CHAPTER 3

Software Requirement Analysis

UML 小结

- UML 是一种可视化的图形符号建模语言，利用它可以进行需求分析。

- UML 有9个图

1 use case diagram,	用例图
2 class diagram,	类图
3 object diagram,	对象图
4 sequence diagram,	顺序图
5 state diagram,	状态图
6 activity diagram,	活动图
7 collaboration diagram,	协作图
8 component diagram,	构件图
9 deployment diagram,	部署图

- UML 主要建立4个图模型

UML分析模型

- 功能模型: 描述处理(数据变换), 指明系统应“做什么”

  use case diagram

- 对象模型: 描述静态结构, 定义做事情的实体

  class & object diagram

- 动态模型: 描述交互过程, 规定什么时候做何事

  sequence, state, collaboration, activity diagram

Object Oriented Analysis 本质

3 个模型

- ✓functional model
- ✓static model
- ✓dynamical model

✓5 个层面:

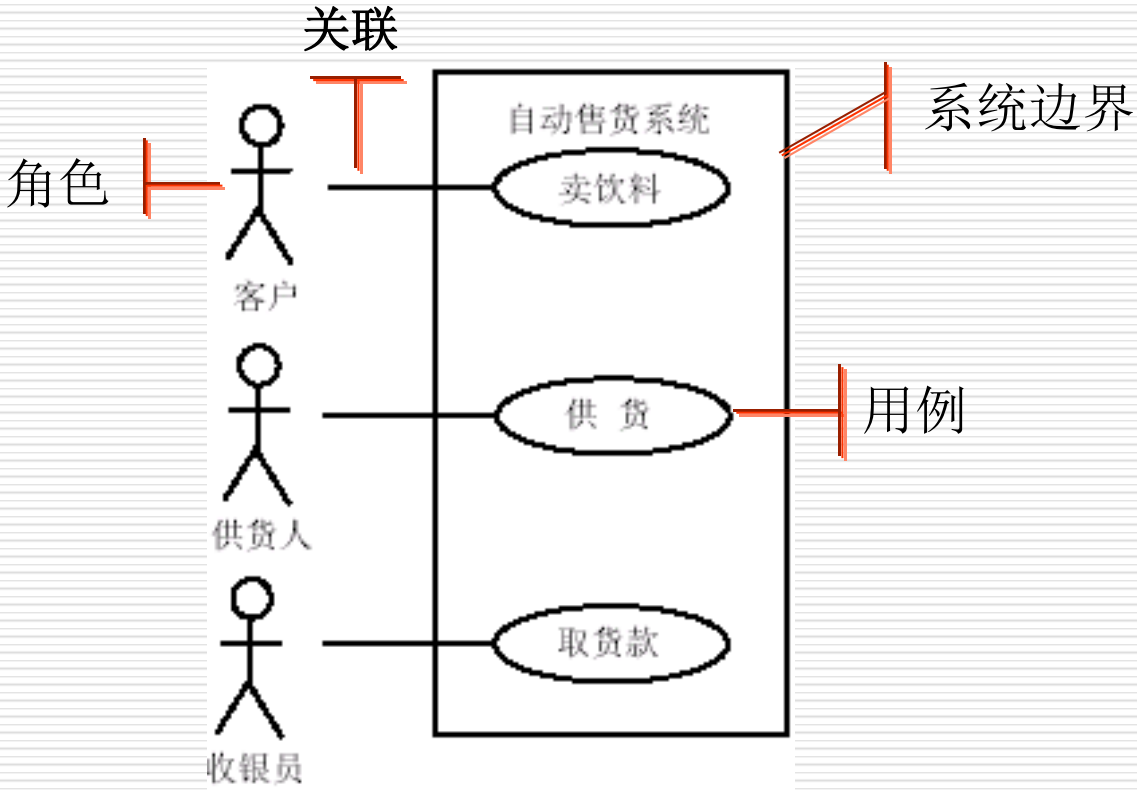
Subject
class & object
structure
attributes
services

OOA, UML, 功能模型

典型用例建模方法：（use case diagram）

- 用例建模，包括四个部分：
  - ✓ 系统边界：包围用例的方框，表示系统的范围，边界内的用例表示系统将来要实现的功能。
  - ✓ 参与者：在系统边界之外，透过系统边界与系统进行有意义的交互的任何人或事物。
  - ✓ 用例：由系统执行的一个动作序列，给角色(actor)提供一项有价值的服务。
  - ✓ 关系：角色和用例、角色之间、用例之间有意义的联系。

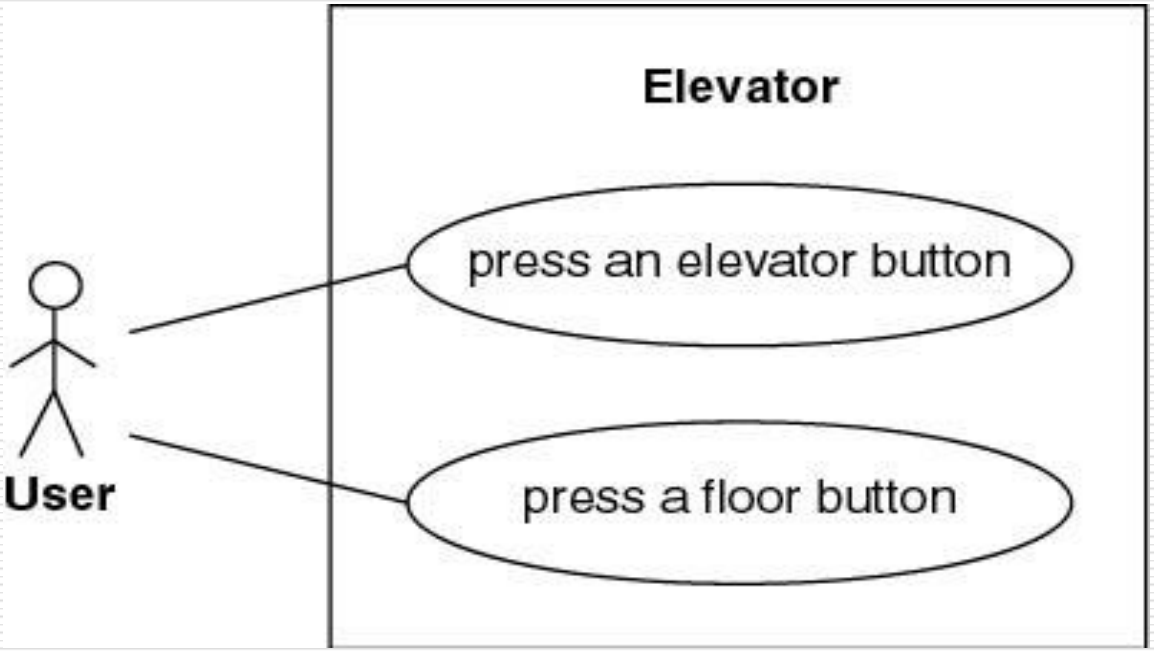
用例建模



用例模型：从和系统有交互的实体（外部用户）角度出发，描述系统应该具备哪些功能。

黑盒功能建模

例子1： 电梯


Elevator
press an elevator button
press a floor button
User

如何识别角色(Actor)

- 识别角色必须从系统本身的问题域出发
  - 谁或什么使用该系统？
  - 交互中，它们扮演什么角色？
  - 谁安装系统？
  - 谁启动和关闭系统？
  - 谁维护系统？
  - 与该系统交互的是其他什么系统？
  - 谁从该系统获取信息，谁提供信息给系统？
  - 有什么事情发生在固定时间？

例：ATM系统的Actor

1、谁使用ATM系统的主要功能（提款）？

答：储户

2、谁使用ATM系统的支持以完成日常工作任务？

答：出纳员？还不肯定，先放在这里

3、谁来维护、管理并保持系统正常运行？

答： ATM系统工程师

4、该系统需要和哪些系统交互？

答：目前还不清楚

5、ATM系统需要处理哪些设备？

答：读卡机、吐钱设备

6、谁对ATM系统运行的结果感兴趣？

答：银行会计、储户

例：ATM系统的Actor

储户

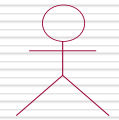
银行出纳

银行会计

系统工程师

读卡机

吐钱设备



如何提取出用例

- 发现用户的目标
- 系统必须做什么（功能需求），而非如何做（设计）
- 系统描述为具有某种职责
- 系统外部行为

如何提取出用例

- 针对已识别角色：

1、某个角色要求系统为其提供什么功能？该角色需要做哪些工作(可能有些工作需要系统帮助完成）？

2、角色需要阅读、创建、销毁、更新或存储系统中的某些（类）信息吗？

3、系统中的事件一定要告知角色吗？角色需要告诉系统一些什么吗？

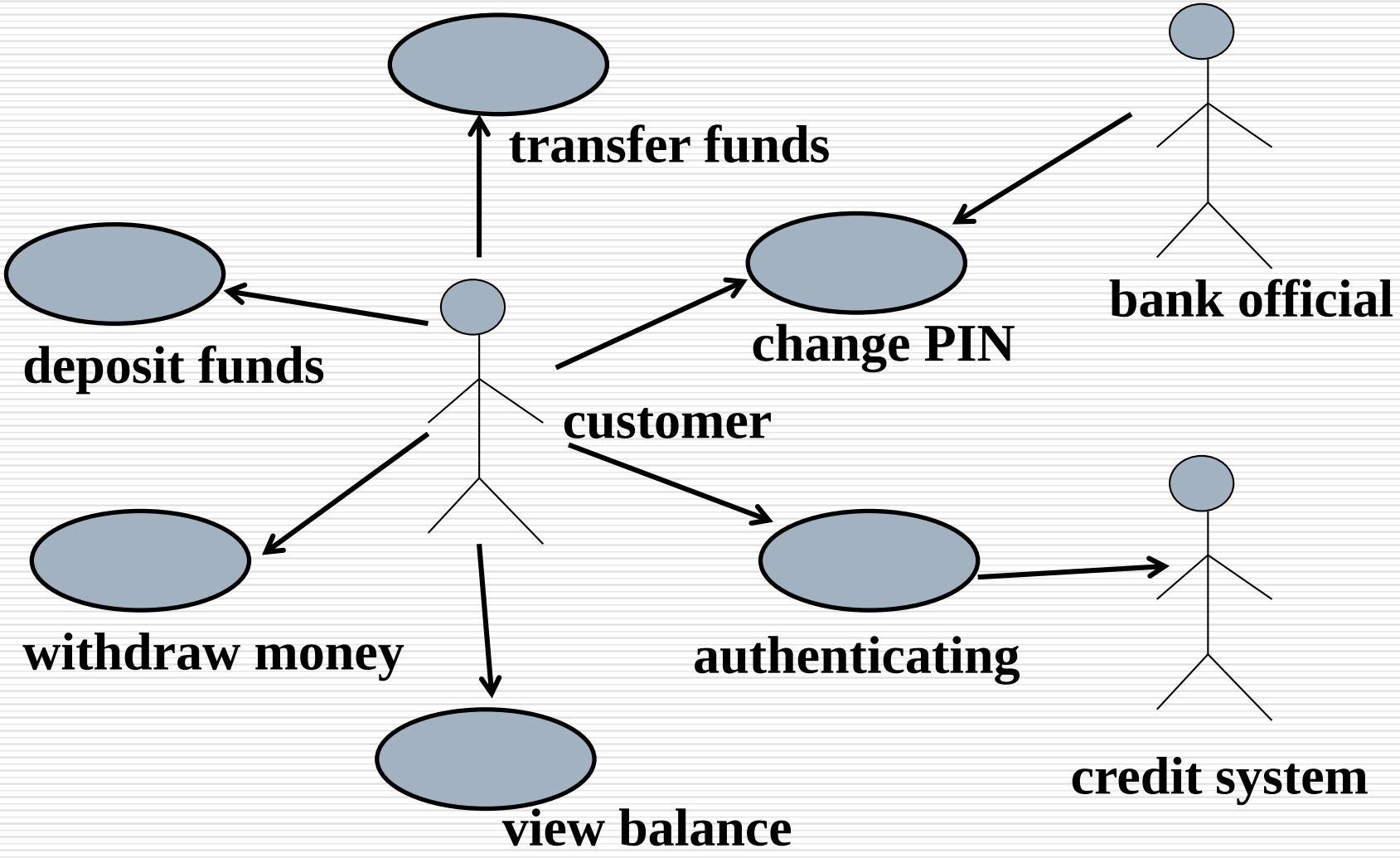
4、由于系统新功能的识别，角色的日常工作被简化或效率提高了吗？（若是，则该用例对于该角色有意义、值得实现）

- 针对系统：

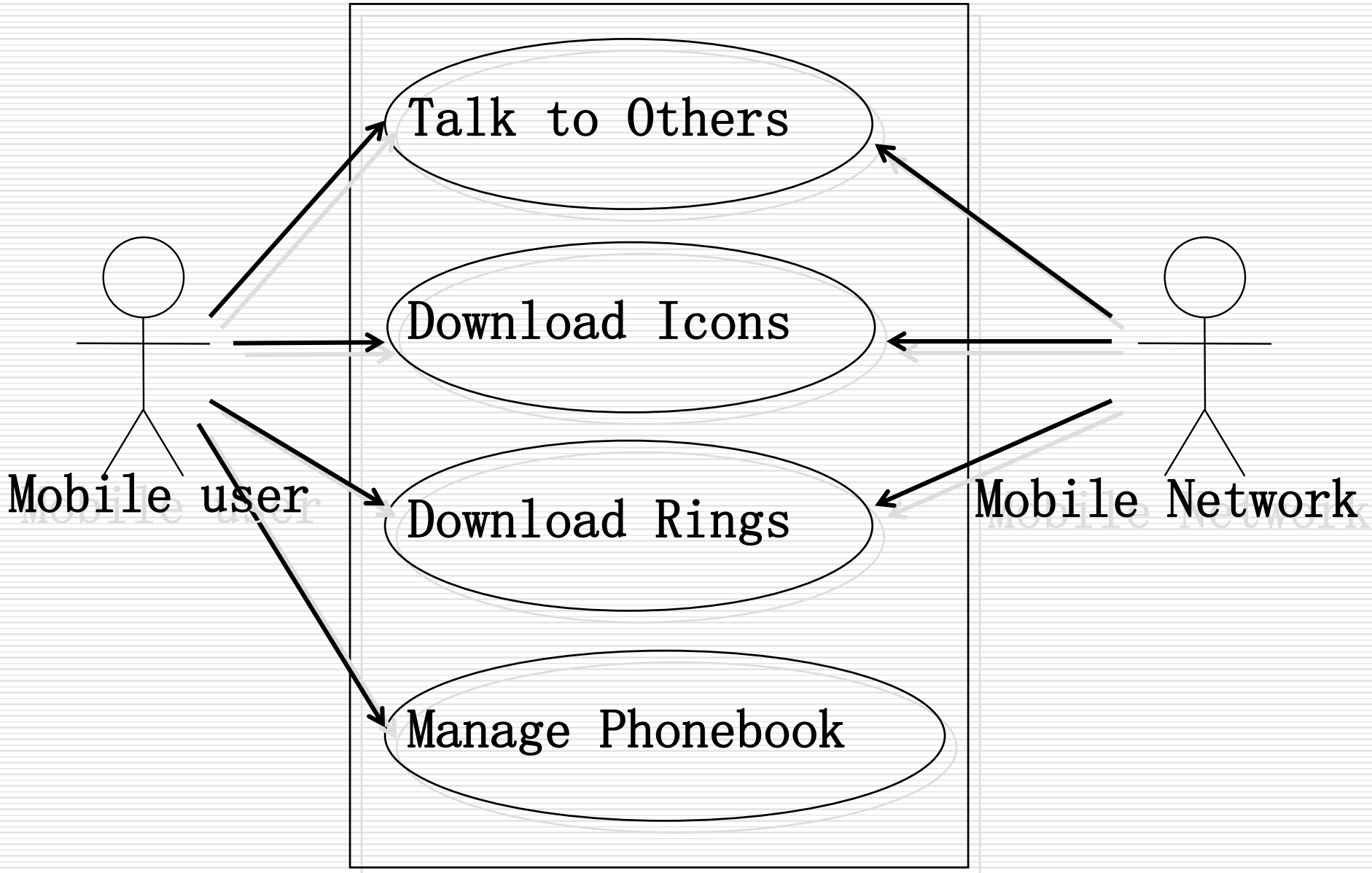
5、系统需要什么样的输入和输出？输入来自哪里？输出去往哪里？

6、该系统的当前状况还存在哪些问题？有哪些改进的方向？

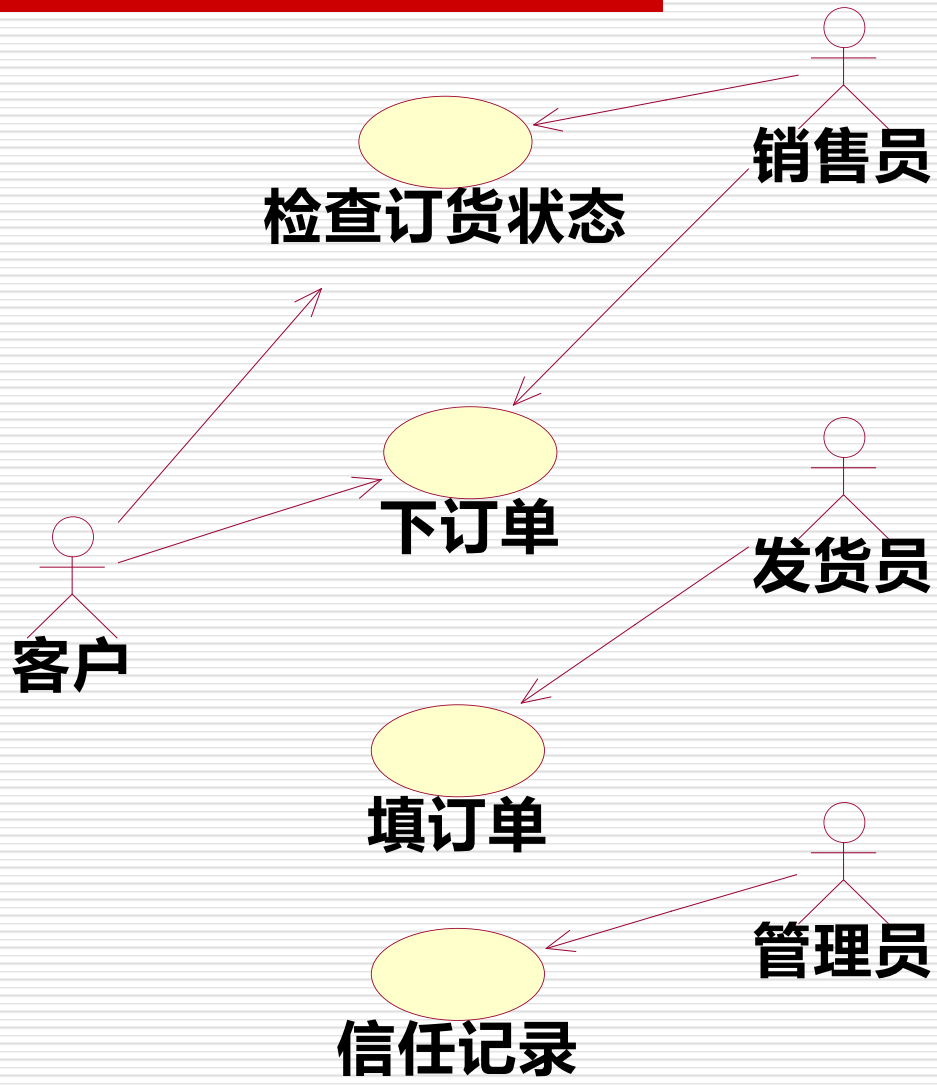
ATM use case diagram



移动电话系统的使用用例图

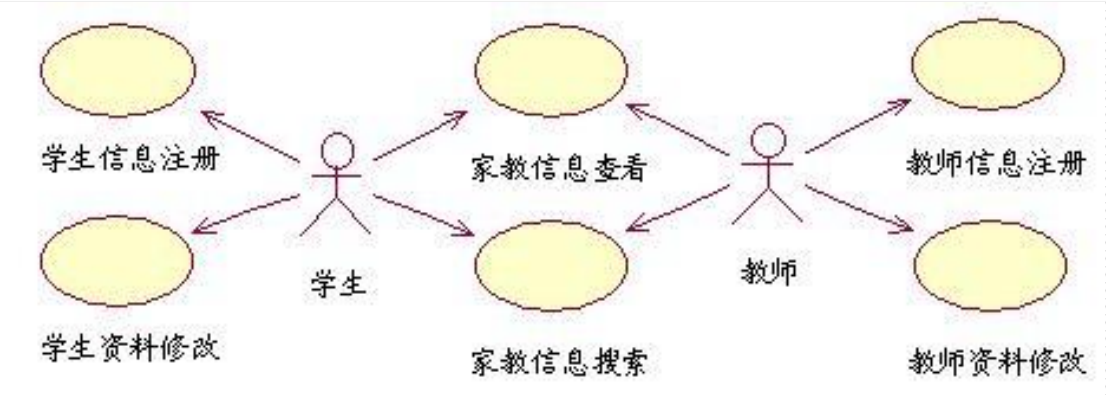


电话订票系统用例图

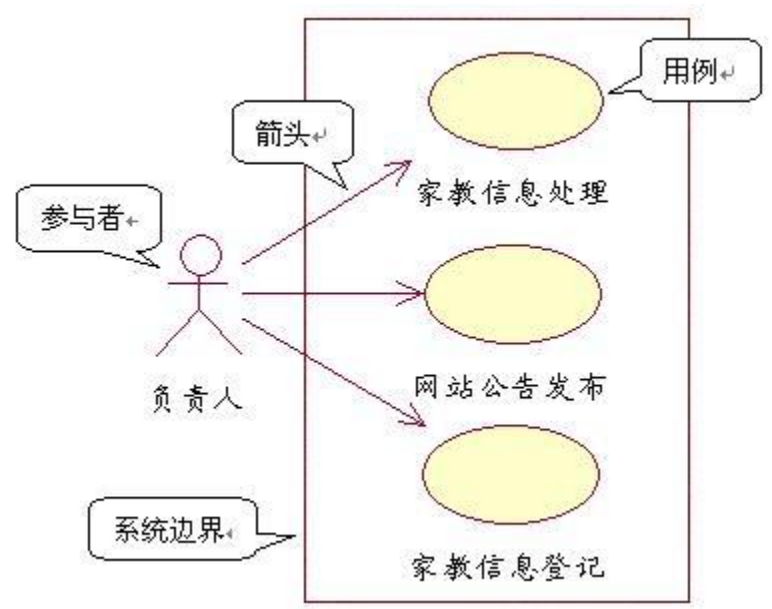


家教网站

前台客户系统的用例图

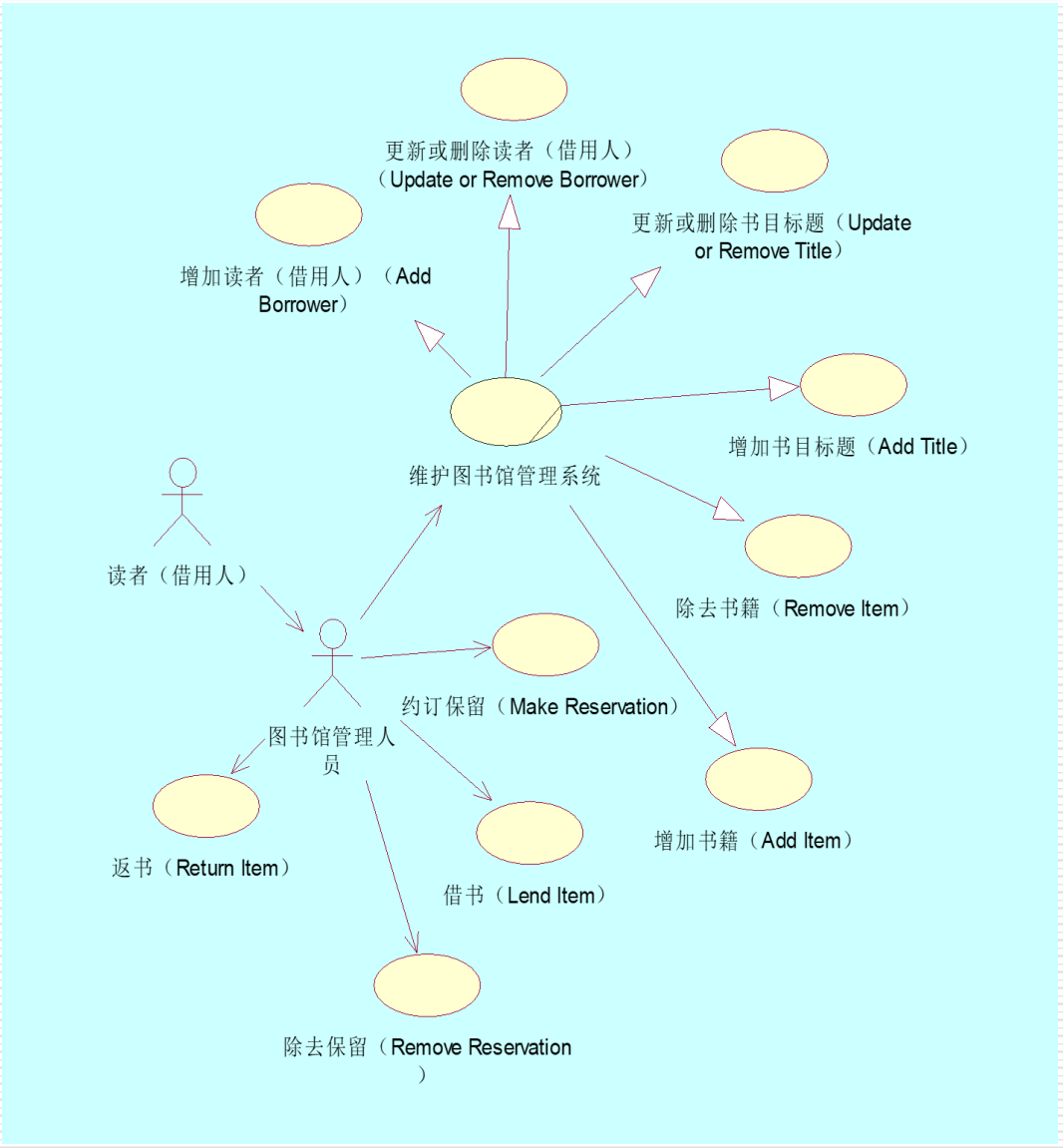


后台管理系统用例图

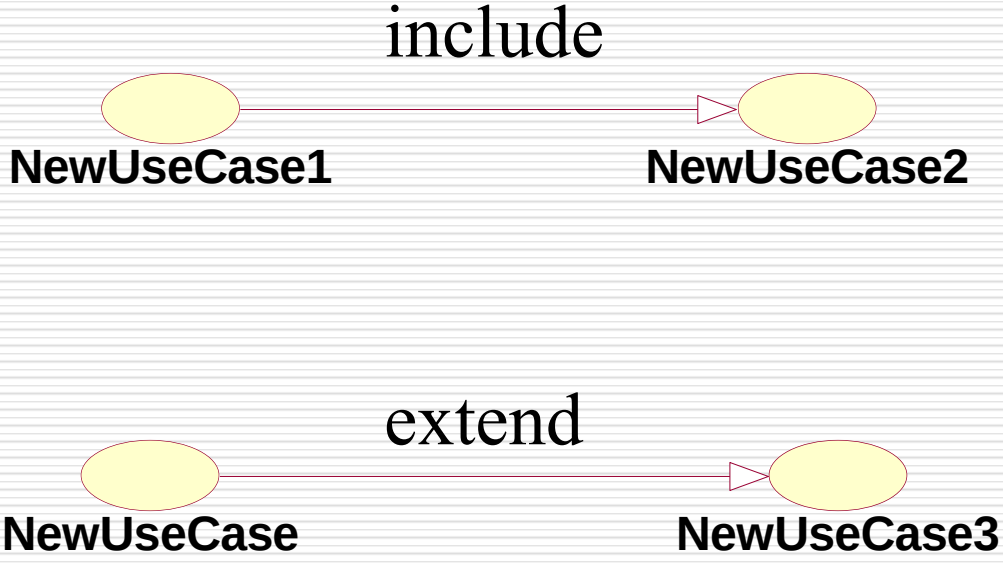


图书馆管理系统的用例

- ✓ 借书（Lend Item）
- ✓ 返书（Return Item）
- ✓ 约订保留（Make Reservation）
- ✓ 除去保留（Remove Reservation）
- ✓ 增加书目标题（Add Title）
- ✓ 更新或删除书目标题（Update or Remove Title）
- ✓ 增加书籍（Add Item）
- ✓ 除去书籍（Remove Item）
- ✓ 增加读者（借用人）（Add Borrower）
- ✓ 更新或删除读者（借用人）（Update or Remove Borrower）



用例图层次



Include：由用例A连向用例B，表示A使用了B的功能(use)

Extend：由用例A连向用例B，表示B描述了基本要求，A描述了特殊要求，A是一种扩展

交易系统用例图层次

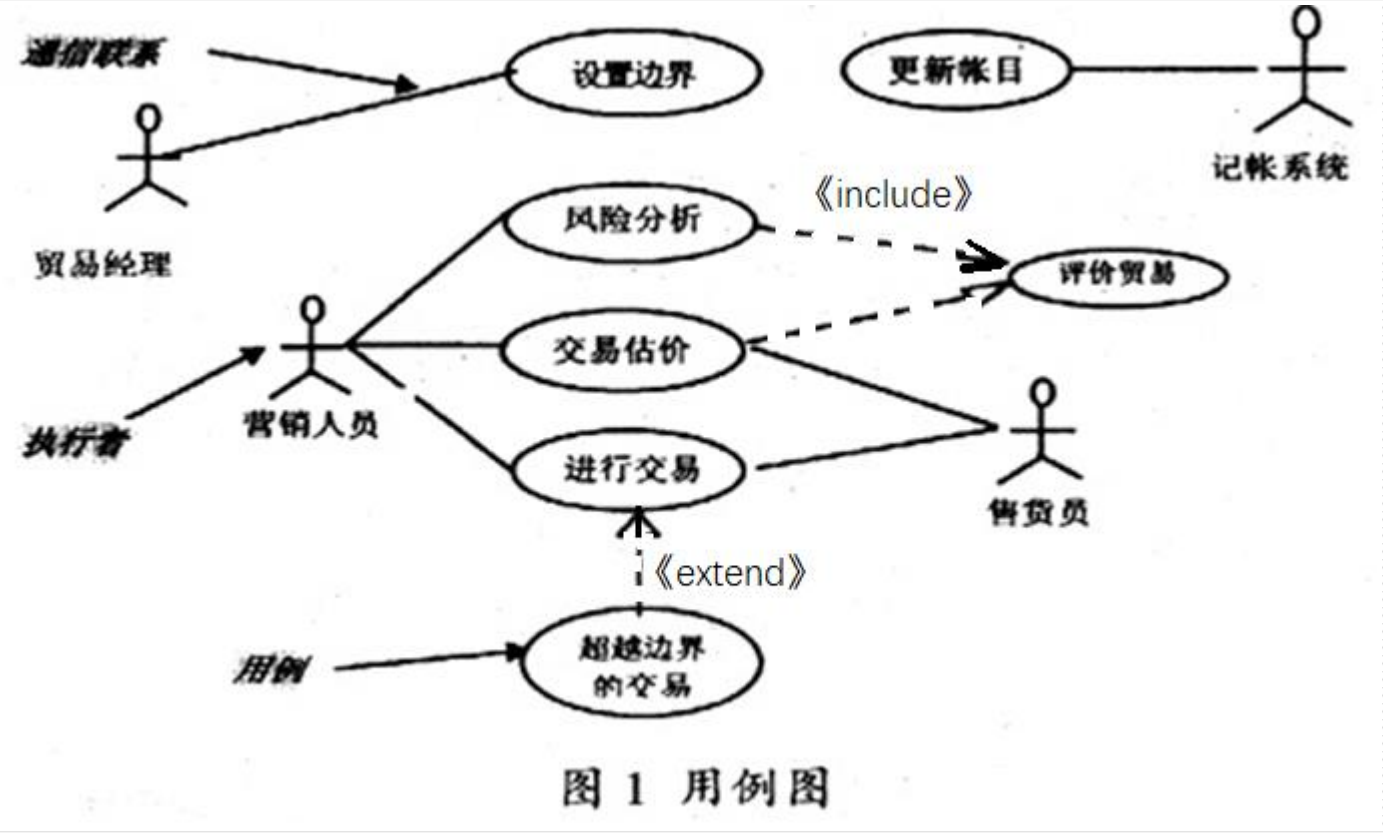


图 1 用例图

对一个十人年的项目来说，大约要80个左右的用例

Use Case图的建立步骤(小结）

(1) 找出系统外部的参与者和外部系统，确定系统的边界和范围；

(2) 确定每一个参与者所期望的系统行为；

(3) 把这些系统行为命名为Use Case；

(4) 用泛化、包含、扩展等关系处理系统行为的公共或变更部分；

(5) 编制和解释每一个Use Case的脚本；

(6) 绘制Use Case图；

[(7)] 区分主事件流和异常情况的事件流，可以把表示异常情况的事件流作为单独的Use Case处理；

(8) 细化Use Case图，解决Use Case间的重复与冲突问题。

OOA, UML, 静态模型

类建模方法：（class diagram, 类图）

The key is finding objects

Object Oriented Approaches describe Systems in terms of
Objects
which interact with each other
by passing messages
in order to perform
the functions required from the system

Thinking about Objects

Object Orientation is natural

Objects exist in the real world

Objects interact with other objects in the real world

Objects use other objects to do things

Objects provide services (i.e. functionality or use) to other objects

如何提取类或对象

5 layers:

- ✓Subject ✓主题
- ✓class & object ✓类 & 对象
- ✓Attributes ✓属性
- ✓Services ✓服务
- ✓Structure ✓结构

OOA，UML的类图

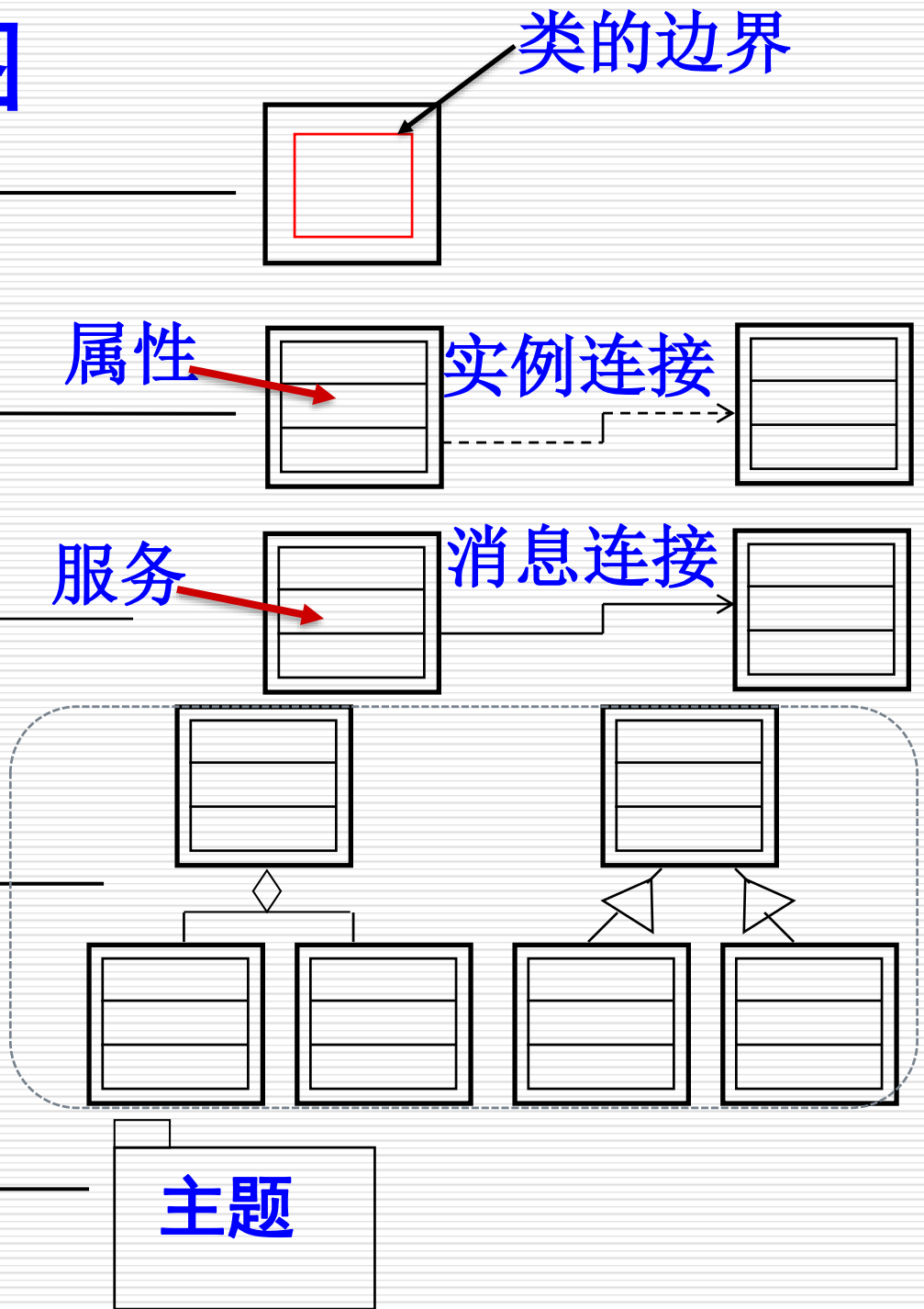
Class &object layer
(类及对象层)

Attribute layer
(属性层)

Service layer
(服务层)

Structure layer
(结构层)

Subject layer
(主题层)



提取阶段由五个活动组成

(1) 标识类及对象

(2) 标识结构

(3) 标识主题

(4) 定义属性及实例连接

(5) 定义服务及消息连接

五个步骤常根据需要交叉进行

步骤1：识别类与对象

发现对象

- 考虑问题域
  - ✓ 人员
  - ✓ 组织
  - ✓ 物品
  - ✓ 设备
  - ✓ 事件
  - ✓ 表格结构
- 考虑系统边界
  - ✓ 人员
  - ✓ 设备
  - ✓ 外系统
- 考虑系统责任

步骤2： 定义属性与服务

- 定义属性
- 定义服务
  - ✓ 词性法
  - ✓ 对象的状态与状态转换图

    状态图的节点，状态图的边）

Mapping parts of speech to object model components [Abbot 1983]

Part of speech	*Model component*	*Example*
Proper noun	object	Jim Smith
Improper noun	class	Toy, doll
Doing verb	method	Buy, recommend
being verb	inheritance	is-a (kind-of)
having verb	aggregation	has an
modal verb	constraint	must be
adjective	attribute	3 years old
transitive verb	method	enter
intransitive verb	method (event)	depends on

实例：饮料自动售货机系统

设置

一个饮料自动售货机可以放置五种不同或部分相同的饮料，可由厂商根据销售状况自动调配，并可随时重新设置售价，但售货机最多仅能放置50罐饮料，其按钮设计在各种饮料样本的下方，若经金额计算器累计金额足够，则选择键灯会亮；若某一种饮料已销售完毕，则售完灯会亮。

销售

顾客将硬币投入售货机，经累加金额足额的饮料选择键灯亮，等顾客按键选择。顾客按键后饮料由取物楼掉出，并自动结算及找钱。

取消交易

顾客可在按下选择键前任何一个时刻，拉动退币杆取消交易收回硬币。

找出饮料自动售货机系统中的对象

设置

一个饮料自动售货机可以放置五种不同或部分相同的饮料，可由厂商根据销售状况自动调配，并可随时重新设置售价，但售货机最多仅能放置50罐饮料，其按钮设计在各种饮料样本的下方，若经金额计算器累计金额足够，则选择键灯会亮；若某一种饮料已销售完毕，则售完灯会亮。

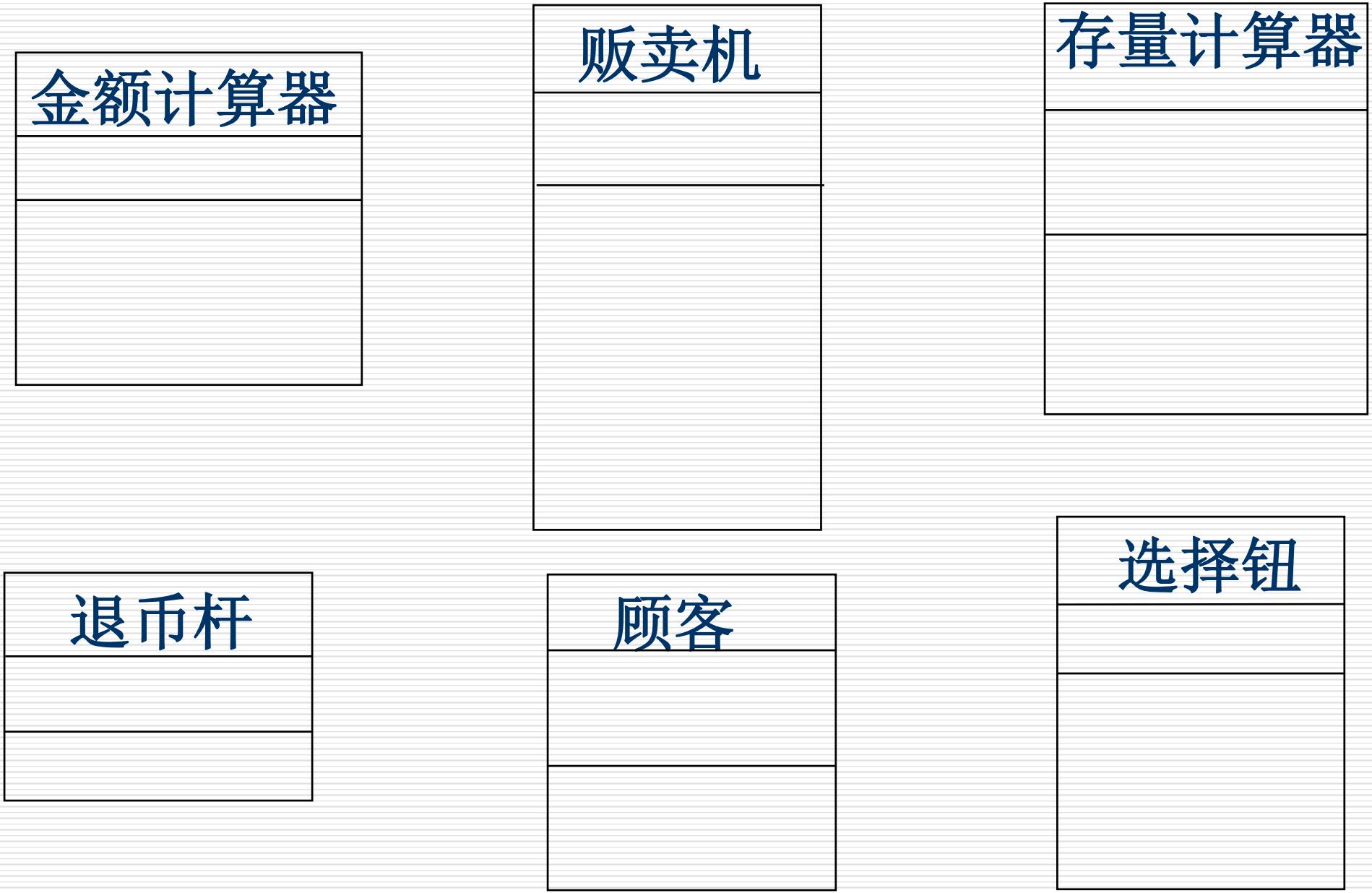
销售

顾客将硬币投入售货机，经累加金额足额的饮料选择键灯亮，等顾客按键选择。顾客按键后饮料由取物楼掉出，并自动结算及找钱。

取消交易

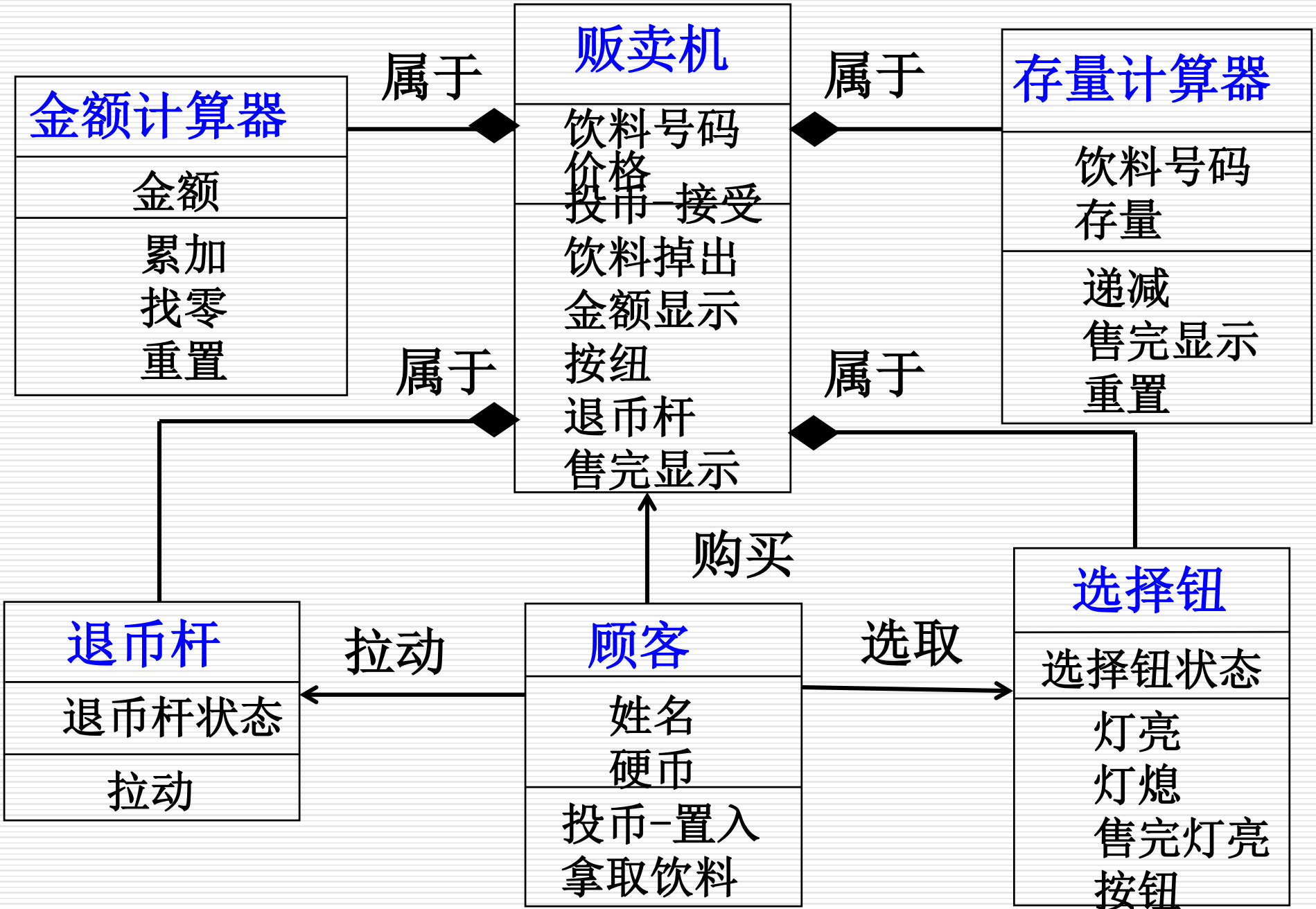
顾客可在按下选择键前任何一个时刻，拉动退币杆取消交易收回硬币。

饮料自动售货机系统对象图



饮料自动售货机系统对象图

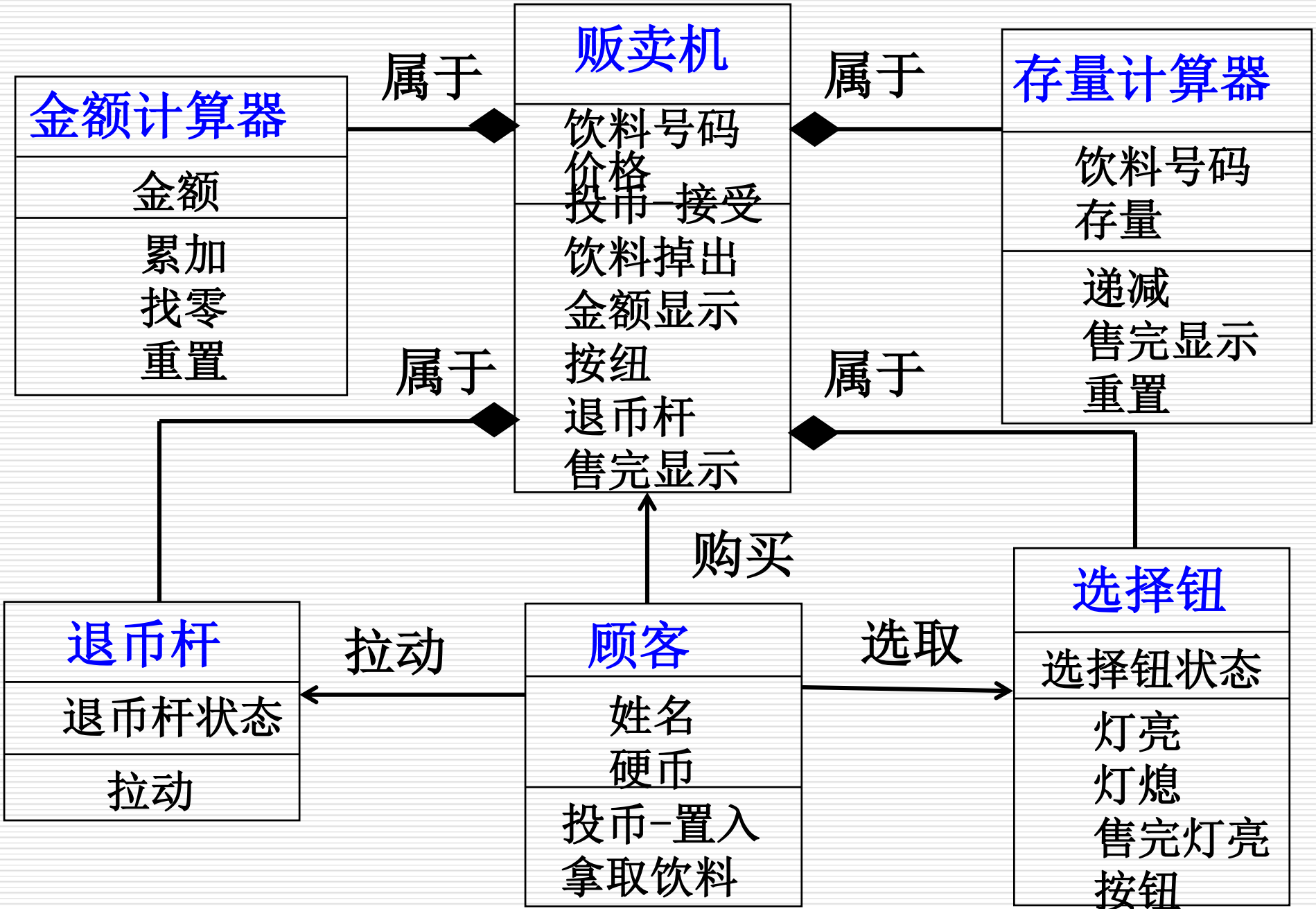
饮料自动售货机系统对象图

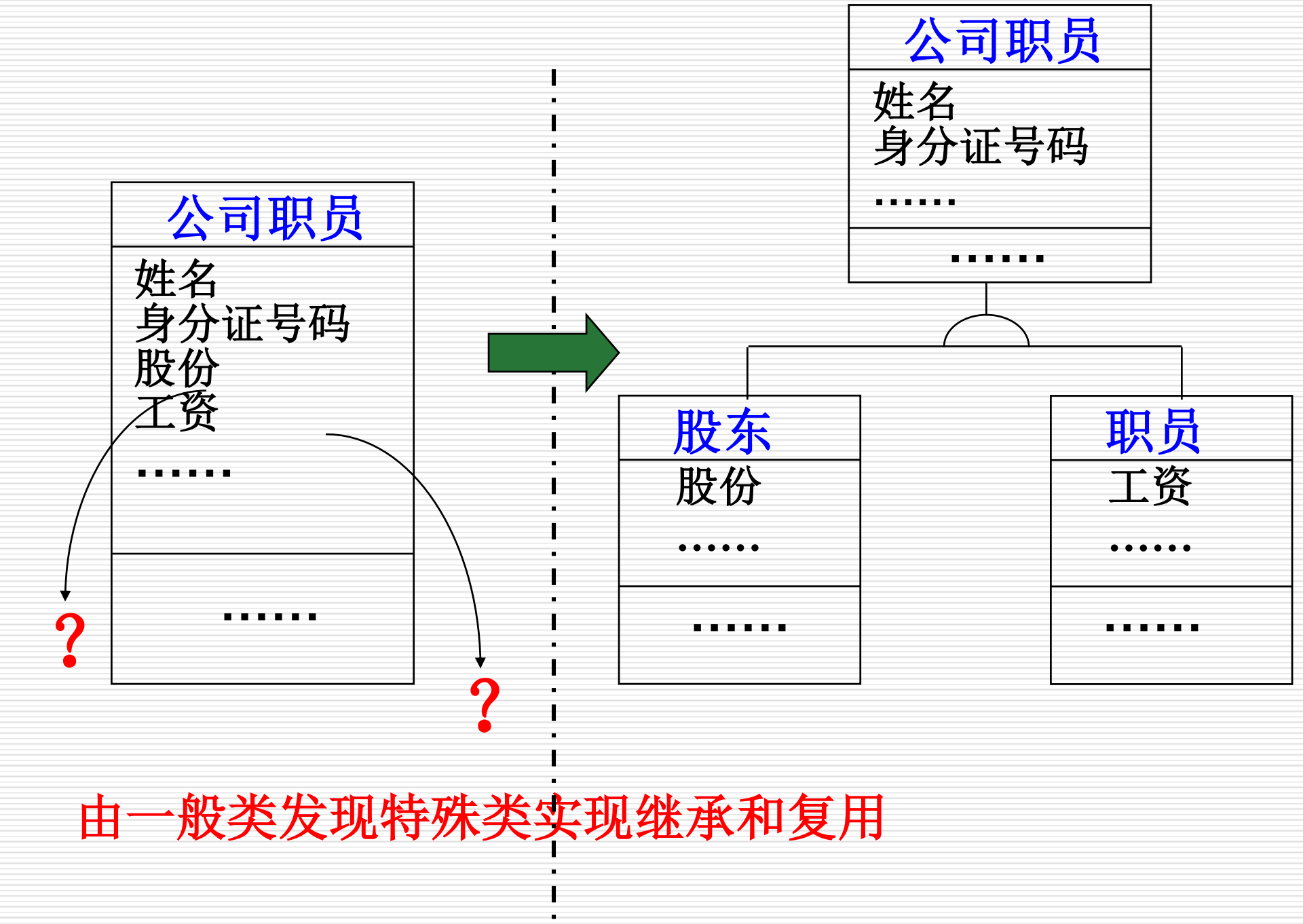


步骤3：定义结构与连接

- 初步确定关联
  - ✓ 对应于描述性动词或动词短语
  - ✓ 需求陈述中隐含
  - ✓ 根据问题域知识得出
- 筛选
- 完善
- 分析标识对象之间的关系
  - ✓ 对象之间的分类关系：一般-特殊结构
  - ✓ 对象之间的组成关系：整体-部分结构
  - ✓ 对象之间的静态联系：实例连接
  - ✓ 对象之间的动态关系：消息连接

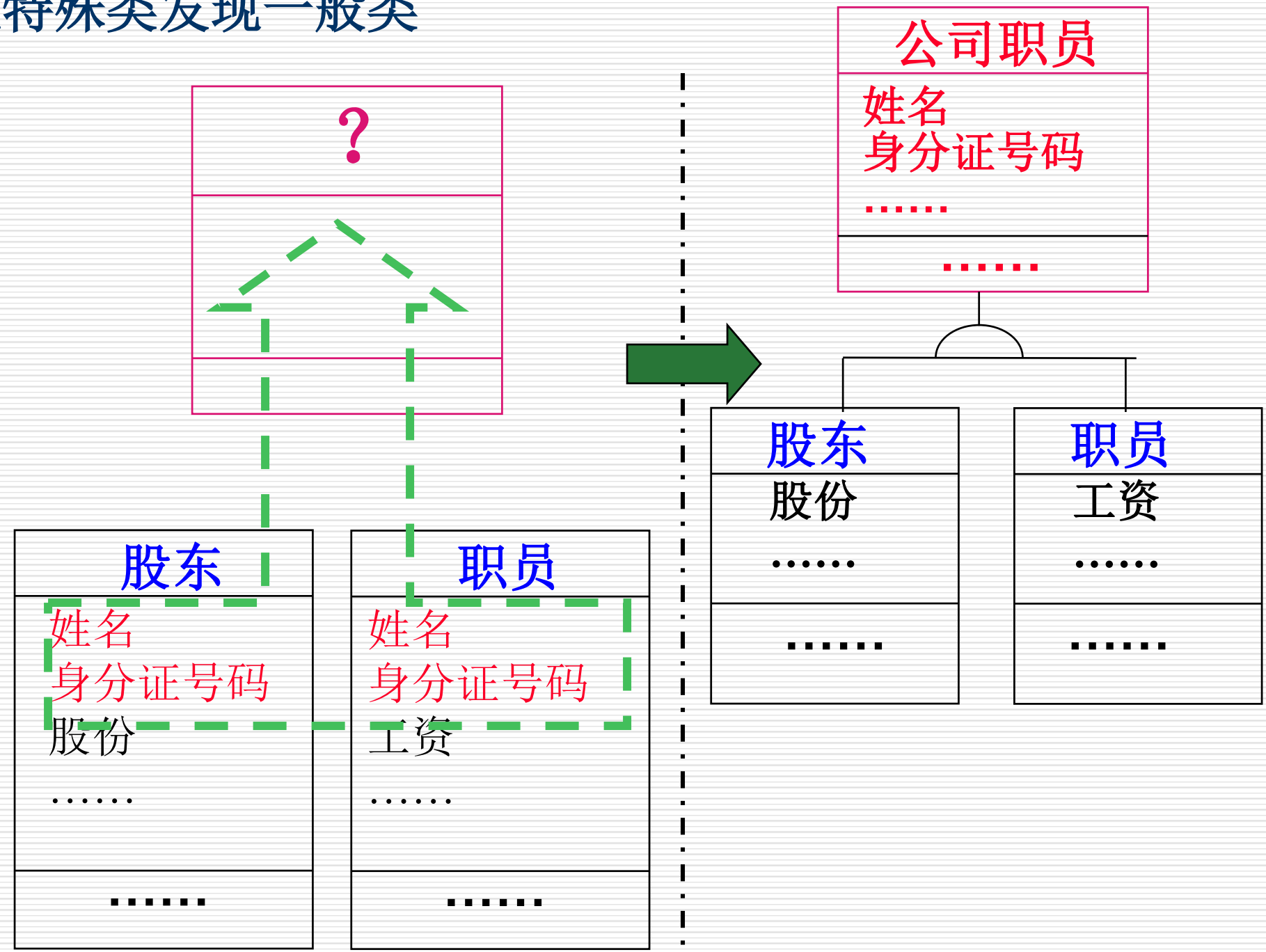
饮料自动售货机系统对象图

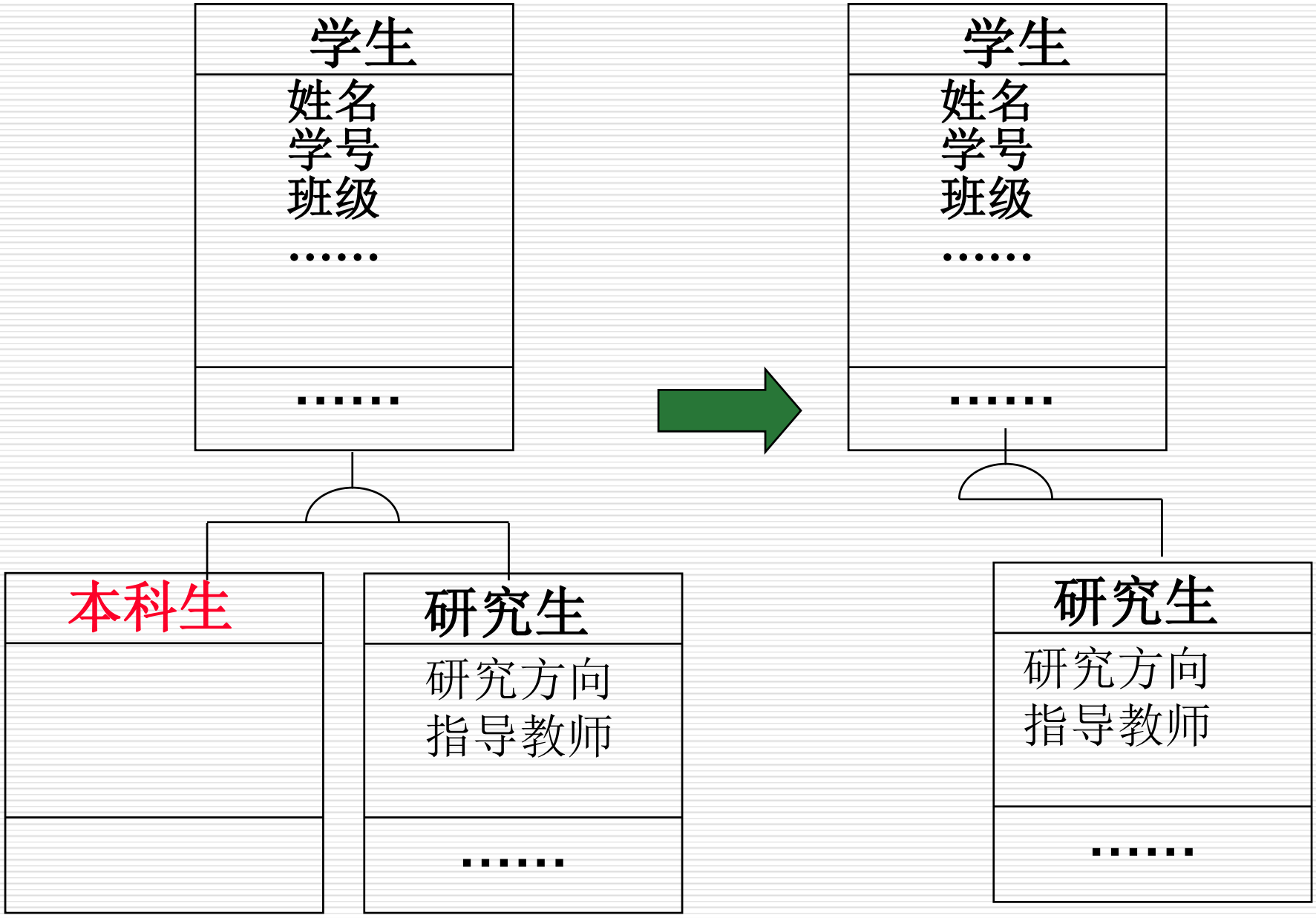




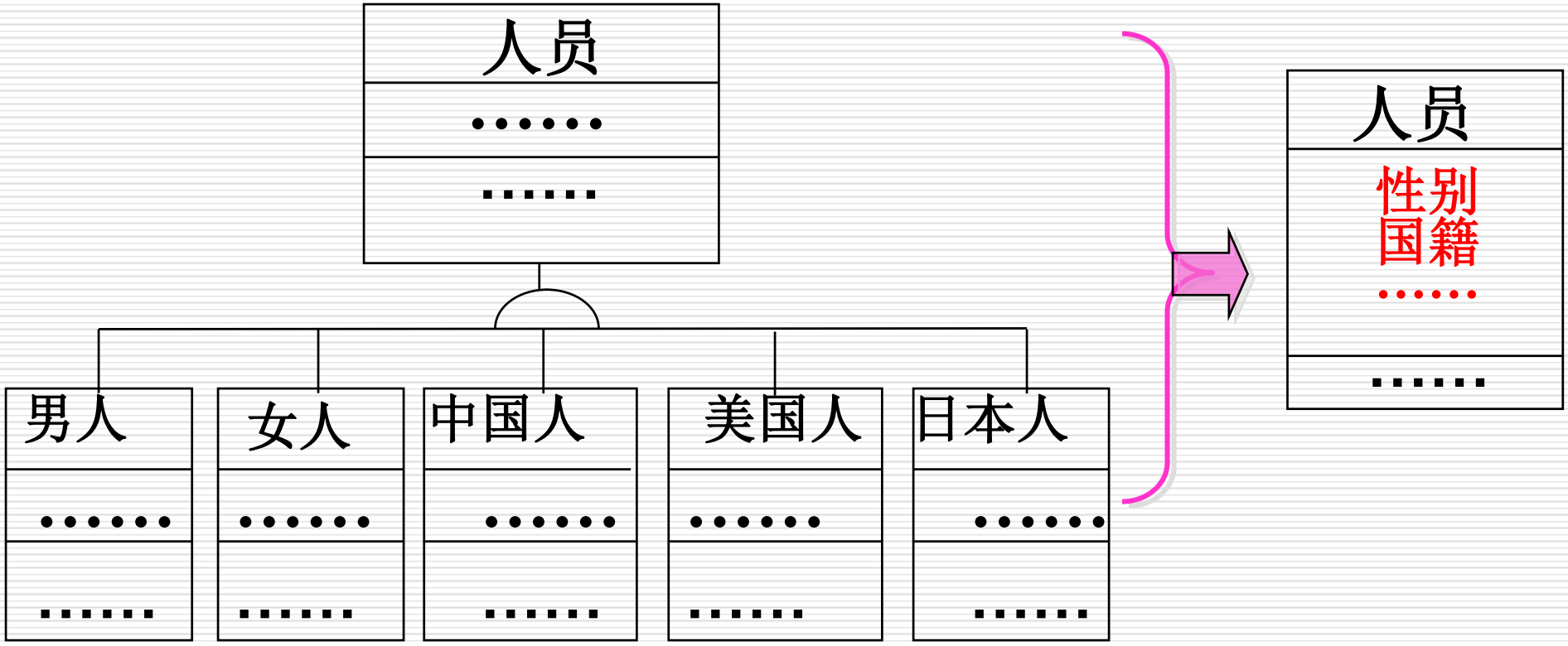
由一般类发现特殊类实现继承和复用

从特殊类发现一般类

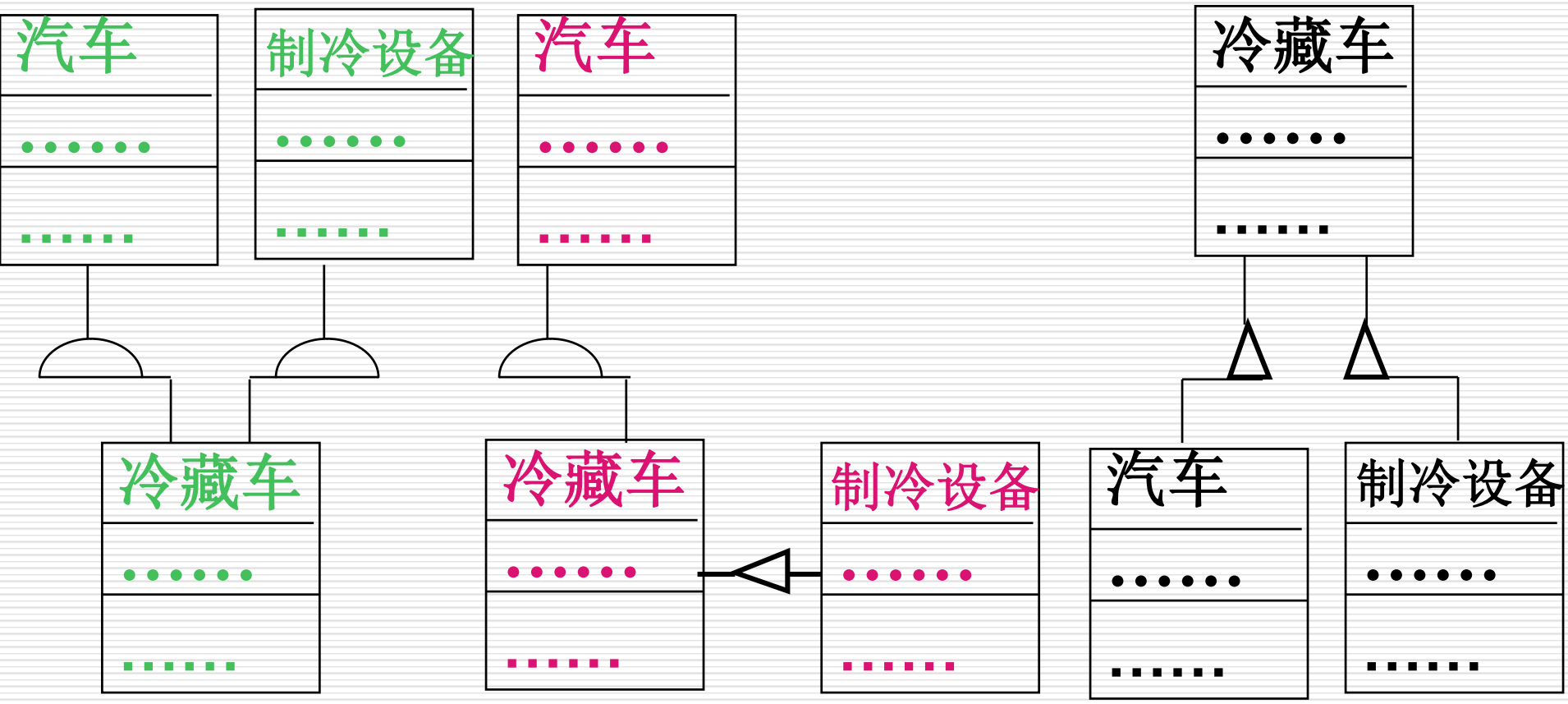



学生
姓名
学号
班级
……
……
本科生
研究生
研究方向
指导教师
……
学生
姓名
学号
班级
……
……
研究生
研究方向
指导教师
……

通过增加属性简化一般-特殊结构



两种结构的变通

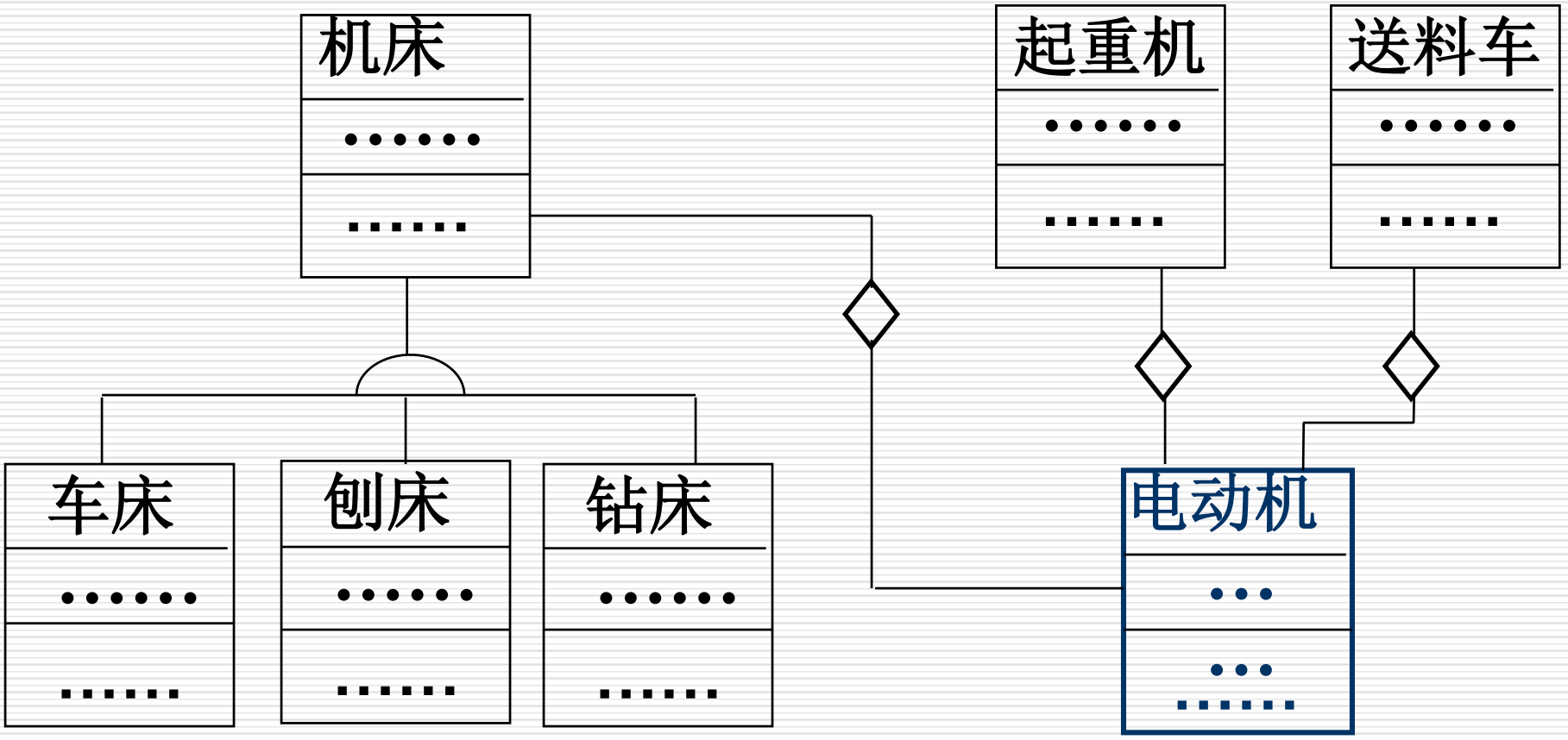


仅用一般
-特殊结构

两种结构
同用

仅用整体
-部分结构

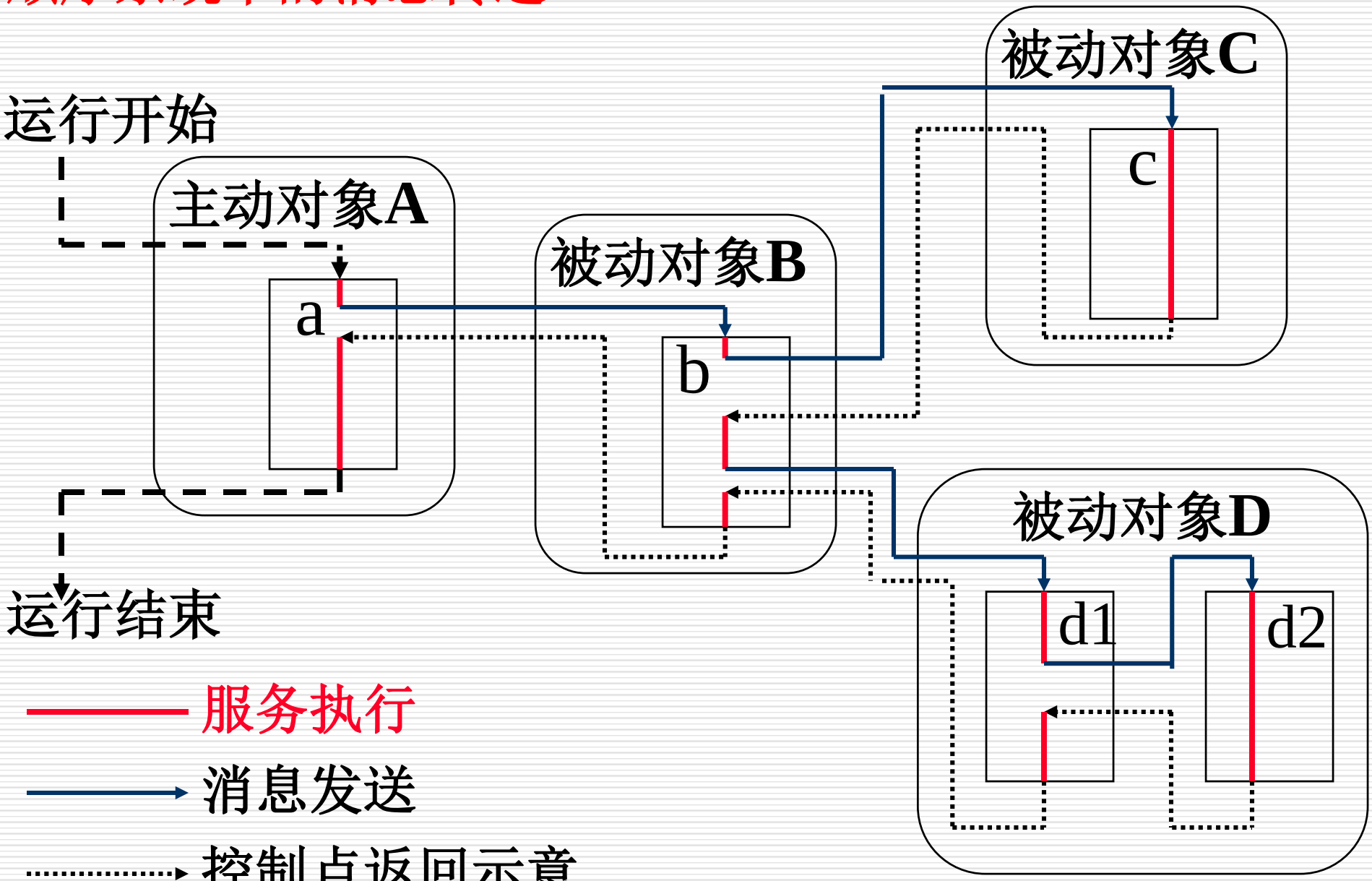
用整体-部分结构实现复用



结构连接中的消息传递

分析和认识对象之间在行为上的往来关系

顺序系统中的消息传递


运行开始
主动对象A
a
被动对象B
b
被动对象C
c
被动对象D
d1
d2
运行结束
服务执行
消息发送
控制点返回示意

步骤4：标识主题

Coad/Yourdon方法中主题的概念：

主题是把一组具有较强联系的类，组织在一起而得到的类的集合。

主题概念及其用途

- 主题层是在OOA基本模型(类图)之上，建立一个能帮助人们从不同的认识层次来理解系统的补充模型；
- 主题是一种比类和对象抽象层次更高、粒度更大的概念，用以建立系统的高层抽象视图；
- 主题有助于指导系统设计者或用户等理解一个大的系统模型，有助于组织一个大项目的工作。

主题概念的特点

- 是由一组类构成的集合
- 一个主题内部的对象类应具有某种意义上的内在联系
  - ✓ 描述系统中相对独立的组成部分（如一个子系统）
  - ✓ 描述系统中某一方面的事物（如人员、设备）
  - ✓ 解决系统中某一方面的问题（如输入输出）
- 主题的划分有一定的灵活性和随意性

如何划分主题

- 把每个结构作为一个主题；
  （选取结构中最上层的类作为一主题）
- 互相联系的类可划分到一个主题；
- 把不属于任何结构，也没有实例连接的类作为一个主题。

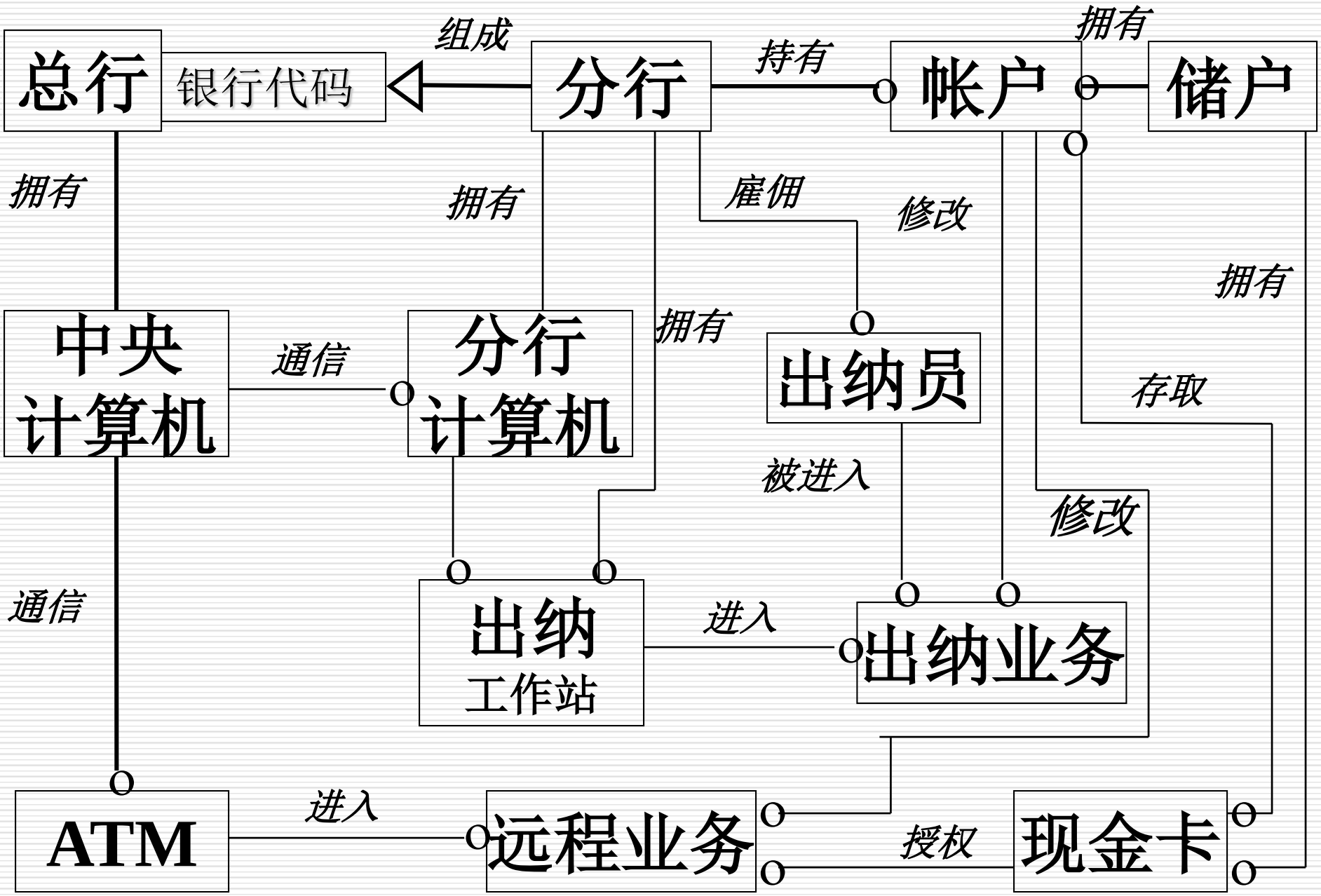
何时引入主题

依赖于模型自身的复杂性

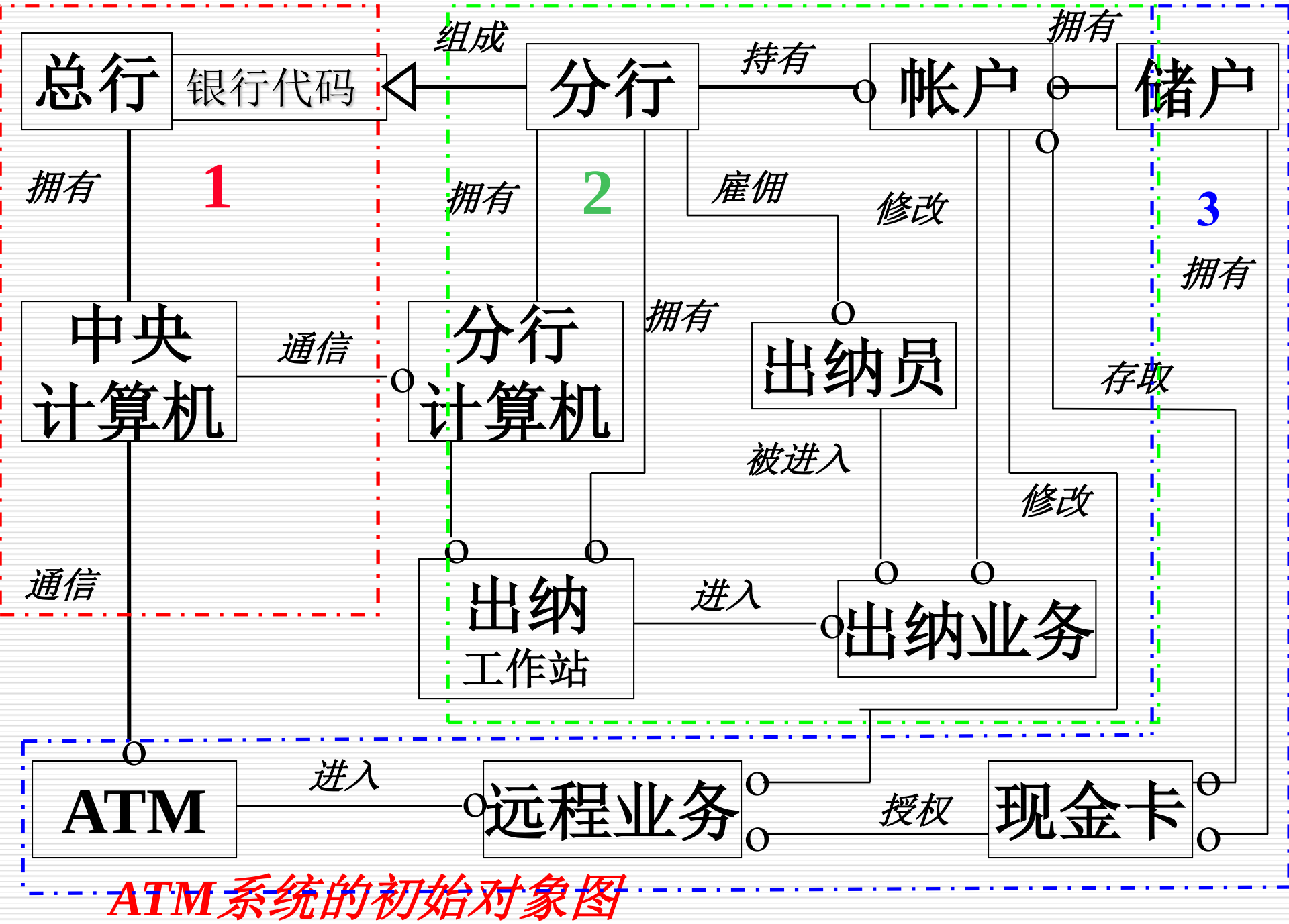
- 小系统： 不需引入主题；
- 中等系统:先标识类及对象，
  然后引入主题；
- 大系统：先标识主题，对问题域进行
  划分，分给不同的任务组；

主题层次的控制

- 中小型系统可只设一层主题，最多不超过两层；
- 大型系统可只设两层主题，最多不超过三层。



ATM系统的初始对象图


总行
银行代码
组成
分行
持有
帐户
拥有
储户
拥有
1
拥有
2
雇佣
修改
3
拥有
中央
计算机
通信
分行
计算机
拥有
出纳员
存取
被进入
修改
通信
出纳
工作站
进入
出纳业务
ATM
进入
远程业务
授权
现金卡
ATM系统的初始对象图

OOA, UML, 动态模型

- 动态模型（sequence, state diagram, 顺序和状态图）

- 用来描述系统与时间相关的动态行为即系统的控制逻辑，表现对象彼此间经过相互作用后，随时间改变的不同运算顺序。

- 动态模型以“事件”（Events）和“状态”（States）为其模型的主要概念。

事件和状态

状态：对象所具有的属性值，具有时间性和持续性。状态是对象属性的一种抽象，状态指明了对象对输入事件的响应。

事件：对象的触发行为，指从一个对象到另一个对象的信息的单向传递(消息，调用）。

脚本：系统某一执行期间内出现的一系列事件。脚本范围，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，也可以执行系统的模块。

在系统中具有属性值、关联的对象，可能相互激发，引起状态的一系列变化。

事件追踪图举例

打电话的场景

1、拿起电话受话器
2、电话忙音开始
3、拨电话号码数5
4、电话忙音结束
5、拨电话号码数5
6、拨电话号码数5
7、拨电话号码数1
8、拨电话号码数2
……
11、对方电话开始振铃
12、打电话者听见振铃声
13、对方接电话
14、打电话方停止振铃声
15、通电话
16、对方挂电话
17、电话切断
18、打电话者挂电话

打电话的脚本

打电话的事件追踪图

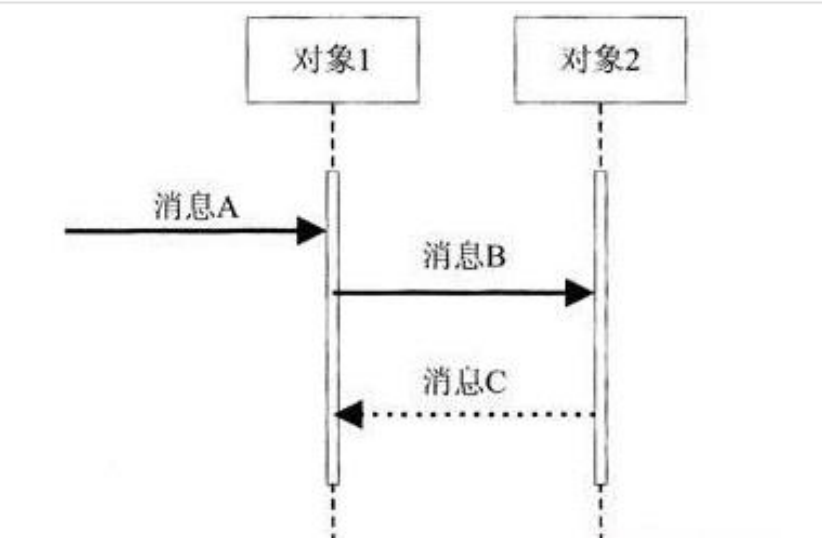
举例：饮料自动售货机的事件追踪图

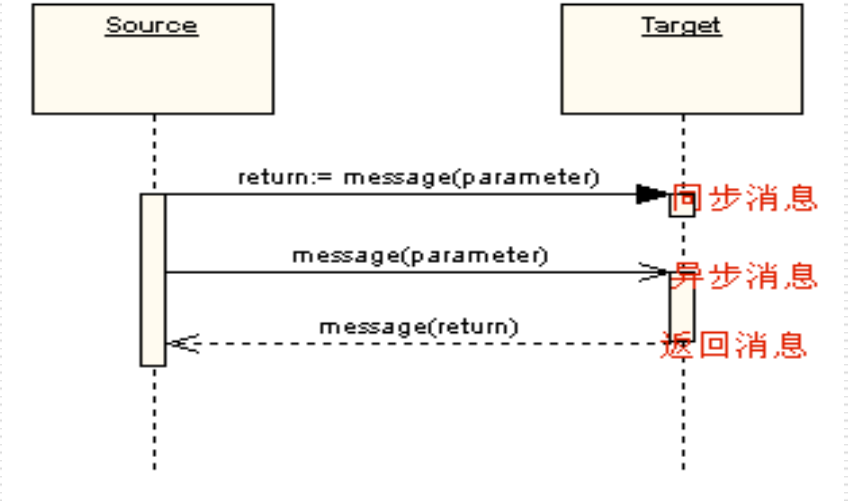
Sequence diagram

顺序图（序列图，时序图）是一种交互图（interaction diagram）强调的是时间和消息的次序。

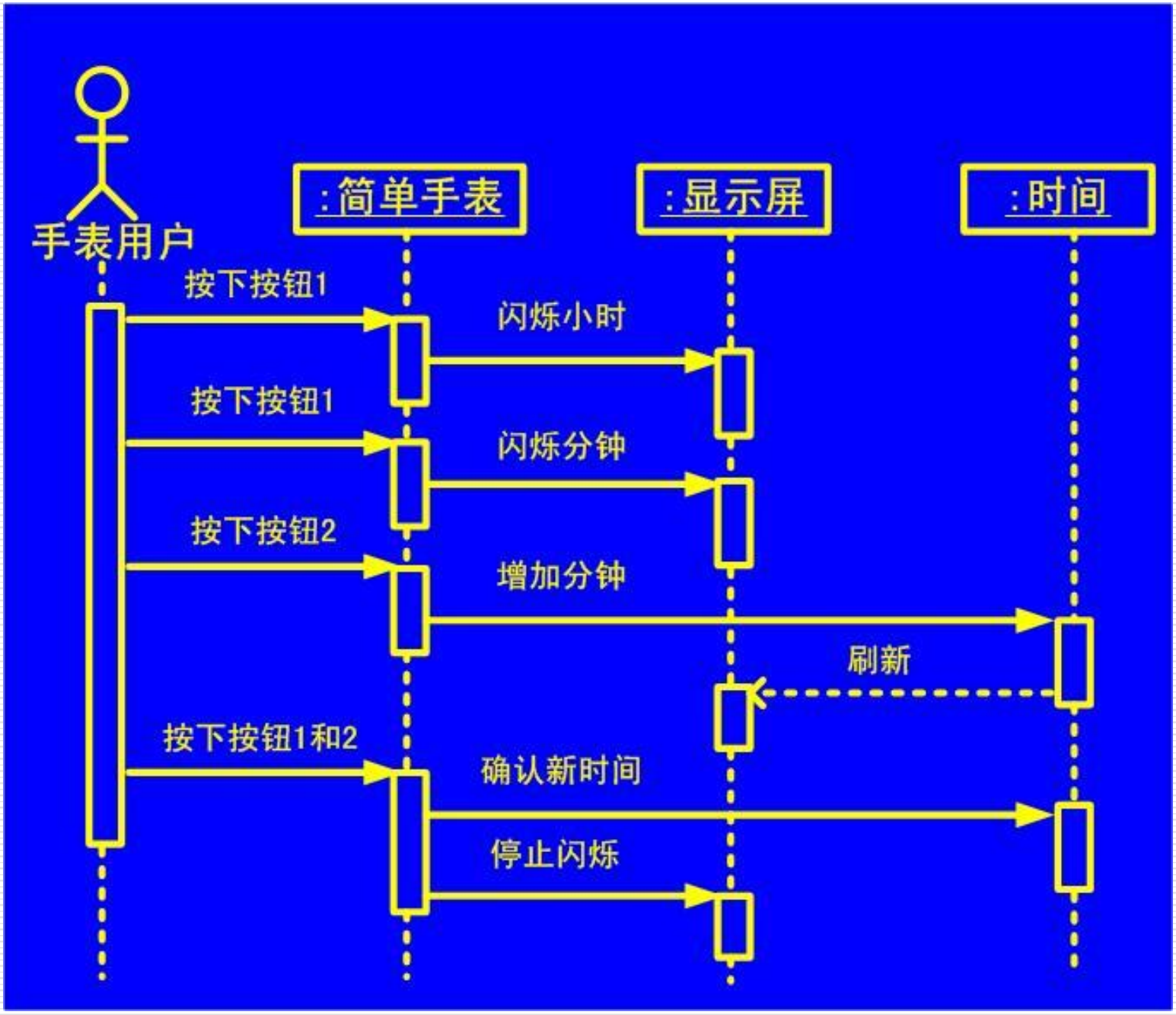
序列图显示了一系列的对象和在这些对象之间发送和接收的消息。

对象通常是命名或匿名的类的实例，也可以代表其他事物的实例

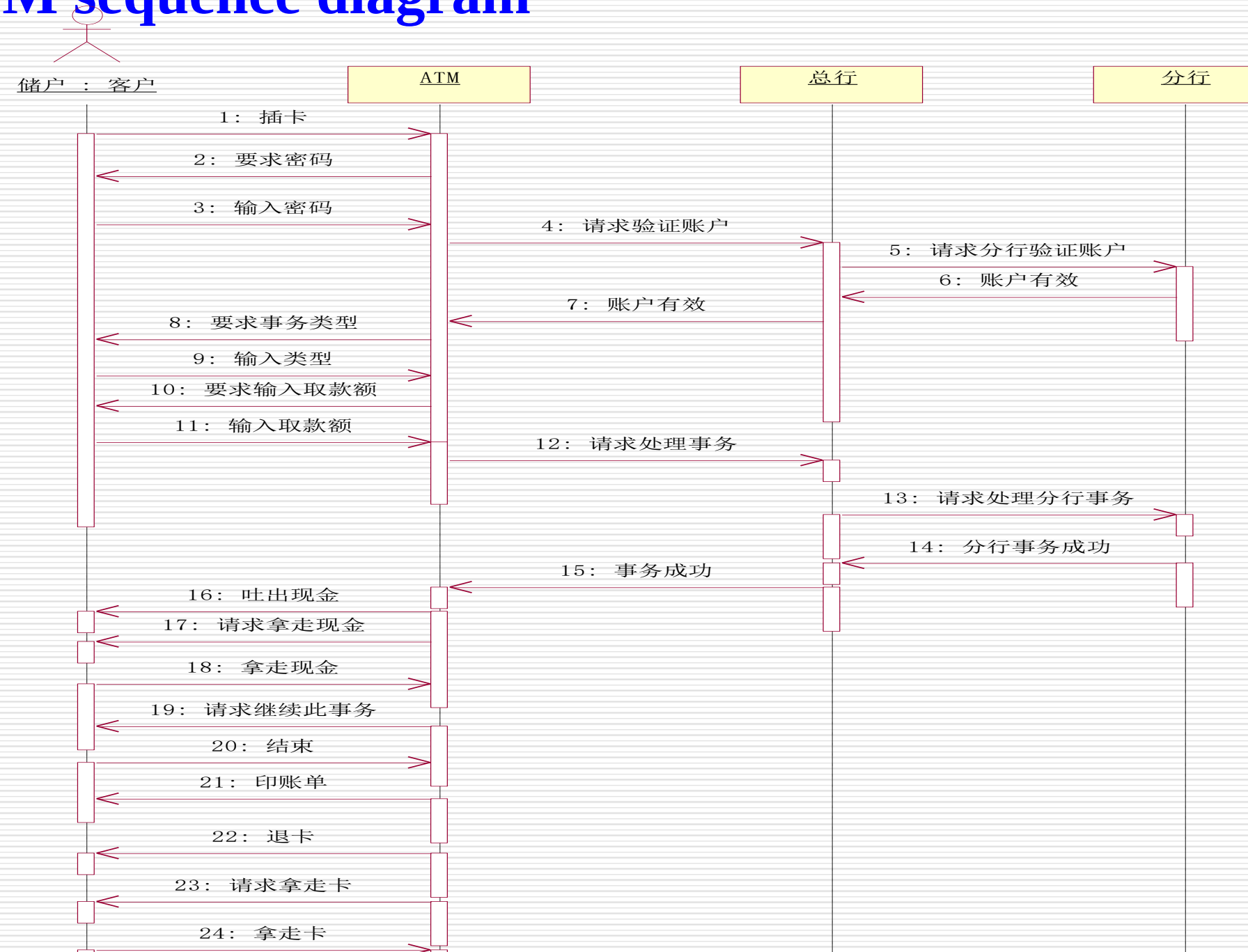




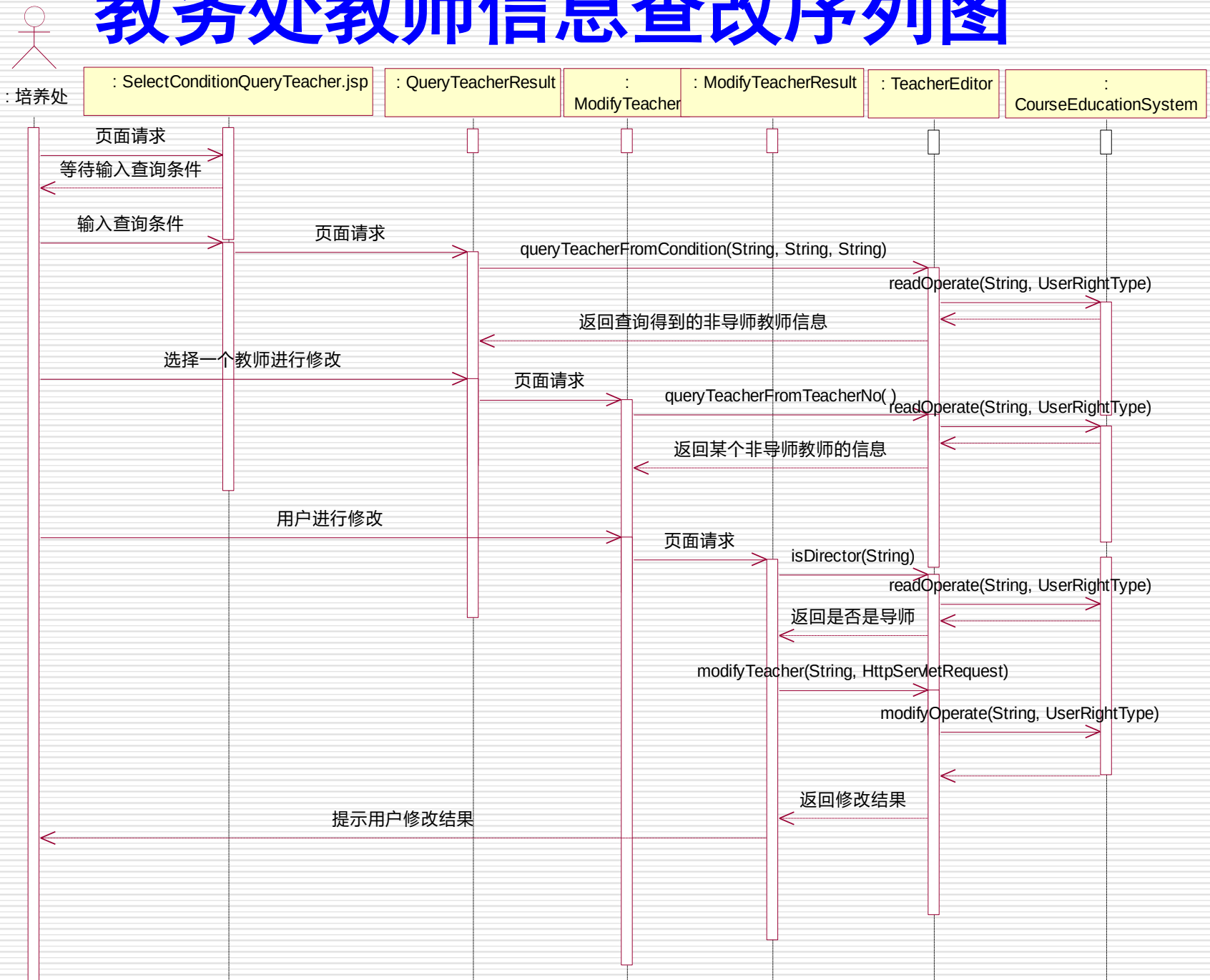
举例：
手表
调时


手表用户
:简单手表
:显示屏
:时间
按下按钮1
闪烁小时
按下按钮1
闪烁分钟
按下按钮2
增加分钟
刷新
按下按钮1和2
确认新时间
停止闪烁

ATM sequence diagram



教务处教师信息查改序列图



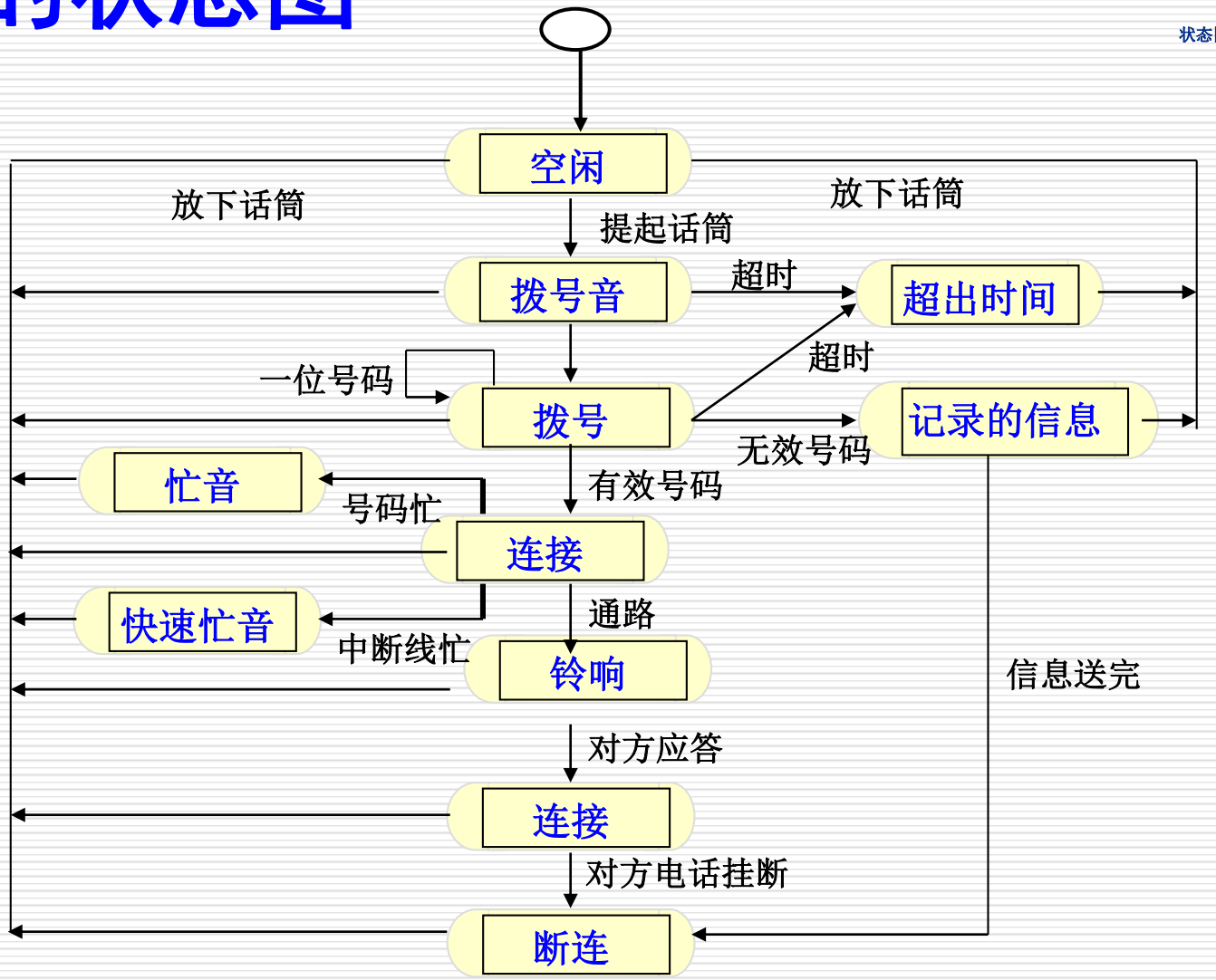
State diagram 状态图

状态图（状态机），由状态、转换、事件和活动组成。

描述单个对象，系统的活动情况。

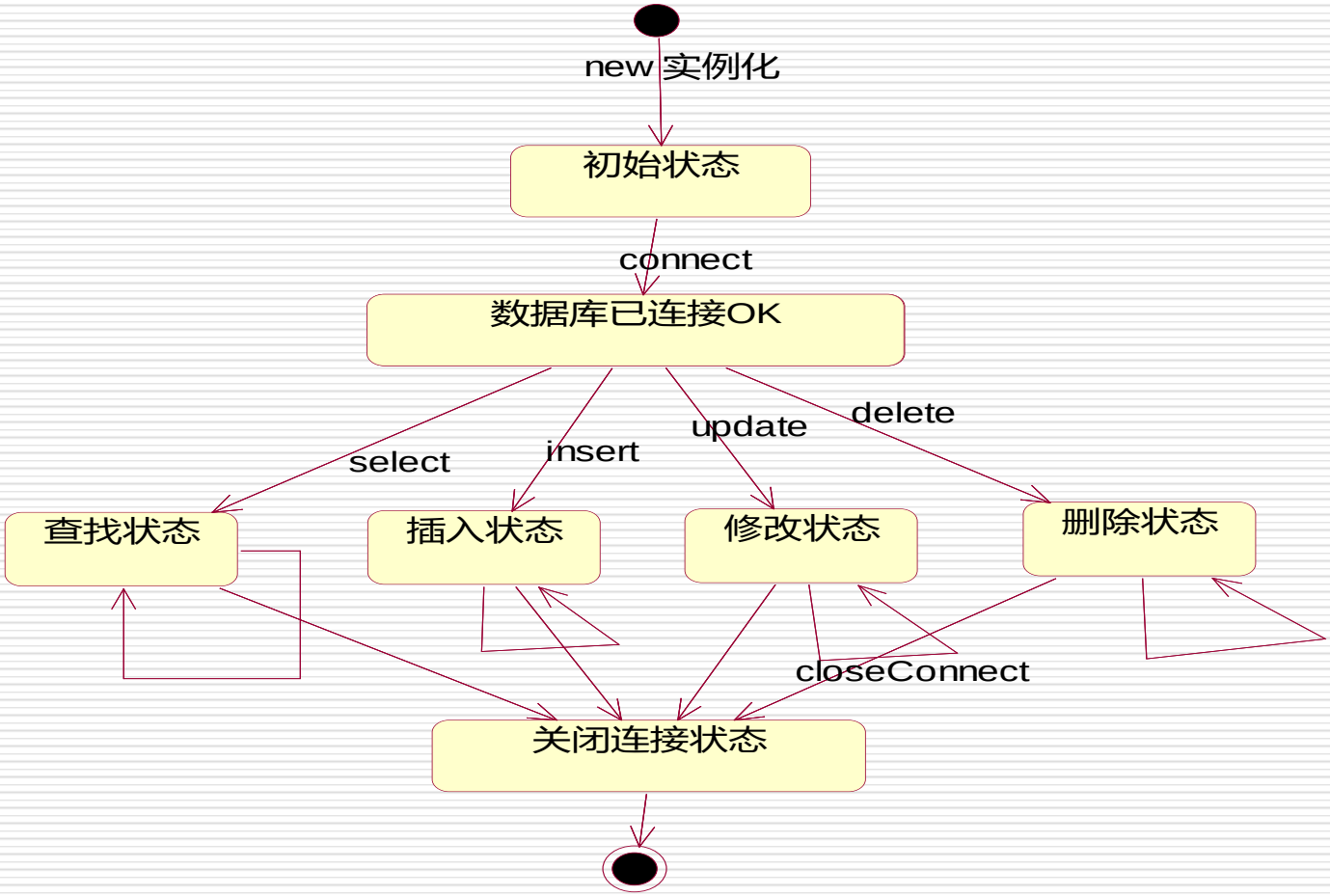
特别是描述单个对象的生命周期，反映式系统

打电话的状态图



电话机的状态图

数据库操作对象SqlOperators 生命周期状态图



Homework 2024-10-12

P 256

用面向对象分析方法，即UML方法

T1

T2

T3

T4