需求分析方法

刘钦

预习测试

- 根据下面用例描述建立系统顺序图
  - 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. 顾客在推荐列表中选定一个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.

预习测试

项目	内容描述
ID	用例的标识
名称	对用例内容的精确描述，体现了用例所描述的任务
参与者	描述系统的参与者和每个参与者的目标
触发条件	标识启动用例的事件，可能是系统外部的事件，也可能是系统内部的事件，还可能是正常流程的第一个步骤
前置条件	用例能够正常启动和工作的系统状态条件
后置条件	用例执行完成后的系统状态条件
正常流程	在常见和符合预期的条件下，系统与外界的行为交互序列
扩展流程	用例中可能发生的其他场景
特殊需求	和用例相关的其他特殊需求，尤其是非功能性需求

- 写出下列用例的详细用例描述
  - 顾客首先查询CD的推荐列表，然后顾客在推荐列表中选定一个CD，接着要求查看更详细的信息，最后顾客购买选定CD，离开.

知识点

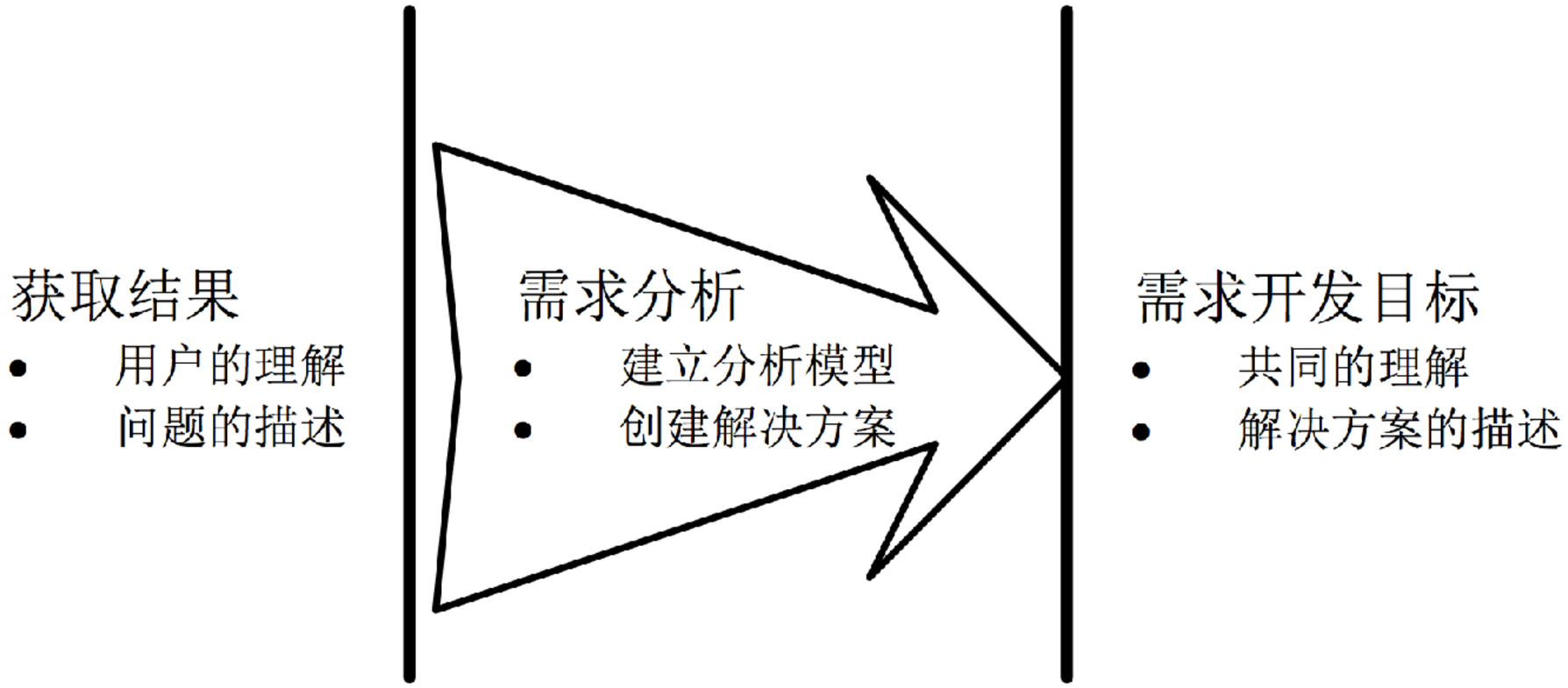
- 需求分析的任务（P7，P8）
- 常见的分析模型（P11，P12)
- 面向对象分析的过程（P14）
- 用例的定义（P17）
- 用例的基本元素（P20)
- 用例图的建立过程（P25）
- 用例模板（P37）
- 概念类图的定义（P44）
- 概念类图的建立过程（P50）
- 顺序图的概念（P63，P64）
- 状态图的概念和图例（P72，P73）
- 建立状态图的过程（P76）
- 结构化分析的过程（P85）
- 数据流图（P87）
- 实体关系图（P103）
- 细化和明确需求（P114）
- 建立系统需求（P118）

Outline

- 需求分析基础
- 面向对象分析
- 结构化分析
- 使用需求分析方法细化和明确需求

Outline

- 需求分析基础
  - 为什么要需求分析
  - 需求分析模型
- 面向对象分析
- 结构化分析
- 使用需求分析方法细化和明确需求



为什么要需求分析

需求分析的任务[Bin2009]

- 建立分析模型，达成开发者和用户对需求信息的共同理解。

- 依据共同的理解，发挥创造性，创建软件系统解决方案。

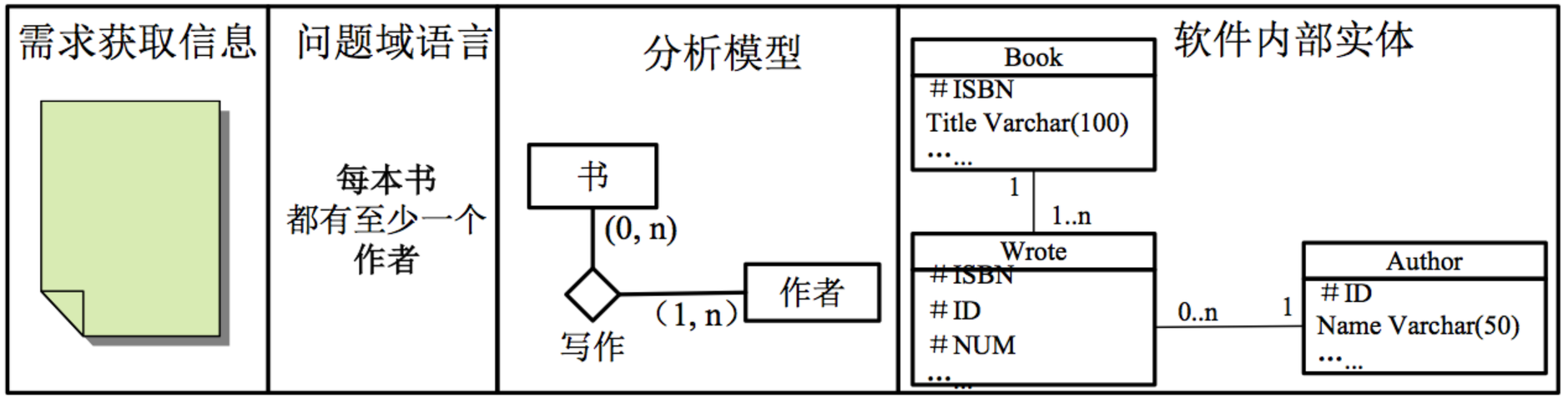
需求分析模型与建模

- “模型是对事物的抽象，帮助人们在创建一个事物之前可以有更好的理解”[Blaha2005]

- 建立模型的过程被称为建模。“它是对系统进行思考和推理的一种方式。建模的目标是建立系统的一个表示，这个表示以精确一致的方式描述系统，使得系统的使用更加容易”[Fishwick1994]。

建模常用手段

- 抽象（Abstraction）
- 分解（Decomposition / Partitioning）


需求获取信息
问题域语言
每本书
都有至少一个
作者
分析模型
书
(0, n)
(1, n)
作者
写作
软件内部实体
Book
#ISBN
Title Varchar(100)
…...
1
1..n
Wrote
#ISBN
#ID
#NUM
…...
0..n
1
Author
#ID
Name Varchar(50)
…...

需求分析模型

方法	模型	描述
结构化方法	数据流图 Data Flow Diagram	从数据传递和加工的角度，描述了系统从输入到输出的功能处理过程。运用功能分解的方法，用层次结构简化处理复杂的问题。
	实体关系图 Entity Relationship Diagram	描述系统中的数据对象及其关系，定义了系统中使用、处理和产生的所有数据。
面向对象方法	用例图 Use-Case Diagram	描述用户与系统的交互。从交互的角度说明了系统的边界和功能范围。
	类图 Class Diagram	描述应用领域当中重要的概念以及概念之间的关系。它捕获了系统的静态结构。
	交互图（顺序图） Interaction(Sequence) Diagram	描述系统中一次交互的行为过程，说明了在交互当中的对象协作关系。
	状态图 State Diagram	描述系统、用例或者对象在其整个生命期内的状态变化和行为过程。

常见分析模型

Outline

- 需求分析基础
- 面向对象分析
  - 用例图与用例描述
  - 概念类图
  - 顺序图
  - 状态图
- 结构化方法

用例模型和对象模型之间的鸿沟

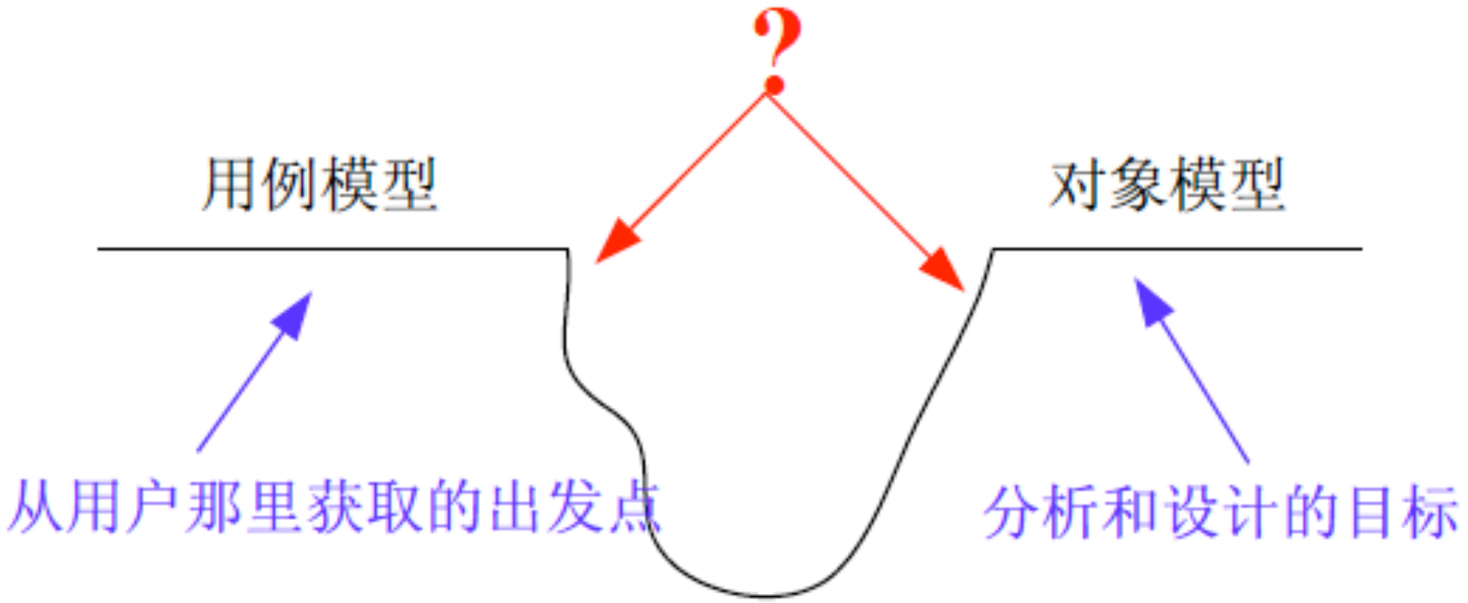
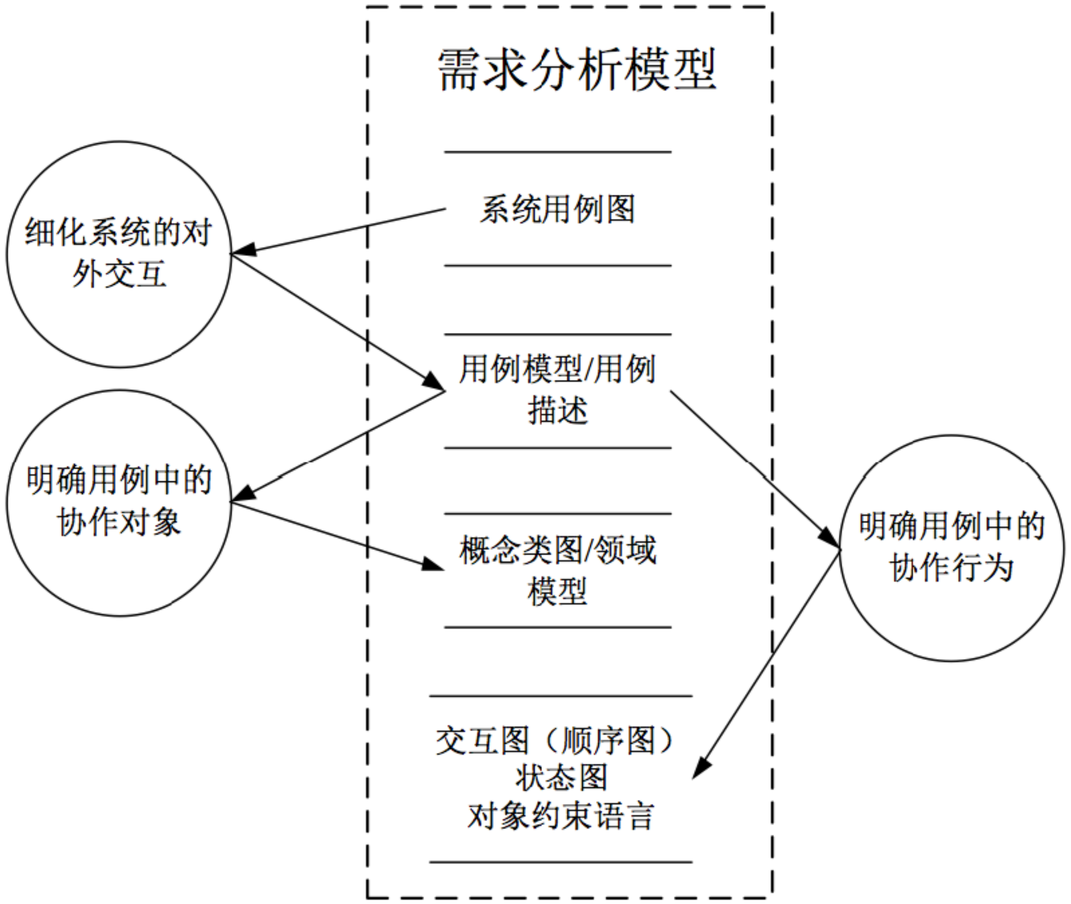


图 14－17、用例模型和对象模型之间的鸿沟



面向对象分析的简单过程

需求与用例

- Traditionally, the requirements are stipulated in a contractual document between the client and the developer….
  - It is hard to capture all the requirements.
- In 1992, Jacobson proposed the Use Case approach.
  - They come from traditional development methods and are adapted to OOAD.

用例

- 用例最初由[Jacobson1992] 在 Objectory 方法中提出的,它将用例定义为“在系统(或者子系统或者类)和外部对象的交互当中所执行的行为序列的描述,包括各种不同的序列和错误的序列,它们能够联合提供一种有价值的服务”[Rumbaugh2004]。

- [Cockburn2001]认为用例描述了在不同条件下系统对 某一用户的请求的响应。根据用户的请求和请求时的系统条件,系统将执行不同的行为序列, 每一个行为序列被称为一个场景。一个用例是多个场景的集合。

Example: Essential Use Case

Actor Action	System Response
1. The Customer identifies themselves.	2. Presents Options.
3.	4. .....

How a customer identifies themselves may change over time, but the identification itself is essential.

Example: **Real Use Case**

Actor Action	System Response
1. The Customer inserts their card.	2. Prompts for PIN.
3. Enters PIN on keypad.	4. Displays options menu.
5.	5. .....

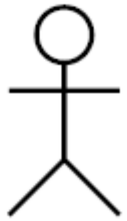
目标、交互与行为序列

案例

- 连锁超市管理系统的收银员为了完成一次销售任务,会使用软件系统处理销售过程,那么就可以建立一个用例“销售处理”。考虑实际销售时的不同条件,会发生不同的行为:
  - 在一切顺利时是一种正常行为流程;
  - 购买多个同样商品时可以逐一输入每个商品,也可以分别输入商品号与数量;
  - 销售过程中可能会发现某个商品无法识别;
  - 有可能一个商品被纳入销售清单后用户又提出退回.....
- 上述的每一个行为都是一个场景。所有的行为联合起来就构成了场景的集合——用例,它的目标与价值是完成销售任务。

用例图基本元素

- 用例
- 参与者
- 关系
- 系统边界

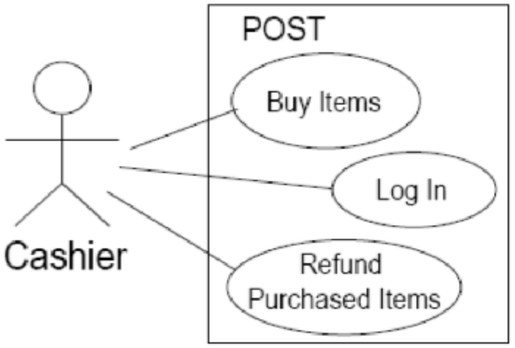


Actor

- An actor is a role that a user or other system plays with respect to the system to be developed.
- A single actor in a use case diagram can represent multiple users (or systems) .
- A single user (or system) also may play multiple roles.
- Actors don‘t need to be human, e.g., an external system that needs some information from the current system is also an actor.

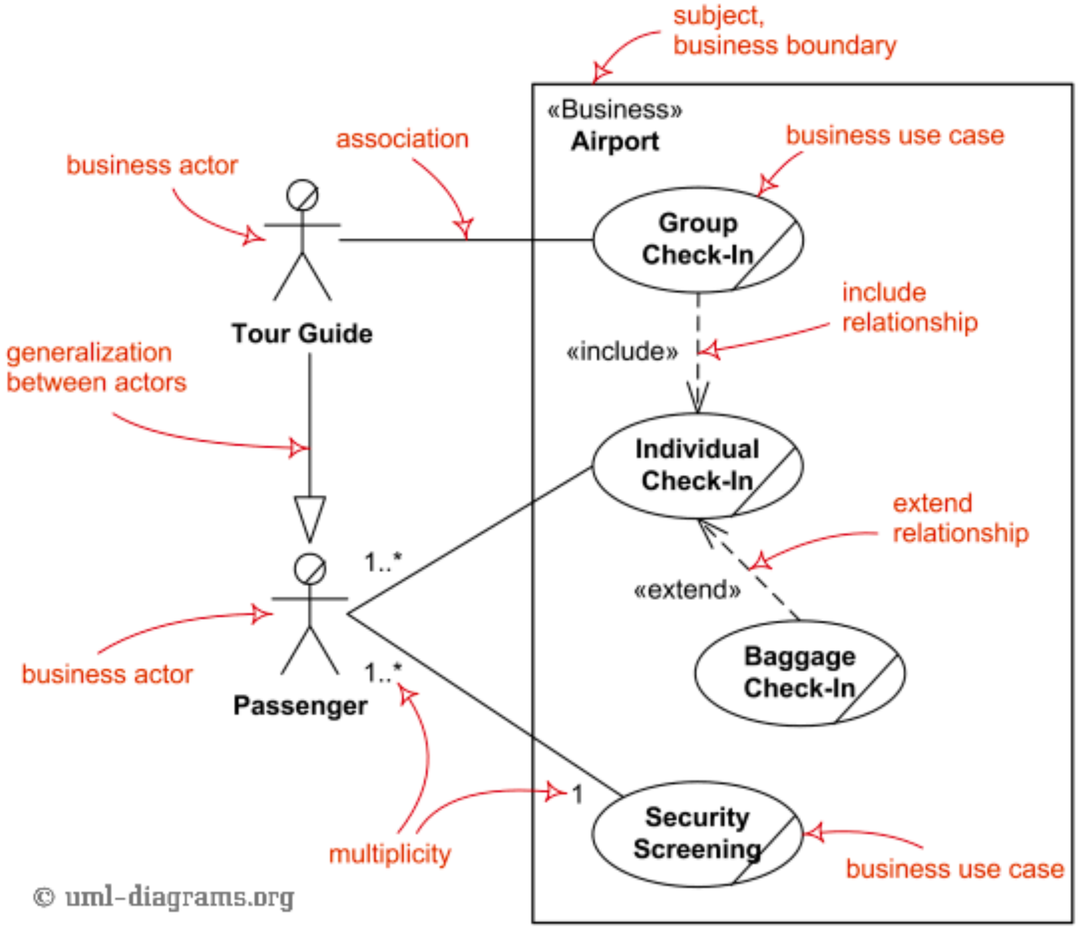
Use Case: requirements in context

- Express requirements in the form of use cases.
- Use cases represent typical sets of scenarios that help to structure, relate and understand the essential requirements.
- Scenarios are descriptions of how a system is used in practice
  - Typical interactions between a user and a computer system.



System Boundary

- Emphasis the focus on what is going to be detailed and what is not.
- The system boundary is implicitly existent in a diagram without an explicitly represented system boundary
- Actors are always outside the boundary and use cases are always inside the boundary.



Relationship

用例图的建立

- 目标分析与解决方向的确定
- 寻找参与者
- 寻找用例
- 细化用例

目标分析

- 问题目标解决方案
- ×××连锁商店是一家刚刚发展起来的小型连锁商店，其前身是一家独立的小百货门面店。
  - 首先是随着商店规模的扩大，顾客量大幅增长，手工作业销售迟缓，顾客购物排队现象严重，导致流失客源。
  - 其次是商店的商品品种增多，无法准确掌握库存，商品积压、缺货和报废的现象上升明显。
  - 再次是商店面临的竞争比以前更大，希望在降低成本，吸引顾客，增强竞争力的同时，保持盈利水平。

业务需求

- BR1：在系统使用6个月后，商品积压、缺货和报废的现象要减少50%
- BR2：在系统使用3个月后，销售人员工作效率提高50%
- BR3：在系统使用6个月后，运营成本要降低15%
  - 范围：人力成本和库存成本
  - 度量：检查平均员工数量和平均每10,000元销售额的库存成本
- BR4：在系统使用6个月后，销售额度要提高20%
  - 最好情况：40%
  - 最可能情况：20%
  - 最坏情况：10%

系统功能

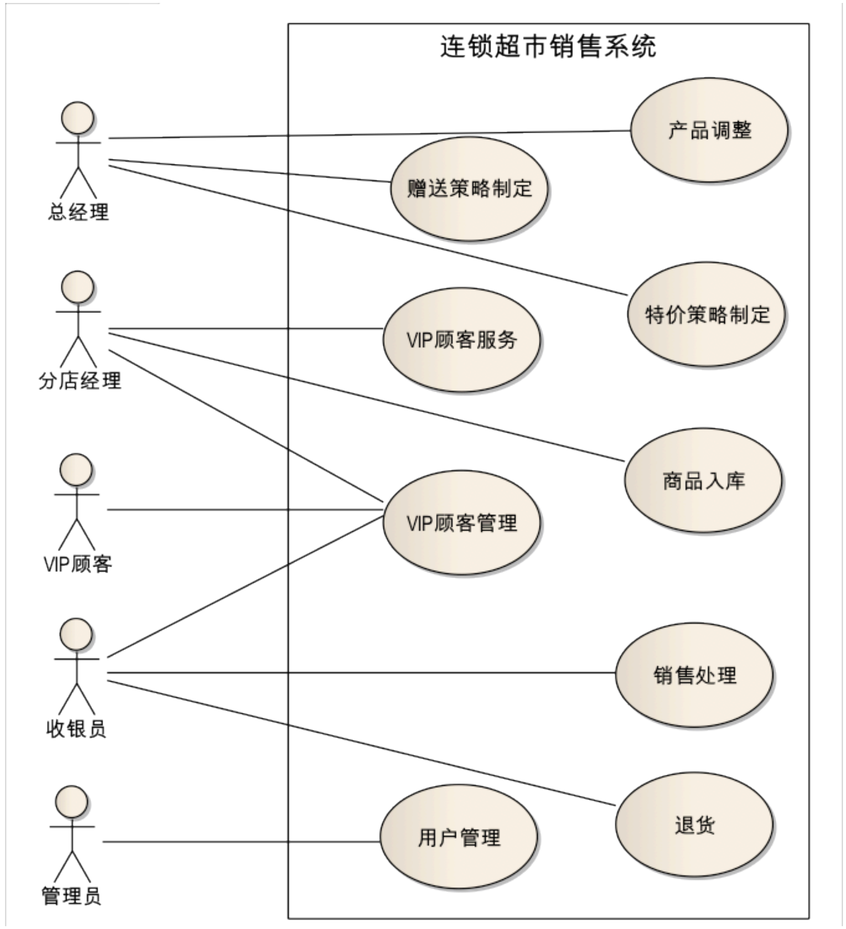
- SF1：分析商品库存，发现可能的商品积压、缺货和报废现象
- SF2：根据市场变化调整销售的商品
- SF3：制定促销手段，处理积压商品
- SF4：与生产厂家联合进行商品促销
- SF5：制定促销手段进行销售竞争
- SF6：掌握员工变动和授权情况
- SF7：处理商品入库与出库
- SF8：发展会员，提高顾客回头率
- SF9：允许积分兑换商品和赠送吸引会员的礼品，提高会员满意度
- SF10：帮助收银员处理销售与退货任务

寻找参与者与用例

- 每个用户的任务（目标）都是一个独立用例

案例中用户的目标

- 总经理的目标有：
  - 产品调整（增删改产品信息）
  - 特价策略制定（增删改特价策略）
  - 赠送策略制定（增删改赠送策略）
  - 库存分析；（分析可能的商品积压）
- 客户经理的目标有：
  - 会员管理；（会员发展、礼品赠送）
  - 库存管理；（商品入库、出库和库存分析）
- 收银员的目标有：
  - 销售处理（销售）
  - 退货；（退货）
- 管理员的目标有：
  - 用户管理（增删改用户信息）


连锁超市销售系统
总经理
分店经理
VIP顾客
收银员
管理员
产品调整
赠送策略制定
特价策略制定
VIP顾客服务
商品入库
VIP顾客管理
销售处理
退货
用户管理

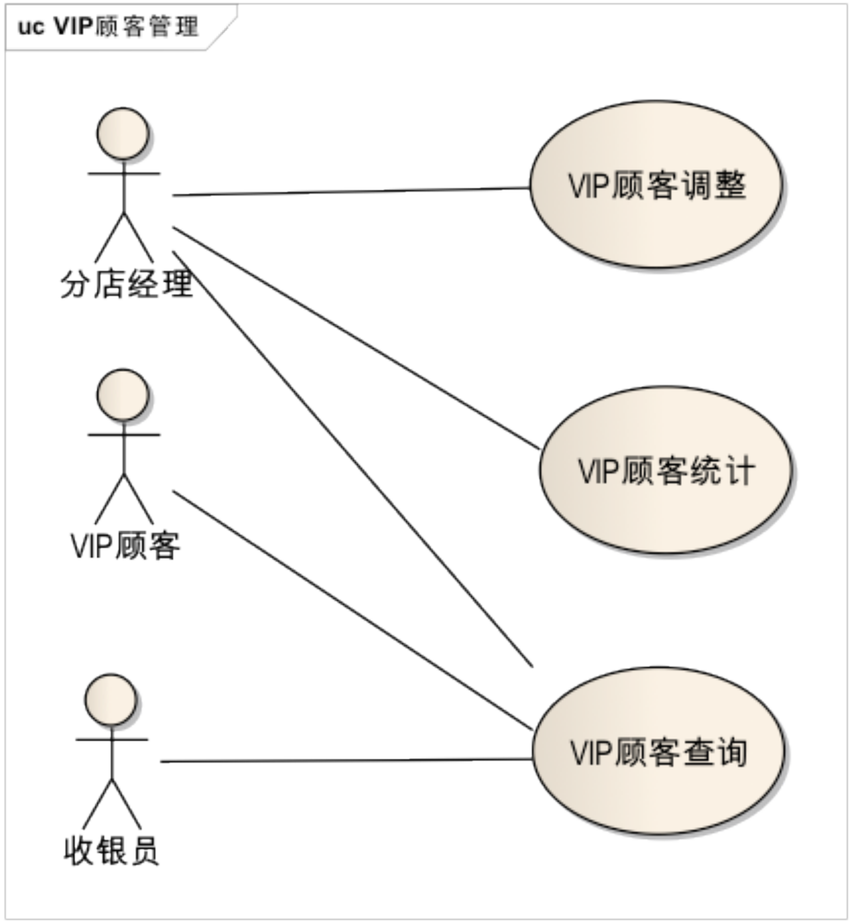
超市销售系统用例

细化用例

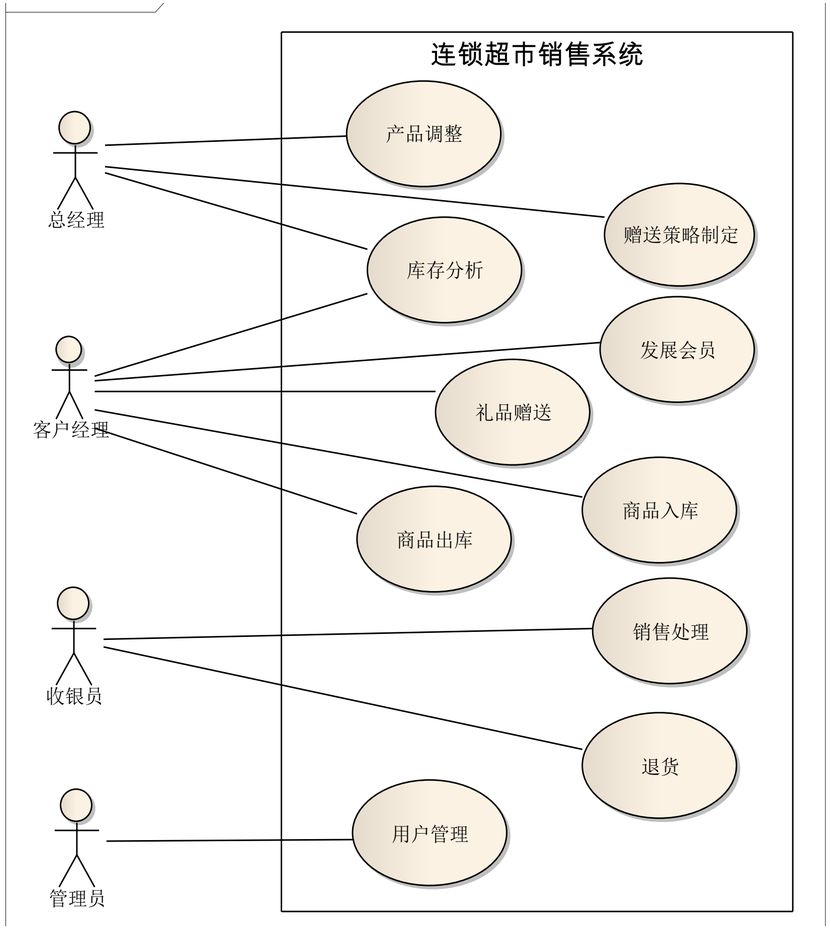
- 如果用例的粒度不合适就需要进行细化和调整。
  - 判断标准是：用例描述了为应对一个业务事件，由一个用户发起，并在一个连续时间段内完成，可以增加业务价值的任务。

细化

- 特价策略制定、赠送策略制定两个用例的业务目的、发起源和过程基本相同，仅仅是业务数据不同，所以可以合并为一个用例销售策略制定。
- 会员管理用例有两个明显不同的业务事件，可以被细化为发展会员和礼品赠送2个更细粒度的用例。
- 客户经理的库存管理用例也有三个不同的业务目标：出库、入库和库存分析，所以也应该细化为三个用例商品出库、商品入库和库存分析，其中库存分析用例与总经理的库存分析用例相同。



用例细化示例


连锁超市销售系统
总经理
客户经理
收银员
管理员
产品调整
赠送策略制定
库存分析
发展会员
礼品赠送
商品入库
商品出库
销售处理
退货
用户管理

细化之后用例图

常见错误

- 不要将用例细化为单个操作
  - 例如,不要将用户管理细化 为增加、修改和删除三个更小的用例,因为它们要联合起来才能体现出业务价值。
- 不要将同一个业务目标细化为不同用例
  - 例如特价策略制定和赠送策略制定。
- 不要将没有业务价值的内容作为用例
  - 常见的错误有“登录”(应该描述为安全性 质量需求)、“数据验证”(应该描述为数据需求)、“连接数据库”(属性软件内部实 现而不是需求)等。

用例模板

项目	内容描述
ID	用例的标识
名称	对用例内容的精确描述，体现了用例所描述的任务
参与者	描述系统的参与者和每个参与者的目标
触发条件	标识启动用例的事件，可能是系统外部的事件，也可能是系统内部的事件，还可能是正常流程的第一个步骤
前置条件	用例能够正常启动和工作的系统状态条件
后置条件	用例执行完成后的系统状态条件
正常流程	在常见和符合预期的条件下，系统与外界的行为交互序列
扩展流程	用例中可能发生的其他场景
特殊需求	和用例相关的其他特殊需求，尤其是非功能性需求

UC1 销售处理	
参与者	收银员，目标是快速、正确地完成商品销售，尤其不要出现支付错误。
触发条件	顾客携带商品到达销售点
前置条件	收银员必须已经被识别和授权。
后置条件	存储销售记录，包括购买记录、商品清单、赠送清单和付款信息；更新库存；打印收据。

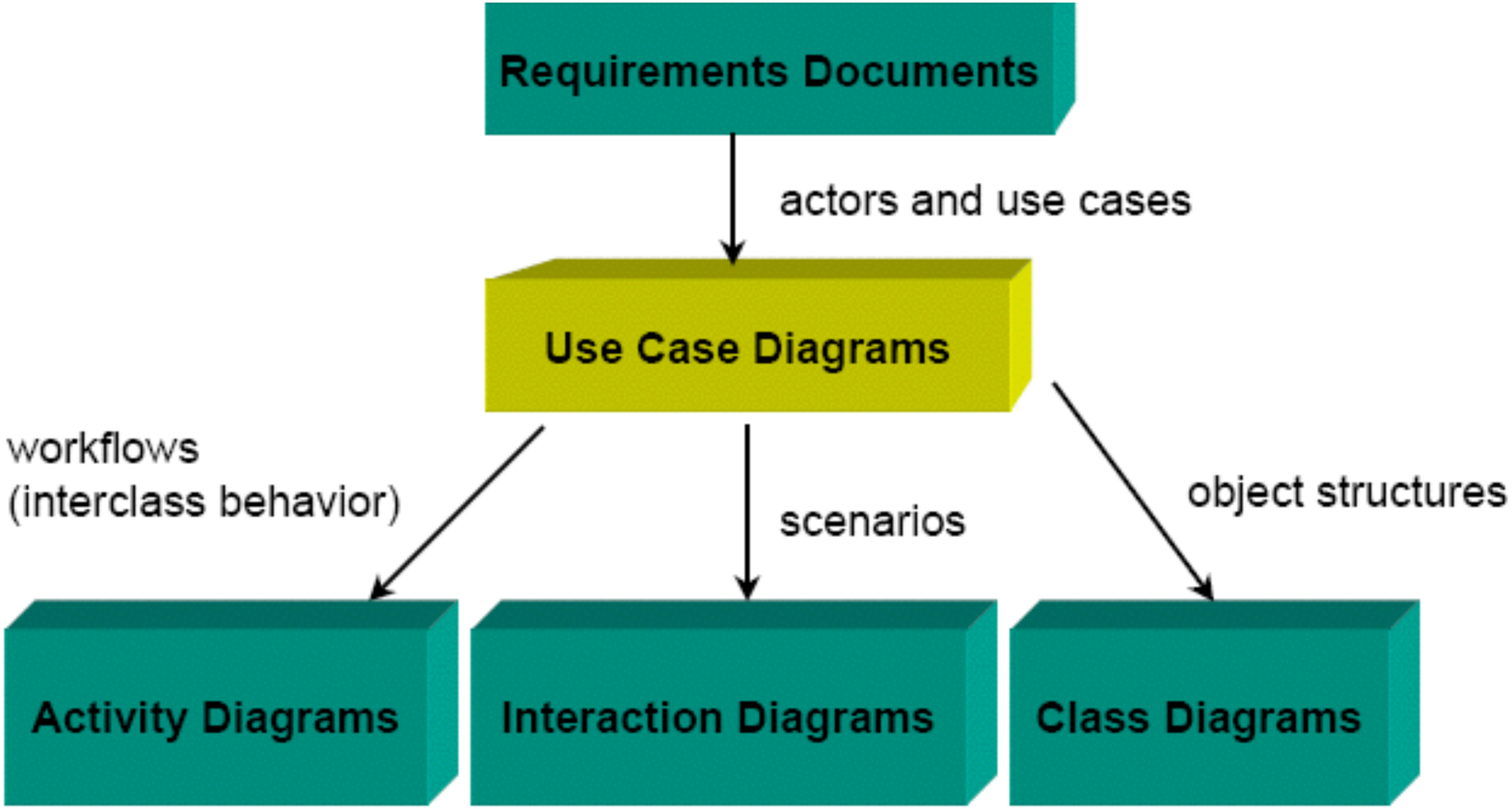
正常流程	1. 收银员开始一次新的销售 2. 如果是*VIP*顾客;收银员输入客户编号 3. 收银员输入商品标识 4. 系统记录商品;并显示商品和赠品信息（如果有商品赠送策略的话）;商品信息包括商品标识、描述、数量、价格、特价（如果有商品特价策略的话）和本项商品总价;赠品信息包括标识、描述与数量。 5. 系统显示已购入的商品清单和赠品清单;商品清单包括商品标识、描述、数量、价格、特价、各项商品总价和所有商品总价;赠品清单包括商品标识、描述和数量。 收银员重复*3-5*步;直到完成所有商品的输入 6. 收银员结束输入;系统计算并显示总价;计算根据总额特价策略进行。 7. 系统根据总额赠送策略补充赠品清单。 8. 收银员请顾客支付账单。 9. 顾客支付;收银员输入收取的现金数额 10. 系统给出应找的余额;收银员找零。 11. 系统记录销售信息、商品清单、赠送清单和账单信息;并更新库存 12. 系统打印收据

扩展流程	*2a*、非法客户编号： 1. 系统提示错误并拒绝输入 *3a*、非法标识： 1. 系统提示错误并拒绝输入 *3b*、有多个具有相同商品类别的商品（如*5*把相同的雨伞） 1. 收银员可以手工输入商品标识和数量 *3-7a*、顾客要求收银员从已输入的商品中去掉一个商品： 1. 收银员输入商品标识并将其删除 2. 系统更新显示的商品列表 3. 系统重新计算总价 4. 系统重新计算赠送商品 *3-7b*、顾客要求收银员取消交易 1. 收银员在系统中取消交易 *6a*、没有可执行的特价策略 1. 系统计算并抵价券，计算根据抵价券策略进行。

扩展流程	*9a*、顾客使用抵价券 1. 营业员输入抵价券编号和余下的现金数额 *9b*、*VIP*顾客使用积分 1. 系统显示可以的积分总额 2. 营业员输入使用的积分数额和余下的现金数额 3. 系统显示剩余的积分总额 *11a*、*VIP*顾客 1. 系统记录销售信息、商品清单、赠送清单和账单信息，并更新库存 2. 计算并更新*VIP*顾客积分和级别 *12a*、系统故障 1. 系统重新启动 2. 收银员重新登录
特殊需求	*1*、系统显示的信息要在*1*米之外能看清 *2*、商品标识为*13*位*0*～*9*的数字 *3*、如果在一个销售任务在第*10*步更新数据过程中发生机器故障，系统的数据要能够恢复到该销售任务之前的状态

关于案例的用例的一些问题

- 顾客为什么不是参与者
- 上传下载为什么不是用例
- 系统可不可以分为服务器和客户端两个系统


Requirements Documents
actors and use cases
Use Case Diagrams
workflows
(interclass behavior)
scenarios
object structures
Activity Diagrams
Interaction Diagrams
Class Diagrams

UML中用例图的作用

概念类图（Conceptual Class Diagram）

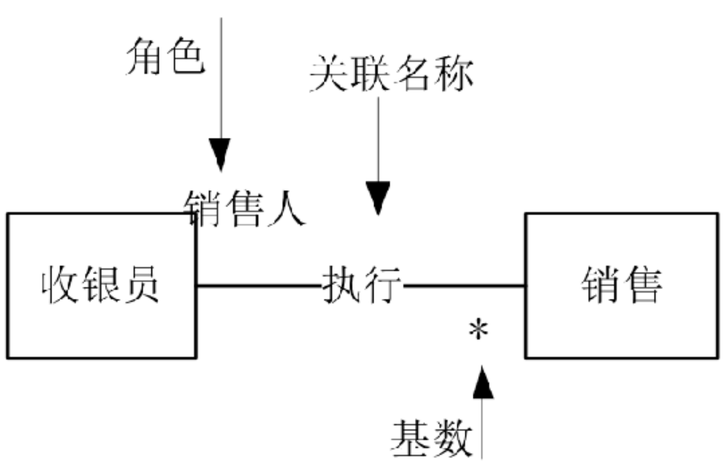
- 又被称为“领域模型”(Domain Model)
- 类图是面向对象分析方法的核心
  - 描述类（对象）和这些类（对象）之间的关系
- 与设计类图有所不同
  - 关注系统与外界的交互，而不是软件系统的内部构造机制
- 类型、方法、可见性等复杂的软件构造细节不会在概念类图中

基本元素

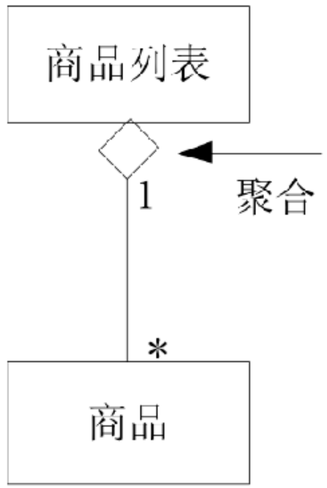
- 对象
  - 标示符
  - 状态
  - 行为
- 类
  - 对象集合的抽象
- 链接（link）（dependency）
  - 对象之间的互相协作的关系
  - 描述了对象之间的物理或业务联系
- 关联
  - 对象之间链接的抽象
  - 聚合与组合
- 继承
  - 泛化关系

关联与依赖

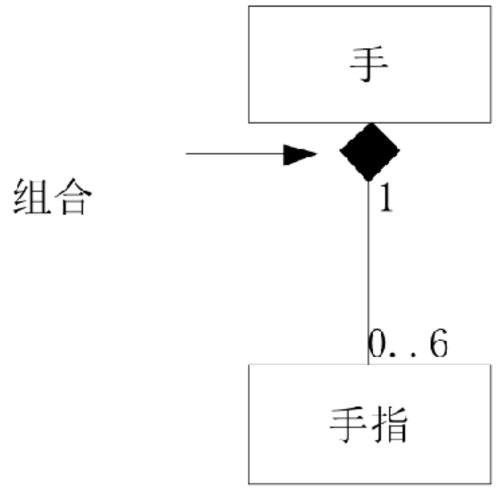
- Two analysis classes are often related to one another in some fashion
  - In UML these relationships are called associations
  - Associations can be refined by indicating multiplicity (the term cardinality is used in data modeling
- If there are associations between classes, then there are links （dependencies） between instances of the classes



(a)



(b)

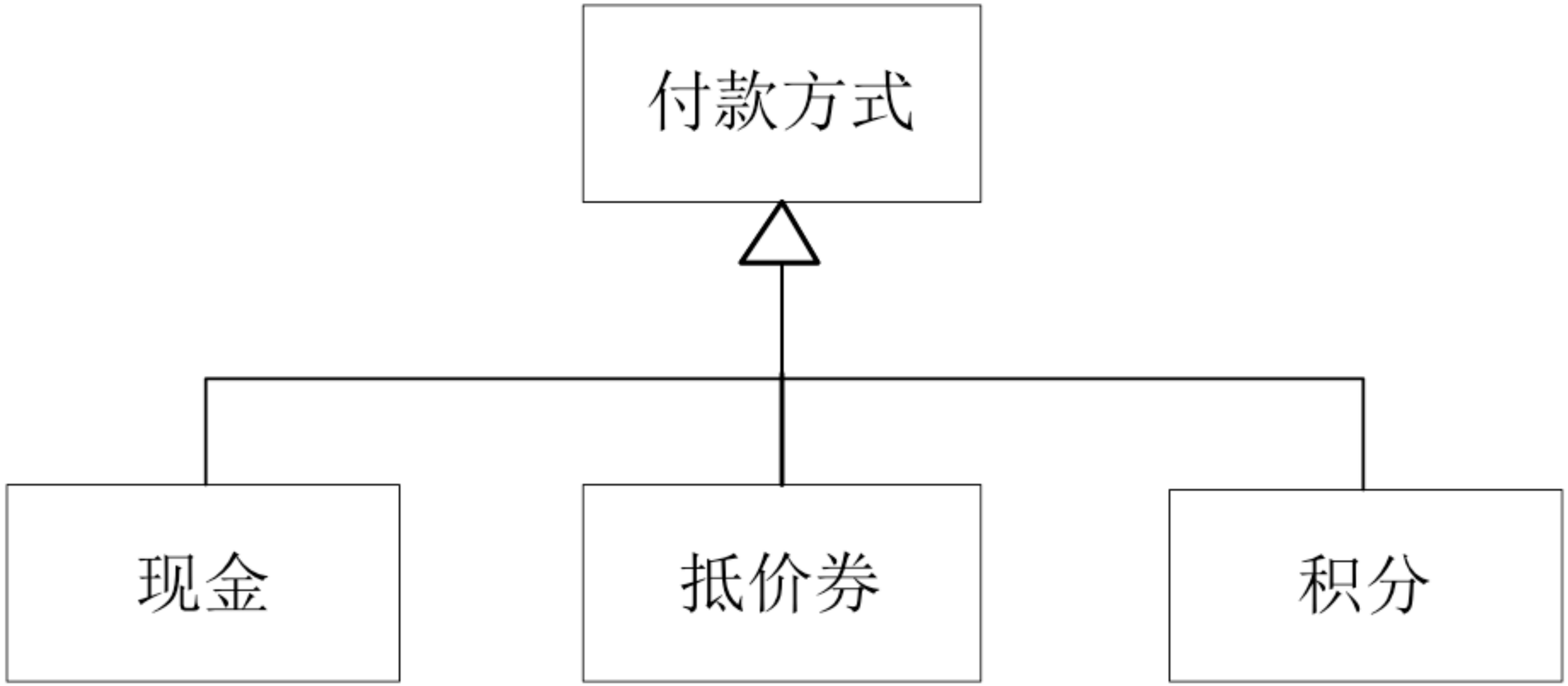


(c)

案例

继承

- Organise the domain object classes into a hierarchy
- Classes at the top of the hierarchy reflect the common features of all classes
- Object classes inherit their attributes and services from one or more super-classes. these may then be specialised as necessary



案例

建立概念类图

- 对每个用例文本描述，尤其是场景描述，建立局部的概念类图
  - 根据用例的文本描述，识别候选类
  - 筛选候选类，确定概念类
  - 识别关联
  - 识别重要属性
- 将所有用例产生的局部概念类图进行合并，建立软件系统的整体概念类图

候选类识别

- 发现软件系统与外界交互时可能涉及的对象与类，它们就是候选类。
- 行为分析、名词分析、CRC等很多种方法都可以用来分析用例文本描述

用例描述（部分）：

1、收银员开始一次新的销售
2、如果是VIP顾客;收银员输入客户编号
3、收银员输入商品标识
4、系统记录商品;并显示商品和赠品信息（如果有商品赠送策略的话）;商品信息包括商品标识、描述、数量、价格、特价（如果有商品特价策略的话）和本项商品总价;赠品信息包括标识、描述与数量。
5、系统显示已购入的商品清单和赠品清单;商品清单包括商品标识、描述、数量、价格、特价、各项商品总价和所有商品总价;赠品清单包括商品标识、描述和数量。
收银员重复3-5步;直到完成所有商品的输入
6、收银员结束输入;系统计算并显示总价;计算根据总额特价策略进行。
7、系统根据总额赠送策略补充赠品清单。
8、收银员请顾客支付账单。
9、顾客支付;收银员输入收取的现金数额
10、系统给出应找的余额;收银员找零。
11、系统记录销售信息、商品清单、赠送清单和账单信息;并更新库存
12、系统打印收据

名词分析

确定概念类的准则

- 依据系统的需求
- 该类的对象实例的状态与行为是否完全必要

候选类向概念类的转化

- 如果候选类的对象实例
  - 既需要维持一定的状态，又需要依据状态表现一定的行为
    - 确定为一个概念类
  - 如只需要维护状态，不需要表现行为
    - 其他概念类的属性
  - 不需要维护状态，却需要表现行为
    - 首先要重新审视需求是否有遗漏，因为没有状态支持的对象无法表现行为
    - 如果确定没有需求的遗漏，就需要剔除该候选类，并将行为转交给具备状态支持能力的其他概念类
  - 既不需要维护状态，又不需要表现行为
    - 应该被完全剔除

概念类案例

- 顾客（无直接关联，无状态行为，应舍弃）
- 会员
- 销售
- 收据（无状态，应舍弃）
- 商品标识（有状态，无行为，是属性）

候选类：	被作为属性的候选类： 姓名、积分→会员 客户编号→ 销售信息、会员 商品的标识、描述、数量、价格、特价、本项商品总价→商品清单 总价；现金数额；余额→ 账单 赠品的标识、描述、数量→ 赠品 被剔除的候选类： 收据;顾客 词语加工： 会员;会员信息→会员 各项商品总价;本项商品总价→总价 账单;账单信息→账单 所有商品总价;总价→总价 现金数额→现金付款 余额→找零	概念类： 收银员 销售信息（客户编号） 会员（客户编号、姓名、积分） 商品清单（标识、描述、数量、价格、特价、总价） 商品特价策略 总额特价策略 赠品清单（标识、描述、数量） 商品赠送策略 总额赠送策略 账单（总价、支付额、找零） 库存
会员;收银员;会员信息;姓名;积分;客户编号;商品标识;商品信息;商品的描述、数量、价格、特价和本项商品总价;商品特价策略；商品清单;各项商品总价；所有商品总价;总价;总额特价策略;商品赠送策略;总额赠送策略;赠品清单;赠品的标识、描述、数量；顾客;账单;现金数额；余额；销售信息；账单信息；库存；收据		

生成概念类

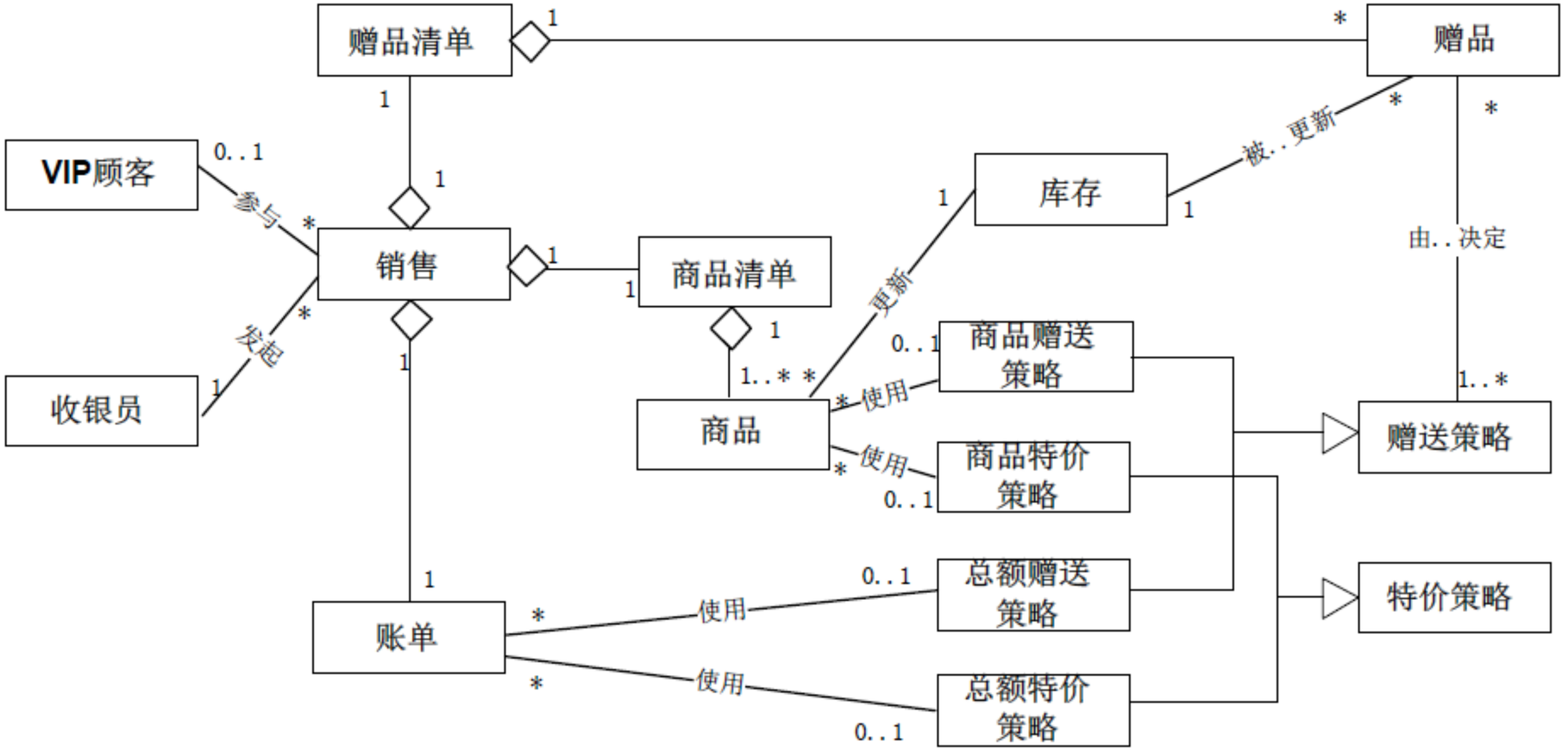
识别关联

- 分析用例文本描述，发现概念类之间的协作
  - 需要协作的类之间需要建立关联。
- 分析和补充问题域内的关系
  - 例如概念类之间的整体部分关系和明显的语义联系。
  - 对问题域关系的补充要适可而止，不要把关系搞得过度复杂化。
- 去除冗余关联和导出关联。

协作：
1、商品 根据 商品特价策略 确定特价
2、商品 根据 商品赠送策略 确定赠品
3、总额特价策略 和 顾客特价策略 用于计算 总价（账单）
4、总额赠送策略 根据总额（账单） 计算赠品
5、库存 去除 商品和 赠品

补充的问题域关系：
1、商品清单 包含 商品
2、赠品清单 包含 赠品
3、销售信息包括收银员工号（收银员）和客户编号（VIP顾客）
4、销售 包括 简要销售信息 和 商品清单、赠品清单、账单

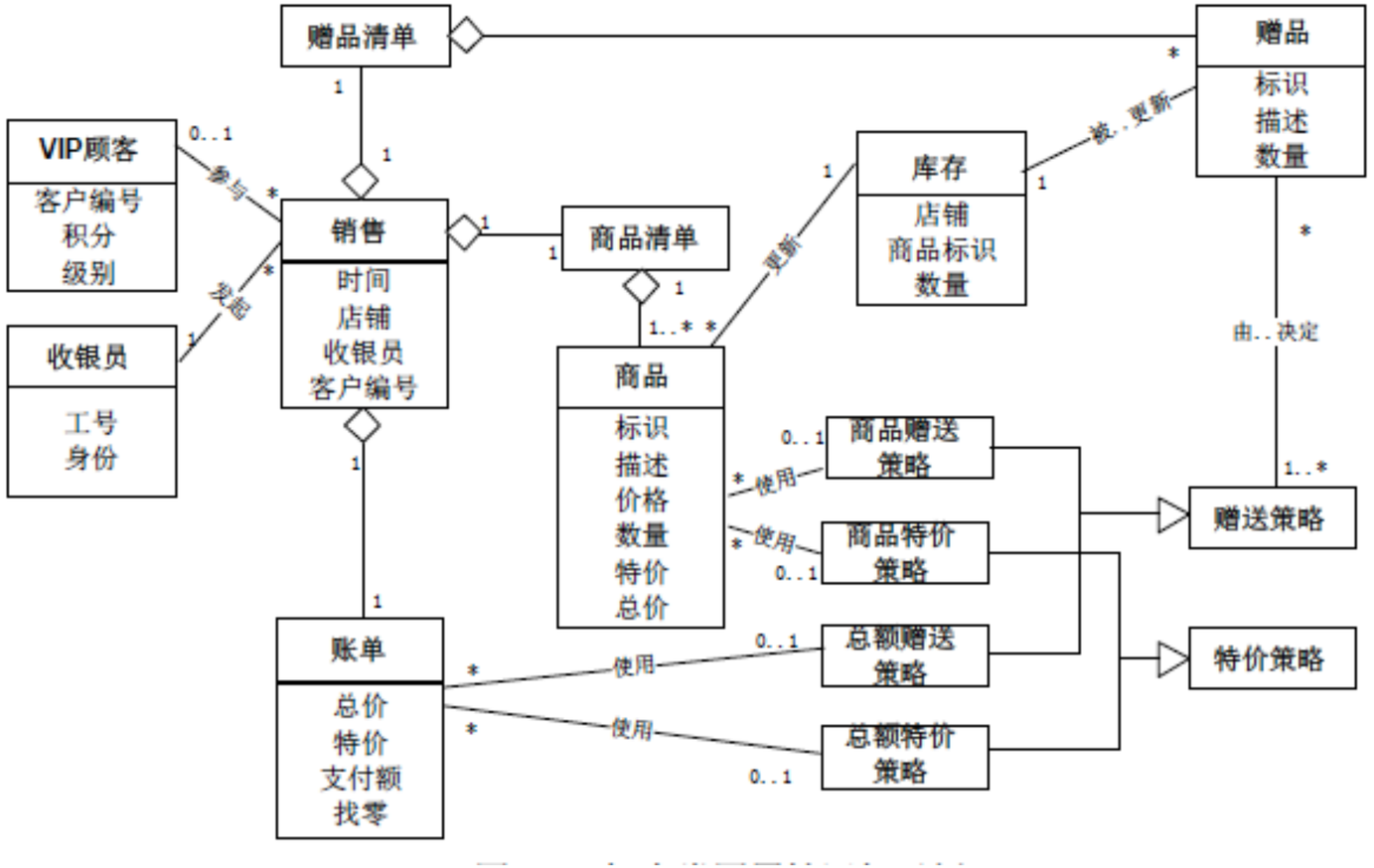
关联分析


赠品清单
赠品
VIP顾客
参与
销售
商品清单
库存
被..更新
由..决定
收银员
发起
更新
商品
商品赠送策略
使用
商品特价策略
赠送策略
总额赠送策略
特价策略
账单
总额特价策略
0..1
1..*
1
*

案例

识别重要属性

- 这些属性往往是实现类协作时必要的信息，是协作的条件、输入、结果或者过程记录。
- 通过分析用例的描述，并与用户交流，补充问题域信息，可以发现重要的属性信息。
- 在分析每个单独的用例（场景）描述时，为各个概念类发现的重要属性可能不多，甚至有些概念类没有任何重要属性。但是，系统通常有多个用例和很多场景，会建立多个局部的概念类图，只有在合并所有局部概念类图之后，各个概念类的重要属性才能得到全面的体现。


赠品清单
VIP顾客
客户编号
积分
级别
参与
0..1
*
销售
时间
店铺
收银员
客户编号
1
收银员
工号
身份
考勤
商品清单
库存
店铺
商品标识
数量
更新
赠品
标识
描述
数量
由..决定
商品
标识
描述
价格
数量
特价
总价
1..*
商品赠送策略
使用
商品特价策略
总额赠送策略
总额特价策略
赠送策略
特价策略
账单
总价
特价
支付额
找零

案例

习题

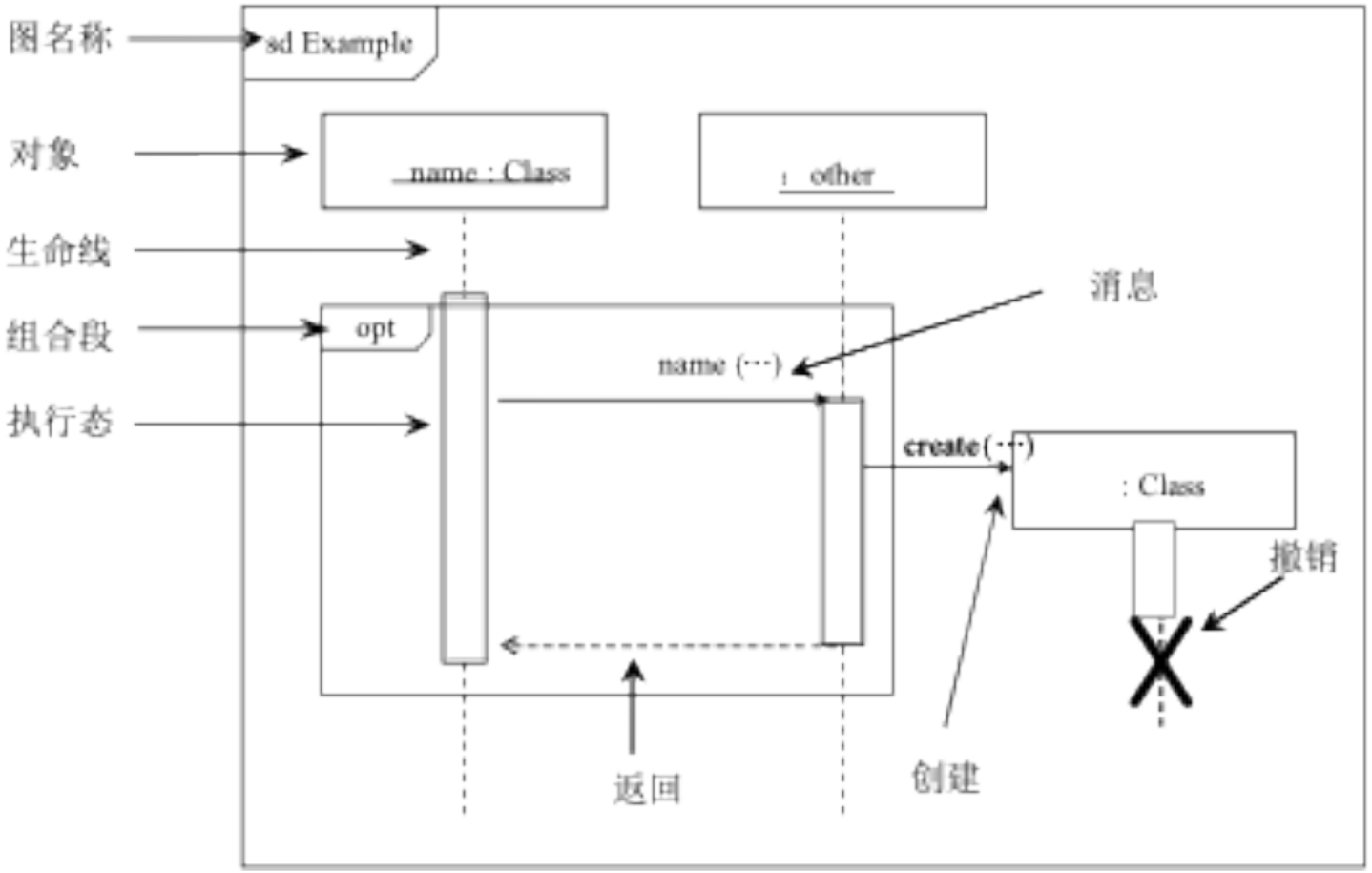
- University Bank will be opening in Oxford, Mississippi, in January, 2000. We plan to use a full service automated teller machine (ATM) system.The ATM system will interact with the customer through a display screen, numeric and special input keys, a bankcard reader, a deposit slot, and a receipt printer.Customers may make deposits, withdrawals, and balance inquires using the ATM machine, but the update to accounts will be handled through an interface to the Accounts system.Customers will be assigned a Personal Identification Number (PIN) and clearance level by the Security system. The PIN can be verified prior to any transaction.In the future, we would also like to support routine operations such as a change of address or phone number using the ATM

顺序图

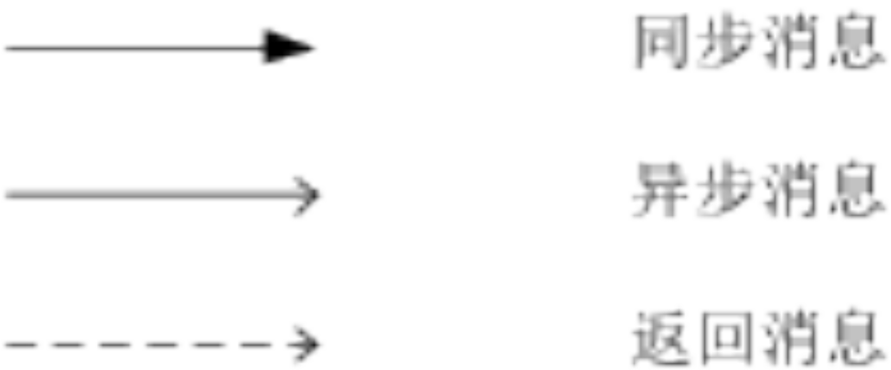
- A behavioral model shows the interactions between objects to produce some particular system behavior that is specified as a use-case

- Sequence diagrams (or collaboration diagrams) in the UML are used to model interaction between objects

- 分析阶段，主要是利用系统顺序图，表达系统和外部参与者之间的交互行为。


图名称
sd Example
对象
name : Class
: other
生命线
消息
组合段
opt
name (···)
执行态
create (···)
: Class
撤销
返回
创建

一个简单示例



消息种类

系统顺序图

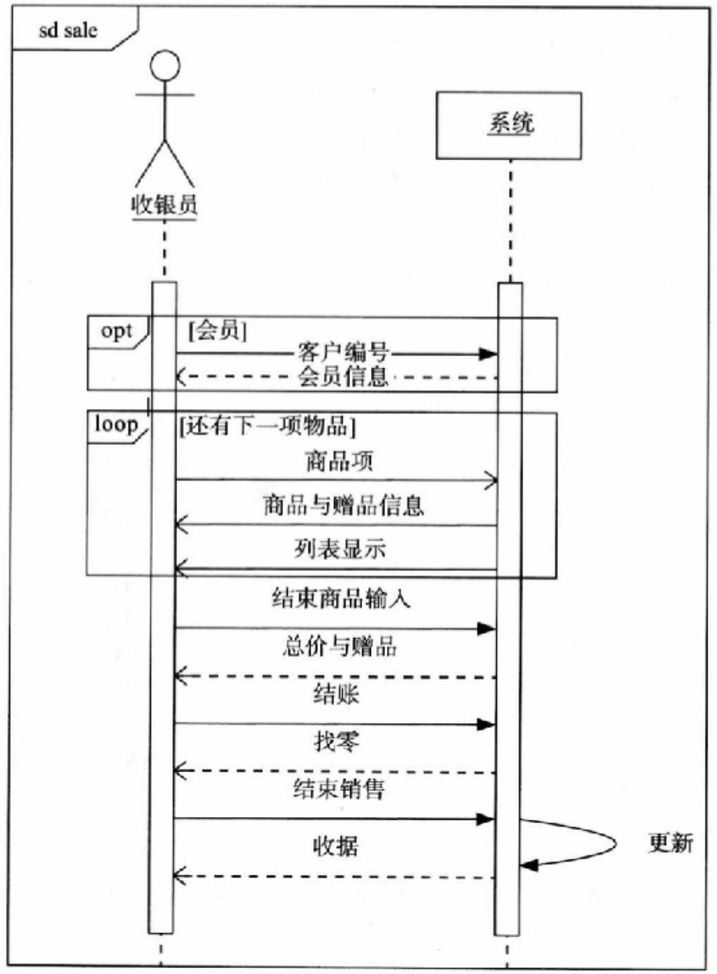
同步消息是实线，三角箭头。

返回消息应该是虚线

Opt 是可选项

loop 是循环

alt 多选一



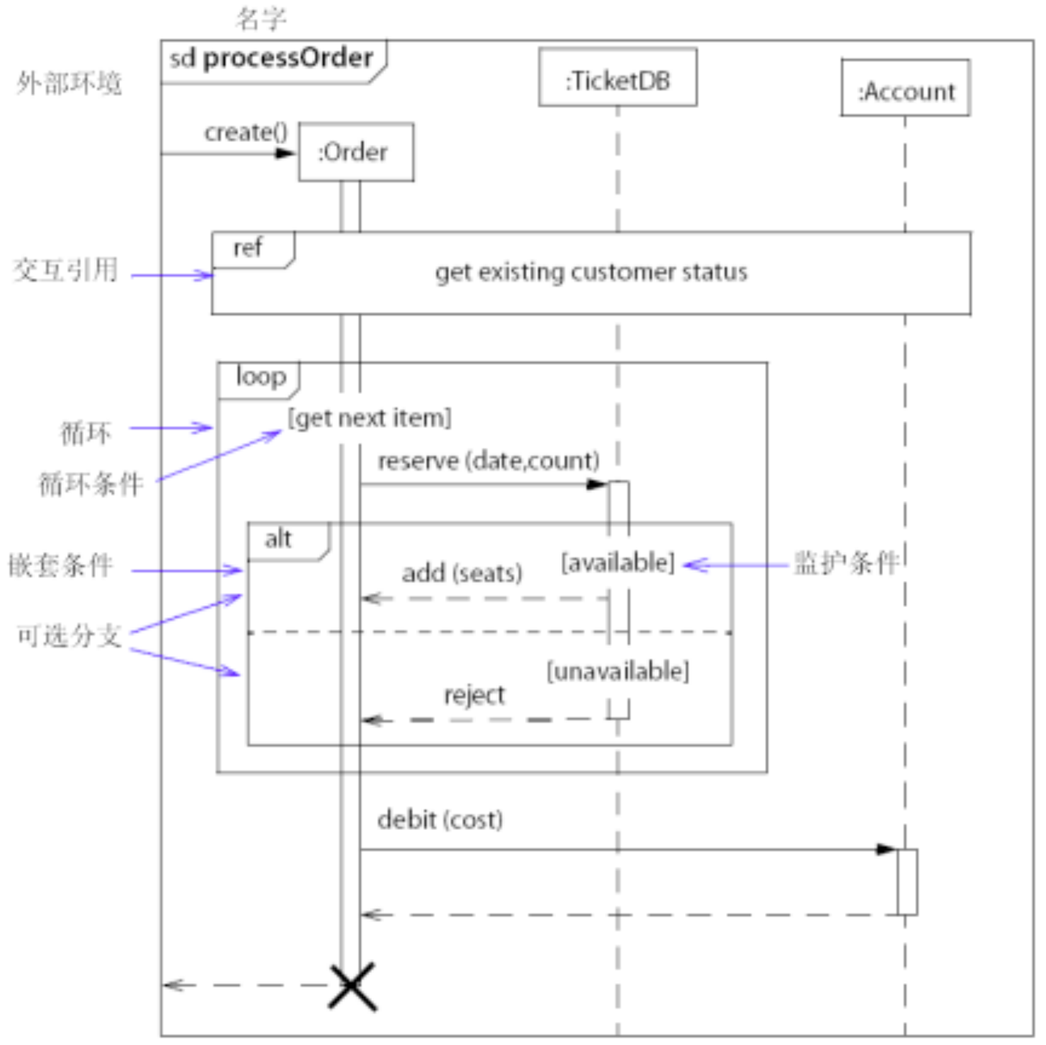


图 14－20、特化图示示例，源自[Rumbaugh2004]

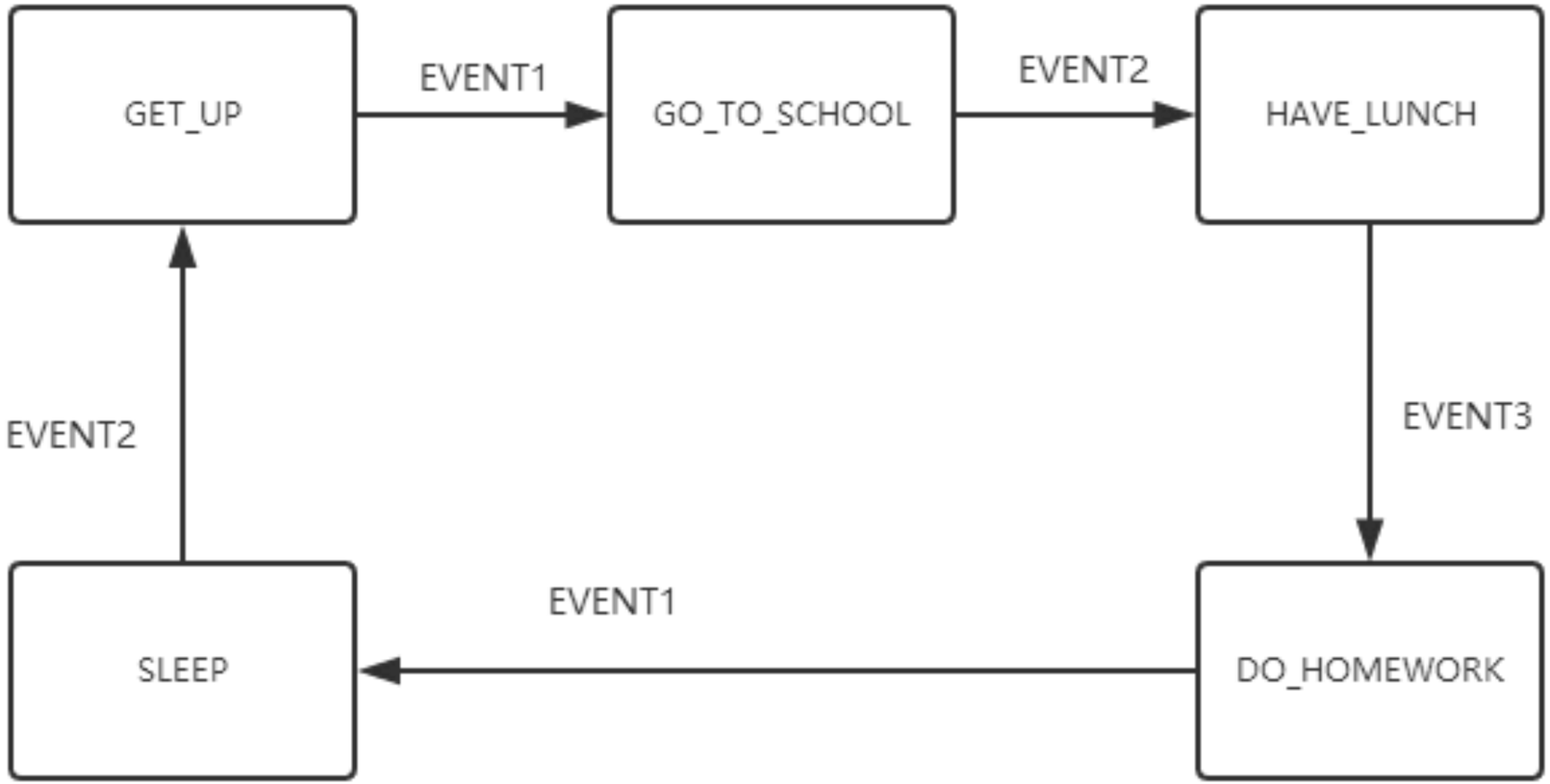
状态图

状态图

- How do I model the software's reaction to some external event?

小明的一天 - Switch

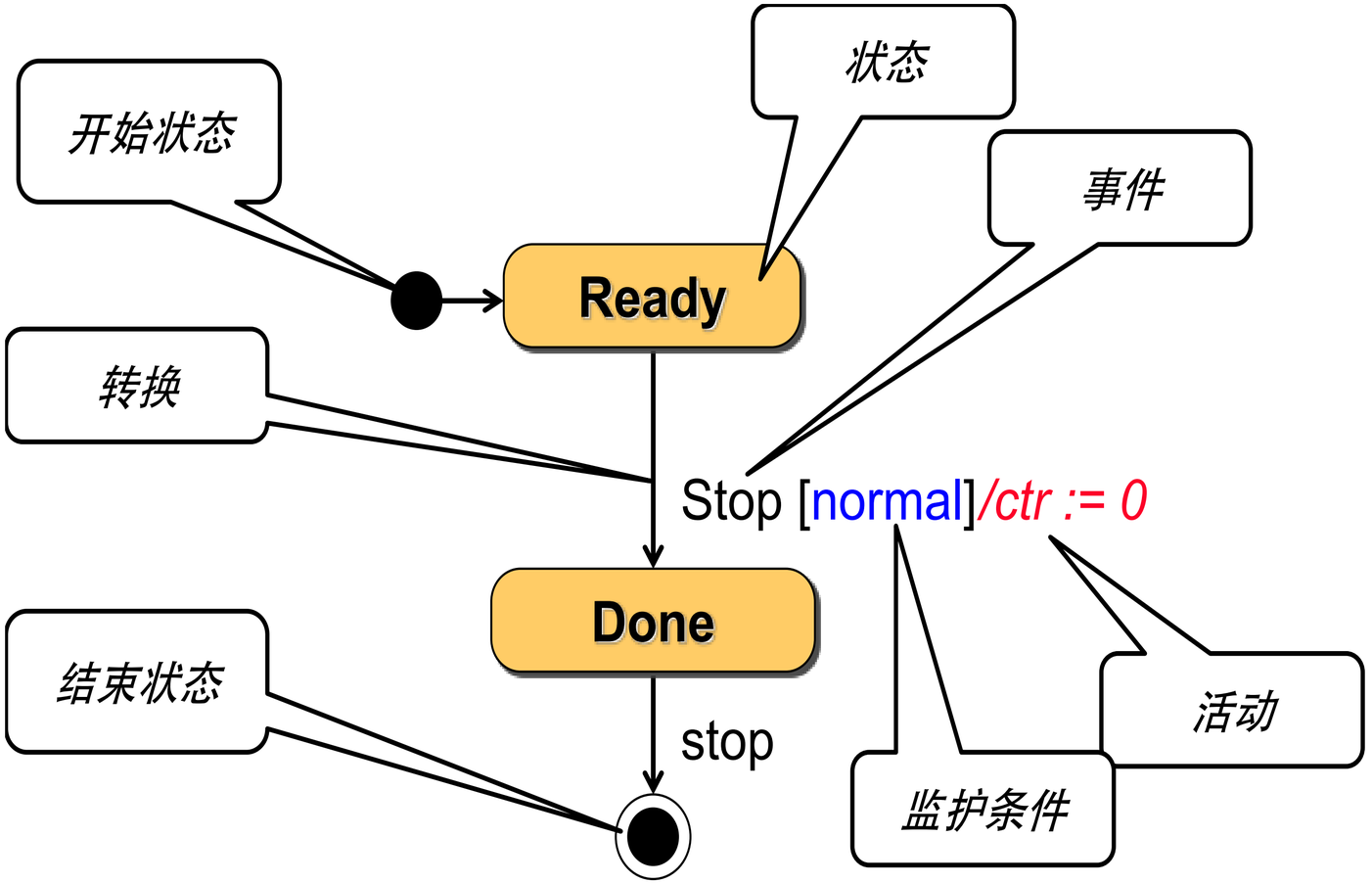
```
int main()
{
    int state = GET_UP;
    //小明的一天
    while (1)
    {

        switch(state)
        {
        case GET_UP:
            GetUp(); //具体调用的函数
            state = GO_TO_SCHOOL;  //状态的转移
            break;
        case GO_TO_SCHOOL:
            Go2School();
            state = HAVE_LUNCH;
            break;
        case HAVE_LUNCH:
            HaveLunch();
            state = GO_HOME;
            break;
            ...
        default:
            break;
        }
    }

    return 0;
}
```



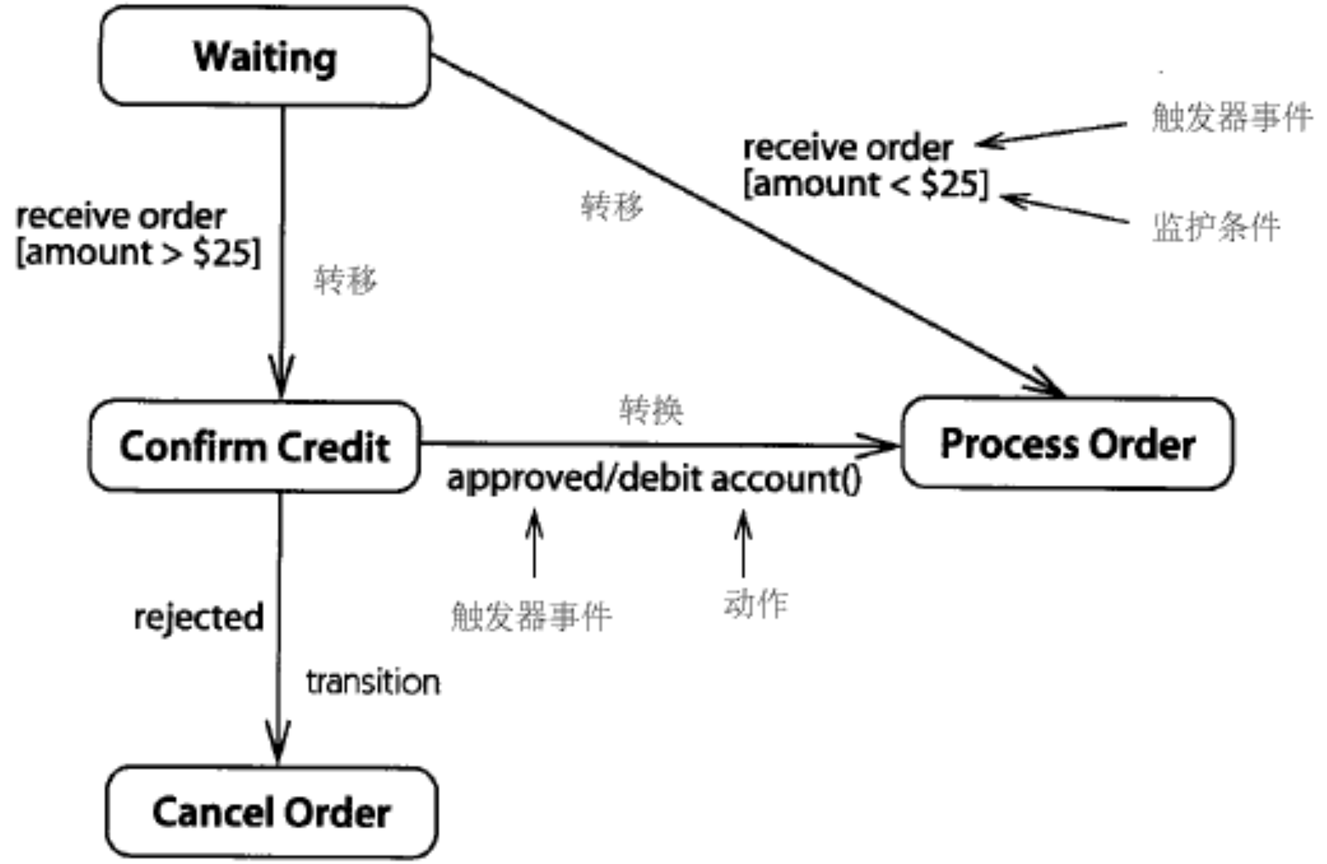
The behavioral model indicates how software will respond to external events or stimuli.

Behavioral Diagram
—— State Diagram Basic concept

- State:
  - a set of observable circumstances that characterizes the behavior of a system at a given time
- State transition:
  - the movement from one state to another
- Event:
  - an occurrence that causes the system to exhibit some predictable form of behavior
- Action:
  - process that occurs as a consequence of making a transition



示例


Waiting
receive order
[amount > $25]
转移
receive order
[amount < $25]
触发器事件
监护条件
转移
Confirm Credit
转换
Process Order
approved/debit account()
触发器事件
动作
rejected
transition
Cancel Order

案例

建立状态图

- 确定上下文环境
  - 状态图是立足于状态快照进行行为描述的，因此建立状态图时首先要搞清楚状态的主体，确定状态的上下文环境。常见的状态主体有：类、用例、多个用例和整个系统。
- 识别状态
  - 状态主体会表现出一些稳定的状态，它们需要被识别出来，并且标记出其中的初始状态和结束状态集。在有些情况下，可能会不存在确定的初始状态和结束状态。
- 建立状态转换
  - 根据需求所描述的系统行为，建立各个稳定状态之间可能存在的转换。
- 补充详细信息，完善状态图
  - 添加转换的触发事件、转换行为和监护条件等详细信息。

销售处理用例状态图

- 明确状态图的主体：用例UC1销售处理。
- 识别用例UC1销售处理可能存在的稳定状态：
  - 空闲状态（开始状态）：收银员已经登录和获得授权，但并没有请求开始销售工作的状态；
  - 销售开始状态：开始一个新销售事务，系统开始执行一个销售任务的状态；
  - VIP顾客信息显示状态：输入了客户编号，系统显示该VIP顾客信息的状态；
  - 商品信息显示状态：刚刚输入了一个物品项，显示该物品（和赠品）描述信息的状态；
  - 列表显示状态：以列表方式显示所有已输入物品项（和赠品）信息的状态；
  - 错误提示状态：输入信息错误的状态；
  - 账单处理状态：输入结束，系统显示账单信息，收银员进行结帐处理的状态。
  - 销售结束状态：更新信息，打印收据的状态。

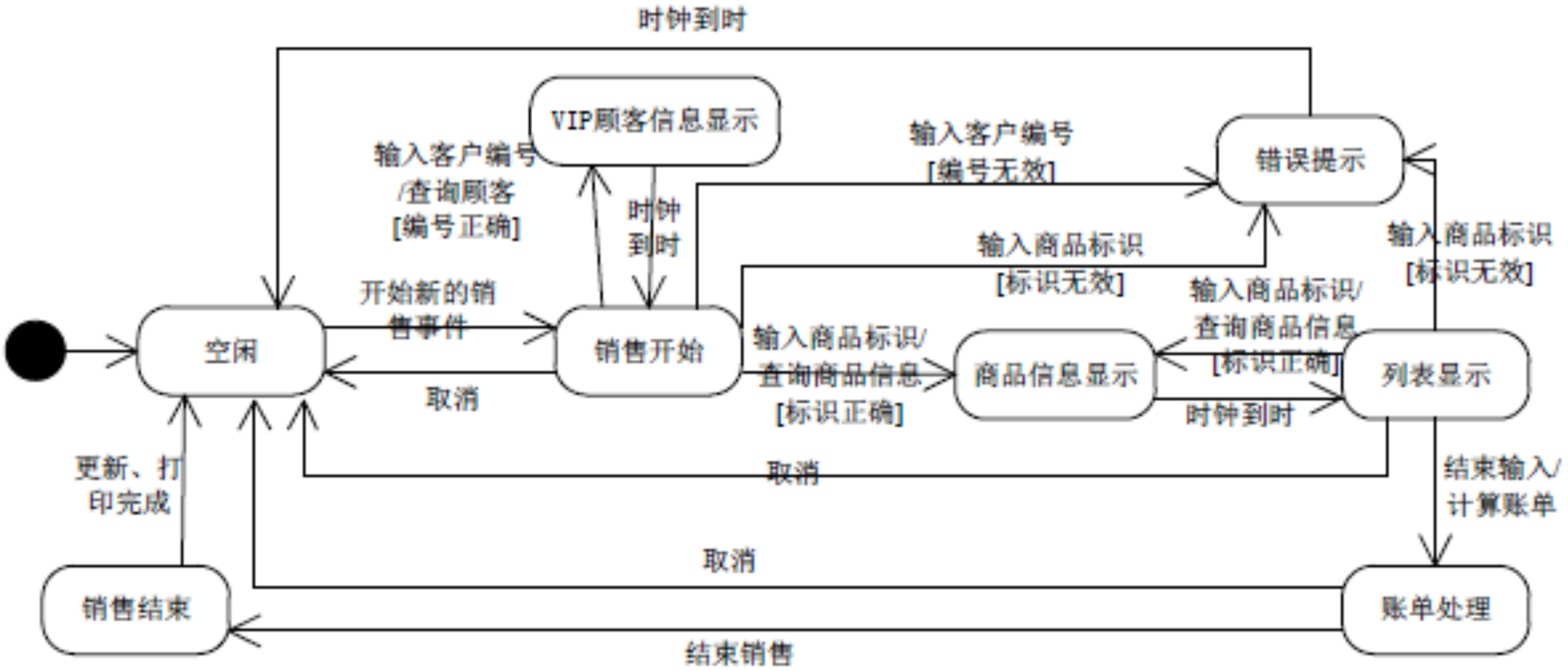
(3) 建立状态转换。可能的状态转换如表 8-4 所示，其中如果第 i 行第 j 列的元素被标记为 Y，则表示第 i 行的状态可以转换为第 j 列的状态。

表 8-4 状态转换表建立示例

	空闲	销售开始	VIP 顾客信息显示	商品信息显示	错误提示	列表显示	账单处理	销售结束
空闲		Y						
销售开始	Y		Y	Y	Y			
VIP 顾客信息显示		Y						
商品信息显示						Y		
错误提示	Y							
列表显示	Y			Y	Y		Y	
账单处理	Y							Y
销售结束	Y							

表 14—6、建立状态转换示例

建立状态转换示例


时钟到时
VIP顾客信息显示
输入客户编号
/查询顾客
[编号正确]
时钟
到时
输入客户编号
[编号无效]
错误提示
输入商品标识
[标识无效]
输入商品标识
[标识无效]
开始新的销
售事件
空闲
销售开始
输入商品标识/
查询商品信息
[标识正确]
商品信息显示
输入商品标识/
查询商品信息
[标识正确]
列表显示
取消
时钟到时
取消
更新、打
印完成
结束输入/
计算账单
取消
销售结束
账单处理
结束销售

8-30 状态图示例

案例

习题

North

West

East

South

You are designing a traffic light system for this intersection.

Draw a state diagram showing the different states and how they transition.

练习—ATM机

- 业务需求
- 用户需求
- 系统需求
- 概念类图
- 系统顺序图
- 状态图

Outline

- 需求分析基础
- 面向对象分析
- 结构化方法
  - 上下文图
  - 数据流图
  - 实体关系图

结构化方法的历史

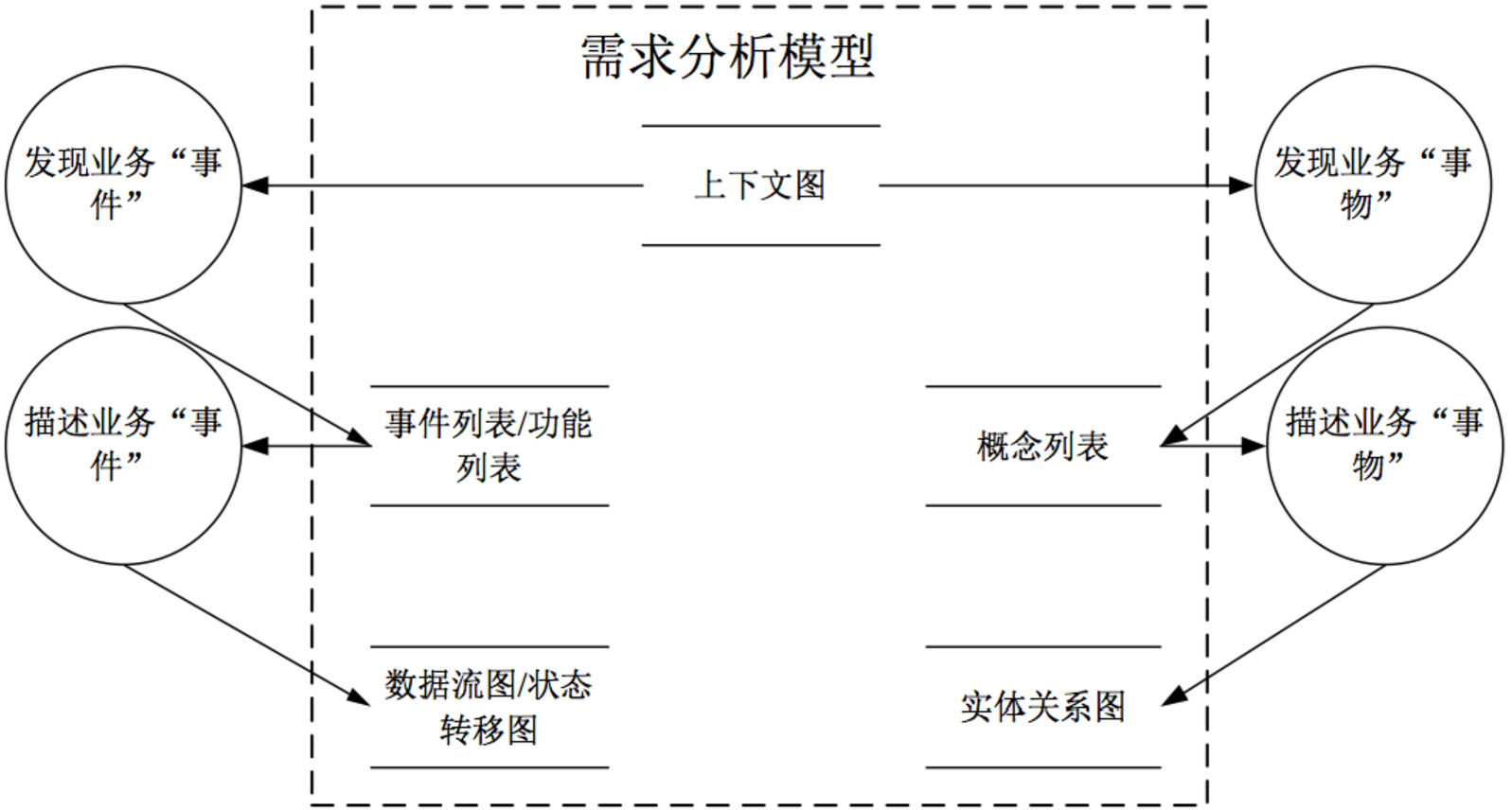
- 结构化方法是针对1960’s到1980’s软件开发界所面临的问题提出一系列分析、设计和编码的技术方法。那个时代：
  - Most commercial programming was done in Cobol and Fortran, then C and BASIC
  - There was little guidance on “good” design and programming techniques
  - There were no standard techniques for documenting requirements and designs
- 关键是软件的复杂度的急剧上升

Multiple Structured Methods emerged

- Structured Programming
  - in circa 1967 with Edsger W.Dijkstra
- Structured Design
  - around 1975 with Larry Constantine and Ed Yourdon
- Structured Analysis
  - in circa 1978 with Tom DeMarco, Yourdon, Gane & Sarson, McMenamin & Palmer
- Information Engineering
  - in circa 1990 with James Martin

结构化分析（Structured Analysis）思想

- 自顶向下分解
- 图
  - 数据流图
  - 实体关系图
  - 状态转移图


需求分析模型
上下文图
发现业务“事件”
发现业务“事物”
描述业务“事件”
描述业务“事物”
事件列表/功能列表
概念列表
数据流图/状态转移图
实体关系图

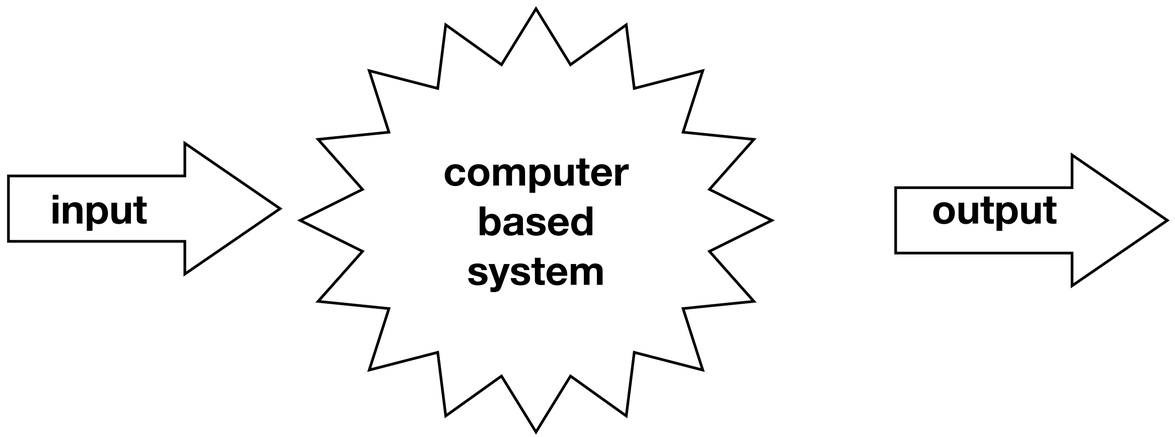
结构化分析的简单过程

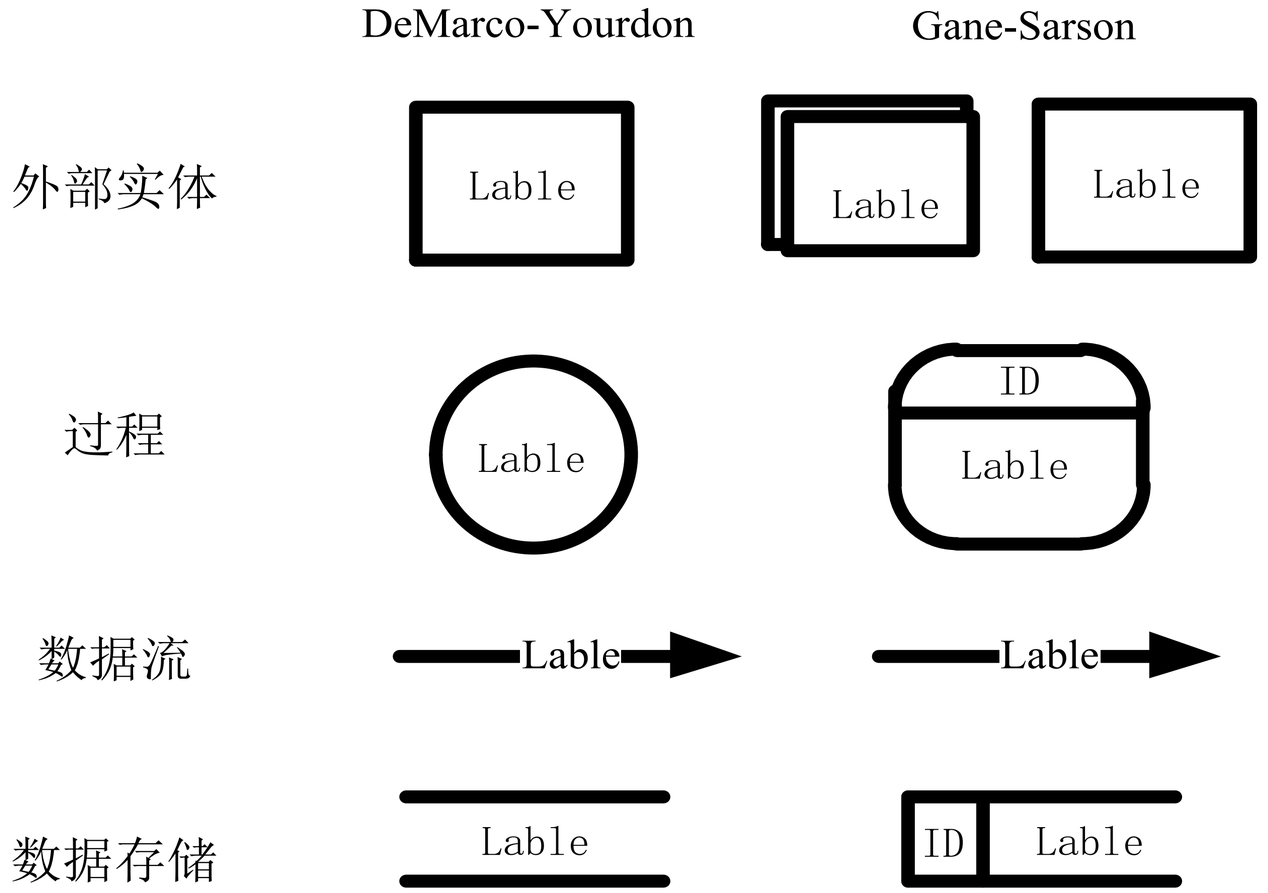
Data Flow Diagram
——Flow Oriented Model

- 将系统看做是过程的集合；
- 过程就是对数据的处理：
  - 接收输入，进行数据转换，输出结果
- Represents how data objects are transformed at they move through the system
- 可能需要和软件系统外的实体尤其是人进行交互
- 数据的变化包括：
  - 被转换、被存储、或者被分布

Data Flow Diagram
—— The Flow Model

- Every computer-based system is an information transform ....



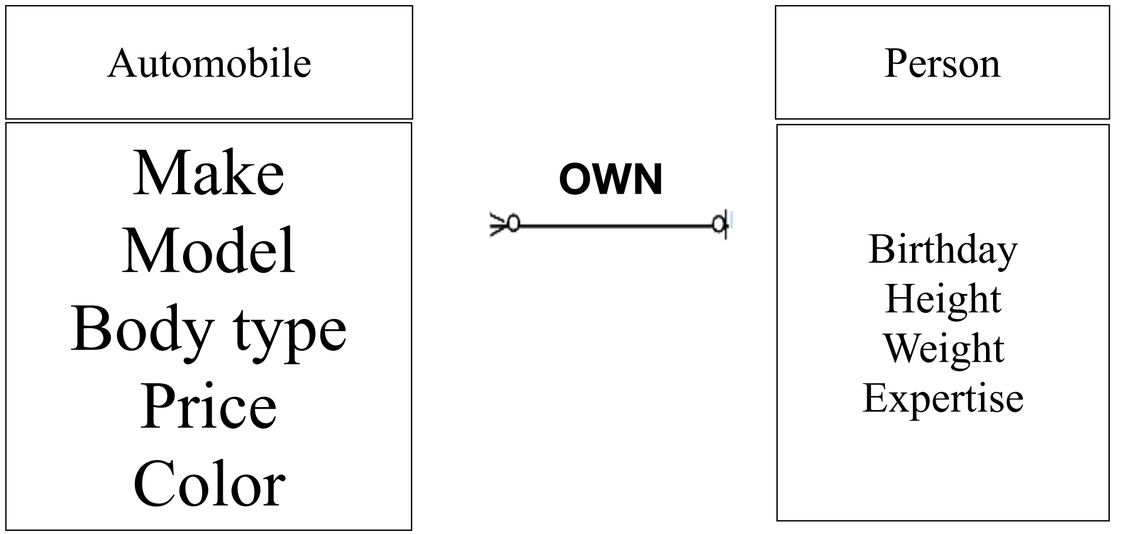


Data Flow Diagram

—— Flow Modeling Notation（基本元素）

实体关系图（ERD）-- 数据的建模

- Examines data objects independently of processing
- Focuses attention on the data domain
- Indicates how data objects relate to one another
-



What is a Data Object?

- Object -- something that is described by a set of attributes(data items) and that will be manipulated within the software (system)
  - each instance of an object(e.g. a book) can be identified uniquely(e.g. ISBN#)
  - each plays a necessary role in the system i.e., the system could not function without access to instances of the object
  - each is described by attributes that are themselves data items

Typical Objects

- external entities
  - printer, user, sensor
- things
  - reports, displays, signals
- occurrences or events
  - interrupt, alarm
- roles
  - manager, engineer, salesperson
- organizational units
  - devision, team
- places
  - manufacturing floor
- structures
  - employee record

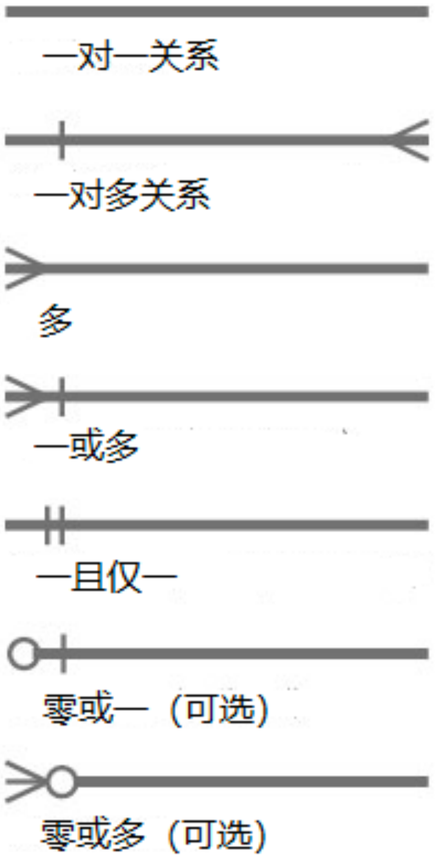
Data Objects and Attributes

- A data object contains a set of attributes that act as an aspect, quality, characteristic, or descriptor of the object

Relationship

- Connectedness
  - A fact that must be remembered by the system and cannot or is not computed or derived
    - Several instance of a relationship can exist
    - Entity can be related in many ways

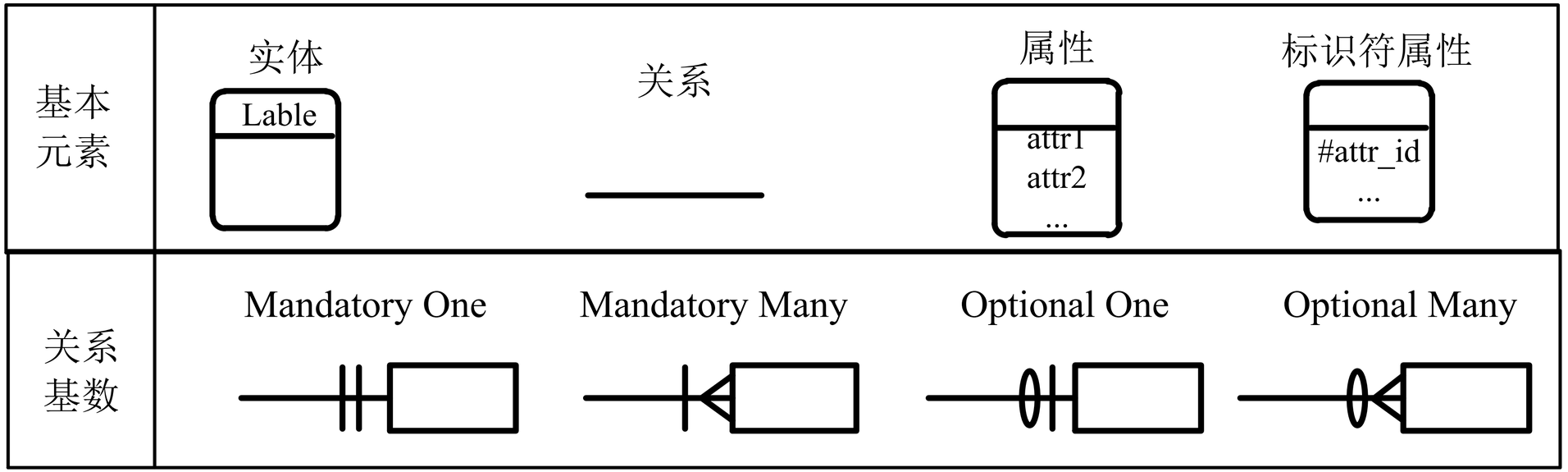
OWN



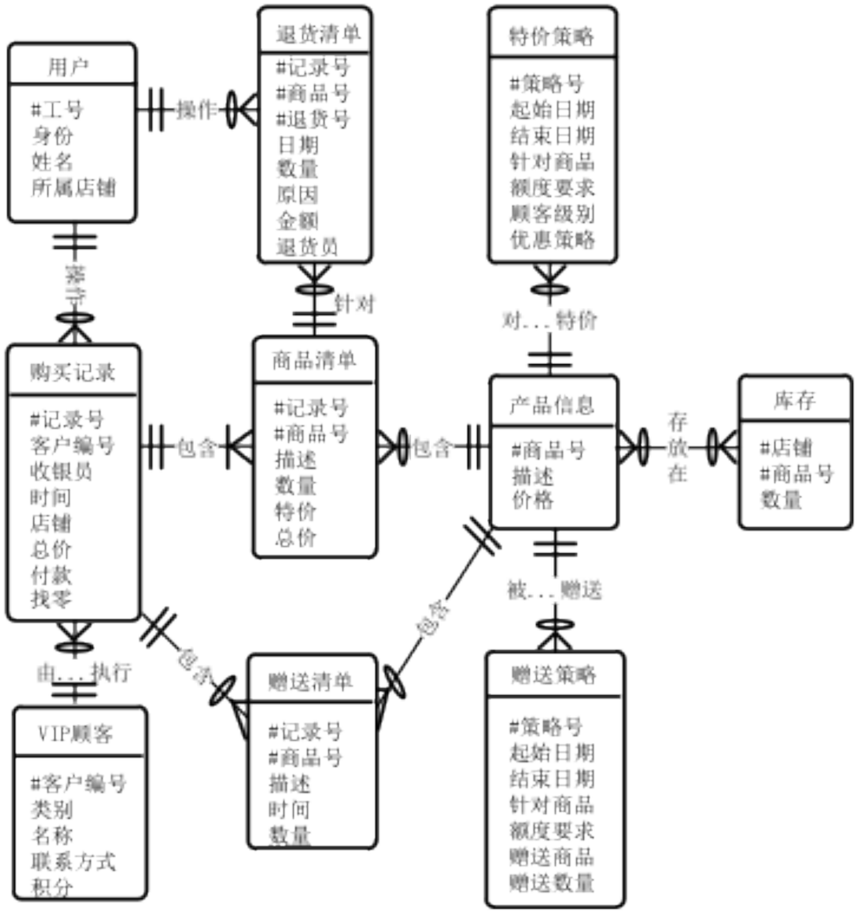
Cardinality and Multiplicity

建立实体关系图

- Level 1—model all data objects (entities) and their “connections” to one another
- Level 2—model all entities and relationships
- Level 3—model all entities, relationships, and the attributes that provide further depth


基本
元素
实体
Lable
关系
属性
attr1
attr2
...
标识符属性
#attr_id
...
关系
基数
Mandatory One
Mandatory Many
Optional One
Optional Many

ERD的图形表示


用户
#工号
身份
姓名
所属店铺
操作
退货清单
#记录号
#商品号
#退货号
日期
数量
原因
金额
退货员
特价策略
#策略号
起始日期
结束日期
针对商品
额度要求
顾客级别
优惠策略
客户
针对
对...特价
购买记录
#记录号
客户编号
收银员
时间
店铺
总价
付款
找零
包含
商品清单
#记录号
#商品号
描述
数量
特价
总价
包含
产品信息
#商品号
描述
价格
存放在
库存
#店铺
#商品号
数量
被...赠送
由...执行
包含
包含
VIP顾客
#客户编号
类别
名称
联系方式
积分
赠送清单
#记录号
#商品号
描述
时间
数量
赠送策略
#策略号
起始日期
结束日期
针对商品
额度要求
赠送商品
赠送数量

实体关系图

键（Key）

- 实体的一个或者多个属性能够唯一确定和标示每个实例，这些属性或者属性组合就被称为实体的标示符，或者键（Key）

Outline

- 需求分析基础
- 面向对象分析
- 结构化分析
- 使用需求分析方法细化和明确需求
  - 细化和明确需求内容
  - 建立系统需求

细化和明确需求

- 为什么要细化
  - 用户需求的描述的模糊性和系统设计所需要的严谨性之间的矛盾
- 如何细化
  - 需求分析建模
  - 发现其中的遗漏、冲突、冗余和错误
  - 迭代（获取、分析、获取、分析。。。）

系统顺序图有助于发现交互性的缺失

1.收银员输入会员编号；
2.收银员输入商品；
3.系统显示购买信息；
收银员重复 2-3 步，直至完成所有输入
4.系统显示总价和赠品信息；
5.顾客付款；
6.系统找零；
7.系统更新数据；
8.系统打印收据；
9.顾客离开

概念类图有助于发现

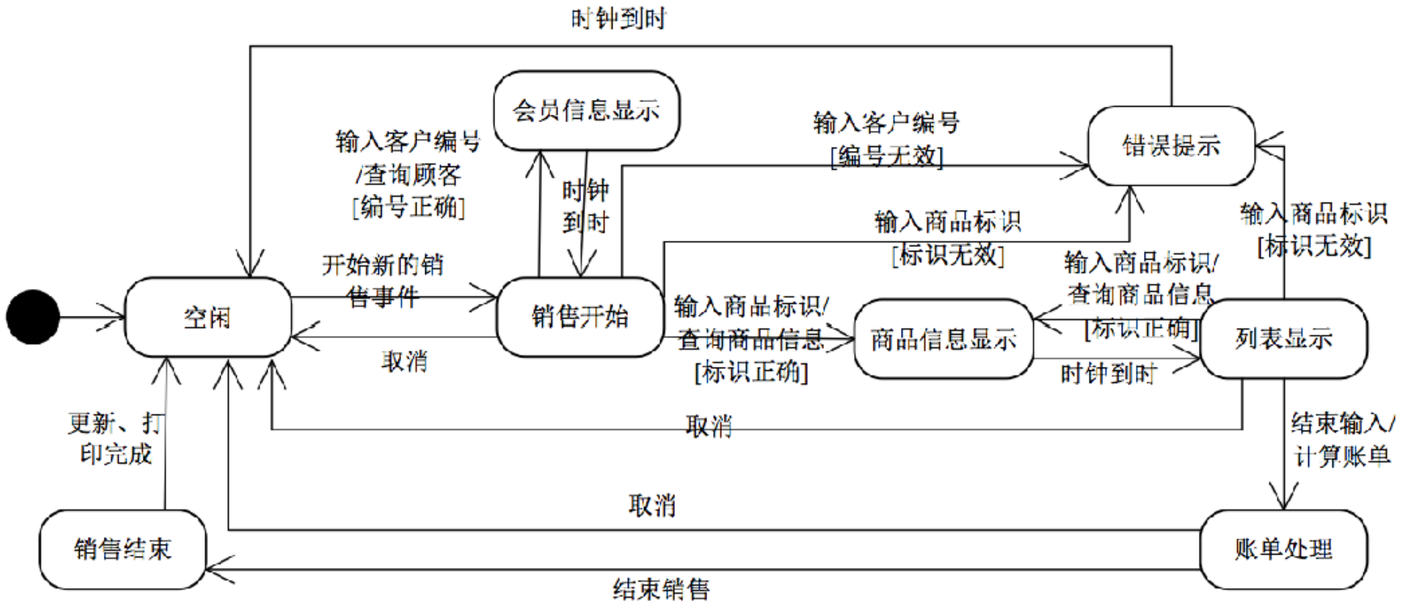
- 部分信息的使用不准确
  - 例如步骤 2 中输入的是商品标识,而不是商品,第 5 步显 示的已输入商品列表信息和总价。
- 部分信息不明确
  - 例如会员信息、商品信息、商品列表信息、赠品信息、更新的数据、收据等等各自的详细内容并没有描述。
- 遗漏了重要内容
  - 例如总价的计算需要使用商品特价策略和总额特价策略,赠品的计算需要使用商品赠送策略和总额赠送策略。

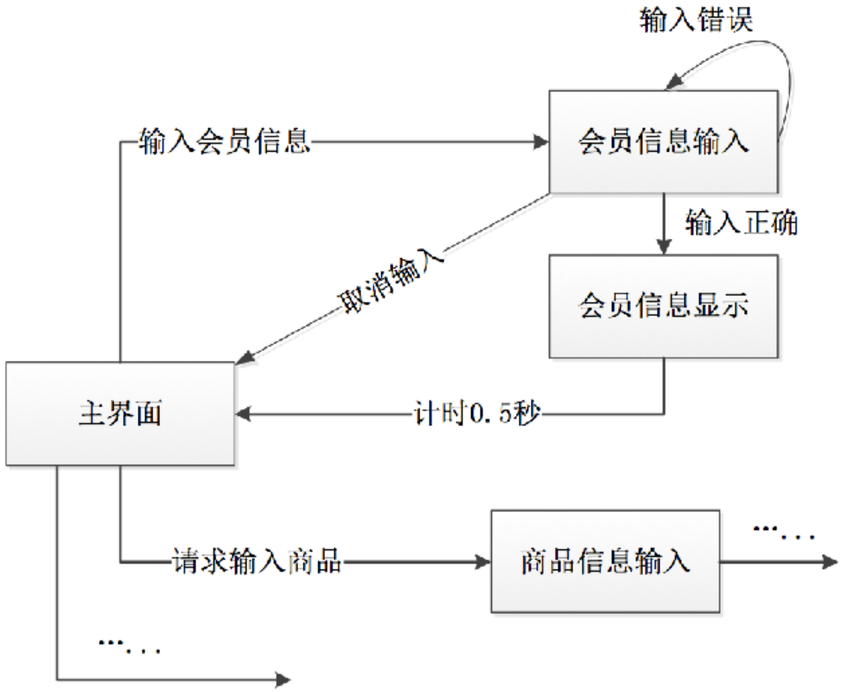
1. *收银员输入会员编号;*
2. *系统显示会员信息;*
3. *收银员输入商品;*
4. *系统显示输入商品的信息;*
5. *系统显示所有已输入商品的信息;*

收银员重复 3-5 步，直至完成所有输入

6. *收银员结束商品输入;*
7. *系统显示总价和赠品信息;*
8. *收银员请求顾客付款;*
9. *顾客支付，收银员输入支付数额;*
10. *系统显示应找零数额，收银员找零;*
11. *收银员结束销售;*
12. *系统更新数据，并打印收据。*

状态图有助于发现界面的跳转





建立系统需求

- 8种规格说明
- 不同的分析方法适合不同的规格说明

A.1 Template of SRS Section 3 organized by mode: Version 1

3. Specific requirements
- 3.1 External interface requirements
  - 3.1.1 User interfaces
  - 3.1.2 Hardware interfaces
  - 3.1.3 Software interfaces
  - 3.1.4 Communications interfaces
- 3.2 Functional requirements
  - 3.2.1 Mode 1
    - 3.2.1.1 Functional requirement 1.1
    - .
    - .
    - .
    - 3.2.1.*n* Functional requirement 1.*n*
  - 3.2.2 Mode 2
  - .
  - .
  - .
  - 3.2.*m* Mode *m*
    - 3.2.*m*.1 Functional requirement *m*.1
    - .
    - .
    - .
    - 3.2.*m*.*n* Functional requirement *m*.*n*
- 3.3 Performance requirements
- 3.4 Design constraints
- 3.5 Software system attributes
- 3.6 Other requirements

by mode

A.3 Template of SRS Section 3 organized by user class

3. Specific requirements
3.1 External interface requirements
  3.1.1 User interfaces
  3.1.2 Hardware interfaces
  3.1.3 Software interfaces
  3.1.4 Communications interfaces
3.2 Functional requirements
  3.2.1 User class 1
    3.2.1.1 Functional requirement 1.1
    .
    .
    .
    3.2.1.*n* Functional requirement 1.*n*
  3.2.2 User class 2
  .
  .
  .
  3.2.*m* User class *m*
    3.2.*m*.1 Functional requirement *m*.1
    .
    .
    .
    3.2.*m*.*n* Functional requirement *m*.*n*
3.3 Performance requirements
3.4 Design constraints
3.5 Software system attributes
3.6 Other requirements

by user class

A.4 Template of SRS Section 3 organized by object

3. Specific requirements
3.1 External interface requirements
3.1.1 User interfaces
3.1.2 Hardware interfaces
3.1.3 Software interfaces
3.1.4 Communications interfaces
3.2 Classes/Objects
3.2.1 Class/Object 1

3.2.1.1 Attributes (direct or inherited)
3.2.1.1.1 Attribute 1
.
.
.
3.2.1.1 *n* Attribute *n*
3.2.1.2 Functions (services, methods, direct or inherited)
3.2.1.2.1 Functional requirement 1.1
.
.
.
3.2.1.2 *m* Functional requirement 1 *m*
3.2.1.3 Messages (communications received or sent)
3.2.2 Class/Object 2
.
.
.
3.2 *p* Class/Object *p*
3.3 Performance requirements
3.4 Design constraints
3.5 Software system attributes
3.6 Other requirements

by object

A.5 Template of SRS Section 3 organized by feature

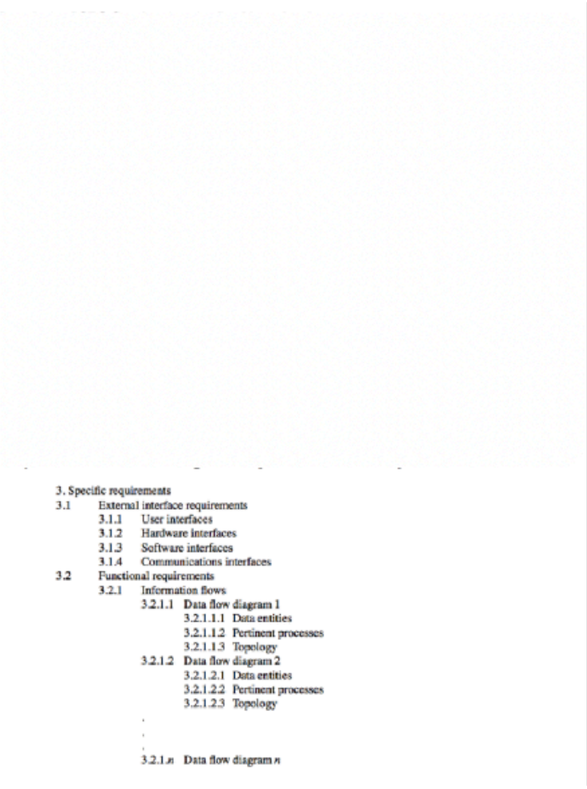
3. Specific requirements
- 3.1 External interface requirements
  - 3.1.1 User interfaces
  - 3.1.2 Hardware interfaces
  - 3.1.3 Software interfaces
  - 3.1.4 Communications interfaces
- 3.2 System features
  - 3.2.1 System Feature 1
    - 3.2.1.1 Introduction/Purpose of feature
    - 3.2.1.2 Stimulus/Response sequence
    - 3.2.1.3 Associated functional requirements
      - 3.2.1.3.1 Functional requirement 1
      - .
      - .
      - .
      - 3.2.1.3.*n* Functional requirement *n*
  - 3.2.2 System feature 2
  - .
  - .
  - .
  - 3.2.*m* System feature *m*
  - .
  - .
  - .
- 3.3 Performance requirements
- 3.4 Design constraints
- 3.5 Software system attributes
- 3.6 Other requirements

by feature

A.6 Template of SRS Section 3 organized by stimulus

3. Specific requirements
3.1 External interface requirements
  3.1.1 User interfaces
  3.1.2 Hardware interfaces
  3.1.3 Software interfaces
  3.1.4 Communications interfaces
3.2 Functional requirements
  3.2.1 Stimulus 1
    3.2.1.1 Functional requirement 1.1
    .
    .
    .
    3.2.1.*n* Functional requirement 1.*n*
  3.2.2 Stimulus 2
  .
  .
  .
  3.2.*m* Stimulus m
    3.2.*m*.1 Functional requirement *m*.1
    .
    .
    .
    3.2.*m*.*n* Functional requirement *m*.*n*
3.3 Performance requirements
3.4 Design constraints
3.5 Software system attributes
3.6 Other requirements

by stimulus



3. Specific requirements
- 3.1 External interface requirements
  - 3.1.1 User interfaces
  - 3.1.2 Hardware interfaces
  - 3.1.3 Software interfaces
  - 3.1.4 Communications interfaces
- 3.2 Functional requirements
  - 3.2.1 Information flows
    - 3.2.1.1 Data flow diagram 1
      - 3.2.1.1.1 Data entities
      - 3.2.1.1.2 Pertinent processes
      - 3.2.1.1.3 Topology
    - 3.2.1.2 Data flow diagram 2
      - 3.2.1.2.1 Data entities
      - 3.2.1.2.2 Pertinent processes
      - 3.2.1.2.3 Topology
    - .
    - .
    - .
    - 3.2.1.*n* Data flow diagram *n*
      - 3.2.1.*n*.1 Data entities
      - 3.2.1.*n*.2 Pertinent processes
      - 3.2.1.*n*.3 Topology
  - 3.2.2 Process descriptions
    - 3.2.2.1 Process 1
      - 3.2.2.1.1 Input data entities
      - 3.2.2.1.2 Algorithm or formula of process
      - 3.2.2.1.3 Affected data entities
    - 3.2.2.2 Process 2
      - 3.2.2.2.1 Input data entities
      - 3.2.2.2.2 Algorithm or formula of process
      - 3.2.2.2.3 Affected data entities
    - .
    - .
    - .
    - 3.2.2.*m* Process *m*
      - 3.2.2.*m*.1 Input data entities
      - 3.2.2.*m*.2 Algorithm or formula of process
      - 3.2.2.*m*.3 Affected data entities
  - 3.2.3 Data construct specifications
    - 3.2.3.1 Construct 1
      - 3.2.3.1.1 Record type
      - 3.2.3.1.2 Constituent fields
    - 3.2.3.2 Construct 2
      - 3.2.3.2.1 Record type
      - 3.2.3.2.2 Constituent fields
    - .
    - .
    - .
    - 3.2.3.*p* Construct *p*
      - 3.2.3.*p*.1 Record type
      - 3.2.3.*p*.2 Constituent fields
  - 3.2.4 Data dictionary
    - 3.2.4.1 Data element 1
      - 3.2.4.1.1 Name
      - 3.2.4.1.2 Representation
      - 3.2.4.1.3 Units/Format
      - 3.2.4.1.4 Precision/Accuracy
      - 3.2.4.1.5 Range
    - 3.2.4.2 Data element 2
      - 3.2.4.2.1 Name
      - 3.2.4.2.2 Representation
      - 3.2.4.2.3 Units/Format
      - 3.2.4.2.4 Precision/Accuracy
      - 3.2.4.2.5 Range
    - .
    - .
    - .
    - 3.2.4.*q* Data element *q*
      - 3.2.4.*q*.1 Name
      - 3.2.4.*q*.2 Representation
      - 3.2.4.*q*.3 Units/Format
      - 3.2.4.*q*.4 Precision/Accuracy
      - 3.2.4.*q*.5 Range
- 3.3 Performance requirements
- 3.4 Design constraints
- 3.5 Software system attributes
- 3.6 Other requirements

by functional hierarchy

A.8 Template of SRS Section 3 showing multiple organizations

3. Specific requirements
3.1 External interface requirements
3.1.1 User interfaces
3.1.2 Hardware interfaces
3.1.3 Software interfaces
3.1.4 Communications interfaces
3.2 Functional requirements
3.2.1 User class 1
3.2.1.1 Feature 1.1
3.2.1.1.1 Introduction/Purpose of feature
3.2.1.1.2 Stimulus/Response sequence
3.2.1.1.3 Associated functional requirements
3.2.1.2 Feature 1.2
3.2.1.2.1 Introduction/Purpose of feature
3.2.1.2.2 Stimulus/Response sequence
3.2.1.2.3 Associated functional requirements
.
.
.
3.2.1.*m* Feature 1.*m*
3.2.1.*m*.1 Introduction/Purpose of feature
3.2.1.*m*.2 Stimulus/Response sequence
3.2.1.*m*.3 Associated functional requirements
3.2.2 User class 2
.
.
.
3.2.*n* User class *n*
.
.
.
3.3 Performance requirements
3.4 Design constraints
3.5 Software system attributes
3.6 Other requirements

multiple organization

下一章重点

- 为什么要文档化
- 模版
- 写作要点