第8章.面向对象建模

主要内容

1. 面向对象分析
2. 对象模型
3. 用例模型
4. 行为模型
5. 对象约束语言OCL

1. 面向对象分析
——现实世界的复杂模型

- 复杂总是简单部分的组合
- 简单部分又是更简单部分的组合
  - 简单组成复杂的过程存在层次性
- 每个最小简单部分独立负责完成一系列相关任务
- 相比较而言，每个组合内部各部分的关系比其内部与外部的关系都更紧密
- 各个部分通过一致的接口进行组合，即一个部分对其它部分的所知仅仅是接口

1. 面向对象分析
——映射现实模型的面向对象思想

- 任何系统都是能够完成一系列相关目标和任务的对象
- 对象完成一个任务时会请求一系列其他对象帮助其完成一些子目标
- 其他对象为了完成其任务又会请求将子目标更细分为子子目标，并请求其他对象帮助完成
- 子目标的分解和责任分担一直进行直到最后产生的子部分可以映射到计算实体
- 计算实体：对象
- 层次关系：聚合（组合）、继承、关联
- 组合接口：一个对象暴露的接口

1. 面向对象分析
——面向对象建模

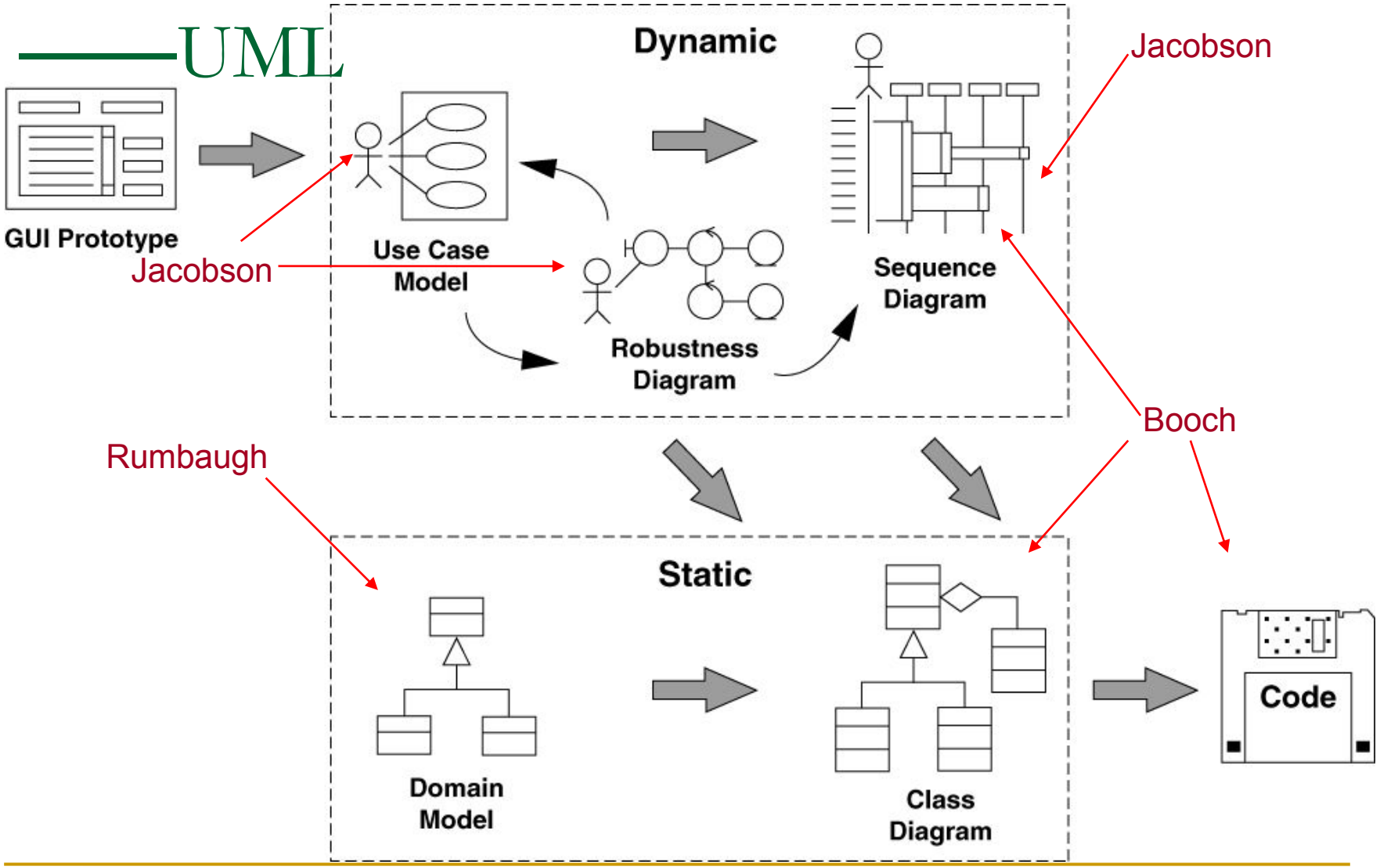
- 面向对象建模：一种用于辨识系统环境中的对象及这些对象之间关系的技术
  - OMT (James Rumbaugh)
  - Booch方法(Grady Booch)
  - OOSE (Ivar Jacobson)

  } UML

  - Coad-Yourdon
  - Shlaer-Mellor
  - Fusion

1. 面向对象分析

——UML


Dynamic
Jacobson
GUI Prototype
Jacobson
Use Case Model
Sequence Diagram
Robustness Diagram
Booch
Rumbaugh
Static
Domain Model
Class Diagram
Code

1. 面向对象分析——UML

- UML
  - 对象模型Object Model (Domain Model)
  - 用例模型Use Case Model
  - 行为模型Behavior Model
    - 交互图(顺序图,通讯图)
    - 活动图
    - 状态图
  - 对象约束语言OCL

主要内容

1. 面向对象分析
2. 对象模型
    1. 组成元素
    2. 重要概念
    3. 领域模型
3. 用例模型
4. 行为模型
5. 对象约束语言OCL

2.1 对象模型组成元素——对象

- 对象
  - 对象是指在一个应用当中具有明确角色的独立可确认的实体
- 每个对象都要包含
  - 标识
    - 唯一的标识自己，引用
  - 状态
    - 对象的特征描述，包括对象的属性和属性的取值
  - 行为
    - 对象在其状态发生改变或者接收到外界消息时所采取的行动

2.1 对象模型组成元素——对象

- 常见的事物都可以是对象
  - 和系统存在交互的外部实体，例如人、设备、其他的软件系统等；
  - 问题域中存在的事物，例如报表、信息展示、信号等；
  - 在系统的上下文环境中发生的事件，例如一次外部控制行为、一次资源变化等；
  - 人们在与系统的交互之中所扮演的角色，例如系统管理人员、用户管理人员、普通用户等；
  - 和应用相关的组织单位，例如分公司、部门、团队、小组等；
  - 问题域中问题发生的地点，例如车间、办公室等；
  - 事物组合的结构关系，例如部分与整体的关系等。

2.1 对象模型组成元素——对象

- 但是也有事物不是对象
  - 无法界定的事物
  - 纯粹的值
  - 纯粹的行为

2.1 对象模型组成元素
——对象

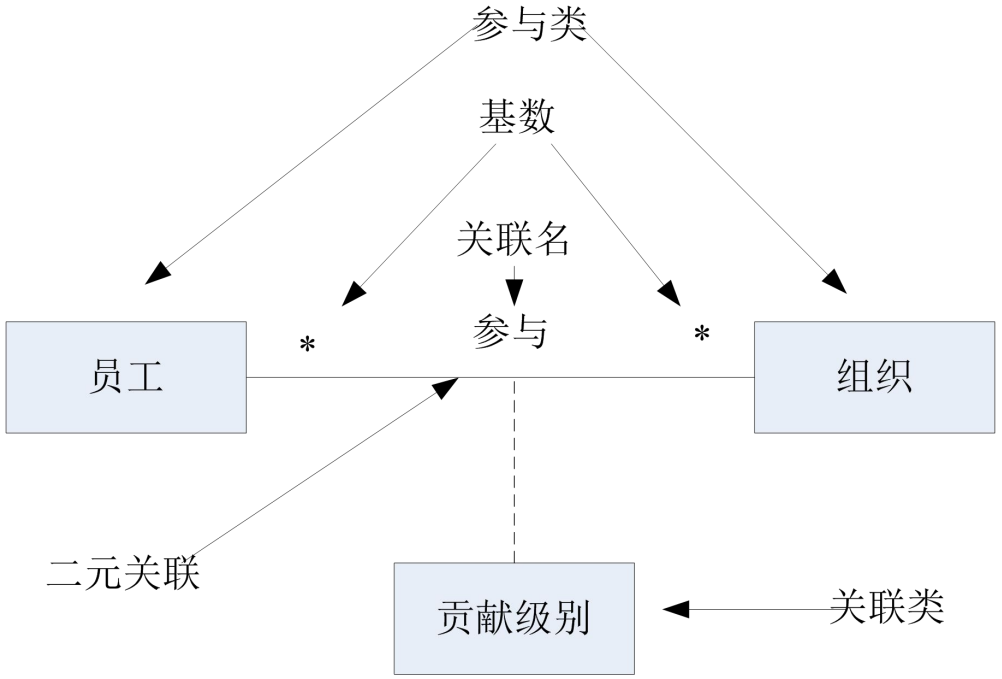
- 一个对象维护其自身的状态需要对外公开一些方法，行使其职能也要对外公开一些方法，这些方法组合起来定义了该对象允许外界访问的方法，或者说限定了外界可以期望的表现，它们是对象需要对外界履行的协议（Protocol）
- 一个对象的整体协议可能会分为多个内聚的逻辑行为组，划分后的每一个逻辑行为组就描述了对象的一个独立职责，体现了对象的一个独立角色
- 对象职责
  - 职责是指对象持有、维护特定知识并基于知识行使固定职能的能力
- 如果一个对象拥有多个行为组，就意味着该对象拥有多个不同的职责，需要扮演多个不同的角色。
- 理想的单一职责对象应该仅仅扮演一个角色

2.1 对象模型组成元素——类

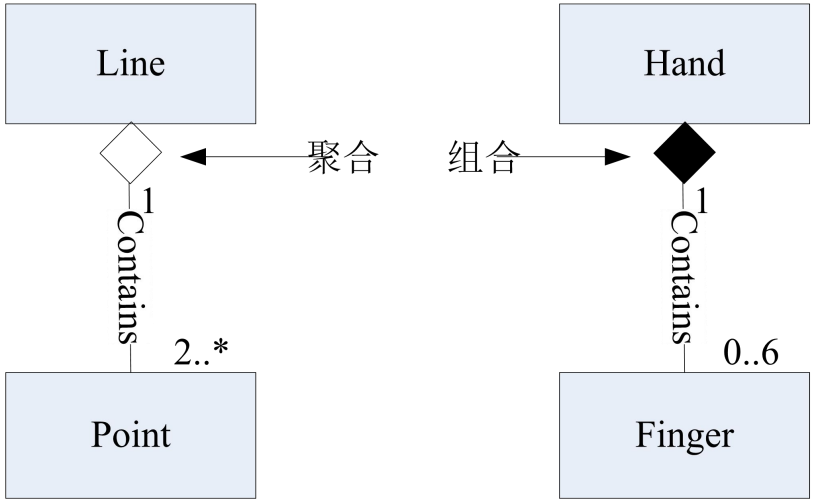
- 类
  - 类是共享相同属性和行为的对象的集合，它为属于该类的所有对象提供统一的抽象描述和生成模板
    - 抽象描述称为接口（Interface），定义了类所含对象对外的（其他类和对象）的统一协议
    - 生成模板称为实现（Implementation），说明了类所含对象的生成机制和行为模式

2.1 对象模型组成元素——关联

- 关联
  - 指出了类之间的某种语义联系
  - 关联是类对其对象实例之间的无数潜在关系的描述

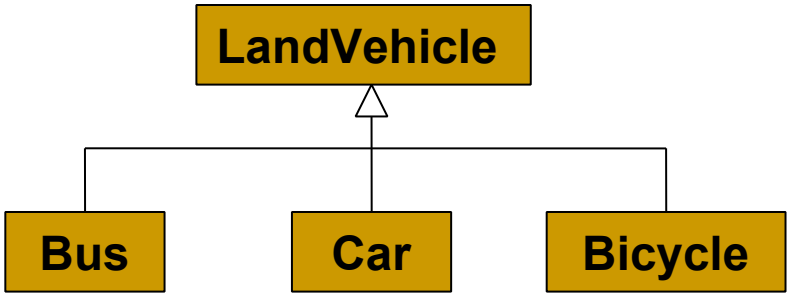


2.1 对象模型组成元素
——聚合和组合



2.2 对象模型重要概念
——继承

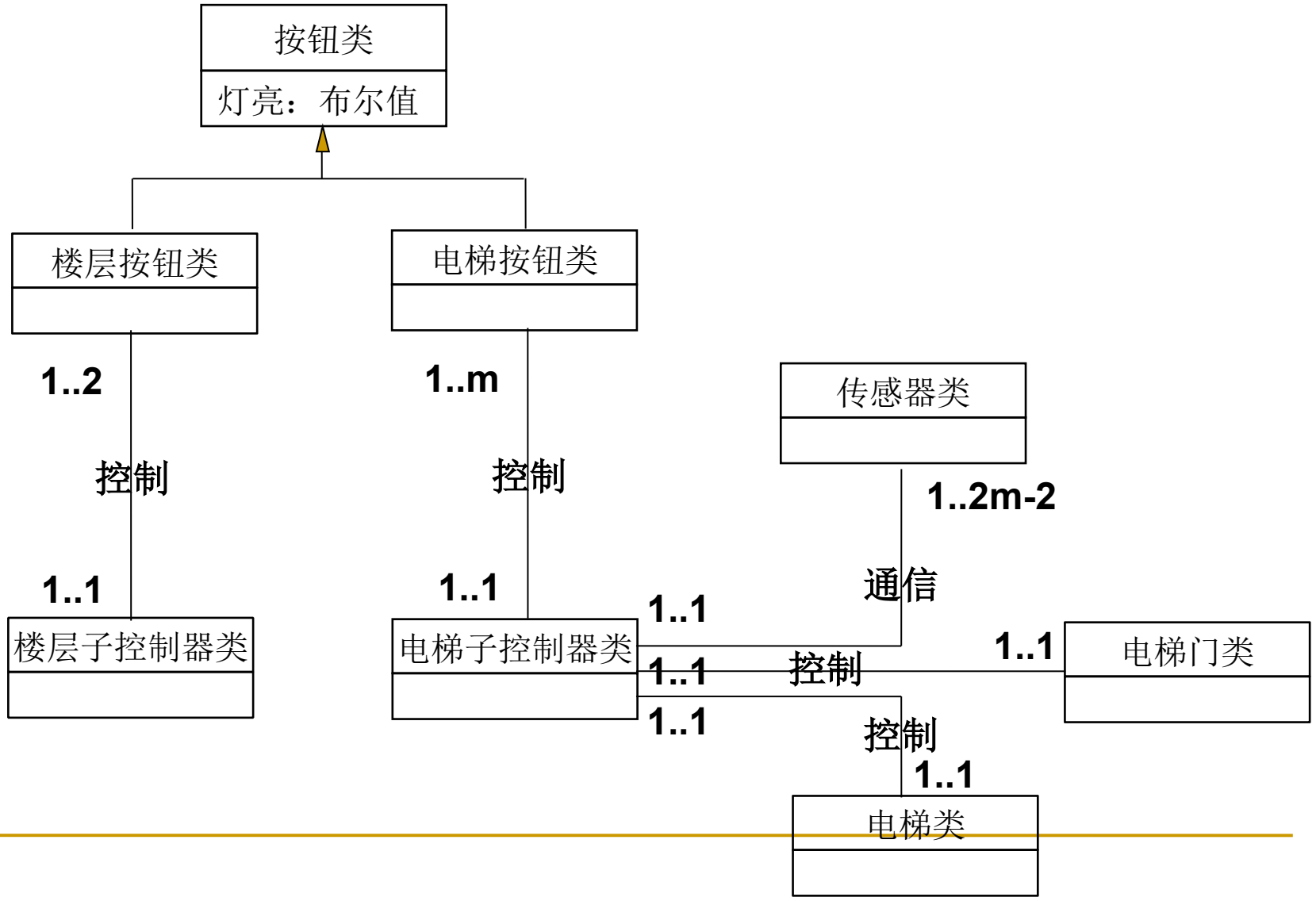
- 如果一个类A继承了对象B，那么A就自然具有B的全部属性和服务，同时A也会拥有一些自己特有的属性和服务，这些特有部分是B所不具备的



2.3 领域模型

- 类大多是概念类（Concept Class），是一个能够代表现实世界事物的概念
- 概念类之间存在指明语义联系的关联，这些关联通常不标记方向，也不标记关联端的可见性
- 概念类会显式的描述自己的一些重要属性，但不是全部的详细属性，而且概念类的属性通常没有类型的约束
- 概念类不显式的标记类的行为，即概念类不包含明确的方法

2.3 领域模型


按钮类
灯亮：布尔值
楼层按钮类
电梯按钮类
1..2
1..m
控制
控制
传感器类
1..2m-2
通信
1..1
1..1
楼层子控制器类
电梯子控制器类
1..1
1..1
1..1
控制
电梯门类
控制
1..1
电梯类

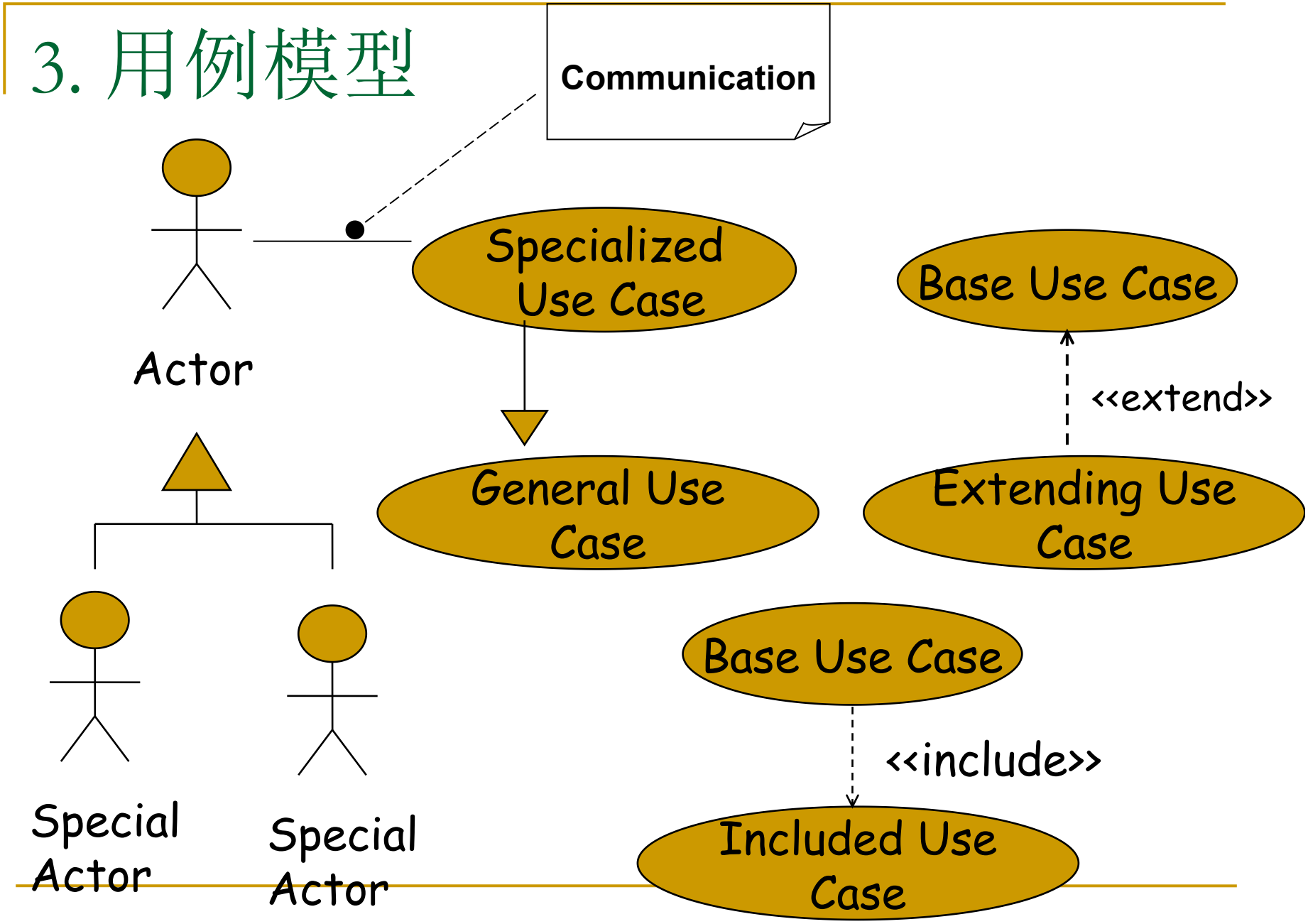
主要内容

1. 面向对象分析
2. 对象模型
3. 用例模型
4. 行为模型
5. 对象约束语言OCL

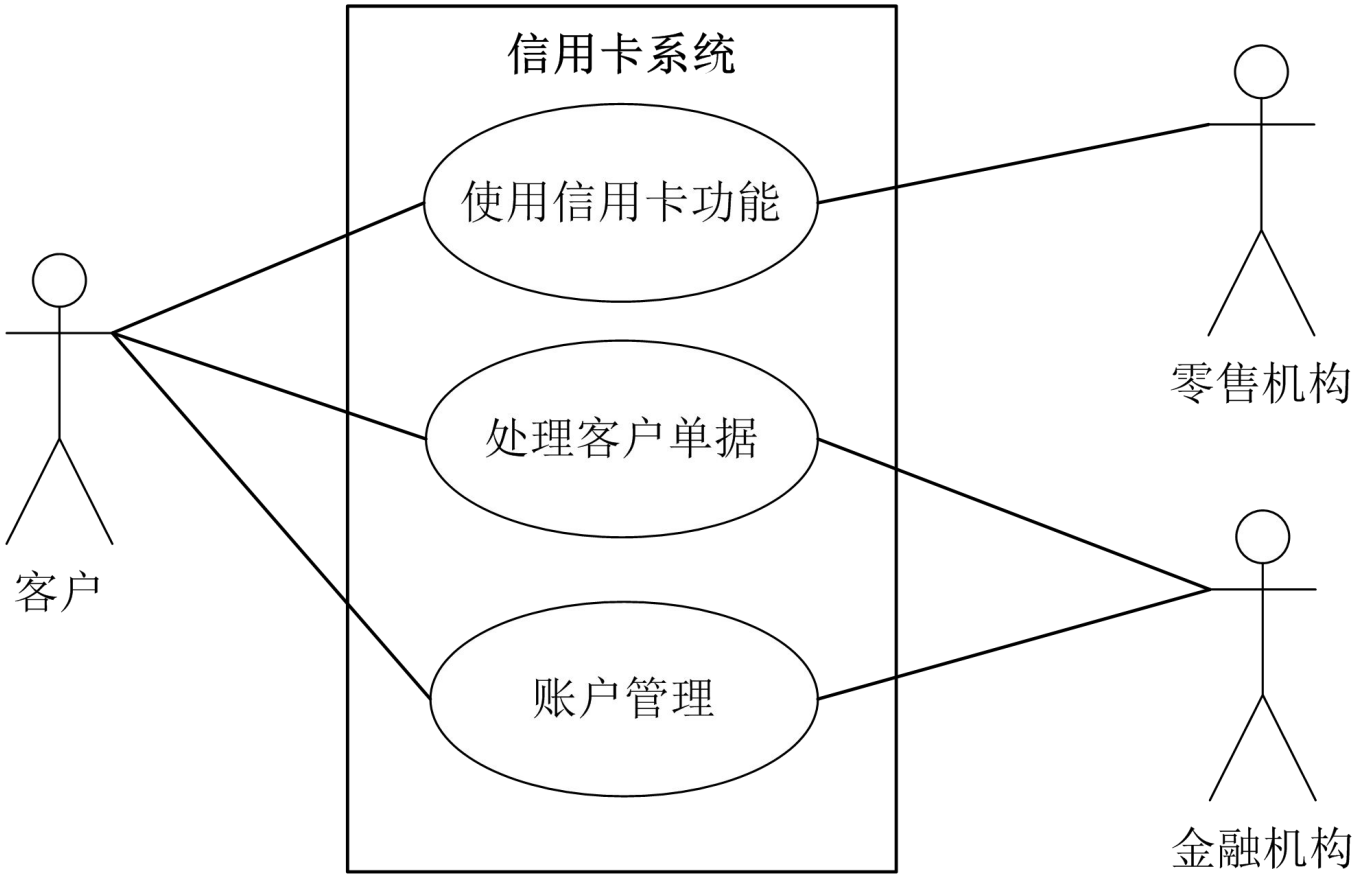
3. 用例模型

- 用例模型就是以用例为基本单位建立的一个系统功能展示模型，它是系统所有用例的集合，以统一、图形化方式展示系统的功能和行为特性
- 用例是获取和组织用户需求的手段
- 基本元素
  - 用例（Use Case）
  - 参与者（Actor）
  - 关系（Relationship）
  - 系统边界（System Boundary）

3. 用例模型


Communication
Actor
Specialized Use Case
General Use Case
Base Use Case
<<extend>>
Extending Use Case
Special Actor
Special Actor
Base Use Case
<<include>>
Included Use Case

3. 用例模型



主要内容

1. 面向对象分析
2. 对象模型
3. 用例模型
4. 行为模型
5. 对象约束语言OCL

4. 行为模型

- 行为模型
  - 交互图（Interaction Diagram）
    - 顺序图（Sequence Diagram）
    - 通信图（Communication Diagram）
    - 交互概述图（Interaction Overview Diagram）
    - 时间图（Timing Diagram）
  - 活动图（Activity Diagram）
  - 状态图（State Diagram）

4. 行为模型
——交互图

- 交互图
  - 以一组对象为中心的交互描述技术
  - 描述在特定上下文环境中一组对象的交互行为
  - 互图通常描述的是单个用例的典型场景
  - 交互图中的每一个交互都描述了环境中的对象为了实现某个目标而执行的一系列消息交换
  - 顺序图和通信图是最常用的 交互图
  - 交互图中出现的对象应该在领域模型中有相应的对象存在

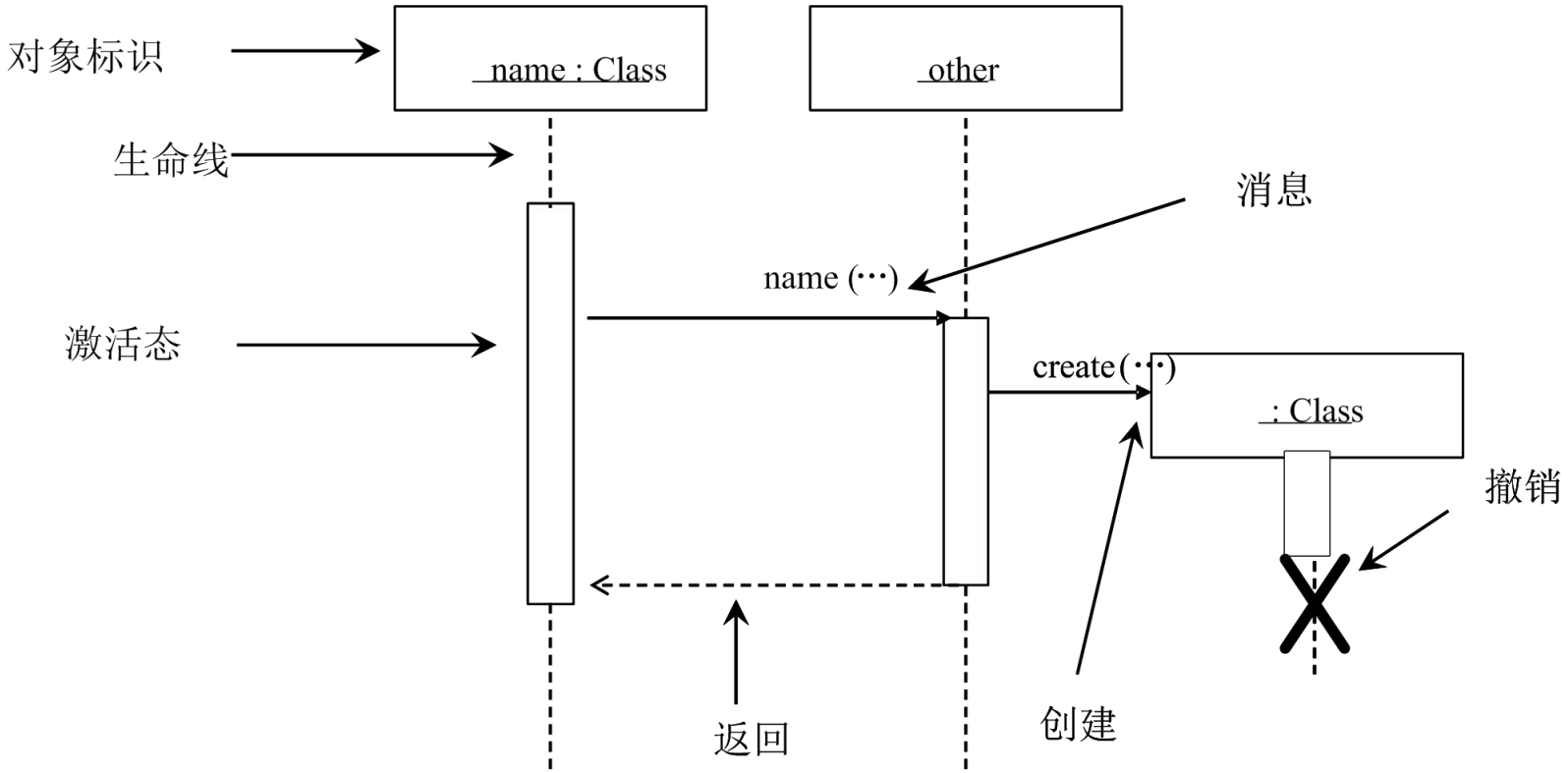
4. 行为模型
——交互图

同步消息

异步消息

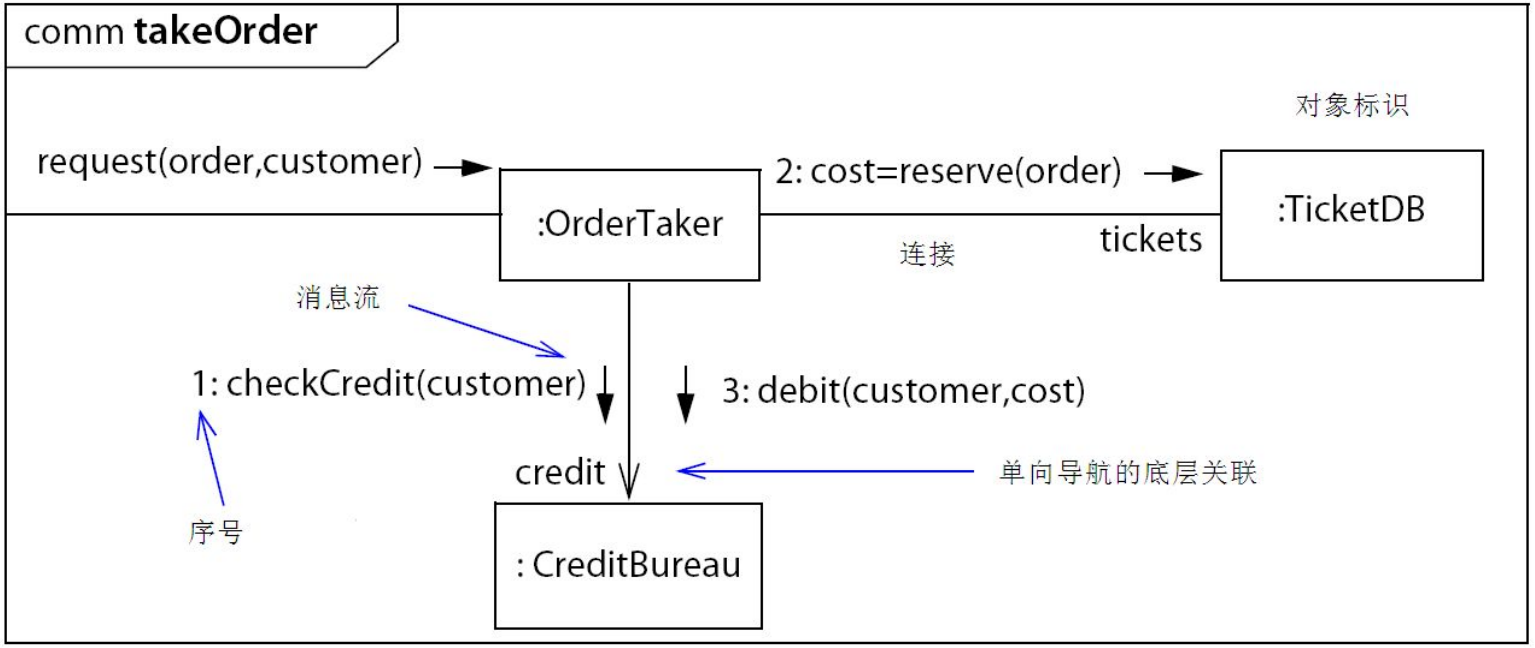
返回消息

- 顺序图



4. 行为模型
——交互图

- 通信图
  - 消息[sequence-expression:]message
    - message＝[attribute=]name[(argument)][:return-value]
    - sequence-expression ＝label [iteration-expression]
      - iteration-expression ＝[*[iteration-clause]|[condition-clause]]

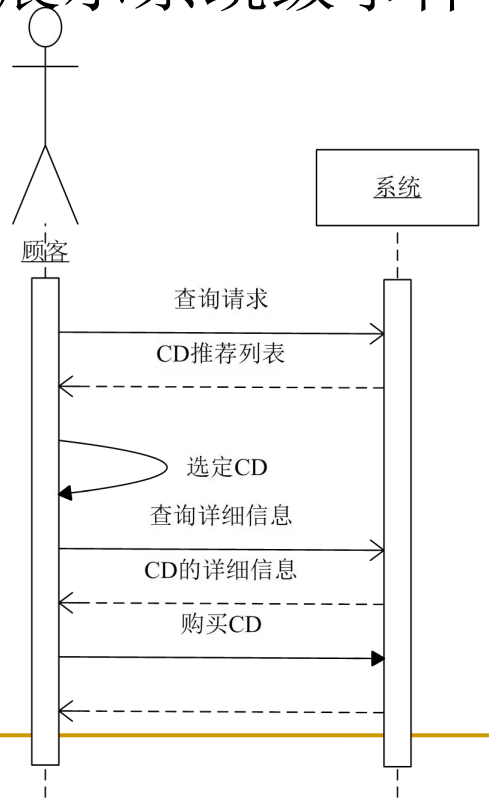


4. 行为模型
——交互图

- 系统顺序图
  - 将整个系统看作一个黑箱的对象，强调外部参与者和系统的交互行为，重点展示系统级事件

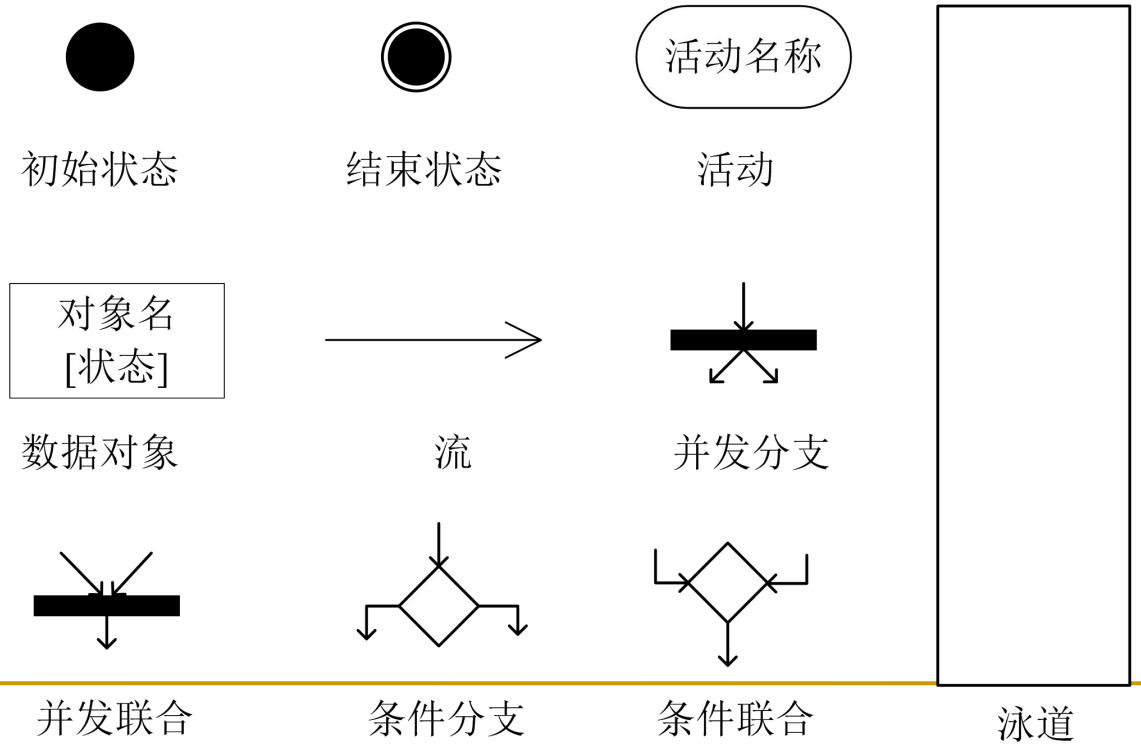
用例描述:

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.



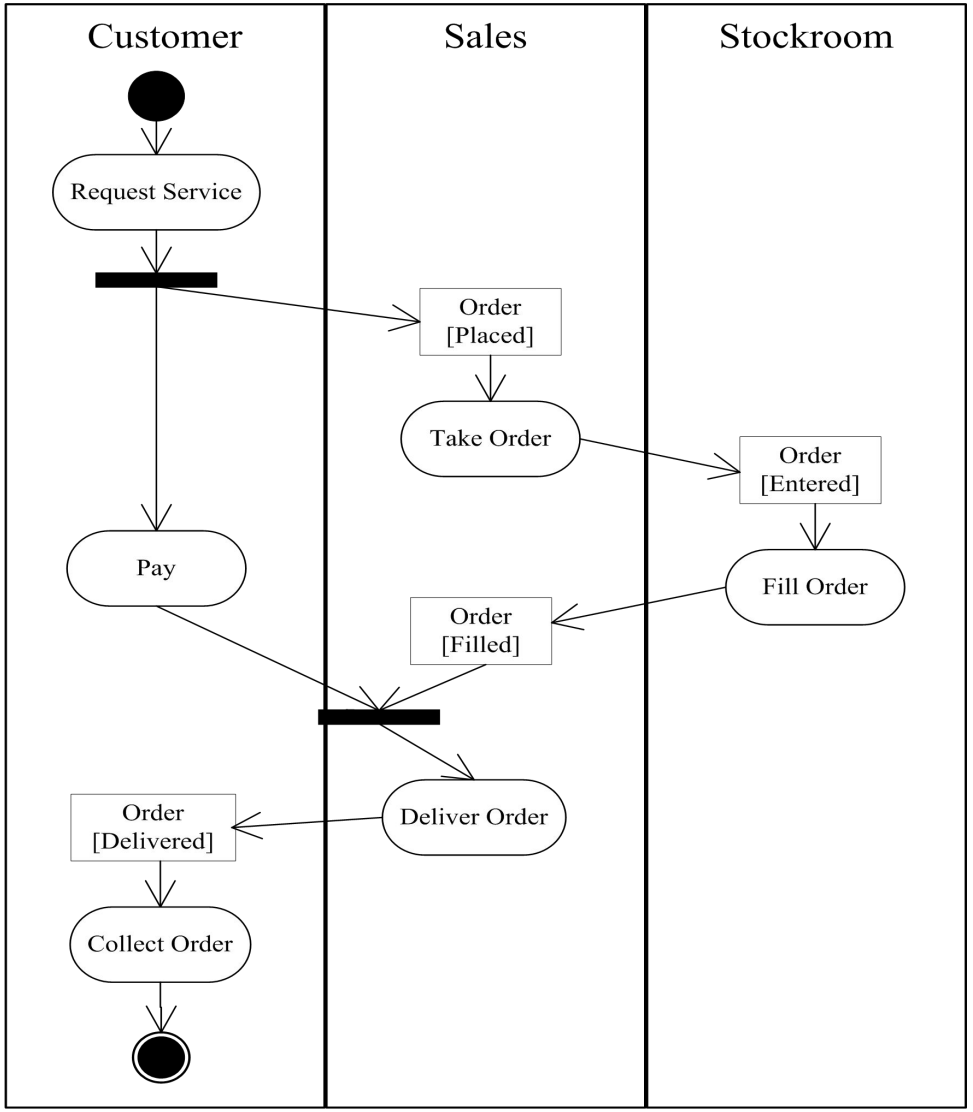
4. 行为模型
——活动图

- 活动图
  - 借鉴了多种BPM后建立的行为图
  - 以“流”（控制流和数据流）处理为侧重点描述系统的行为
  - 通常以组织的整体业务流程为描述对象的



4. 行为模型——活动图

- 示例


Customer
Sales
Stockroom
Request Service
Order [Placed]
Take Order
Order [Entered]
Pay
Fill Order
Order [Filled]
Deliver Order
Order [Delivered]
Collect Order

4. 行为模型 ——状态图

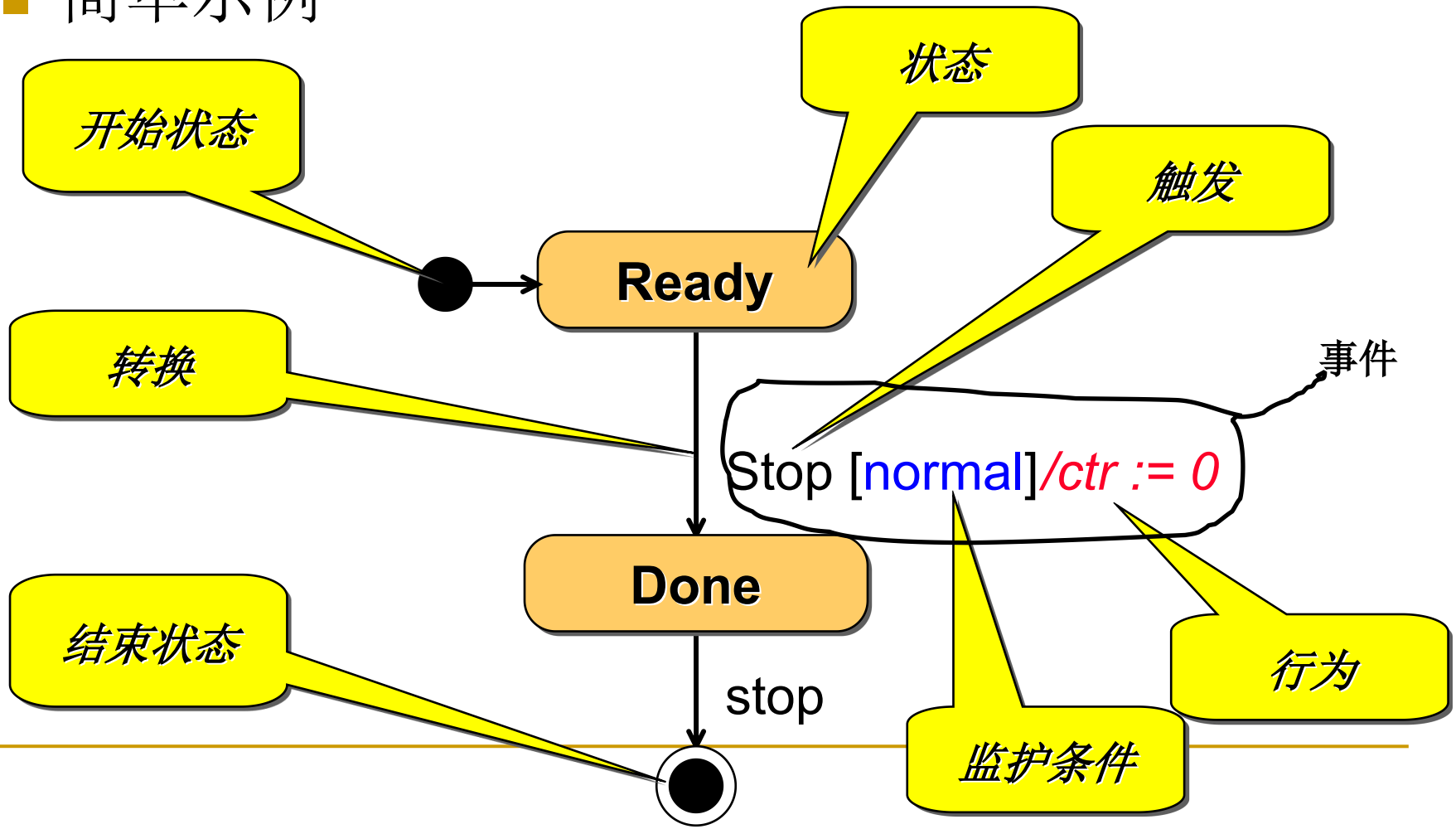
- 状态图
  - 以状态机理论为基础建立的对系统行为的描述手段
    - 状态机是以“状态”概念为基础解释系统行为的一种技术
    - 有限状态机FSM（Finite State Machine）是用于建模的最简单的状态机
    - 在FSM技术基础之上，发展出了多种分支技术，UML的状态图SD（State Diagram） 也是其中之一。
  - 主要用于描述重要而且复杂的对象的所有行为
    - 这个对象的行为通常要涉及很多（甚至大部分）的用例

4. 行为模型
——状态图

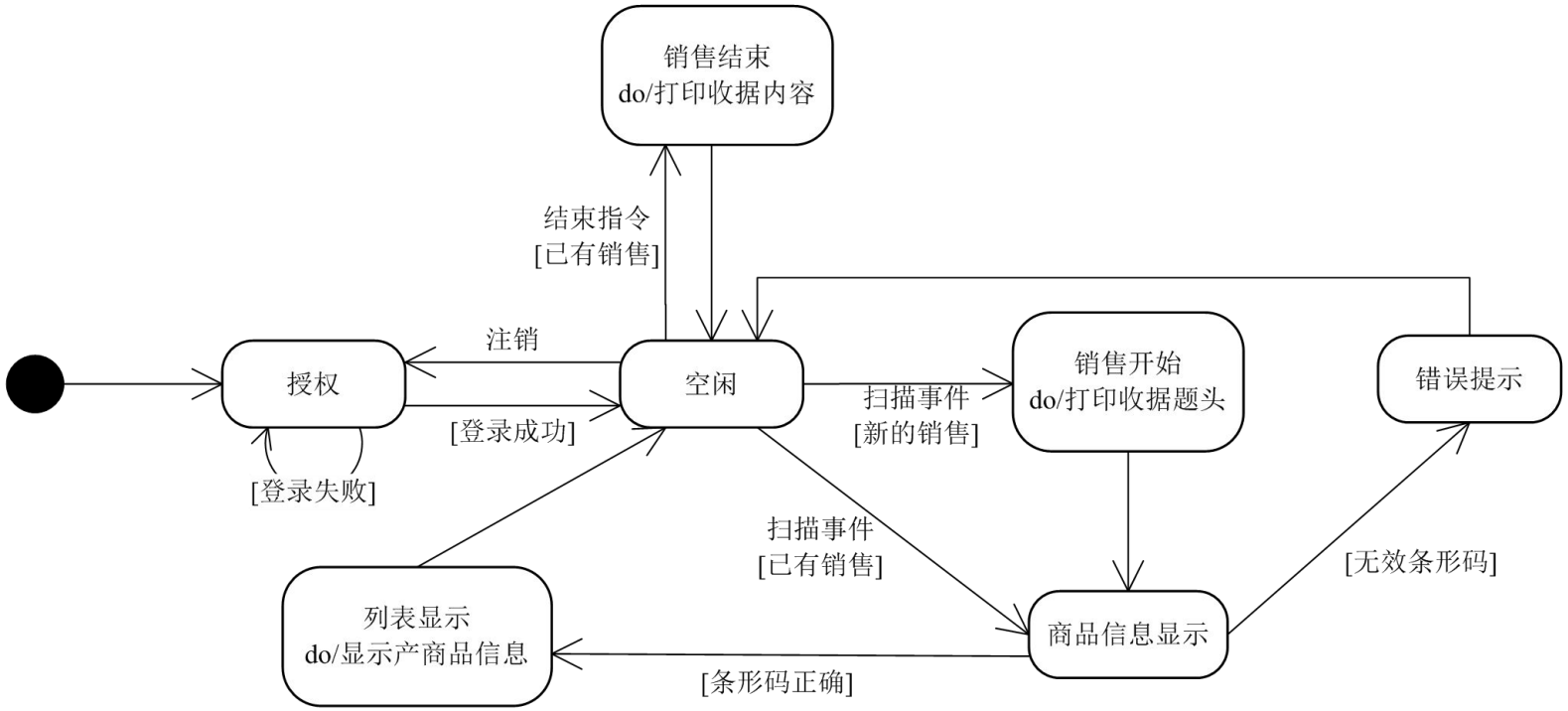
- 状态机理论
  - 状态机理论认为，系统总是处于一定的状态之中。而且，在某一时刻，系统只能处于一种状态之中。
  - 系统在任何一个状态中都是稳定的，如果没有外部事件触发，系统会一直持续维持该状态。
  - 如果发生有效的触发事件，系统将会响应事件，从一种状态转移到唯一的另一种状态。
- 如果能够罗列出系统所有可能的状态，并发现所有有效的外部事件，那么就能够从状态转移的角度完整的表达系统的所有行为

4. 行为模型
——状态图

- 简单示例



4. 行为模型
——状态图


销售结束
do/打印收据内容
结束指令
[已有销售]
注销
授权
空闲
销售开始
do/打印收据题头
错误提示
[登录成功]
扫描事件
[新的销售]
[登录失败]
扫描事件
[已有销售]
[无效条形码]
列表显示
do/显示产商品信息
商品信息显示
[条形码正确]

主要内容

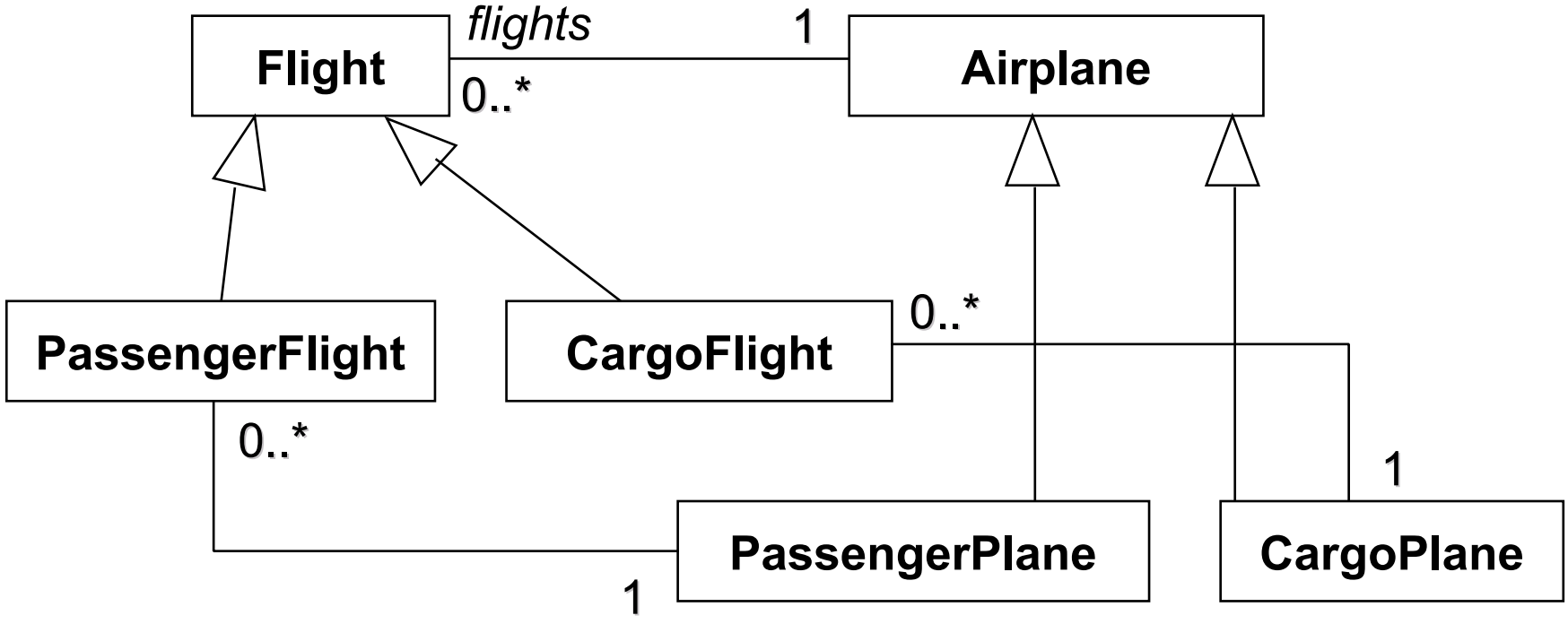
1. 面向对象分析
2. 对象模型
3. 用例模型
4. 行为模型
5. 对象约束语言OCL

5. 对象约束语言OCL

- OCL并不是UML中单独的一个模型，而是被应用在其他的模型当中，丰富其他模型的语义
- OCL是一种无副作用的规约语言
  - 以表达式的方式定义对其他模型元素的约束
  - 约束和限制其他模型元素的行为和状态变化
  - 不会修改任何其他模型元素的表述
- OCL不是一种编程语言。
  - OCL的首要定位是建模语言，因此它在保证一定表达能力的前提下，注重于语言的简洁性和抽象性
  - 它无法被用来描述程序的控制逻辑和工作流程，它的表达式定义也无法在程序中得到直接的执行

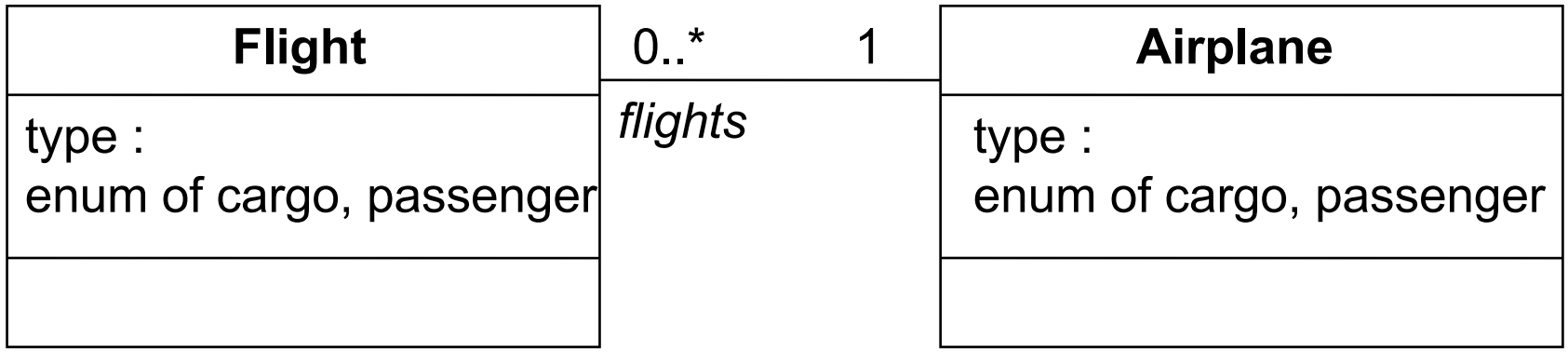
5. 对象约束语言OCL

- 示例



5. 对象约束语言OCL

- 示例



{context Flight
inv: type = #cargo implies airplane.type = #cargo
inv: type = #passenger implies airplane.type = #passenger}

本章小结

- 面向对象分析是90年代之后的主流分析方法，它以UML为基础，综合使用了多种不同的分析技术，主要有：
  - 对象模型Object Model (Domain Model)
  - 用例模型Use Case Model
  - 行为模型Behavior Model
    - 状态机模型
  - 对象约束语言OCL
- 面向对象分析任务的成功执行，除了要掌握多种面向对象分析技术之外，还需要掌握利用这些技术的建模方法