



# 软件工程

## 第六章 需求分析与建模

乔立民

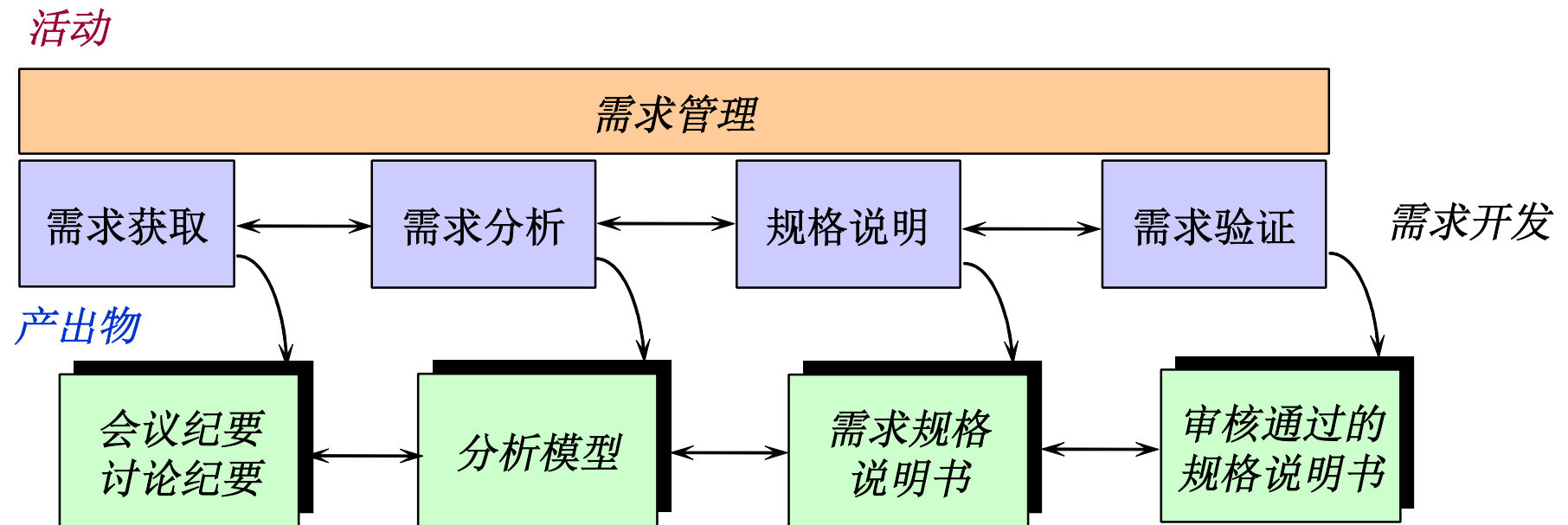
[qlm@hit.edu.cn](mailto:qlm@hit.edu.cn)

2011年4月27日

# 主要内容

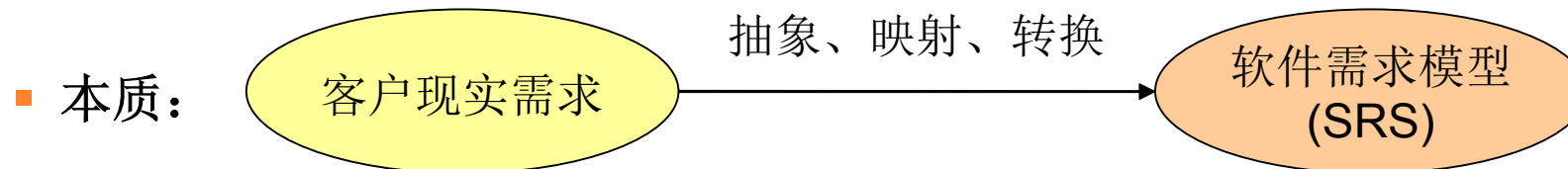
- **6.1** 需求分析
- **6.2** 面向对象需求分析建模
- **6.3** 结构化需求分析建模

# 需求工程的总体流程



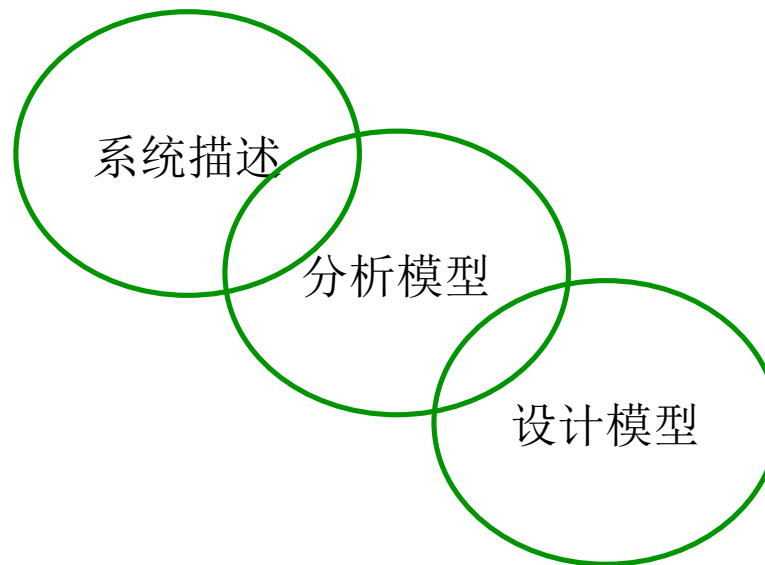
# 需求分析的本质

- **需求分析(Requirement Analysis)**: 对收集到的需求进行提炼、分析和审查, 为最终用户所看到的系统建立概念化的分析模型
  - 分析需求可行性
  - 细化需求
  - 建立需求分析模型
    - 功能活动
    - 分析问题类和类之间关系
    - 系统和类行为
    - 数据流



# 分析模型的目标

- 描述客户需要什么（软件的信息、功能和行为）
- 为软件设计奠定基础（结构、接口、构件设计）
- 定义在软件完成后可以被确认的一组需求



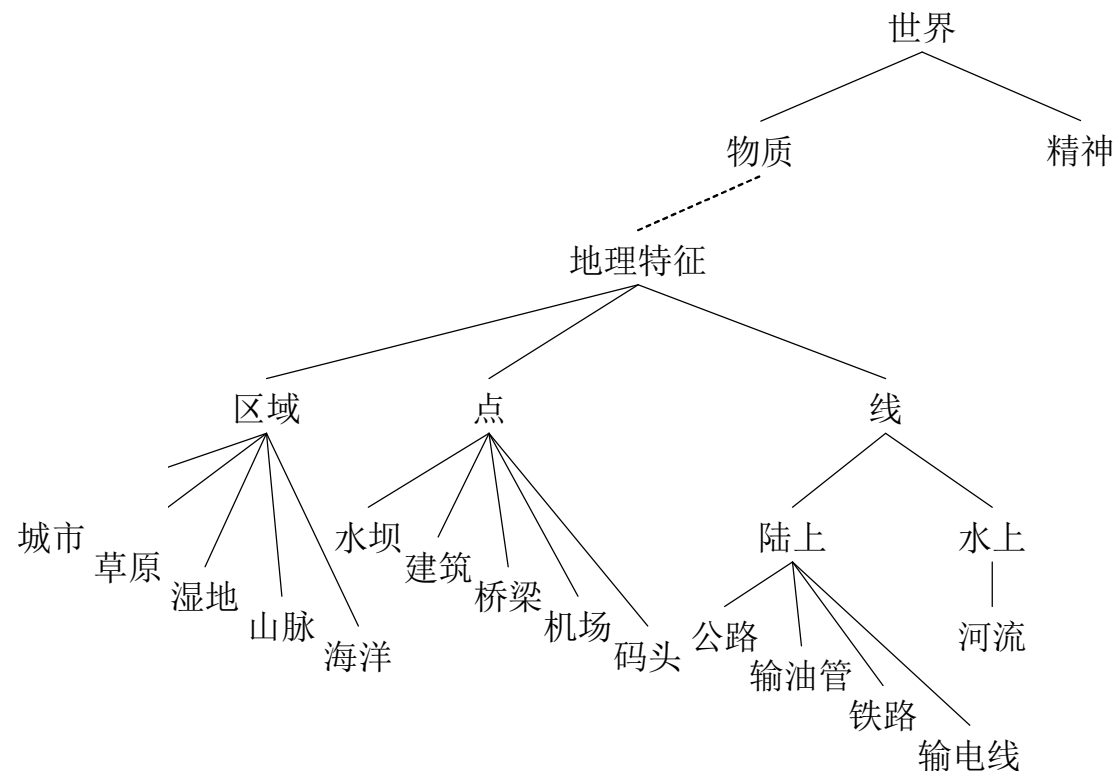
# 分析的经验原则

- 模型应关注在问题域或业务域内可见的需求，抽象的级别应该相对高一些。不需要陷入细节，即不要试图解释系统将如何工作
- 分析模型的每个元素都应该能增加对软件需求的整体理解，并提出对信息域、功能和系统行为的深入理解
- 关于基础结构和其他非功能的模型应推延到设计阶段再考虑
- 最小化整个系统内的关联
- 确认分析模型为所有共利益者都带来价值（客户、设计人员、测试人员）
- 尽可能保持模型简洁

# 需求分析的基本思想：抽象

## ■ 抽象：透过现象看本质

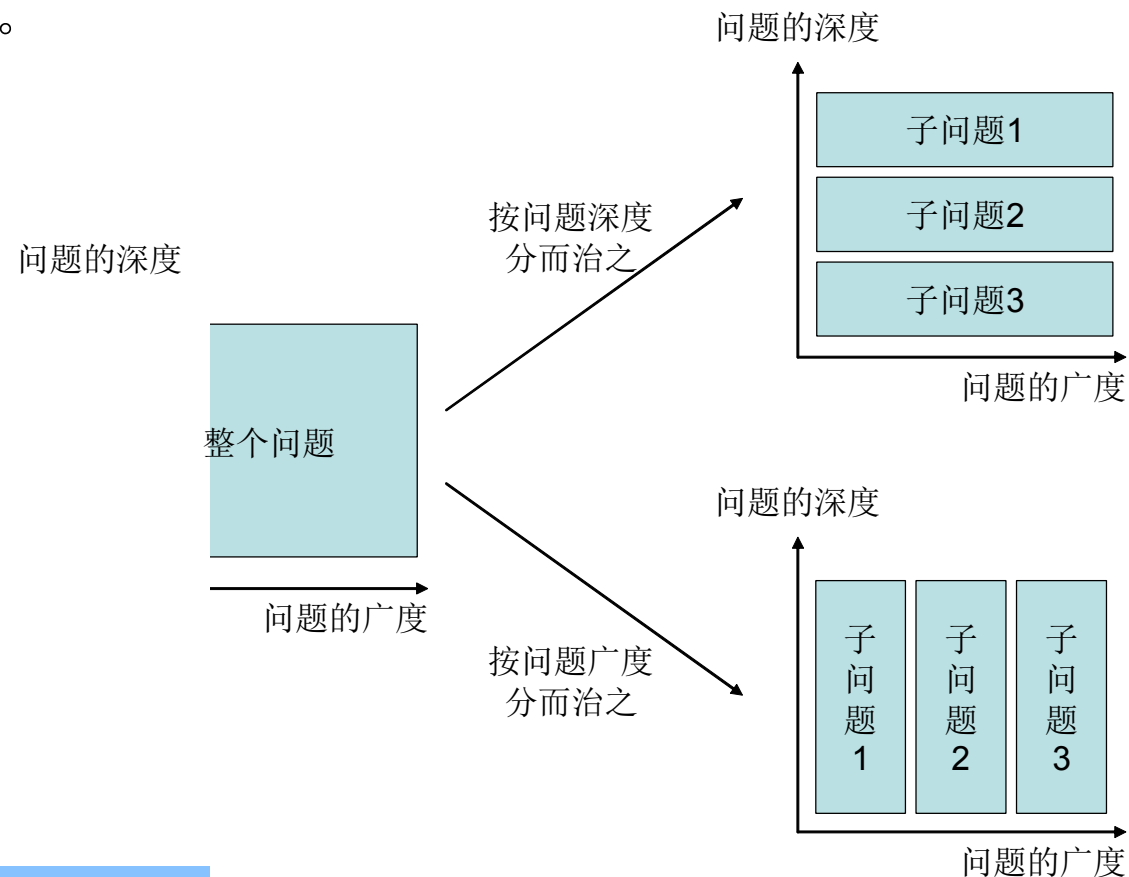
- 抓住事物的本质，捕获问题空间的“一般/特殊”关系是认识、构造问题的一般途径。



# 需求分析的基本思想：划分

## ■ 划分：分而治之

- 分离问题，捕获问题空间的“整体/部分”关系是降低问题复杂性的基本途径。

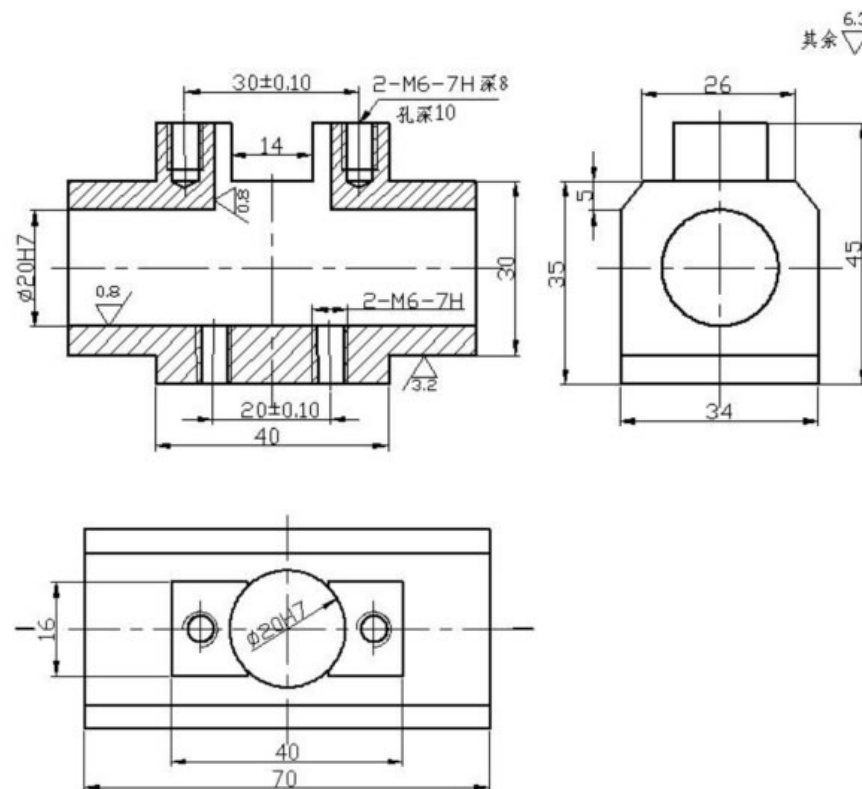




# 需求分析的基本思想：投影

## ■ 投影：从不同视角看问题

— 捕获并建立问题空间的多维视图是描述问题的基本手段。



# 需求分析的基本思想：建模

## ■ 建模：规格严格、功夫到家

- 采用规范的描述方法，将模糊的、不确定的用户需求表达为清晰的、严格的模型，作为进一步设计与实现的基础。
- 模型的作用：
  - 增强对需求的理解
  - 检测不一致性、模糊性、错误和遗漏
  - 在项目的参与者之间更高效的交流

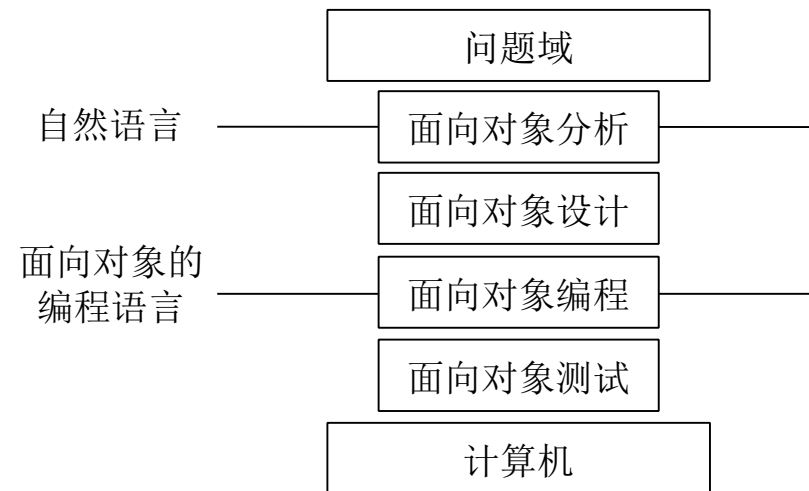
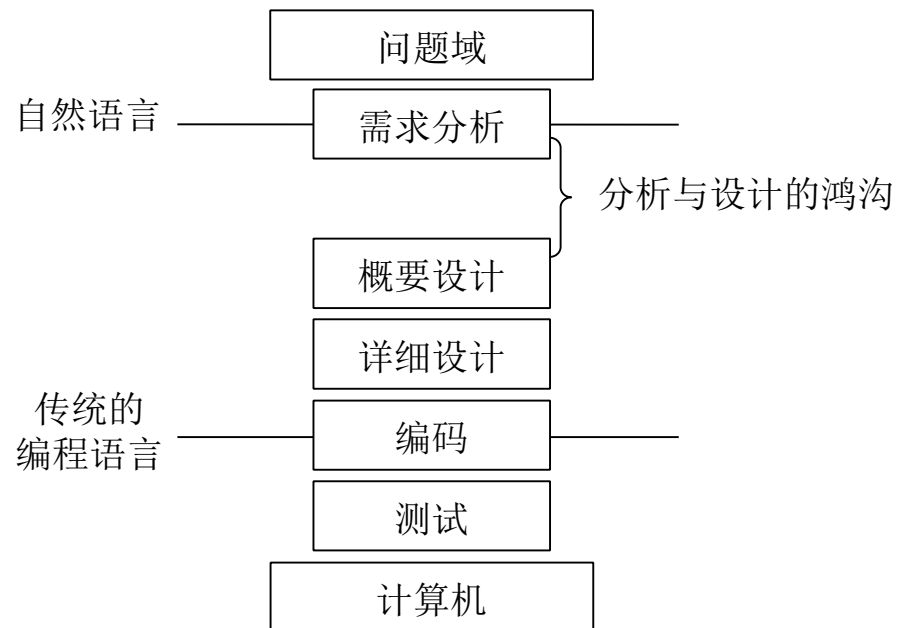
## ■ 两种模型形态：

- 形式化的数学模型(formal mathematical model)
- 非形式化的图形化模型 (informal graphical model)

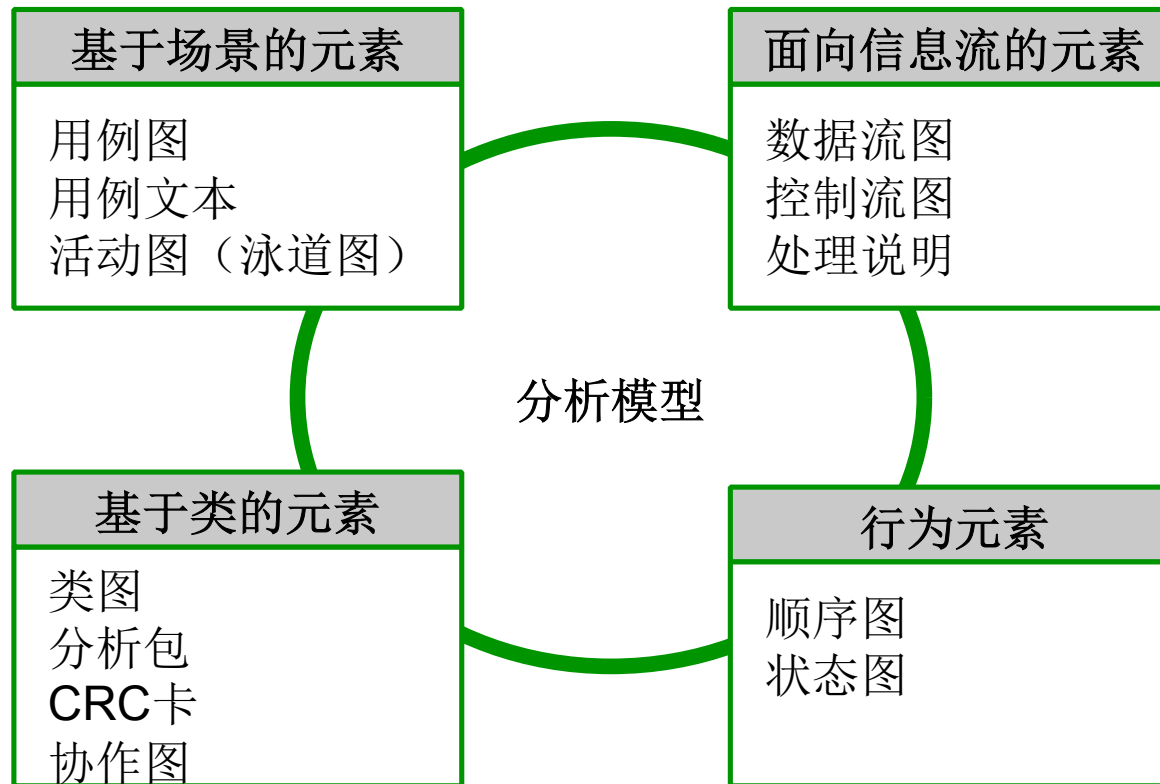
# 需求分析方法

- 两种主要需求分析方法:

- 结构化分析与设计方法(Structured Analysis and Design Technique, SADT)
- 面向对象分析与设计方法(Object-Oriented Analysis and Design, OOAD)



# 分析模型的元素



# 主要内容

- **6.1 需求分析**
- **6.2 面向对象需求分析建模**
  - 6.2.1 面向对象需求分析
  - 6.2.2 静态分析建模
  - 6.2.3 动态分析建模
- **6.3 结构化需求分析建模**

# 面向对象的分析

- 理解由问题陈述所描述的真实世界的系统，并把它的功能抽象成模型
- 分析模型描述对象三个方面：
  - 对象的静态结构（领域模型）
  - 对象之间的交互（交互模型）
  - 对象的生存期（状态模型）



# 面向对象的分析

- 面向对象的分析模型由三个独立的模型构成：

- 功能模型：从用户的角度获取功能需求，由用例模型表示(已在上堂课学习过)；

- 静态结构模型(领域模型)：是对领域内的概念类或现实世界中对象的可视化表示；

- 动态行为模型：描述对象之间的交互行为，由顺序图和协作图表示。

建立与实现  
技术无关的  
系统逻辑结  
构

# 面向对象的分析

面向对象的分析  
OOA

面向对象的设计  
OOD

面向对象的实现  
OOP

面向对象的测试  
Testing

需求模型  
从用户的角度  
获取功能需求

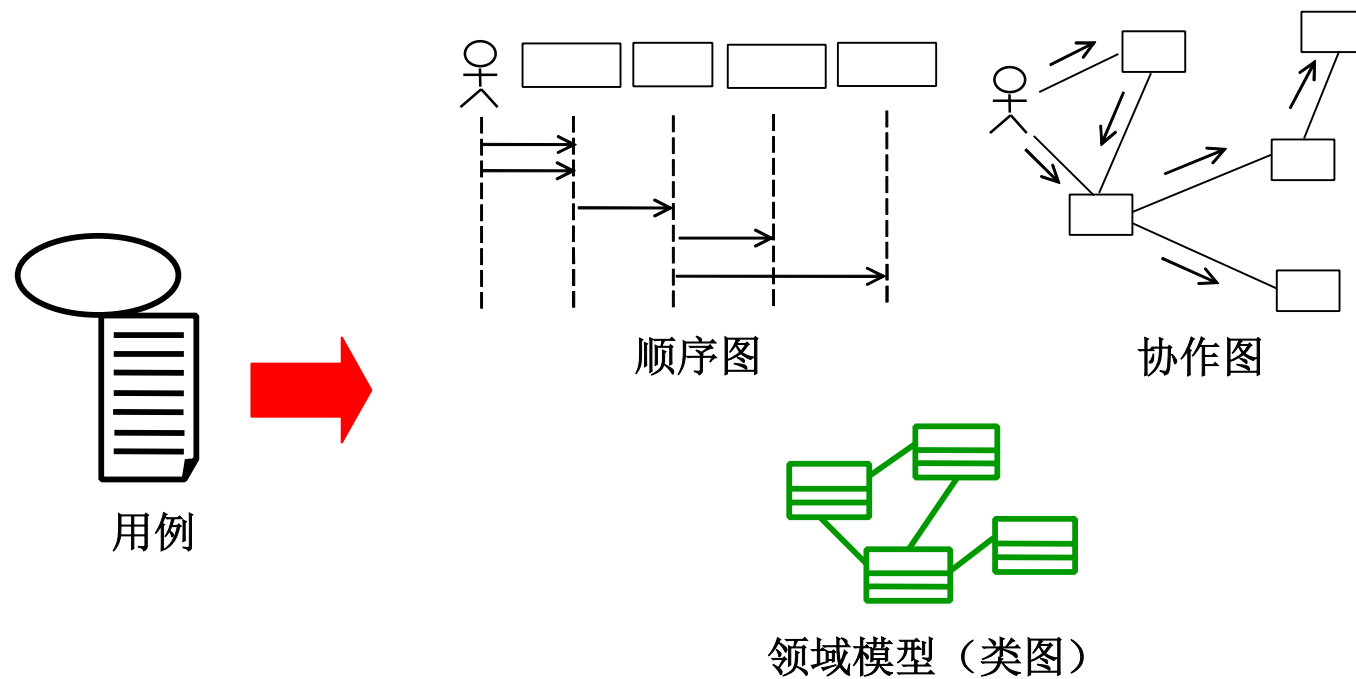
(用例模型)

分析模型  
建立系统的概念模型  
，细化系统行为

(  
1 静态结构模型：领域模型、类图  
2 动态行为模型：顺序图、状态图、协作图  
)



# 面向对象的分析



# 面向对象分析的过程

- 第一阶段：建立静态模型（领域建模）
- 第二阶段：建立动态模型

# 主要内容

- **6.1 需求分析**
- **6.2 面向对象需求分析**
  - 6.2.1 面向对象需求分析
  - 6.2.2 静态分析建模
  - 6.2.3 动态分析建模
- **6.3 结构化需求分析**

## 面向对象分析的过程

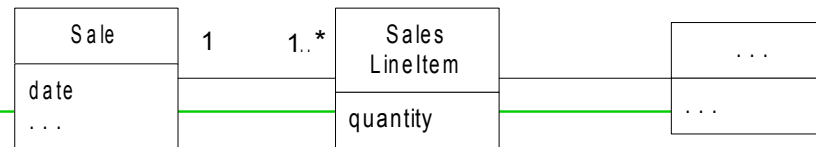
- 第一阶段：建立静态模型（领域建模）
  1. 从用例文本入手，寻找概念类
  2. 细化概念类，识别边界类、控制类和实体类
  3. 添加关联和属性

# 领域模型的作用

业务建模

制品关系样例

领域模型



概念类、术语、概念、属性、关联

经历状态变化的领域对象、属性和关联

领域模型中某些术语的细化

需求

Use-Case Model

销售过程

1. 顾客到达 ...
2. ...
3. 收银员输入商品 ID .
4. ....

用例文本

Operation: enterItem(...)

Post-conditions :  
- ....

操作契约

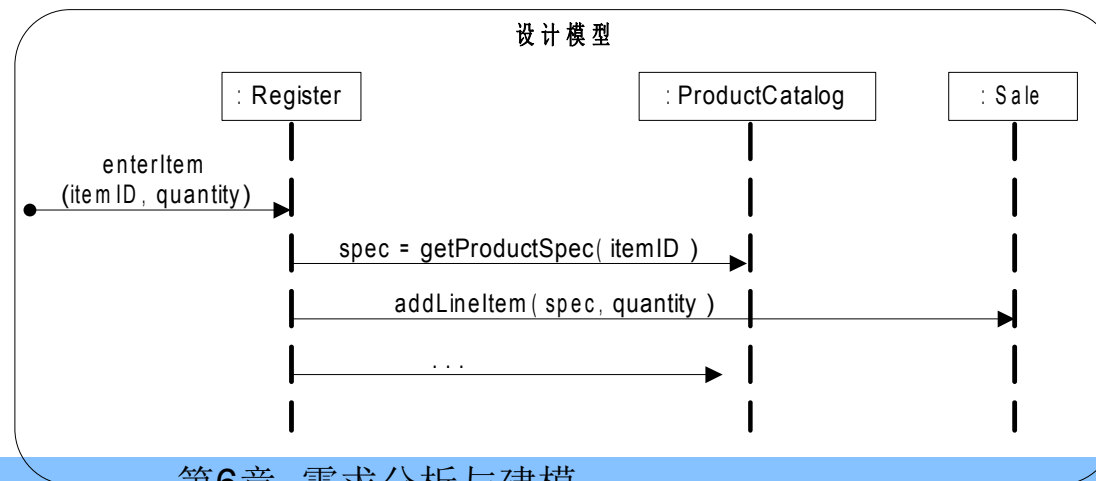
收银员: ...  
商品 ID: ...  
...

词汇表

领域中的概念会给某些设计软件类的名称带来启示

设计模型

设计



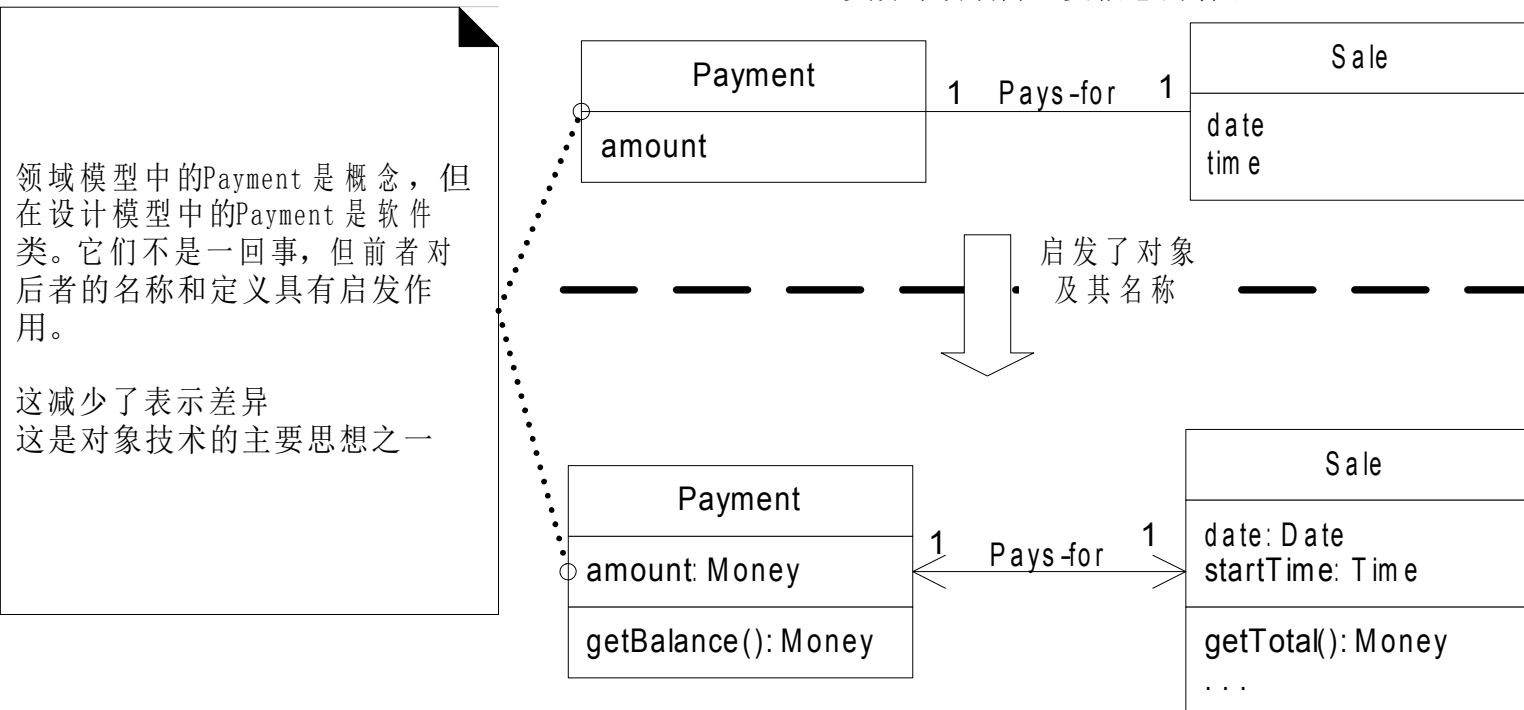
# 领域建模

- 发现和描述对象（或概念）是面向对象的核心
- 领域模型又称为概念对象模型，领域模型展示重要的领域概念或对象
- 领域模型描述的是真实世界的概念，而非软件对象的概念
- 领域模型不是数据模型

# 领域建模



- 通过领域建模能够降低与OO建模之间的表示差异



因此，涉众所设想的领域与其在软件的表示之间的表示差异被降低

# 1.寻找概念类

- **方法1：重用和修改现有的模型**
  - 参考已有系统或书籍资料等
- **方法2：使用分类列表**
  - 业务交易
  - 角色
  - 记录
  - 交易项目
  - 地点
  - 物理对象
- **方法3：确定名词短语**
  - 分析用例文本



# 寻找概念类——确定名词短语

用例名称：处理销售

参与者与关注点：

- 收银员：希望准确、快速地输入，而且没有支付错误，因为如果少收货款，将从其工资中扣除。
- ... ..

前置条件：收银员必须经过确认和认证

成功保证（或后置条件）：存储销售信息。准确计算税金。更新账务和库存信息。

主成功场景（或基本流程）：

- 1.顾客携带所购商品或服务到收银台通过POS机付款。
- 2.收银员开始一次新的销售交易。
- 3.收银员输入商品条码

扩展（或替代流程）：

3a.无效商品ID（在系统中未发现）：

系统提示错误并拒绝输入该ID。

收银员响应该错误

特殊需求：

- 使用大尺寸平面显示器触摸屏，文本信息可见距离为1米
- ... ..

发生频率：可能会不断地发生

未解决问题：

提成处理规则不确定

收银员换班时如何处理

... ..

## 选择概念类的标准

1. 固定信息
  2. 必要的服务
  3. 多属性
  4. 公共属性
  5. 公共操作
  6. 基本需求
- **+状态,+行为**, 独立对象
  - **状态,-行为**
    - 系统需要它的状态数据, 其他对象的属性
    - 否则, 摒弃
  - **+行为,-状态**, 信息遗漏
  - **-行为,-状态**, 摒弃

# 选择概念类示例

## 用例描述：

1. 顾客携带商品到销售终端POS前.
2. 收银员开始一个新的销售处理.
3. 收银员输入物品项标识.
4. 系统记录销售的物品项列表并且显示物品描述、价格和总价.  
    收银员重复步骤3-4，直至输入所有物品项。
5. 系统显示最后的总价.
6. 收银员告诉顾客总价，要求顾客支付账款.
7. 顾客付款，系统结账.
8. 系统记录整个销售处理，更新产品库存目录.
9. 系统打印收据.
10. 顾客离开.

a )

## 确定对象：

顾客，商品，POS，收银员，销售处理，物品项列表，物品描述，账款，产品目录

## 摒弃对象：

物品项标识：只有状态没有行为  
价格：只有状态没有行为  
总价：只有状态没有行为  
收据：既无状态也无行为

b )

## 概念类：

顾客，  
商品，  
POS，  
收银员，  
销售处理，  
物品项列表，  
物品描述，  
账款，  
产品目录

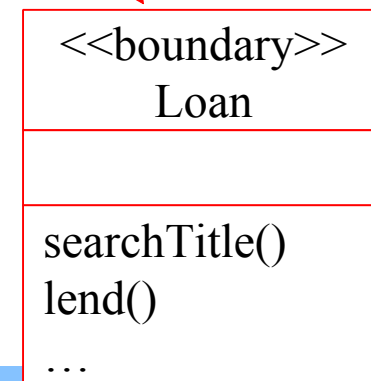
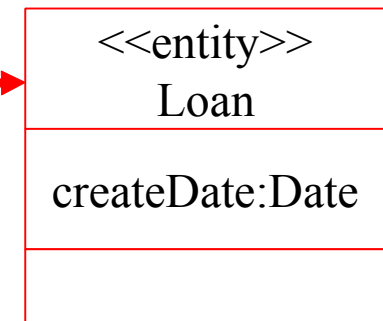
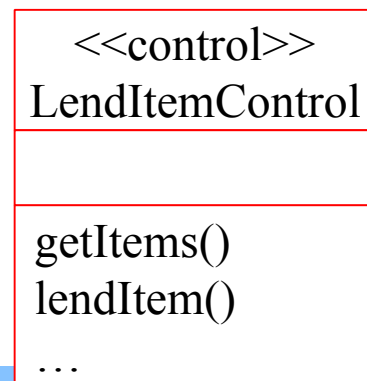
c )

# 概念类



## 2.细化概念类（分析类）

- 概念类是概念层次上的内容，用于描述系统中较高层次的对象
- 概念类直接与应用逻辑相关，而不关注于技术实现的问题
- 概念类的类型
  - 实体类：表示系统存储和管理的持久性信息
  - 边界类：表示参与者与系统之间的交互
  - 控制类：负责用例实现



# 边界类(Boundary Class)

- 边界类:

- 描述外部的参与者与系统之间的交互
- 目的: 将用例的内部逻辑与外部环境进行隔离, 使得外界的变化不会影响到内部的逻辑部分。
- 类型: 用户界面、系统接口、设备接口

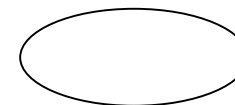
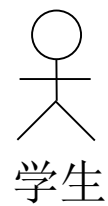
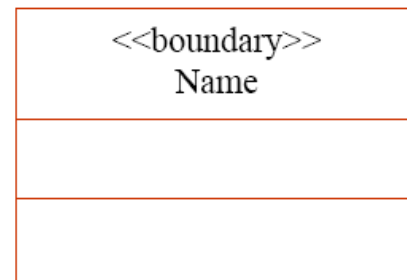
- 对用户界面来说:

- 描述用户与系统的交互信息(传入哪些信息/指令, 传出哪些信息/指令), 而不是用户界面的显示形式(如按钮、菜单等);

- 对系统接口/设备接口来说:

- 描述通信协议, 但不必说明协议如何实现的。

# 边界类(Boundary Class)



课程注册



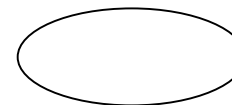
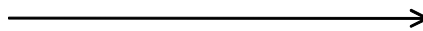
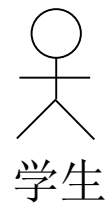
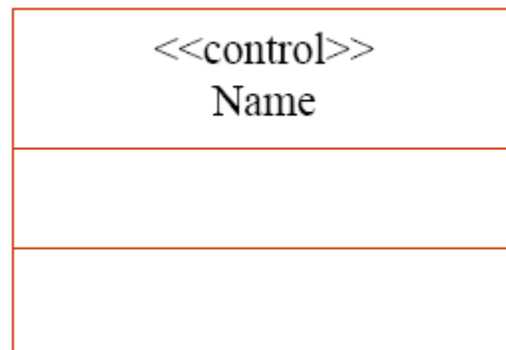
“学生注册课程”的界面

## 控制类(Control Class)

- 描述一个用例所具有的事件流的控制行为;
- 实现了对用例行为的封装, 将用例的执行逻辑与边界和实体进行隔离, 使得边界类和实体类具有较好的通用性。



# 控制类(Control Class)



课程注册

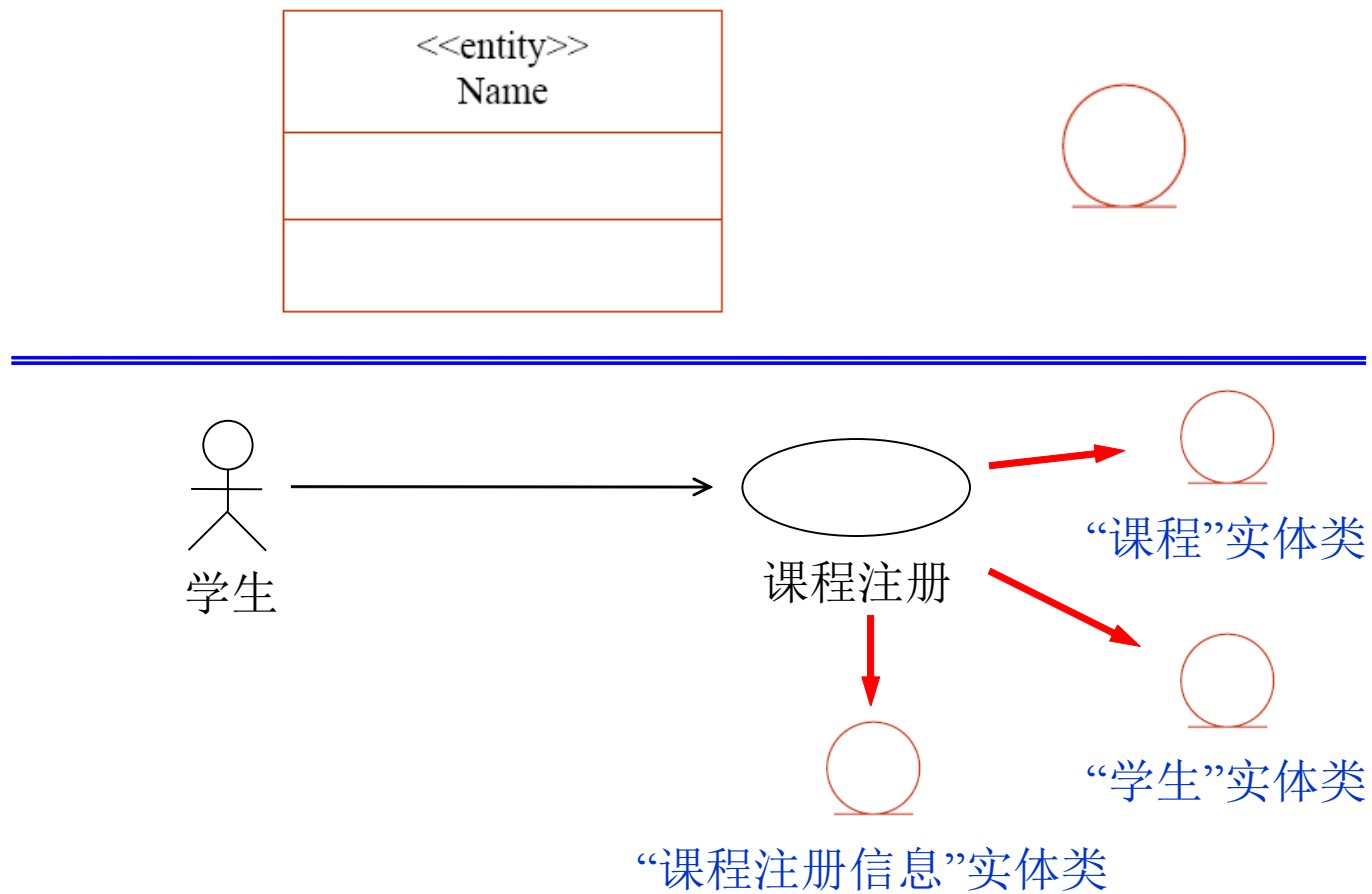


“课程注册”控制类

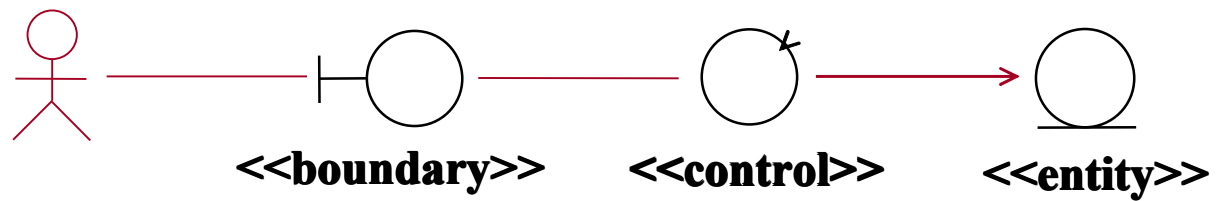
## 实体类(Entity Class)

- 描述必须存贮的信息及其相关行为
- 对系统的核心信息建模，通常这些信息需要长久的保存
- 通常对应现实世界中的“事物”

# 实体类(Entity Class)

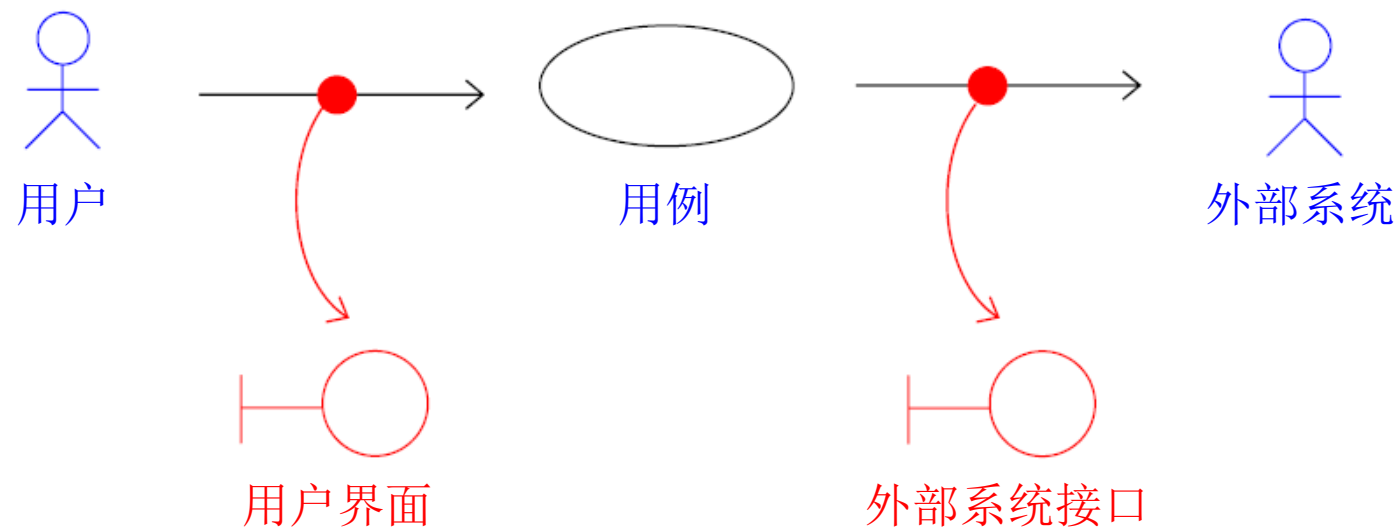


## 三种分析类之间的关系

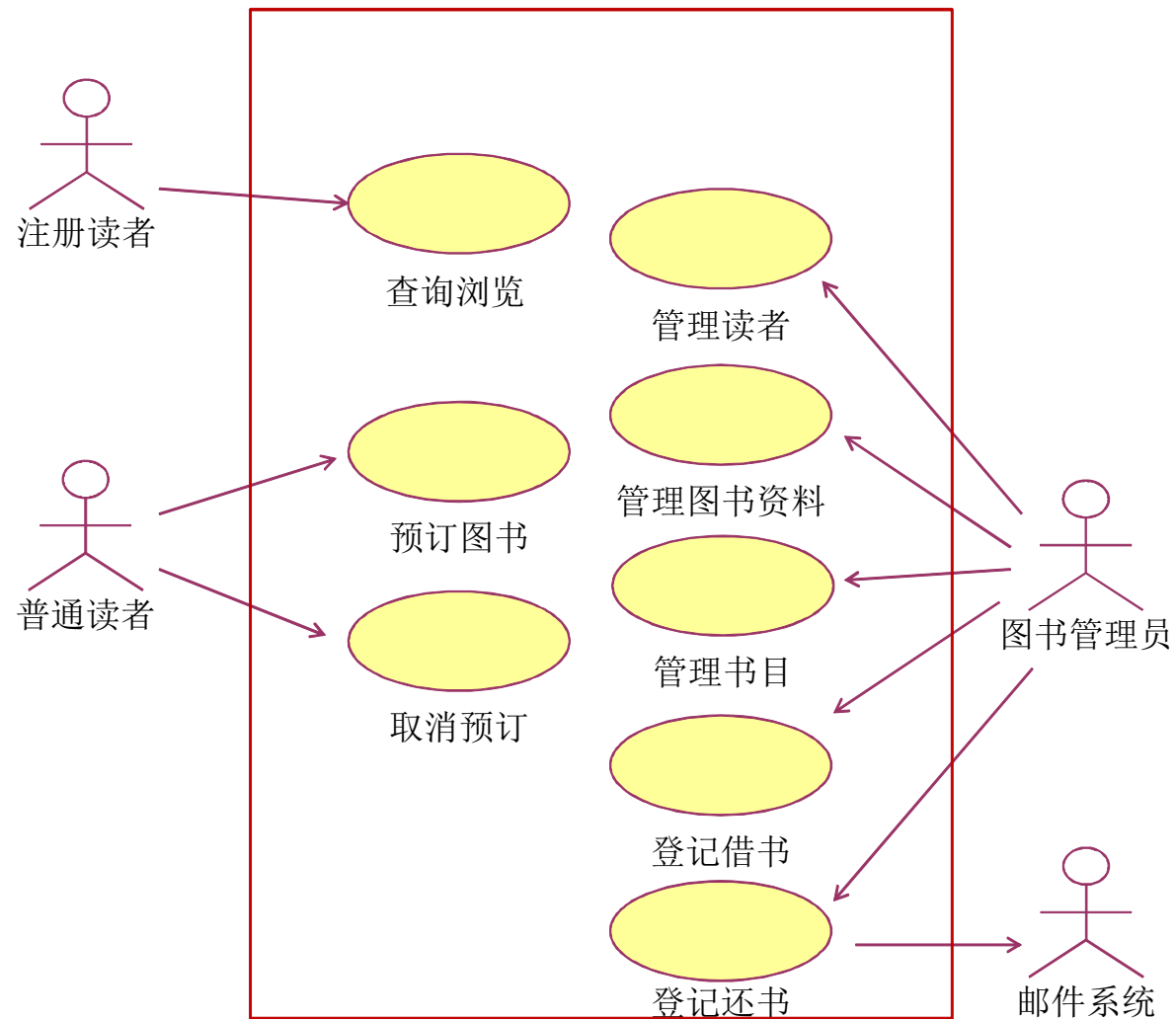


# 识别边界类









- 通常，一个参与者与一个用例之间的交互或通信关联对应一个边界类



## [案例]图书管理系统的用例图



## [案例]图书管理系统的边界类

边界类		说明
<b>LoginForm</b>		注册用户进行登录的操作界面
<b>BrowseForm</b>		注册用户进行查询浏览的操作界面
<b>MakeReservationForm</b>		普通读者预定图书的操作界面
<b>RemoveReservationForm</b>		普通读者取消预定的操作界面
<b>ManageBrowsersForm</b>		图书管理员管理读者的操作界面
<b>ManageTitlesForm</b>		图书管理员管理图书资料的操作界面
<b>ManageltemsForm</b>		图书管理员管理书目的操作界面
<b>LendltemForm</b>		图书管理员登记借书的操作界面
<b>ReturnItemForm</b>		图书管理员登记还书的操作界面
<b>MailSystem</b>		与邮件系统的接口

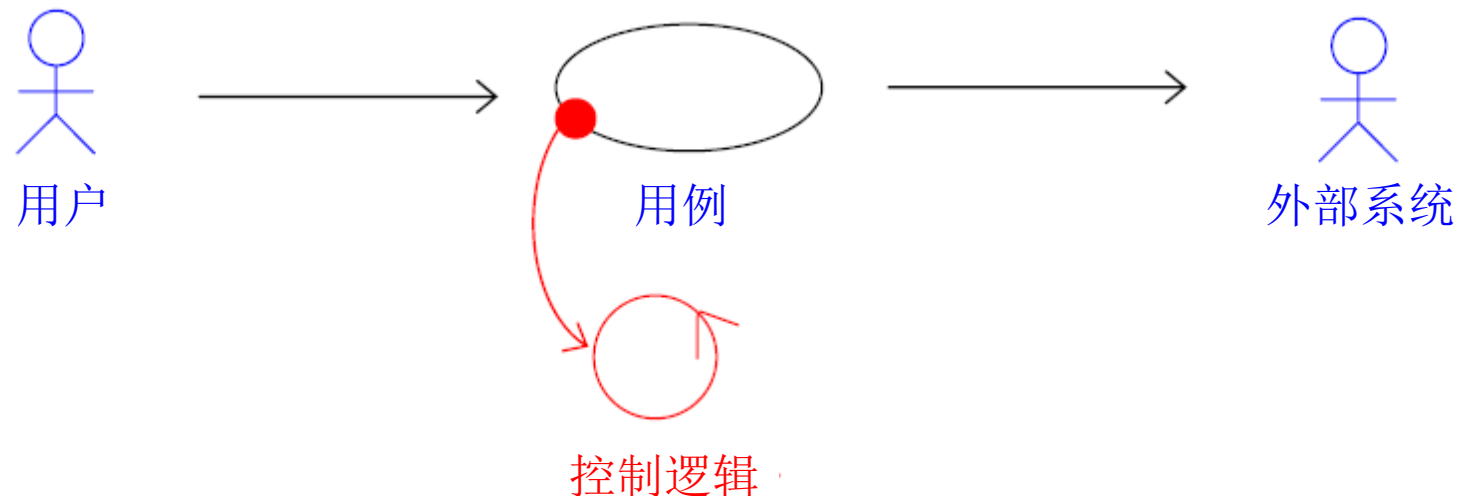
## 识别边界类应当注意的问题

- 边界类应关注于参与者与用例之间交互的信息或者响应的事件，不要描述窗口组件等界面的组成元素；
- 在分析阶段，力求使用用户的术语描述界面；
- 边界类实例的生命周期并不仅限于用例的事件流，如果两个用例同时与一个参与者交互，那么它们有可能会共用一个边界类，以便增加边界类的复用性。



# 识别控制类

- 控制类负责协调边界类和实体类，通常在现实世界中没有对应的事物；
- 负责接收边界类的信息，并将其分发给实体类；
- 控制类与用例存在着密切的关系，它在使用例开始执行时创建，在使用例结束时取消。一般来说，一个用例对应一个控制类。



## [案例]图书管理系统的控制类

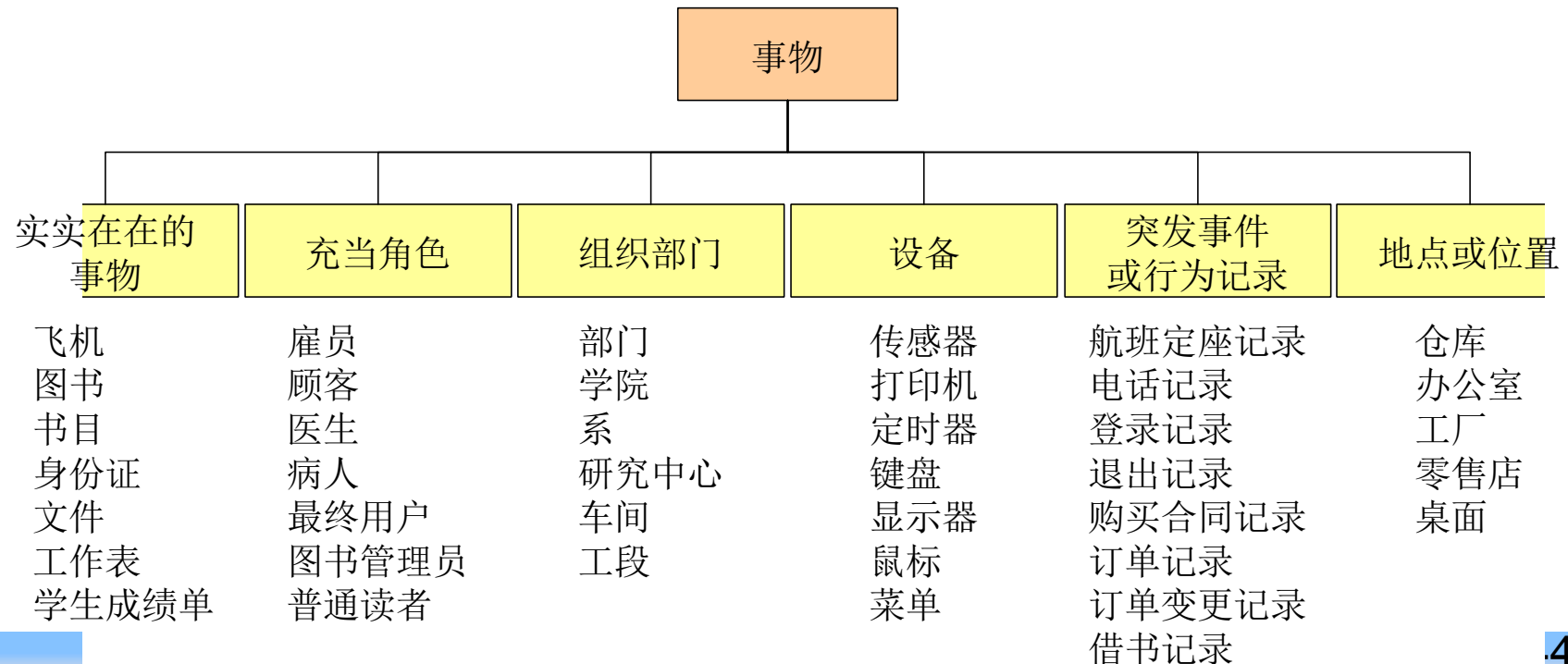
控制类		说明
<b>BrowseControl</b>		负责执行注册用户的查询浏览
<b>MakeReservationControl</b>		负责执行普通读者的预定图书
<b>RemoveReservationControl</b>		负责执行普通读者的取消预定
<b>ManageBrowsersControl</b>		负责执行图书管理员对读者的管理
<b>ManageTitlesControl</b>		负责执行图书管理员对图书资料的管理
<b>ManageItemsControl</b>		负责执行图书管理员对书目的管理
<b>LendItemControl</b>		负责执行图书管理员登记借书
<b>ReturnItemControl</b>		负责执行图书管理员登记还书

## 识别控制类应当注意的问题

- 当用例比较复杂时，特别是产生分支事件流的情况下，一个用例可以有多个控制类。
- 在有些情况下，用例事件流的逻辑结构十分简单，这时没有必要使用控制类，边界类可以实现用例的行为。
  - 例如：图书管理系统中的“登录”用例
- 如果不同用例包含的任务之间存在着比较密切的联系，则这些用例可以使用一个控制类，其目的是复用相似部分以便降低复杂性。
  - 通常情况下，应该按照一个用例对应一个控制类的方法识别出多个控制类，再分析这些控制类找出它们之间的共同之处。

# 识别实体类

- 实体类通常是用例中的参与对象，对应着现实世界中的“事物”
- 判断一个“名词”是否为实体类，其标准是：
  - 系统是否需要管理该名词所拥有的信息？
  - 系统是否需要管理该名词所能发出或接受的动作？



## [案例]图书管理系统的实体类

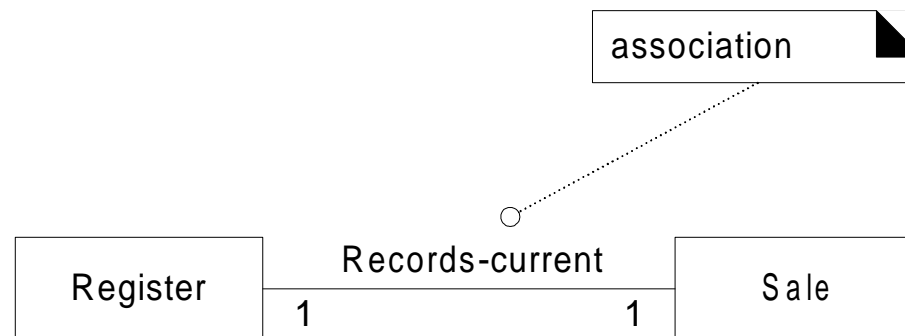
实体类		说明
<b>BrowserInfo</b>	○	普通读者的基本信息
<b>Loan</b>	○	普通读者的借书记录
<b>Reservation</b>	○	普通读者的预定信息
<b>Title</b>	○	图书资料的基本信息
<b>Item</b>	○	书目
(由于图书资料中包括书籍和杂志等类型，因此可以进一步划分子类)		
<b>BookItem</b>	○	书籍的基本信息
<b>MagazineItem</b>	○	杂志的基本信息

## 识别实体类应当注意的问题

- 实体类的识别质量在很大程度上取决于分析人员书写文档的风格和质量；
- 自然语言是不精确的，因此在分析自然语言描述时应该规范化描述文档中的一些措辞，尽量弥补这种不足；
- 在自然语言描述中，名词可以对应类、属性或同义词等多种类型，开发人员需要花费大量的时间进行筛选。

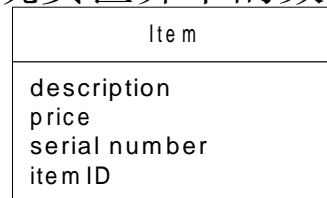
### 3.添加关联和属性

- 关联：关联是类之间的关系，表示有意义的连接



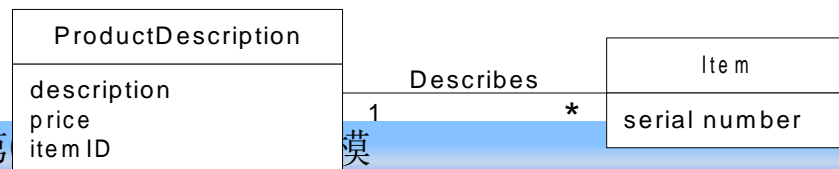
- 属性还是类？

- 如果我们认为某概念类X不是现实世界中的数字或文本，那么X可能是概念类而不是属性



**Worse**

- 何时使用描述类？



**Better**

# 描述分析类的属性

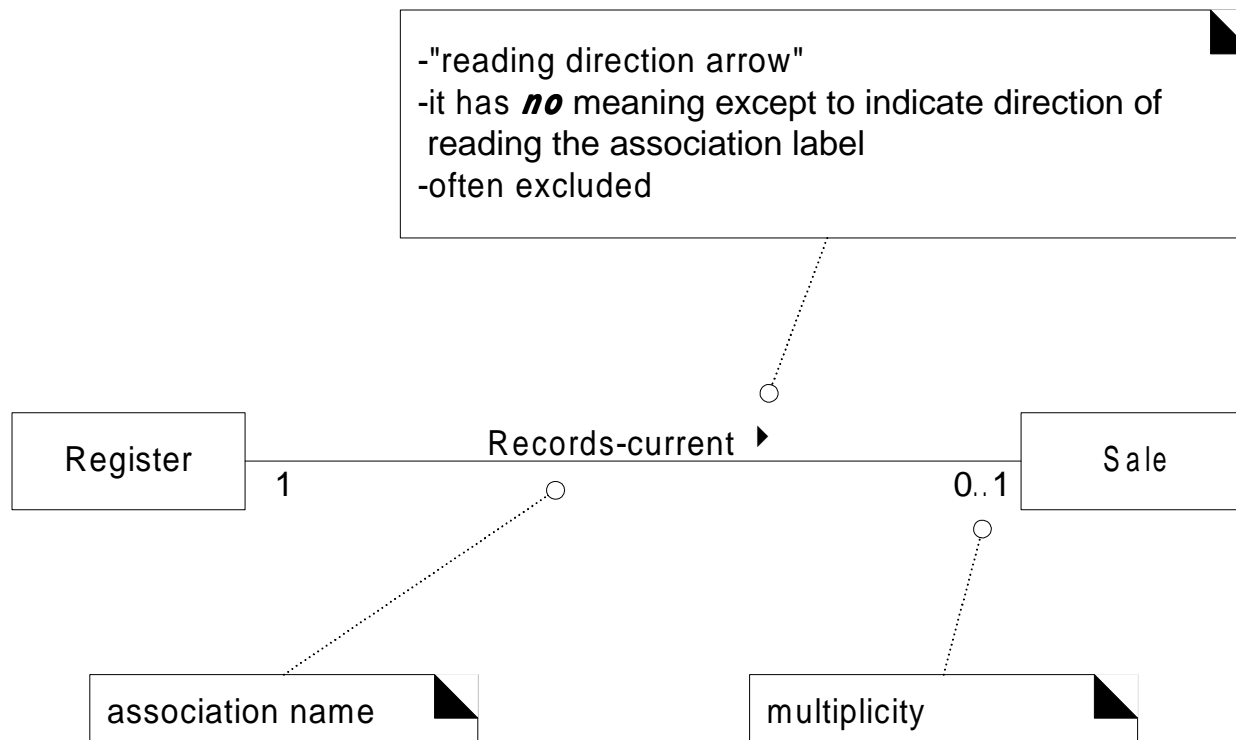
- 按照一般常识，找出对象的某些属性；
  - 人员的姓名、性别、年龄、地址等；
- 认真研究问题域，找出对象的某些属性；
  - 商品的条形码、学生的学号；
- 根据系统责任的要求，找出对象的某些属性；
- 考虑对象需要系统保存的信息，找出对象的相应属性；
- 对象为了在服务中实现其功能，需要增设一些属性；
- 识别对象需要区别的状态，考虑是否需要增加一个属性来区别这些状态；
- 确定属性表示整体与部分结构和实例连接。



# 添加关联

## ■ 关联

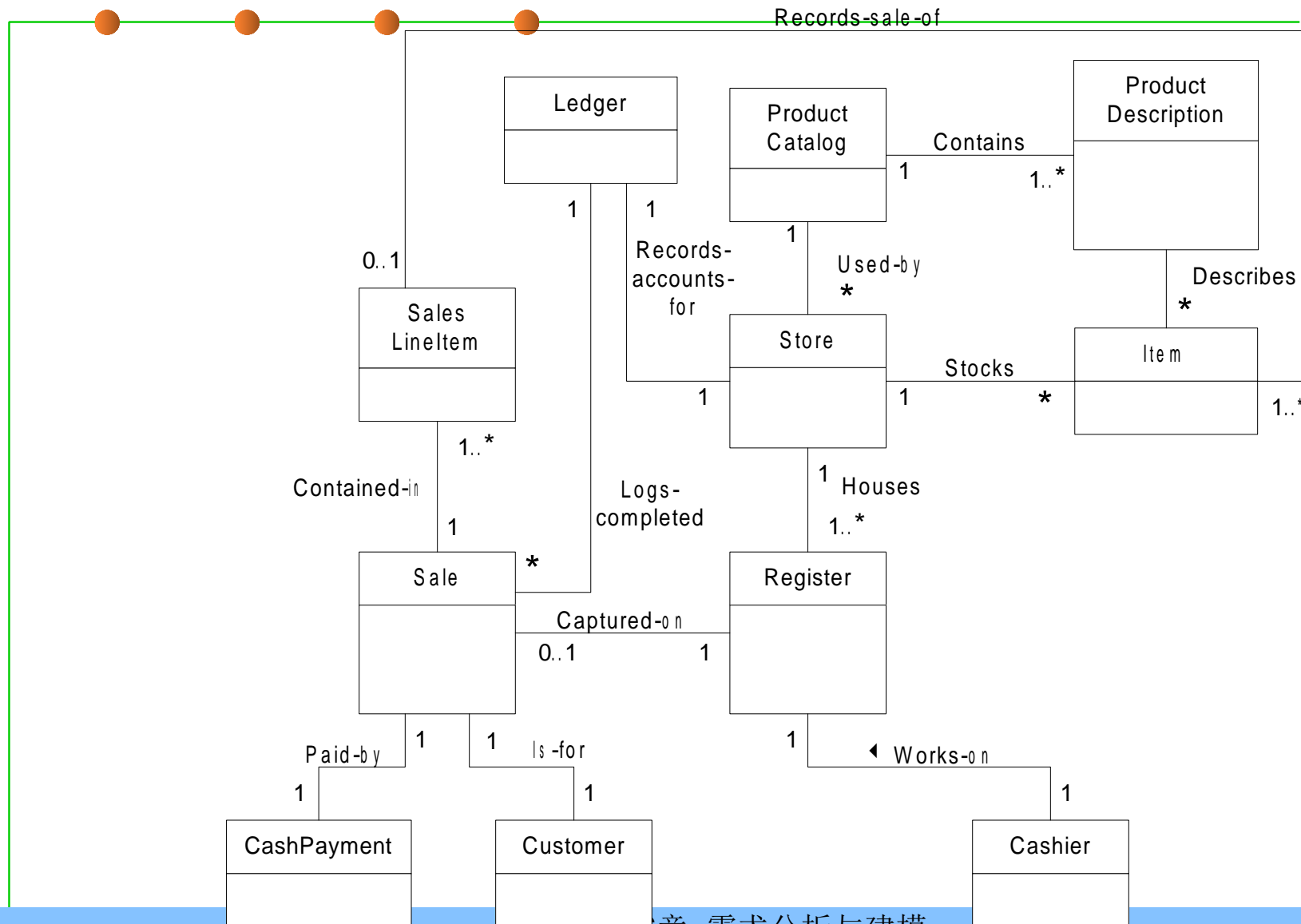
- 避免加入太多的关联，重点关注“需要记住”的关联



# 建立类之间的关系

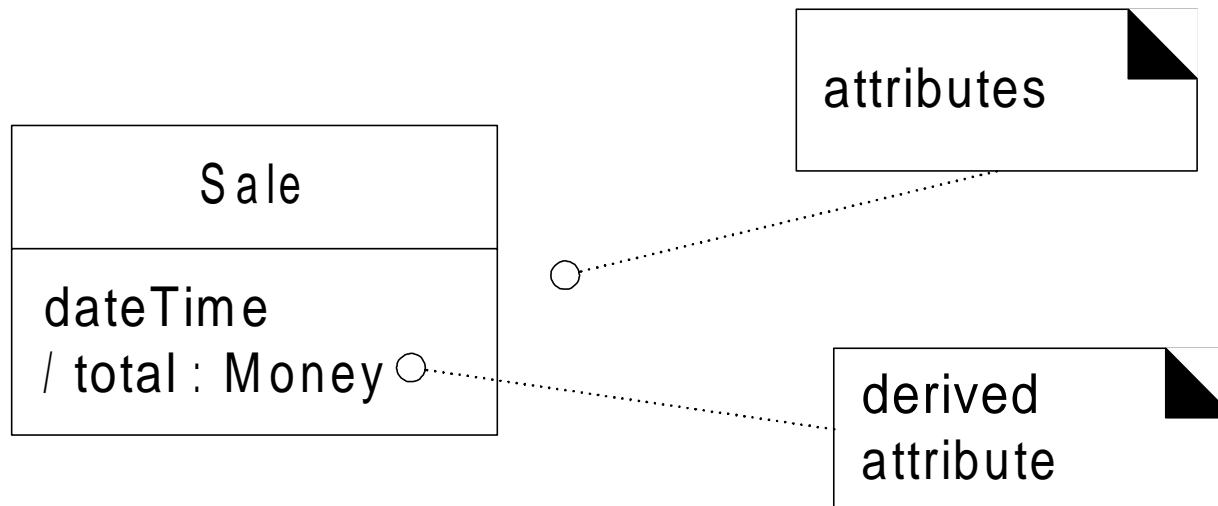
- 关系：
  - 泛化(generalization)
  - 关联(association)
  - 组合(composition)
  - 聚合(aggregation)
- 例如：
  - “读者”类与“预定”类之间有1:n的关联关系；
  - “读者”类与“借书记录”类之间有1:n的关联关系；
  - “图书资料”类与“借书记录”类之间有1:1的关联关系；
  - “书目”类与“书籍”类、“杂志”类之间是泛化关系。

# 概念类图示例1

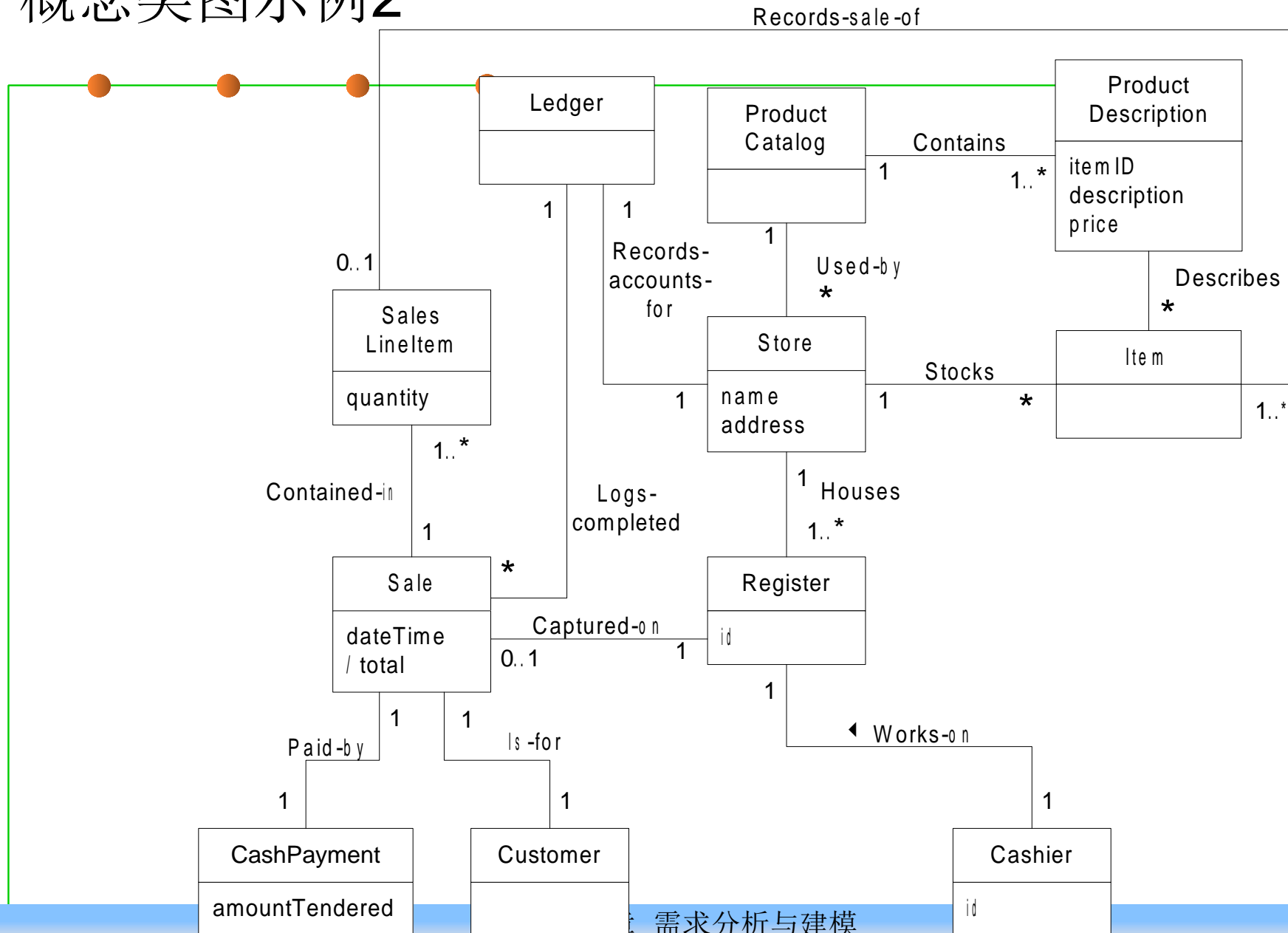


# 属性

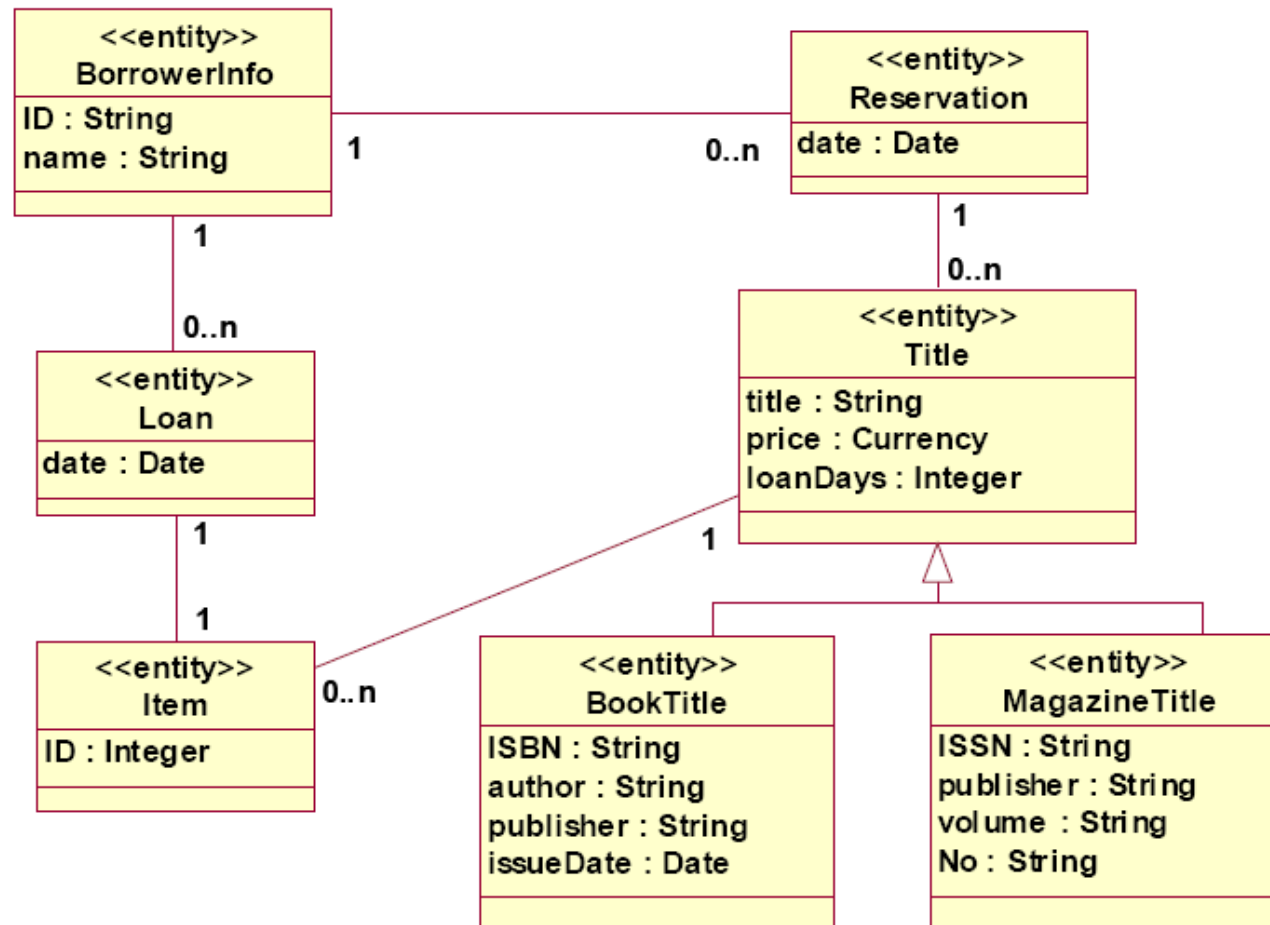
- 当需求（例如，用例）建议或暗示需要记住的信息时，引入属性



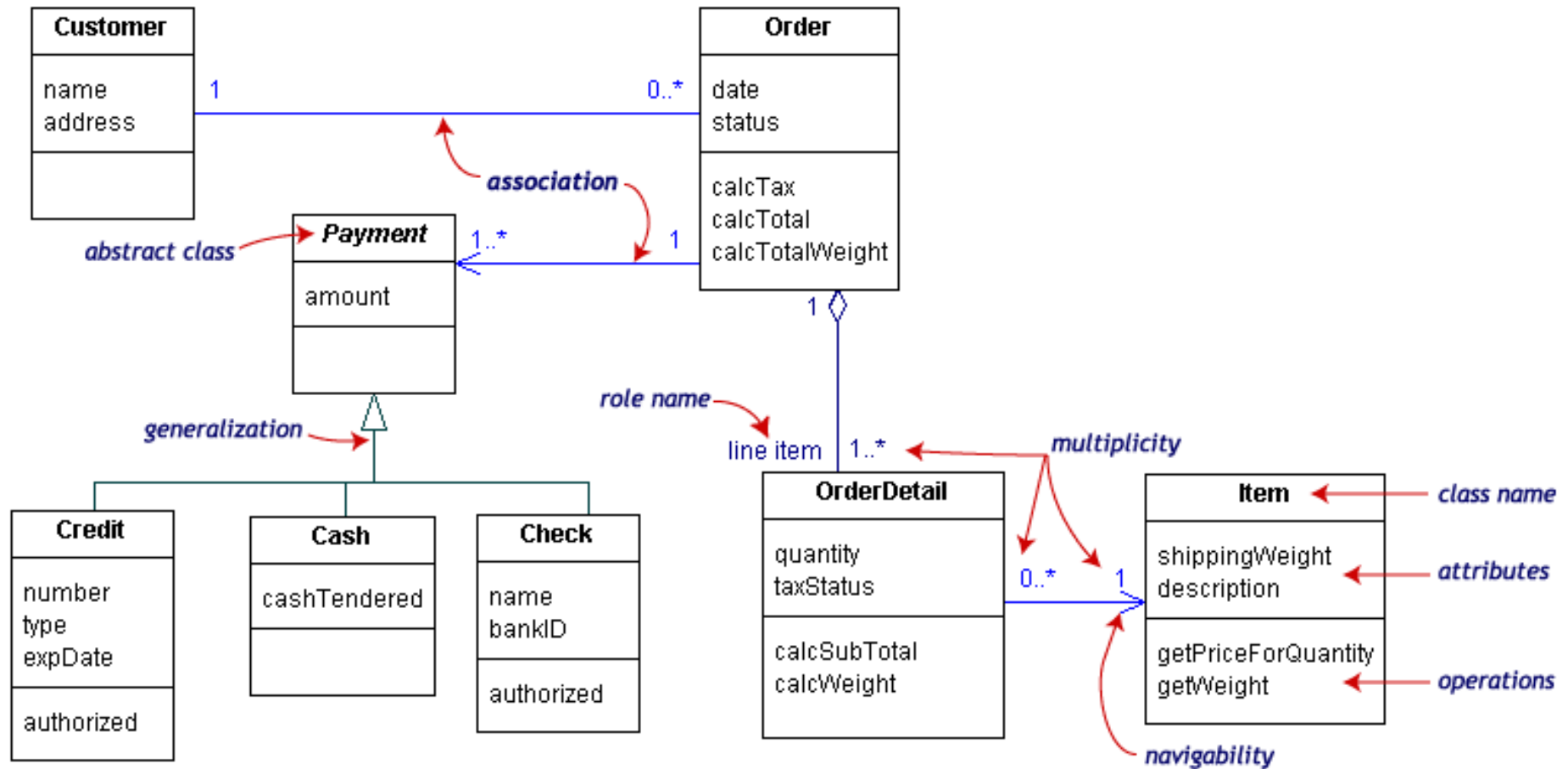
## 概念类图示例2



# 绘制类图



## 类图的另一个例子



# 主要内容

- **6.1 需求分析**
- **6.2 面向对象需求分析**
  - 6.2.1 面向对象需求分析
  - 6.2.2 静态分析建模
  - 6.2.3 动态分析建模
- **6.3 结构化需求分析**



## 面向对象分析的过程

### ■ 第二阶段：建立动态模型

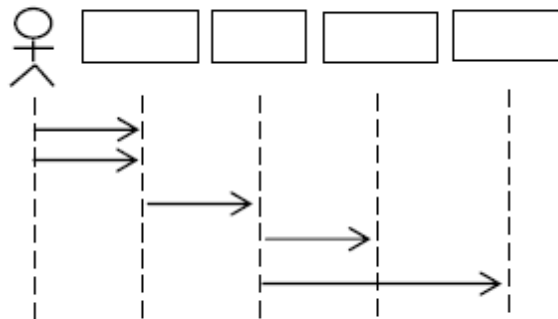
1. 从用例模型和活动图入手，建立系统顺序图
2. 建立分析对象的顺序图
3. 确定分析类的操作

# 建立动态行为模型

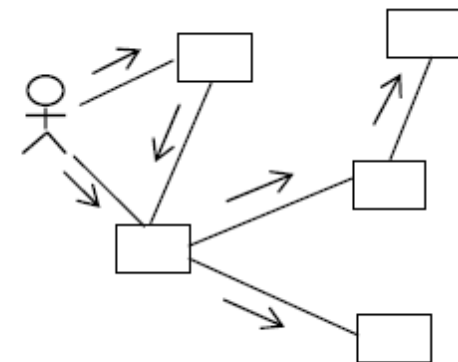
- 动态行为模型可用两个新视图描述：
  - 顺序图(Sequence Diagram)
  - 协作图(Collaboration Diagram)



用例



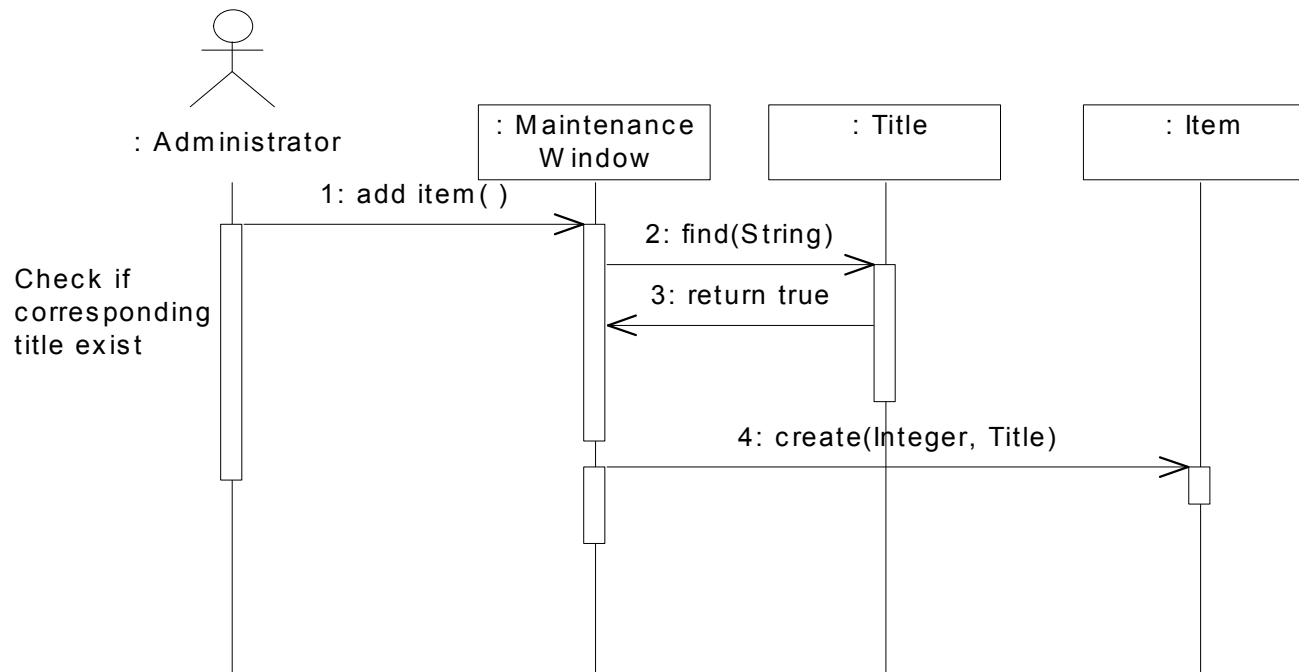
顺序图



协作图

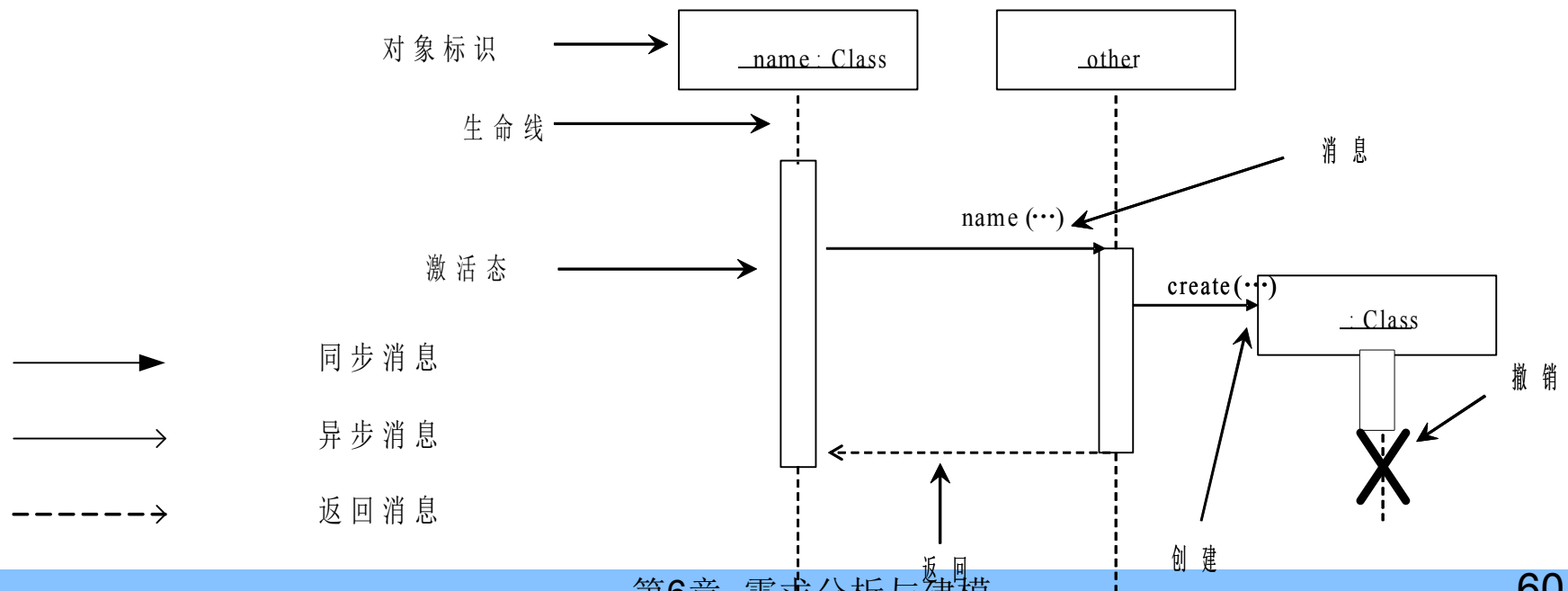
# 顺序图

- 顺序图是强调消息时间顺序的交互图
- 顺序图描述了对对象之间传送消息的时间顺序，表示用例中的行为顺序
- 顺序图将交互关系表示为一个二维图。其中，纵轴是时间轴，时间沿竖线向下延伸。横轴代表了在协作中各独立的对象



# 顺序图的组成

- 对象 (**Object**)
- 生命线 (**Lifeline**) : 生命线是一个时间线
- 消息 (**Message**)
- 激活 (**Activation**)



# 消息

- 消息定义的是对象之间某种形式的通信，它可以激发某个操作、唤起信号或导致目标对象的创建或撤销。
- 消息是两个对象之间的单路通信，从发送方到接收方的控制信息流。
- 消息可以用于在对象间传递参数。
- 消息可以是信号，也可以是调用。
- 在**UML**中，消息使用箭头来表示，箭头的类型表示了消息的类型。

# 激活

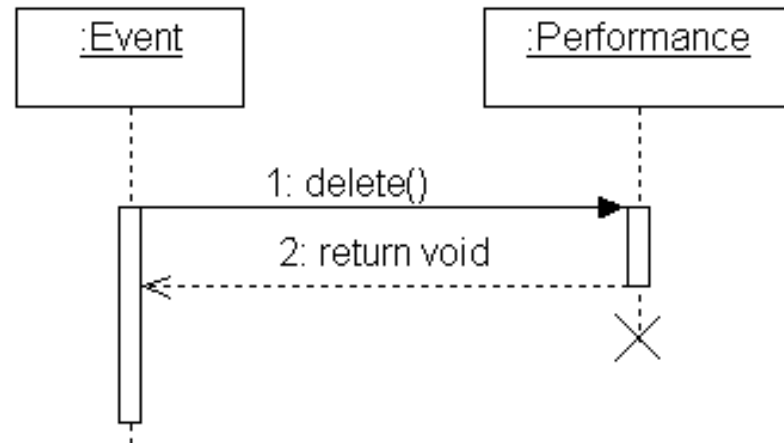
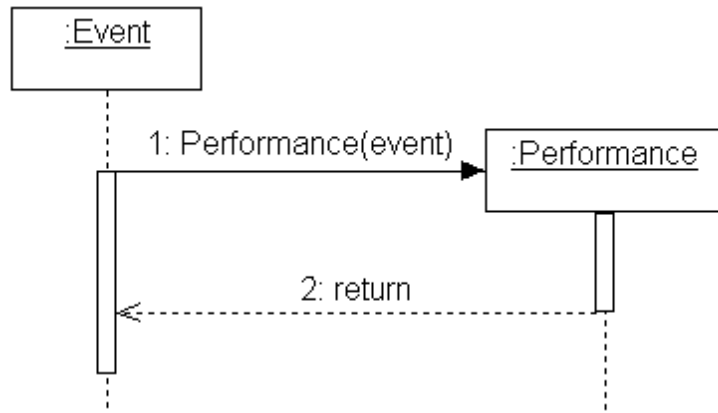
- 激活表示该对象被占用以完成某个任务，去激活指的则是对象处于空闲状态、在等待消息。
- 在**UML**中，为了表示对象是激活的，可以将该对象的生命线拓宽成为矩形。其中的矩形称为激活条或控制期，对象就是在激活条的顶部被激活的，对象在完成自己的工作后被去激活。

## ■ 激活条



# 对象的创建和撤销

- 如果对象位于顺序图的顶部，说明在交互开始之前该对象已经存在了。如果对象是在交互的过程中创建的，那么它应当位于图的中间部分
- 对象在创建消息发生之后才能存在，对象的生命线也是在创建消息之后才存在的。
- 如果要撤销一个对象，只要在其生命线终止点放置一个“X”符号即可，该点通常是对删除或取消消息的回应。

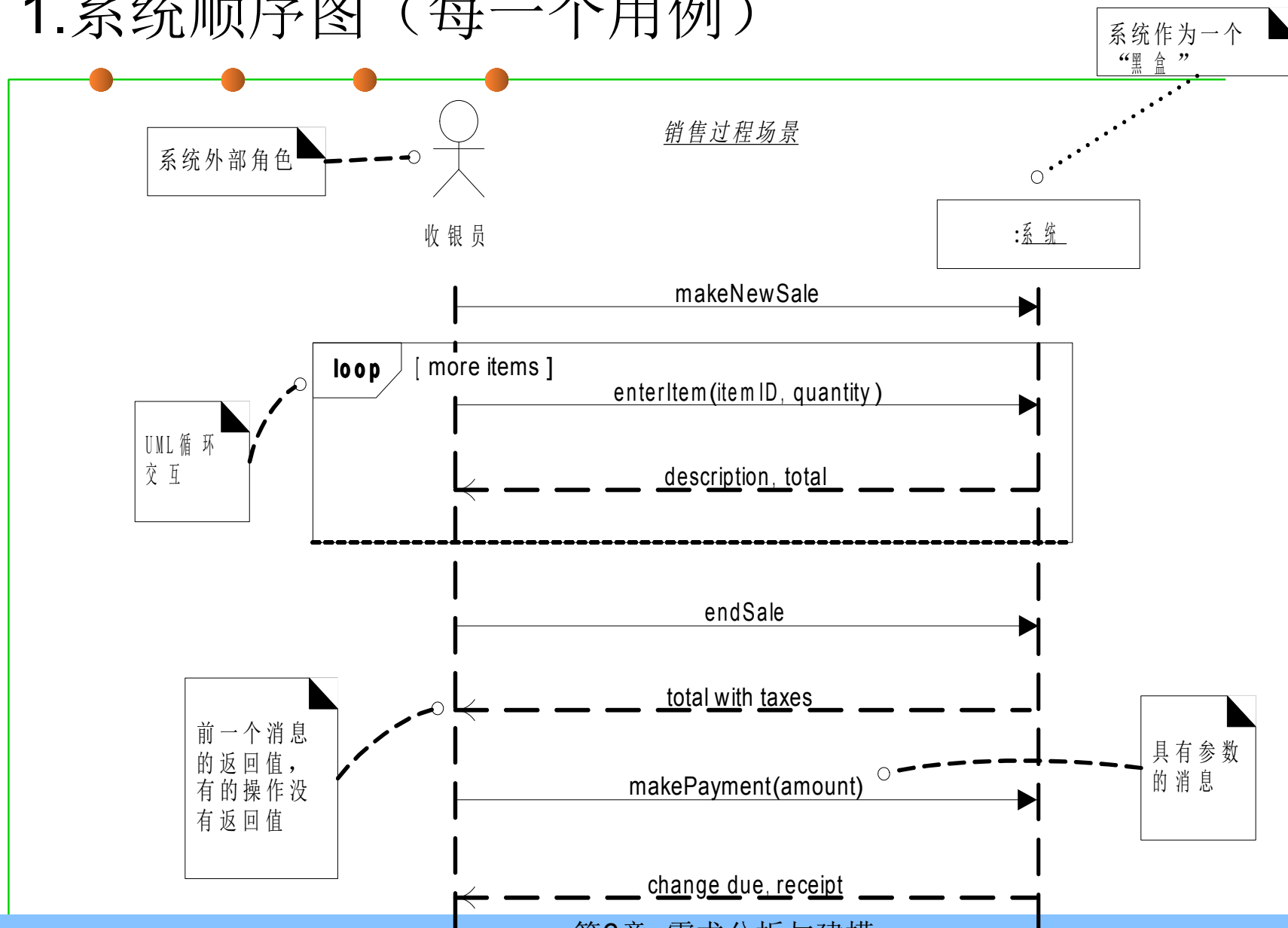


## 顺序图建模过程

1. 设置交互的语境
2. 通过识别对象在交互中扮演的角色，设置交互的场景
3. 为每个对象设置生命线
4. 从引发某个消息的信息开始，在生命线之间画出从顶到底依次展开的消息，显示每个消息的特性（如参数）
5. 如果需要可视化消息的嵌套或实际计算发生时的时间点，可以用激活修饰每个对象的生命期
6. 如果需要说明时间或空间的约束，可以用时间标记修饰每个消息，并附上合适的时间和空间约束
7. 如果需要更形式化的说明某控制流，可以为每个消息附上前置和后置条件



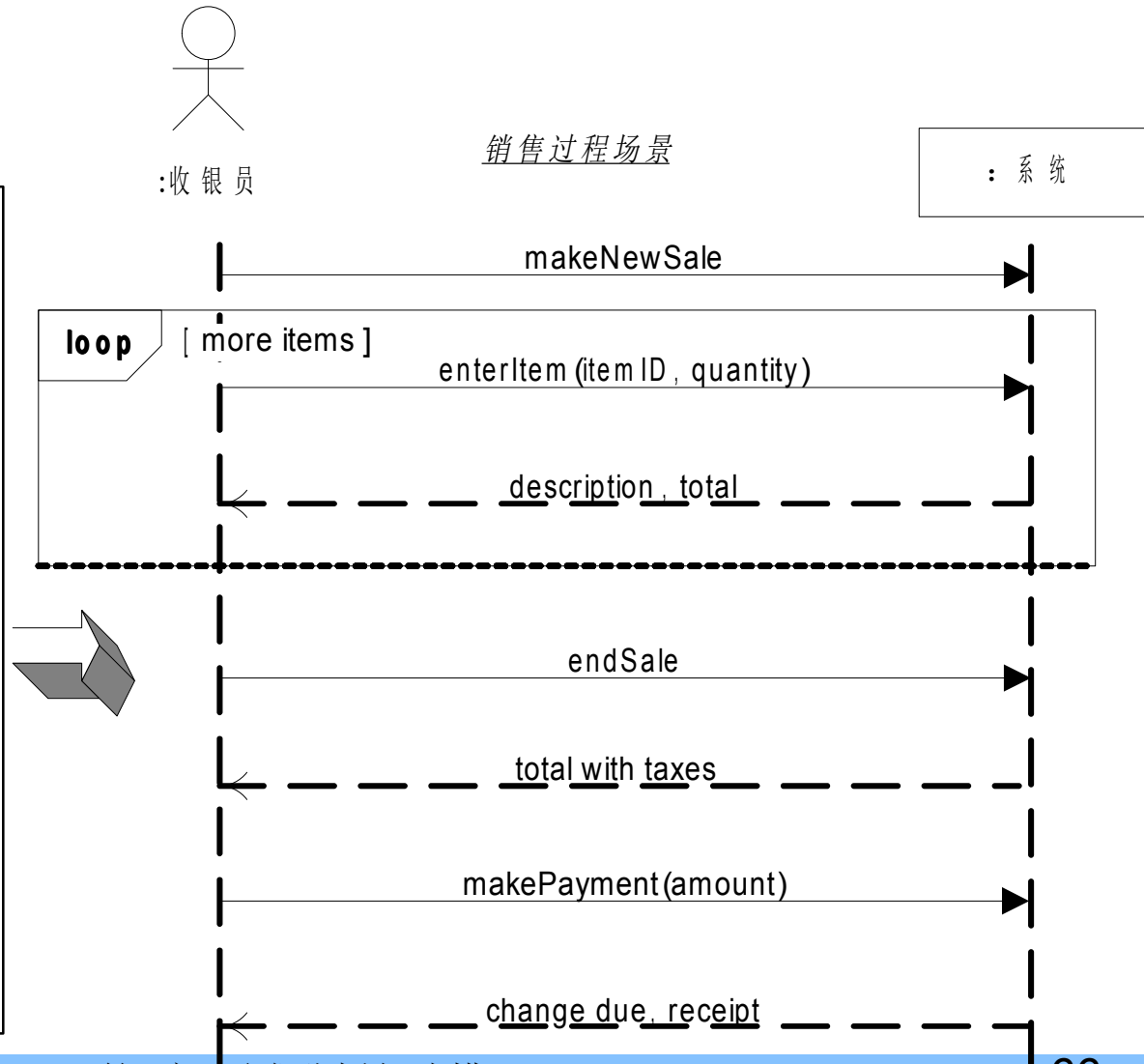
# 1.系统顺序图（每一个用例）



# 1.系统顺序图

简单现金销售过程场景：

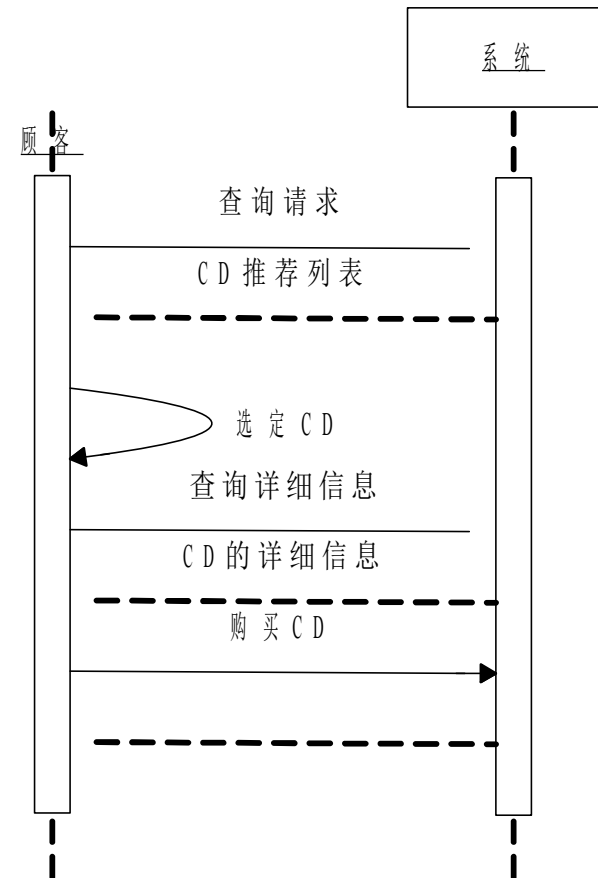
1. 顾客带商品到收银台
2. 收银员开始一个新销售
3. 收银员输入商品标识
4. 系统记录销售商品并显示描述和价格，计算总额！  
收银员重复步骤 3-4 直到录入完毕。
5. 系统显示总价并计算税金。
6. 收银员告诉顾客总价并要求支付。
7. 顾客付款，系统处理支付。
- ...



# 1.系统顺序图

用例描述：

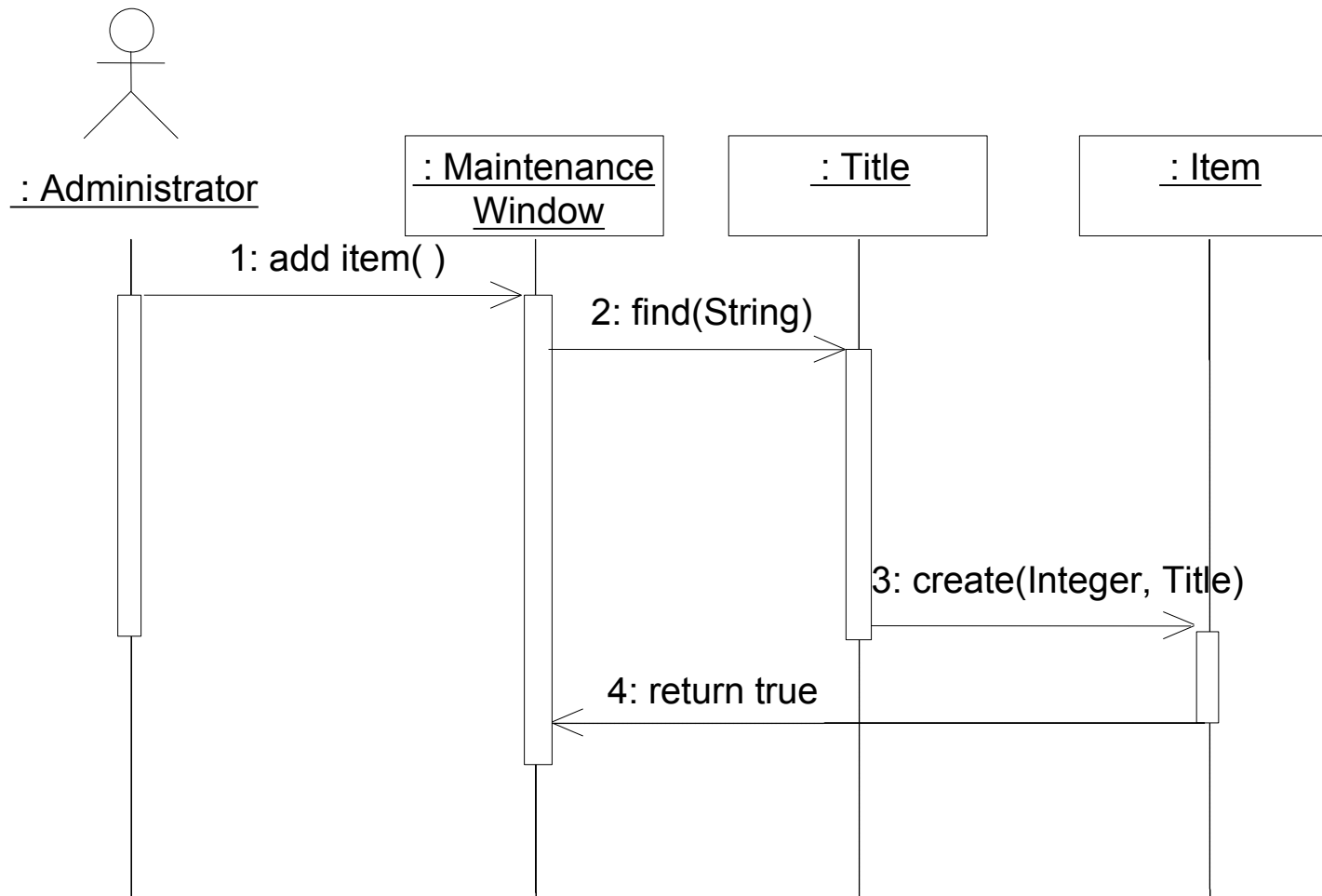
1. 顾客向系统提起查询请求
2. 系统根据请求为顾客提供一个CD 的推荐列表
3. 顾客在推荐列表选定一个CD ，然后要求查看更详细的信息
4. 系统为顾客提供选定CD 的详细信息
5. 顾客购买选定CD .
6. 顾客离开 .



## 2.分析对象顺序图

- 将用户与分析类结合在一起，实现将用例的行为分配到所识别的分析类中；
- 注意与系统顺序图的一致性
- 步骤：
  - 列出启动该用例的参与者；
  - 列出启动用例时参与者使用的边界对象；
  - 列出管理该用例的控制对象；
  - 根据用例描述的流程，按时间顺序列出分析类之间进行消息访问的序列。

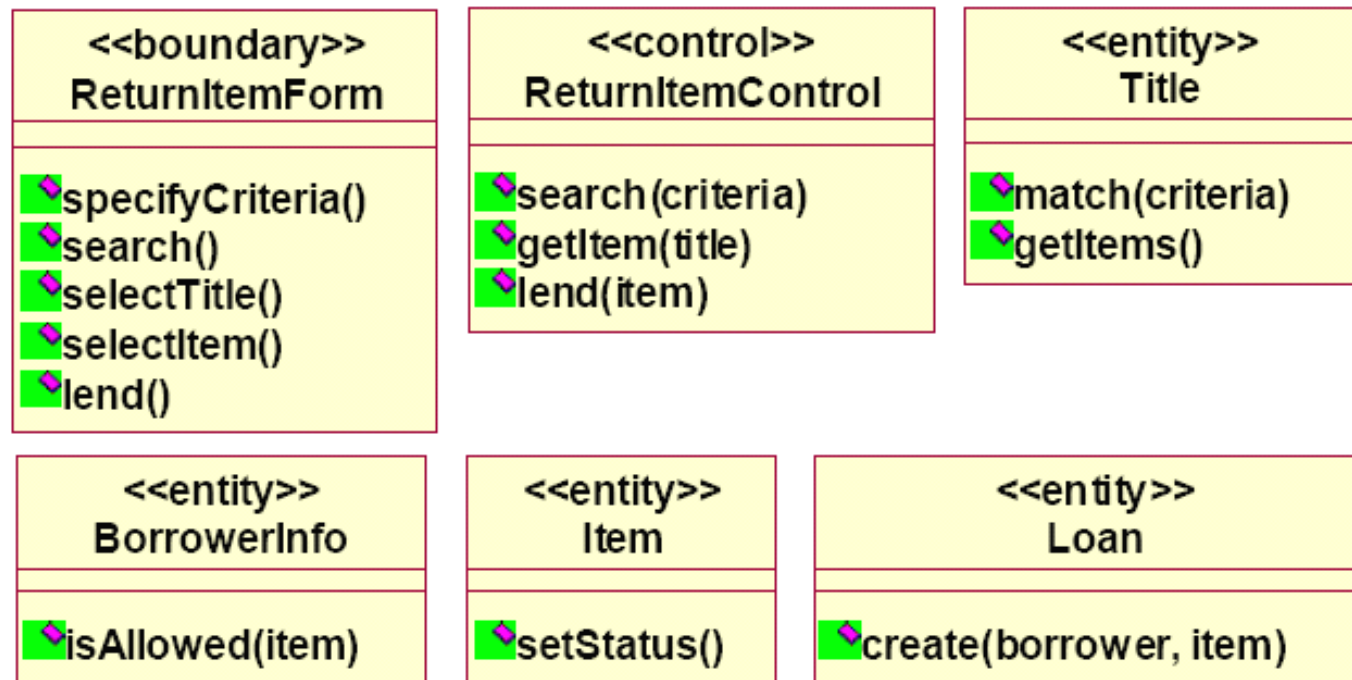
## [例]图书管理的对象顺序图



### 3.确定分析类操作

- 通过顺序图将用例行为分配到相应的分析类之后，确定对象职责
- 每一个消息对应一个操作，并确定消息参数
- 属性操作
  - 查询属性: `getAttribute()`
  - 修改属性: `setAttribute()`

## [例]确定分析类的操作

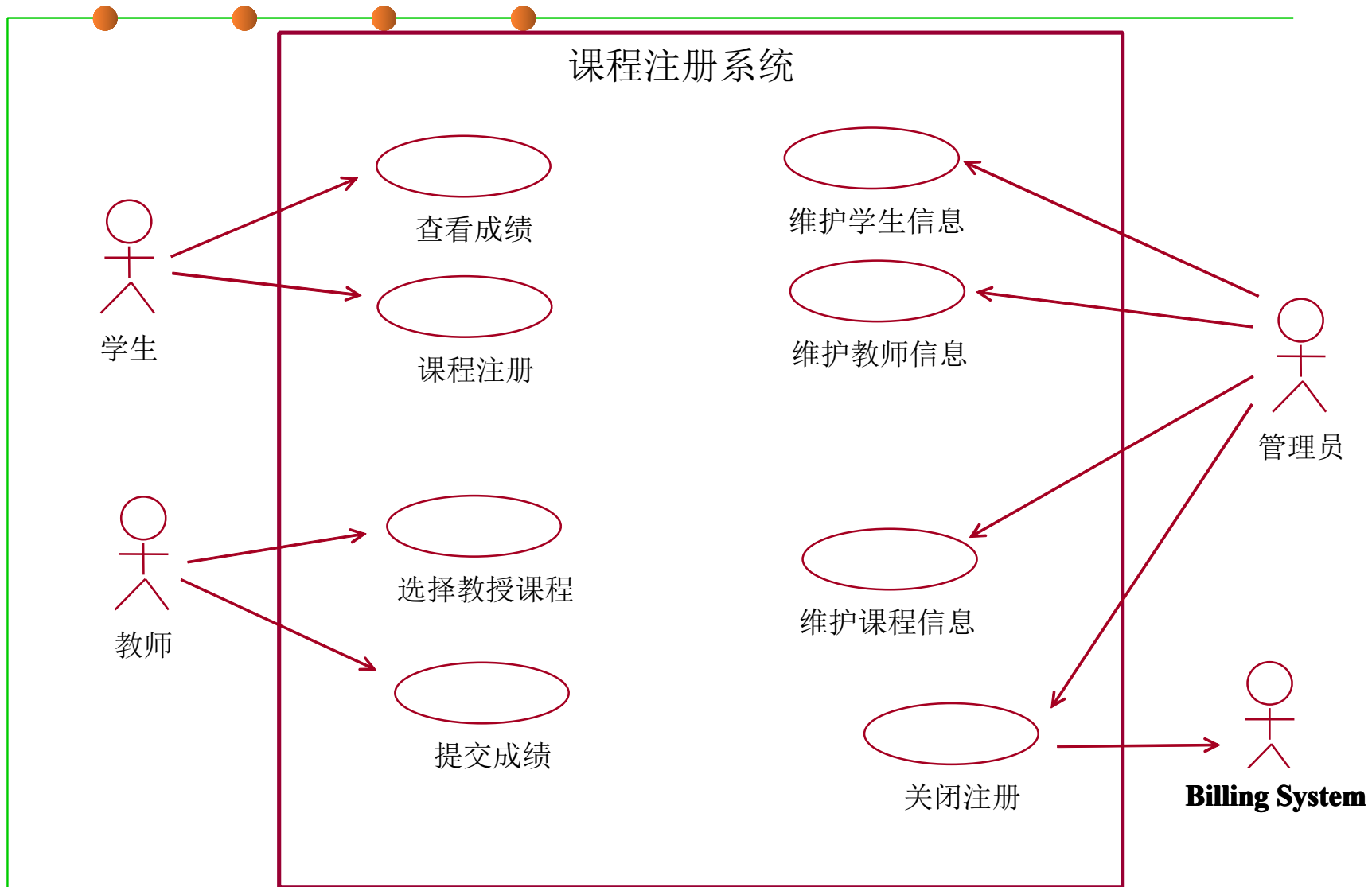




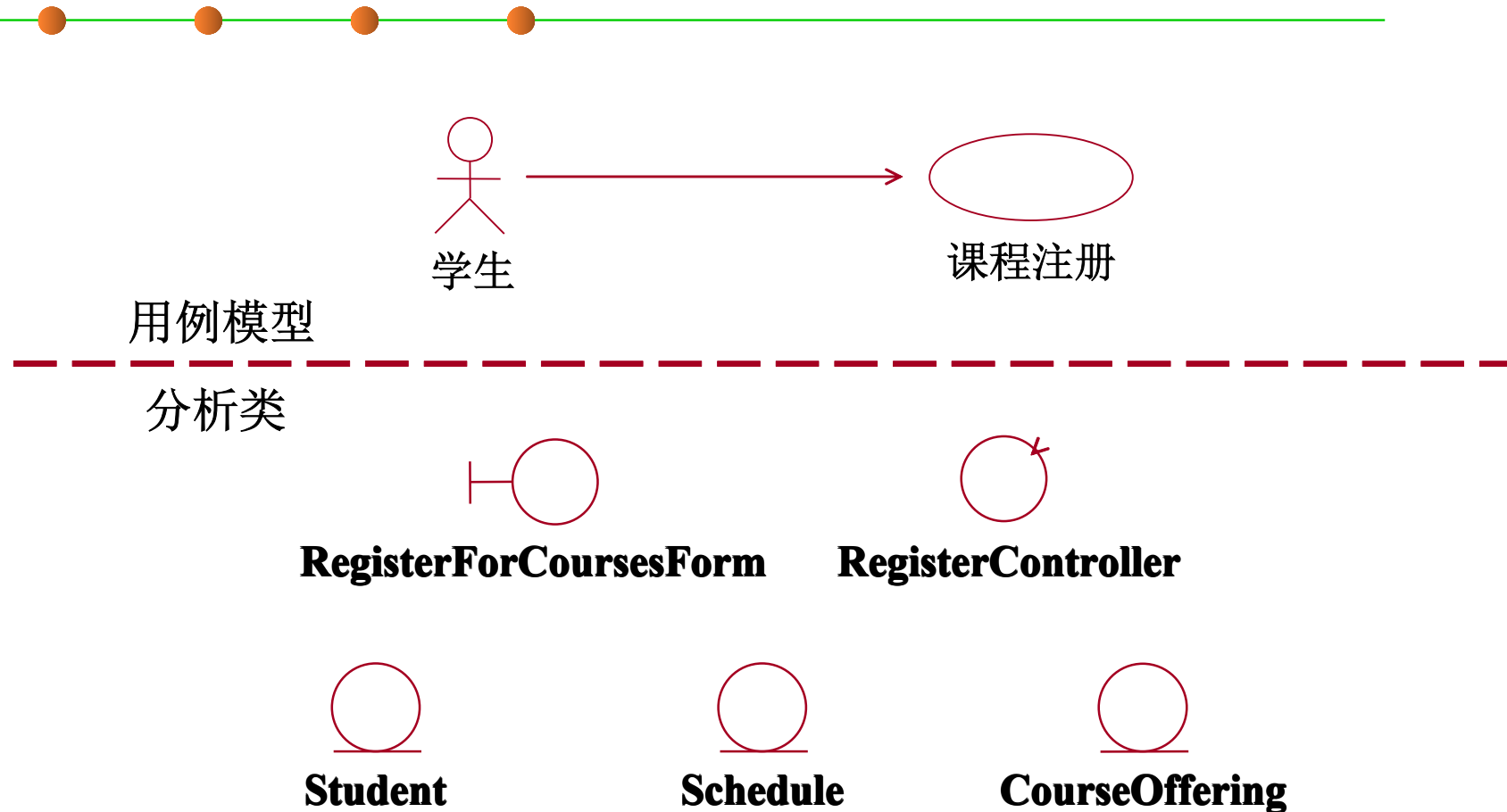
## 案例分析



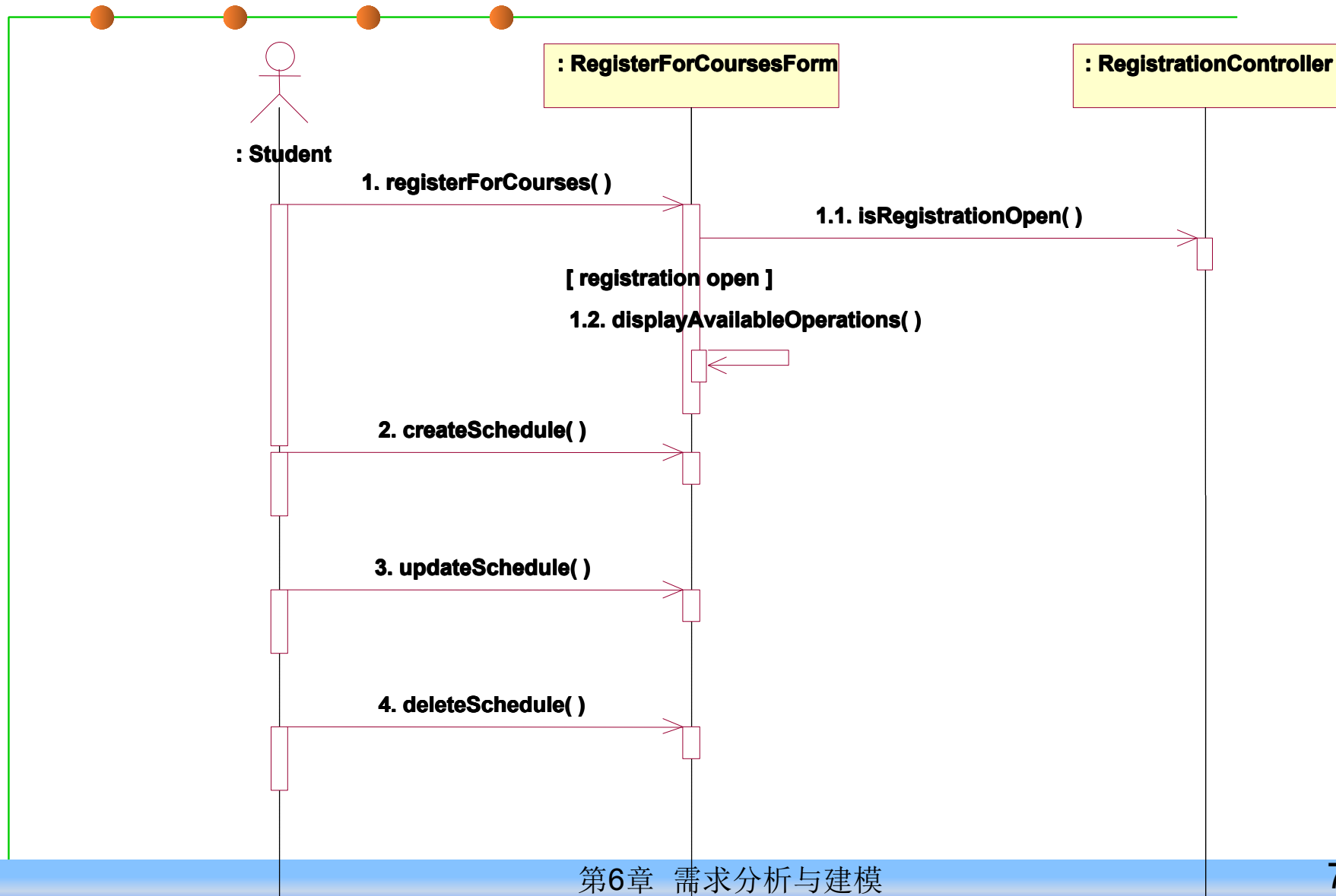
# 用例图：课程注册管理系统



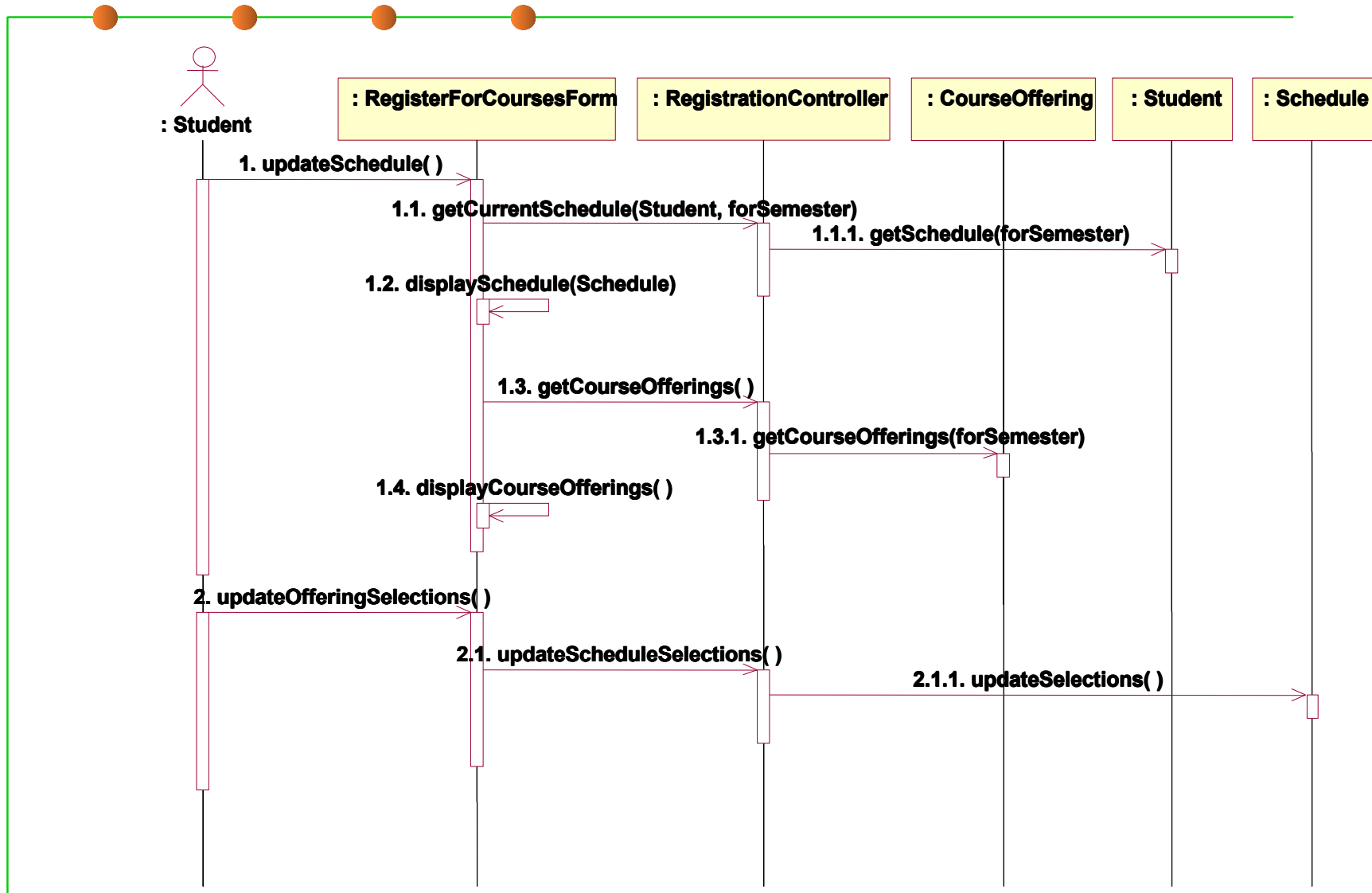
## 针对用例“课程注册”得到的分析类



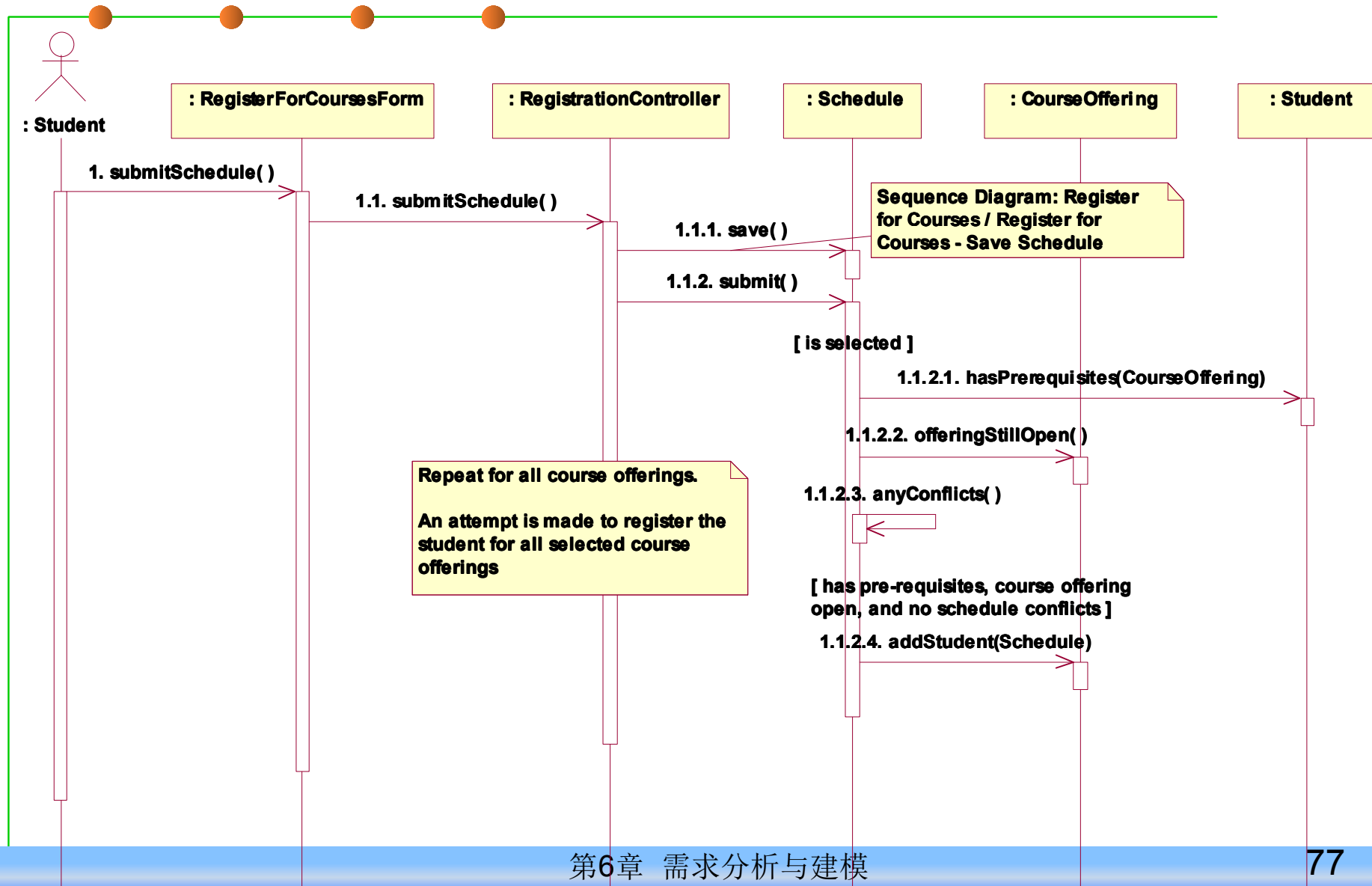
# 用例“学生课程注册”的顺序图



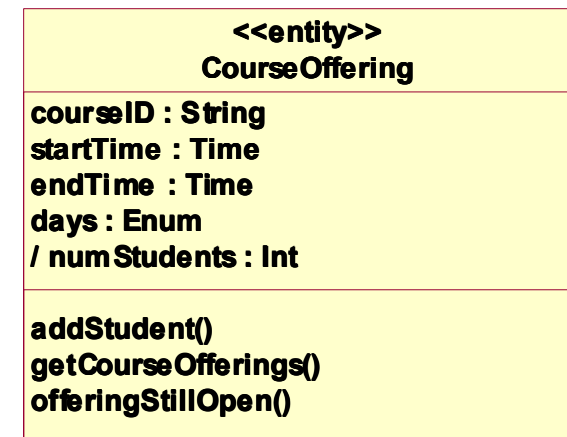
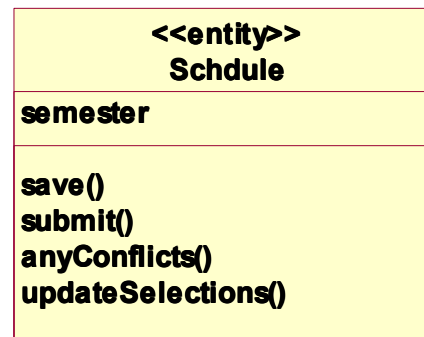
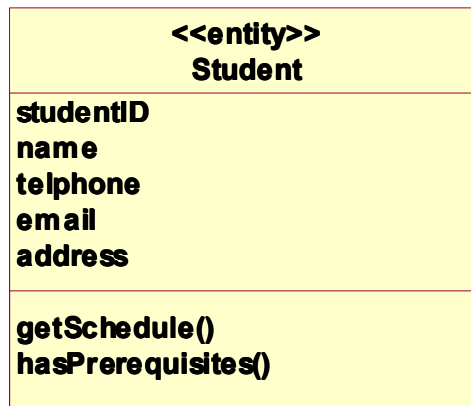
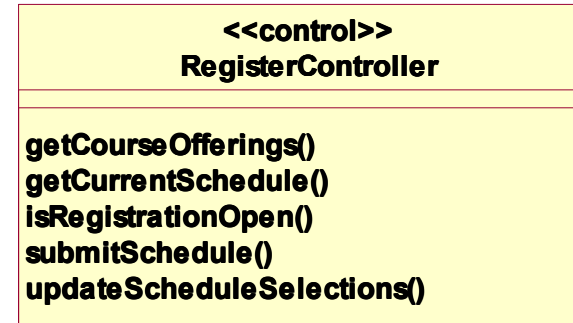
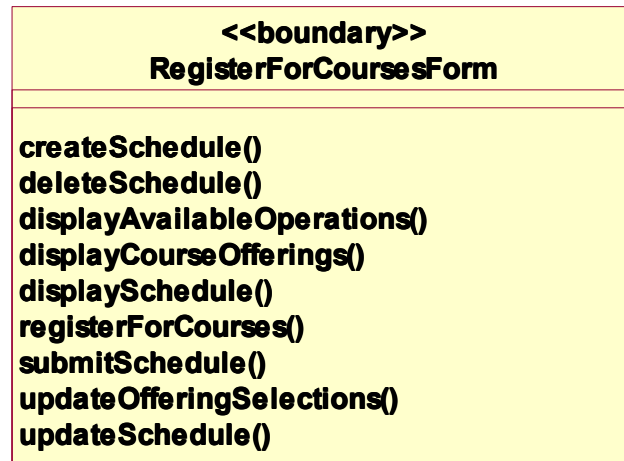
# 用例“学生修改已注册课程”的顺序图



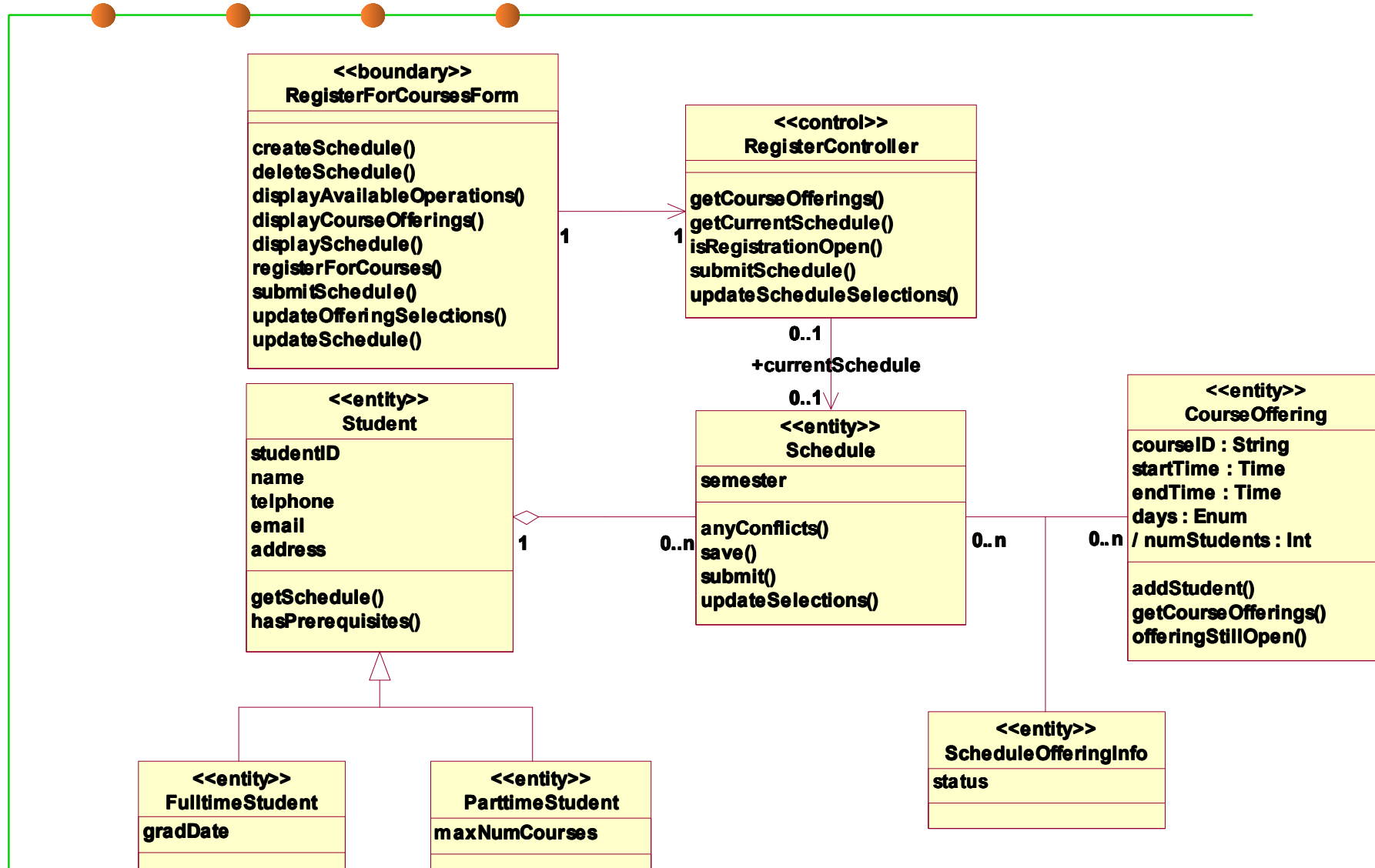
# 用例“学生提交已注册课程”的顺序图



# 类的属性与操作



# 类之间的关系：类图



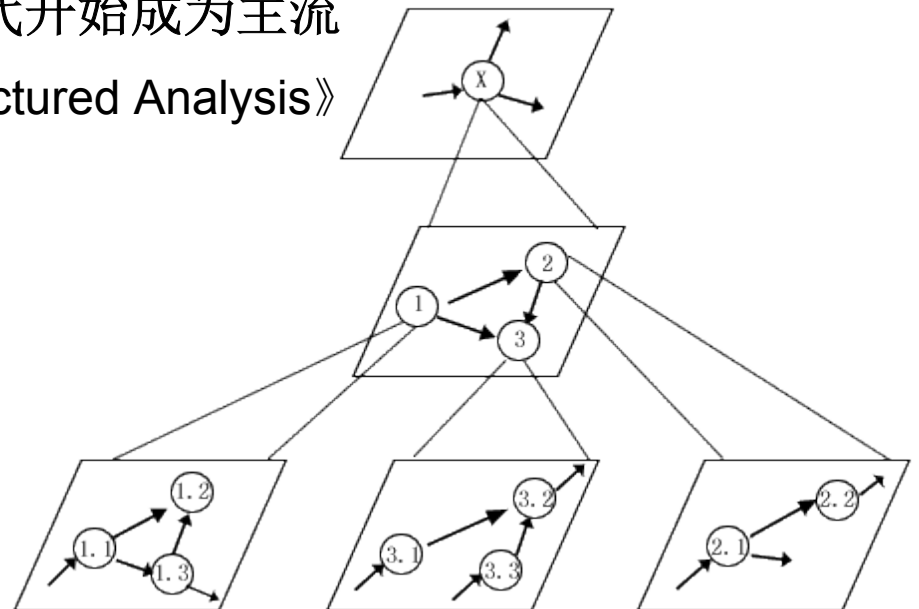
# 主要内容

- **6.1 需求分析**
- **6.2 面向对象需求分析**
- **6.3 结构化需求分析**
  - 6.3.1 结构化分析
  - 6.3.2 数据流图
  - 6.3.3 数据字典



# 结构化需求分析方法的起源

- 结构化分析方法(SA): 将待解决的问题看作一个系统, 从而用系统科学的思想方法(抽象、分解、模块化)来分析和解决问题。
- 起源于结构化程序设计语言(事先设计好每一个具体的功能模块, 然后将这些设计好的模块组装成一个软件系统);
- 最早产生于**1970**年代中期, **1980**年代开始成为主流
  - Yourdon于1989年出版《Modern Structured Analysis》
- 核心思想:
  - 自顶向下的分解(top-down)



# 结构化方法的模型

- 结构化需求分析方法通常需建立以下模型：
  - 数据流图(Data Flow Diagram, DFD)
    - 描述系统由哪些部分组成、各部分之间有什么联系等
  - 数据字典(Data Dictionary, DD)
    - 定义了数据流图中每一个数据元素
  - 结构化语言(Structured Language)
  - 判定表或判定树(Decision Table/Tree)
    - 详细描述数据流图中不能被再分解的每一个加工的内部处理逻辑
  - 实体联系图(Entity-Relationship Diagram, E-R)
  - 状态转换图(State Transition Diagram, STD)

# 主要内容

- **6.1 需求分析**
- **6.2 面向对象需求分析**
- **6.3 结构化需求分析**
  - 6.3.1 结构化分析
  - 6.3.2 数据流图
  - 6.3.3 数据字典

# 1.数据流图(DFD)

**DFD**的作用

**DFD**的模型元素及图形化表示

**DFD**的层次性

绘制**DFD**应遵循的约束

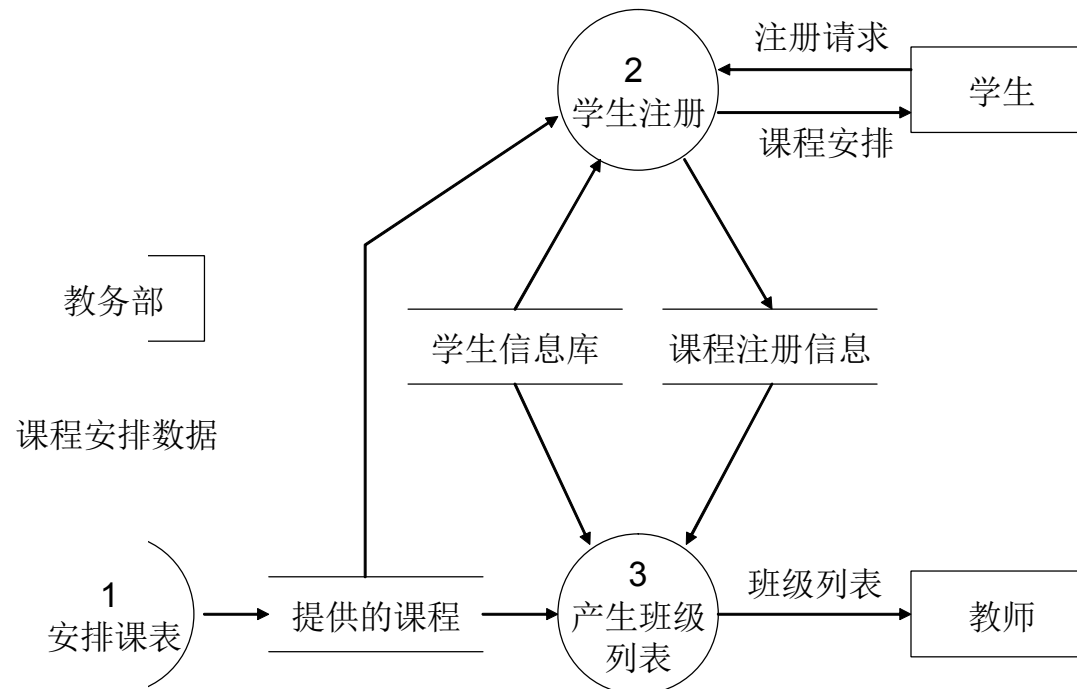
**DFD**应用案例

**DFD**树

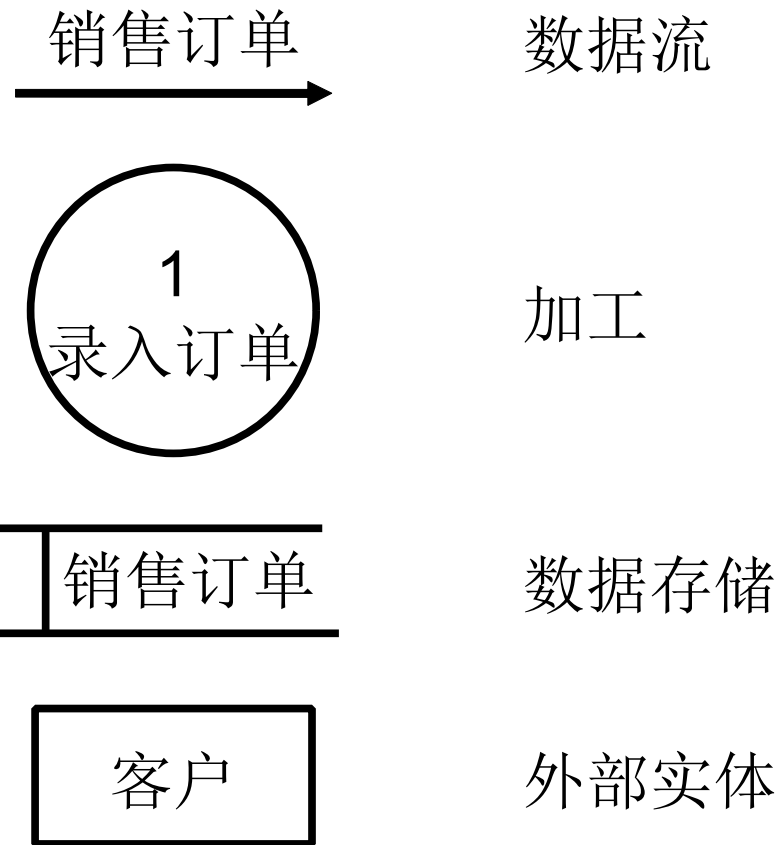
# 1 数据流图(Data Flow Diagram, DFD)

## ■ 数据流图(DFD): 结构化系统分析的基本工具

- 描绘数据在系统中各逻辑功能模块之间的流动和处理过程，是一种功能模型
- 主要刻画“功能的输入和输出数据”、“数据的源头和目的地”

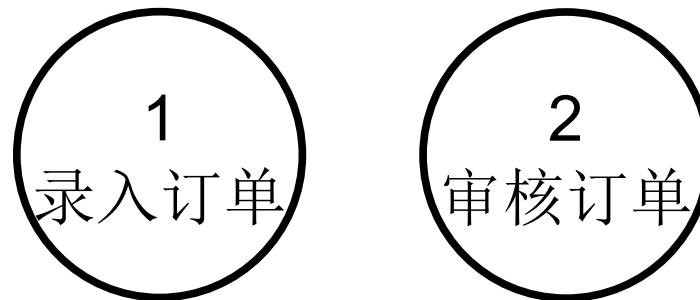


# DFD的主要元素



## DFD的主要元素(1): 加工

- 加工(又称数据处理, **data processing**): 对数据流进行某些操作或变换。
  - 收集、排序、选择、聚集、分析等
  - 加工要有名字, 通常是动词短语, 简明地描述完成什么事情
  - 在分层的数据流图中, 加工还应编号
  - 三种类型: 计算机自动加工、手工加工、人机协作的加工



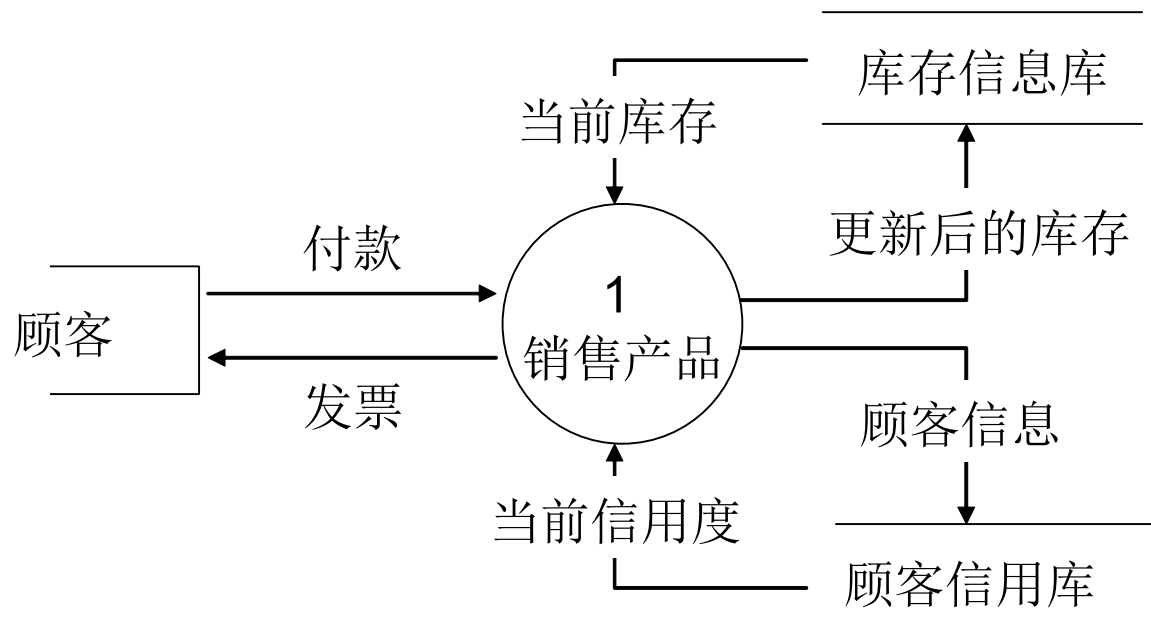
## DFD的主要元素(2):

- 数据存储(**data storage**)  
存的数据，它可以是数  
— 以名词命名

销售订单

销售订单

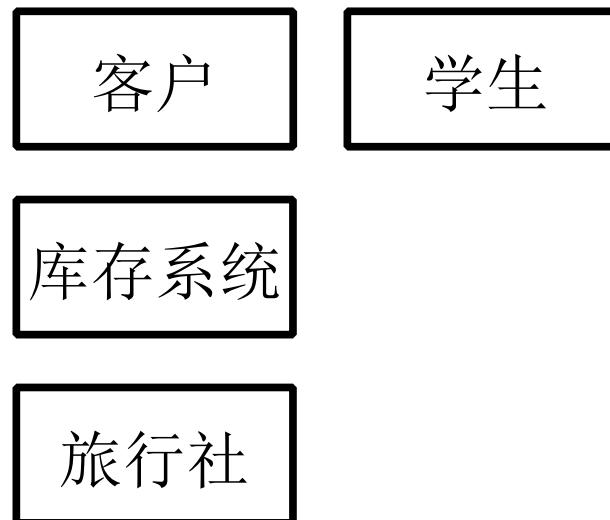
销售订单





## DFD的主要元素(3): 外部实体

- **外部实体(external entity):** 本系统外部环境中的实体(包括人员、组织或其他软件系统)
  - 也称为“数据源点/数据终点”，表示产生数据的源头或消费数据的终点
  - 以名词短语命名
  - 不能直接访问数据存储



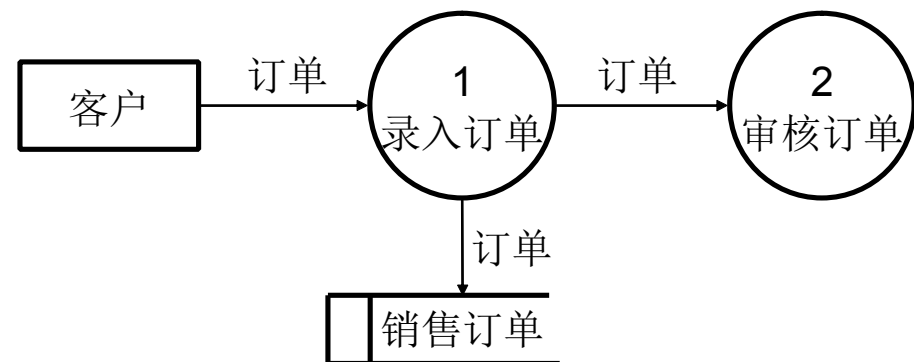
## DFD的主要元素(4): 数据流

### ■ 数据流(data flow): 数据在系统内传播的路径

- 由一组成成分固定的数据组成
- 由于数据流是流动中的数据, 因此它必须具有方向性
- 应用名词或名词短语命名
- 可能是纸张上的数据、电子数据

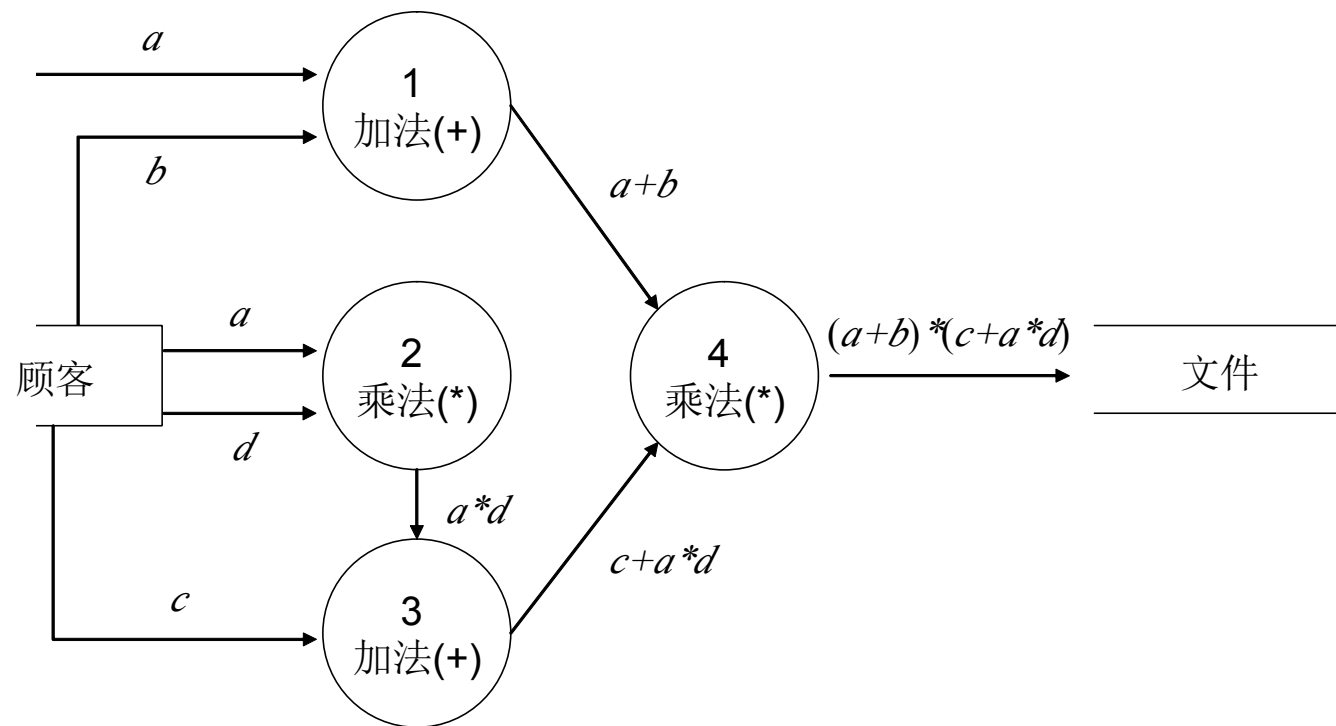
### 销售订单

- 可能存在于:
  - 外部实体与加工之间;
  - 加工与加工之间;
  - 加工与数据存储之间



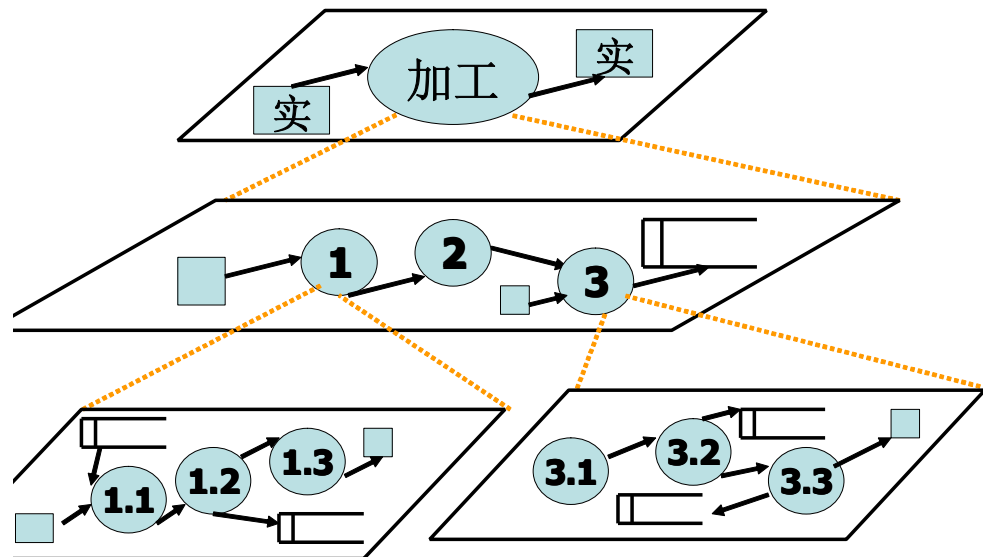
# DFD的简单练习

- 背景：用户输入数据，系统处理后输出到一个文件
- 问题：绘制该系统的DFD



# DFD的层次性

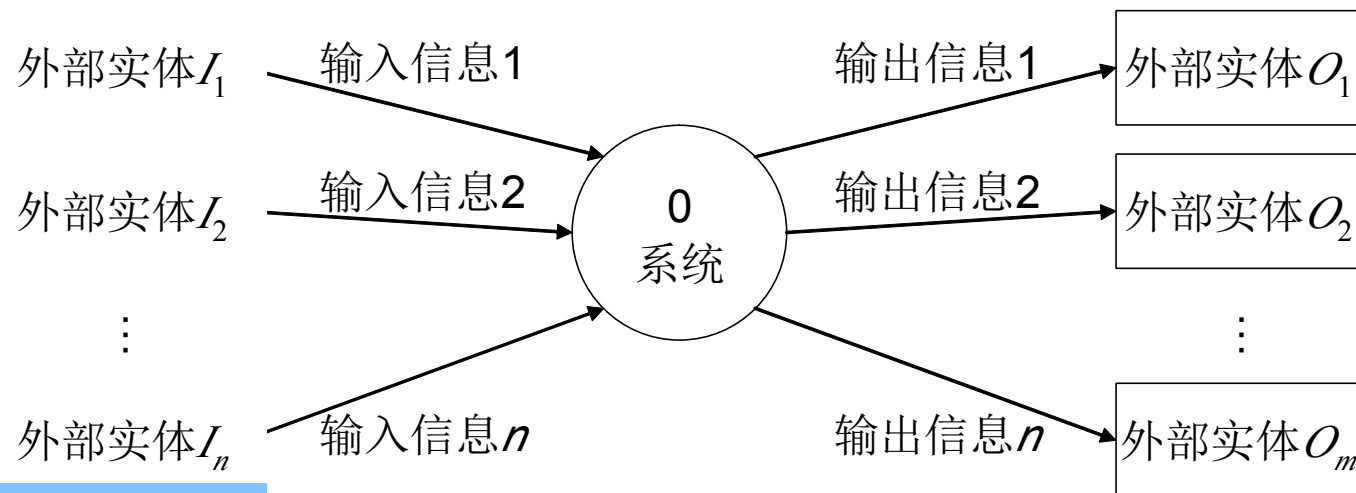
- **DFD的层次性：自顶向下的分解(top-down)**
- **DFD的两种类型：**
  - 环境关联DFD图(Context-level DFD, 或Context Diagram): 也称顶层DFD图, 描述了系统与外部
  - 系统内部DFD图(Inner-level 数据流动关系)
    - 0-层DFD图
    - 1-层DFD图
    - ...
    - N层DFD图



# 顶层DFD

## ■ 顶层DFD图(关联图)

- 通过系统和外部世界之间的联系来描述系统的范围
- 确定了通过某一接口与系统相连的外部实体，同时也确定了外部实体和系统之间的数据流
- 只包含一个加工，用以表示被开发的系统，然后考虑该系统有哪些输入数据、输出数据流
- 编号：0



示例：顶层

旅行社

教务部

课程安排数据

0  
课程注册  
系统

注册请求

学生

班级列表

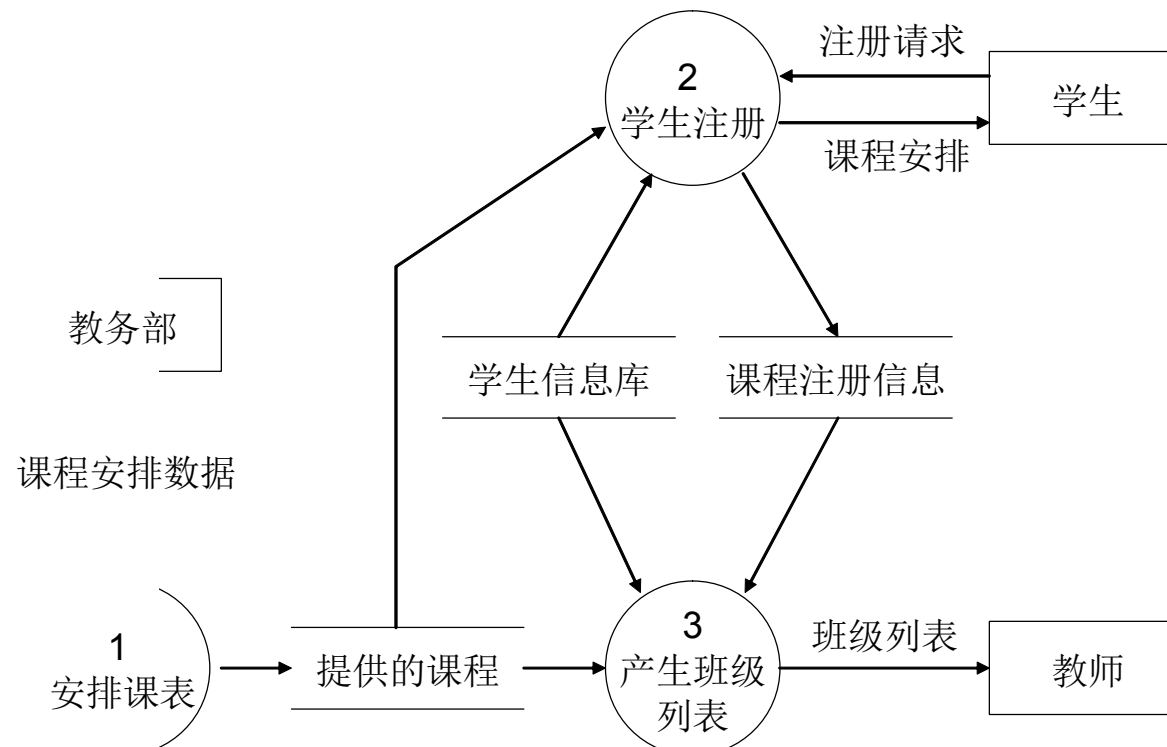
教师

课程安排



## 0层DFD

- 将顶层**DFD**图中的系统分解为若干个子系统，决定每个子系统间的数据接口和活动关系，得到**0层DFD图**；
- 编号：1、2、 --



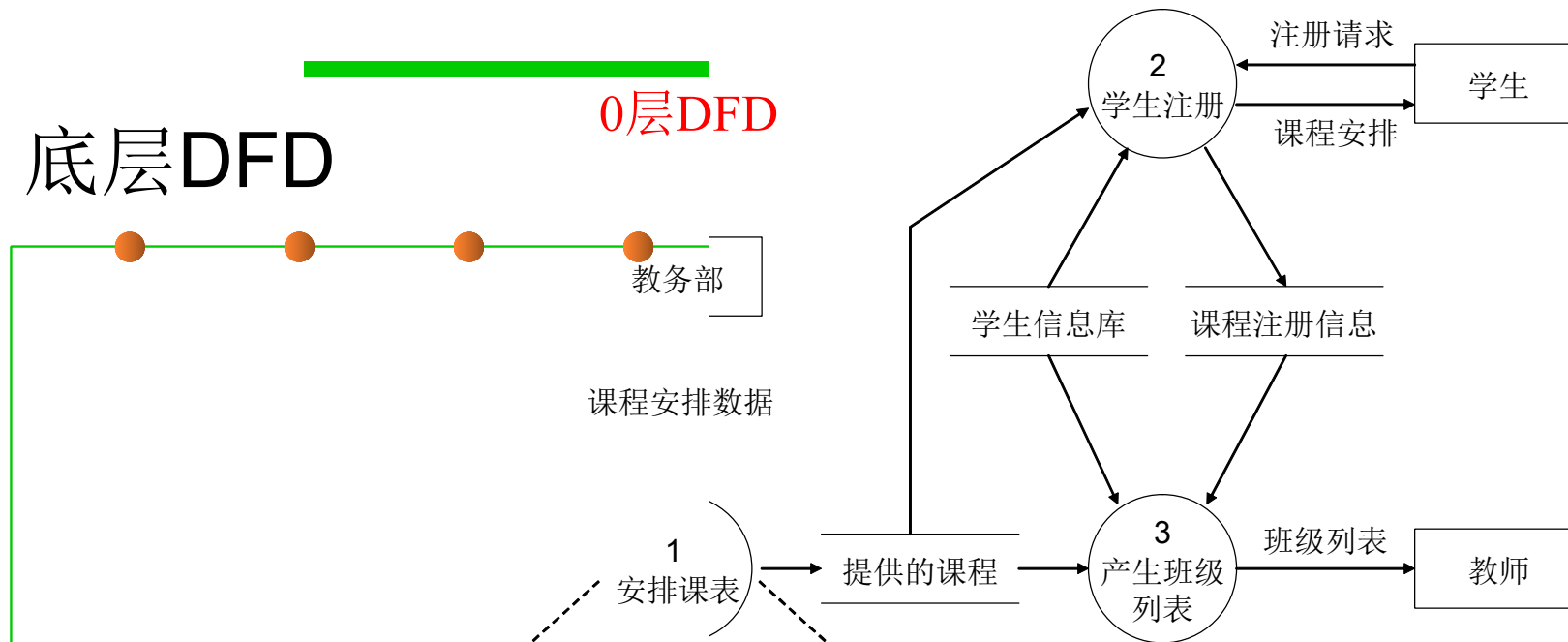
## 底层DFD

- 针对**0层DFD**中的每一个子系统，对其继续分解得到细化的加工，进而逐渐向下构造得到**1层DFD**、**2层DFD**、...、**n层DFD**，一直到不能或不需再分解为止。
- 最底层**DFD**中的加工称为“基本加工”。
- 编号：
  - 1层DFD: 1.1、1.2、...、1.n
  - 2层DFD: 1.1.1、1.1.2、...、1.1.n
  - ...

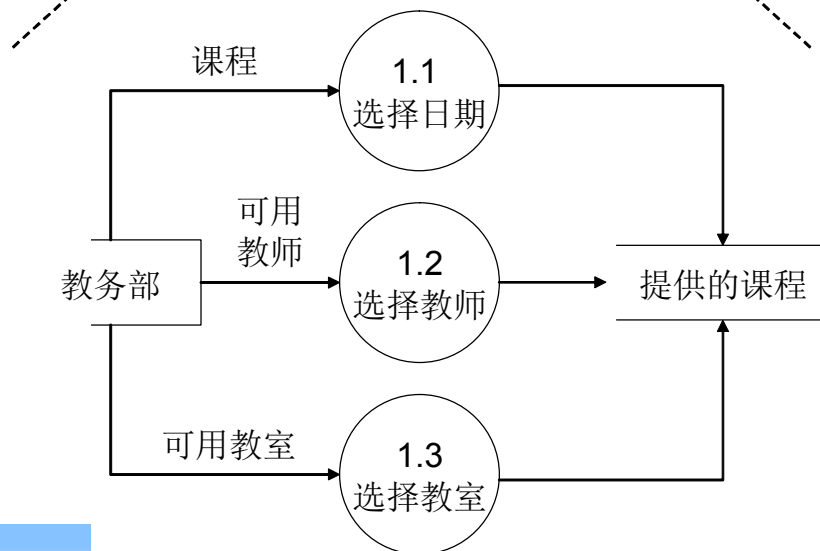


# 底层DFD

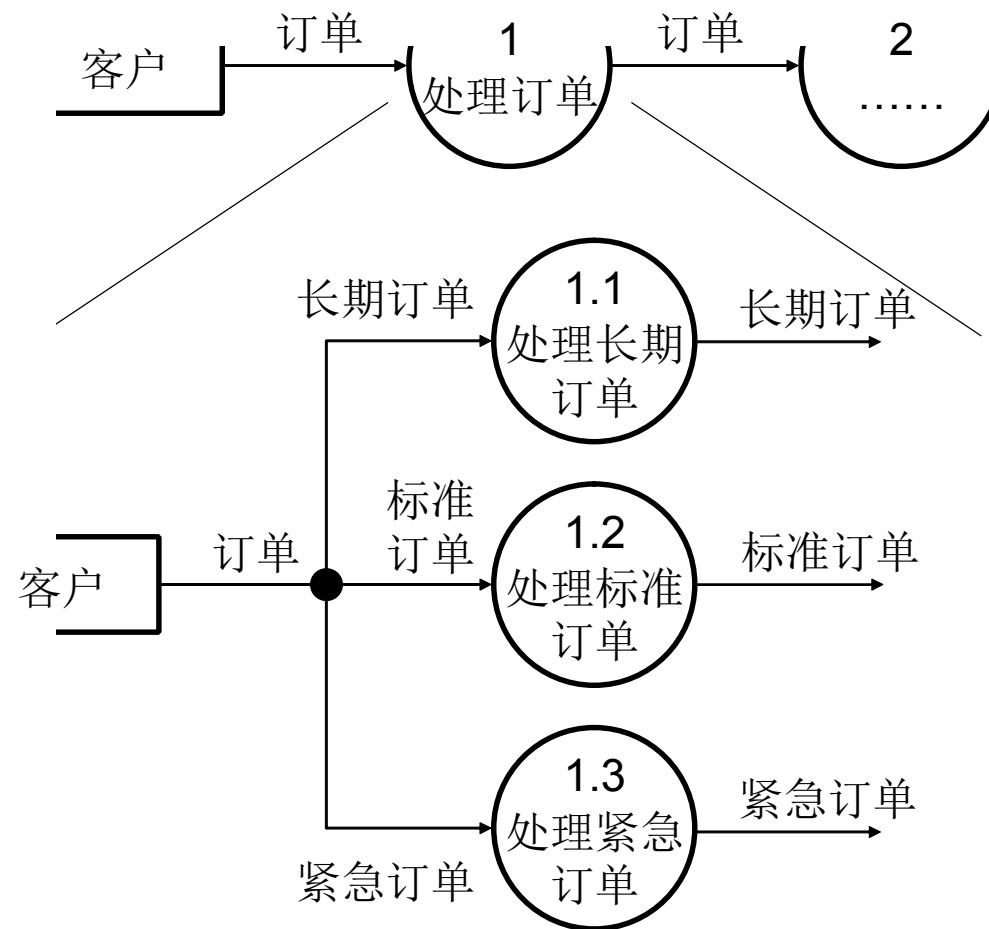
## 0层DFD



## 1层DFD



# 数据流的分解

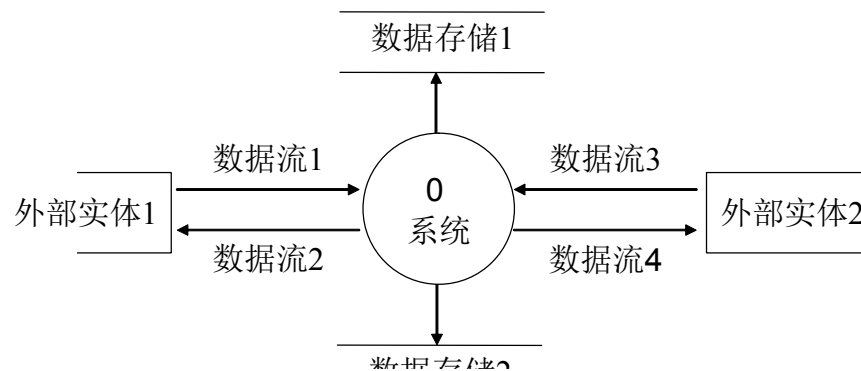


# 如何识别数据流

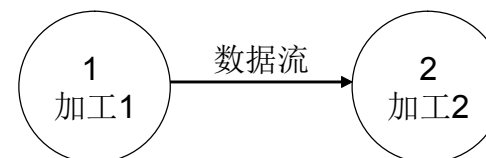
- 通过识别“事件”来识别数据流，进而识别得到加工、数据存储
- 事件的分类：
  - 外部事件(External events): 外部实体与系统进行交互(顾客下订单、供应商的货物到达)
  - 决策事件(Decision events): 需要外部实体为系统某些业务做出决策(是否接受订单)
  - 时间性事件(Temporal events): 由时间所触发的周期性时间(每月25号编制下月计划、每天17点盘点库存)
  - 状态事件(State events): 由某些数据的变化所自动触发的事件(当库存量下降到100以下时，启动采购流程)

# 绘制DFD的一些基本原则

- 把数据存储放在0层数据流图或更上

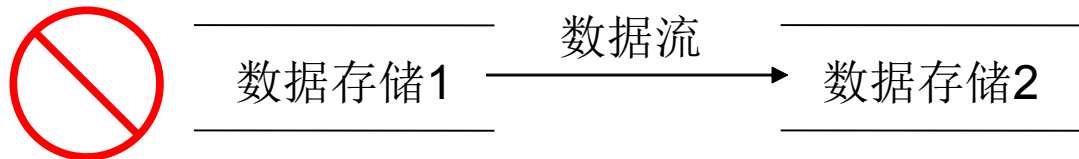


- 使用数据流图时，不要试图运行时的时间特性
- 加工通过数据存储进行通信

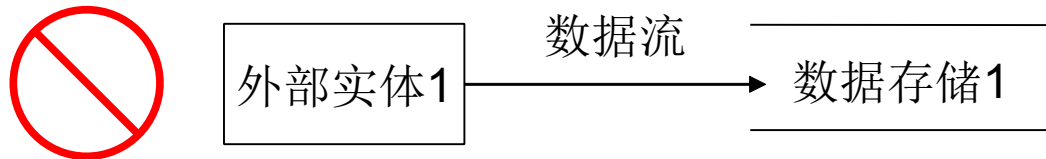


# 绘制DFD的一些基本原则

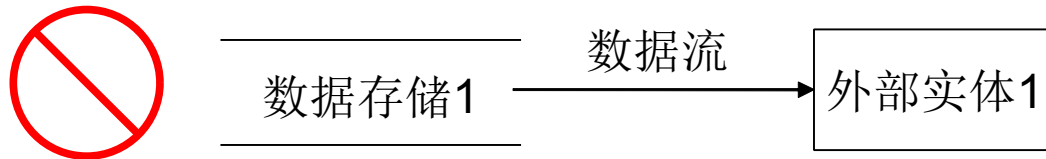
- 数据不能直接由一个数据存储直接流到另一个数据存储



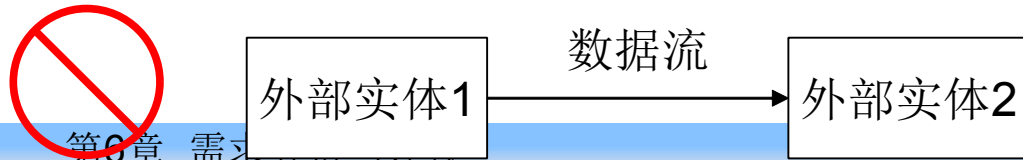
- 数据不能直接从一个外部实体直接流到一个数据存储



- 数据不能直接从一个数据存储直接流到一个外部实体

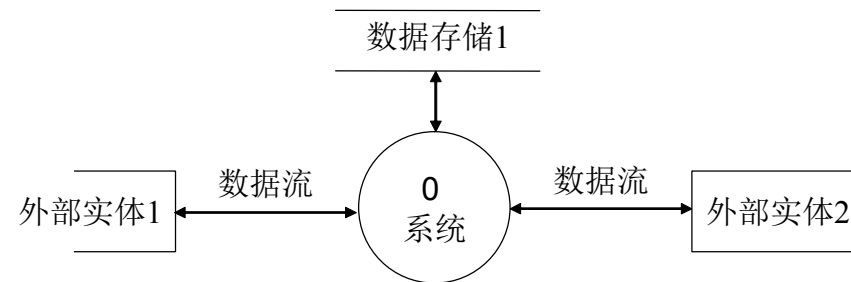


- 数据不能直接在外实体之间流动



# 绘制DFD的一些基本原则

- 数据流是单向的



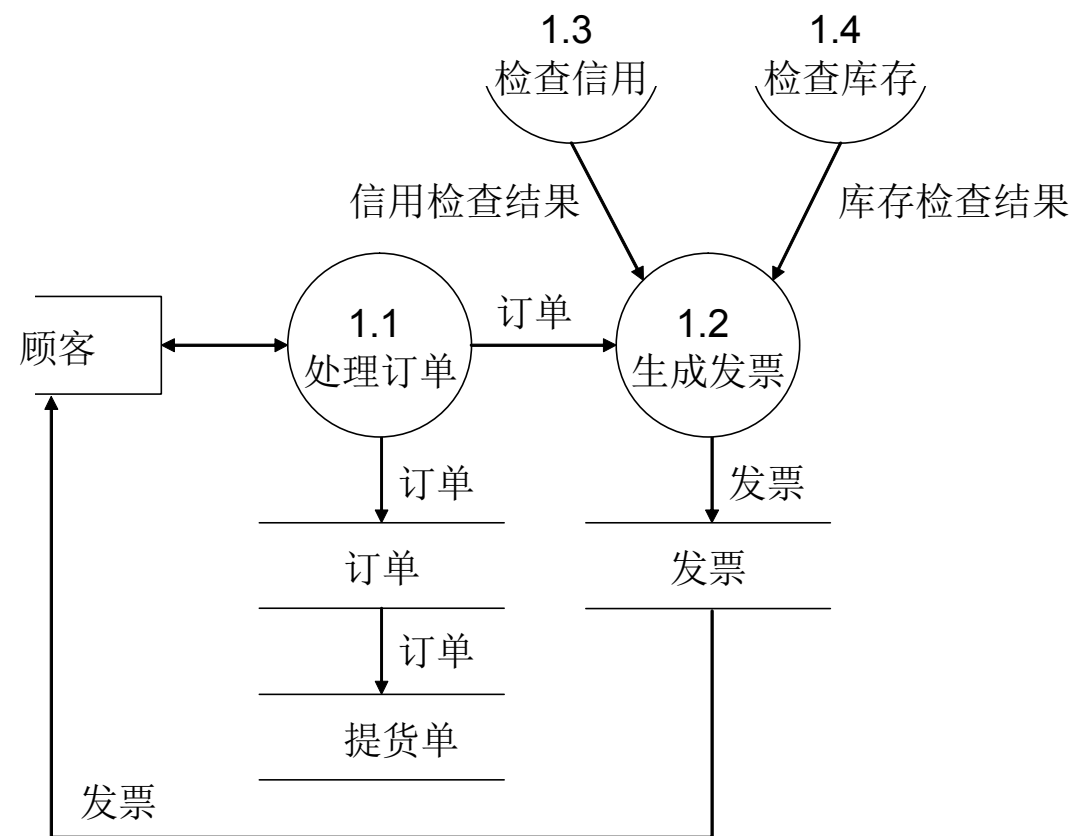
- 任何加工必须有输入和输出数据流



- 对现有加工进行持续的分解和组合，直到所有加工之间达到较高的聚合度；
- 尽量将每一张**DFD**上的所有元素数目控制在**7-12**个。

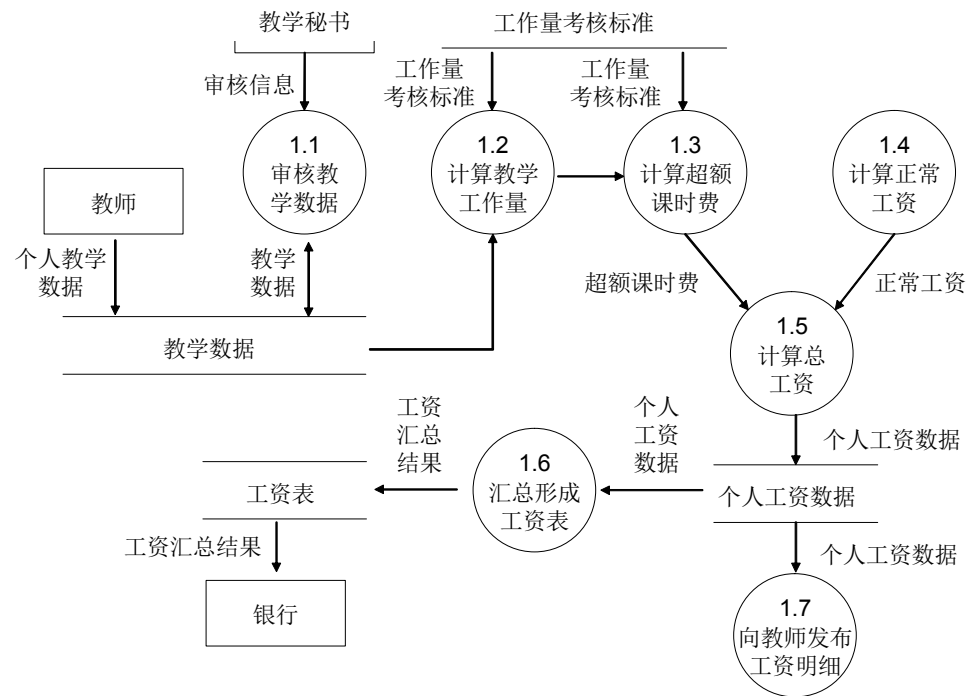
# 错误的DFD

- 找出下面**DFD**中存在的错误，并说明如何修改。



# 错误的DFD

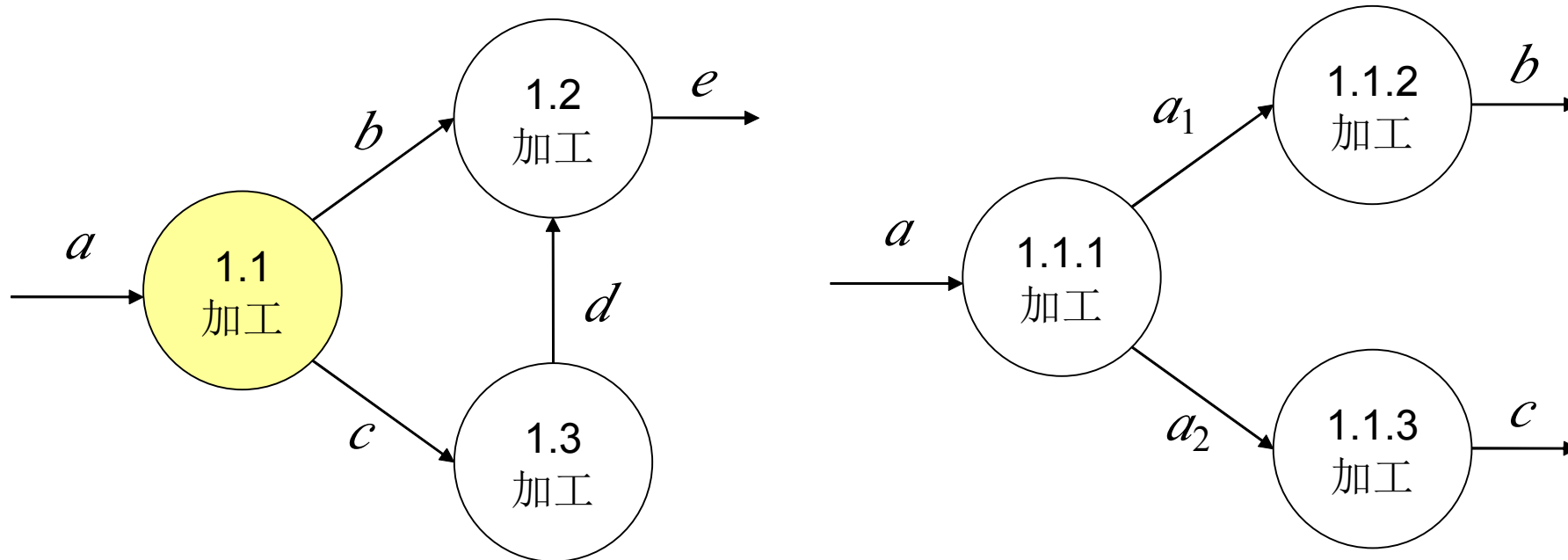
- 教师将个人教学数据录入系统，教学秘书对数据进行审核，发现错误并进行修改。
- 系统根据工作量考核标准和教师的教学数据，计算教师的工作量。
- 教师的工资分为两部分：正常工资和超额课时费。
- 对于前者，系统读取事先已存在的工资标准加以计算；对于后者，系统根据教师的工作量和工作量考核标准来计算得到。
- 得到这两部分工资数据之后，系统计算教师的总工资，并将所有教师的工资数据汇总形成工资表，发送给银行，由银行进行工资发放。
- 与此同时，系统需要向每个教师发布其个人的工资计算结果。





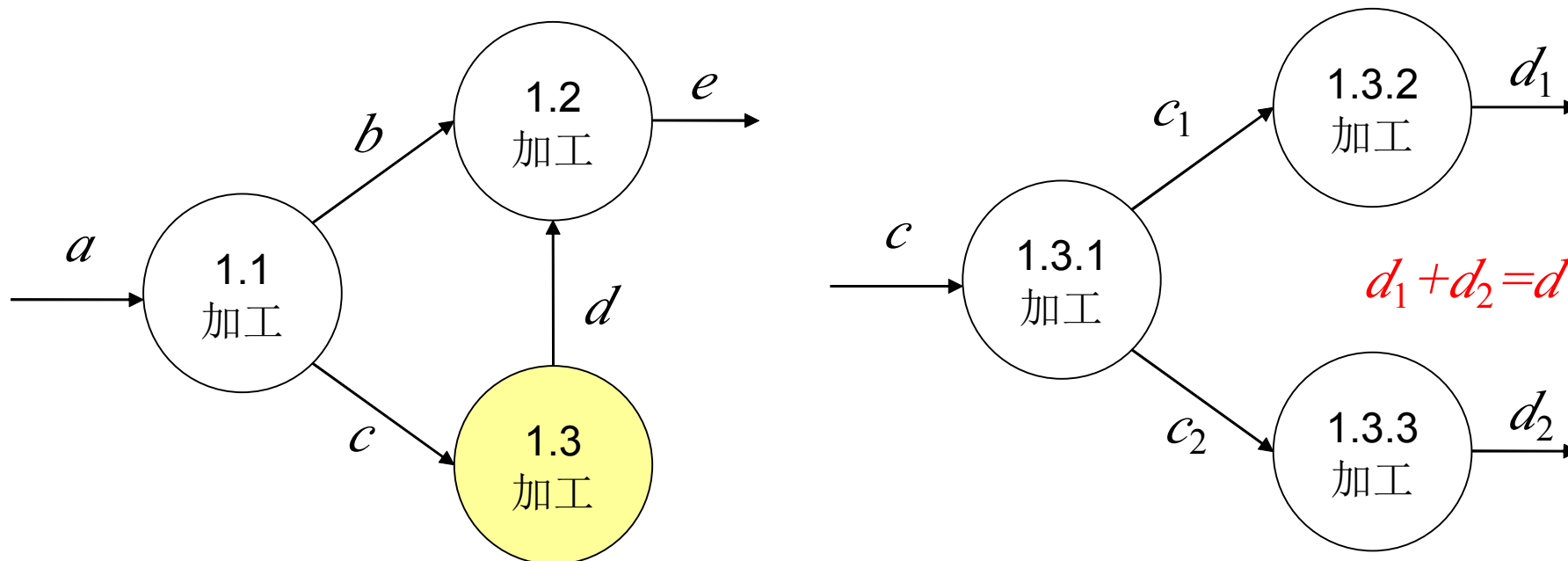
## 父图与子图的平衡

- 下层**DFD**中的输入输出数据流同上层**DFD**中相应加工的输入输出数据流必须一致，此即父图与子图的平衡。



## 父图与子图的平衡

- 数据流本身可以分解，但其包含的数据内容应保持平衡



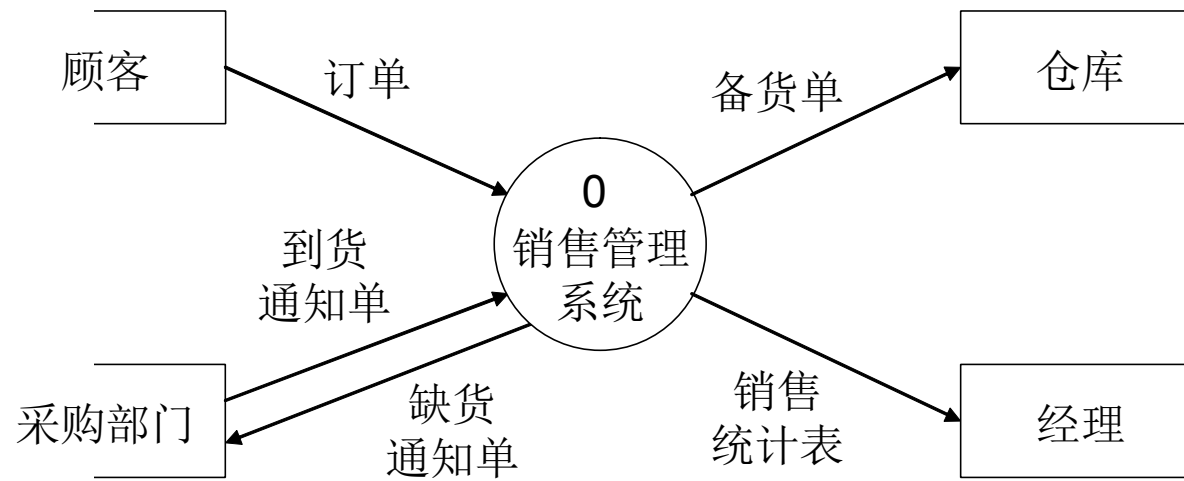
# DFD实例：销售系统

## ■ 某企业销售管理系统：

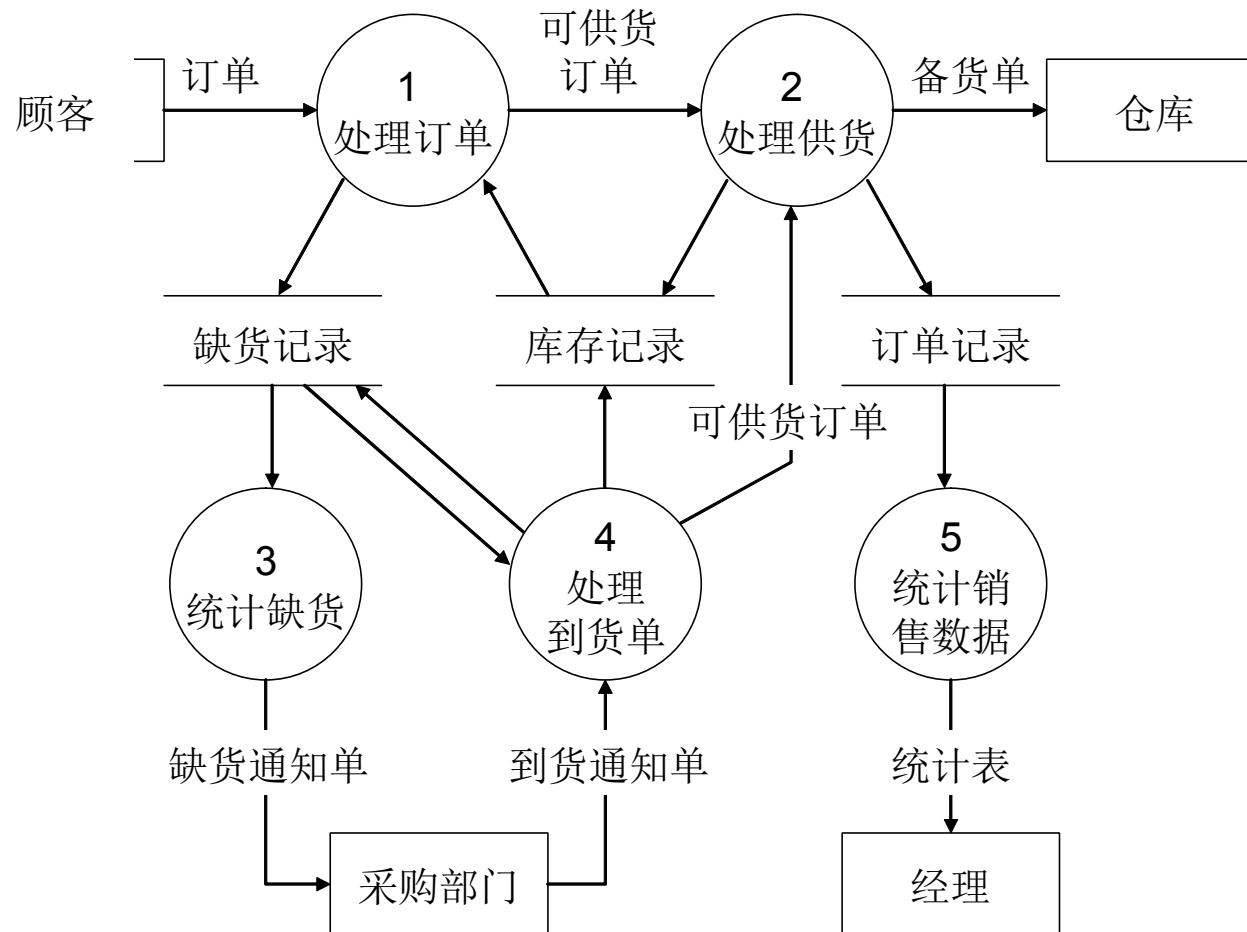
- 接受顾客的订单，检验订单，若库存有货，进行供货处理，即修改库存，给仓库开备货单，并且将订单留底；若库存量不足，将缺货订单登入缺货记录。
- 根据缺货记录进行缺货统计，将缺货通知单发给采购部门，以便采购。
- 根据采购部门发来的进货通知单处理进货，即修改库存，并从缺货记录中取出缺货订单进行供货处理。
- 根据留底的订单进行销售统计，打印统计表给经理。

## ■ 绘制上述系统的顶层、0层、1层DFD图

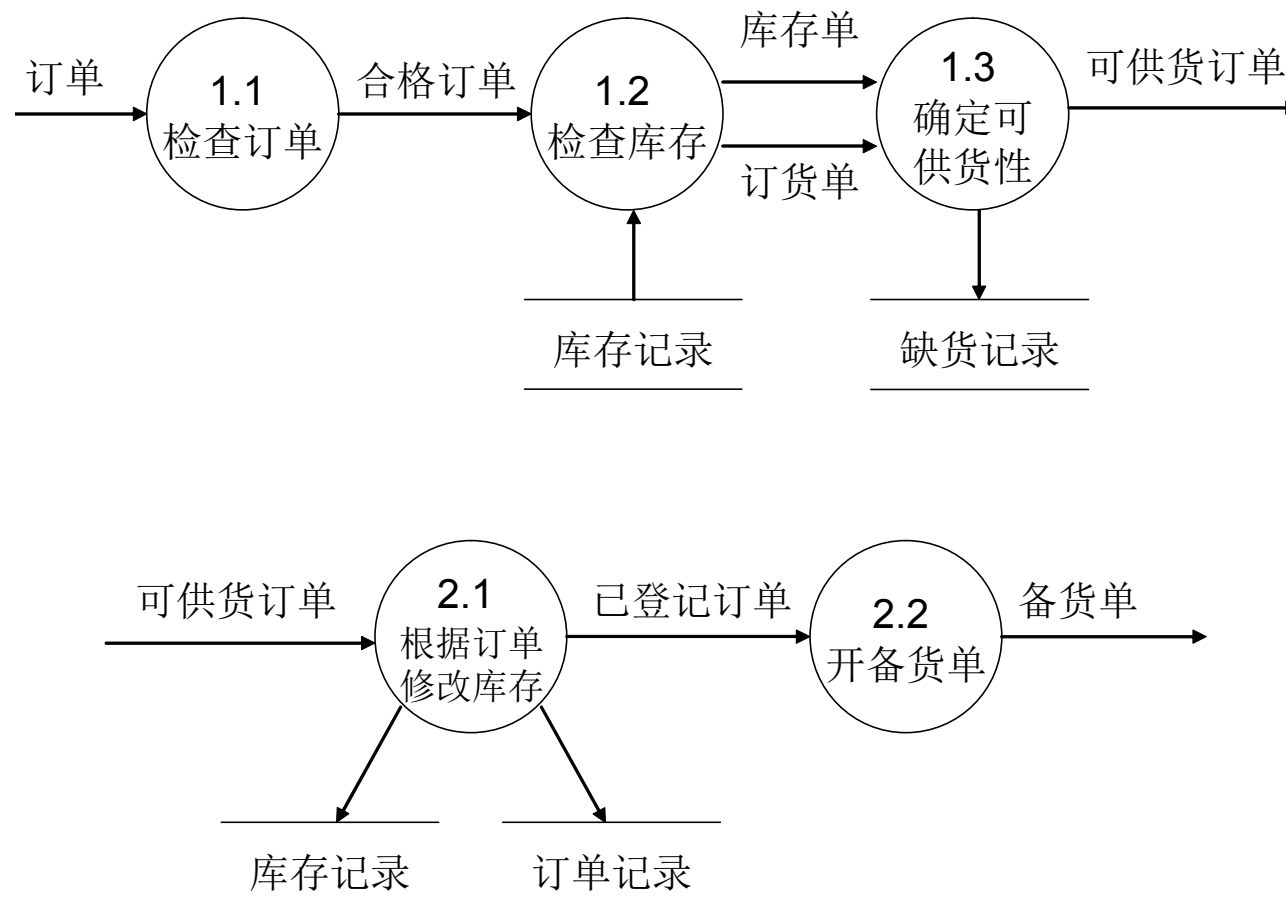
## 销售系统顶,



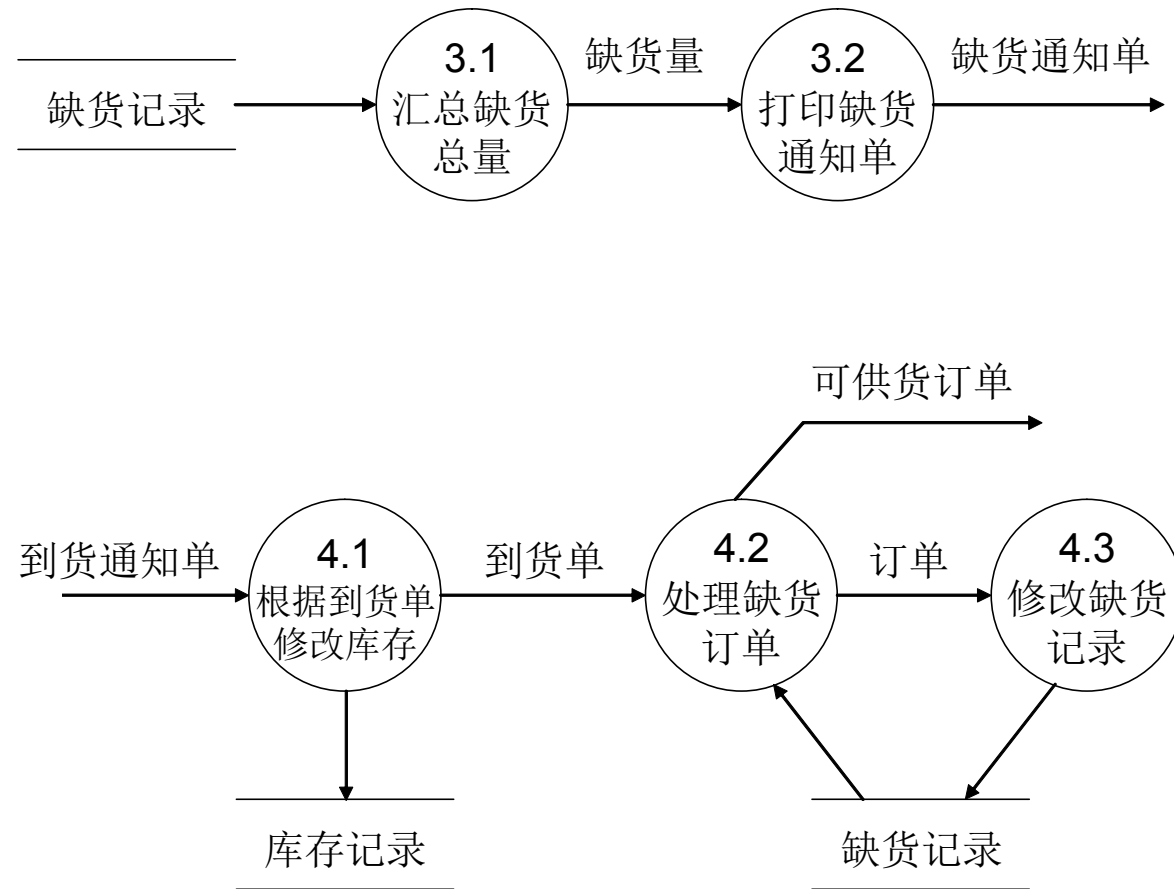
# 销售系统0层DFD



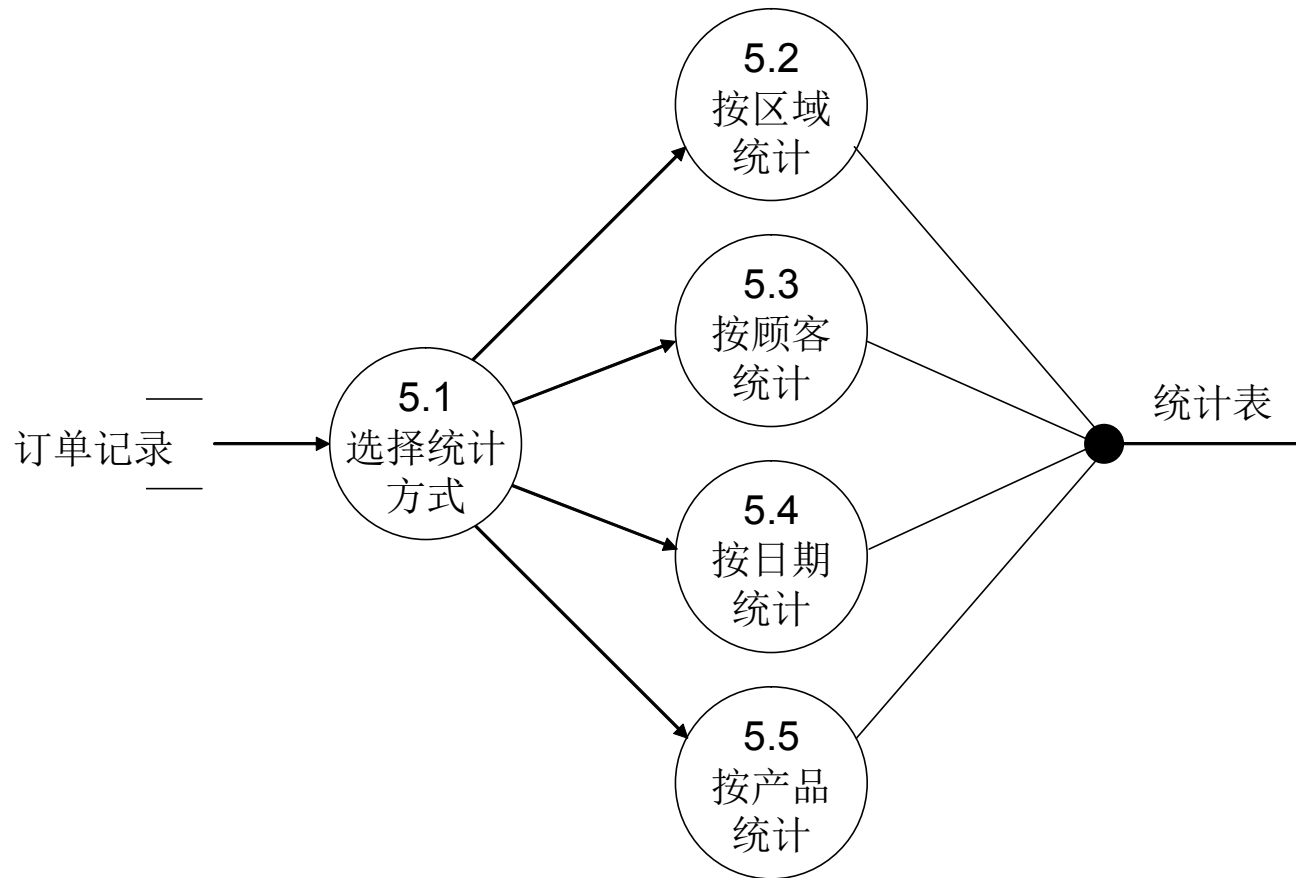
# 销售系统1层DFD



# 销售系统

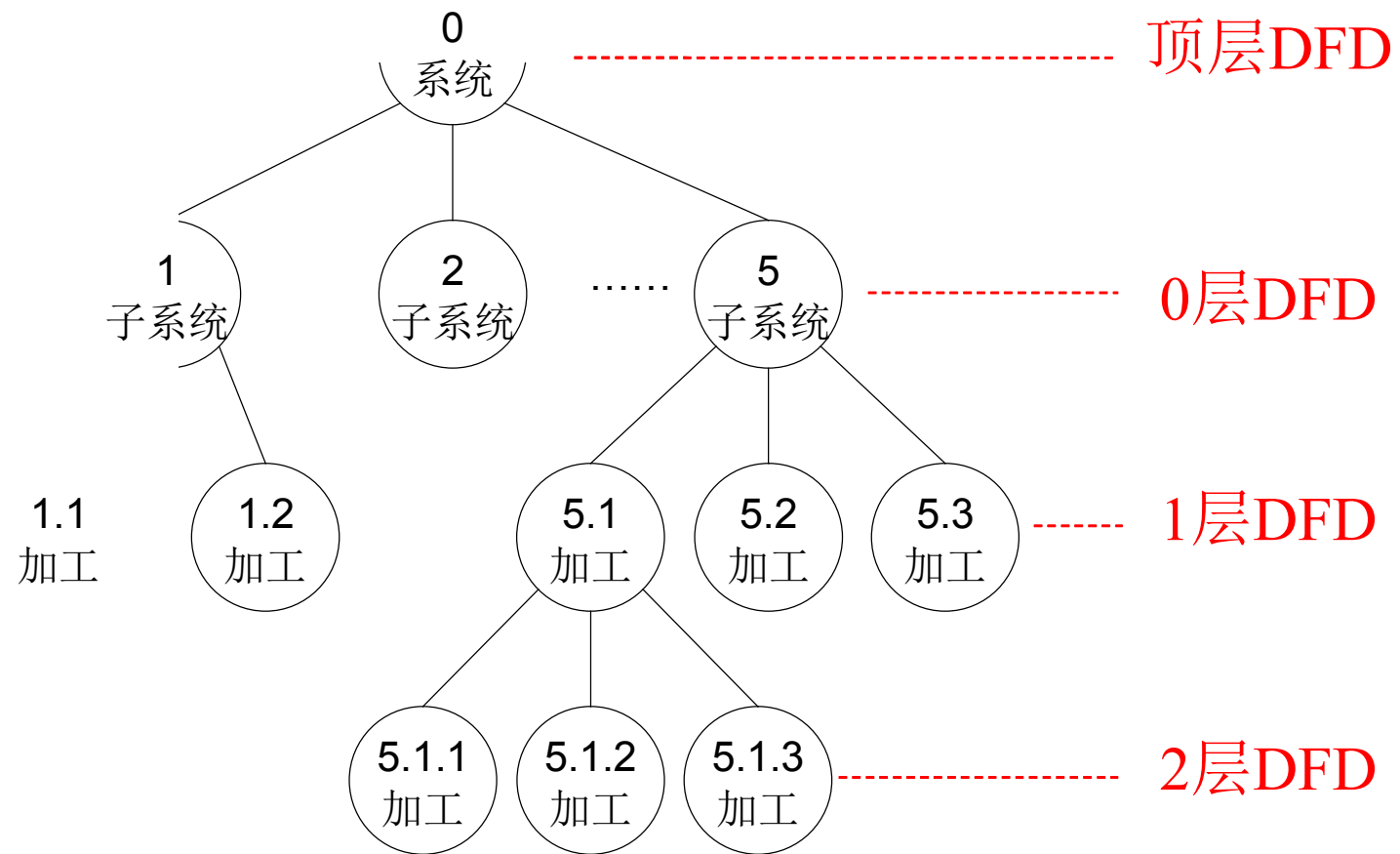


# 销售系统1层DFD





# DFD树



# 主要内容

- **6.1 需求分析**
- **6.2 面向对象需求分析**
- **6.3 结构化需求分析**
  - 6.3.1 结构化分析
  - 6.3.2 数据流图
  - 6.3.3 数据字典

## 2.数据字典(DD)

- **DFD**只是绘制了系统各功能之间的数据流动和处理关系，还需进一步考虑各数据的具体内容。
- 采用数据字典(**Data Dictionary**)作为描述工具
- 对于**DFD**中出现的所有被命名的图形元素(**数据流、数据项、数据存储、加工**)在**DD**中作为一个词条加以定义，使得每一个图形元素的名字都有一个确切的解释。
  - **DD**所有的定义应是严密的、精确的，不可有半点含混，不可有二义性。



## 数据字典的格式

符号	含义	示例
=	被定义为	
+	与	<b>x=a+b</b>
[... ...]	或	<b>x=[a b]</b>
$\{...\}^m_n$	重复(最多 <b>m</b> 次, 最少 <b>n</b> 次)	<b>x={a}</b> <sup>5</sup> <sub>2</sub>
{...}	重复	<b>x={a}</b>
(...)	可选	<b>x=(a)</b>
“...”	基本数据元素	<b>x=“a”</b>
..	连接符	<b>x=1..9</b>

## 数据字典的格式：示例1

国内机票=姓名+身份证号+日期+航班号+起点+终点+费用

姓名=[ {字母}<sup>18</sup><sub>1</sub> | {汉字}<sup>5</sup><sub>2</sub> ]

航班号=[CA|MU|CZ|SC|FM] + “1000”..“9999”

## 数据字典的格式：示例2

存折=户名+开户行号+账号+开户日期+性质+(印密)+存取记录

户名={字母}

开户行号="001".."999"

账号="00000001".."99999999"

开户日期=年+月+日

性质="1".."6"

印密="0"

存取记录=日期+(摘要)+支出+存入+余额+操作员+复核员

# 数据项的定义

- 数据项：不可再分解的数据单位，包括：

- 名称
- 描述
- 数据类型
- 长度(精度)
- 取值范围及缺省值
- 计量单位
- 相关数据元素及数据结构

## 示例

数据项名称：产品编码

别名：Product\_No

简述：用来唯一标识产品的文字

类型：字符串

长度：10

取值范围及含义：

第1位：进口/国产

第2-4位：类别

第5-7位：规格

第8-10位：品名编号

# 数据流的定义

- 数据流来源
- 数据流去向
- 数据流的组成
- 流动属性描述：
  - 频率、数据量等

## 示例

数据流名称：订单

别名：无

简述：顾客订货时填写的项目

来源：顾客

去向：加工1“检验订单”

数据流量：1000份/每周

组成：编号+订货日期+顾客编号+地址+电话+  
银行账号+货物名称+规格+数量

频率：平均40条/天



# 数据存储的定义

- 文件名
- 描述
- 数据结构
- 数据存储方式
- 关键码
- 存取频率和数据量
- 安全性要求

## 示例

数据存储名称：库存记录

别名：无

简述：存放库存所有可供产品的信息

组成：产品编号+名称+生产厂家+单价+库存量

组织方式：索引文件，以产品编号为关键字

查询要求：要求能随时查询

## 示例：数据字典(1)

- 某酒店所提供的电话服务系统的功能为：
  - 客人可以通过拨分机号联络酒店内的其他房间，也可拨外线号码与酒店外联络。
  - 分机号从8201至8299。
  - 外线号码需先拨0，然后加拨市话号码或长途电话号码。
  - 长途电话号码由区号和市话号码组成，其中区号可以为010、021~029、0300~0999中的任意一个数字串。
  - 市话号码是任意7位或8位长度的数字串。

## 示例：数据字典(1)

- “电话号码”的数据字典：
  - 电话号码=[分机号|外线号码]
  - 分机号=8201..8299
  - 外线号码=0+[市话号码|长途电话号码]
  - 市话号码={数字}
  - 长途电话号码=区号+市话号码
  - 区号=[010|021..029|0300..0999]

名字:	零件编号
别名:	编码
描述:	唯一的标识库存清单中一个特定零件
定义:	零件编号={字符} <sup>8</sup>
位置:	库存清单 采购订单 订货报表

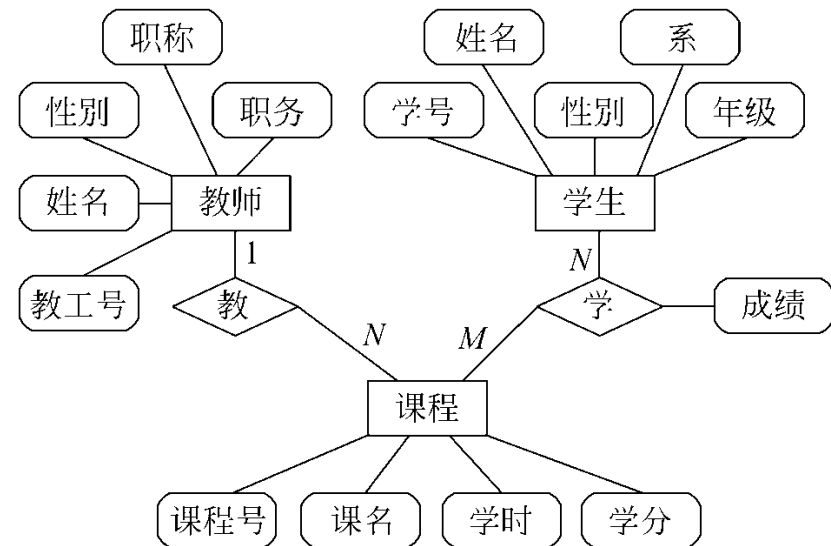
名字:	采购订单
别名:	采购单
描述:	由各部门采购人员定期向供应商发出的单据
定义:	采购订单=供应商ID+供应商名称+采购日期+总金额+ {零件编号+零件名称+数量+价格}
位置:	订货报表

# 数据建模——实体联系图(ER图)\*

## ■ 实体联系图(Entity-Relationship Diagram)

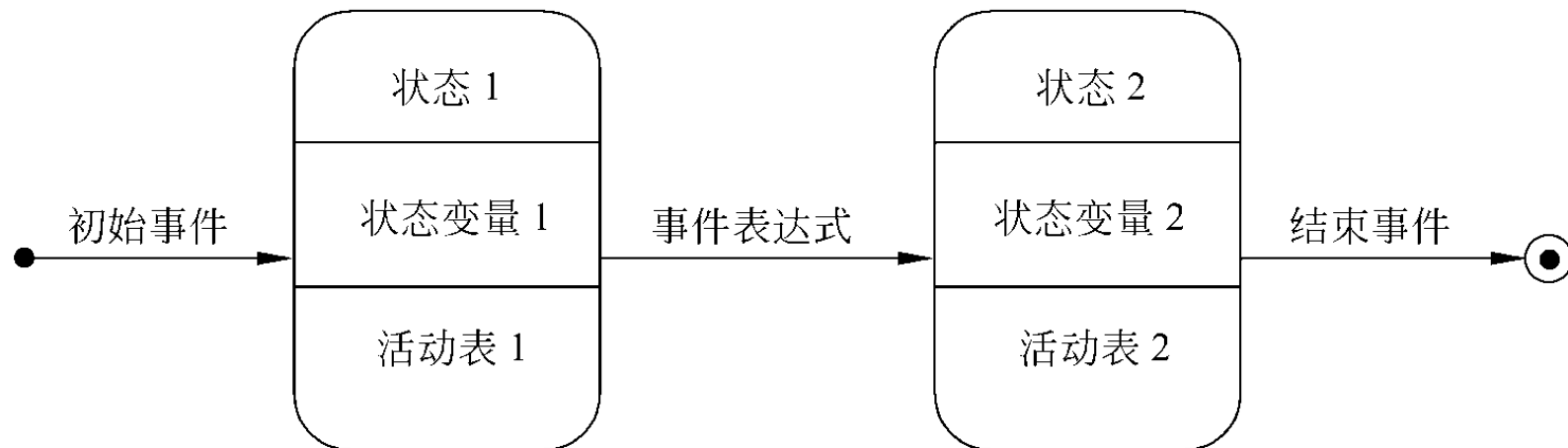
- 目标：把用户的数据要求清除、准确的描述出来，需建立一个概念性的数据模型(或称信息模型)。
- 形式：系统中包含哪些数据实体(entity)、每个实体所具备的主要属性(attribute)、这些实体之间存在哪些关联关系(relationship)；

关于E-R的细节，  
在《数据库系统》课程中学习



## 状态转换图\*

- **DFD+DD+结构化英语**：功能模型
- **E-R**：信息模型
- 需求分析阶段还需要建立起软件系统的行为模型：状态转换图(**State Transition Diagram**)
  - 通过描绘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；
  - 指明了作为特定事件的结果系统将做哪些动作。



# 状态转换图的基本要素(1)

## ■ 状态(state)

- 是任何可以被观察到的系统行为模式，一个状态代表系统的一种行为模式。
- 规定了系统对事件的响应方式。
- 系统对事件的响应，既可以是做一个(或一系列)动作，也可以是仅仅改变系统本身的状态，还可以是既改变状态又做动作。
- 在状态图中定义的状态主要有：初态(即初始状态)、终态(即最终状态)和中间状态。
- 在一张状态图中只能有一个初态，而终态则可以有0至多个。

## ■ 事件(event)

## 状态转换图的基本要素(2)

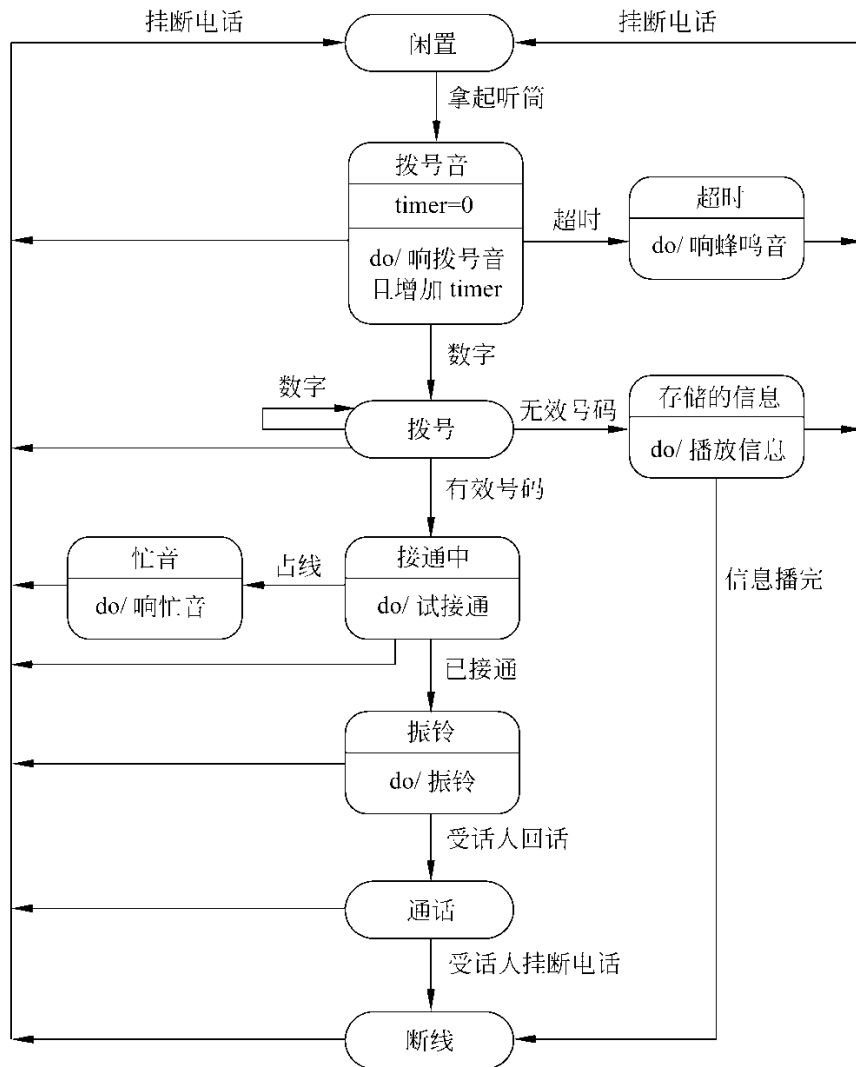
- 状态(**state**)

- 事件(**event**)

- 在某个特定时刻发生的事情，它是对引起系统做动作或(和)从一个状态转换到另一个状态的外界事件的抽象。
- 例如，内部时钟表明某个规定的时间段已经过去，用户移动或点击鼠标等都是事件。
- 简而言之，事件就是引起系统做动作或(和)转换状态的控制信息。

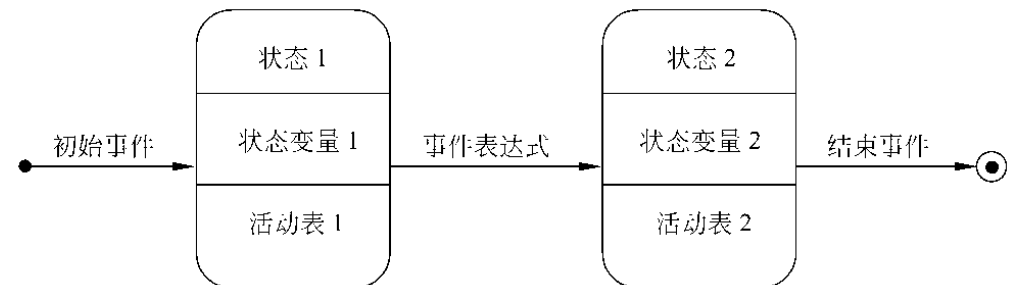


# 状态转换图的图形化符号



- 无人打电话时电话处于闲置状态；
- 有人拿起听筒则进入拨号音状态，到达这个状态后，电话的行为是响起拨号音并计时；
- 这时如果拿起听筒的人改变主意不想打了，他把听筒放下(挂断)，电话重又回到闲置状态；
- 如果拿起听筒很长时间不拨号(超时)，则进入超时状态；

...

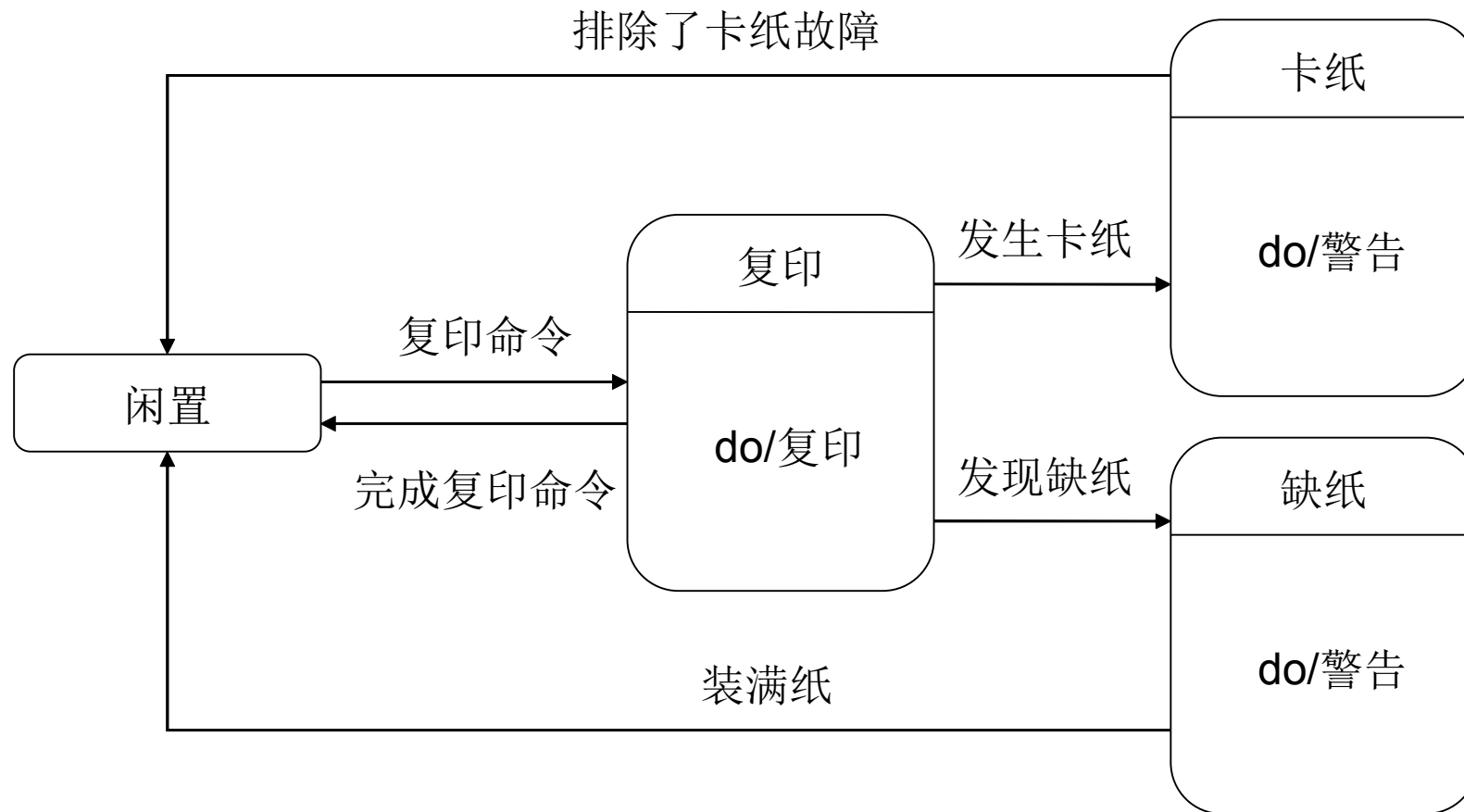


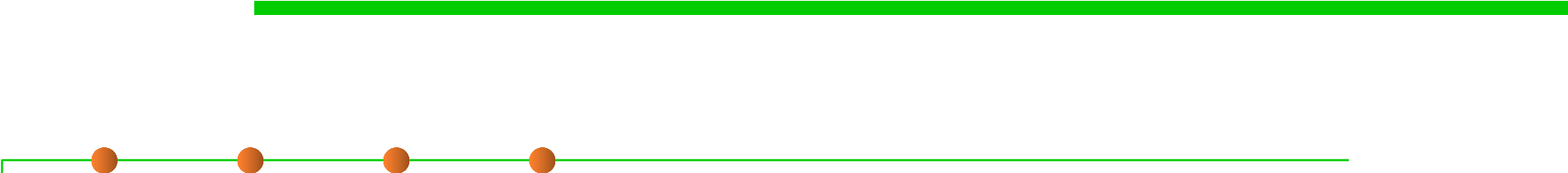
# 状态转换图示例

## ■ 复印机的工作过程为：

- 未接受到复印命令时处于闲置状态，一旦接受到复印命令则进入到复印状态；
- 完成一个复印命令规定的工作后又回到闲置状态，等待下一个复印命令；
- 如果执行复印命令时发现缺纸，则进入缺纸状态，发出警告，等待装纸；
- 装满纸后进入闲置状态，准备接收复印命令；
- 如果复印时发生卡纸故障，则进入卡纸状态，发出警告等待维修人员来排除故障；
- 故障排除后回到闲置状态。

## 状态转换图示例





结束

2011年4月27日