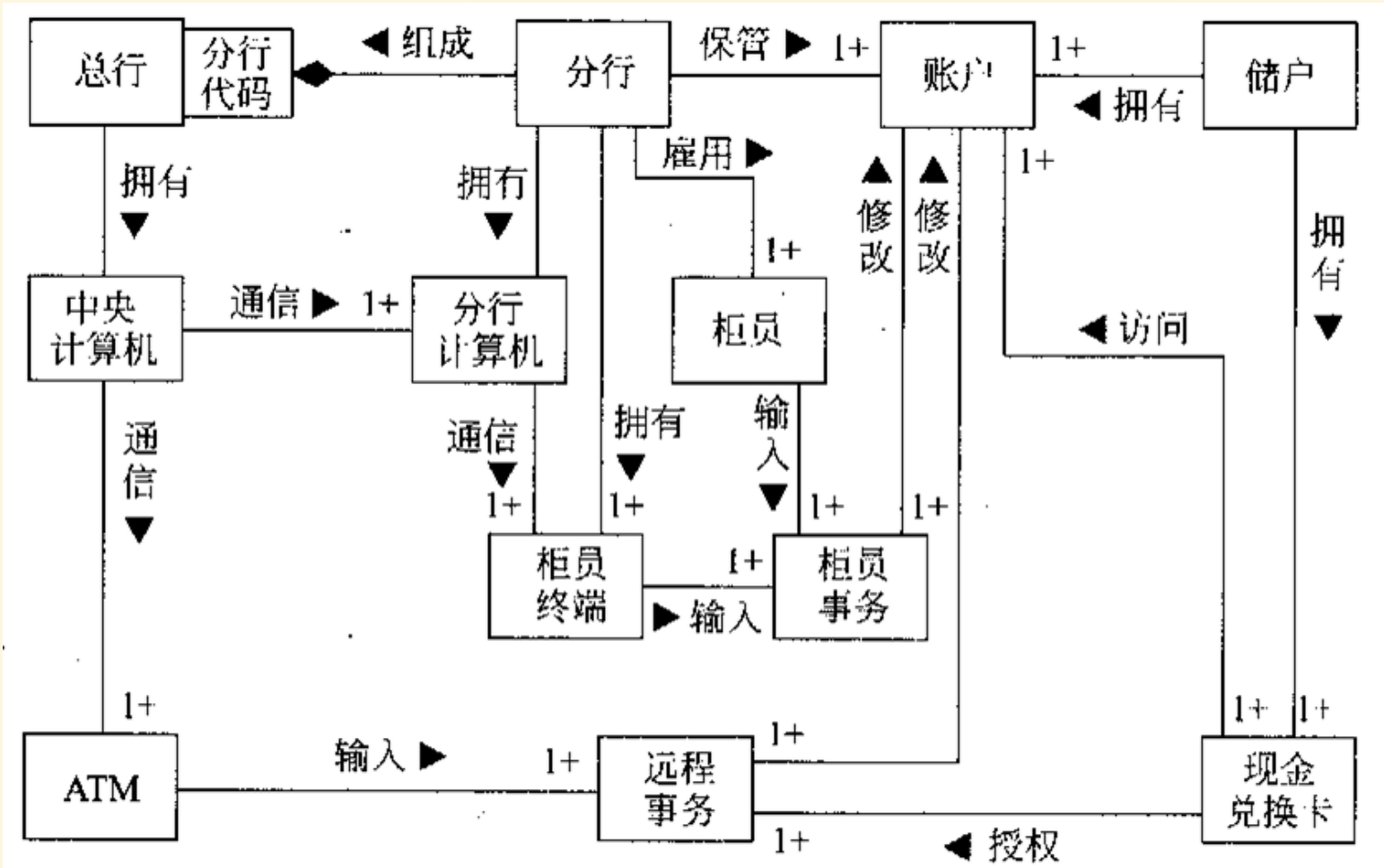


图 10.3 ATM 系统原始类图

### 3. 划分主题

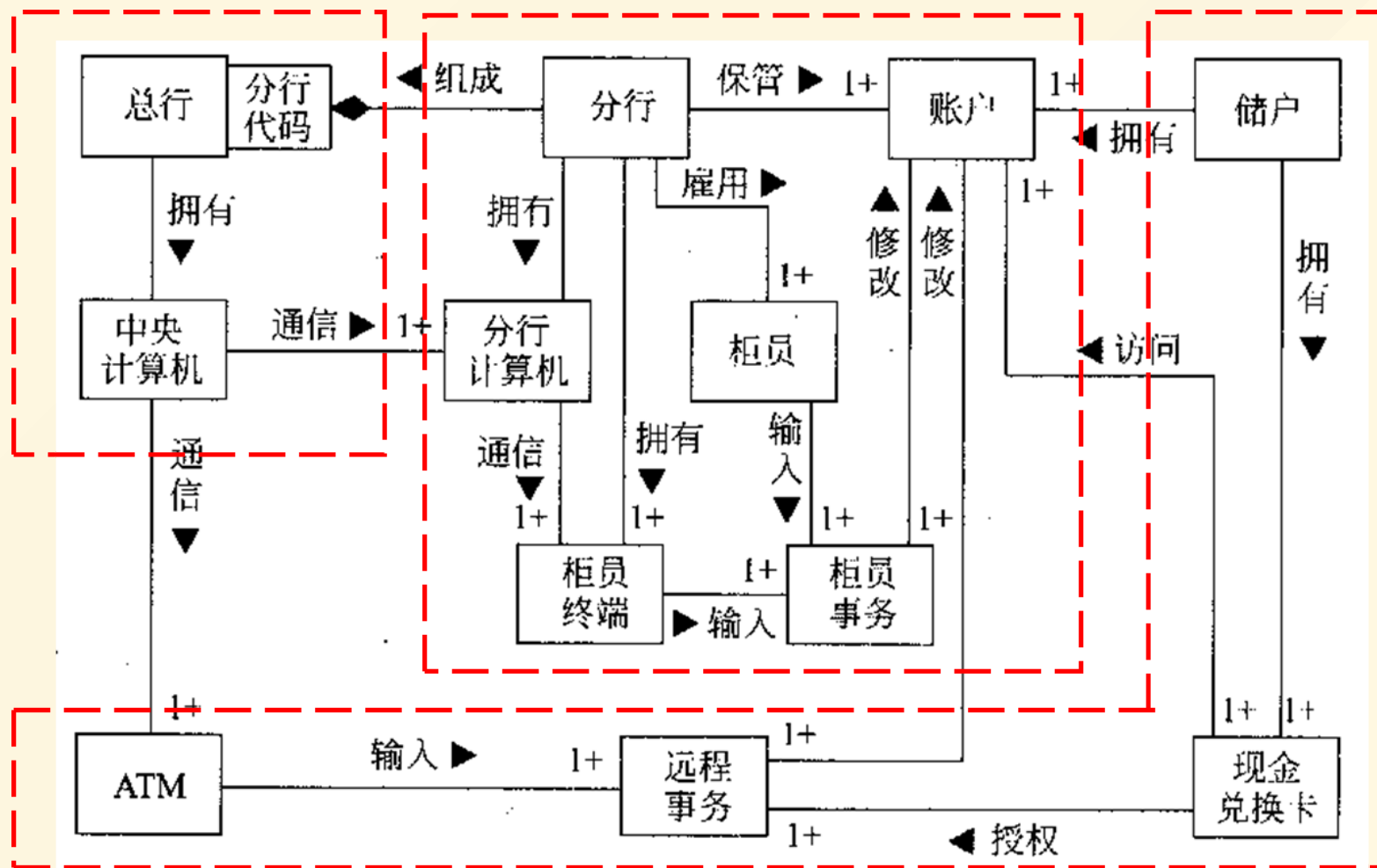


图 10.3 ATM 系统原始类图

## 4. 确定属性

- 分析

需求陈述不能得到所有属性，须借助领域知识和常识。

- 选择

根据下述情况，删除不必要的属性：

- 1 ) 误把对象当属性
- 2 ) 误把关联类的属性当作一般对象的属性
- 3 ) 误把限定当成属性（如“站号”不是“分行计算机”的属性）
- 4 ) 误把内部状态当属性



# ATM系统中的对象的属性

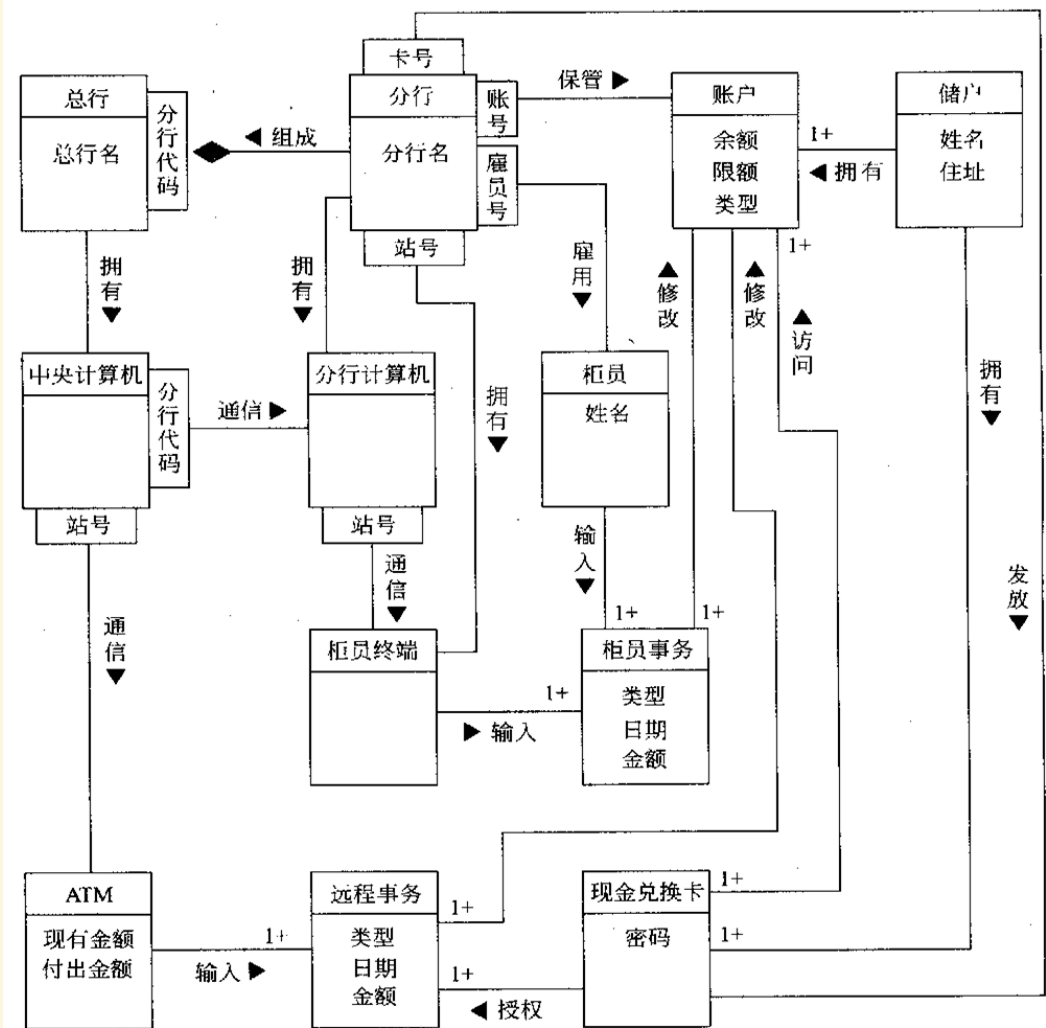


图 10.4 ATM 系统对象模型中的属性

## 5. 识别继承

可以使用两种方式建立继承（泛化）关系：

- 自底向上：从现有类泛化出父类；  
如：“远程事务”和“柜员事务”泛化出“事务”；  
“ATM”和“柜员终端”泛化出“输入终端”。
- 自顶向下：把现有类细化成子类。

# 带有继承关系的ATM对象模型

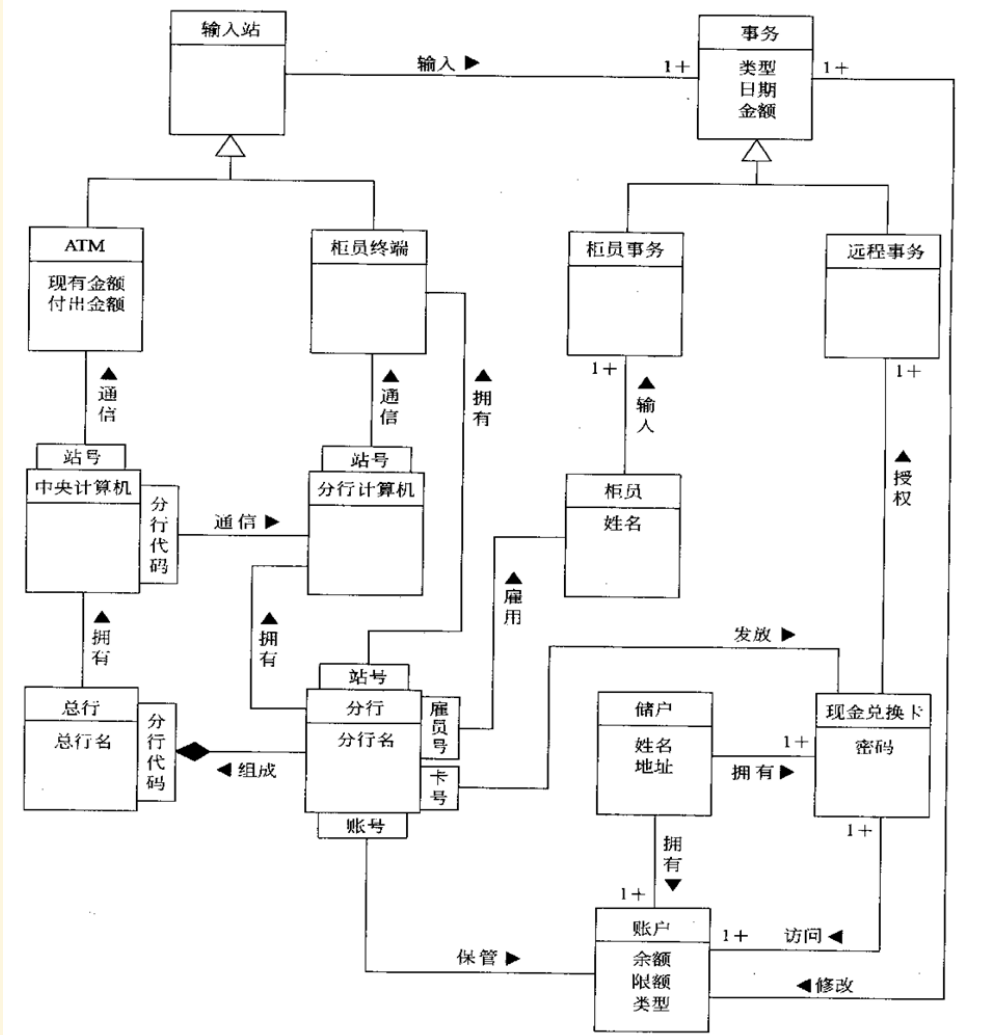


图 10.5 带有继承关系的 ATM 对象模型

## 6. 反复修改

### 1. 分解类

如：分解“现金兑换卡”类为“卡权限”和“现金兑换卡”，使每个类功能更单一；

### 2. 合理组织类

如：“事务”由“更新”组成，“更新”包括“取款”、“存款”、“查询”等动作。

### 3. 合并类

如：合并“分行”与“分行计算机”、“总行”与“中央计算机”。

# 修改后的ATM系统对象模型

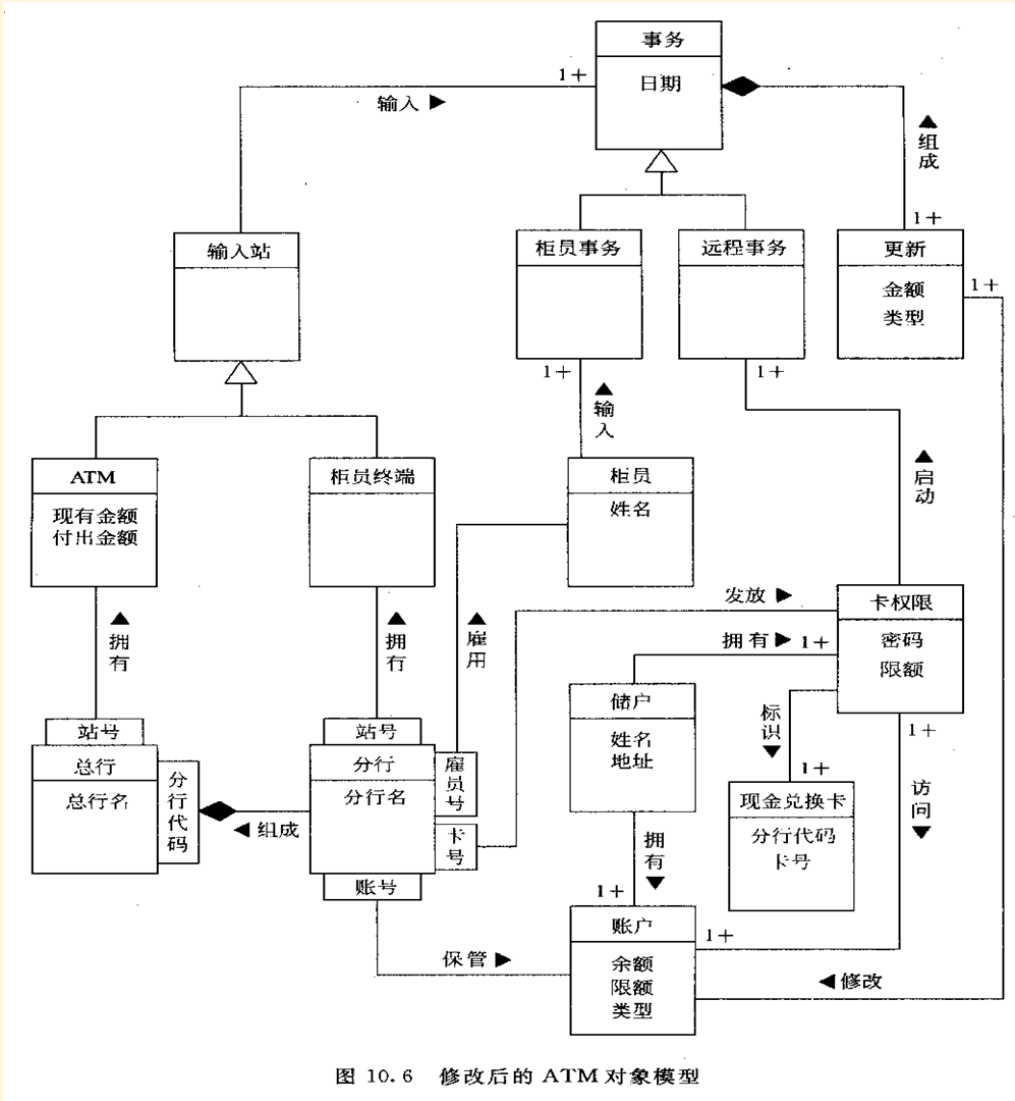


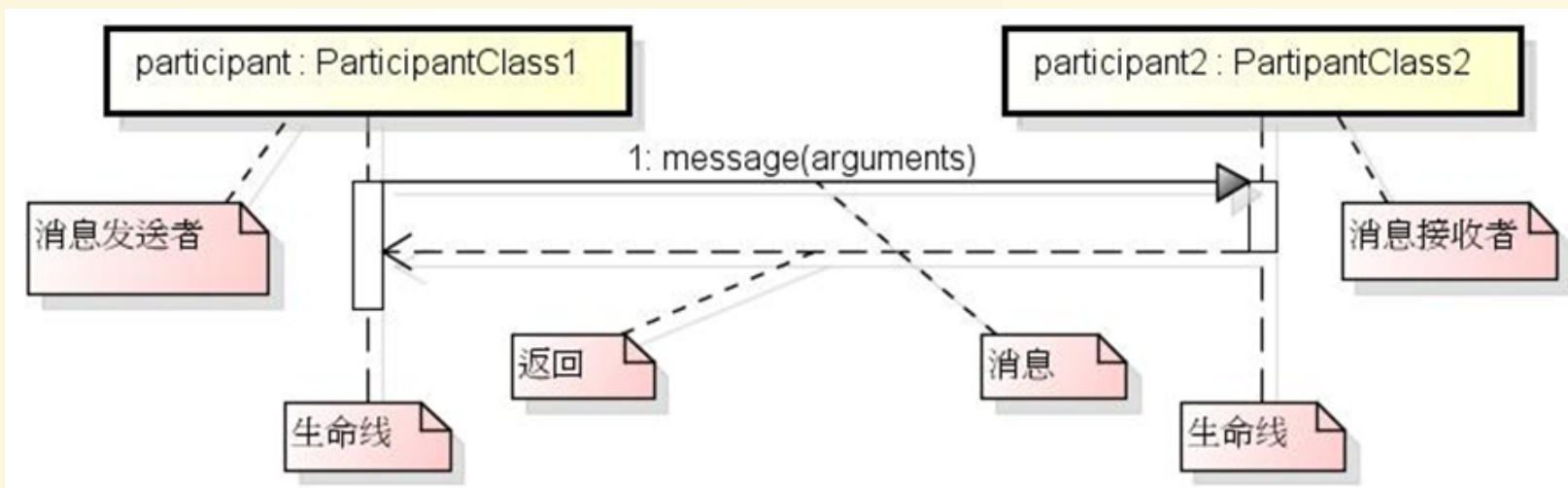
图 10.6 修改后的 ATM 对象模型

## 10.4 建立动态模型

- 从脚本（或用例）中提取所有外部事件，确定每类事件发送和接收对象。  
针对系统中的典型功能，画出顺序图。
- 用一张状态图描绘类的行为，集中考虑具有交互行为类。

# UML顺序图

- 顺序图描述系统各个组成部分之间的有序交互
  - 参与者 ( participant )
  - 生命线 ( lifeline )
  - 消息 ( message )

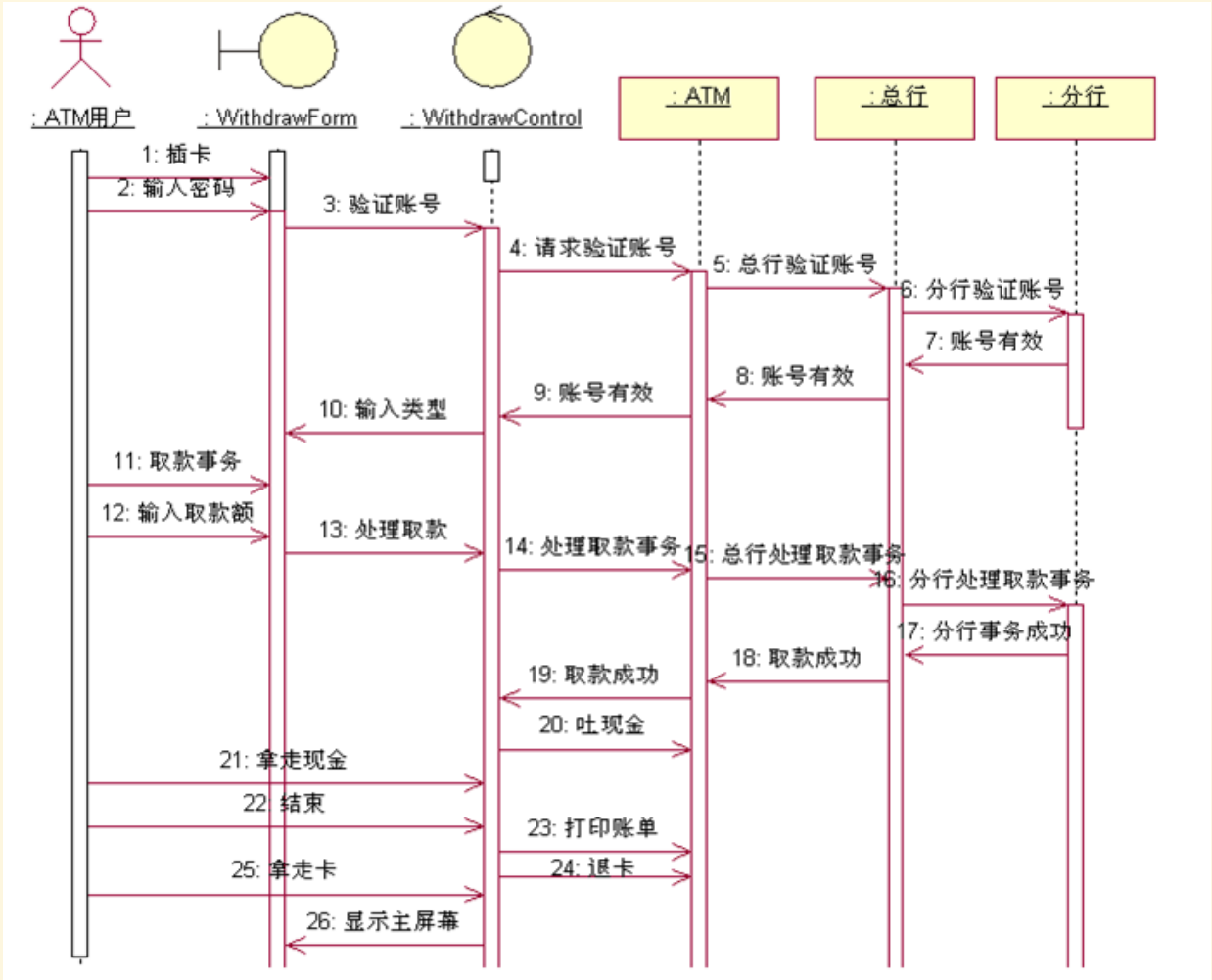


# 顺序图中的消息

- 消息的具体形式
  - 函数调用（最常见）
  - 信号的发送
  - 对象的创建和删除
- 消息的类型
  - 同步消息（synchronous message）：消息发送者等到返回消息之后才能继续工作
  - 异步消息（asynchronous message）：消息发送者发送消息后，不作等待，即继续后面的工作
  - 返回消息（return message）：表示活动的控制流返回给原始消息的发送者
    - 返回消息的标注可以省略
    - 每一个同步消息都隐含一个后续的返回消息

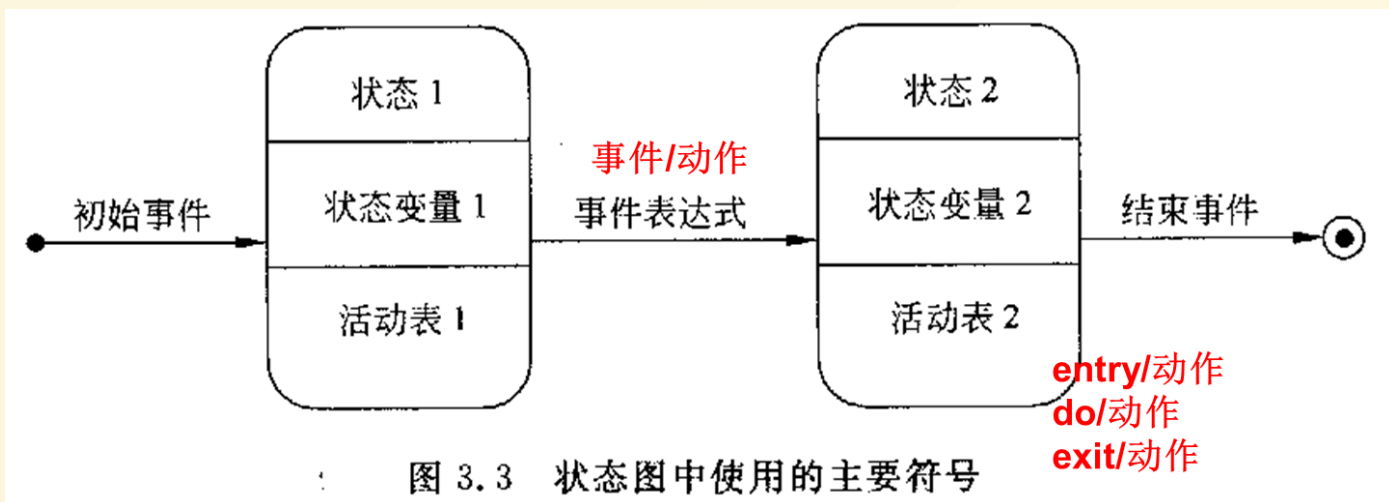


# ATM用户取款顺序图

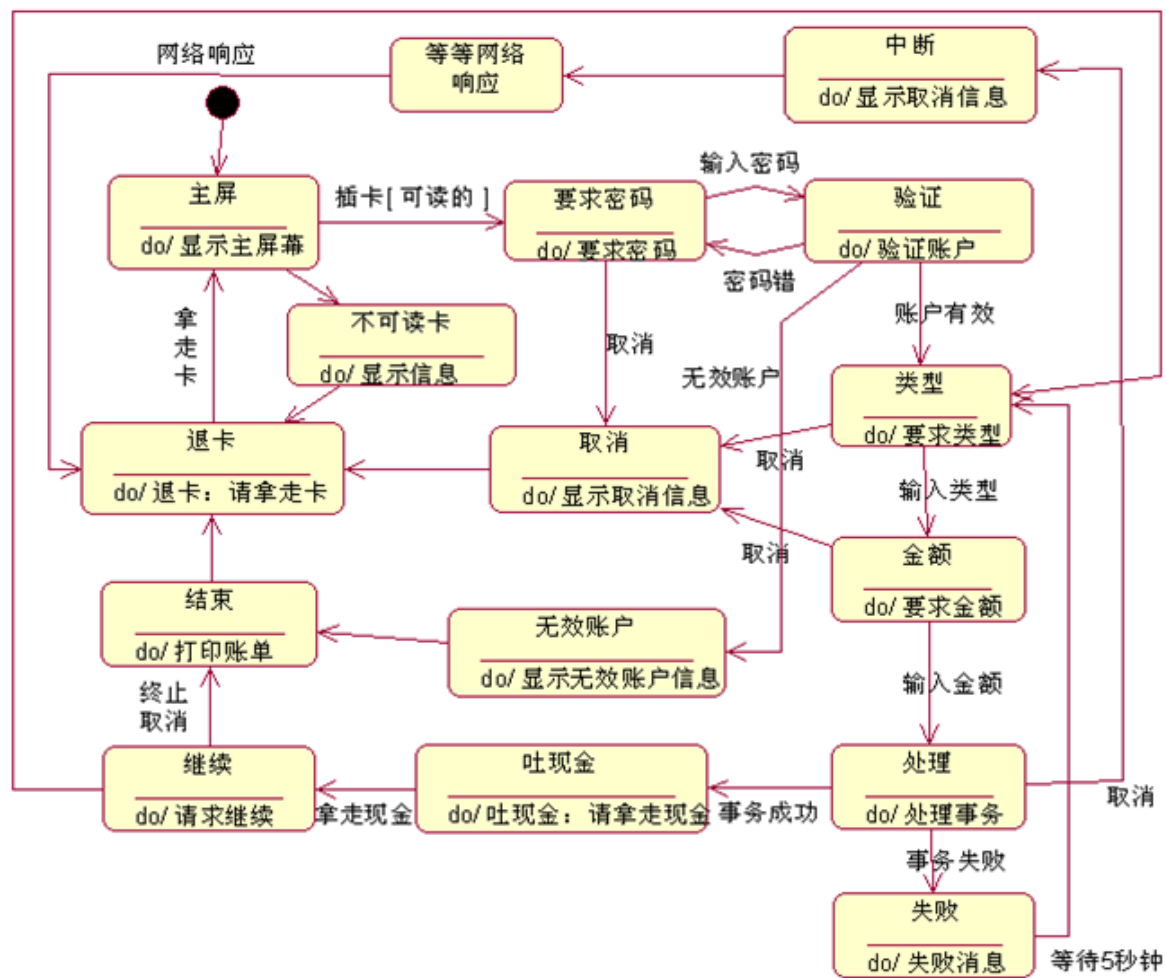


# UML状态图

- 事件：事件是某个特定时刻所发生的事情。它是引起对象状态转换的控制信息。
- 状态：状态就是对象在其生命周期中的某个特定阶段所处的某种情形。
- 行为：行为是指对象达到某种状态时所做的一系列处理操作。



# ATM类状态图



# 本章重点

- UML用例图、类图、顺序图、状态图
- 建立功能模型、对象模型和动态模型的方法