



# 第五章

## 需求建模方法与技术



# 课程内容

- **5.1 什么是模型**
- **5.2 结构化的需求建模方法**
- **5.3 面向对象的需求建模方法**



## 5.1 什么是模型

### ■ 模型的各种定义：

- 由某些人根据其目的而对事物进行的抽象描述。
- 根据实物、设计图或设想，按比例生成或其他特征制成的同实物相似的物体。
- 当一个数学结构作为某个形式语言（即包括常量符号、函数符号、谓词符号的集合）的解释时，称为模型。
- 为了理解事物而对事物作出的一种抽象，是对事物的一种无二义性的书面描述。



## 什么是模型 (续)

- 软件工程中的模型：

对客观世界的问题域进行抽象，并用某描述方法表示的结果成为模型。

- 在建模过程中应该注意问题域中有什么**对象**，应该选择什么样的**关系或动作**，然后用适当的模型表示。



## 什么是模型 (续)

■ 根据具体的建模要求和抽象的内容：

- 开发过程模型 (规约性)：瀑布式模型、增量式模型、螺旋式模型等
- 信息流模型 (描述性)：DFD、SADT等
- 设计模型：类图、功能层次图等
- 交互作用模型：实例图、交互作用图、时序图等
- 状态迁移模型：状态图、Petri网等
- 用于构造细节的原理模型：设计模式、实体关联图等
- 过程成熟度模型：CMM、SPICE
- 其他模型：可靠性模型、成本估算模型等



## 什么是模型 (续)

- 根据模型与客观世界的关系：
  - **描述性模型**：能真实和较完整地反映客观世界（如照片）
  - **规约性模型**：能用于创造新事物的规约（如需求模型）
  - **探测性模型**：是过渡性的，经常被修改而非最终决定的模型



## 什么是模型 (续)

- 例：一个建筑师画下一栋旧房子，然后决定对旧房子进行改造，并修改了所画的图。
- 软件工程中的模型有的是描述性模型，有的是规约性模型。
- 需求模型既是描述性模型，又是规约性模型。



## 什么是模型 (续)

### ■ 需求建模

根据待开发软件系统的需求，利用某种建模方法建立该系统的逻辑模型（需求模型或分析模型）

### ■ 目的

帮助软件开发人员检测软件需求的一致性、完全性、二义性和错误等





## 什么是模型 (续)

### ■ 建模方法应具备如下共同点

- 提供描述手段：包括记录什么内容、用什么符号来表达等
- 提供基本步骤：包括确定每一步的目的，要产生什么样的结果，每一步要注意哪些概念，完成该步工作需要掌握哪些必要的信息等



# 课程内容

- 5.1 什么是模型
- **5.2 结构化的需求建模方法**
- 5.3 面向对象的需求建模方法



## 5.2 结构化的需求建模方法

- 5.2.1 结构化分析方法SA
- 5.2.2 数据流图DFD
- 5.2.3 数据词典DD
- 5.2.4 实体关联图ERD



## 5.2 结构化的需求建模方法

- 5.2.1 结构化分析方法SA
- 5.2.2 数据流图DFD
- 5.2.3 数据词典DD
- 5.2.4 实体关联图ERD



## 5.2.1 结构化分析方法SA

- 结构化的分析方法SA(Structral Analysis): 一种传统的需求建模方法
  - 20世纪70年代中期提出: 用于数据处理和分析系统的功能
  - 20世纪90年代进行扩充: 在数据流图中加入了控制成分, 增加了控制流图CFD



## 结构化分析方法SA(续)

1. SA方法的基本思想
2. SA方法的描述手段
3. SA方法的分析步骤



## 结构化分析方法SA(续)

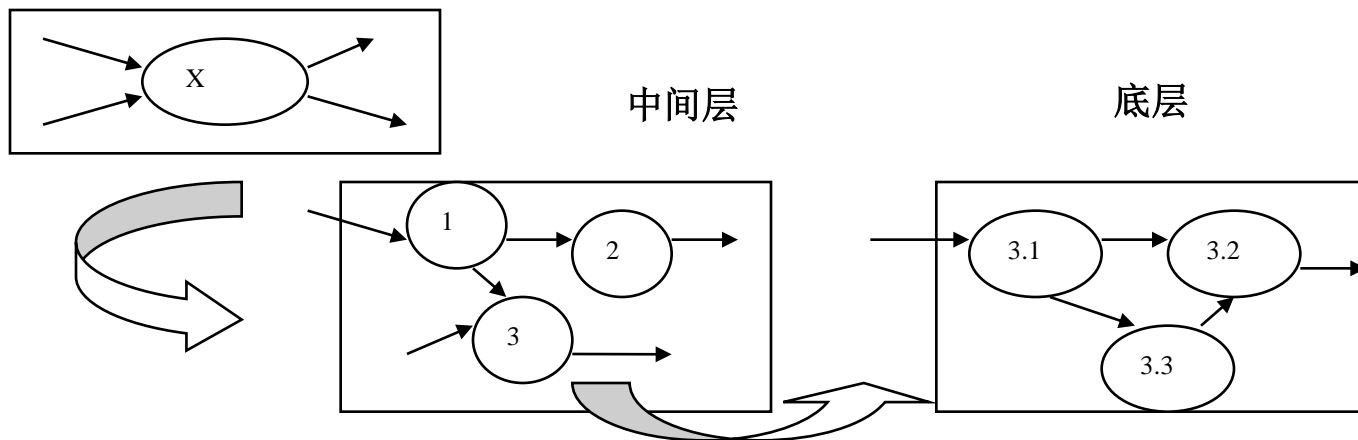
1. **SA方法的基本思想**
2. SA方法的描述手段
3. SA方法的分析步骤



## 结构化分析方法SA(续)

### ■ SA方法的基本思想：

按照由**抽象**到具体、逐层**分解**的系统分析方法来定义系统的需求







## 结构化分析方法SA(续)

### ■ SA方法的基本特点

- 表达问题时尽可能使用图形符号的方式
- 利用数据流图(Data Flow Diagram, DFD)来帮助理解问题，对问题进行分析
- 设计数据流图时只考虑**系统必须完成的基本功能**，不需要考虑如何具体地实现这些功能



## 结构化分析方法SA(续)

- SA方法的三个目标：
  - 描述用户需要
  - 建立创建软件设计的基础
  - 定义软件可被确认的一组需求



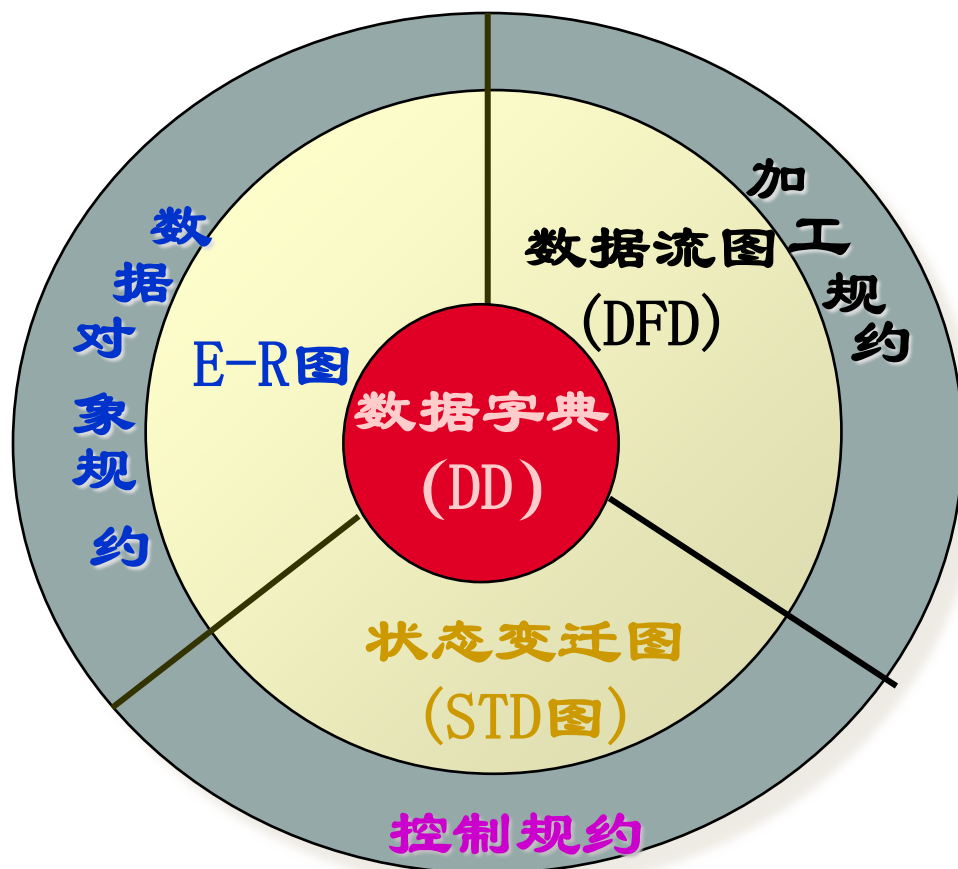
## 结构化分析方法SA(续)

1. SA方法的基本思想
2. **SA方法的描述手段**
3. SA方法的分析步骤



# 结构化分析方法SA(续)

## SA方法的描述手段





## 结构化分析方法SA(续)

### ■ 数据词典

— 模型核心，为每个数据元素提供详细的说明

### ■ E-R图(ERD)

— 表示数据对象以及相互的关系，用于数据建模

### ■ 数据流图(DFD)

— 主要说明系统由哪些部分组成，以及各部分之间的联系，用于功能建模

### ■ 状态变迁图(STD)

— 指明作为外部事件的结果，系统将如何动作，用于行为建模。



## 结构化分析方法SA (续)

1. SA方法的基本思想
2. SA方法的描述手段
3. **SA方法的分析步骤**



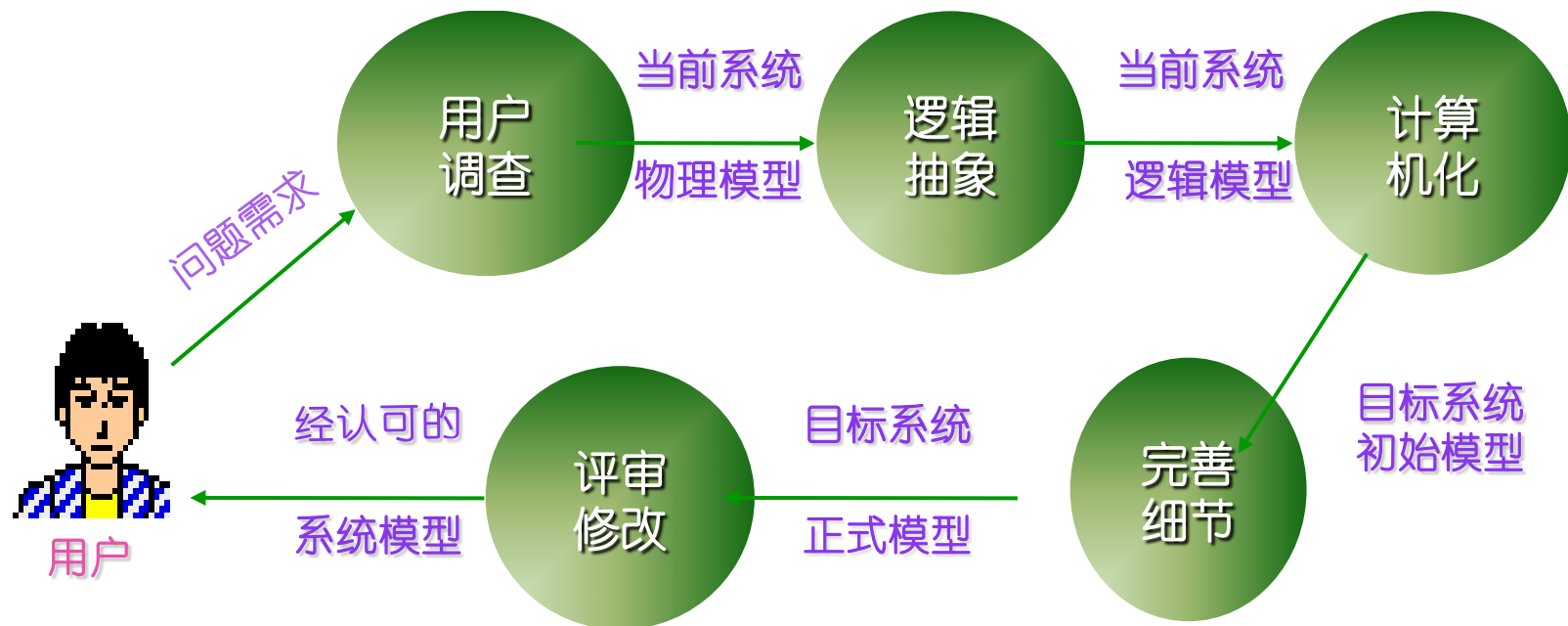
## 结构化分析方法SA (续)

- 当前系统：现实中已存在的人工系统
- 目标系统：待开发的计算机系统（主要是指软件系统）
- 当前系统与目标系统的关联：
  - 在功能方面应当是基本相同的
  - 在实现细节方面存在一定的差别



## 结构化分析方法SA (续)

### ■ SA方法分析的步骤







## 结构化分析方法SA (续)

|      | 逻辑模型                        | 物理模型              |
|------|-----------------------------|-------------------|
| 当前系统 | 描述重要的业务功能，无论系统是如何实施的        | 描述当前系统是如何在物理上实现的  |
| 目标系统 | 描述新系统的主要业务功能和用户需求，无论系统应如何实施 | 描述新系统是如何实施的（包括技术） |



## 结构化分析方法SA (续)

- (1) 理解和分析当前的现实环境，以获得当前系统的物理模型
- (2) 建立当前系统的逻辑模型
- (3) 建立目标系统的逻辑模型
- (4) 进一步完善目标系统的逻辑模型
- (5) 需求分析的验证



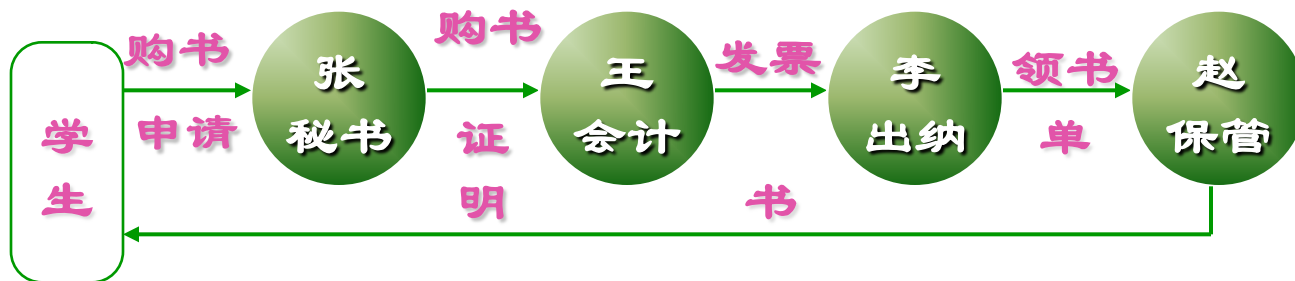
## 结构化分析方法SA (续)

- (1) 理解和分析当前的现实环境，以获得当前系统的物理模型
  - 在获取的需求信息的基础上，利用DFD将现实环境中的人工系统表达出来
  - 容易理解
  - 刚着手分析问题，还不清楚哪些是本质和非本质的因素，照搬现实比较合适



## 结构化分析方法SA (续)

- (1) 理解和分析当前的现实环境，以获得当前系统的物理模型



学生购买教材的物理模型



## 结构化分析方法SA (续)

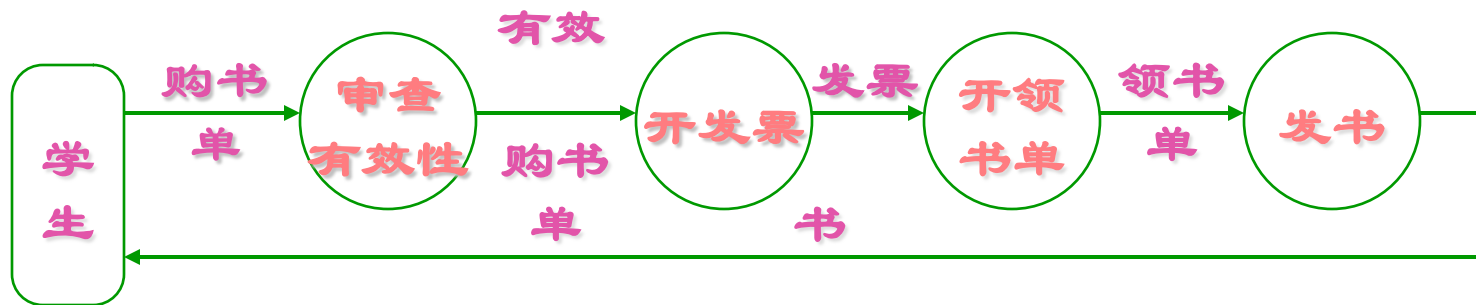
### (2) 建立当前系统的逻辑模型

- 从系统的物理模型中，除去非本质因素或一些“具体”因素，抽取现实系统的实质，抽象出当前系统的逻辑模型
- 反映当前系统必须满足的性质，即当前系统“做什么”



## 结构化分析方法SA (续)

### (2) 建立当前系统的逻辑模型



学生购买教材的逻辑模型



## 结构化分析方法SA (续)

### (3) 建立目标系统的逻辑模型(关键步骤)

— 分析目标系统与当前系统的差别，以当前系统的逻辑模型为基础，做些适当的修改和去掉所有的“具体”因素，建立目标系统的逻辑模型



## 结构化分析方法SA (续)

### (3) 建立目标系统的逻辑模型 (关键步骤)

#### ■ 方法一

- ① 确定当前系统逻辑模型的“改变范围”
- ② 把“改变范围”视为一个加工，并确定此加工的输入/输出数据流，当该加工比较抽象时，可进行逐层分解，然后画出各层DFD





## 结构化分析方法SA (续)

### (3) 建立目标系统的逻辑模型(关键步骤)

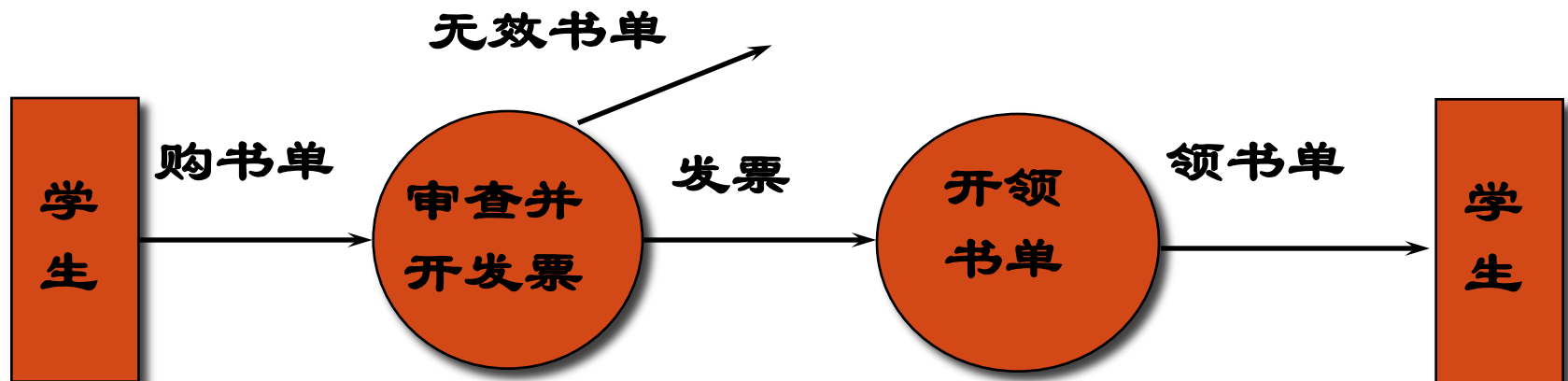
#### ■ 方法二

- ① 建立目标系统的顶层DFD和第一层DFD
- ② 参照当前系统的逻辑模型，去掉所有“具体”因素和细化各子系统



## 结构化分析方法SA (续)

### (3) 建立目标系统的逻辑模型 (关键步骤)



计算机教材管理系统的逻辑模型



## 结构化分析方法SA (续)

### (4) 进一步完善目标系统的逻辑模型

一对目标系统的逻辑模型进行细化、改进与优化

- 至今尚未说明的处理细节
- 某些需要的输入/输出格式或用户界面的说明
- 增加性能需求和其他一些约束限制等



## 5.2 结构化的需求建模方法

- 5.2.1 结构化分析方法SA
- **5.2.2 数据流图DFD**
- 5.2.3 数据词典DD
- 5.2.4 实体关联图ERD



## 5.2.2 数据流图DFD

1. 数据流图DFD
2. 分层的DFD
3. 画分层DFD的步骤



# 数据流图DFD(续)

## 1. 数据流图DFD

### ■ 定义

描述系统内部处理流程、表达软件系统需求模型的一种图形工具，亦即是描述系统中数据流程的图形工具。



# 数据流图DFD(续)

## 1. 数据流图DFD

### ■ 特点

- 图形描述简明、清晰、不涉及技术细节
- 可以描述原系统/新系统/子系统
- 便于理解、交流，是软件开发人员与用户之间进行交流的有效手段



# 数据流图DFD(续)

## 1. 数据流图DFD

### ■ 基本元素

- 数据流：由一组数据项组成的数据
- 加工：表示对数据进行的操作或变换
- 文件：存放数据的逻辑单位
- 源点和终点：表示数据的来源和最终去向

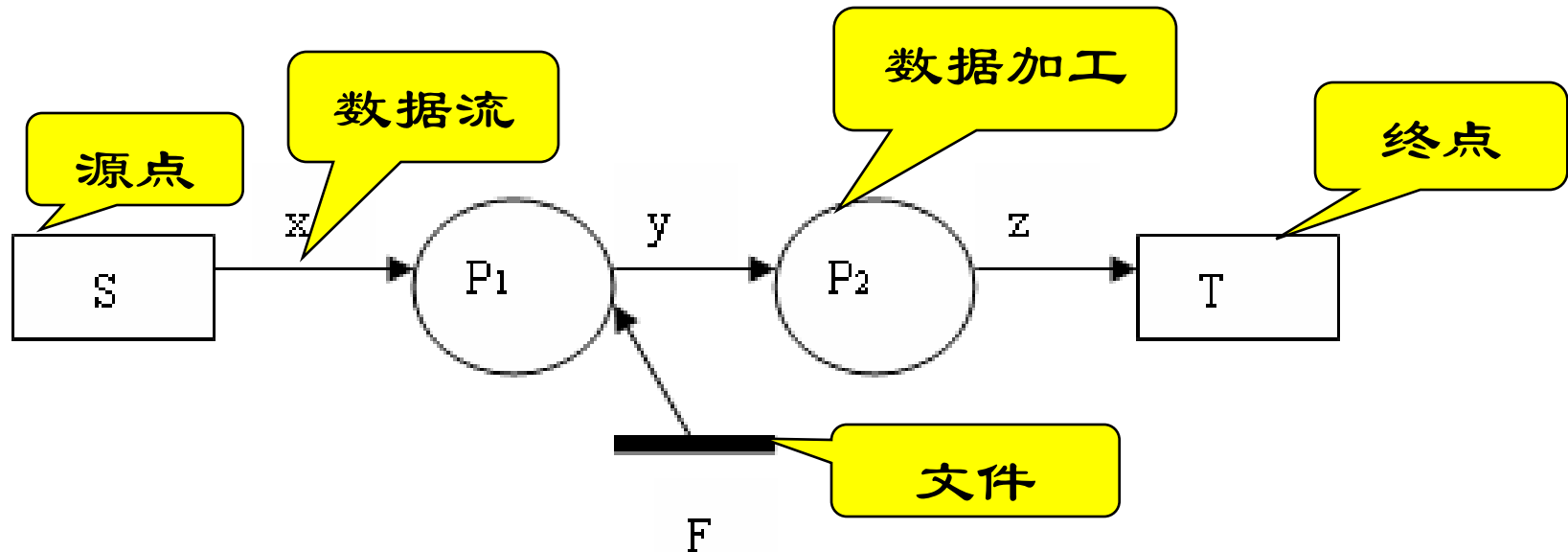




## 数据流图DFD(续)

### 1. 数据流图DFD

#### ■ DFD简例

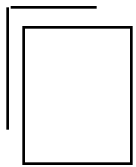




## 数据流图DFD(续)

### 1. 数据流图DFD

四种基本符号的另一种表示方法：



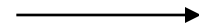
外部项 (S)



数据加工 (P)



数据存储 (D)



数据流 (F)



## 数据流图DFD(续)

### 1. 数据流图DFD—数据流

- 数据流是由一组数据项组成的数据，通常用带标识的有向弧表示
- 关于数据流的规则：
  - ① 数据流可以从加工流向加工，从源点流向加工，从加工流向终点，从加工流向文件，从文件流向加工。
  - ② 从加工流向文件或从文件流向加工的数据流可以不指定数据流名，但要给出文件名



## 数据流图DFD(续)

### 1. 数据流图DFD—数据流

#### ■ 关于数据流的规则：

- ③ 两个加工之间允许多股数据流
- ④ 每个数据流要有一个合适的名字，不能使用缺乏具体含义的词当作为数据流名。
- ⑤ 数据流经过一个加工后，其数据结构/数据含义/数据的顺序一定要有所变化，否则这个加工就没有意义了
- ⑥ 不能把控制流作为数据流。



## 数据流图DFD(续)

### 1. 数据流图DFD—加工(变换)

- 对数据进行的操作或变换称为加工，常用圆圈、椭圆等表示
- 关于加工的规则
  - ① 各加工与数据流或文件相连接，每一个加工必须有输入和输出，且输入和输出数据流要有所变化
  - ② 加工的符号分为两部分：
    - 标识部分用于标注**加工编号**，编号应具有唯一性
    - 功能描述部分用来写**加工名**，加工名应反映该加工的含义



## 数据流图DFD(续)

### 1. 数据流图DFD—加工(变换)




#### ■ 加工的命名方法

- 最高层的加工可以是软件系统的名字；
- 加工的名字最好由一个谓语动词加上一个宾语组成；
- 不能使用空洞或含糊的动词作为加工名；
- 当遇到不能合适命名的加工时，可以考虑将加工分解。



## 数据流图DFD(续)

### 1. 数据流图DFD—文件

- 文件是存放数据的逻辑单位，且通常用图形符号“”，“”和“”分别表示加工写文件，读文件和读写文件

- 关于文件的规则

- ① 在文件的图形符号中要给出文件名
- ② 文件的命名最好与文件中存放的内容相对应
- ③ 文件名可等同于数据流名
- ④ 为了避免DFD中出现交叉线，同一文件可以在多处画出



## 数据流图DFD(续)

### 1. 数据流图DFD—源点和终点

- 源点和终点用于表示数据的来源和最终去向，通常用图形方框表示。
- 源点和终点主要代表软件系统外的实体，包括人、组织、设备或者其他软件系统，它们不受系统的控制
- 源点和终点与系统有信息交流，从源点到系统的信息叫系统的输入；从系统到终点的信息称系统的输出
- 目的是为了帮助理解系统，不需要对它们进行描述
- 源点和终点可以在多处出现





# 数据流图DFD(续)

## 1. 数据流图DFD

### ■ 数据流图的作用

建立目标系统的DFD是一项十分重要的工作。因为建立的DFD是**系统开发**乃至**系统维护**的依据，是系统的重要文档之一。系统分析员要在详细调查中，在与用户的反复交流中修改DFD，力求新建DFD是正确的、准确的。

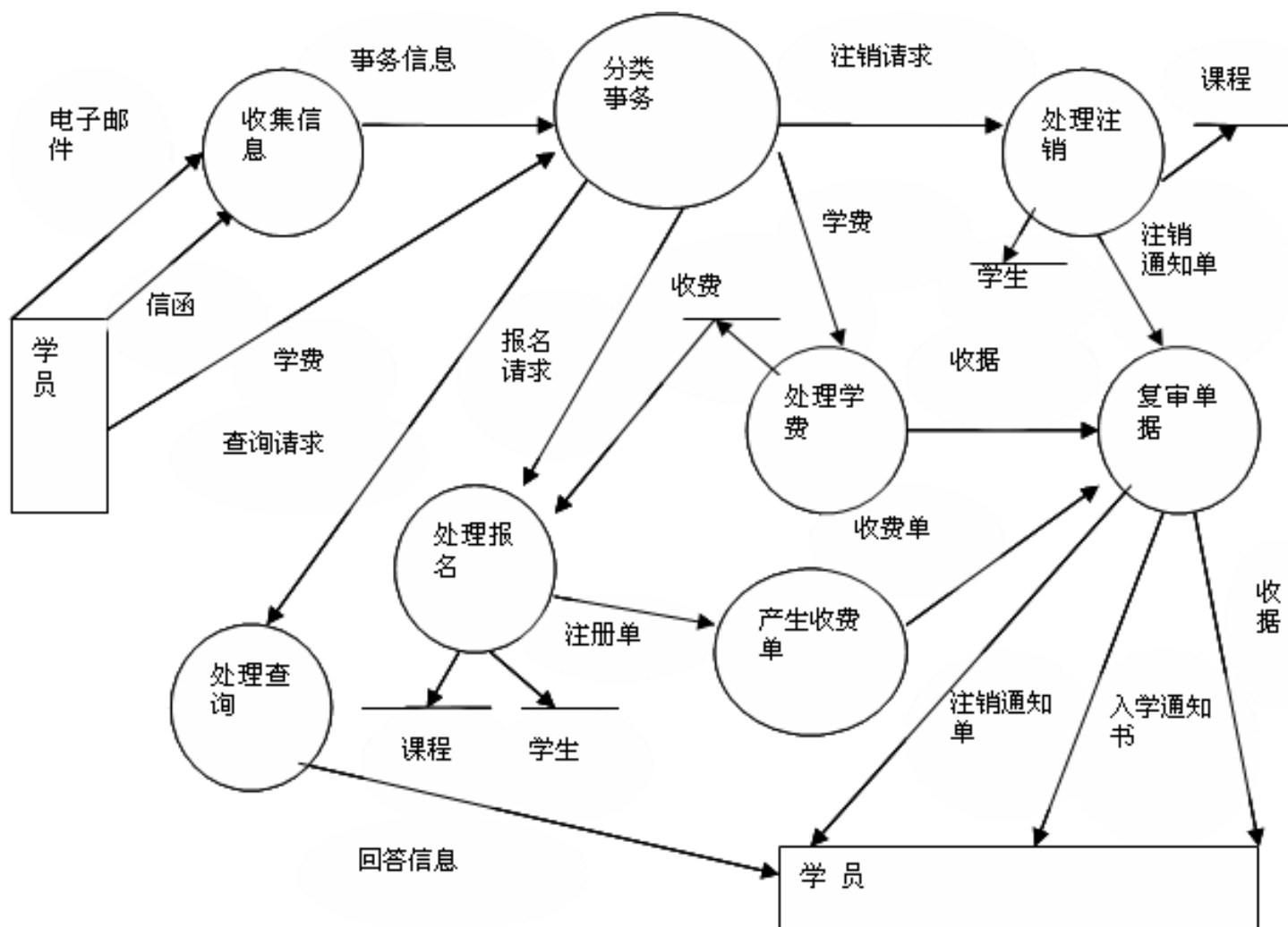


## 数据流图DFD(续)

### 例：4.1节中培训中心管理信息系统的DFD

培训中心的主要工作是学员提供课程培训服务，学员可以通过电子邮件、信函等报名，选修或注销课程，或询问课程计划等。培训中心收取一定的培训费用，学员可以用现金或支票形式支付。

该系统应具有记录和分类由电子邮件或信函表达的信息，处理报名、询问、注销和付款，以及输出回答信息的功能。





## 数据流图DFD(续)

### 2. 分层的DFD

- 分层的方法体现了“抽象”的原则，有目的地逐步增加细节
- 一个比较抽象的过程可以被展开为一个子过程更加具体的DFD图
- 可以在不同的抽象层次上进行系统的描述
- 目的：
  - 控制复杂度
  - 有助于理解



## 数据流图DFD(续)

### 2. 分层的DFD

#### ■ DFD的层次结构：

- 顶层：用于注明系统的边界，即系统的输入/输出
- 中间层：描述加工的分解
- 底层：由一些不能再分解的加工（基本加工）组成



## 数据流图DFD(续)

### 2. 分层的DFD

- **顶层图(上下文图)：**将整个系统看做一个加工，这个加工实现系统的所有功能，是系统的最高抽象
  - 上下文图中存在且仅存在一个方法，表示整个系统
  - 上下文图中需要表示出所有和系统交互的外部实体，并描述交互的数据流，包括系统输入和系统输出
  - 上下文图中不会出现文件实例
  - 非常适合于描述系统的应用环境、定义系统的边界



## 数据流图DFD(续)

### 2. 分层的DFD

- N层图：对N层图的过程分解后产生的子图称为N+1层图( $N > 0$ )，过程分解是可以持续进行的，直至最终产生的子图
- 位于上下文图下面一层的数据流图是整个系统的功能概图，是对系统的第一次功能分解
- 功能概图往下的子图上通常不显示外部实体



# 数据流图DFD(续)

## 2. 分层的DFD

### ■ 画完整的分层的DFD需注意的几个问题

#### ① 应区别于流程图

- DFD注重于数据在系统中的流动，对加工间的多股数据流不需考虑前后次序问题
- 流程图则需考虑对数据处理的次序和具体细节





## 数据流图DFD(续)

### 2. 分层的DFD

#### ■ 画完整的分层的DFD需注意的几个问题

##### ② DFD的完整性问题

- 加工产生的输出流没有输出到其他任何加工或外部实体
- 某些加工有输入但不产生输出



## 数据流图DFD(续)

### 2. 分层的DFD

#### ■ 画完整的分层的DFD需注意的几个问题

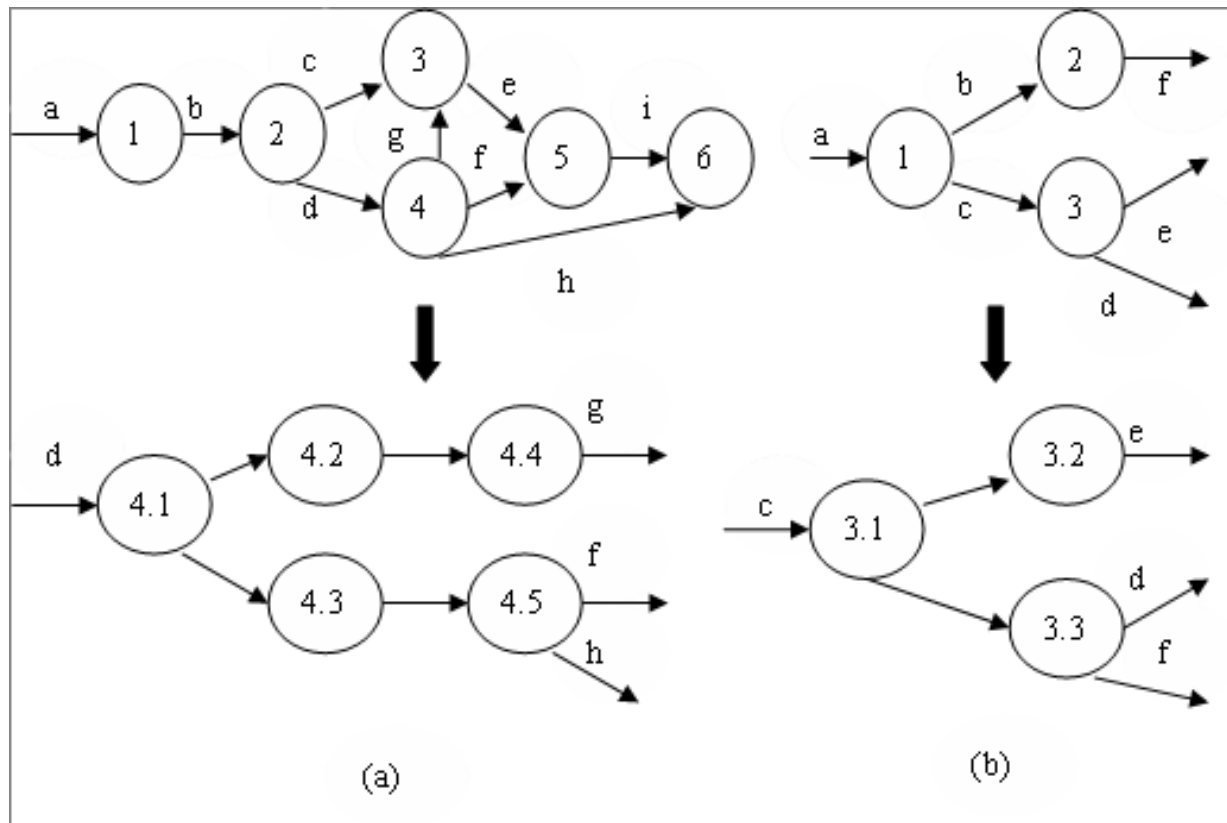
##### ③ DFD的一致性问题的

父图与子图的平衡问题，是指父图中某加工的输入/输出与分解该加工的子图的输入/输出必须完全一致，即输入/输出应该相同



## 数据流图DFD(续)

### 2. 分层的DFD——一致性问题的示例





## 数据流图DFD(续)

### 2. 分层的DFD

#### ■ 画完整的分层的DFD需注意的几个问题

##### ④ 在分层DFD中文件的表示

- 在抽象层中未用到的文件可以不表示，在子图中用到的文件表示在该子图中
- 在抽象层中表示的文件，则应该在相应的某些子图中表示
- 当文件共享于某些加工之间时，则该文件必须表示出来



# 数据流图DFD(续)

## 2. 分层的DFD

### ■ 画完整的分层的DFD需注意的几个问题

#### ⑤ 分解层次的深度

- 分解最好不超过7或8层，尽量减少分解层次；
- 分解应根据问题的逻辑特性进行，不能硬性分解；
- 每个加工被分解为子加工后，子图中的子加工数不要太多，通常为7~10个；
- 上层可分解快些，下层应该慢些，因为上层比较抽象，易于理解



## 数据流图DFD(续)

### 2. 分层的DFD

#### ■ 画完整的分层的DFD需注意的几个问题

##### ⑤ 分解层次的深度

- 分解要均匀，即避免在一张DFD中，有些已是基本加工，另外一些还要被分解为多层；
- 分解到什么程度才能到达底层DFD呢？



## 数据流图DFD(续)

- 分解到什么程度才能到达底层DFD呢？
  - 所有过程都已经被简化为一个选择、计算或数据库操作
  - 所有数据存储都仅仅表示了一个单独的数据实体
  - 用户已经不关心比子图更为细节的内容，或者子图的描述已经详细得足以支持后续的开发活动
  - 每一个数据流都已经不需要进行更详细的切分，以展示对不同数据的不同处理方式
  - 系统的每一个最底层菜单选项都能在子图中找到独立的加工
  - 每一个业务表单、事务、计算机的屏幕显示和都已经被表示为一个单独的数据流



## 数据流图DFD(续)

### 3. 画分层DFD的步骤

- ① 确定软件系统的输入/出数据流、源点和终点;
- ② 将基本系统模型加上源点和终点构成顶层DFD;
- ③ 确定系统的主要信息处理功能, 按此将整个系统分解成几个加工环节(子系统), 确定每个加工的输入和输出数据流, 以及与这些加工有关的数据存储
- ④ 根据自顶向下, 逐层分解的原则, 对上层图中全部或部分加工环节进行分解





## 数据流图DFD(续)

### 3. 画分层DFD的步骤

- ⑤ 重复第4步，直到逐层分解结束
- ⑥ 对图进行检查和合理布局，主要检查分解是否恰当、彻底，DFD中各层是否有遗漏、重复、冲突之处，各层DFD及同层DFD之间关系是否合理，及命名、编号是否确切、合理等，对错误和不当之处进行修改
- ⑦ 和用户进行交流，在用户完全理解数据图的内容的基础上征求用户的意见



## 数据流图DFD(续)

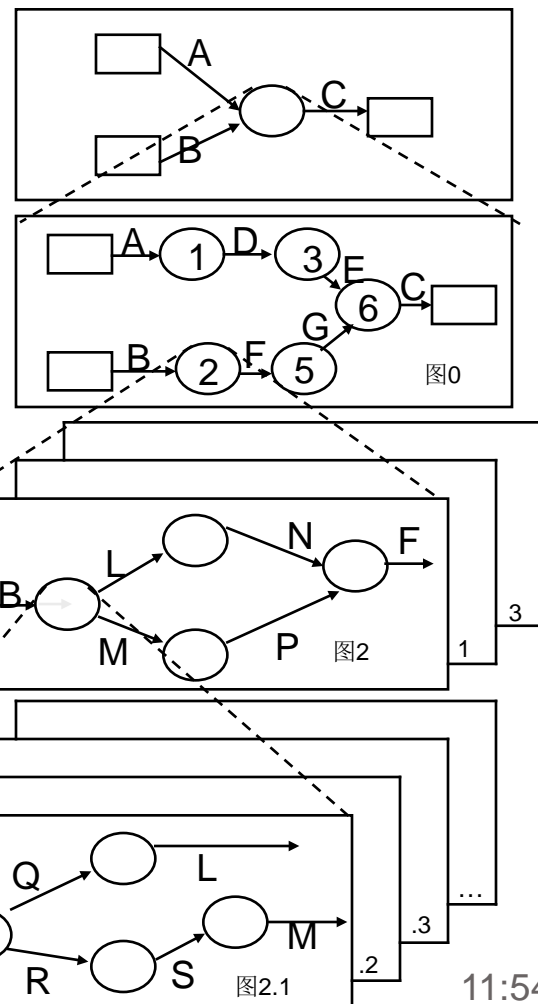
### 3. 画分层DFD的步骤 **A**顶层图

绘制数据流图过程示意图

**B**一层图

**C**二层图

**D**三层图





## 数据流图DFD(续)

### 3. 画分层DFD的步骤

#### ■ 画DFD的主要原则

- 明确系统边界
- 自顶向下逐层扩展
- 合理布局
- 数据流绘制过程，即系统的逻辑模型的形成过程，必须始终与用户密切接触，详细讨论，不断修改，也要和其他系统建设者共同商讨以求一致意见



## 数据流图DFD(续)

### 3. 画分层DFD的步骤

#### ■ 画每张DFD时，应遵循的准则

- 将所有软件的输入/出数据流用一连串加工连接起来；
- 当加工需要用到共享和暂存数据时，设置文件及其标识；
- 分析加工的内部，如果加工还比较抽象或其内部还有数据流，则需将该加工进一步分解，直至到达底层图；
- 为所有的数据流命名；
- 为所有加工命名编号，编号从DFD第一层开始（子图的加工编号=图号+小数点+局部顺序号）。



## 数据流图DFD(续)

### 3. 画分层DFD的步骤

#### ■ 在画DFD时还应注意的情况

- 画图时只考虑如何描述实际情况，不要急于考虑系统应如何启动，如何工作，如何结束等与时间序列相关的问题；
- 画图时可暂不考虑一些例外情况，如出错处理等；
- 画图的过程是一个重复的过程，一次性成功可能性较小，需要不断地修改和完善。



## 数据流图DFD(续)

### 例1：银行取款系统

#### 简单银行取款应用描述

储户将填好的取款单、存折交银行，银行做如下处理：

- ①审核并查对帐目，将不合格的存折、取款单退回储户，合格的存折、取款单送取款处理。
- ②处理取款修改帐目，将存折、利息单、结算清单及现金交储户，同时将取款单存档。



## 数据流图DFD(续)

第一步：画出关联数据图

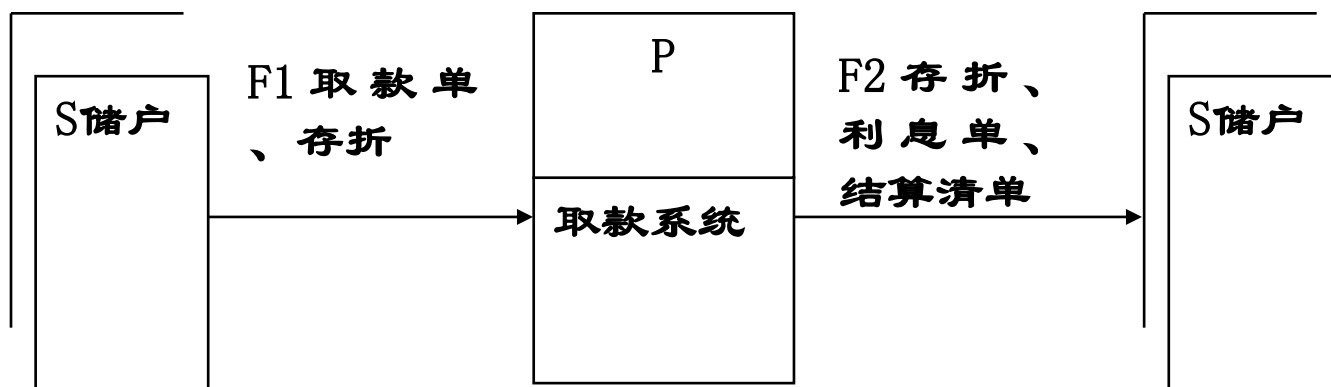
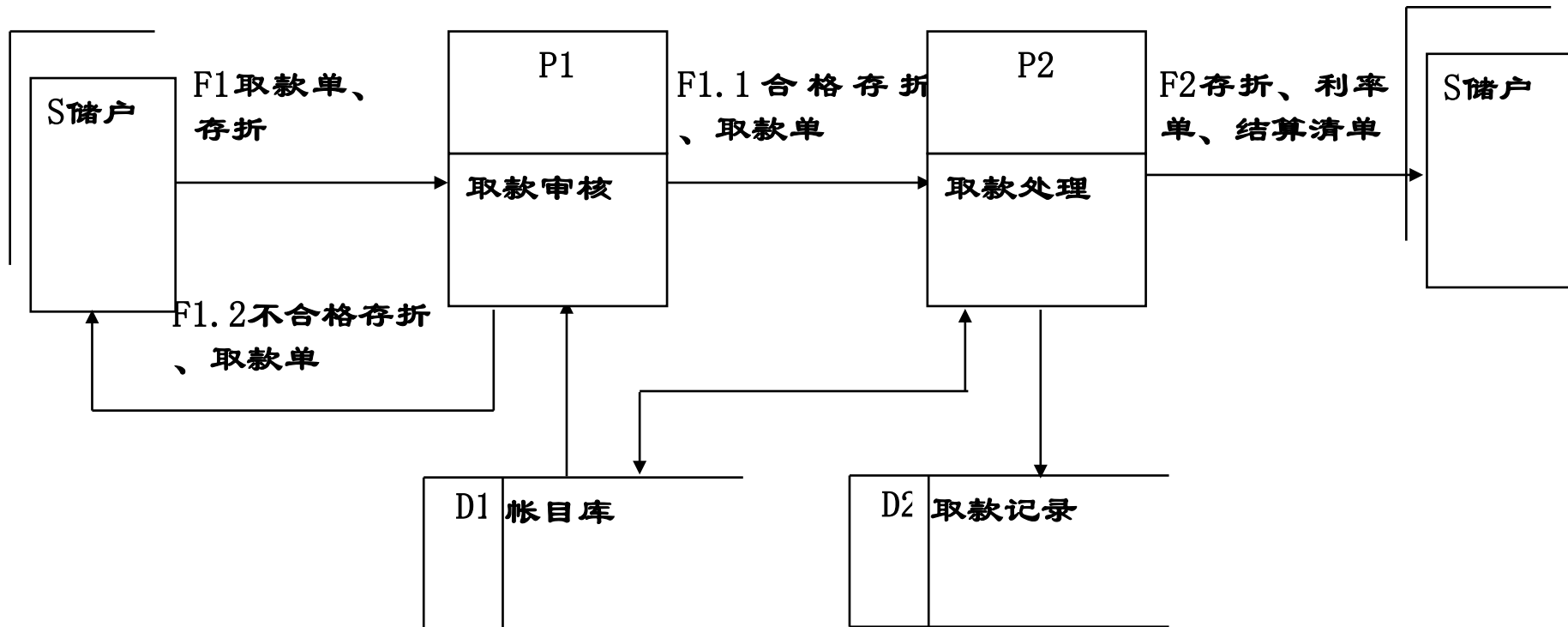


图 银行取款系统关联图



## 数据流图DFD(续)

第二步：逐层分解加工，画出下层DFD







## 数据流图DFD(续)

- 例2：某医院拟开发一个分布式患者监护系统PMS。PMS将用于监视病房中每个患者的重要生理信号（如体温、血压、脉搏信号等），并能定时更新和管理患者的病历。此外，当患者的生理信号超过医生规定的安全范围时，系统能立即通知护理人员，并且护理人员在需要时可随时通过系统产生某患者有关报告。



## 数据流图DFD(续)

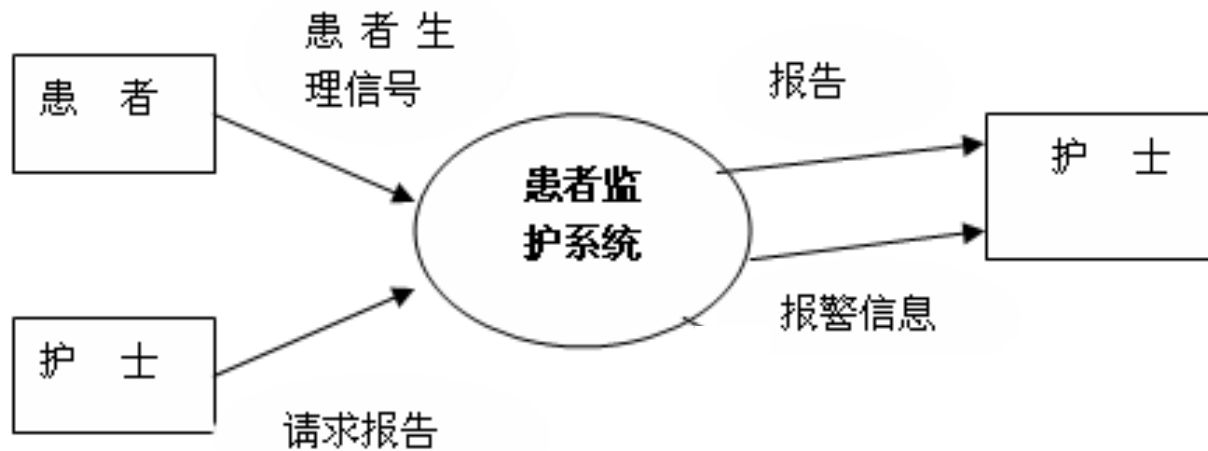
PMS的主要功能为：

- 通过一个病床监视器实现本地监测，以获得患者的生理信号。
- 在护士办公室实现中央监测
- 更新和管理患者病历
- 产生患者情况的报告以及报警信息



## 数据流图DFD(续)

### 第0层数据流图

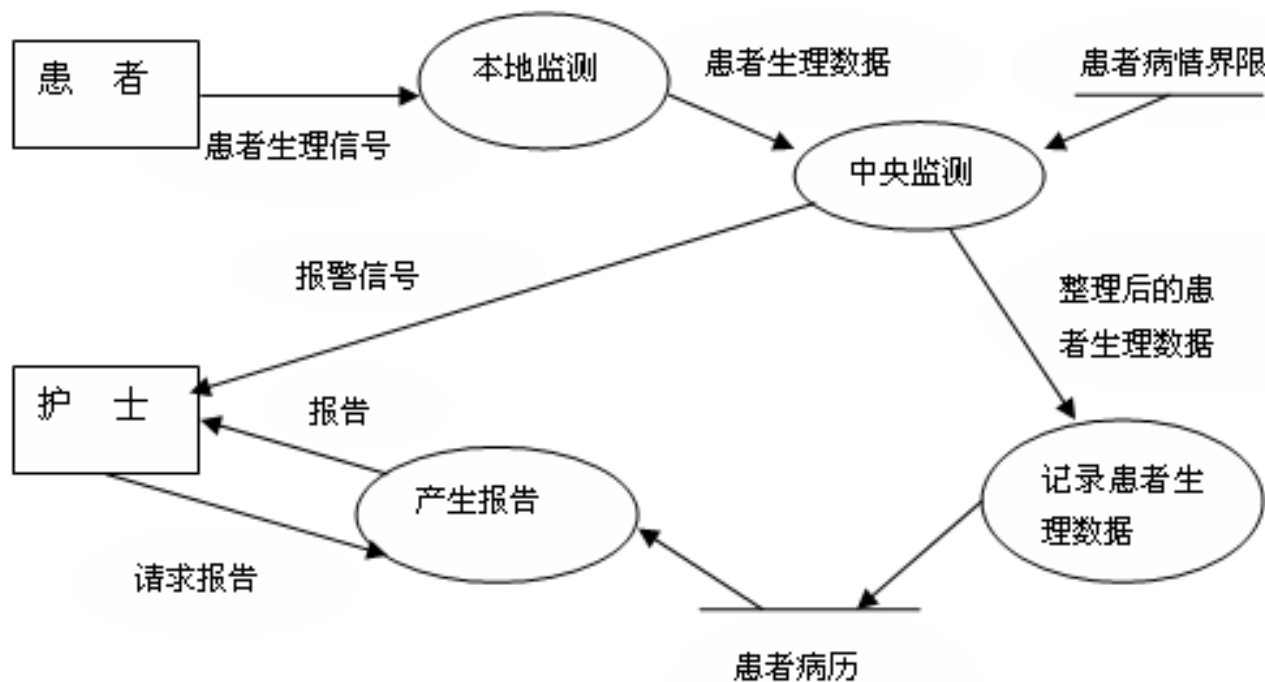


(a)



## 数据流图DFD(续)

### 第1层数据流图

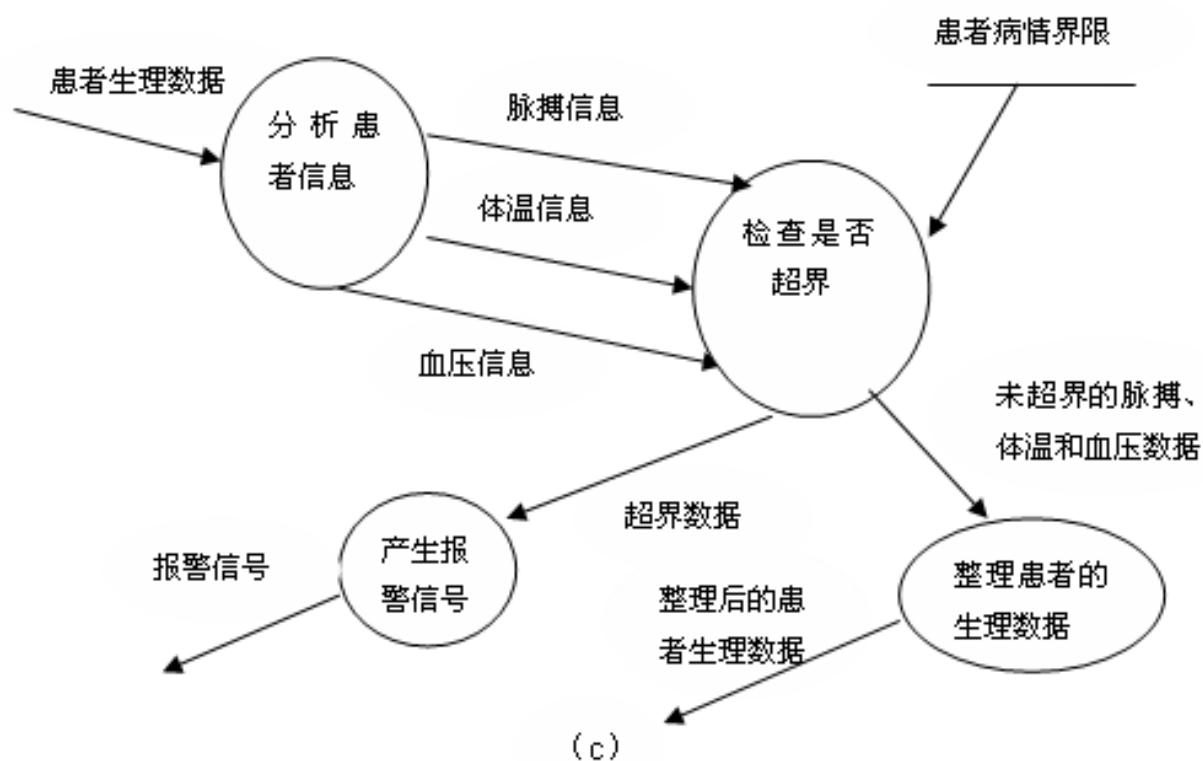


(b)



## 数据流图DFD(续)

### 第2层数据流图





## 数据流图DFD(续)

### 例3：图书预定系统

书店向顾客发放订单，顾客将所填订单交由系统处理，系统首先依据图书目录对订单进行检查并对合格订单进行处理，处理过程中根据顾客情况和订单数目将订单分为优先订单与正常订单两种，随时处理优先订单，定期处理正常订单。最后系统根据所处理的订单汇总，并按出版社要求发给出版社。



## 5.2 结构化的需求建模方法

- 5.2.1 结构化分析方法SA
- 5.2.2 数据流图DFD
- **5.2.3 数据词典DD**
- 5.2.4 实体关联图ERD



## 5.2.3 数据词典DD

- 数据词典是由DFD中所有元素的“严格定义”组成。
- 作用：为DFD中出现的每个元素提供详细的说明，即DFD中每个数据流名、文件名和加工名都在数据词典中应有一个条目以定义相应的含义。
- 由数据词典和数据流图可构成软件系统的逻辑模型（需求模型）





## 数据词典DD (续)

### ■ 数据词典中的条目类型：

- ① 数据流条目：用于定义数据流
- ② 文件条目：用于定义文件
- ③ 加工条目：用于说明加工



## 数据词典DD (续)

### ■ 数据词典中的基本元素和含义：

| 符 号  | 含 义       | 说 明                                |
|--|-----------|------------------------------------|
| =  | 表示定义为     | 用于对=左边的条目进行确切的定义                   |
| +  | 表示与关系     | $X=a+b$ 表示X由a和b共同构成                |
| $\begin{matrix} [   ] \\ [ , ] \end{matrix}$ | 表示或关系     | $X=[a b]$ 与 $X=[a,b]$ 等价，表示X由a或b组成 |
| ( )  | 表示可选项     | $X=(a)$ 表示a可以在X中出现，也可以不出现          |
| { }  | 表示重复      | 大括号中的内容重复0到多次                      |
| $m\{ \}n$                                    | 表示规定次数的重复 | 重复的次数最少m次，最多n次                     |
| “ ”  | 表示基本数据元素  | “ ” 中的内容是基本数据元素，不可再分               |
| ..   | 连接符       | $month=1..12$ 表示month可取1—12中的任意值   |
| * *  | 表示注释      | 内容为注释信息                            |



## 数据词典DD (续)

### ■ 数据词典中的基本元素和含义 一示例

| 定义  | 说明   |
|---|--|
| <b>telephone no. = [ local extension  <br/>outside no.   0 ]</b><br><b>local extension = 3{0-9}3</b><br><b>outside no. = 9 + [ service code  <br/>domestic no. ]</b><br><b>service code = [ 110   120  ... ]</b><br><b>domestic no. = (area code ) + local<br/>number</b><br><b>area code = 3{0-9}4</b><br><b>local number= 8{0-9}8</b> | 电话号码可能是内线、外线或者转接主机（拨0）<br><br>内线号码是3位数字<br>外线要先拨9，然后再拨特服号码或普通电话号码<br><br>特服号码有110、120、...<br>普通电话号码为可选的区号加本地号<br><br>区号是3到4位数字<br>本地号是8位数字 |



## 数据词典DD (续)

### ■ 数据流条目

数据流在数据流图中主要用于说明数据结构在系统中的作用和流动方向，因此数据流也被称作“流动的数据结构”。数据字典中数据流条目应包括以下几项主要内容：**数据流名称、数据流别名、说明、数据流来源、数据流流向、数据流组成和数据流量**等。

数据流名：

数据流别名：

说明：简要介绍作用即它产生的原因和结果。

数据流来源：即该数据流来自何方。

数据流流向：去向何处。

数据流组成：数据结构。

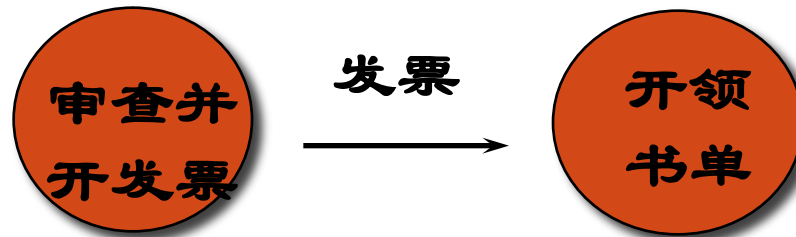
数据量流量：数据量、流通量。



## 数据词典DD (续)

### ■ 数据流条目

例



数据流名：发票

说明：用作学生已付书款的依据

数据流来源：来自加工“审查并开发票”

数据流去向：流向加工“开领书单”。

数据流组成：学号+姓名+书号+单价总价+书费合计



## 数据词典DD (续)

### ■ 数据流条目

例：工资系统中的**出勤表数据流**

数据流名称：出勤表

数据流别名：无

说明：由人事部门每月月底上报的职工考勤统计数字

数据流来源：人事部门

数据流流向：加工1.1.1(统计出勤、请假及旷工时数)

数据流组成：出勤表 = 年份 + 月份 + 职工号 + 出勤时数 +  
病假时数 + 事假时数 + 旷工时数

数据流量：1份/月



## 数据词典DD (续)

### ■ 数据流条目

- 在数据流条目中数据结构的定义采用简单的形式符号方式。

如：订票单 = 顾客信息 + 订票日期 + 出发日期 + 航班号 + 目的地

- 对于复杂的数据流，可采用自顶向下逐步细化的方式定义数据项。

如：顾客信息 = 姓名 + 性别 + 身份证号 + 联系电话



## 数据词典DD (续)

### ■ 数据流条目

- 当某些数据项是几个不同的数据流的公用数据项时，可将它们列为专门的数据项条目。

如：教室 = 101| 102|.....

航班号 = Mu712| Mu814| .....





## 数据词典DD (续)

### ■ 数据流条目

数据流图中每个数据结构都是由若干个数据项构成的。数据字典的数据项条目中应包含的主要内容有：  
**数据项名称、数据项别名、说明、类型、长度、取值范围及含义等。**



## 数据词典DD (续)

### ■ 数据流条目

例：出勤表中的**职工号****数据项**在数据字典中的条目描述为

数据项名称：职工号

数据项别名：employee\_no

说明：本单位职工的惟一标识

类型：字符串

长度：6

取值范围及含义：1-2位(00..99)为部门编号：3-6位  
(XX0001..XX9999)为人员编号



## 数据词典DD (续)

### ■ 数据流条目

当所有出现在DFD中的数据流都给予定义后，也可以表格的形式汇录每一数据项，对出现在数据流中的数据项进行汇总。

如：

| 标识符 | 类型 | 长度 | 中文名称 | 来 源 | 去向 |
|-----|----|----|------|-----|----|
| JS  | 整型 | 3  | 教室   | 教务处 | 学员 |



## 数据词典DD (续)

### ■ 文件条目

文件是数据流图中数据结构的载体。数据字典的数据文件条目中应包含的主要内容有：**数据文件名称、说明、数据文件组成、组织方式、存取方式、存取频率等。**



## 数据词典DD (续)

### ■ 文件条目

例：工资系统中的**职工工资档案文件**在数据字典中的条目描述为

数据文件名称：工资档案

说明：单位职工的基本工资、各项津贴及补贴信息

数据文件组成：职工号+国家工资+国家津贴+职务津贴+职龄津贴+交通补贴+部门补贴+其他补贴

组织方式：按职工号从小到大排列

存取方式：顺序

存取频率：1次/月



## 数据词典DD (续)

### ■ 加工条目

加工条目主要描述加工的处理逻辑或“做什么”。  
加工条目中应包含的主要内容有：**数据加工名称、加工编号、说明、输入数据流、输出数据流、加工逻辑等。**



## 数据词典DD (续)

### ■ 加工条目

例： 4.1培训中心管理信息系统中的“处理报名”加工：

加工名称：处理报名

输入数据流：报名请求+费用

输出数据流：注册单+课程+学生

加工逻辑：

- 根据报名要求查询收费标准文件，确定相应费用。
- 学生注册
- 根据选修课程登录课程统计文件
- 产生注册单等



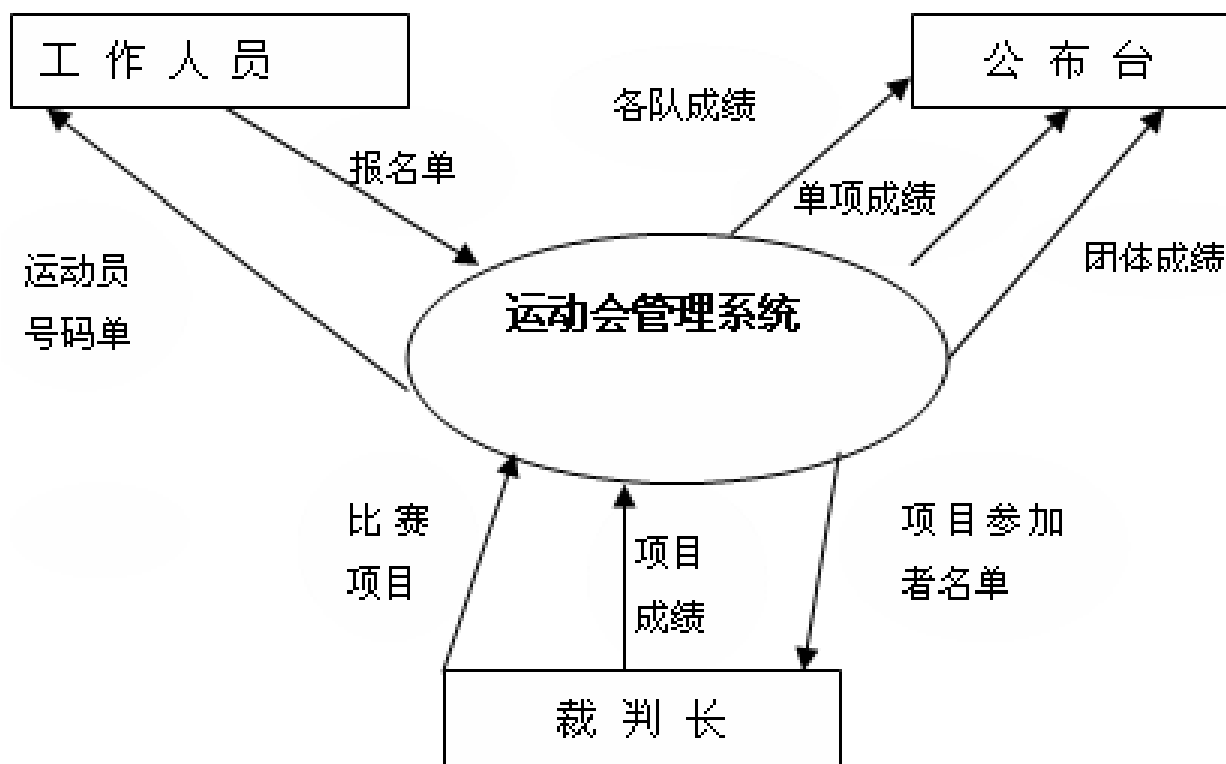
例：某学校拟开发一个运动会管理系统：

- ① 确定运动会的举办时间和地点，设置哪些项目，报名时间等
- ② 确定一些限制规定，如每人最多可参加几个项目，每个项目每队最多可由多少人参加，取前几名，打破单项比赛记录后的处理等。
- ③ 由各参加队提供报名表后，需给每个运动员编号，并统计每个项目的参加人数及名单，最后根据每个项目的参加人数等具体情况排出比赛日程。
- ④ 在运动会期间不断接受各项比赛的成绩，及时公布单项名次，累计团体总分。
- ⑤ 比赛结束后，公布最终的团体名次。





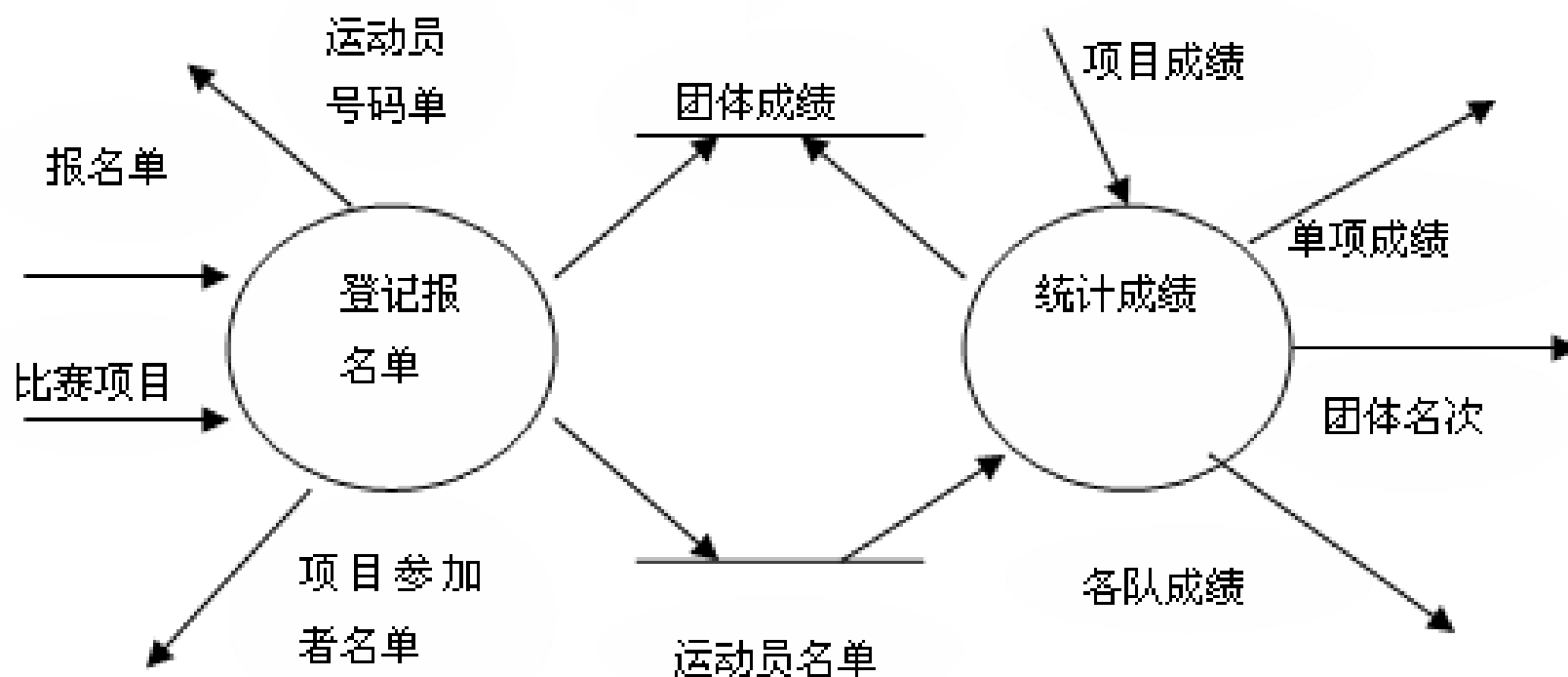
## 1. 建立系统的DFD



第0层数据流图



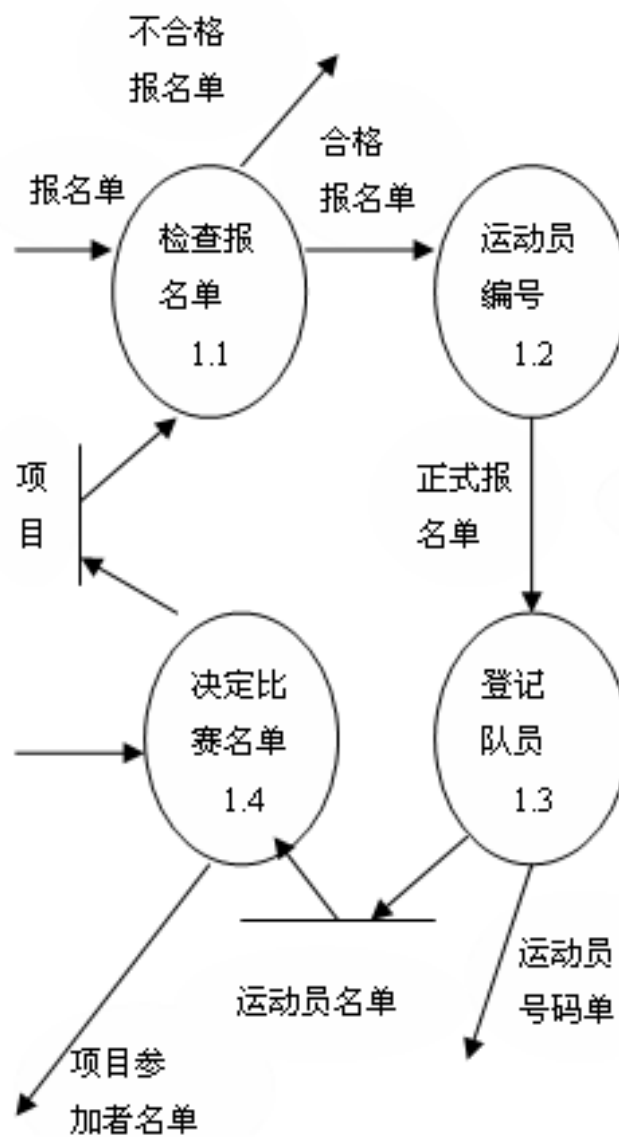
## 1. 建立系统的DFD



第1层数据流图



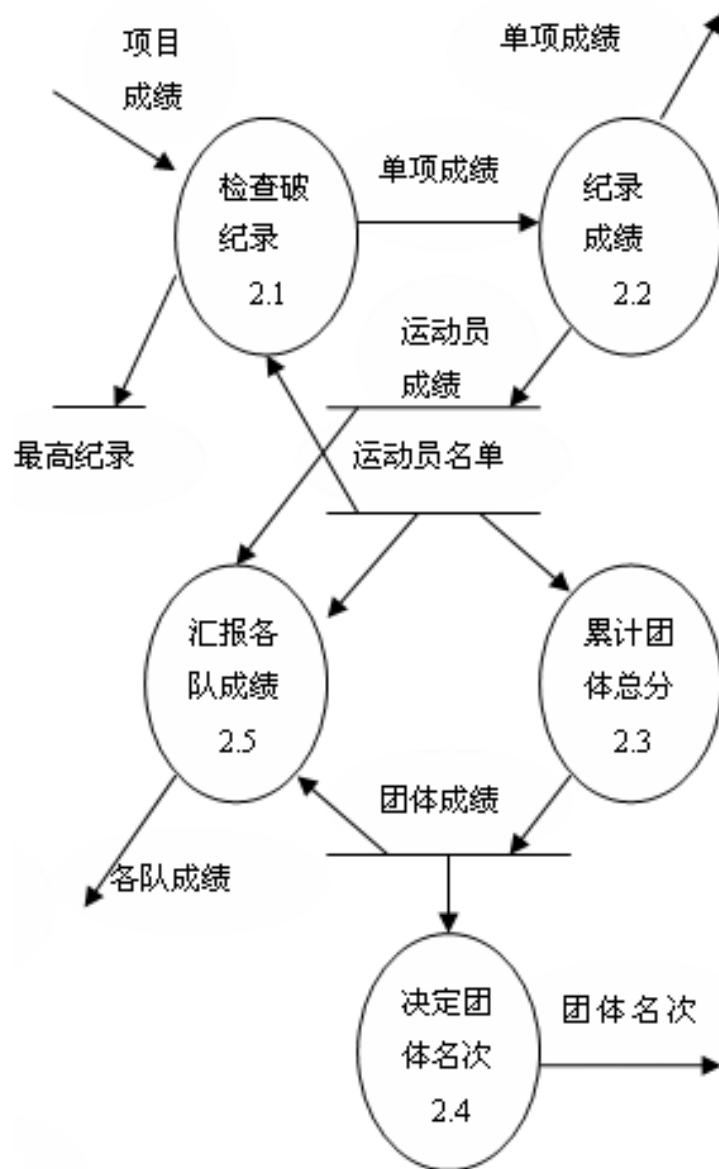
## 1. 建立系统的DFD



第2层数据流图



## 1. 建立系统的DFD



第2层数据流图



## 2. 数据词典说明

### (1) 数据流条目

| 数据流名 | 标识符  | 组成                           |
|------|------|------------------------------|
| 项目成绩 | XMCJ | 项目名+ {运动员号 + 成绩}             |
| 单项名次 | DXMC | 项目名+ {名次+ 运动员号+ 成绩+ 破记录}     |
| 单项成绩 | DXCJ | 项目名+ {运动员号+ 成绩+ 破记录}         |
| 各队成绩 | GDCJ | 队名+ 总分+ {运动员号+ 项目名+ 成绩+ 破记录} |



## 2. 数据词典说明

### (2) 汇总后的数据项

| 数据项名 | 值       | 位数 |
|------|---------|----|
| 项目名  | 字符串     | 8  |
| 姓名   | .....   | 4  |
| 运动员号 | 1~ 4999 | 4  |
| 破记录  | 1   0   | 1  |
| 成绩   | 0 ~ 999 | 4  |
| 总分   | 正整数     | 4  |



## 2. 数据词典说明

### (3) 文件条目

| 文件名   | 文件标识  | 组成                       | 组织          |
|-------|-------|--------------------------|-------------|
| 团体成绩  | TTCJ  | 项目名+ { 队名 + 总分 }         | 按队名拼音顺序递增排列 |
| 运动员名单 | YDYMD | 队名+运动员号+ 姓名+<br>{ 项目名 }  | 按运动员号递增排序   |
| 运动员成绩 | YDYCJ | 运动员号+ { 项目名+成绩<br>+破纪录 } | 同上          |



## 2. 数据词典说明

### (4) 加工条目

加工名：记录成绩 编号：2.2 启动条件：收到单项成绩

加工说明：

- a. 取单项成绩
- b. 根据项目名、运动员号将运动员成绩记录到运动员成绩文件中
- c. 处理破纪录情况
- d. 建立单项成绩表

执行频率：100项/天





### 3. 其他补充材料

#### (1) 表格输入/输出格式说明

如各队成绩的输出格式：

| 表头 |     |    |     |
|----|-----|----|-----|
| 队名 |     | 总分 |     |
| 姓名 | 项目名 | 成绩 | 破纪录 |
|    |     |    |     |



## 5.2 结构化的需求建模方法

- 5.2.1 结构化分析方法SA
- 5.2.2 数据流图DFD
- 5.2.3 数据词典DD
- **5.2.4 实体关联图ERD**



## 5.2.4 实体关联图ERD

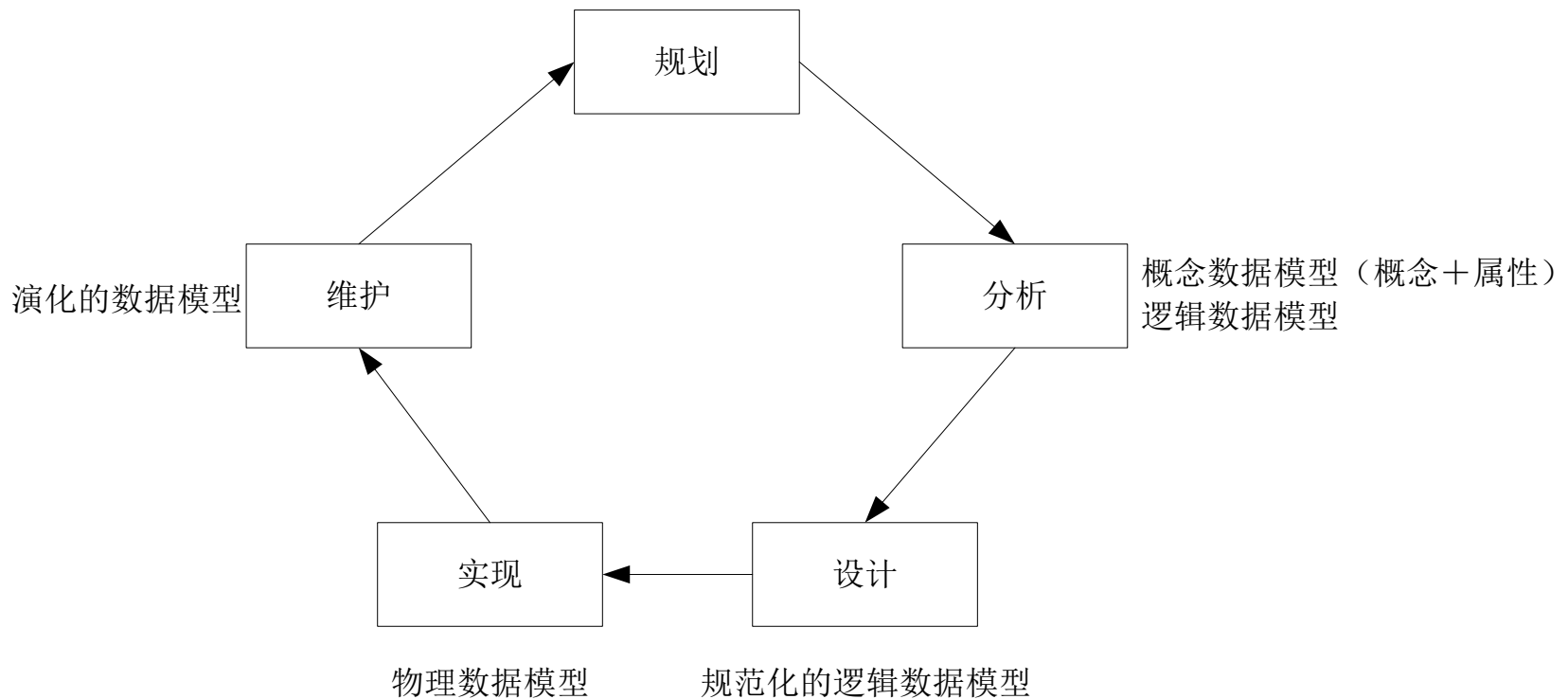
- 数据模型：描述数据的**定义、结构和关系**等特性的模型
- 实体关联图：一种数据模型  
亦称ER图 (Entity-relationship diagram)或称实体联系图，主要用于描述系统的数据关系。



# 实体关联图ERD(续)

## ■ 数据模型的应用

概念数据模型（仅仅是概念）





## 实体关联图ERD(续)

### ■ ERD组成

- 实体：数据项的集合
- 属性：实体的性质
- 实体间的关联：实体之间相互连接的方式
  - 一对一关联(1:1)
  - 一对多关联(1:N)
  - 多对多关联(M:N)



## 实体关联图ERD(续)

### ■ 实体

- 实例(Instance): 需要在系统中收集和存储的现实世界事物
- 实体(Entity): 具有相同特征和属性的实例集的分类描述

实例

**Name: Sandra Dee**

**ID: 205-7123**

**DOB: Jan 17, 1962**

实体

**Student**

**ID  
Name  
DOB**

Student

概念实体

逻辑实体

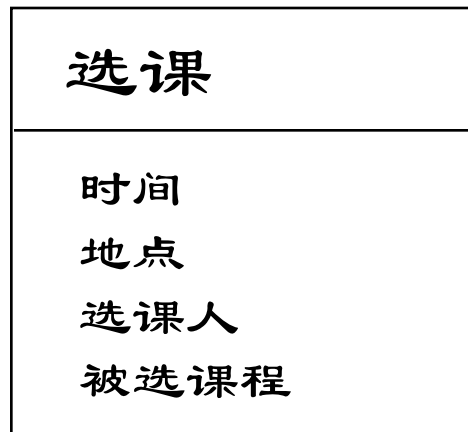


## 实体关联图ERD(续)

### ■ 实体

除了静态的事物和抽象的概念外，**行为和事件**也是常见的实体类型，称为**进程实体**。

系统需要它们在某些时刻的快照或者它们的运行环境信息，而不是它们所体现出来的功能和达成的效果。





## 实体关联图ERD(续)

### ■ 属性

- 属性是实体的**特征**，**不是数据**
- 属性会以一定的形式存在，这种存在才是数据，称为**属性的值**(Value)
- 一系列属性的存在集成起来就可以描述一个实体的实例





## 实体关联图ERD(续)

### ■ 属性

- 属性的值应该是一个合法的、有业务含义的值，这个合法的取值范围称为域(Domain)

| 数据类型        | 类型说明 | 域             | 例子                    |
|-------------|------|---------------|-----------------------|
| Number      | 整数   | {最小—最大}       | 月份的域： {1—12}          |
| Real        | 实数   | {最小—最大}       | 考试得分： {0.0—100.0}     |
| Text        | 文本   | TEXT(属性的最大长度) | 电话号码： TEXT (20)       |
| Date        | 日期   | {最早—最晚}       | 出生日期： {1900-01-01—今天} |
| Time        | 时间   | {最早—最晚}       |                       |
| Boolean     | 布尔   |               |                       |
| Enumeration | 枚举   | {值1、...、值n}   | 性别： {男、女、未知}          |



## 实体关联图ERD(续)

### ■ 标识符/键(key)

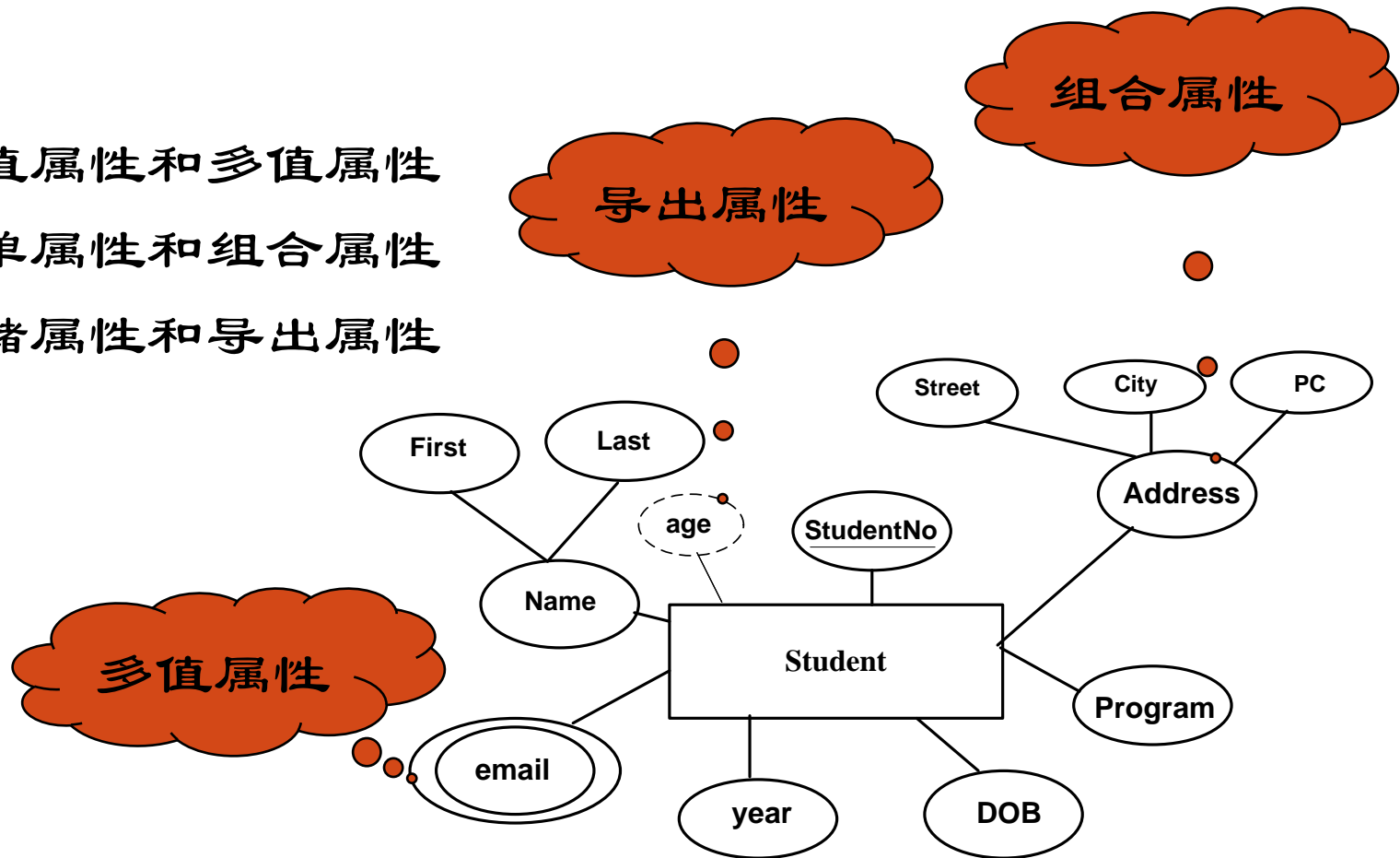
- 用来唯一地确定和标识每个实例的属性或属性组合
- 一个实体可能有多个键，都称为**候选键**(Candidate Key)。通常会从多个候选键中选择固定的某一个或几个键进行实例的标识。
  - 被选中的候选键称为**主键**(Primary Key)
  - 没有被选作主键的候选键称为**替代键**(Alternate Key)



## 实体关联图ERD(续)

### ■ 属性

- 单值属性和多值属性
- 简单属性和组合属性
- 存储属性和导出属性





## 实体关联图ERD(续)

- 关联：存在于一个或多个实体之间的自然业务联系
  - 所有的关系隐含地都是双向的
  - 关系表达的不是实体物理上的联系（如车和轮胎），而是逻辑上的连接

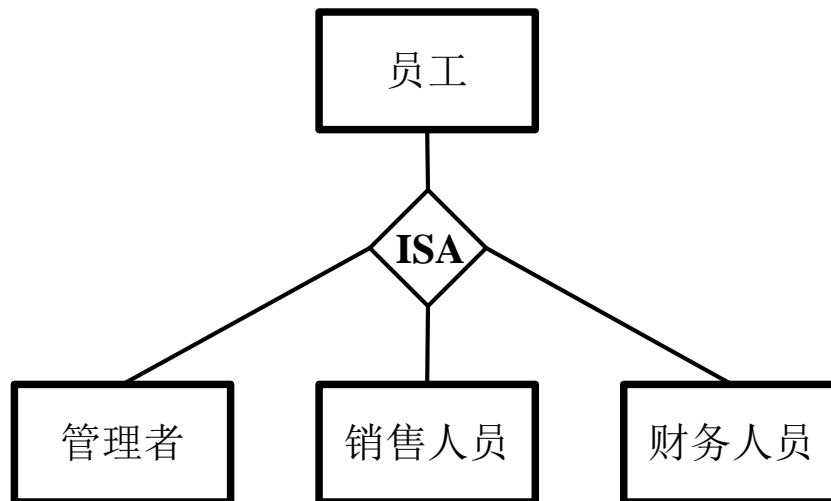




## 实体关联图ERD(续)

### ■ 子类型关系

在多个实体大部分相似，小部分不同时，可以从相似的实体中抽取共性，建立一个公共的**超类型**，所有实体都是超类型的**子类型**。

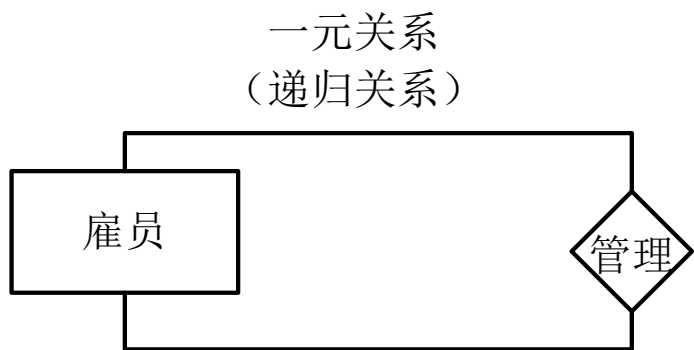




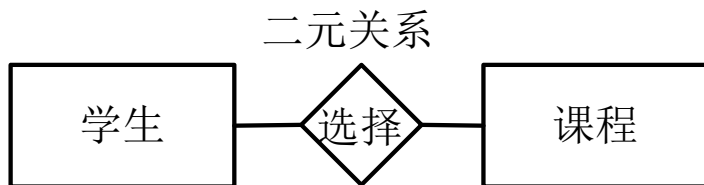
## 实体关联图ERD(续)

### ■ 度数：参与关系的实体数量

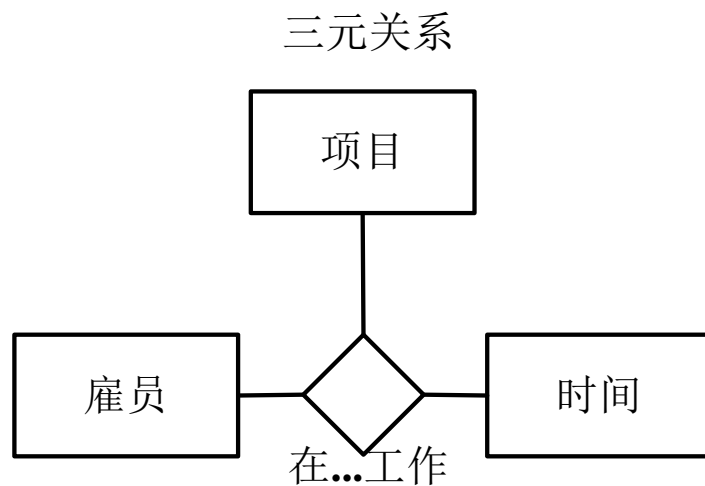
是度量关系复杂度的一个指标



(a)



(b)



(c)



## 实体关联图ERD(续)

### ■ 基数(约束)

- 最大基数(键约束)：对关系中任意的其他实体实例，该实体实例可能参与关系的最大数量
- 最小基数(参与约束)：对关系中任意的其他实体实例，该实体实例可能参与关系的最小数量





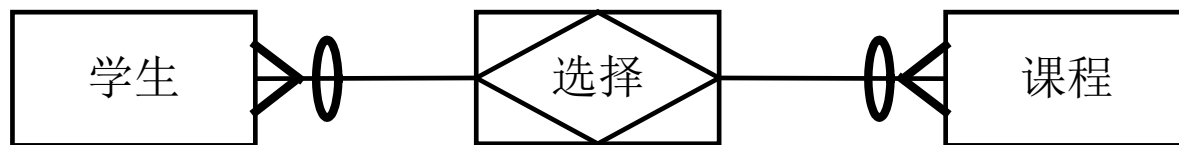
## 实体关联图ERD(续)

### ■ 被关系影响的实体

- 弱实体：指存在和标识需要依赖于其他实体的实体



- 关联实体：实体间建立关系时的副产品



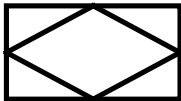





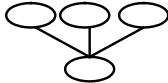

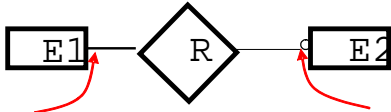
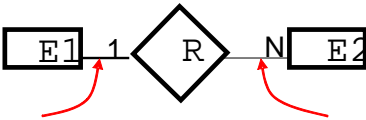






# 实体关联图ERD(续)

## ■ ERD表示方法

|    |  |  |   |   |   |
|----|--|--|---|---|---|
| 实体 | 实体<br>              | 弱实体<br>   | 关联实体<br>                    |   |   |
| 关系 | 关系<br>              | 子类型关系<br> |   |   |   |
| 属性 | 属性<br>              | 标识符属性<br> | 多值属性<br>                    | 组合属性<br> | 导出属性<br> |
| 基数 | E1强制参与，E2可选参与<br> |  | E1最多一个实例参与，E2最多N个实例参与<br> |   |   |

Mandatory One

Mandatory Many

Optional One

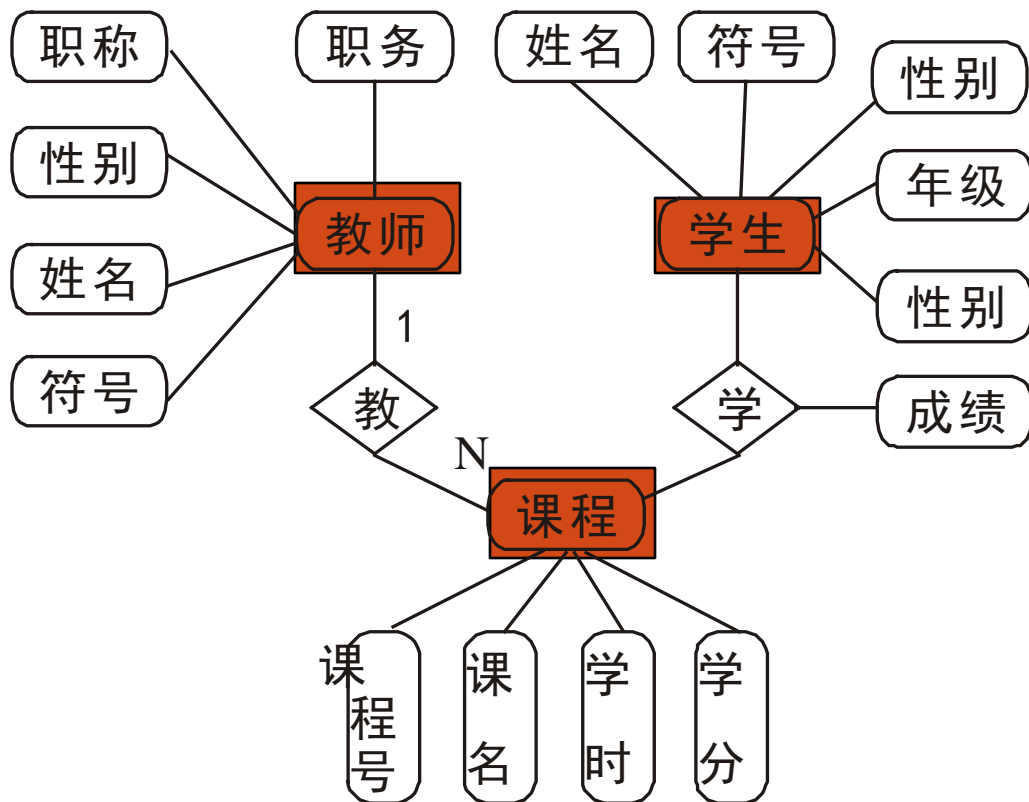
Optional Many





## 实体关联图ERD(续)

### ■ 示例：某校学生管理系统的ER图





## 实体关联图ERD(续)

### ■ 好处：

- ER图使用简单的图形符号，具有较好的易理解性
- ER图可以与数据词典相结合对属性进行详细定义
- 通过实体间的关联关系可发现遗漏和冗余的数据项



## 实体关联图ERD(续)

### ■ RED建模

#### (1) 从描述信息中辨识实体

— 可以重点关注描述信息中名词

#### (2) 确定实体的标识符

#### (3) 建立实体间关系，判断各个关系的建立是否会产生新的关联实体或者影响已有的实体特性

#### (4) 添加详细的描述信息

— 实体的详细属性和关系的基数



## 实体关联图ERD(续)

### ■ 示例：

- ❖ 研讨班在每个学年开始的时候开设，然后持续一个学年。
- ❖ 每个研讨班针对一个或几个研究方向。
- ❖ 每个研讨班由一位或几位教师主持。
- ❖ 在研讨班开设之后，学生可以根据主持教师（的姓名）和研讨班的方向来选择和参加某个研讨班。
- ❖ 所有的学生必须且只能参加一个研讨班的学习。
- ❖ 研讨班时常会开展活动，由教师来决定活动的时间、地点、主题和做报告的学生（的姓名）。
- ❖ 每次活动时，由一位或多位同学围绕活动主题做学习报告，交流自己对新技术的学习心得。
- ❖ 每个学生一次活动最多只能作一个报告，但每个学生至少会在一次活动中做一个报告。
- ❖ 教师对每份活动中的学生报告进行一次点评和指导，提出建议和意见。



## 实体关联图ERD(续)

### ■ 示例：

(1) 研讨班

~~学年~~

~~研究方向~~

教师

学生

活动

~~(活动的)~~ 时间 ~~(活动的)~~ 地点

~~(活动的)~~ 主题

学习报告

~~学习心得~~

~~建议和意见~~

学生

教师

研讨班

活动

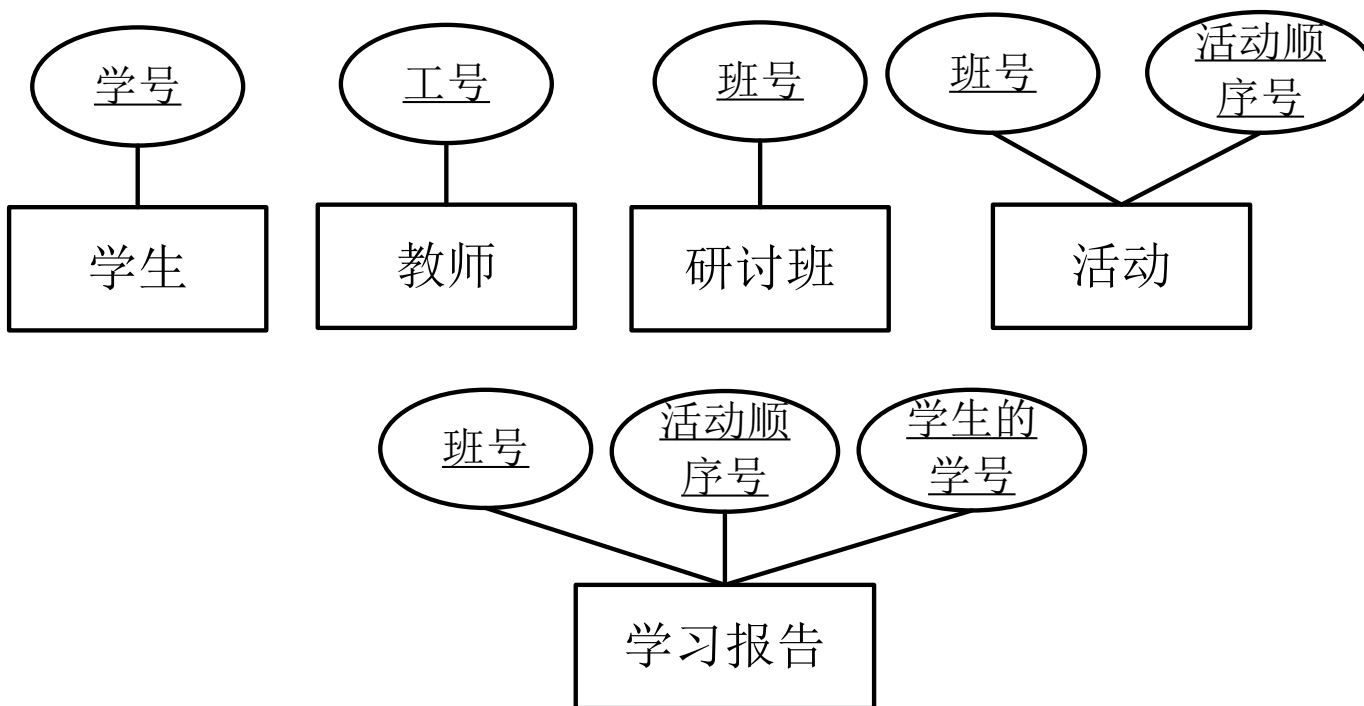
学习报告



## 实体关联图ERD(续)

■ 示例：

(2)

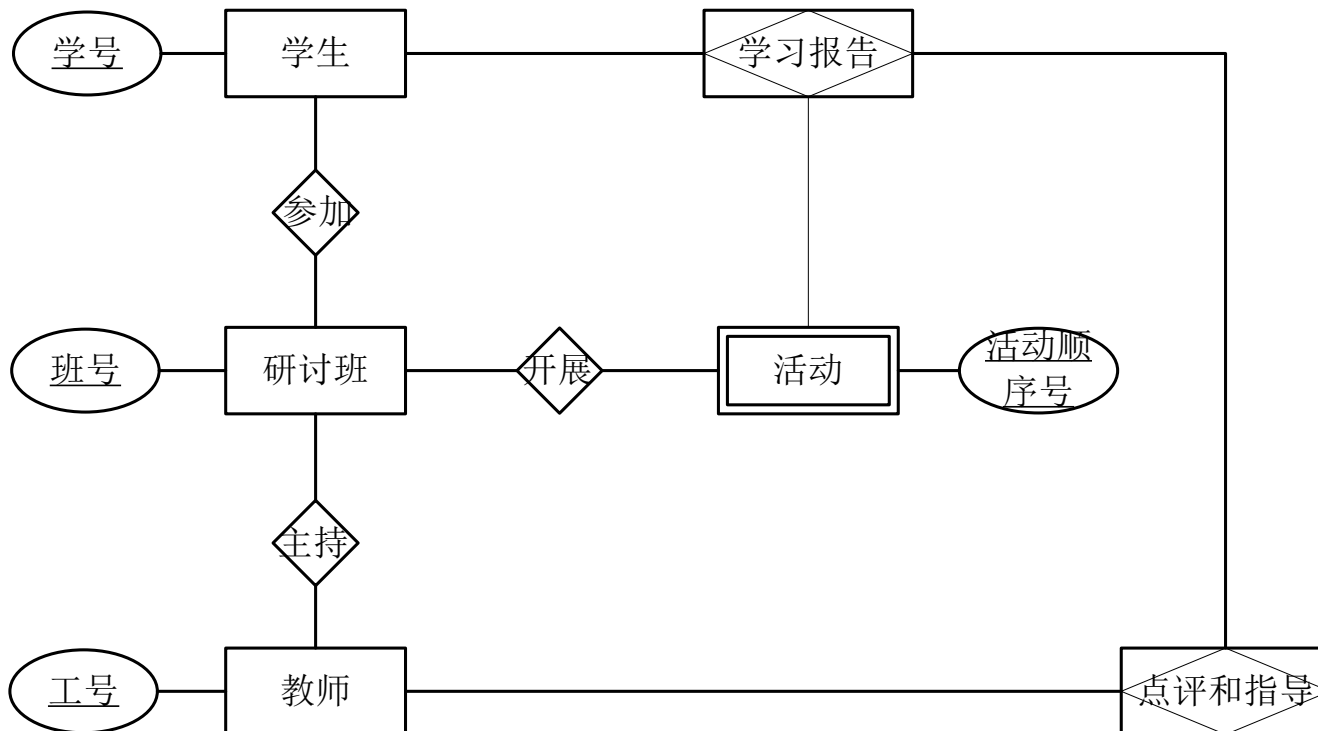




## 实体关联图ERD(续)

■ 示例：

(3)



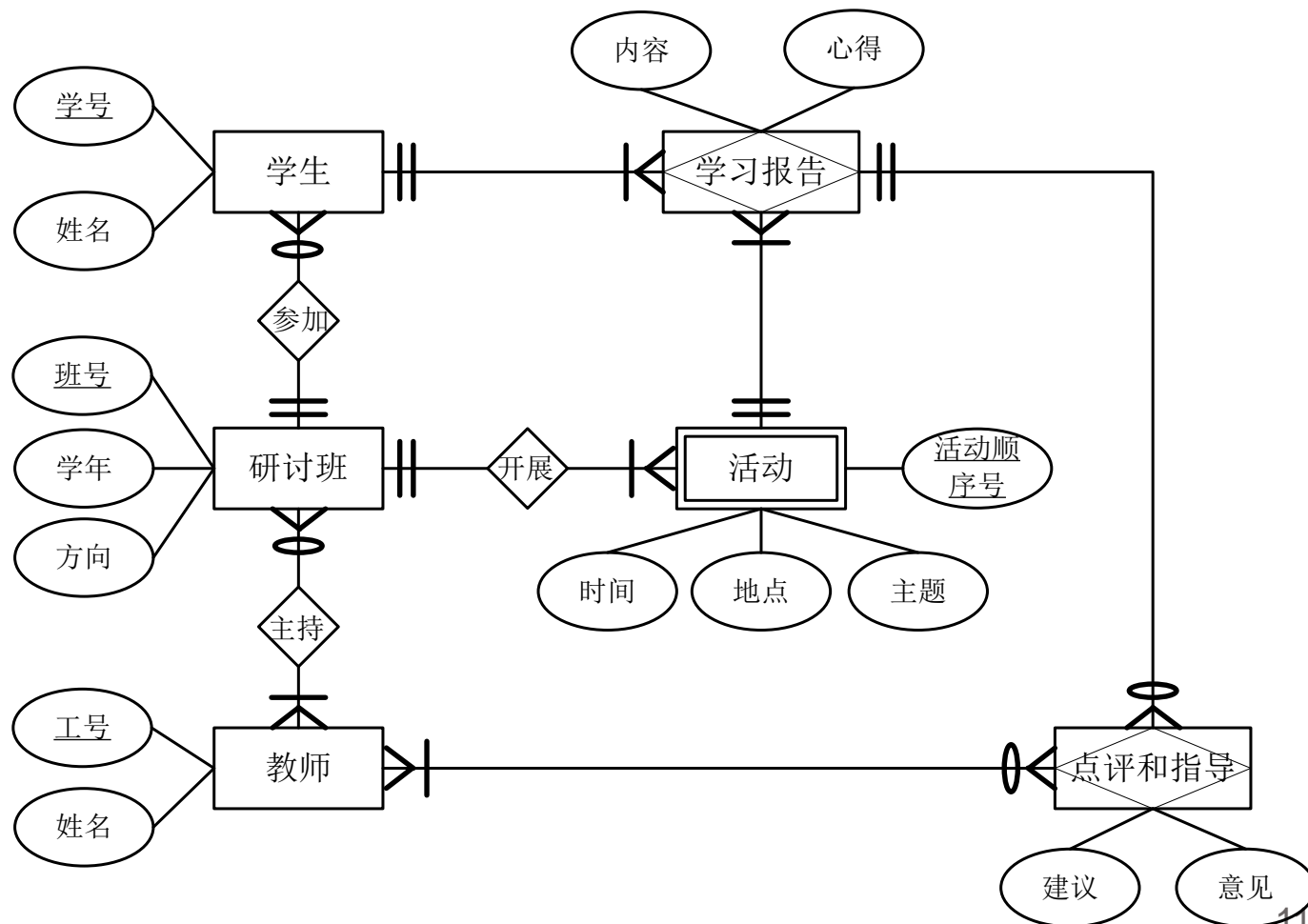




## 实体关联图ERD(续)

■ 示例：

(4)





## 课程内容

- 5.1 什么是模型
- 5.2 结构化的需求建模方法
- **5.3 面向对象的需求建模方法**



## 5.3 面向对象的需求建模方法

- G.Booch：面向对象设计方法OOD
- J.Rumbaugh：面向对象建模技术OMT
- I.Jacobson：面向对象的软件工程OOSE
- P. Coad和Ed. Yondon：面向对象的分析/设计方法OOAD
- J.Rumbaugh和G.Booch：综合OMT和OOD

结合OOSE：统一的面向对象的需求建模/设计方法、  
统一建模语言UML、支持需求建模的工具系统



## 面向对象的需求建模方法(续)

- 5.3.1 面向对象的需求分析
- 5.3.2 基于OMT方法的需求建模
- 5.3.3 基于UML的需求建模

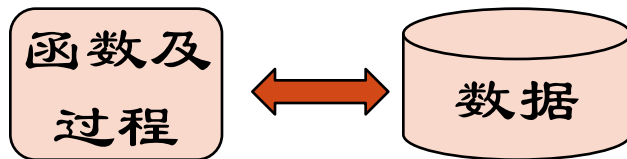


## 5.3.1 面向对象的需求分析

### ■ 结构化方法 VS 面向对象方法

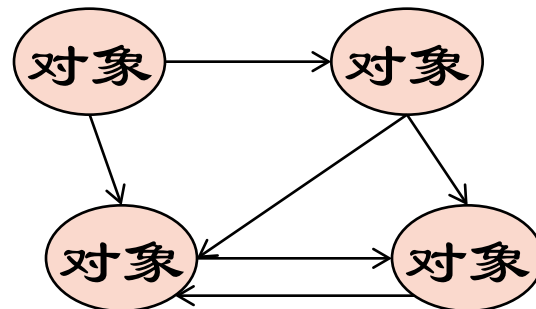
结构化方法：

一个系统应该被划分为两个部分：数据（使用数据模型建模）和功能（使用过程模型建模）



面向对象方法：

把系统定义为一组正在交互的对象，对象可以完成一些事情（对象具有功能），对象也知道一些事情（对象有数据）





## 面向对象的需求分析(续)

### ■ 面向对象的需求建模的关键

从获取的需求信息中识别出问题域中的**类与对象**，  
并分析它们之间的**关系**，最终建立起简洁、精确和  
易理解的需求模型



# 面向对象的需求分析 (续)

## 对象与类

- 对象是客观实体的抽象，是构成概念模型的基本单元
  - 人类能感知的物理实体
  - 人或者组织
  - 现实中发生的事件
  - 两个以上实体的相互作用
  - 需要说明的概念



## 面向对象的需求分析 (续)

### 对象与类

- 类是对具有相同性质和操作的一个或多个对象的描述，是一组对象的集合，是创建对象的有效模板
- 类与对象的关系  
类给出了该类中所有对象的抽象定义（主要指属性和方法两部分），而对象是符合该定义的一个实例



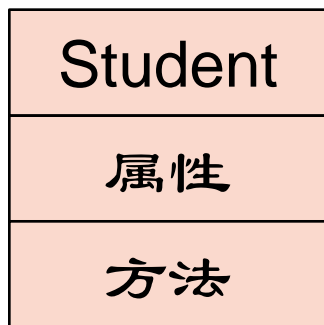


## 面向对象的需求分析 (续)

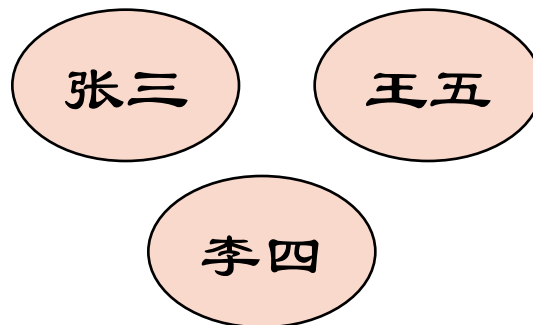
### 对象与类

#### ■ 类的定义

- 属性：定义数据
- 方法：定义功能



学生类



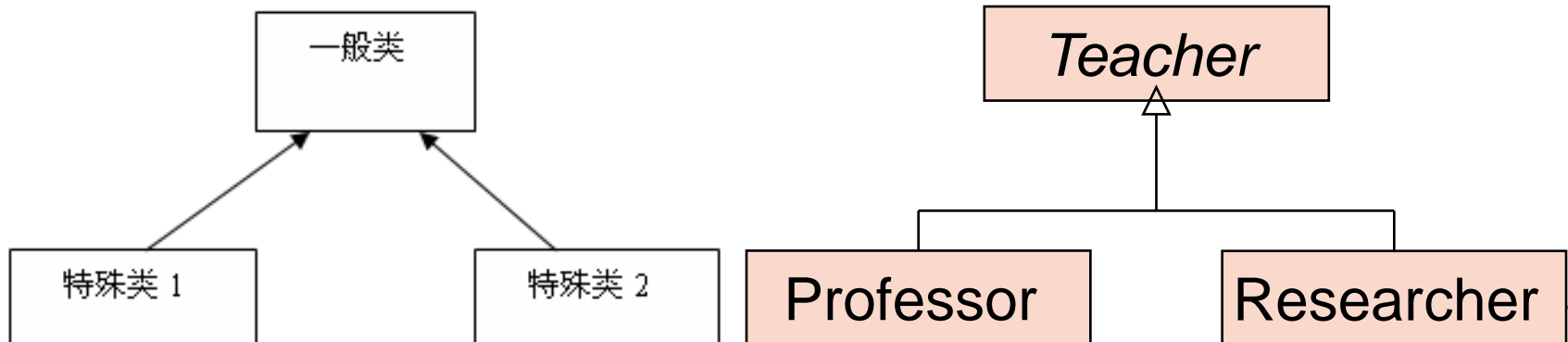
学生对象



## 面向对象的需求分析 (续)

### 继承

- 不同的类之间经常会存在相似性，相同的属性或相同的方法  
如：教授和研究员都有姓名、地址、学术方向等属性
- 继承主要由父类和子类的关系（“一般-特殊”或”Is-a”关系）引起
- 继承关系的反关系称为**泛化关系**

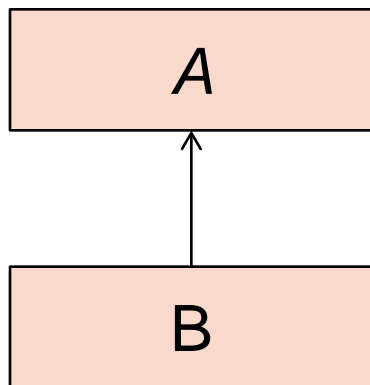




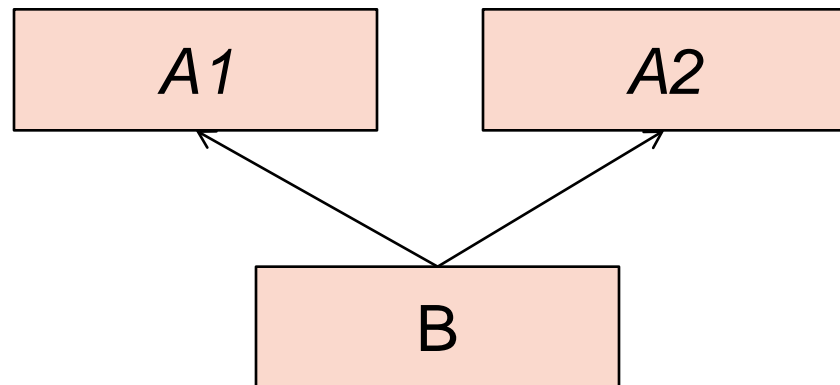
## 面向对象的需求分析 (续)

### 继承

- 单一继承
- 多重继承



单一继承



多重继承



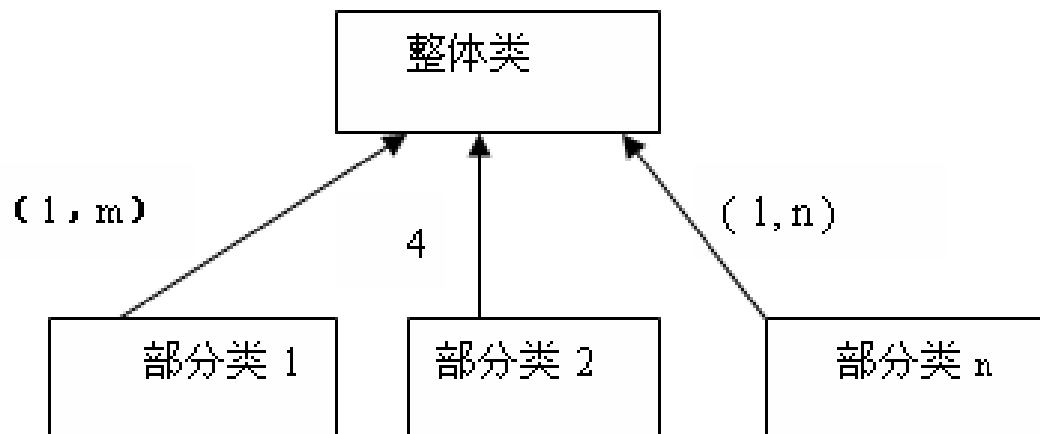
## 面向对象的需求分析 (续)

### 聚合/组合

- 一个对象会由其他对象组成

如：汽车由车头、车身和轮子等组成

- 聚合/组合关系也称“整体-部分”或“Part-of”关系

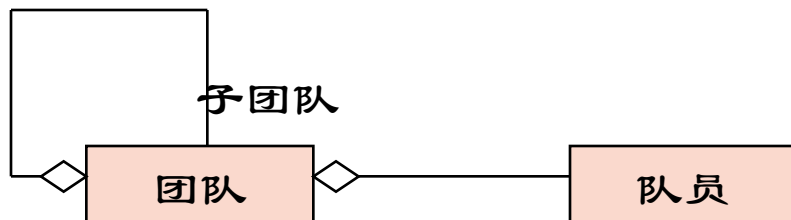




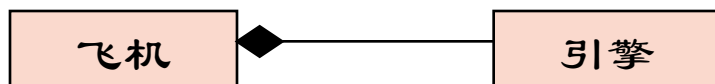
## 面向对象的需求分析 (续)

### 聚合/组合

- 聚合：部分类可以独立于整体类而独立存在。



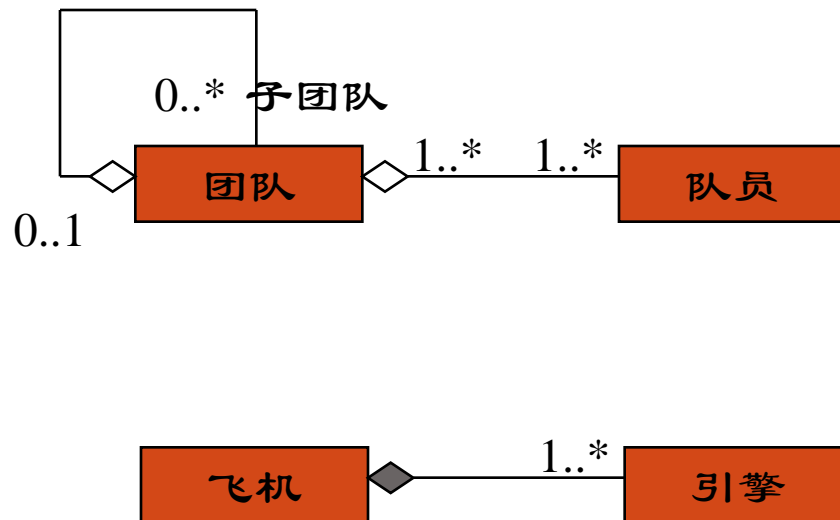
- 组合：部分类的存在完全依赖于整体类





## 面向对象的需求分析 (续)

### 聚合/组合



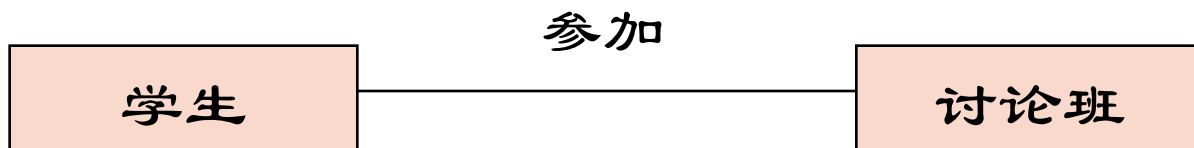


## 面向对象的需求分析 (续)

### 关联

- 类(对象)之间存在关系和关联

如：司机开汽车、学生参加讨论班



- 对象之间的关联可以通过对象中的属性形成

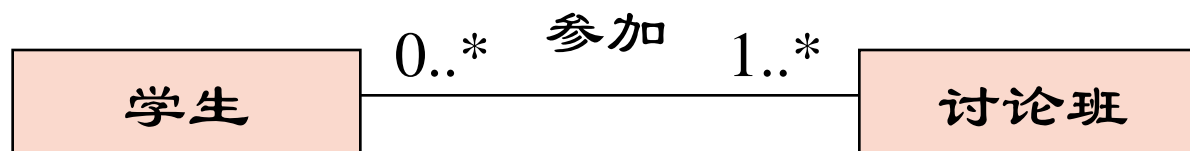
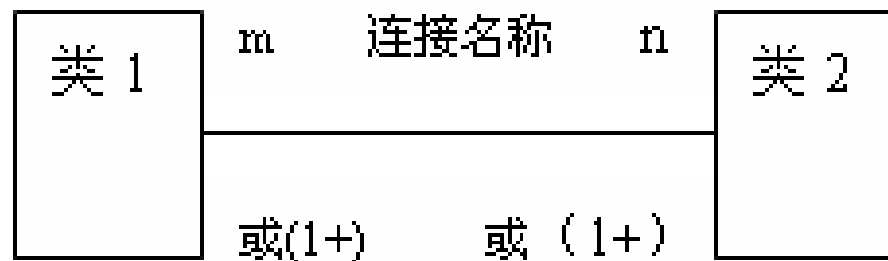
如：司机与汽车对象之间可以通过司机对象中的“可驾驶的汽车类型”属性来建立关联



## 面向对象的需求分析 (续)

### 关联

#### ■ 关联的数量表示







## 面向对象的需求分析 (续)

### 消息

- 消息是系统运行过程中对象之间相互传递的、请求服务的信息。
- 消息实际上就是一段数据结构，包括接收消息的对象名、请求的方法名、输入参数等
- 消息协议：一个对象对外服务所要求的消息格式



## 面向对象的需求分析 (续)

### 耦合

- 当一个类依赖另一个类时，称其为**耦合**的
  - 当一个类与另一个类交互，但不知道该类的实现细节时，称其为**松耦合**的
  - 当一个类依赖于另一个类的实现时，称其为**紧耦合**的

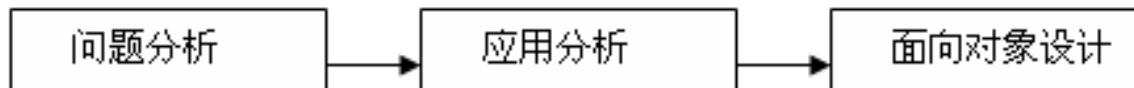


## 面向对象的需求分析(续)

■ 根据面向对象的过程模型：

(1) 问题分析

(2) 应用分析





## 面向对象的需求分析(续)

### (1) 问题分析

- ① 收集并确认用户的需求信息；
- ② 对实际问题进行功能分析和过程分析，从中抽象出问题中的基本概念、属性和操作；
- ③ 用继承、聚合和关联结构描述概念实体间的静态关系；
- ④ 将概念实体标识为问题域中的对象类，以及定义对象类之间的静态结构关系和信息连接关系，最终建立**关于对象的分析模型**。



## 面向对象的需求分析(续)

### (2) 应用分析

动态描述系统中对象的合法状态序列，并用**动态模型**表达对象的动态行为、对象之间的消息传递和协同工作的动态信息。



## 面向对象的需求分析(续)

### ■ OMT方法

- 基本思想：将面向对象的分析过程视为一个模型的构建过程，即整理获取的需求信息并逐步分析和建立需求模型的过程
- 3个模型：
  - 对象模型：描述系统静态数据结构
  - 动态模型：描述系统控制结构
  - 功能模型：描述系统功能



## 面向对象的需求分析(续)

### ■ OOAD方法

- 基本思想：使用基本的结构化原则，结合面向对象的概念，构造一个描述系统功能的需求模型
- 5个概念层次：
  - 主题层
  - 类与对象层
  - 结构层
  - 属性层
  - 服务层



## 面向对象的需求建模方法(续)

- 5.3.1 面向对象的需求分析
- **5.3.2 基于OMT方法的需求建模**
- 5.3.3 基于UML的需求建模





## 5.3.2 基于OMT方法的需求建模(续)

### ■ 基于OMT方法的需求模型

- 对象模型
- 动态模型
- 功能模型



## 基于OMT方法的需求建模(续)

### ■ 对象模型

- **侧重点**：定义系统的静态结构，描述系统内对象的标识、属性，与其他对象的关系
- **作用**：为建立动态模型和功能模型提供概念性框架
- **描述工具**：类图、ER图



## 基于OMT方法的需求建模(续)

### ■ 动态模型

- **侧重点**：表达在系统动态交互行为中对象的状态发生变化的时序过程
- **作用**：描述对象的动态行为，利用事件和对象状态表达系统的动态特性
- **描述工具**：
  - 状态转换图：描述对象的生存过程
  - 序列图：描述对象之间的交互



## 基于OMT方法的需求建模(续)

### ■ 功能模型

- **侧重点**：表达系统的功能信息
- **作用**：定义系统应该“做什么”
- **描述工具**：数据流图



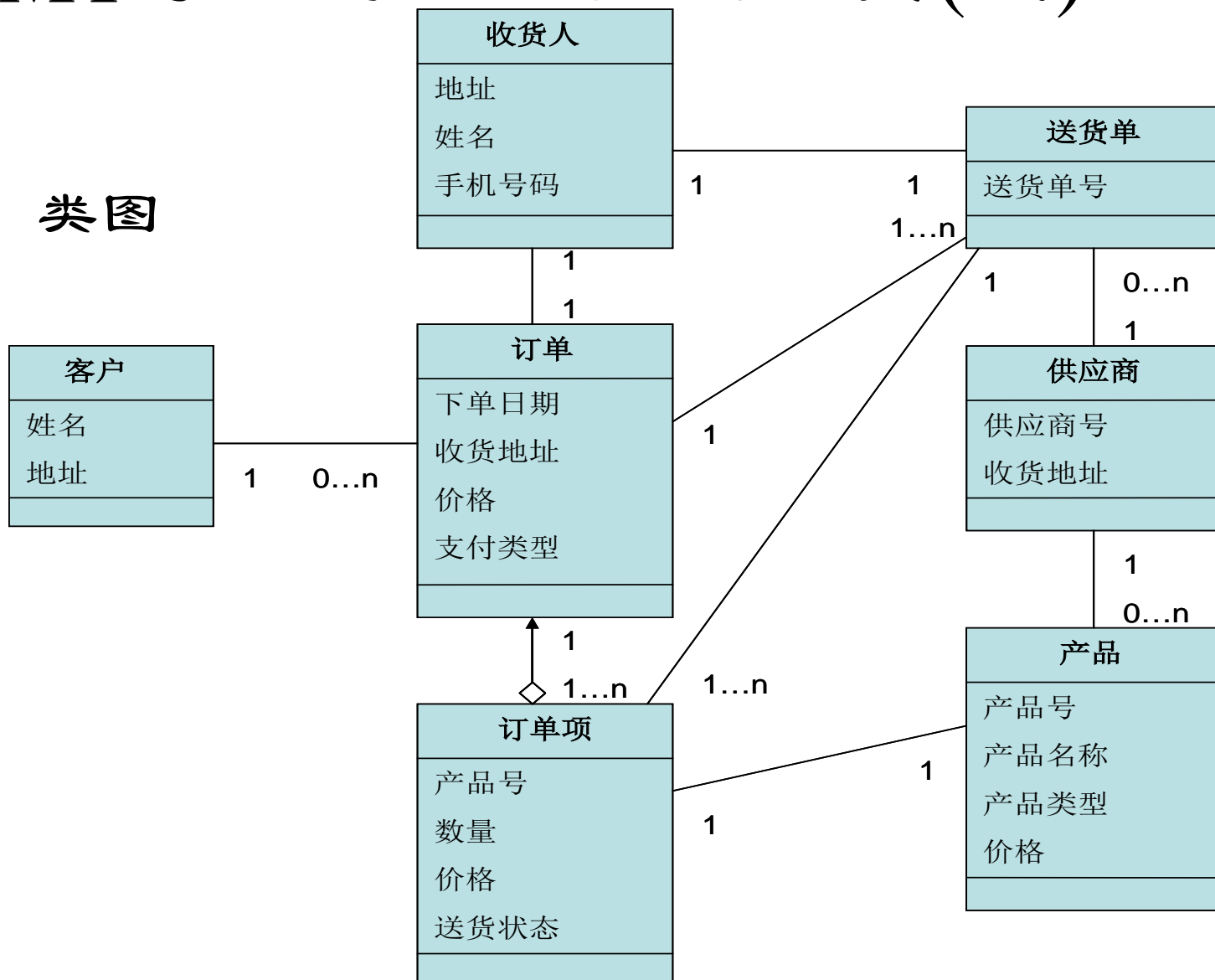
## 基于OMT方法的需求建模(续)

- **OMT方法的图形描述工具**
- 基于OMT方法的需求建模步骤



# OMT方法的图形描述工具(续)

## 类图





## OMT方法的图形描述工具(续)

### ■ 状态转换图(状态图)

通过描述系统的状态及引起系统状态转换的事件来表示系统的行为。

- 状态
- 事件
- 状态转换



# OMT方法的图形描述工具(续)

## ■ 状态转换图

- 状态是任何可以被观察到的系统行为模式

- 初始状态：实心圆

- 结束状态：同心圆

- 中间状态：圆角矩阵(或圆)

只能有一个初态，而结束状态可有0至多个。





# OMT方法的图形描述工具(续)

## ■ 状态转换图

### ● 事件

事件是在某个特定时刻发生的事情，能引起系统做动作和状态转换

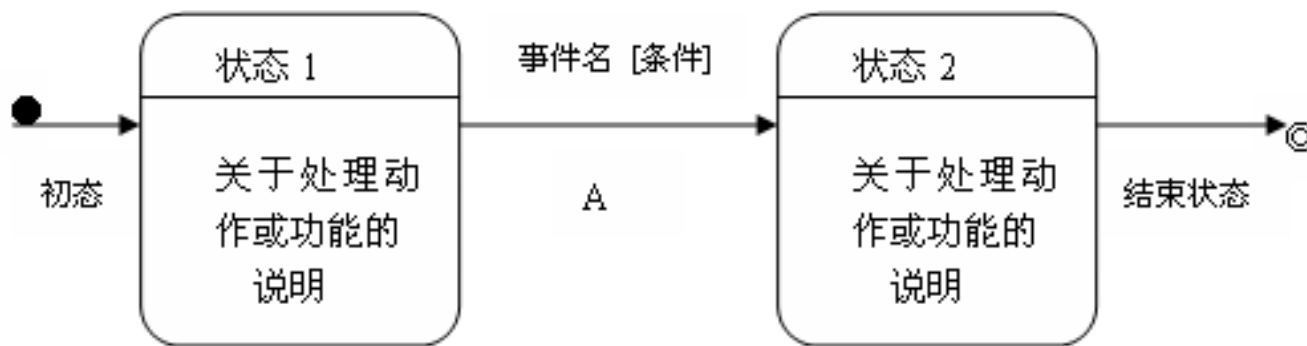
### ● 状态转换

由某事件引起的两个状态之间的变化称为状态转换



## OMT方法的图形描述工具(续)

### ■ 状态转换图



状态图的简例



# OMT方法的图形描述工具(续)

## ■ 状态转换图

画状态图的基本步骤：

- ① 确定初态；
- ② 确定事件（事件可由动作或输入信息等形式），并根据事件以及某些限制条件确定由当前状态转到下一个状态，以形成一个状态转换；
- ③ 重复2的过程，直到最后确定结束状态为止。



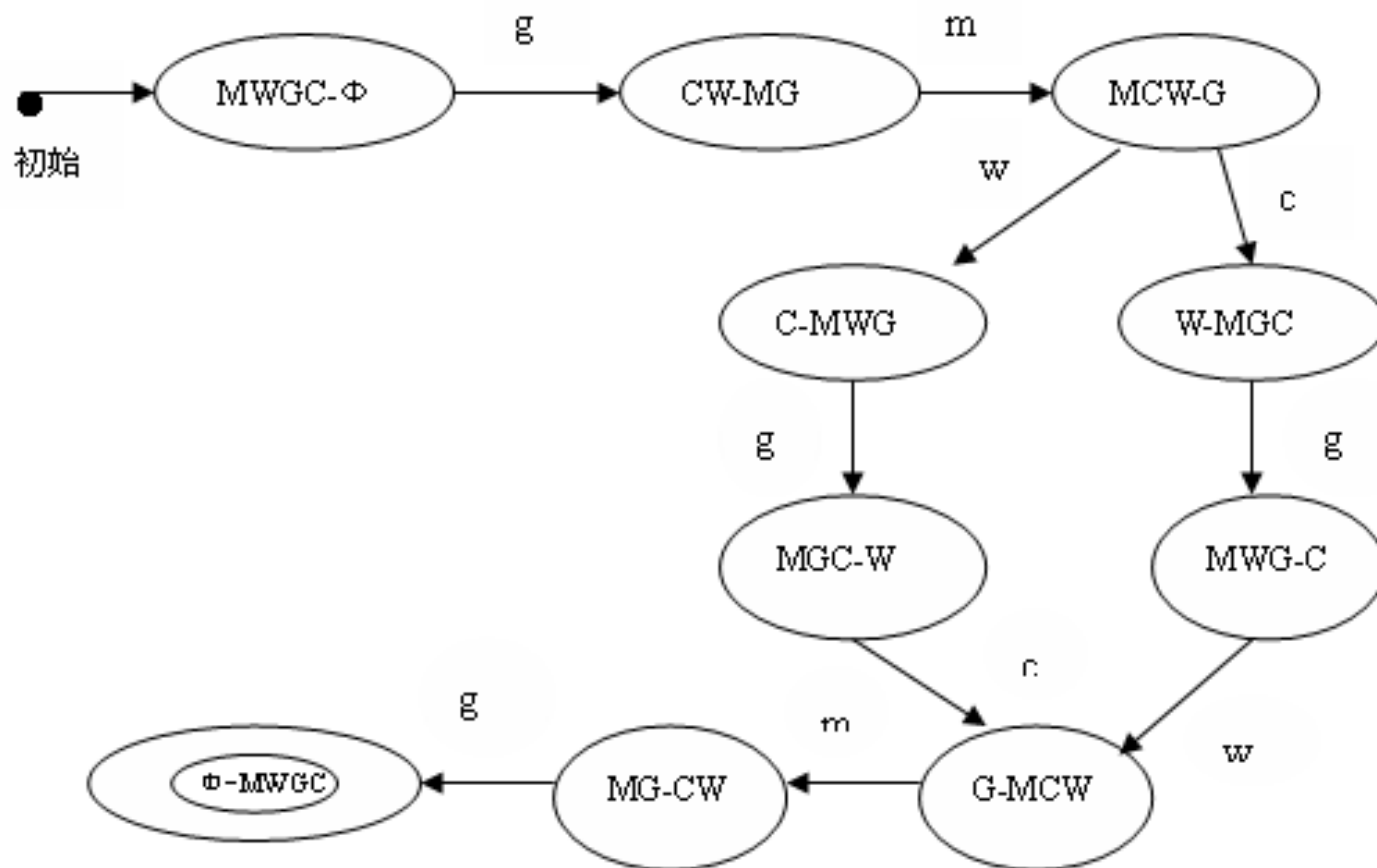
## OMT方法的图形描述工具(续)

### ■ 状态转换图

例：一个人带着一头狼、一头羊以及一棵青菜，处于河的左岸。有一条小船，每次只能携带人和其余的三者之一。人和他的伴随品都希望渡到河的右岸，而每摆渡一次，人仅能带其中之一。然而，如果人留下狼和羊不论在左岸还是在右岸，狼肯定会吃掉羊。类似地，如果单独留下羊和菜，羊也肯定会吃掉菜。如何才能既渡过河而羊和菜又不被吃掉呢？



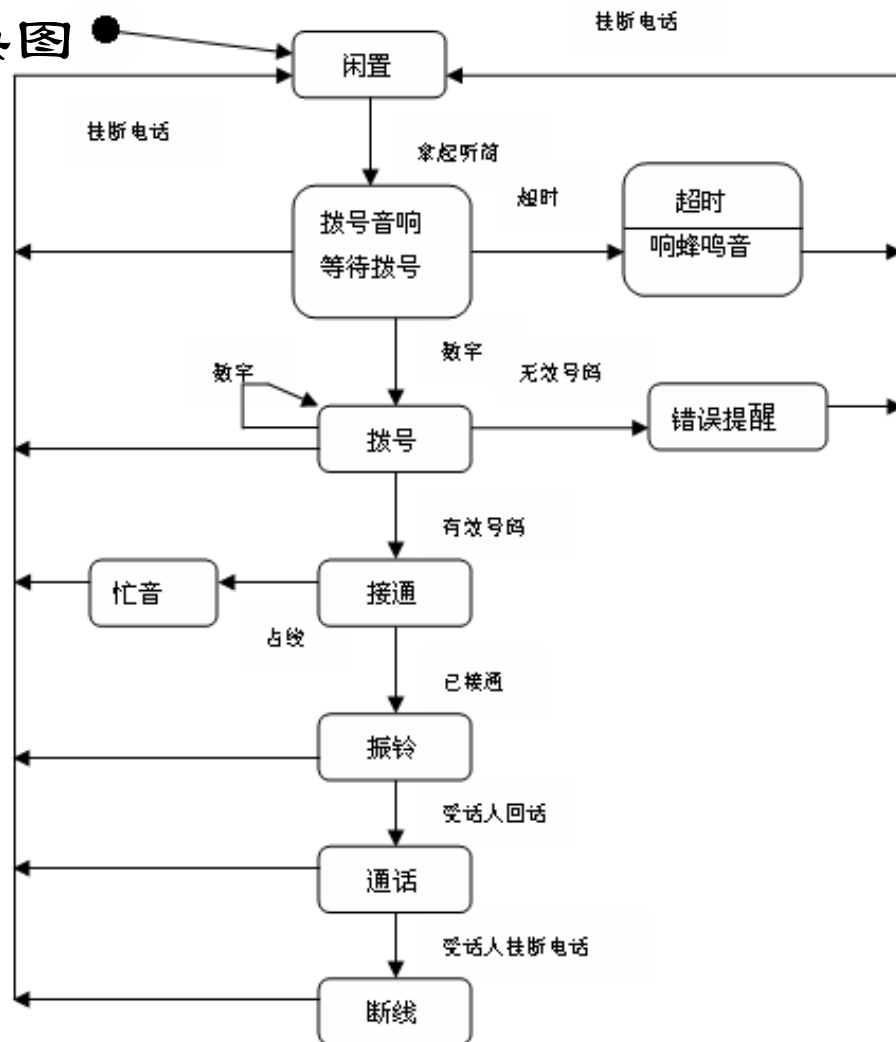
## OMT方法的图形描述工具(续)





# OMT方法的图形描述工具(续)

例：电话系统的状态转换图

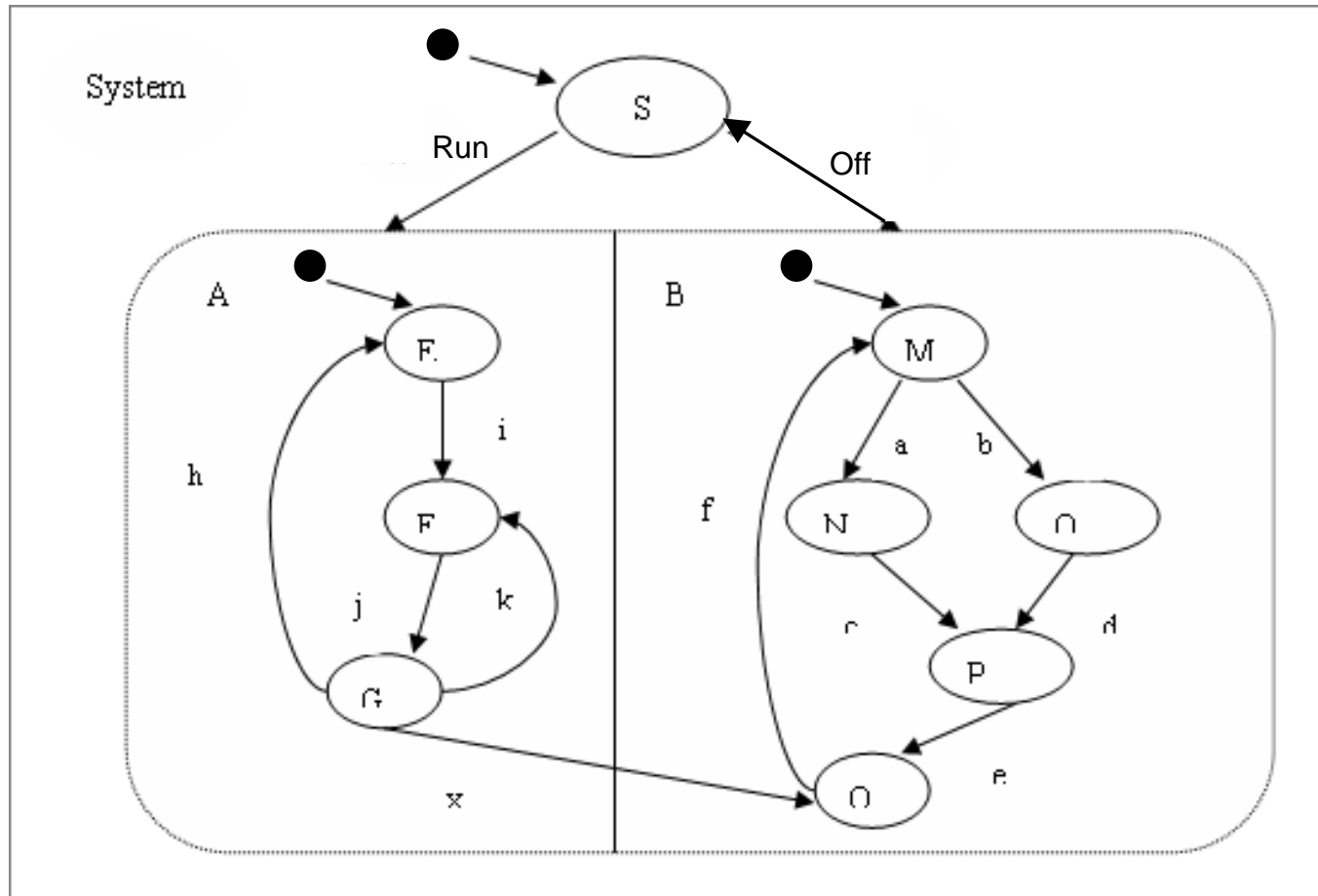




## OMT方法的图形描述工具(续)

### ■ 扩充的状态转换图

- 原来的状态图不能描述并行处理过程的情况
- 会面临“状态爆炸”的问题



扩充的状态转换图简例





# OMT方法的图形描述工具(续)

## ■ 扩充的状态转换图

- ① 扩充后的状态图仍继续沿用原来状态图符号。
- ② 引入超状态（也称抽象状态）的概念，而超状态又可表示为由多个状态组成的状态图（或称子状态图）
- ③ 基于超状态概念，状态图可表示为层次式的，并且每个超状态可表示一个处理过程。超状态间的关系可用“与”和“或”关系给予表示。



# OMT方法的图形描述工具(续)

## ■ 扩充的状态图

- ④ 一个上层状态图的事件可同时引起多个并行状态图（处理过程）工作。此外，并行的状态图之间可互相通过事件使对方发生状态转换。
- ⑤ 多个并行的状态图首先处于各自的初始状态，然后各状态图根据事件各自发生状态转换。这些状态图能通过某一事件同时到达上层状态图中的某一状态；也可以由某一子状态图通过某一事件到达上层状态图中的某一状态，从而强制结束并行执行的状态。
- ⑥ 每个子状态图都有各自的初始状态，而结束状态的有无可视具体情况设置。



# OMT方法的图形描述工具(续)

## ■ 扩充的状态图

### 例：温度控制系统

系统自动控制空调装置进行制冷或制热，以调节室内温度。系统最初处于Halt状态，当操作员按下On按钮后，系统开始运行。

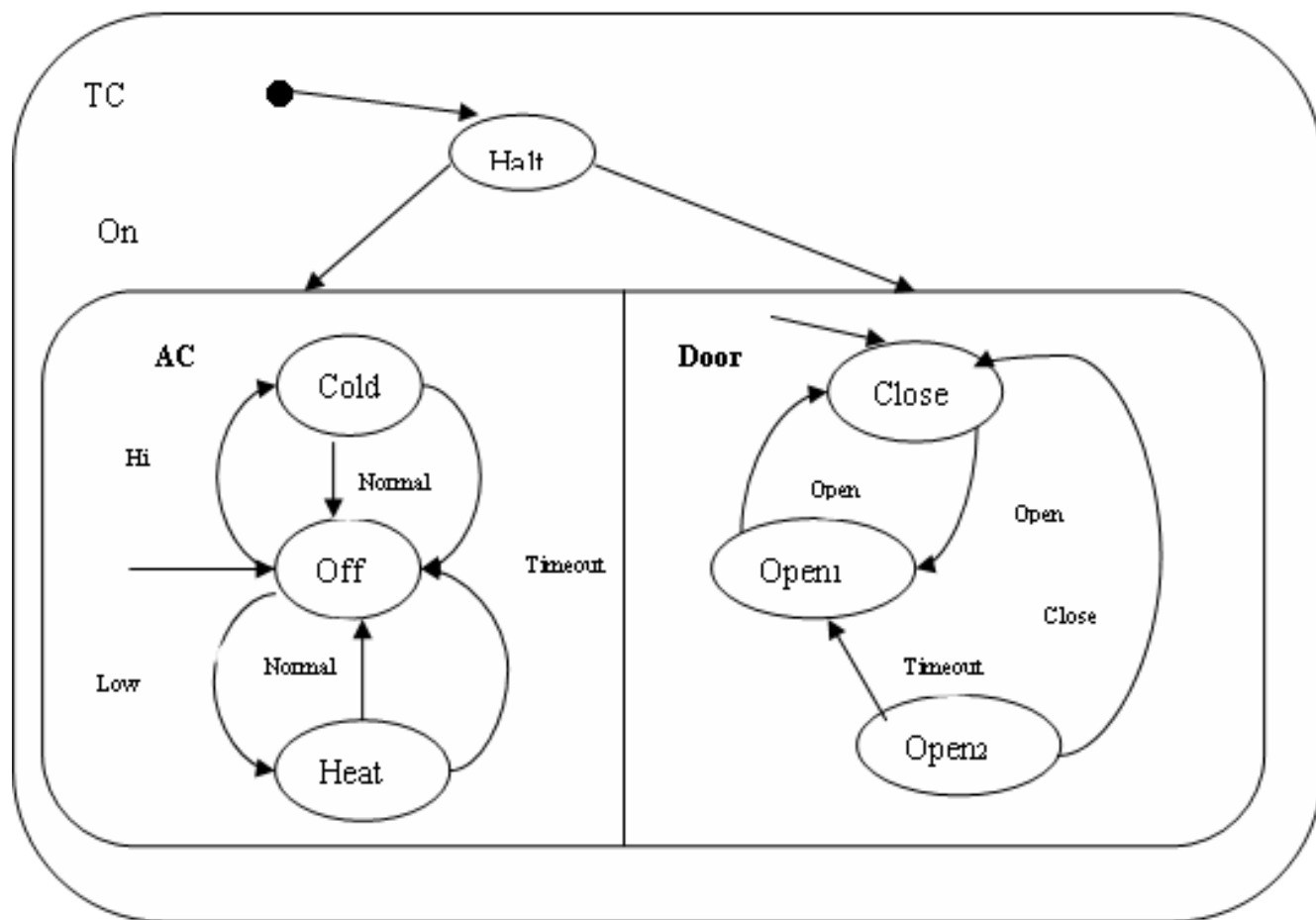
当室温高于或低于规定的正常范围且房门关闭时，系统启动空调装置制冷或制热。

同时，该系统也将根据房门的开关情况控制空调装置工作。当房门打开且超过系统规定的时间限制时，系统将自动产生超时事件TimeOut，以停止空调装置工作

当操作员按下Stop按钮时，整个系统将强行停止运行。

约定：启动该控制系统运行前，房门关闭，空调未运行；

空调装置制冷或制热时，房门被打开，但在规定的时间内关闭，将不影响空调工作



温控系统的状态转换图



## OMT方法的图形描述工具(续)

### ■ 序列图

序列图主要用于表达对象与对象之间可能发生的所有事件，以及按事件发生时间的先后顺序列出所有事件的一种图形工具。



## OMT方法的图形描述工具(续)

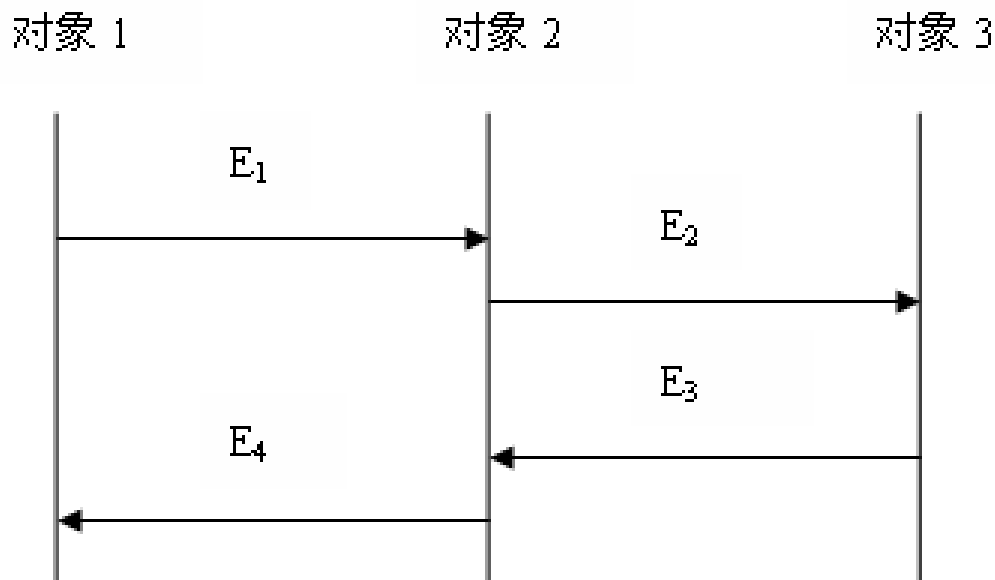
### ■ 序列图

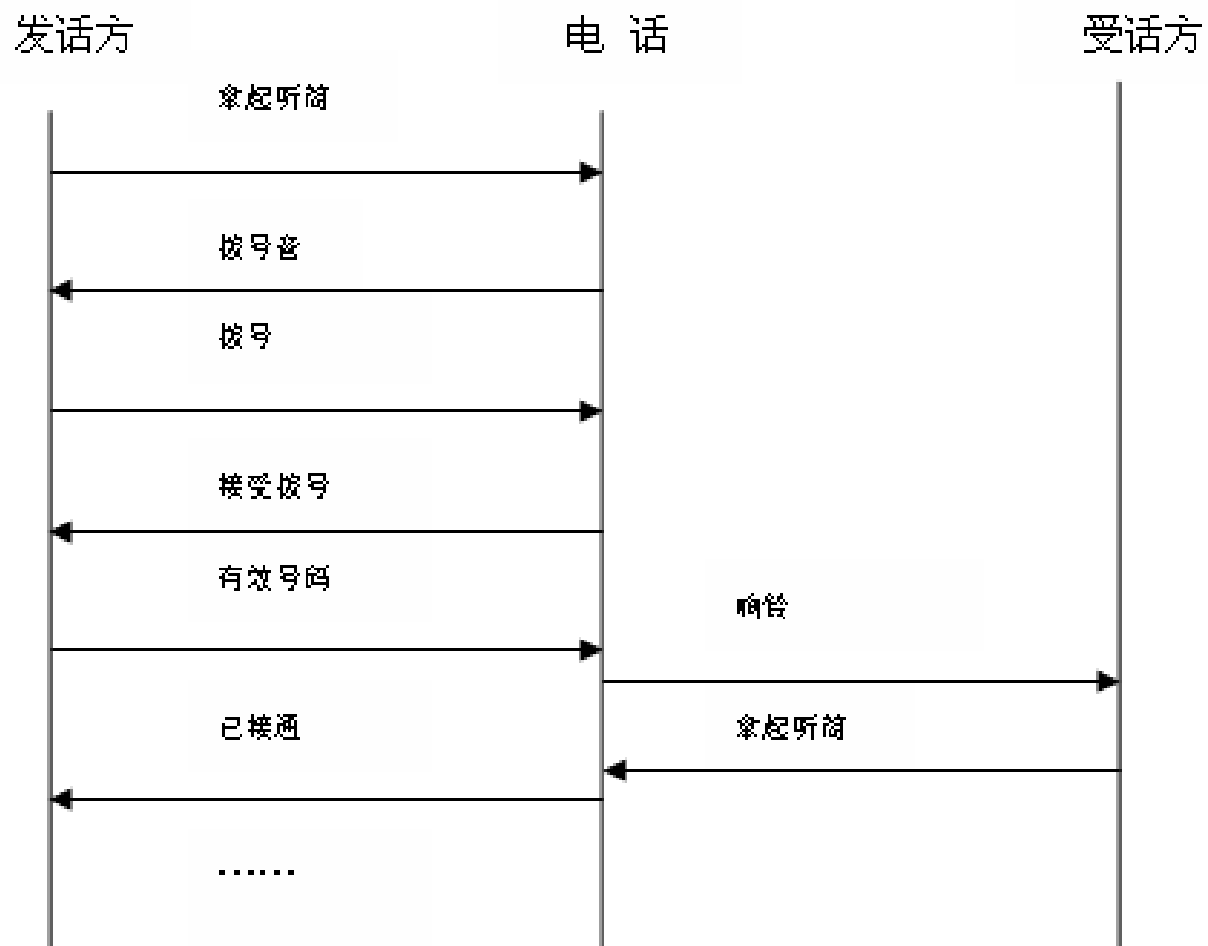
- 用一条**竖直线**表示一个对象或类
- 用一条**水平的带箭头的直线**表示一个事件，**箭头方向是从发送事件的对象指向接受事件的对象。**
- 事件按产生的时间从上向下逐一系列出。



## OMT方法的图形描述工具(续)

### ■ 序列图





电话系统的序列图



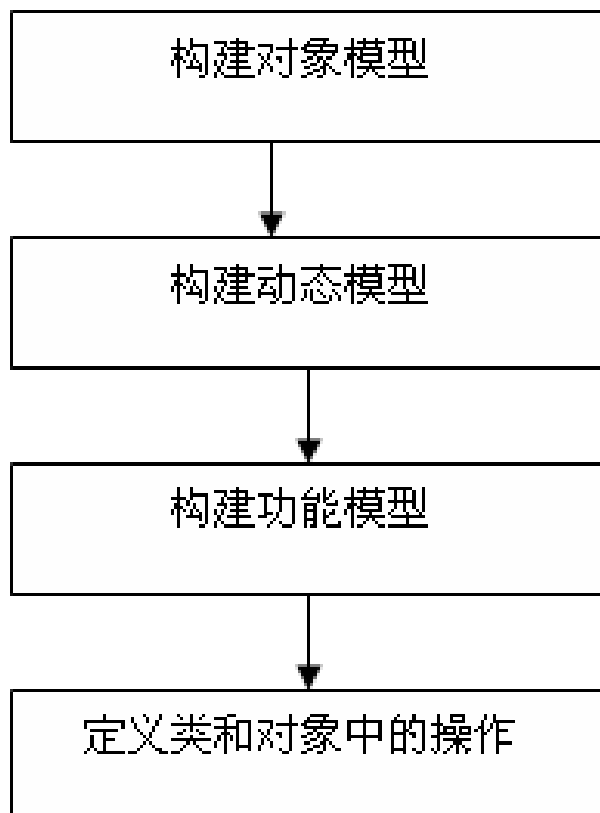


## 基于OMT方法的需求建模(续)

- OMT方法的图形描述工具
- **基于OMT方法的需求建模步骤**



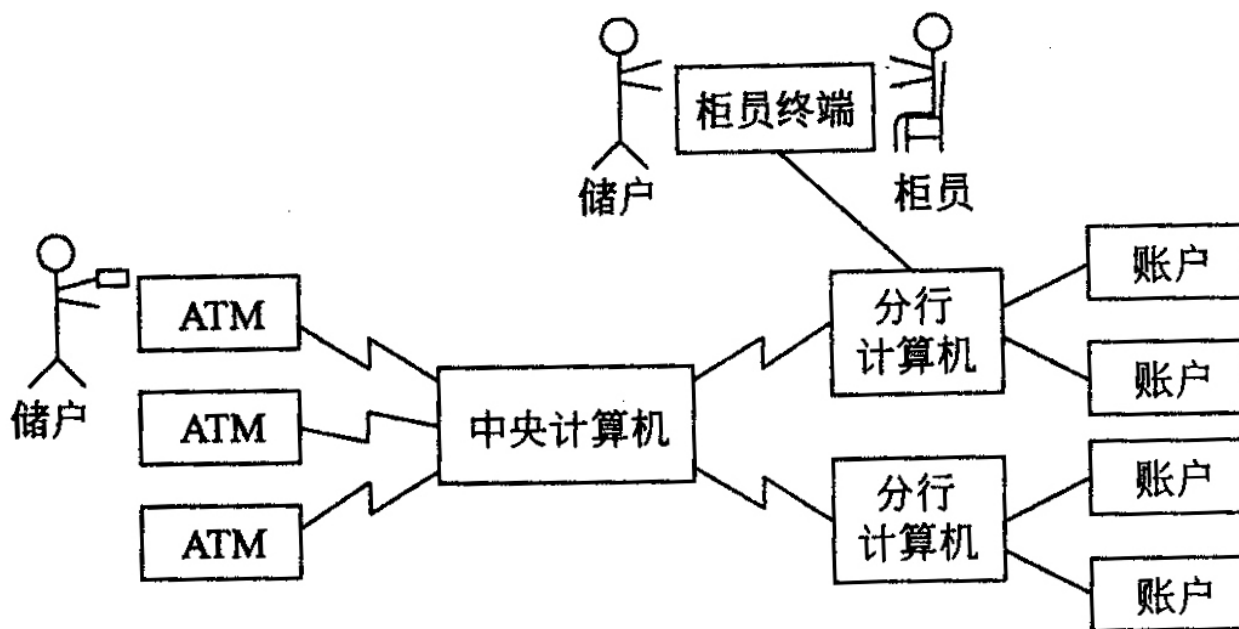
## 基于OMT方法的需求建模步骤





## 基于OMT方法的需求建模步骤(续)

### ■ 示例：ATM系统





# 基于OMT方法的需求建模步骤(续)

## ■ 示例：ATM系统

- 1) 某银行拟开发一个自动取款机系统，它是一个由自动取款机、中央计算机、分行计算机及柜员终端组成的网络系统。ATM和中央计算机由总行投资购买。总行拥有多台ATM，分别设在全市主要街道上。分行负责提供分行计算机和柜员终端，柜员终端设在分行营业厅及分行下属的各个储蓄所内。该系统的软件开发成本由各个分行分摊。
- 2) 银行柜员使用柜员终端处理储户提交的储蓄事务。柜员负责把储户提交的存款或取款事务输进柜员终端，接收储户交来的现金或支票，或付给储户现金。柜员终端与相应的分行计算机通信，分行计算机具体处理针对某个账户的事务并且维护账户。
- 3) 储户可以用现金或支票开设新账户。储户也可以从自己的账户存款或取款。通常，一个储户可能拥有多个账户。拥有银行账户的储户有权申请领取银行卡。使用银行卡可以通过ATM访问自己的账户、提取现金，存储现金或查询有关自己账户的信息。



# 基于OMT方法的需求建模步骤(续)

## ■ 示例：ATM系统

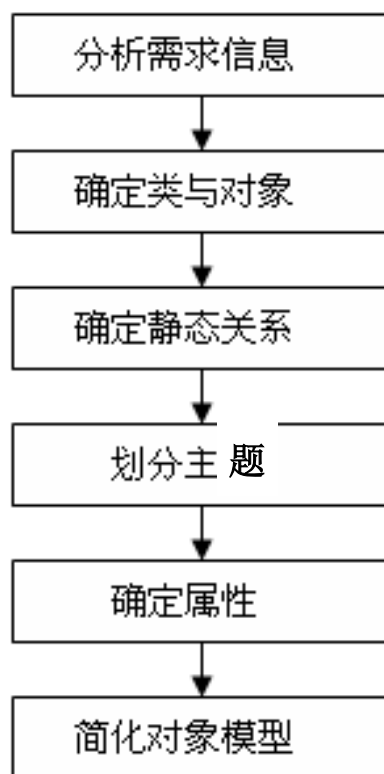
4) 银行卡是一张特制的磁卡，上面有分行代码和卡号。分行代码唯一标识总行下属的一个分行，卡号确定可以访问哪些账户。每张银行卡仅属于一个储户，但同一张卡可能由多个副本。因此，必须考虑同时在若干台ATM上使用同样的银行卡的可能性。也就是说，系统应该能够处理并发的访问。

5) 当用户把银行卡插入ATM之后，ATM就与用户交互，获取有关这次事务的信息，并与中央计算机交换有关事务的信息。首先，ATM要求用户输入密码，接下来ATM把读到的信息以及用户输入的密码传给中央计算机，请求中央计算机核对这些信息并处理这次事务。中央计算机根据卡上的分行代码确定这次事务与分行的对应关系，委托相应的分行计算机验证用户密码。如果用户输入的密码是正确的，ATM就要求用户选择用户选择事务类型（取款、查询等）。当用户选择取款时，ATM请求用户输入取款项。最后，ATM从现金出口吐出现金，打印出账单交给用户。



# 基于OMT方法的需求建模步骤(续)

## 1. 构建对象模型





# 基于OMT方法的需求建模步骤(续)

## 1. 构建对象模型

### (1) 分析需求信息

从需求信息中分析出哪些是系统必要的需求信息，  
以及与系统相关的边界等



# 基于OMT方法的需求建模步骤(续)

## 1. 构建对象模型

### (2) 确定类与对象

寻找问题域中存在的客观实体，然后筛选出实际需要的实体，视其为类与对象

- **非正式分析**：把陈述中的名词作为类与对象的候选者
- **严格筛选**：删除不正确的和不必要的候选实体





# 基于OMT方法的需求建模步骤(续)

## 1. 构建对象模型

### (2) 确定类与对象

ATM示例：

#### ● 非正式分析：

银行、ATM、中央计算机、分行计算机、柜员终端、网络、总行、分行、街道、储蓄所、营业厅、柜员、储户、现金、支票、账户、事务、银行卡、余额、磁卡、分行代码、卡号、用户、信息、密码、类型、取款额、账单等

#### ● 严格筛选：

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、柜员事务、远程事务、银行卡



# 基于OMT方法的需求建模步骤(续)

## 1. 构建对象模型

### (3) 确定静态关系

分析问题域中实体之间的复杂关系，确定类或对象之间的相互依赖或相互作用关系

- 提取需求陈述中的动词来确认对象或类之间的关系
- 分析问题或与用户及有关专家交流，发现隐含的关系



## 1. 构建对象模型

### (3) 确定静态关系

#### ATM实例

##### ● 提取动词短语得出的关系

- ATM、中央计算机、分行计算机及柜员终端组成网络
- 总行拥有多台ATM
- ATM设在主要街道上
- 分行提供分行计算机和柜员终端
- 柜员终端在分行营业厅及储蓄所内
- 储户拥有账户
- 分行计算机处理针对账户的事务
- 柜员输入针对账户的事务
- ATM与中央计算机交换关于事务的信息
- 中央计算机确定事务与分行的对应关系
- ATM读银行卡
- ATM与用户交互
- ATM吐出现金
- ATM打印账单



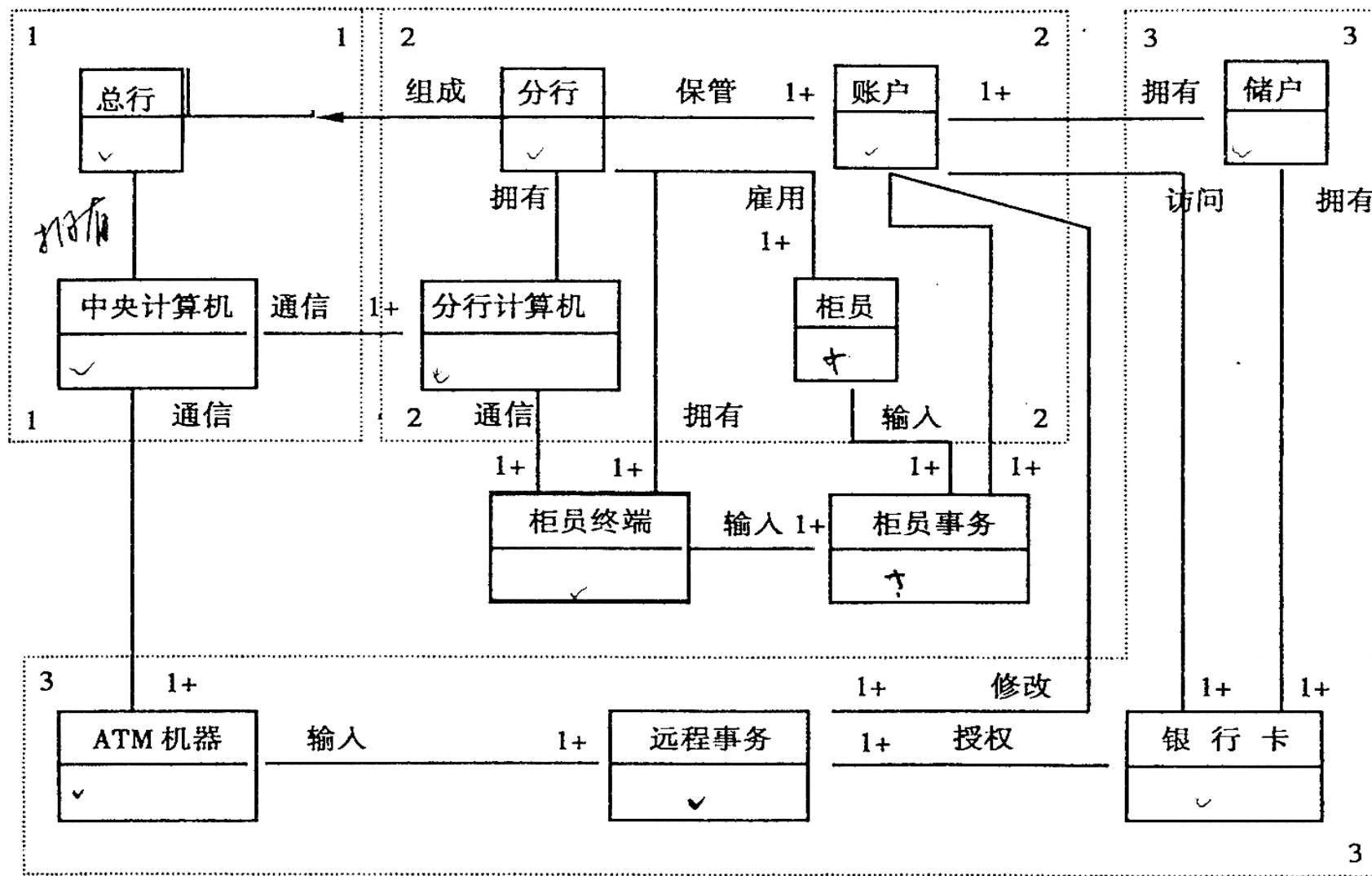
## 1. 构建对象模型

### (3) 确定静态关系

#### ATM实例

##### ● 分析得出的隐含关系

- 总行由各个分行组成
- 分行保管账户
- 总行拥有中央计算机
- 系统维护事务日志
- 系统提供必要的安全性
- 银行卡访问账户
- 分行雇佣柜员



ATM系统初始类图



# 基于OMT方法的需求建模步骤(续)

## 1. 构建对象模型

### (4) 划分主题

用主题来划分大型复杂软件系统的范围，以降低系统的复杂程度

- 按问题域而不是按功能分解方法来确定主题
- 应考虑不同主题内的对象间相互依赖和交互最少

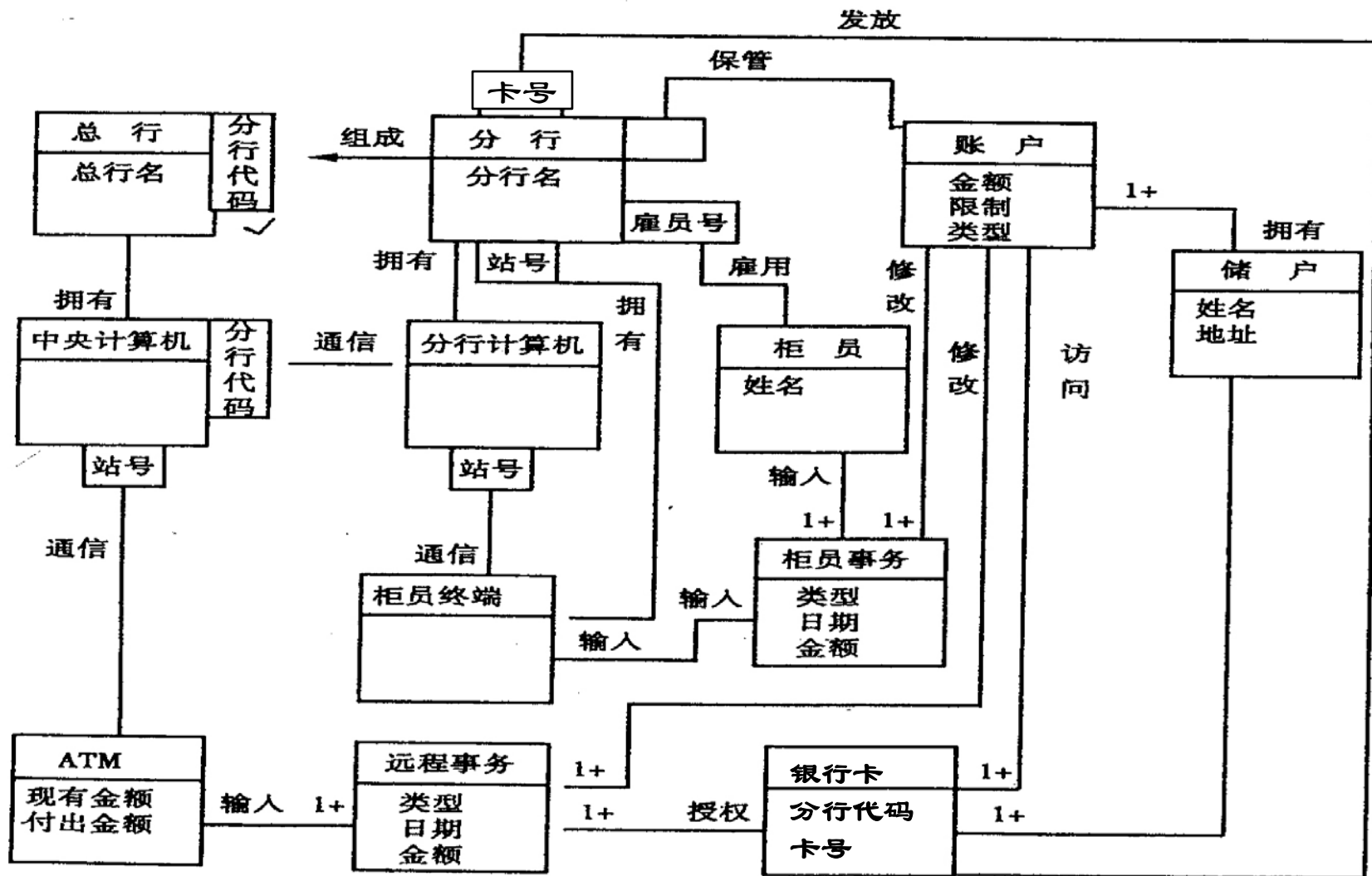


# 基于OMT方法的需求建模步骤(续)

## 1. 构建对象模型

### (5) 确定属性

- 在需求陈述中根据名词词组、形容词找出属性
- 回答以下问题得到有关属性
  - ① 一般情况下，需要哪些信息描述对象或类？
  - ② 在当前问题中，需要哪些信息描述对象或类？
  - ③ 按照系统的要求，需要哪些信息描述对象？
  - ④ 该对象需要保持哪些信息？
  - ⑤ 该对象需要记录哪些状态信息？
  - ⑥ 该对象提供的服务中需要哪些信息？
  - ⑦ 用什么属性表示“整体-部分”结构和实例连接？



带属性标识的ATM系统的对象模型





# 基于OMT方法的需求建模步骤(续)

## 1. 构建对象模型

### (6) 简化对象模型

利用**继承关系**，并按照自顶向下和自底向上的方式对许多的类进行**重组**，实现类的共享并达到简化对象模型的目的

- 以自底向上的方式抽象出若干个类具有的共同性质并组成父类
- 以自顶向下的方式把一个类细化为更具体的类



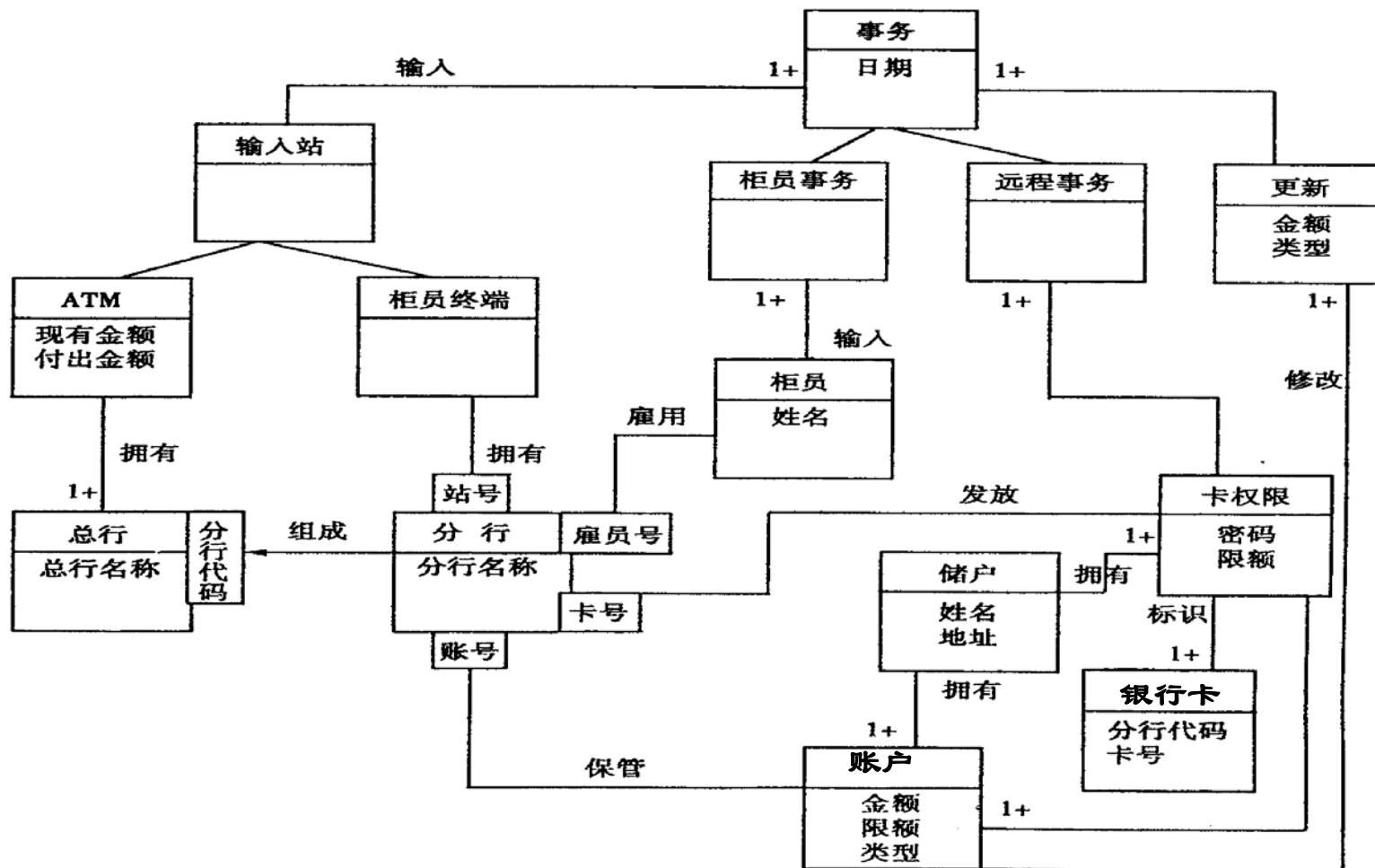
# 基于OMT方法的需求建模步骤(续)

## 1. 构建对象模型

### (6) 简化对象模型

ATM实例：

- “远程事务”和“柜员事务”抽象出“事务”父类
- “ATM”和“柜员终端”抽象出“输入站”父类
- “银行卡”类分解为“卡权限”和“银行卡”两个类
- “事务”类由“更新”类组成
- “分行”类与“分行计算机”类合并
- “总行”类与“中央计算机”类合并

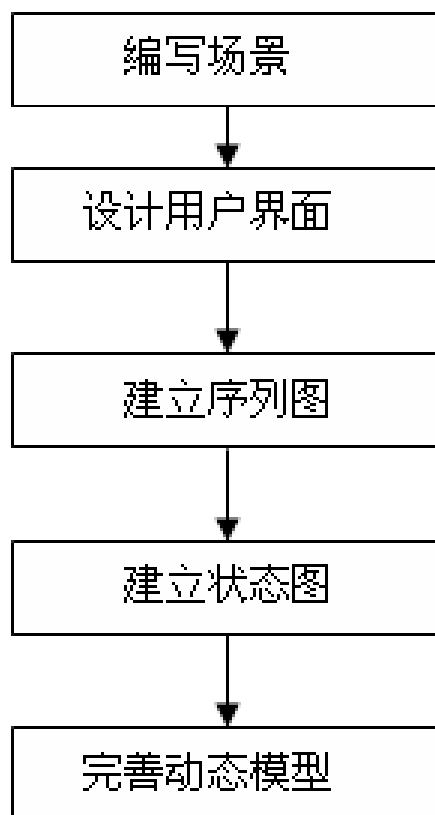


简化后的ATM系统的对象模型



# 基于OMT方法的需求建模步骤(续)

## 2. 构建动态模型





# 基于OMT方法的需求建模步骤(续)

## 2. 构建动态模型

### (1) 编写场景

用自然语言来描述用户与软件系统之间一个或多个交互行为的过程。

场景的编写过程实质上是分析用户对系统交互行为的需求的过程

- ① 首先编写正常情况的脚本
- ② 再考虑特殊情况，如输入/输出的最大值情况
- ③ 最后考虑出界情况，如输入值为非法值



- ATM请储户插卡；储户插入一张银行卡。
- 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，如“123456”等数字。
- ATM请求总行验证卡号和密码；总行要求分行核对储户密码，然后通知ATM说这张卡有效。
- ATM要求储户选择事务类型（取款、存款、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“1000”。
- 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号的新余额。
- ATM吐出现金，请求储户取现金；储户拿起现金。
- ATM问储户是否继续本次事务；储户回答“不”。
- ATM打印账单，退出银行卡，请求储户取卡；储户取走账单和卡。

## ATM系统的正常情况场景



- ATM请储户插卡；储户插入一张银行卡。
- ATM接受这张卡并顺序读它上面的数字。
- ATM要求密码；储户误输入“88888”
- ATM请求总行验证输入数字和密码；总行在向有关分行询问之后拒绝这张卡。
- ATM显示“密码错”，并请储户重新输入密码；储户输入“123456”；ATM请总行验证后知道这次输入的密码正确。
- ATM请储户选择事务类型；储户选取“取款”。
- ATM询问取款额；储户改变主意不想取款了，他敲“取消”键。
- ATM退出银行卡，请求储户取卡；储户拿走他的卡。

## ATM系统的异常情况场景



# 基于OMT方法的需求建模步骤(续)

## 2. 构建动态模型

### (2) 设计用户界面

在设计用户界面时，用户界面的细节并不太重要，重要的是在这种界面下的信息交换方式。

- 有助于用户对系统的理解和与用户的交流
- 确保能够完成全部必要的信息交换，不会丢失重要的信息





# 基于OMT方法的需求建模步骤(续)

## 2. 构建动态模型

### (2) 设计用户界面

| 向用户显示的信息             |   |                      |   |     |
|----------------------|---|----------------------|---|-----|
| 0                    | 1 | 2                    | 3 | 4   |
| 5                    | 6 | 7                    | 8 | 9   |
| 确 认                  |   | 取 消                  |   | 清 除 |
| <input type="text"/> |   | <input type="text"/> |   |     |
| 账单出口                 |   | 现金出口                 |   |     |

ATM初步的用户界面



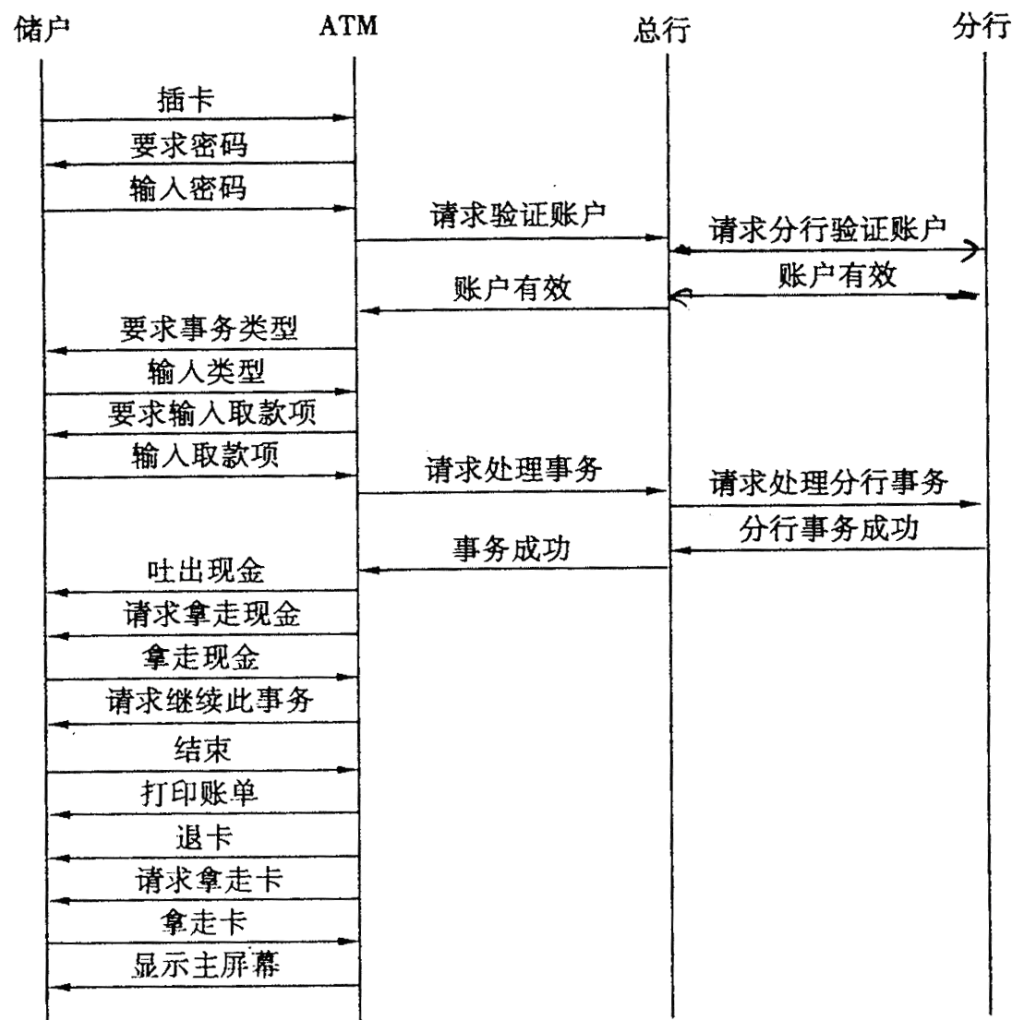
# 基于OMT方法的需求建模步骤(续)

## 2. 构建动态模型

### (3) 建立序列图

序列图实质上是场景的图形表示，**每个场景对应一张序列图**

- ① 首先应认真分析每个场景的内容，从中提取所有外部事件信息及异常事件和出错条件的信息。
- ② 利用序列图将事件序列以及事件与对象间的关系清晰和形象地表示出来。



ATM系统正常情况场景的序列图



# 基于OMT方法的需求建模步骤(续)

## 2. 构建动态模型

### (4) 建立状态图

状态图适用于表示动态模型，刻画事件与对象状态之间的关系

- ① 确定序列图中对象的初态
- ② 分析序列图，把序列图中标记的事件作为状态图中的有向边，边上标记事件名，某对象类两个输入事件之间就代表该对象类的一个状态，以此类推，直至到达该对象的终止状态；

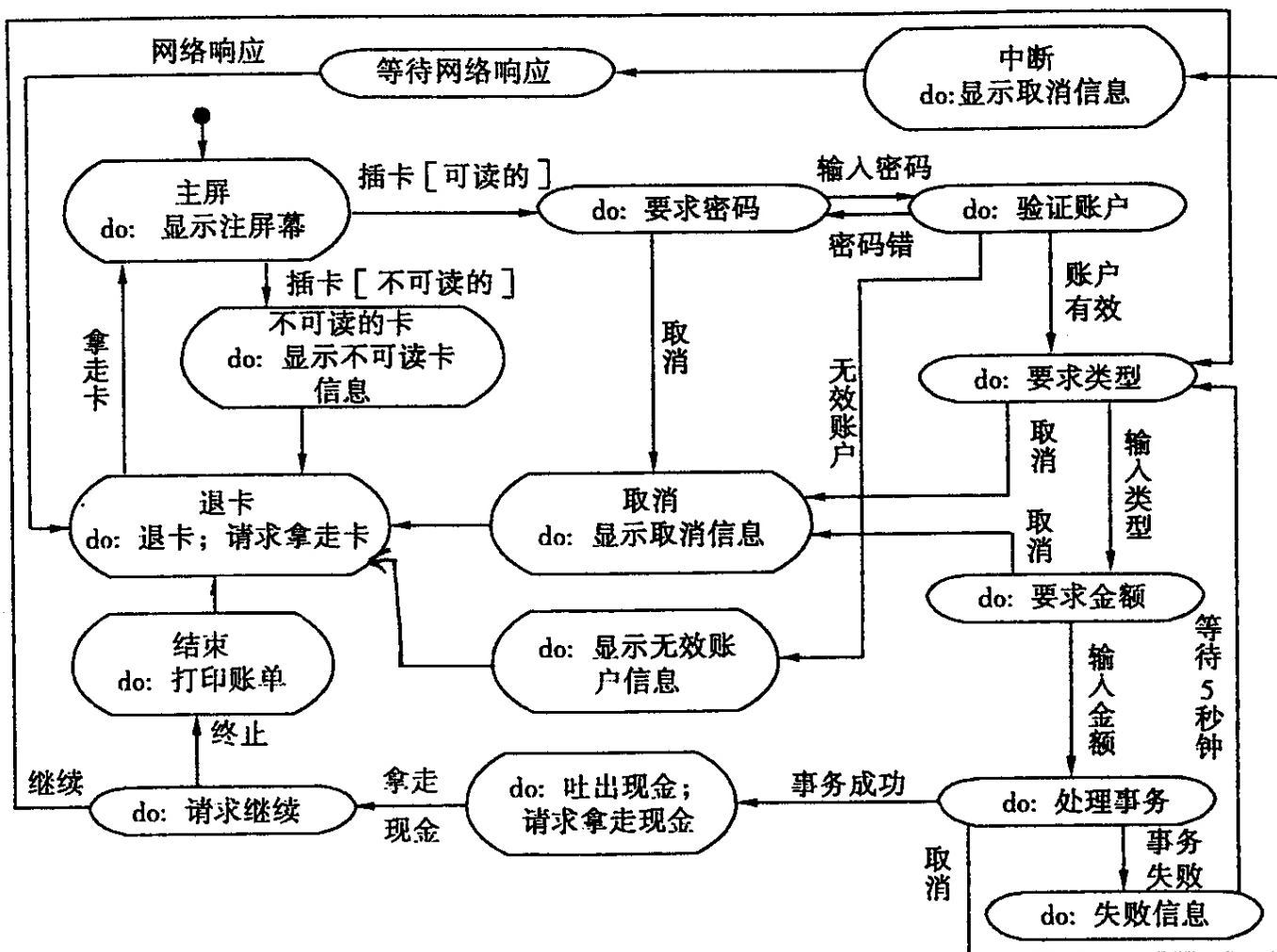


# 基于OMT方法的需求建模步骤(续)

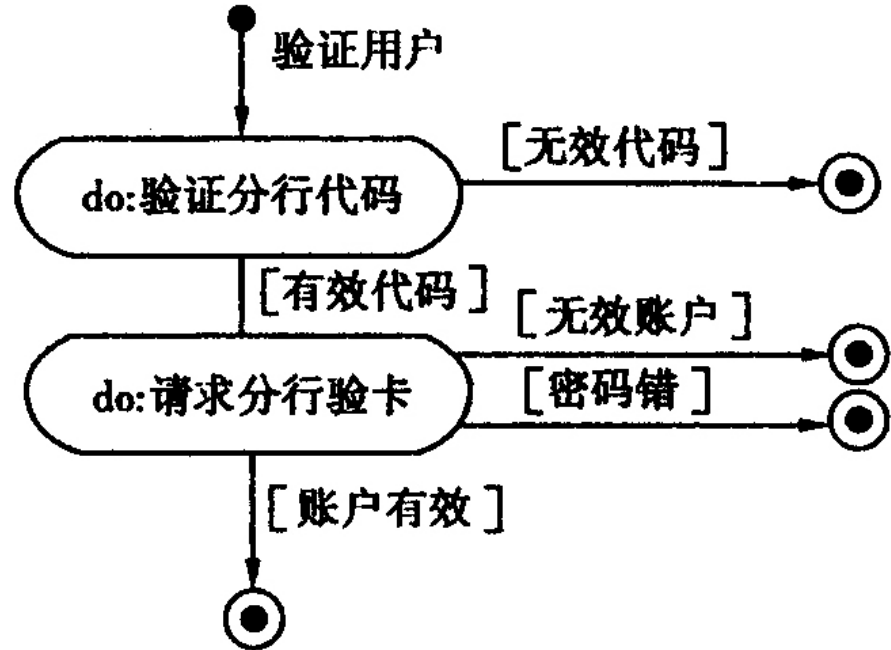
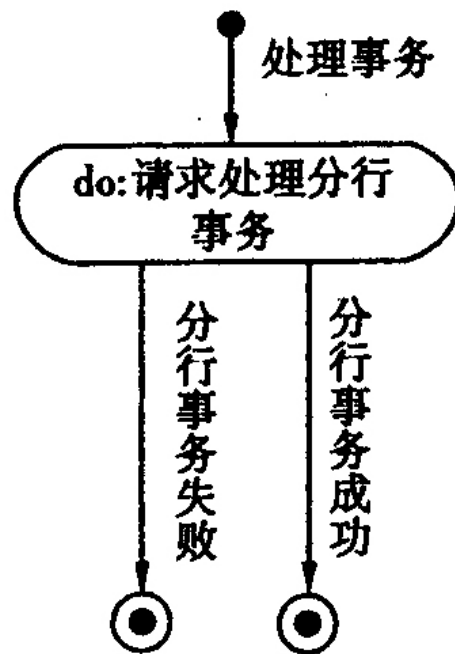
## 2. 构建动态模型

### (4) 建立状态图

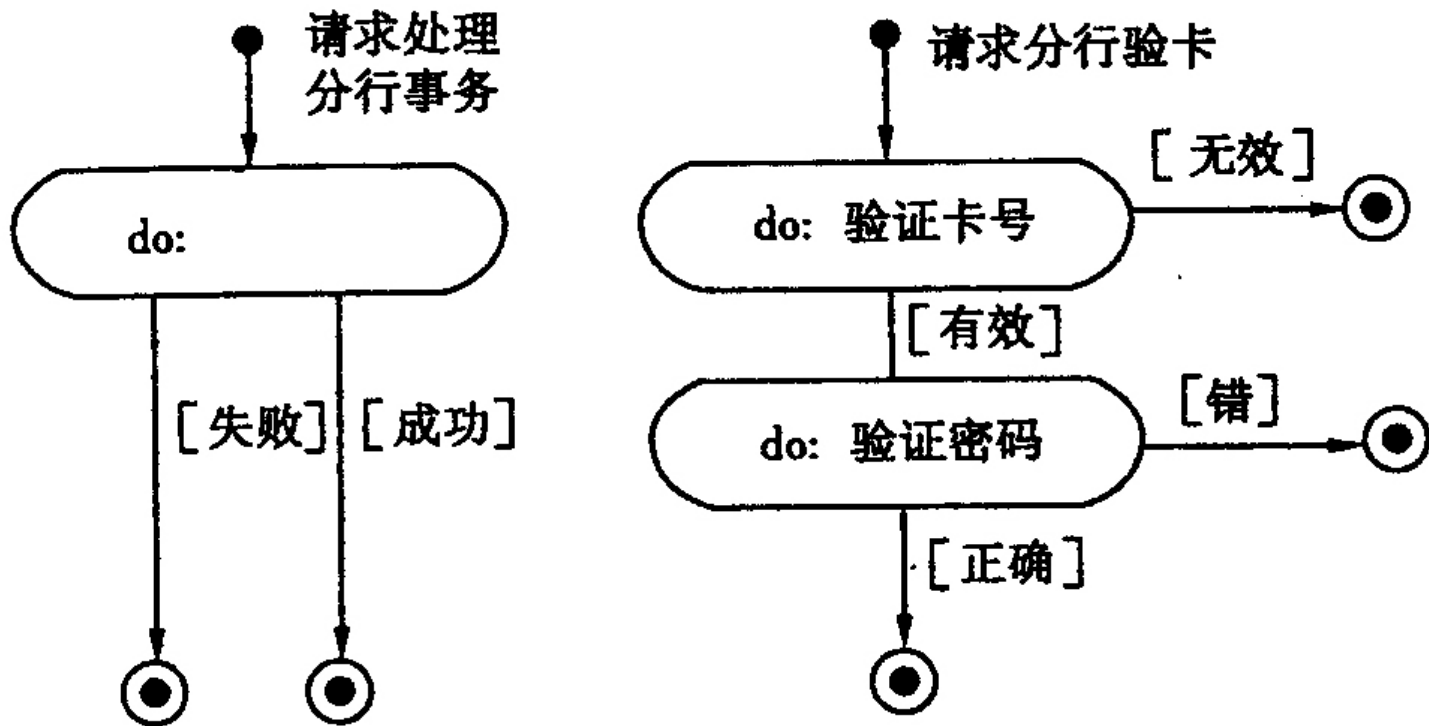
- ③ 正常事件描述之后，还应考虑边界情况下可能发生的事件；
- ④ 根据一张序列图画对象或类状态图之后，再把其它与该对象或类相关的场景（如异常情况场景）的序列图加入到已画出的状态图中。



ATM系统的部分状态图



总行的状态图



分行的状态图





## 基于OMT方法的需求建模步骤(续)

### 3. 构建功能模型

- 功能模型主要表达系统内部数据流的传递和处理的过程。
- 数据流图适用于描述系统的功能模型。
- 好处：有助于软件开发人员更深入地理解问题域，修改和完善自己的设计



## 基于OMT方法的需求建模步骤(续)

### 4. 定义类和对象中的操作

- 目的：建立一个完整的对象模型
- 动态模型和功能模型为确定类和对象的操作提供了基本依据



## 基于OMT方法的需求建模步骤(续)

### 4. 定义类和对象中的操作

- 定义类和对象中操作的基本原则：
  - 基本的属性操作
  - 事件的处理操作；
  - 完成数据流图中处理框对应的操作；
  - 利用继承机制优化服务集合，减少冗余服务。



## 面向对象的需求建模方法(续)

- 5.3.1 面向对象的需求分析
- 5.3.2 基于OMT方法的需求建模
- **5.3.3 基于UML的需求建模**



## 5.3.3 基于UML的需求建模(续)

- UML以各种图形描述为主分别表示面向对象方法中的不同方面的模型
- 按静态结构和动态结构分类：
  - 静态结构类：用例图、类图、构件图等；
  - 动态结构类：状态图、活动图、序列图、协作图和配置图等。



## 基于UML的需求建模(续)

- 活动图
- 协作图



## 活动图

- UML的活动图是用于表示系统控制流的，是状态图的特殊形式。
- 活动图与流程图类似，区别在于：
  - 不仅可用于程序设计级，还能用于描述概念级的模型
  - 能描述并行动作



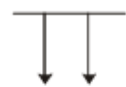
## 活动图(续)

● 启动

● 终止

○ 动作

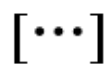
◇ 选择



并行分支(fork)



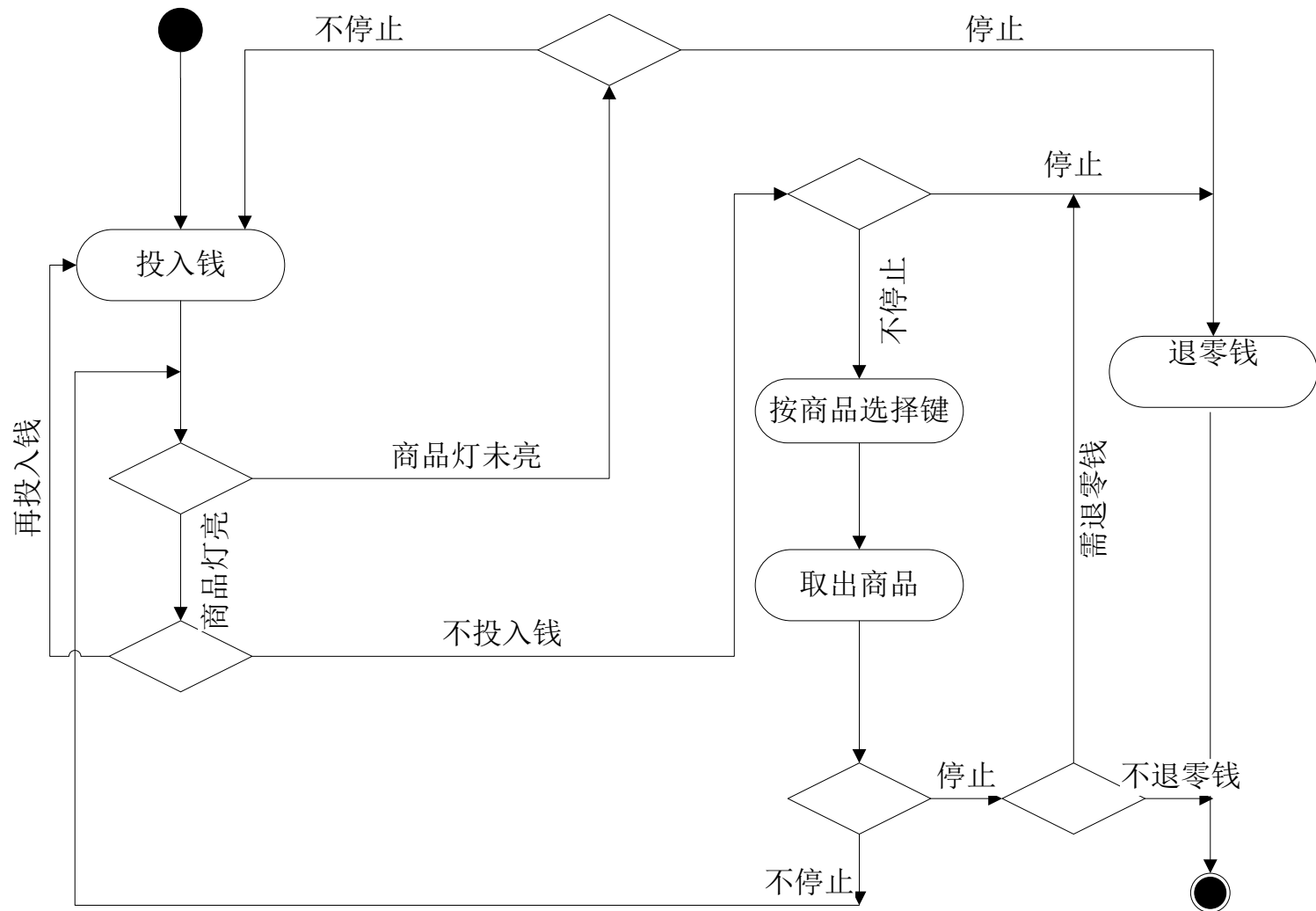
并行合成(join)



卫士条件(guard)

活动图的构成元素



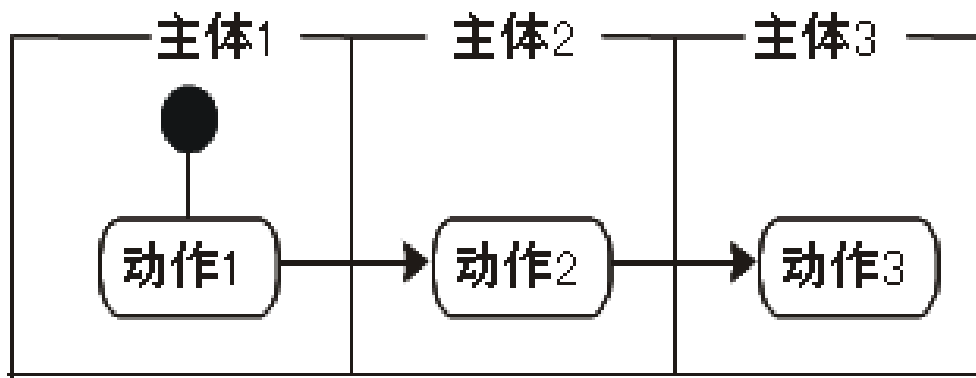


自动贩卖机的活动图



## 活动图(续)

- 为了明确地表示活动的主题，可使用标识主体的描述方式。



- 当把活动图视为图形时，其路径可解释为执行路径
- 活动图可表示为层次结构。



## 基于UML的需求建模(续)

- 活动图
- 协作图

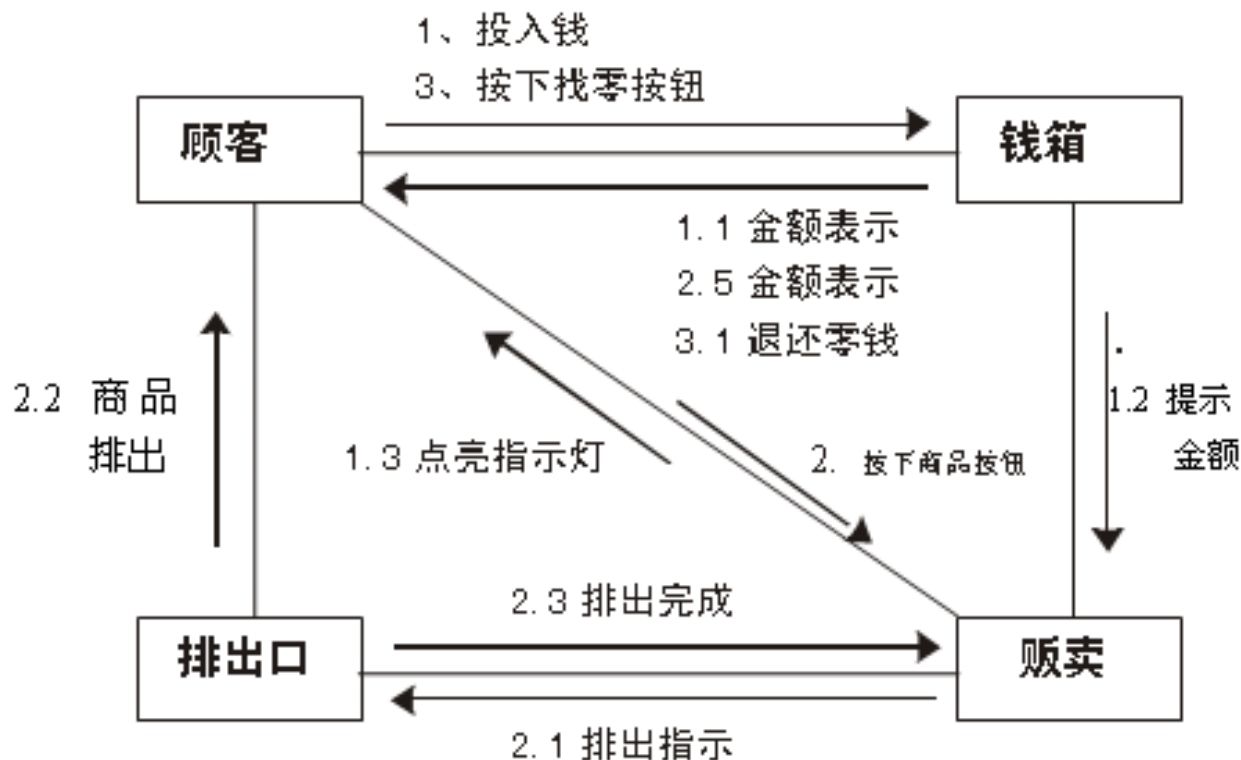


## 协作图

- 协作图用于表示对象间的消息往来。
- 协作图便于描述对象间有什么样的协作关系，不需要像一个序列图只能对应于一个场景一样，可以将多个场景中的协作关系一次性地全部描述出来。



## 协作图(续)



自动贩卖机的活动图