



第二部分：如何运用UML建模

第五章 用例建模

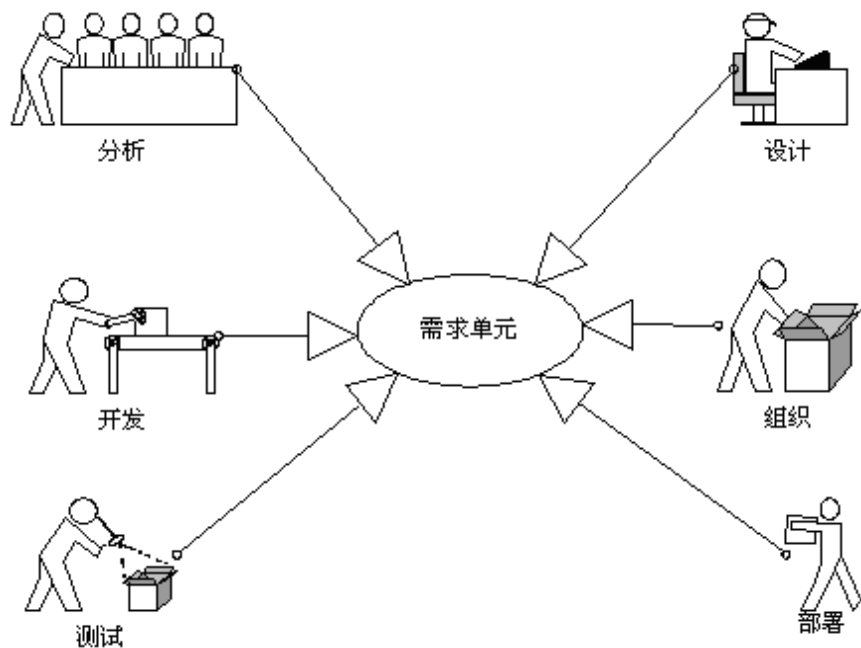
提纲



- 概述
- 基本概念
- 用例建模的步骤
- 用例规格说明
- 用例之间的关系
- 小结

概述

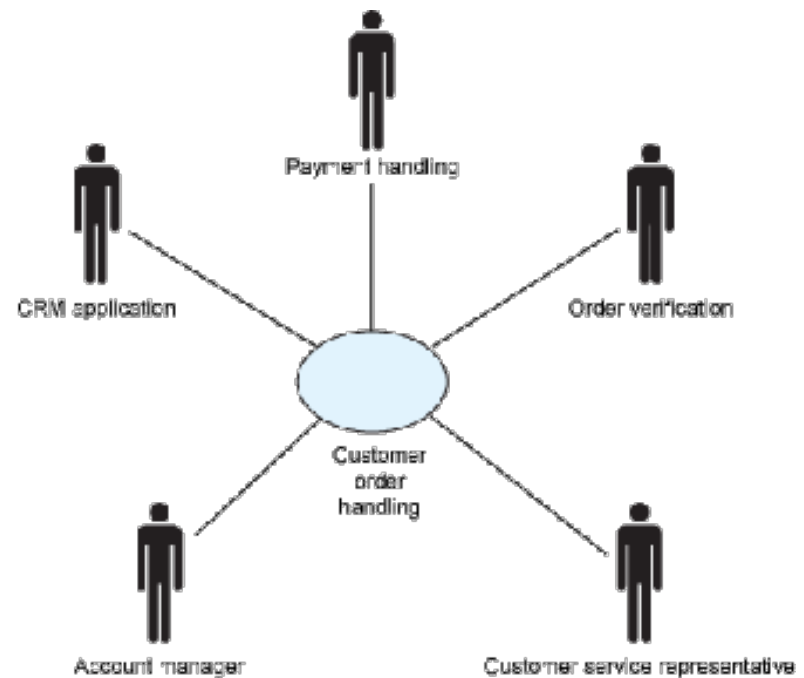
- 用例
 - 捕获系统的功能性需求
 - 提供了由需求过渡到软件开发过程的桥梁
 - UML建模的“驱动力”



- 用例建模的目标
 - 阐明和增进对系统需求的理解
 - 帮助定义待开发系统的边界
 - 为产生初始的类提供输入
 - 为分析模型的建立设立出发点
 - 为客户、用户和系统测试人员提供查看系统文档的功能性“入口”

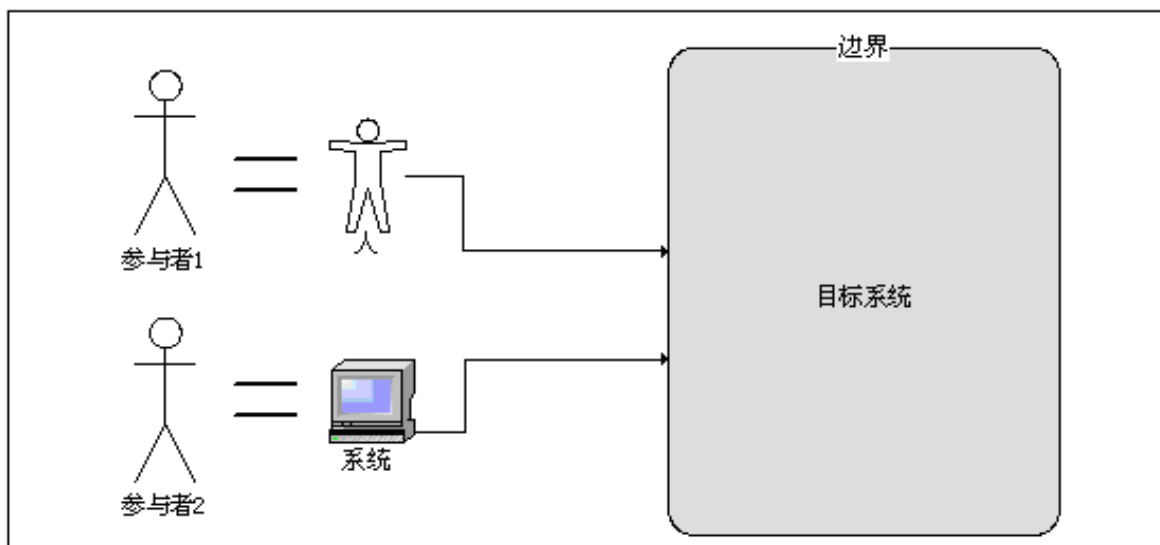
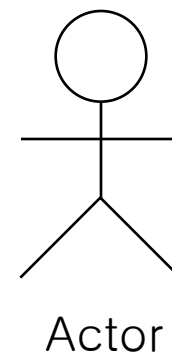
基本概念

- 用例模型的构成
 - Actor: 参与者
 - 系统边界
 - UseCase: 用例
 - 通信关系



基本概念

- Actor: 参与者
 - 与系统交互的任何人或事物
 - 处于系统的外部



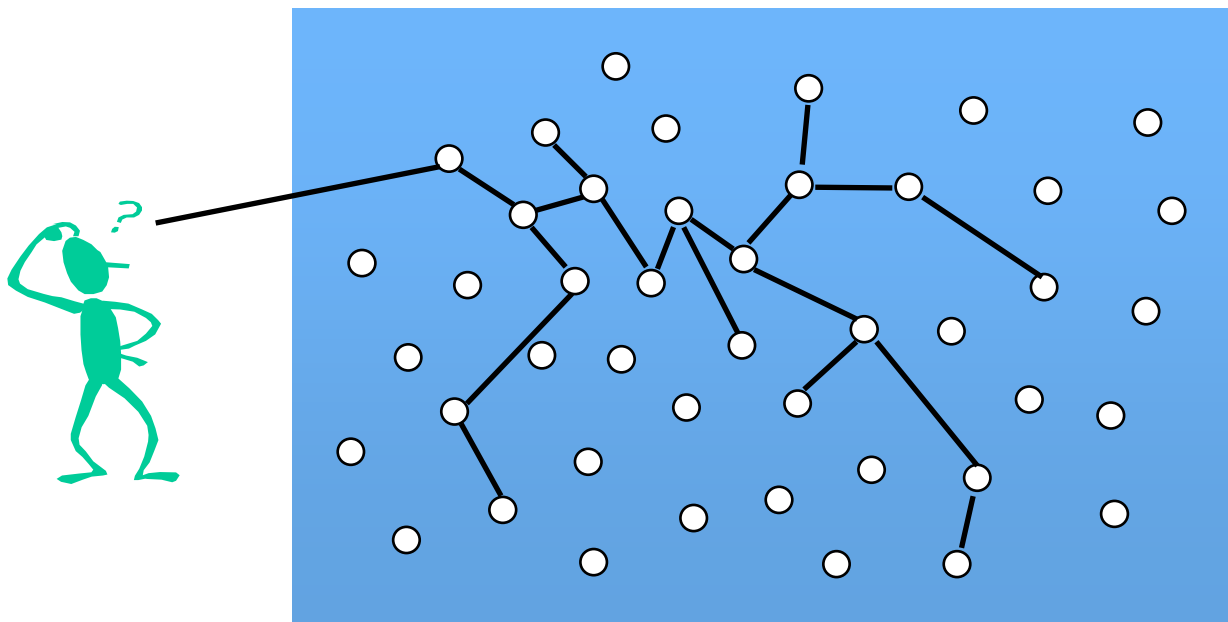
基本概念

- 系统边界
 - 界定系统的范围
 - 规定了
 - 什么是系统的组成部分
 - 什么是系统的外部
 - 边界的定位对功能需求有很大影响



概述

- **UseCase: 用例**
 - 从用户角度描述系统的行为
 - 描述要实现的行为，而不必说明如何实现



- UseCase: 用例

- 描述系统行为

- “how a system acts and reacts”
 - 系统外在的可观测行为

- 定义

“用例定义了一组用例实例，其中每个实例都是系统所执行的一系列操作，这些操作生成特定Actor可以观测的值。”

基本概念

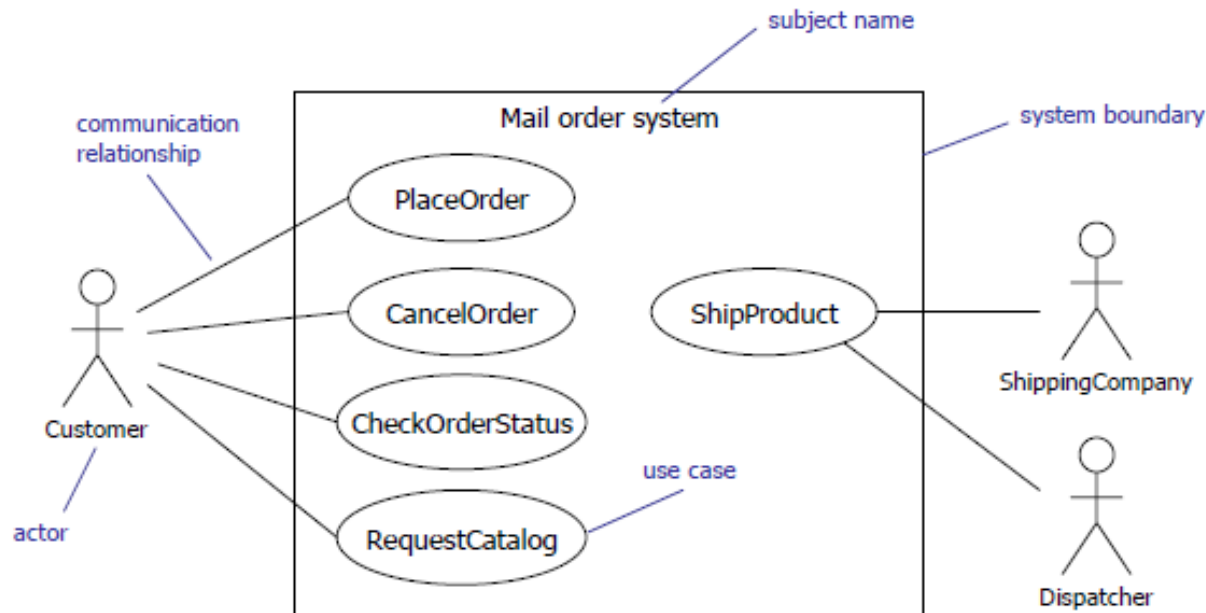
- UseCase: 用例
 - 系统与参与者交互所执行的动作序列
 - “参与者想要系统做的事”
 - 用例总是由参与者开始
 - 用例总是从参与者的角度书写

Use Case



基本概念

- 通信关系
 - 表示参与者与用例之间的关系
 - UML关联记号，表明参与者和用例的通信



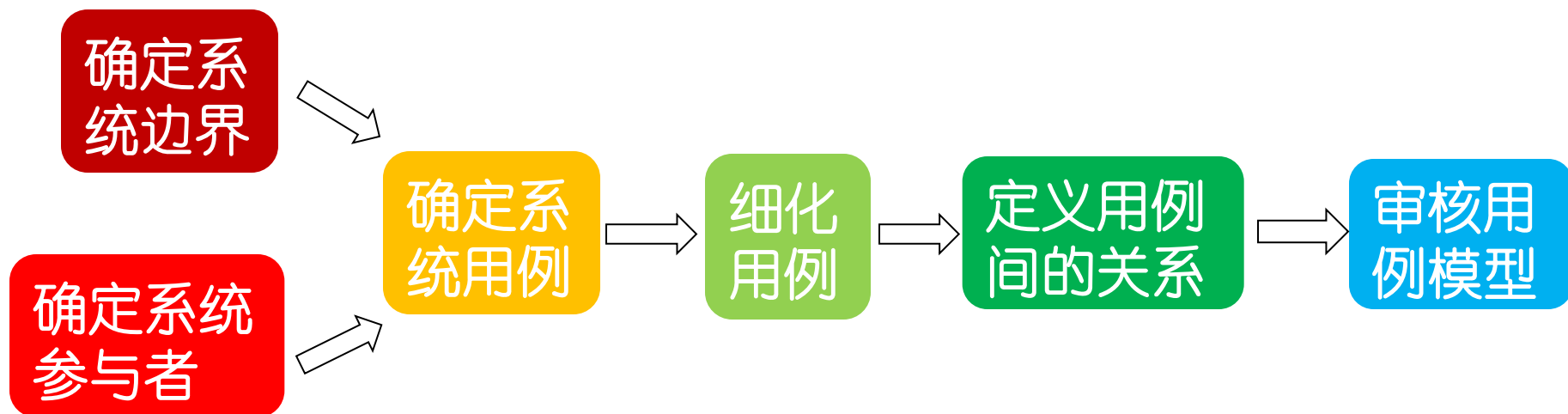
用例图



- 用例图
 - 用于对系统、子系统或类的行为进行可视化
 - 表示一组用例、参与者及它们之间的关系
 - 通常包含
 - 用况，参与者，通信关系，边界
 - 注解，约束
 - 包，用例的实例

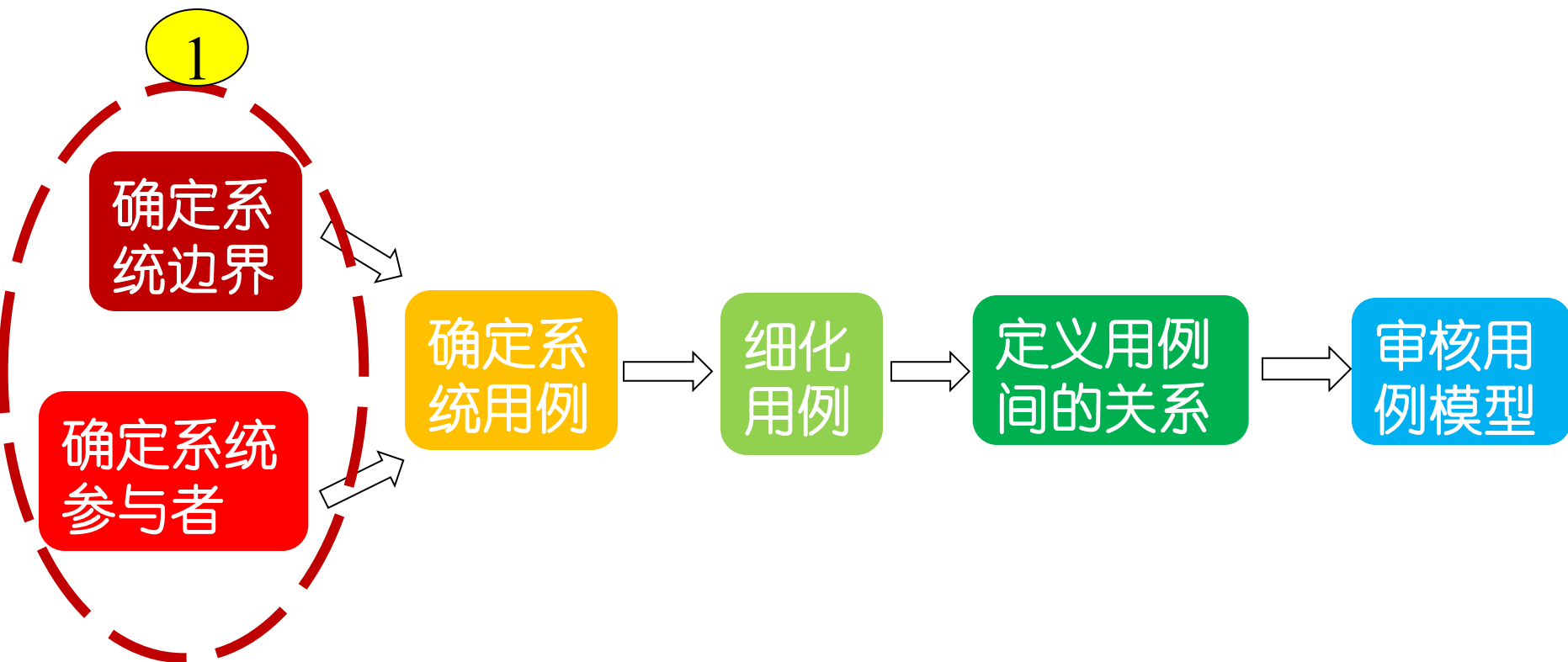
用例建模的步骤

- 用例建模的步骤



用例建模的步骤

- 用例建模的步骤



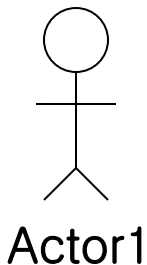
确定Actor——用例建模的步骤

- Actor参与者 ——(执行者、主角)
 - 是指系统以外的、需要使用系统或与系统交互的东西，包括人、设备、外部系统等。
- 例3：在一个银行业务系统中，可能会有以下一些Actor参与者。
- 客户， 管理人员， 厂商， mail系统

确定Actor——用例建模的步骤

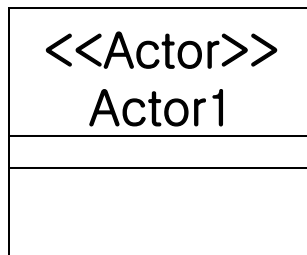
• Actor参与者的3种表现形式

人形图
标表示
参与者
是人

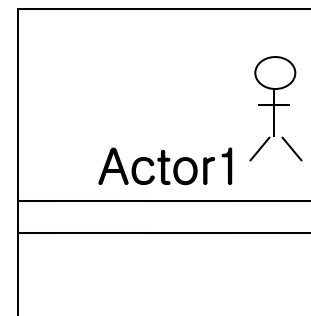


Icon形式

带版型标
记的类图
标表示参
与者是外
部系统



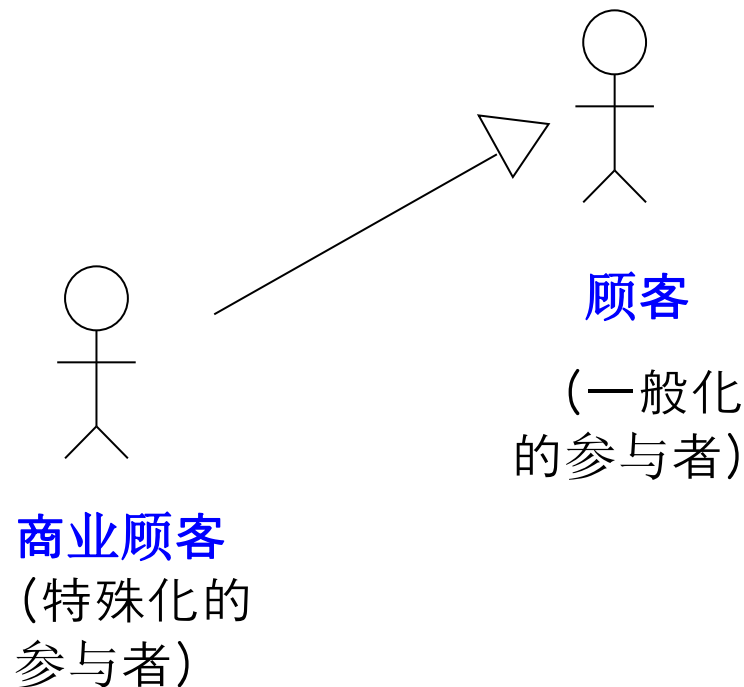
Label形式



Decoration形式

确定Actor——用例建模的步骤

- Actor参与者之间的继承（泛化）关系



确定Actor——用例建模的步骤



- 识别边界与参与者的要点

- Actor参与者

- 人、外部系统、外部因素、时间
 - 表示扮演的角色，不是特定的人或事物
 - 位于边界之外，不是系统的成分
 - 直接与系统交互

- 系统边界——责任边界而非物理边界

确定Actor——用例建模的步骤



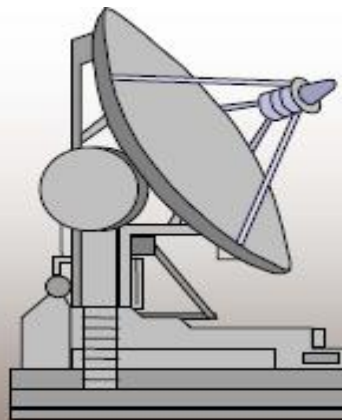
- Actor参与者
 - “在系统之外，透过系统边界与系统进行有意义交互的任何事物”
 - 用户；外部系统；外部硬件；时间触发器



人



外部系统



设备

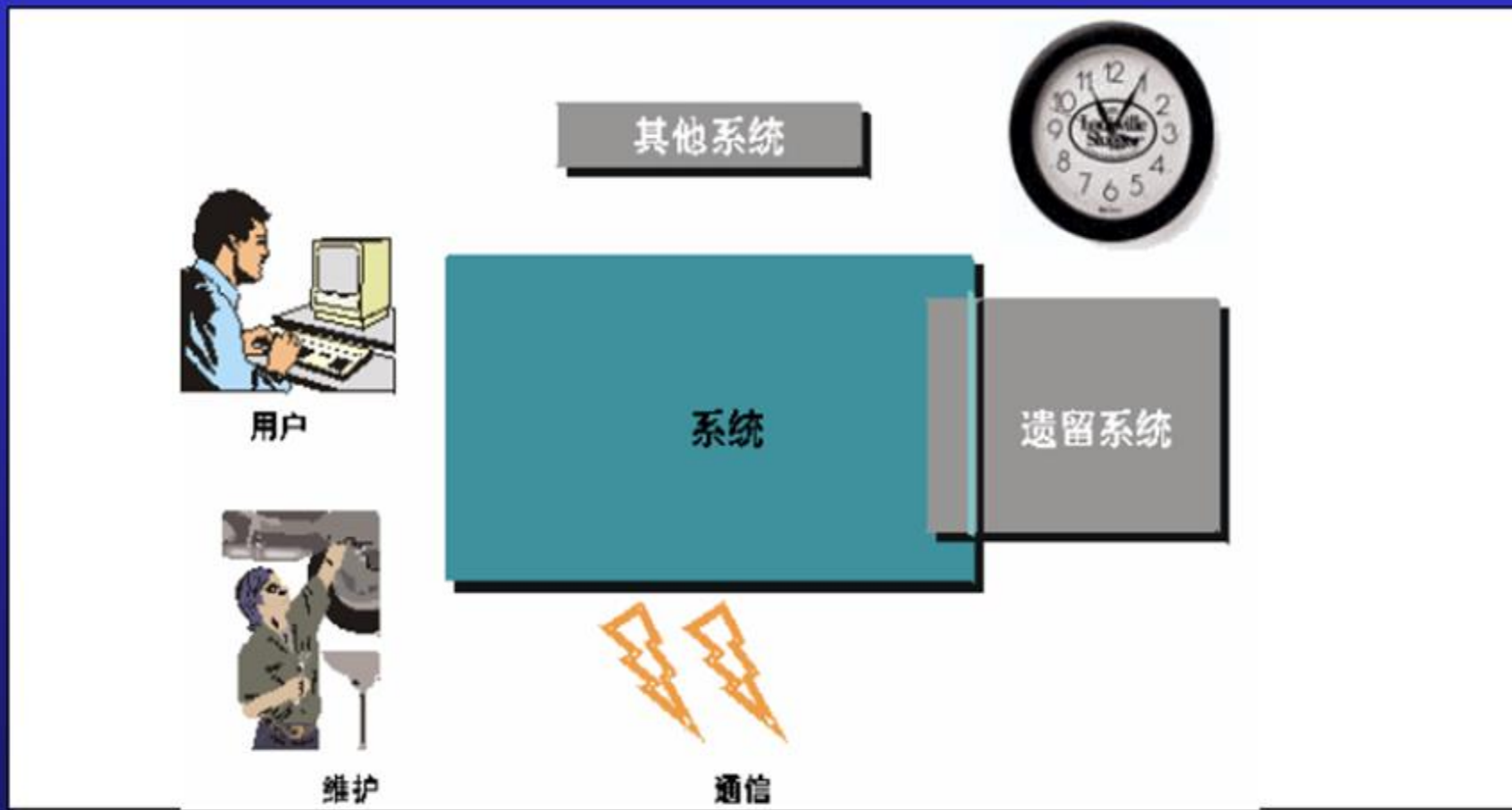


外部组织

确定Actor——用例建模的步骤



——任何事物



确定Actor——用例建模的步骤

Actor is
a ROLE



Bank
Customer

Not a specific
Person



Jane
Smith

Bruce
Jones

确定Actor——用例建模的步骤

——在系统外：不在里面



边界



类：顾客

确定Actor——用例建模的步骤

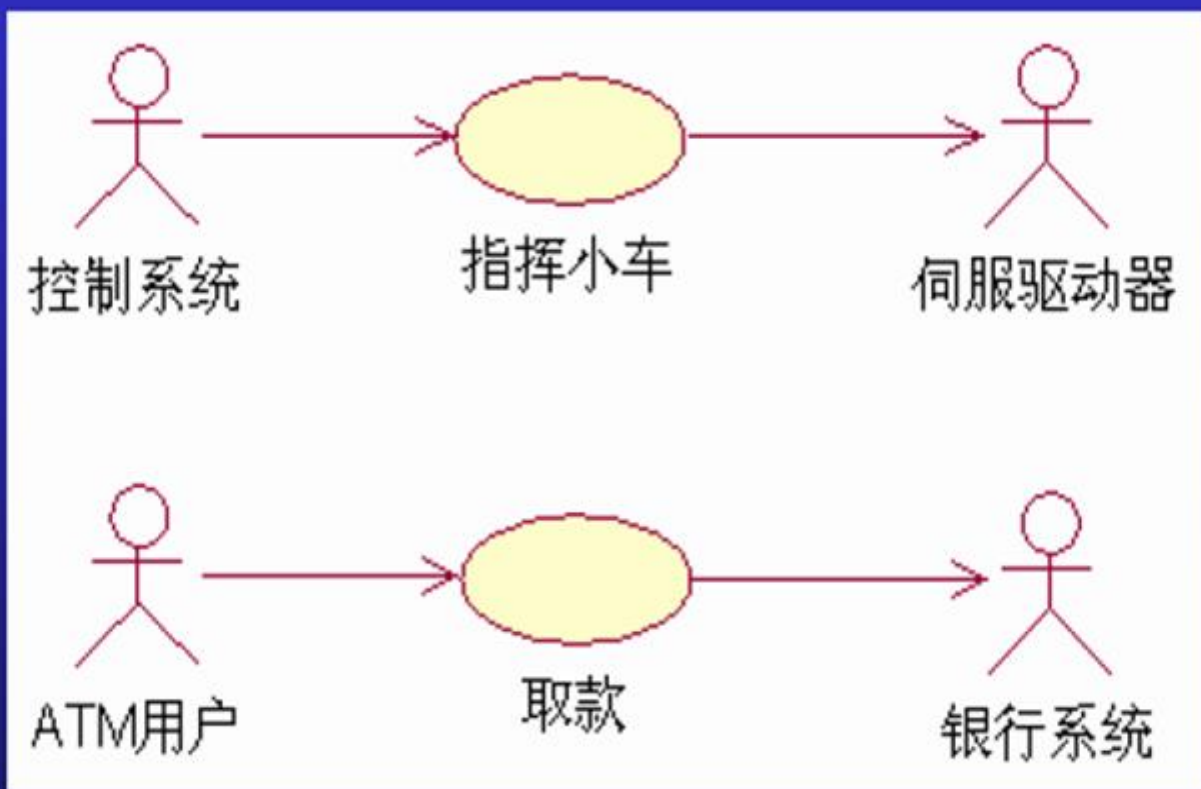
——直接与系统交互



那么，他算什么？

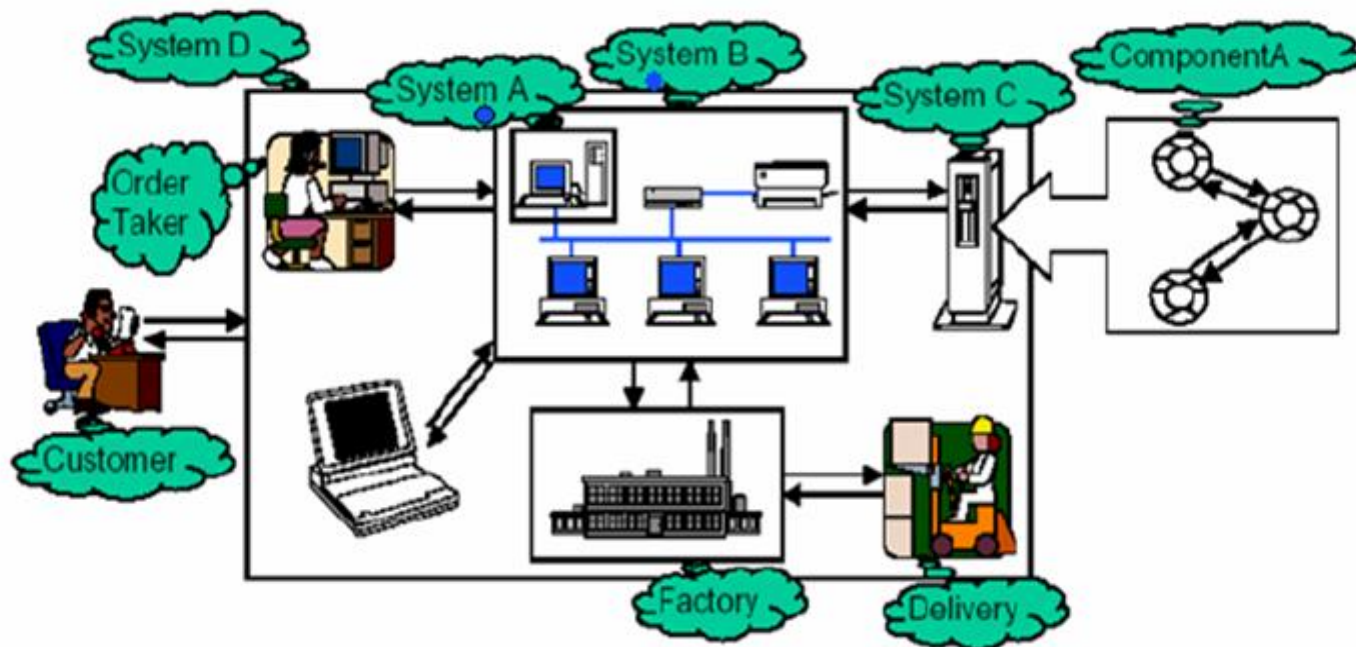
确定Actor——用例建模的步骤

——系统外：已经存在，无可选择



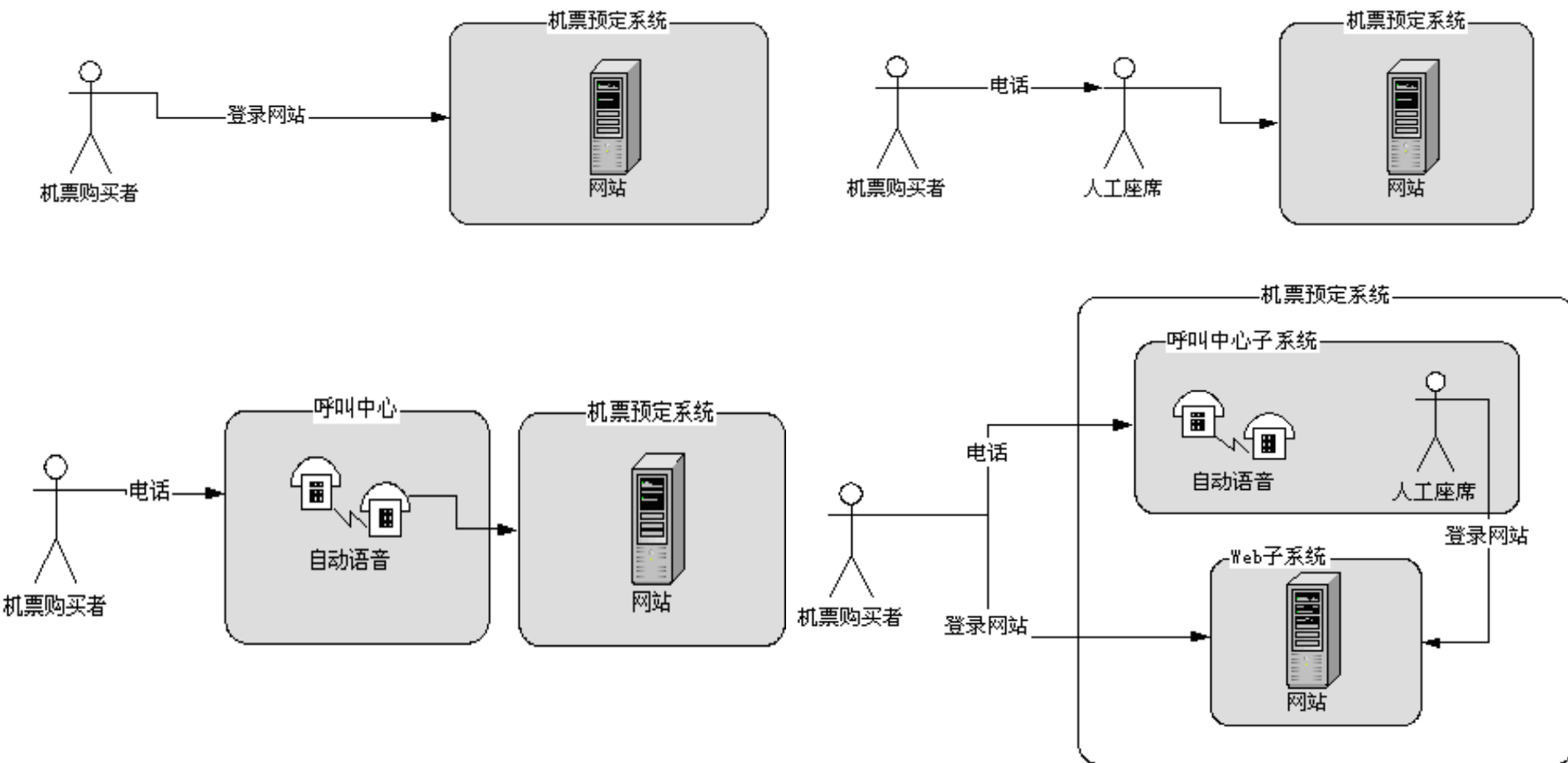
确定Actor——用例建模的步骤

——责任的边界，不是物理的边界



即使最终都是操作同一数据库同一张表，也可以有不同的系统

确定Actor——用例建模的步骤

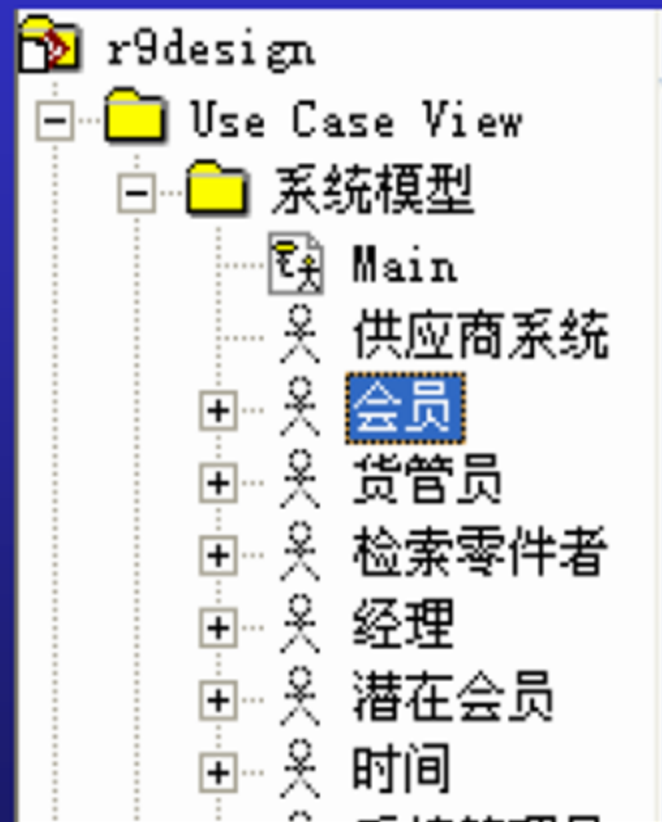


确定Actor——用例建模的步骤

——零件销售系统

- ❖ 谁使用系统的主要功能? — 潜在会员, 会员
- ❖ 谁改变系统的数据? — 会员, 货管员, 经理
- ❖ 谁从系统获得信息 — 潜在会员, 会员, 经理, 货管员
- ❖ 谁需要系统的支持以完成日常工作任务? — 经理, 货管员
- ❖ 谁负责维护、管理并保持系统正常运行? — 系统管理员
- ❖ 系统需要应付(处理)哪些硬设备? — 没有特殊硬设备
- ❖ 系统需要和哪些外部系统交互? — 可能与供应商的系统交互
- ❖ 谁(或什么)对系统运行产生的结果感兴趣? — 会员, 经理
- ❖ 有没有自动发生的事件? — 检查帐户

确定Actor——用例建模的步骤



潜在会员



会员



货管员



系统管理员



经理



供应商系统



时间

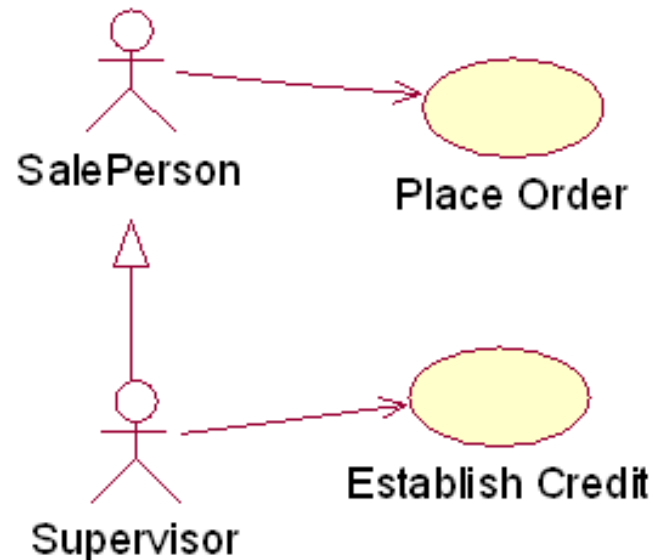
确定Actor——用例建模的步骤



- 参与者之间的关系

- 泛化 (Generalization)

- Generalization from A to B: an instance of A can communicate with the same kinds of use case instances as an instance of B.

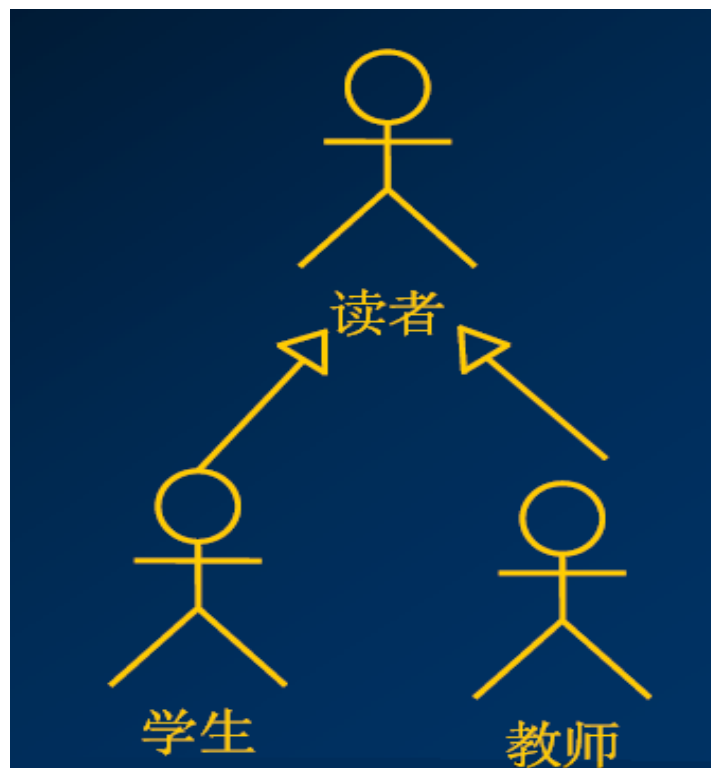
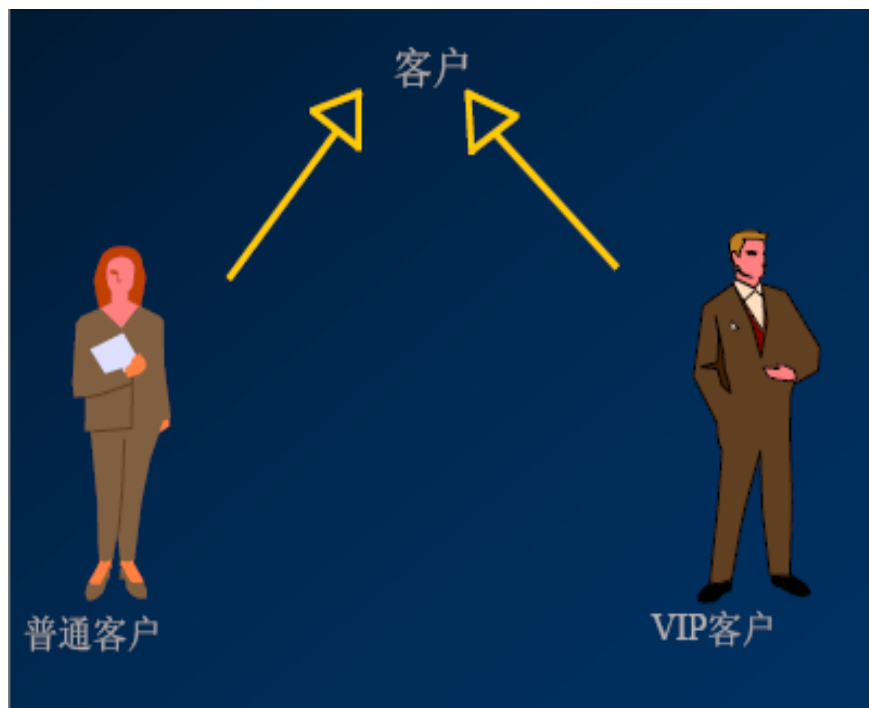


确定Actor——用例建模的步骤

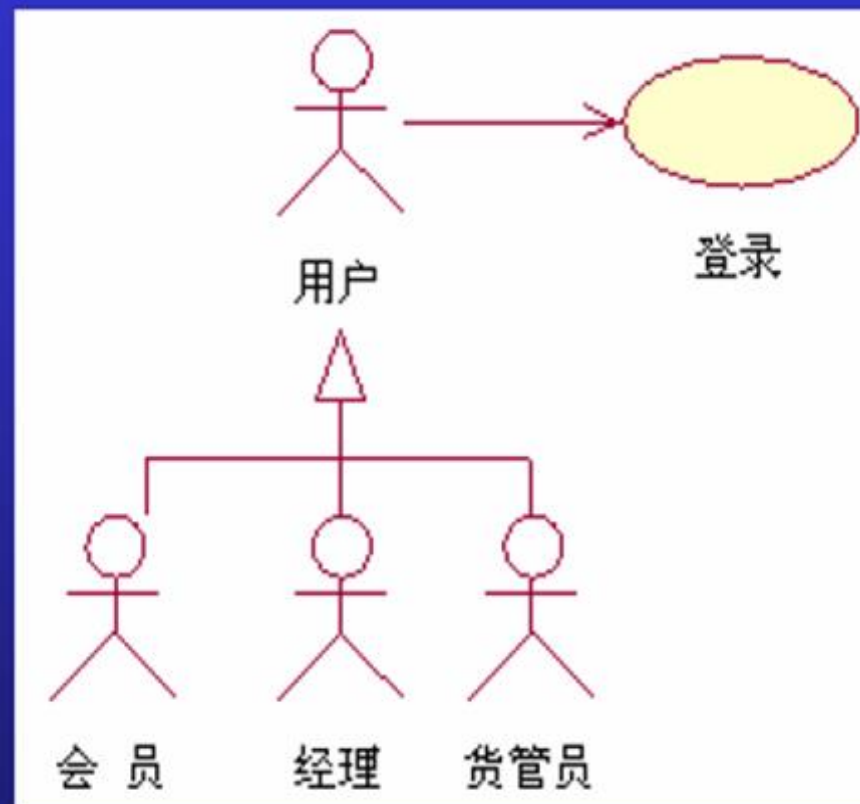
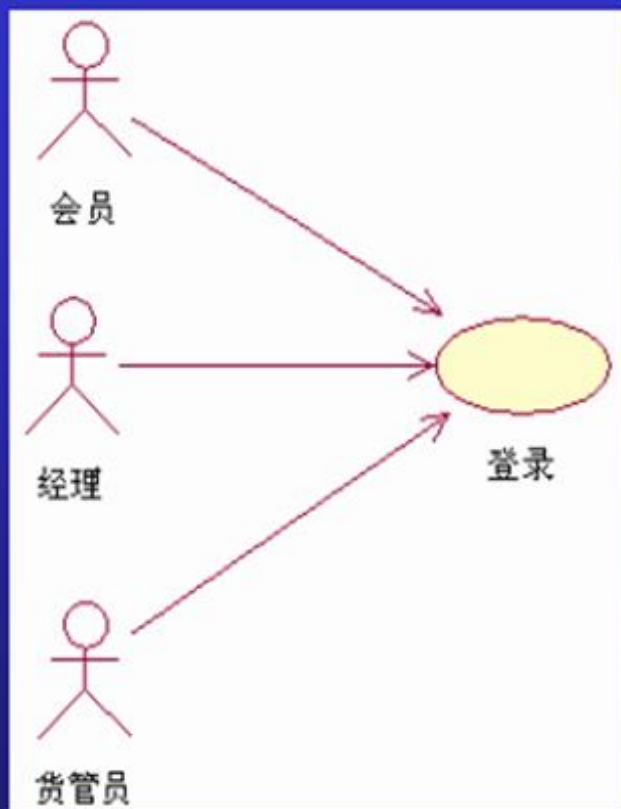


- 抽象Actor参与者

- 表示两个或者多个参与者共享的或者相同行为的角色



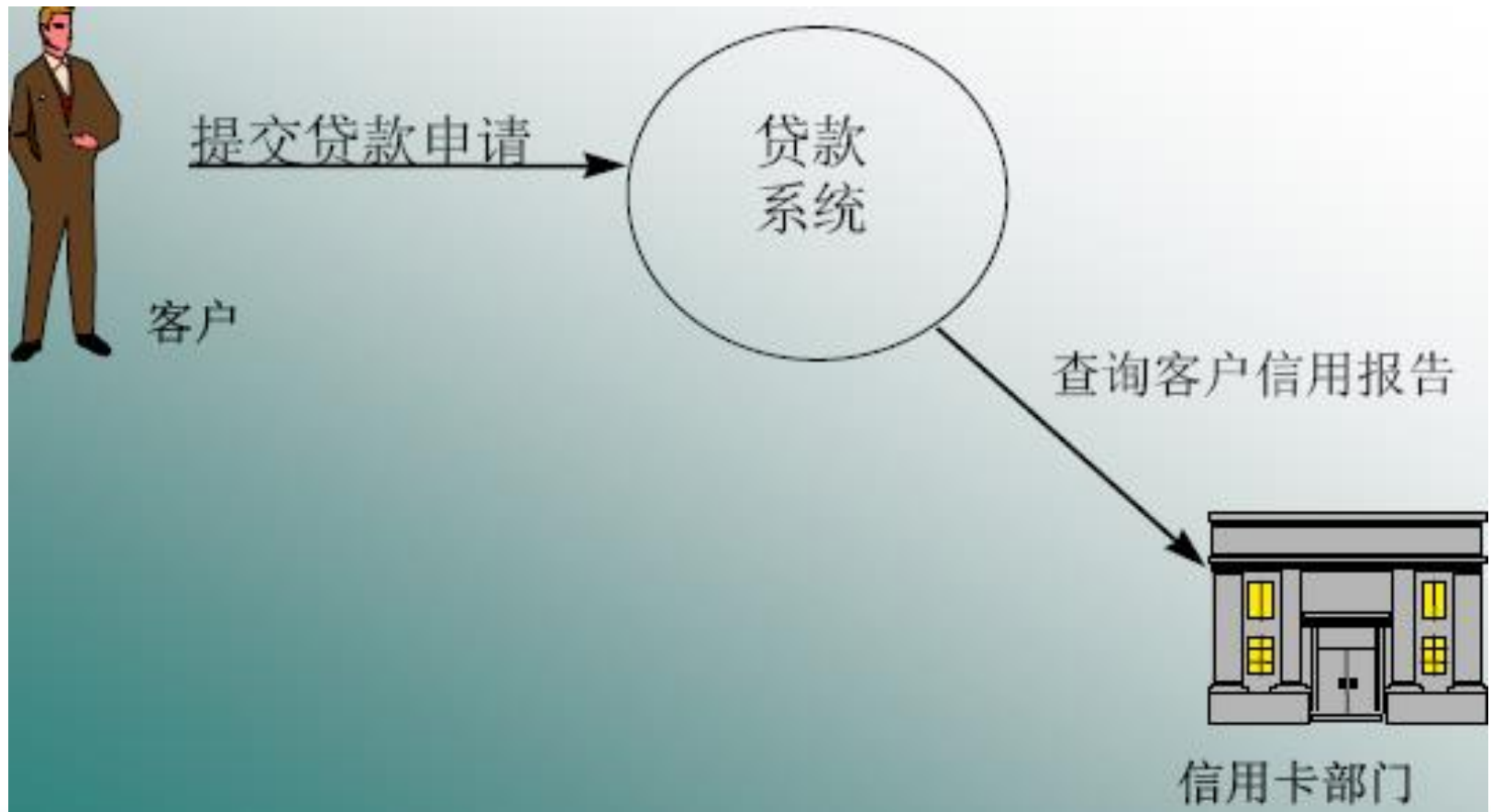
确定Actor——用例建模的步骤



责任类似或重叠——泛化出一个抽象执行者

主动Actor vs. 被动Actor

- 发起者——触发系统行为的外部Actor
- 外部服务和接收者——支持系统行为的Actor

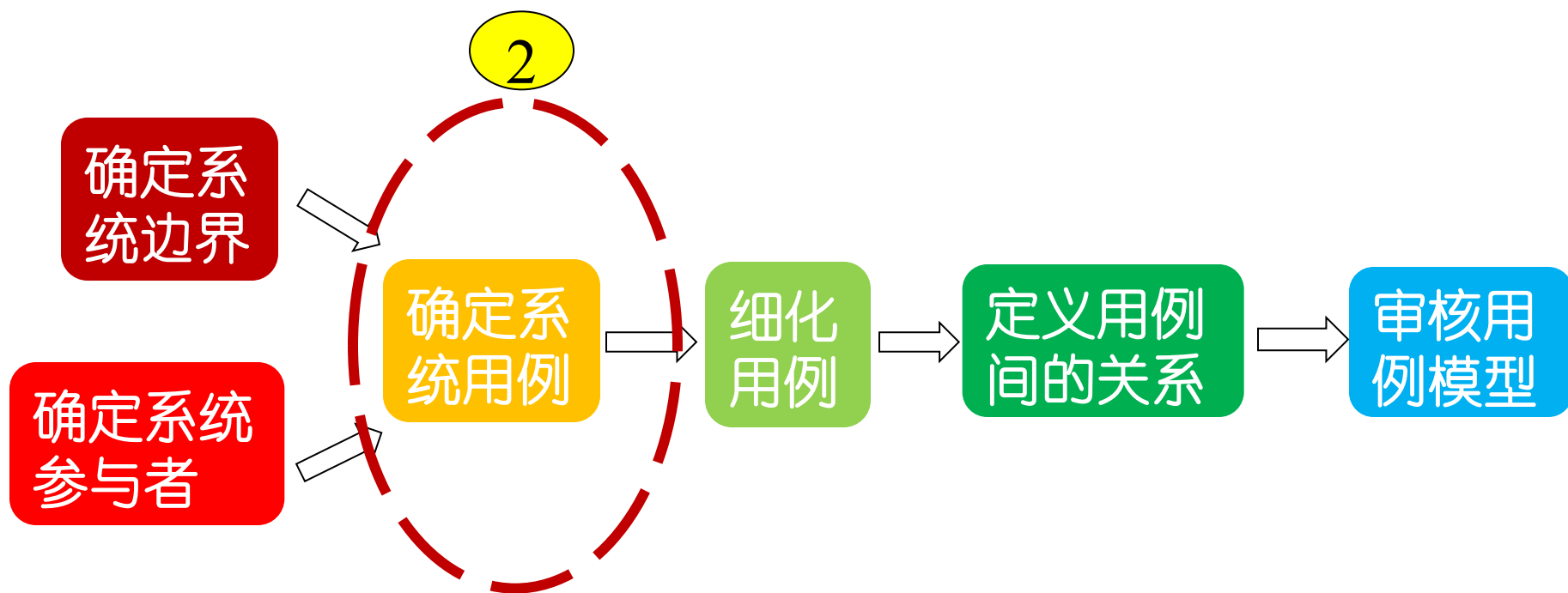


确定Actor——用例建模的步骤

- 主要和次要参与者
 - 主要参与者(primary actor): 是从系统获得可度量价值的用户。主要参与者的需求驱动了用例所表示的行为或功能。如果他们的需求或角色发生了变化, 那么系统将必须有重大的修改。
 - 对系统有某种目标
 - 从系统获得某种可测量的价值
 - 次要参与者(secondary actor): 在用例中提供服务, 并且不能脱离主要参与者而存在。
 - 系统对其有某种目标
 - 为其他actor创造价值

用例建模的步骤

- 用例建模的步骤



确定UseCase——用例建模的步骤



- UseCase: 用例

- 描述系统行为

- “how a system acts and reacts”
 - 系统外在的可观测行为

- 定义

“用例定义了一组用例实例，其中每个实例都是系统所执行的一系列操作，这些操作生成特定Actor可以观测的值。”

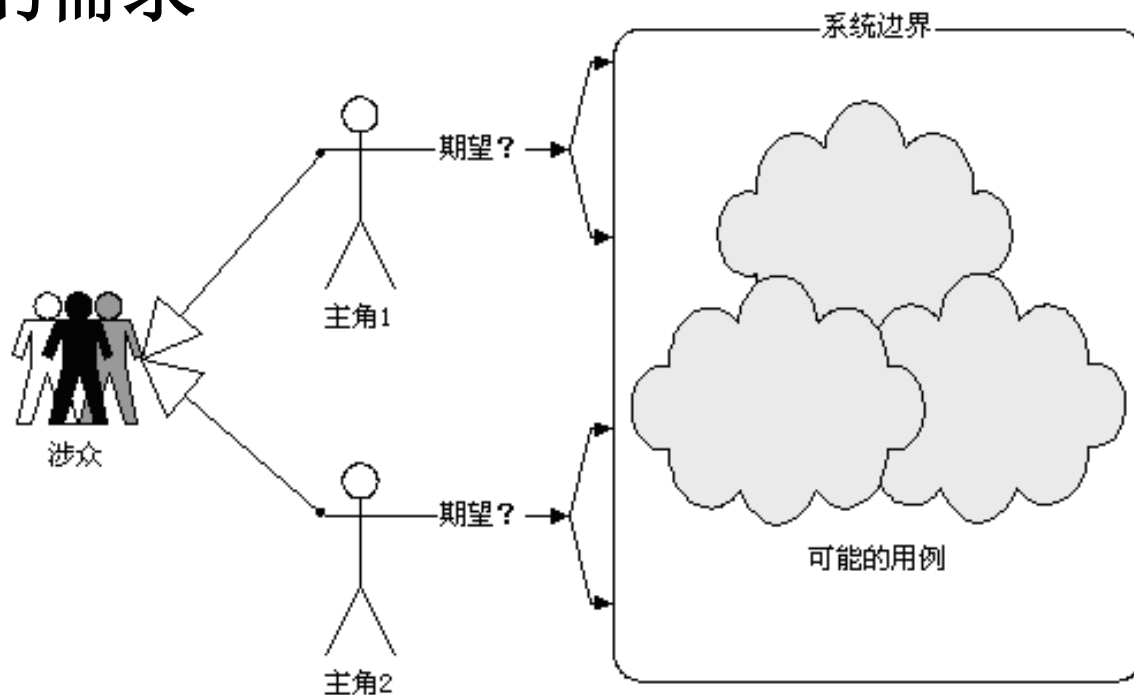
确定UseCase——用例建模的步骤

- 用例 (Use Case) 的**定义**:
 - **定义1**
 - 用例是对一个**参与者 (Actor)** 使用系统的一项功能时所进行的交互过程的一个文字描述序列。
 - **定义2**
 - 用例是系统、子系统或类和外部的**参与者 (Actor)** 交互的动作序列的说明，包括可选的动作序列和会出现异常的动作序列。
 - **定义3**
 - 用例是对一个系统或一个应用的一种单一的使用方式所作的描述，是关于单个**参与者 (Actor)** 在与系统对话中所执行的处理行为的陈述序列。
 - **定义4**
 - 用例是用来描述**参与者 (Actor)** 使用系统完成某个事件时的事情发生顺序。

确定UseCase——用例建模的步骤



- 确定系统用例
 - 查找用例的最佳方法：考虑各个参与者对系统的需求



确定UseCase——用例建模的步骤



- 确定系统用例

- 考虑如下问题

- 此参与者希望系统执行的主要任务是什么？
 - 此参与者是否将在系统中创建、存储、更改、删除或读取数据？
 - 此参与者是否需要将突发变更或外部变更通知给系统？
 - 是否需要将系统中发生的某些特定事件通知给此参与者？
 - 此参与者是否将执行系统启动或关闭操作？

确定UseCase——用例建模的步骤

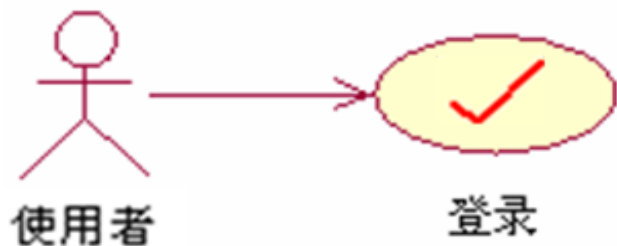


- 用例的特征

- 相对独立，结果有意义

- 从“功能”上是完备的

- 不能完整达到参与者需求的不能称为用例



确定UseCase——用例建模的步骤



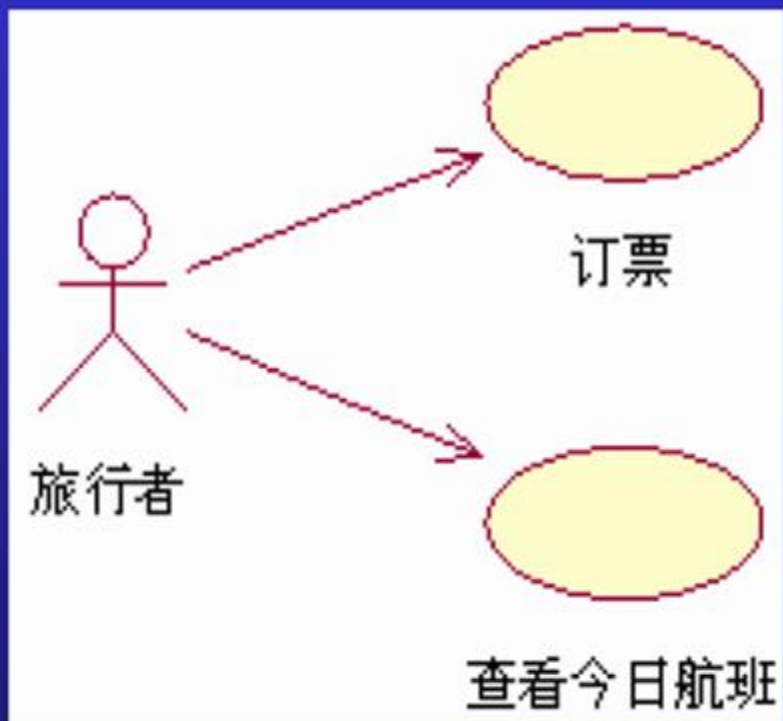
- 用例的特征
 - 执行结果可见
 - 结果对参与者是可观测的
 - 采用业务语言
 - 从用户观点说明



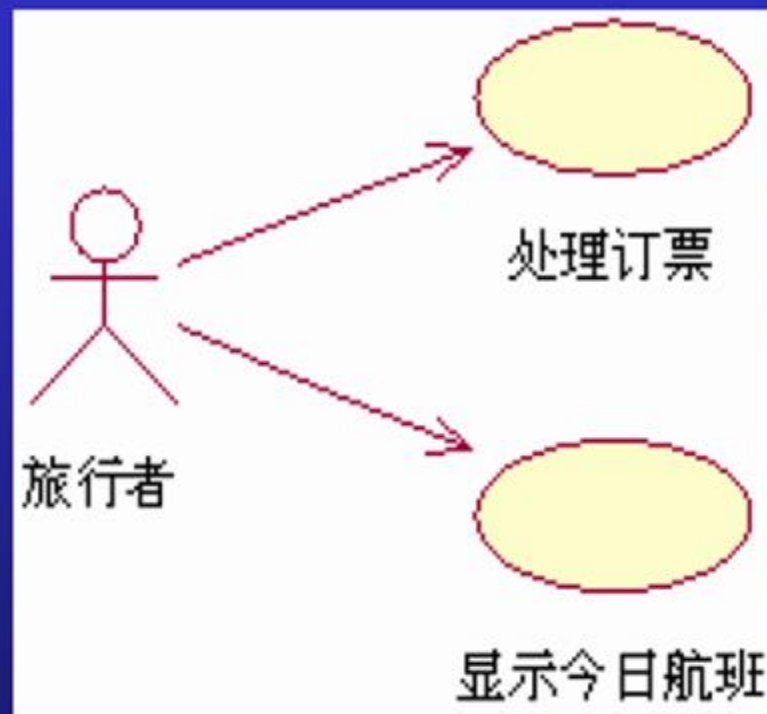
确定UseCase——用例建模的步骤



——用户观点而非系统观点



用户观点

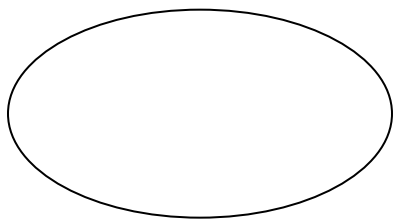


系统观点

确定UseCase——用例建模的步骤

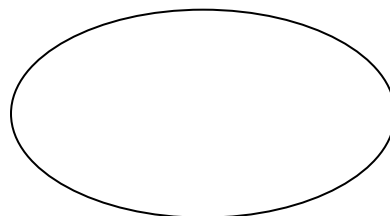
- 用例用一个椭圆表示。
- 用例名用**动宾结构**或**主谓结构**命名。
 - 例1：在字处理程序中，“**置正文为黑体**”是一个用例。

Use Case



置正文为黑体

较简单的用例



创建索引

较复杂的用例

为以后逐层细化具体的实现用例作铺垫

确定UseCase——用例建模的步骤



• 用例名的特征

— 以动宾短语形式出现

- 必须有一个动作和动作的受体
- 慎用弱动词和弱名词

❖ 弱动词：进行、使用、复制、加载、重复...

❖ 弱名词：数据、报表、表格、表单、系统...



确定UseCase——用例建模的步骤



- 用例的粒度

- 粒度的影响

- 粒度过小：需求过于众多和琐碎而无法控制
 - 粒度过大：需求过于模糊而容易忽略细节

- 粒度选择以能完成参与者的完整目的为依据

- 同一阶段的所有用例粒度应该是同一量级

- 粒度选择的本质：边界认定的不同

确定UseCase——用例建模的步骤

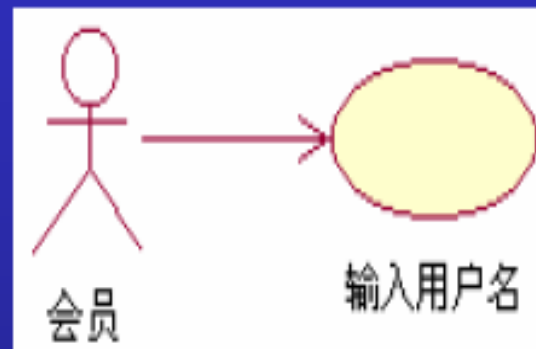


- 用例粒度的误区
 - 分不清目标和步骤
 - 导致不准确的需求获取
 - 用例的粒度过于细小
 - 同一需求阶段中的用例粒度大小不一
 - 导致需求难以理解；程序结构混乱
 - 产生本质：建模者心中没有清楚的边界

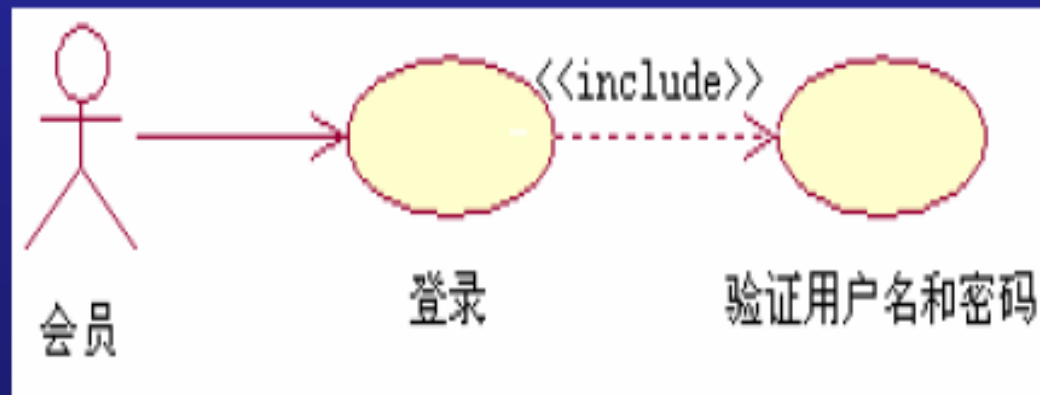
确定UseCase——用例建模的步骤

❖ 最常犯错误——把步骤当作用例

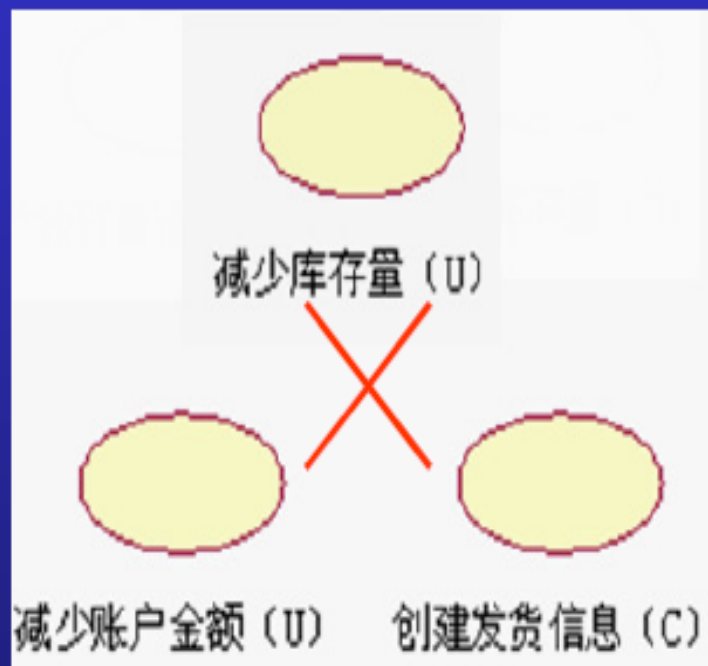
❖ 把执行者动作当作用例



❖ 把系统活动当作用例

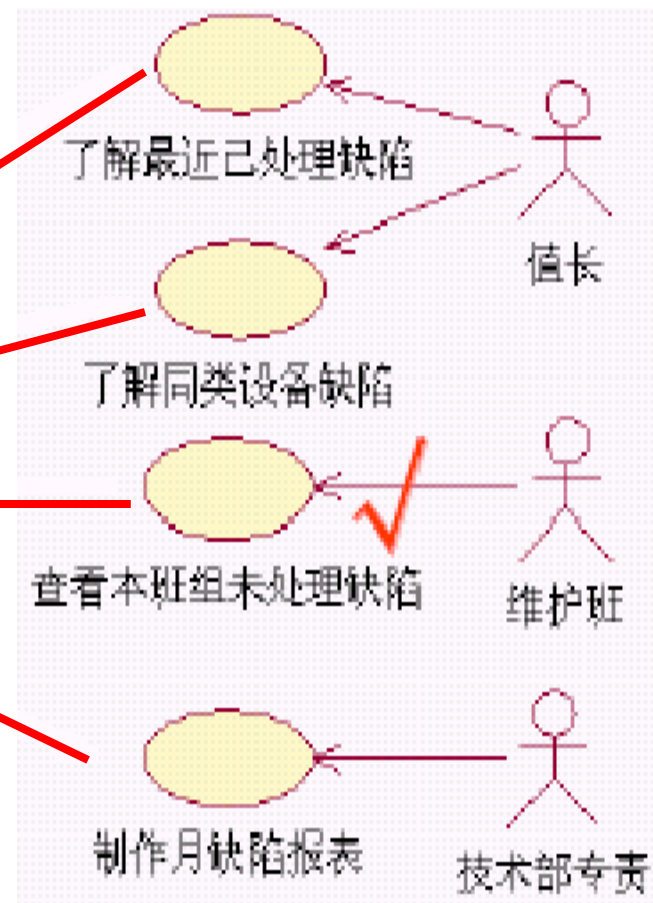
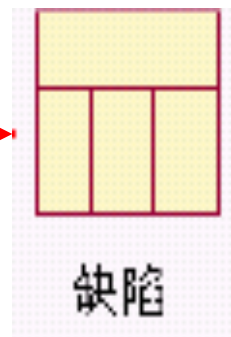
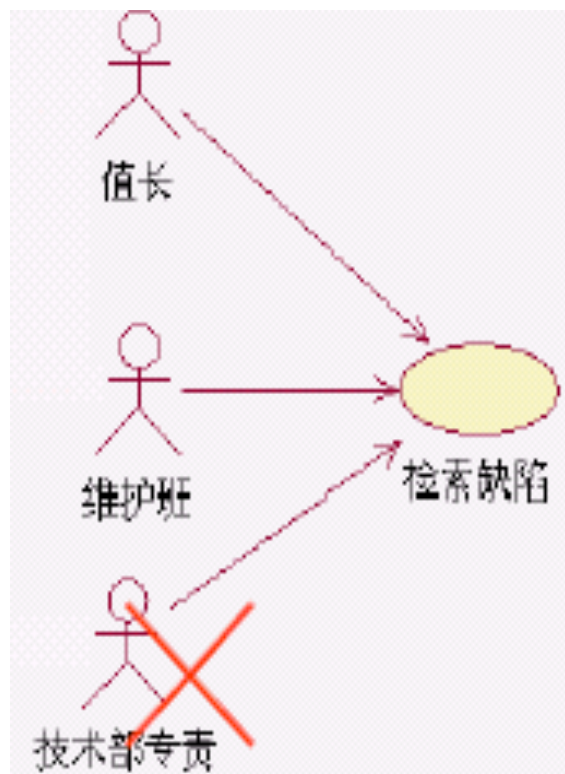


确定UseCase——用例建模的步骤



一个用例背后可能隐藏着许多数据操作

确定UseCase——用例建模的步骤



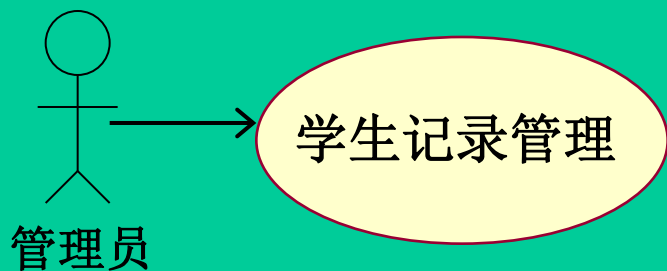
确定UseCase——用例建模的步骤

- 寻找用例时常见的一些**主要问题**：
- **用例的粒度问题**
 - 对于一个系统来说，不同的人进行用例分析后得到的用例数目有多有少
 - 如果用例粒度大，则得到的用例数就会很少
 - 如果用例粒度小，则得到的用例数就会很多
- 对于一个10人年的项目，Ivar Jacobson认为，20个左右用例是合适的
- 对于一个10人年的项目，Martin Fowler使用了100多个用例

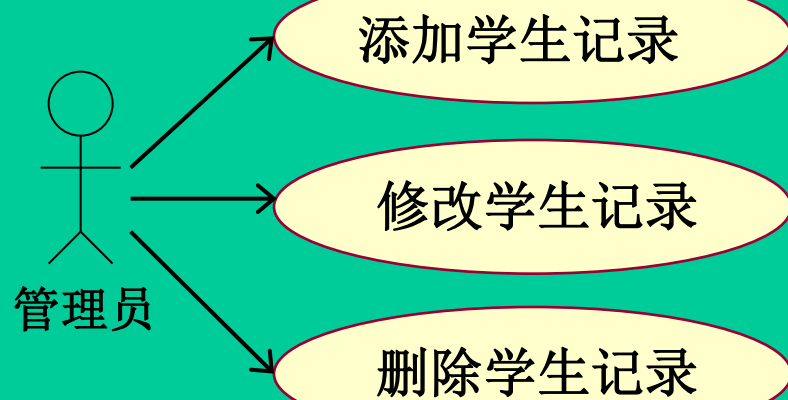
确定UseCase——用例建模的步骤

- 寻找用例时常见的一些**主要问题**:
- **用例的细分问题**
- 在一个系统中，有几个相似的功能。那么，是将它们放在同一个用例中，还是分成几个用例？
 - 例如，在学生档案管理系统中，管理员经常做3件事：增加一条学生信息，修改一条学生信息，删除一条学生信息。

方案一

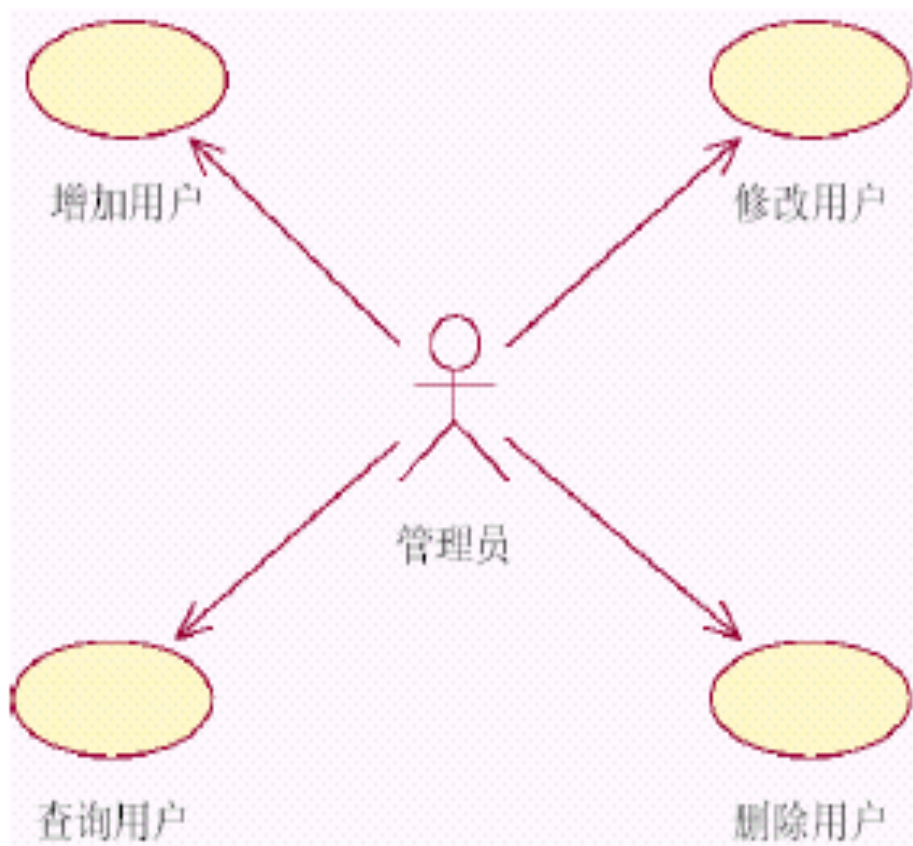


方案二

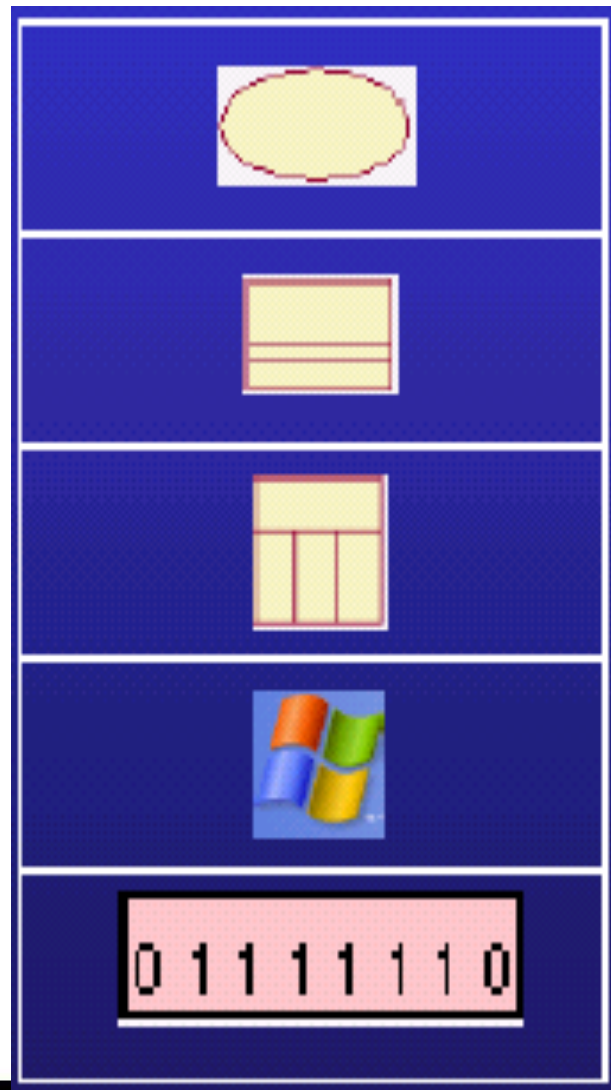


确定UseCase——用例建模的步骤

- 用例的粒度

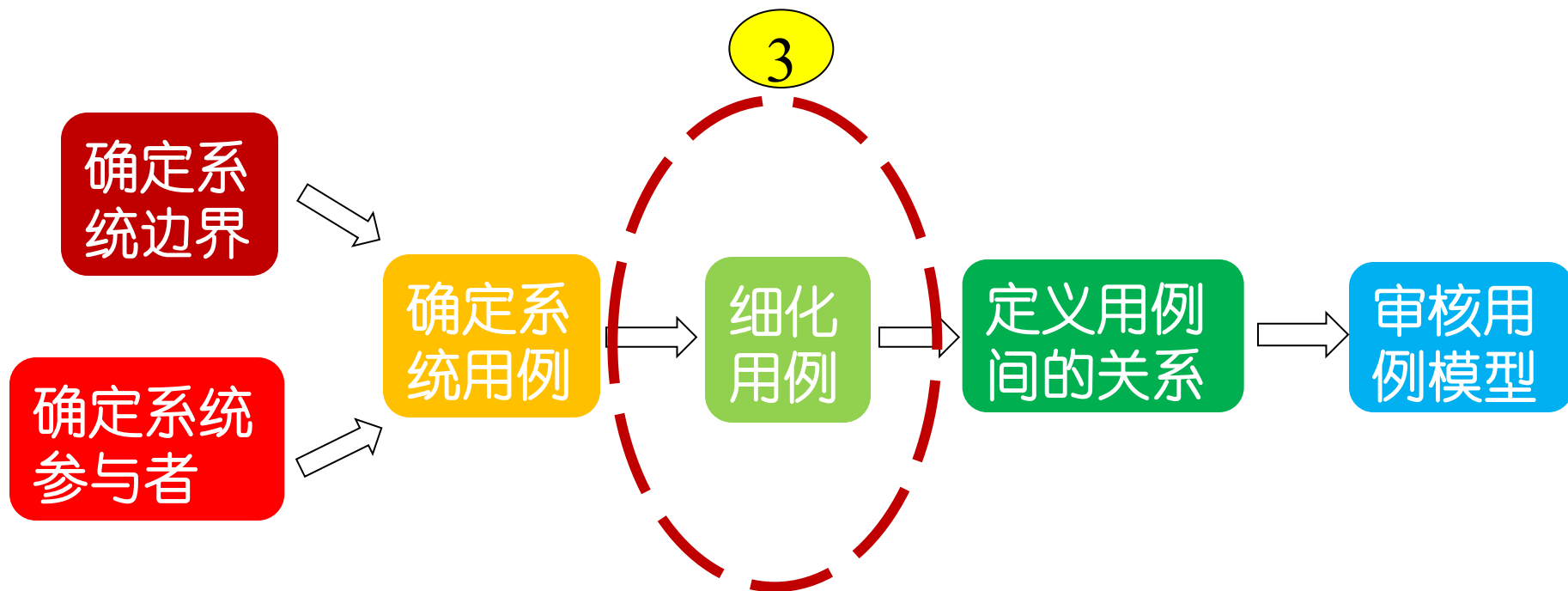


CRUD: 新增、读取、修改、删除



用例建模的步骤

- 用例建模的步骤



细化UseCase——用例建模的步骤

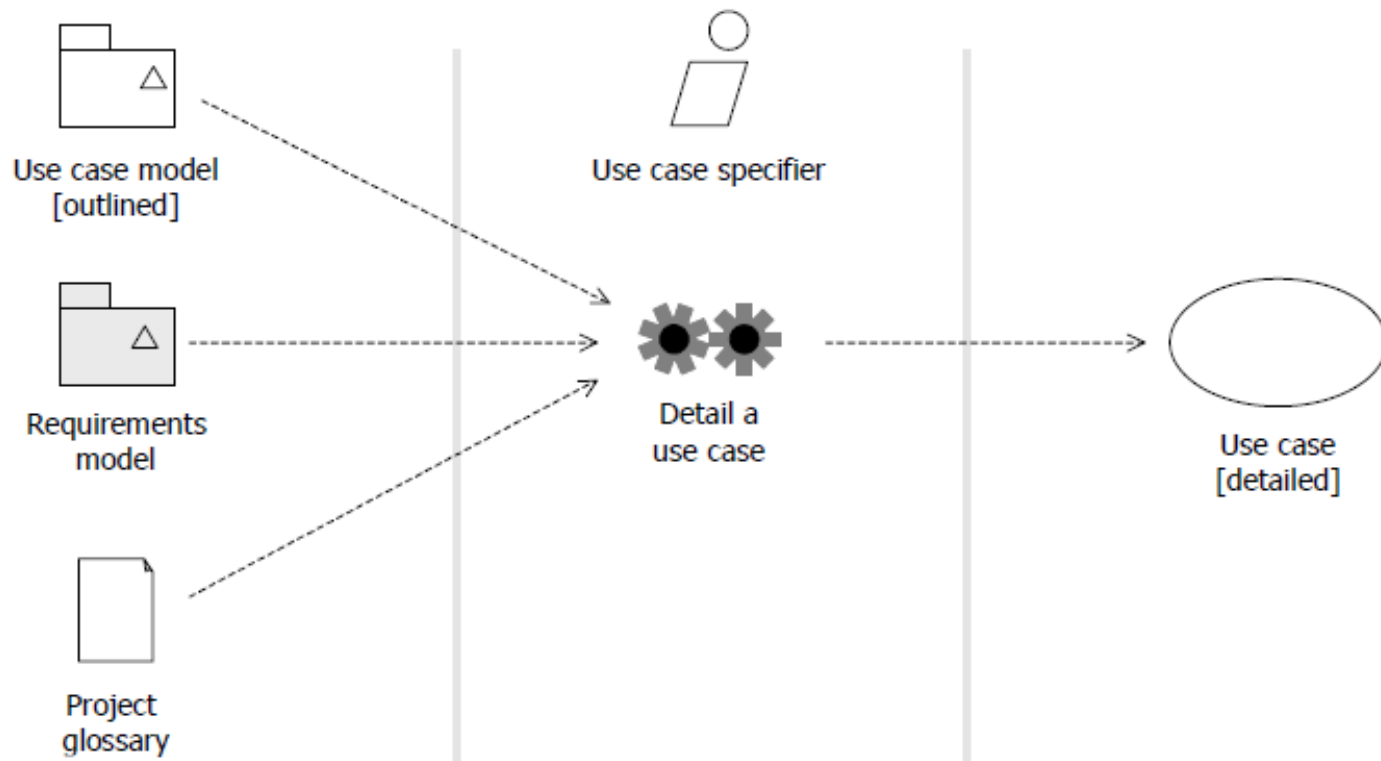


- 建立用例的参考原则
 - 用例是短文
 - 用例可以是一个场景，包括动作和互交
 - 用例可以是一组场景，描述不同场景下的行为
 - 用例里不要有系统设计
 - 用例里不要有界面设计
 - 用例里不要有测试
 - 用例应该描述行为需求
 - 用例的主场景最好不要超过九步
 - 用例的最大价值不在于主场景，而在于备选行为

细化UseCase——用例建模的步骤



- 细化用例
 - 输出：用例规格说明

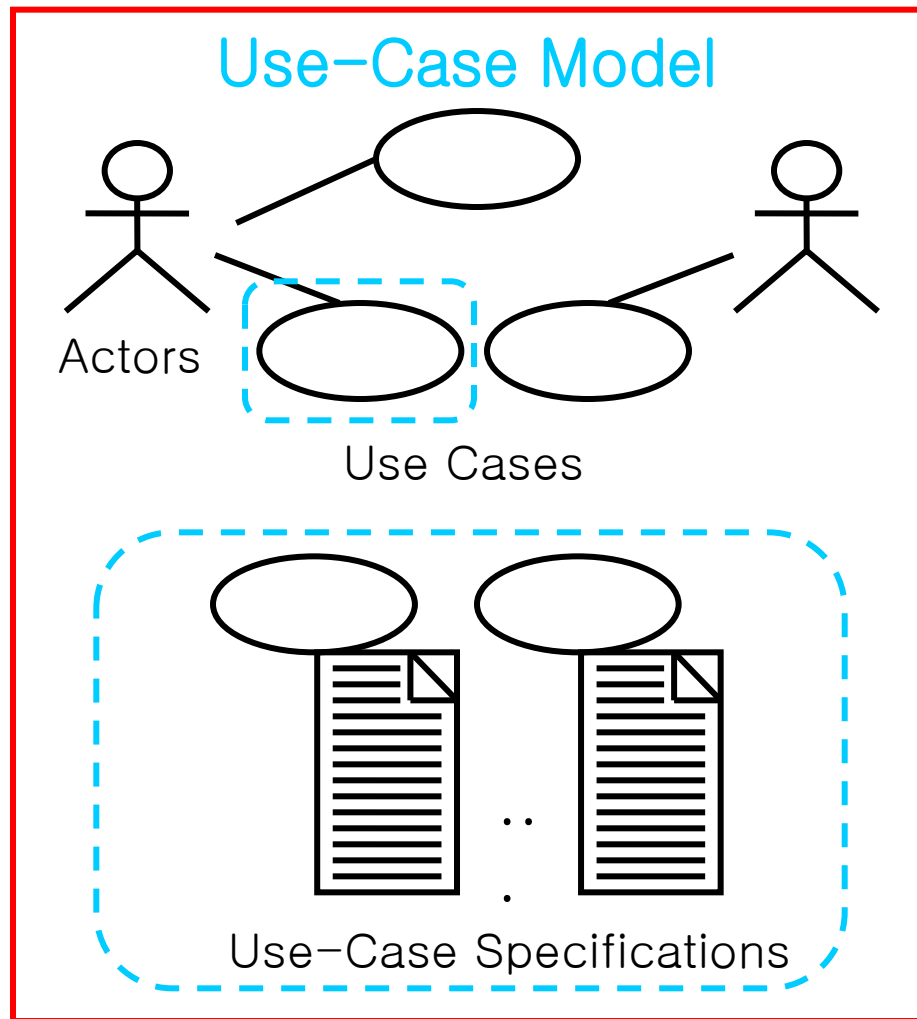


细化UseCase——用例建模的步骤



- 用例规格说明
 - 以充足的细节描述用例
 - 促进参与者代表或客户的理解
 - 使其作为软件开发的起点

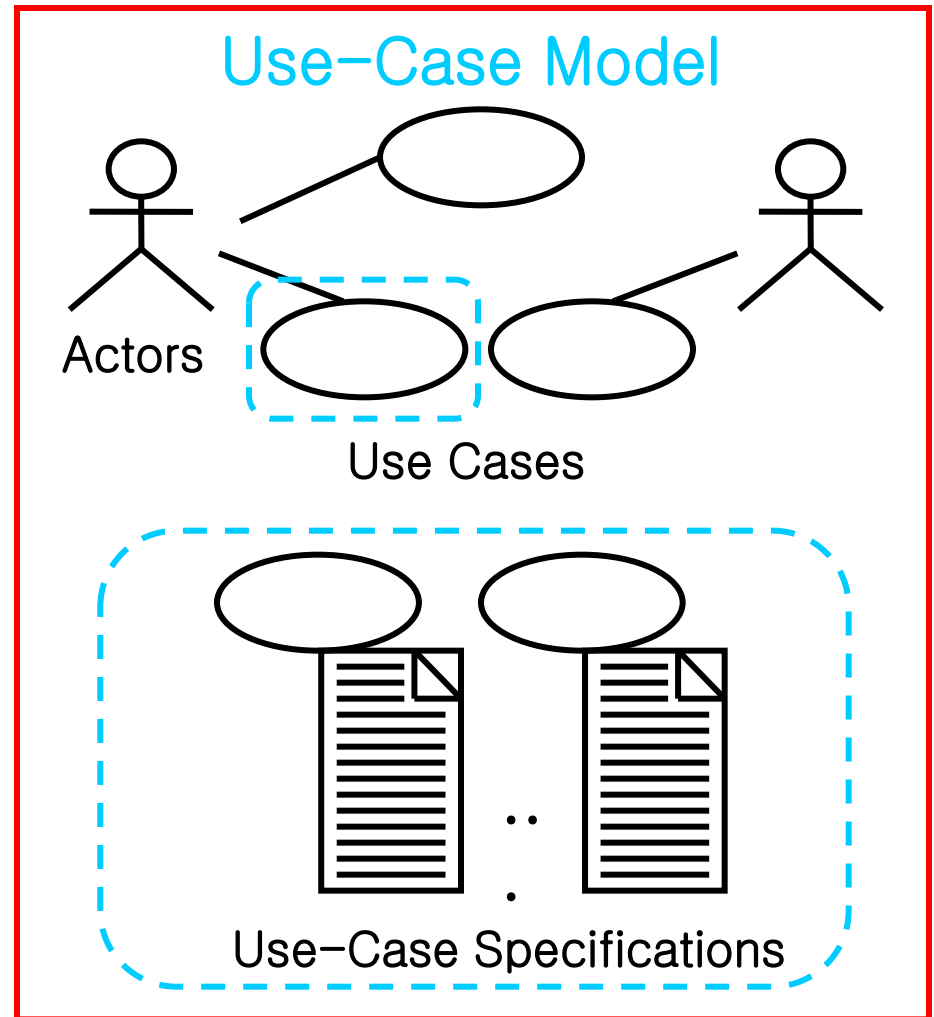
用例是文本文档，而非图形；用例建模主要是编写文档的活动，而非制图！



细化UseCase——用例建模的步骤

Use-Case Specifications

- Name
- Brief description
- Flow of Events
- Relationships
- Activity diagrams
- Use-Case diagrams
- Special requirements
- Pre-conditions
- Post-conditions
- Other diagrams



用例是文本文档，而非图形；用例建模主要是编写文档的活动，而非制图！

细化UseCase——用例建模的步骤



用例规格说明

use case name

use case identifier

brief description

the actors involved in the use case

the system state before the use case can begin

the actual steps of the use case

the system state when the use case has finished

alternative flows

Use case: PaySalesTax	
ID: 1	
Brief description:	Pay Sales Tax to the Tax Authority at the end of the business quarter.
Primary actors:	Time
Secondary actors:	TaxAuthority
Preconditions:	1. It is the end of the business quarter.
Main flow:	<div>implicit time actor</div> <div>1. The use case starts when it is the end of the business quarter.</div> <div>2. The system determines the amount of Sales Tax owed to the Tax Authority.</div> <div>3. The system sends an electronic payment to the Tax Authority.</div>
Postconditions:	1. The Tax Authority receives the correct amount of Sales Tax.
Alternative flows:	None.

细化UseCase——用例建模的步骤



用例规格说明

- 用例规格说明包含的信息
 - 用例名称
 - 简要描述
 - 涉及的参与者
 - 前置条件/后置条件
 - 事件流
 - 特殊需求
 -

用例规格说明：Example

User Management

- 用例描述 User Management
 - 1. Brief Description
 - This use case is for adding and deleting users. When adding user, the privilege of the user will be assigned.
 - 2. Basic Flow
 - B1 Start of use case
 - Actor chooses User Management in main screen.
 - B2 Prompt welcome screen
 - System shall prompt actor the user management welcome screen.
 - A1 Choose delete user
 - B3 Choose add user
 - Actor choose Add User.
 - B4 Prompt add user screen
 - System shall prompt actor the add user screen which needs actor to input some information. These information include : User Name, User ID, User Password, User Role.
 - B5 submit
 - Actor inputs the information and submits the screen.
 - E1 User ID already exists
 - E2 Not enough information
 - B6 Prompt success
 - System shall prompt actor success information.

用例规格说明：Example

User Management

- 3. Alternative Flows
 - A1 Choose Delete User
 - From B2 Prompt welcome screen
 - A1.1 Choose delete
 - actor chooses delete user
 - A1.2 Prompt delete user screen
 - System shall prompt actor the delete user screen.
 - In this screen, system shall lists all the user ID stored in system.
 - A1.3 Choose user to delete
 - Actor chooses the user ID to delete and submit this screen.
 - E3 No User ID chosen
 - A1.4 Prompt success
 - System shall prompt actor success information
- 4. Exception Flows
 - E1 User ID already exists
 - From B5 Submit, The User ID actor inputs has already existed in the system. The system shall prompt actor to try another User ID
 - E2 Not enough information
 - From B5 Submit, If actor does not input all the information, system can not add a user. The system shall prompt actor to input all the information.
 - E3 No user ID chosen
 - From A1.3 Choose user to delete, If actor does not choose a user to delete before submitting the screen, the system shall prompt actor an error that he/she should choose a user to delete.

细化UseCase——用例建模的步骤

用例规格说明

学生选课

- 举例

用例名称：学生选课

简要描述：完成一次学生选课的完整过程

参与者：学生

过程描述：

- (1) 学生输入ID，系统识别ID的有效性；
- (2) 系统对学生进行注册识别；
- (3) 学生浏览本学期预开课程；
- (4) 学生选择自己要上的课程并确认；
- (5) 系统给出所选课程列表及相应学分合计。

细化UseCase——用例建模的步骤



用例规格说明

学生选课

- 举例

.....

异常事件流处理:

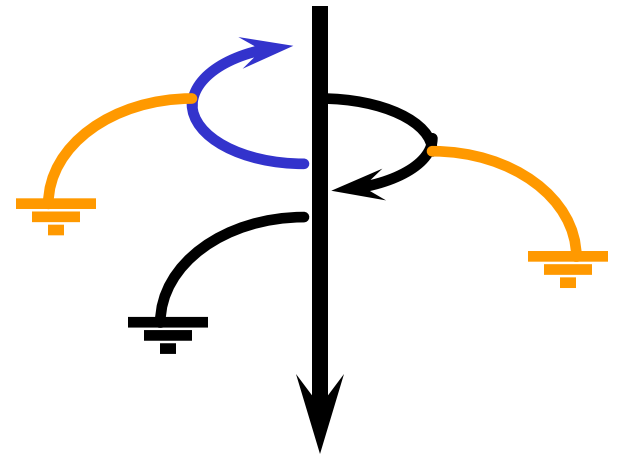
- (1) 标识码有效性检查失败, 允许学生重新输入 (3次机会);
- (2) 注册识别失败, 没有注册 (尚未交学费) 的学生不能选课;
- (3) 选择课程确认失败, 所选几门课程中在上课时间上发生冲突时, 系统提示重选。

细化UseCase——用例建模的步骤



用例规格说明

- 事件流
 - 列出用例中的各个步骤
 - 包含
 - 用例何时开始，如何结束
 - 何时与参与者交互，交换什么对象
 - 该行为的基本流，可选流，和异常流



细化UseCase——用例建模的步骤

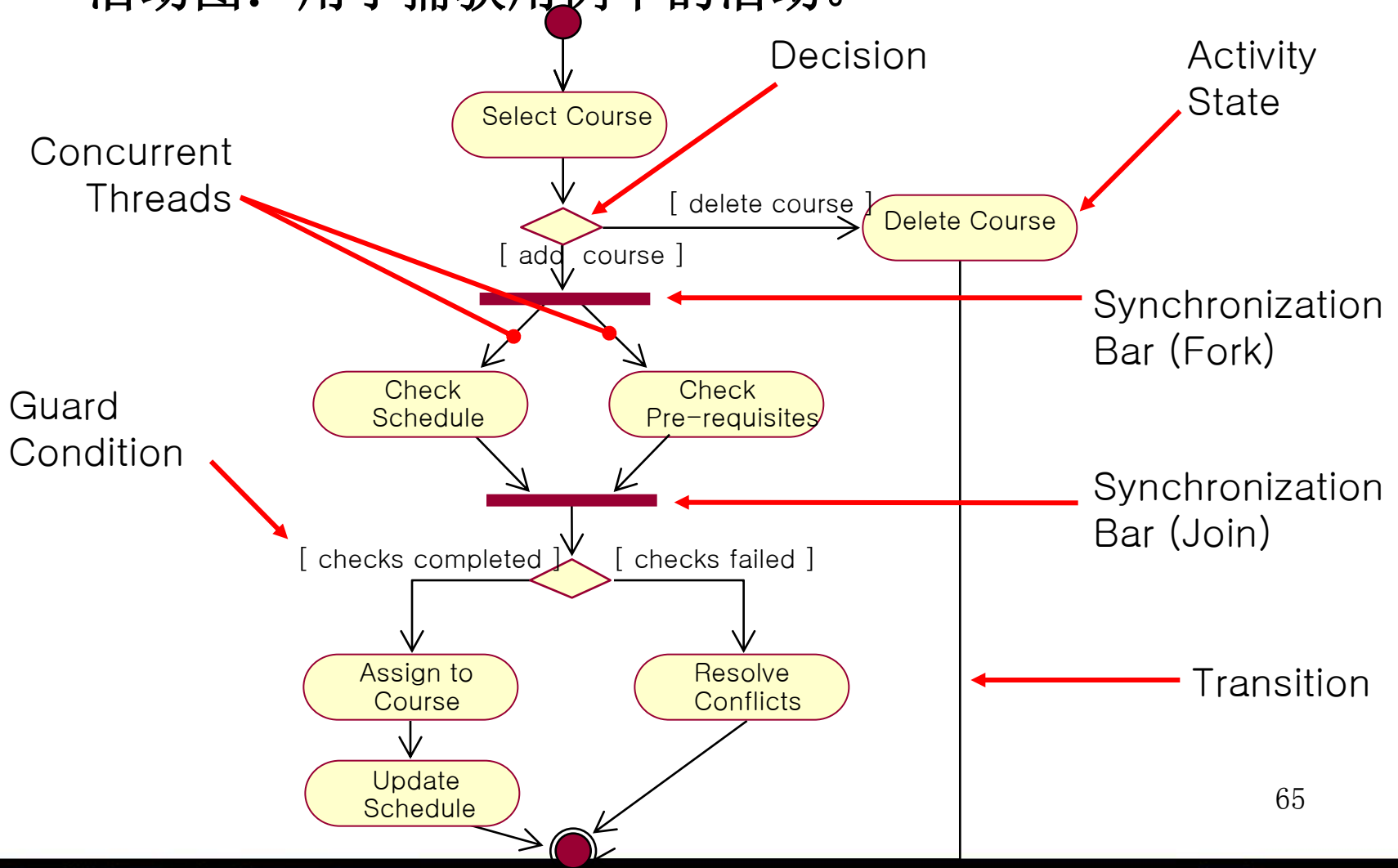


用例规格说明

- 事件流的描述方式
 - 非形式化的结构化文字
 - 形式化的结构化文字（使用前/后置条件）
 - 状态机（适用于反应式系统）
 - 活动图（适用于工作流）
 - 伪码

细化UseCase——用例建模的步骤

- 活动图：用于捕获用例中的活动。



细化UseCase——用例建模的步骤

- **Use Case Report: Scenarios 场景**
 - A use case actually describes a set of sequences in which each sequence in the set represents one possible flow through all those variations.
 - A scenario is a specific sequence of actions that illustrates behavior.
 - Scenarios are to use cases as instances are to classes, meaning that a scenario is basically one instance of a use case.

细化UseCase——用例建模的步骤

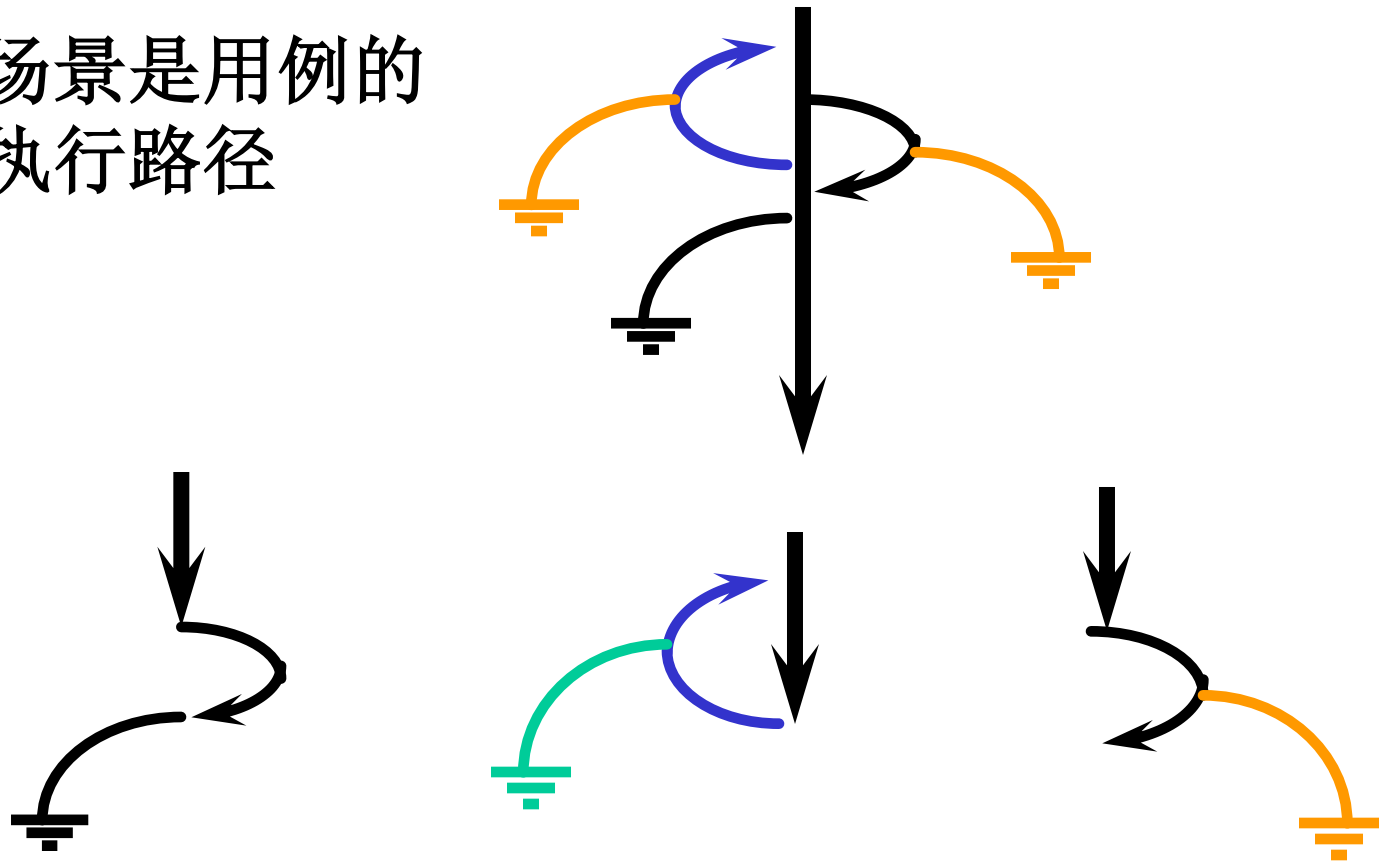


- 场景Scenarios
 - 用例实例 (Use case Instance)
 - 参与者使用系统的一个特定情节
 - 例如
 - 在ATM机上成功取款的场景
 - 因余额不足取款失败的场景

细化UseCase——用例建模的步骤



- 场景
 - 一个场景是用例的一条执行路径



细化UseCase——用例建模的步骤



用例规格说明

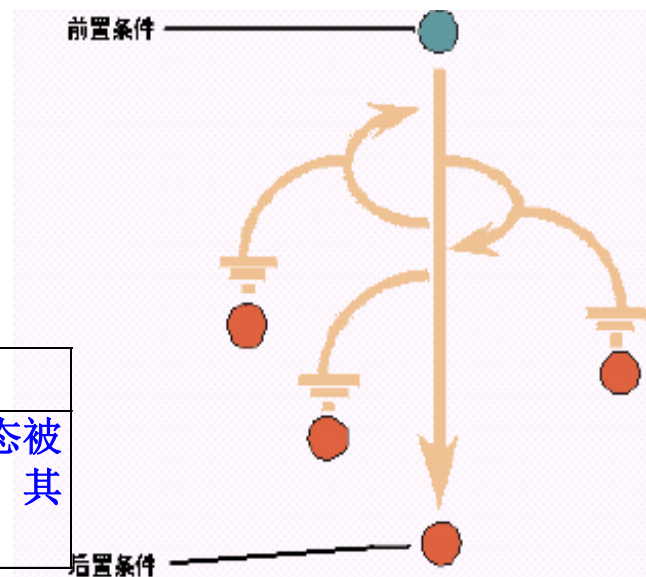
- 前置条件和后置条件
 - 对系统状态的简单描述和约束
 - 前置：阻止参与者触发用例直到满足条件
 - 后置：说明用例执行后什么必须为真
 - 表现形式：布尔条件

“竞拍” Use Case 的前置条件

买主登录	在买主能够竞拍之前必须登录系统
买主展示 拍卖物项	买主找到其准备竞拍的拍卖项

“终结拍卖” Use Case 的后置条件

拍卖物项被解除	当针对某一拍卖物项的拍卖终结的时候，其状态被设置为“终结”，该拍卖物项将不能再被浏览到，其相关信息也不能在被改变。
---------	---



细化UseCase——用例建模的步骤

用例规格说明

• 前置条件和后置条件

— 前置条件必须是系统能够检测到的



前置条件: ~~账户有足够金额~~

前置条件: **ATM机正常运行;**
用户有可插入的银行卡

后置条件	最小保证	客户活动被摄像或记录; 银行系统和数据保持完整性。
	成功保证	客户取回银行卡、凭条和正确数量的现金, 账户数据被正确修改, 银行记录了取款记录。

细化UseCase——用例建模的步骤

- 说明用例的特殊需求
 - 说明事件流没有包括的所有用例需求

举例：特殊（性能）需求

在卖主发出账单支付请求时，在 90% 的情况下，系统应该在 1 秒钟内对请求的验证做出响应。验证的时间绝对不能超过 10 秒钟，除非网络中断（这时应该通知用户）。

“竞拍” Use Case 的特殊需求

系统必须保证处理竞拍的次序与竞拍的实际次序严格一致。

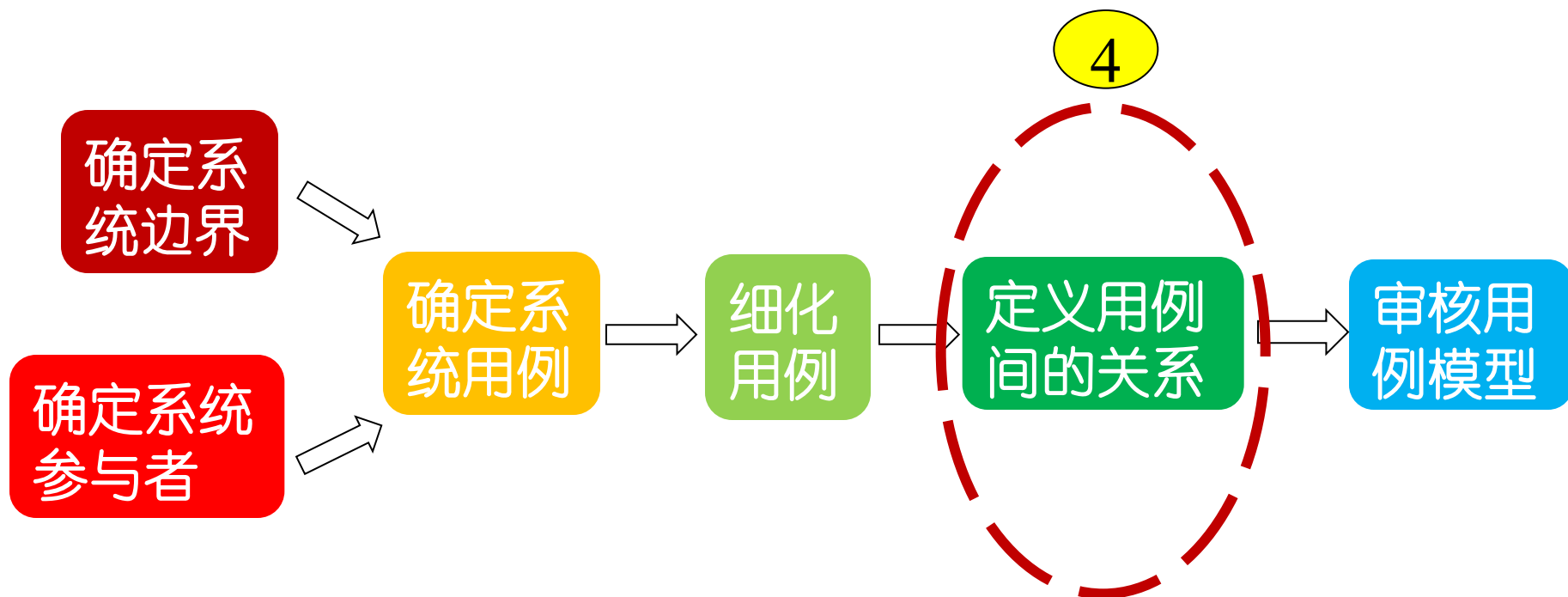
买主进入拍卖目录之后，在鼠标三次点击之内就能够完成针对某一拍卖物项的竞拍

细化UseCase——用例建模的步骤

- 说明用例的特殊需求
 - 性能需求
 - 触发事件、周期性、频率、反应事件和反应时间等
 - 安全性
 - 健壮性
 - 吞吐量
 - 冗余度策略
 - 要求达到的正常运行时间
 - 要求达到的软件缺陷率
 - 分布性
 - 持久性存储
 - 应该遵守的标准
 - 应该遵守的质量标准(例如: ISO 9000)

用例建模的步骤

- 用例建模的步骤



定义用例之间的关系——用例建模的步骤

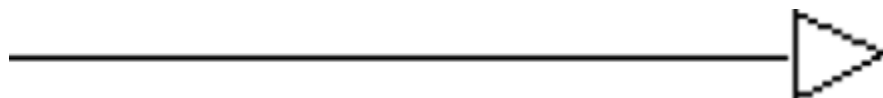


- 构建用例之间的关系
 - 组织用例，整理用例文档
 - 将基本用例分解为粒度缩小的部分
 - 分解公共行为
 - 分解变体
 - 抽象出关键和核心的工作单元

定义用例之间的关系——用例建模的步骤

- 用例之间的关系

- 泛化：同一业务目的的不同实现



- 包含：提取公共步骤，便于复用

«include»



- 扩展：显示用例的可选分支

«extend»

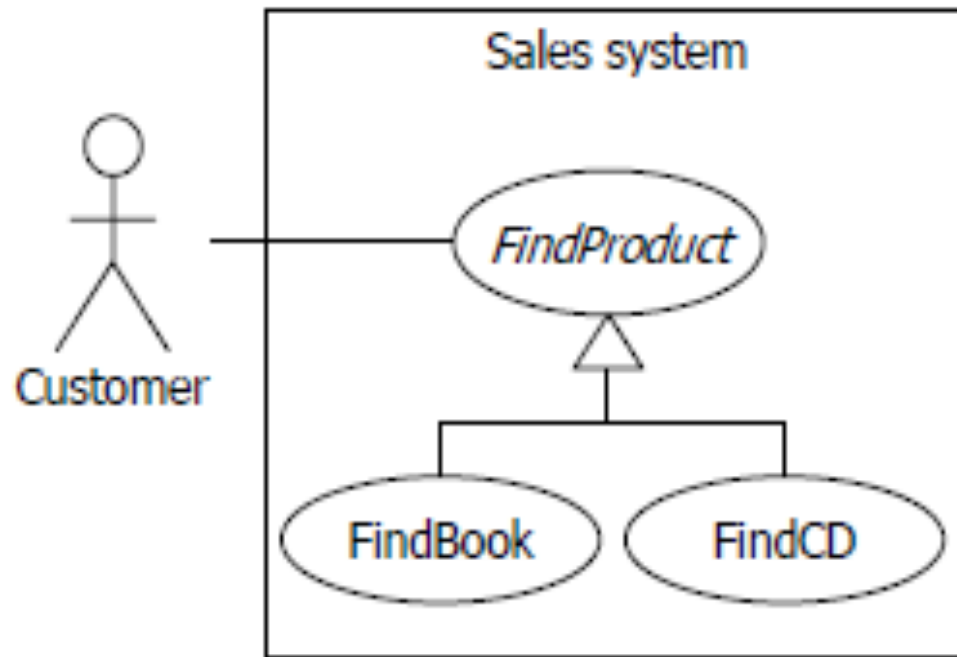


定义用例之间的关系——用例建模的步骤

- 用例之间的泛化关系
 - 子用例是父用例的特殊形式
 - 从父用例继承特征
 - 添加新的特征
 - 覆写已继承的特征

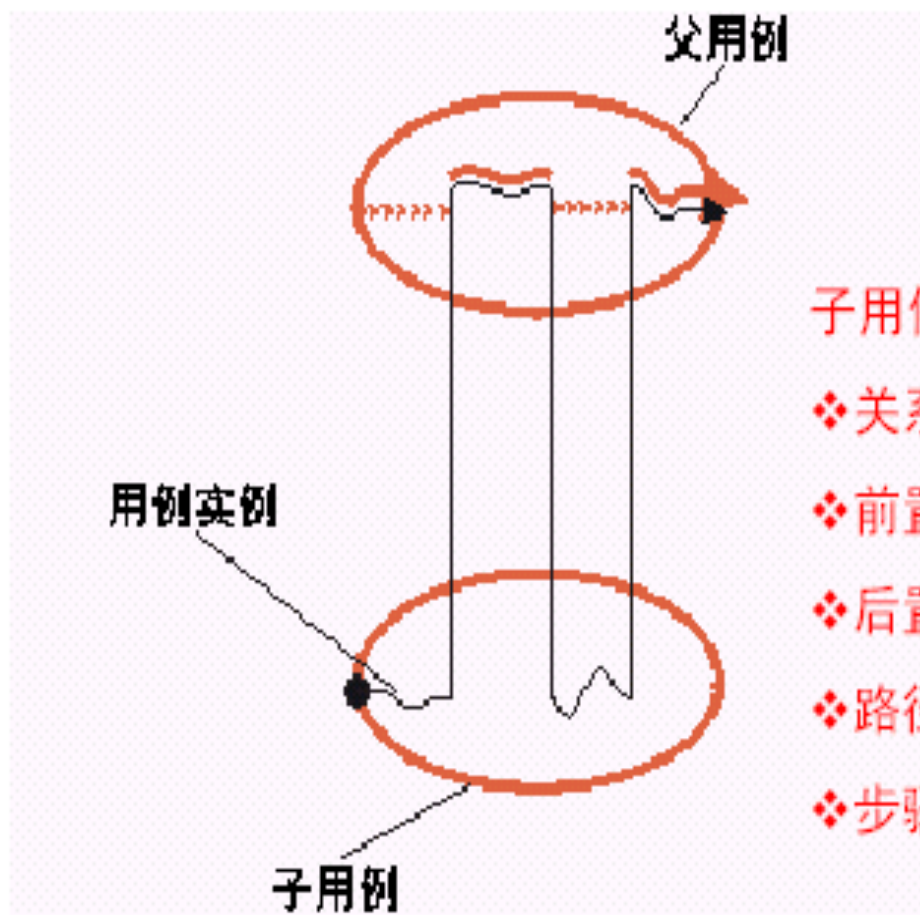
定义用例之间的关系——用例建模的步骤

- 泛化关系示例



定义用例之间的关系——用例建模的步骤

• 用例泛化



子用例继承父用例的一切:

- ❖ 关系
- ❖ 前置条件
- ❖ 后置条件
- ❖ 路径
- ❖ 步骤

定义用例之间的关系——用例建模的步骤

- 何时使用泛化关系？

- 面临的问题

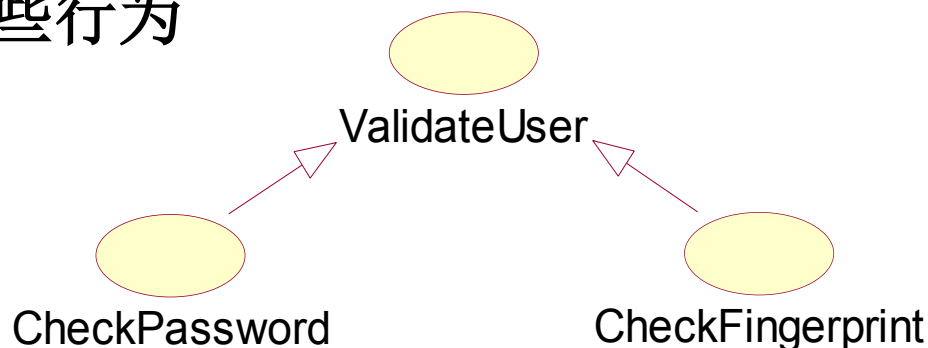
- 用例间的公共行为如何将其抽象出来？

- 解决方案

- 解析出多个用例的公共行为
 - 子用例继承父用例的行为和含义
 - 还可增加和覆写一些行为

- 举例

- 验证用户的用例

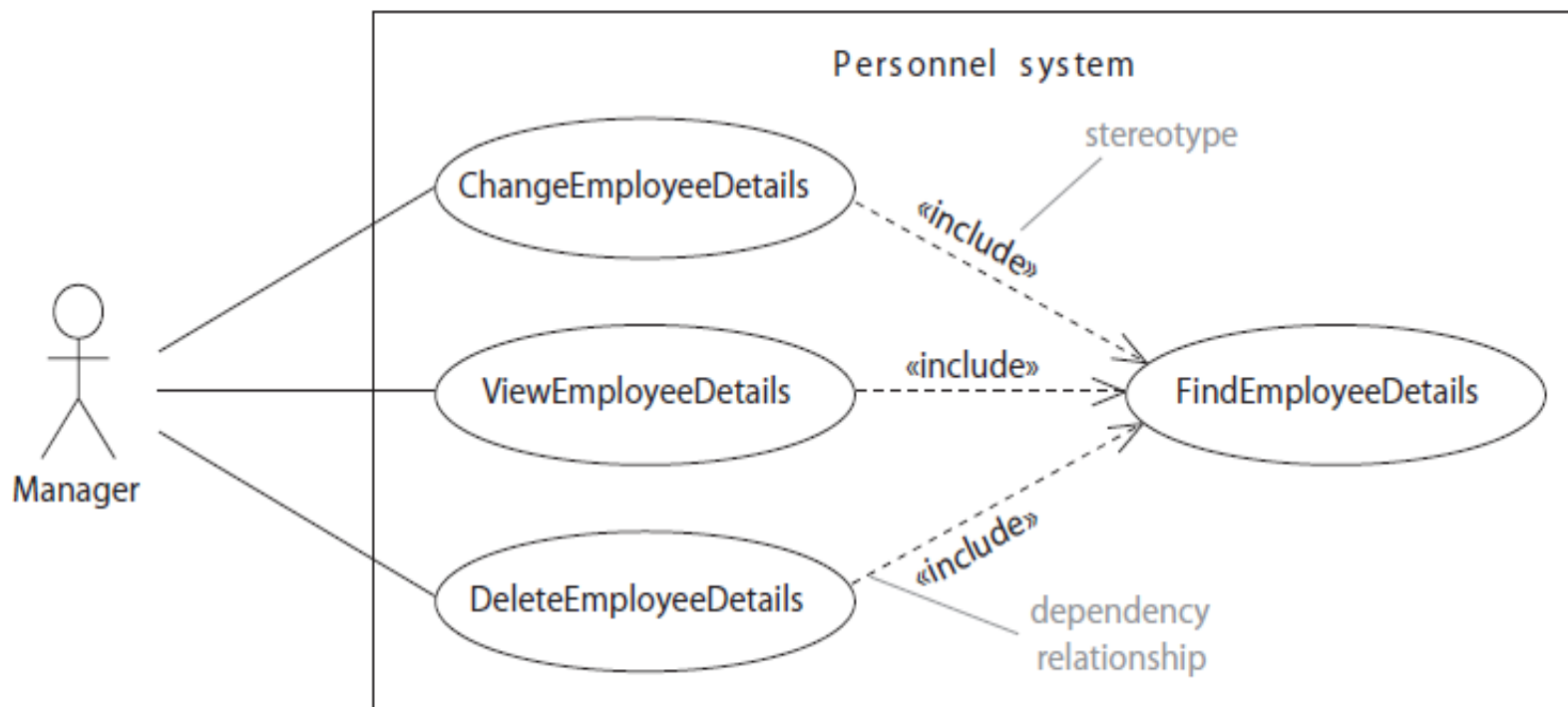


定义用例之间的关系——用例建模的步骤

- 用例之间的**包含**
 - 一个用例中含有另一个用例的行为
 - 基本用例
 - 包含用例
 - 包含用例的行为被插入的基本用例中

定义用例之间的关系——用例建模的步骤

- 包含关系示例



定义用例之间的关系——用例建模的步骤

- 用例包含



定义用例之间的关系——用例建模的步骤

- 何时使用**包含**关系？

- 面临的问题

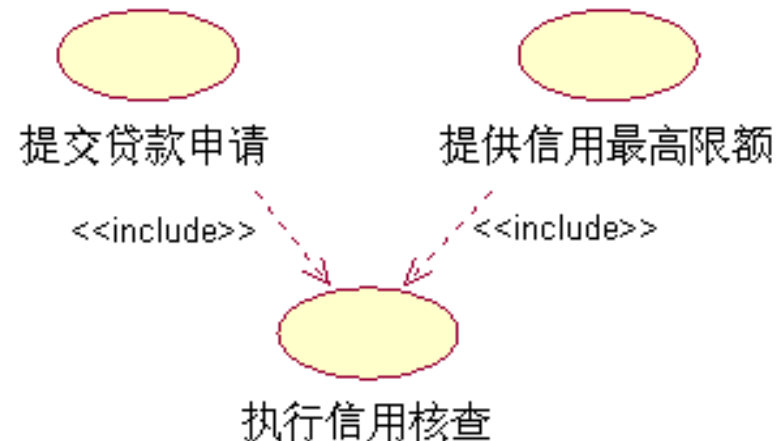
- 已有功能如何复用？

- 解决方案

- 用例A到B的包含关系指明A的实例执行所有B中描述的行为
 - “A将职责委派给B”

- 举例

- 核查信用的用例



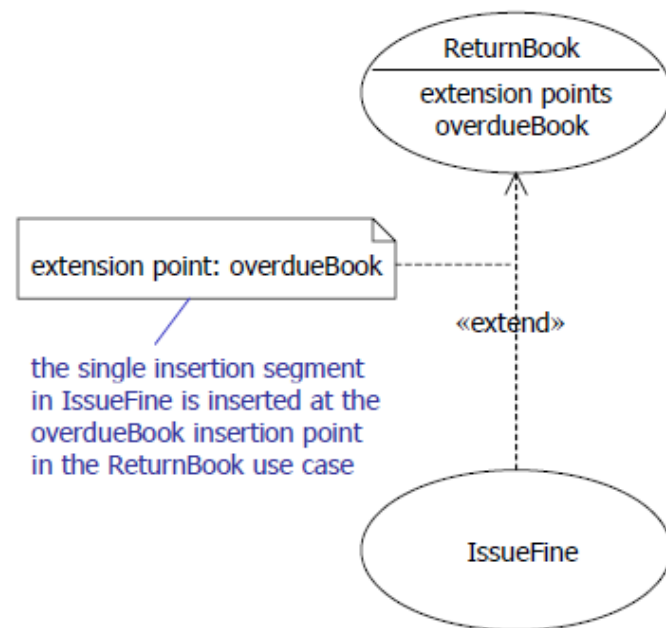
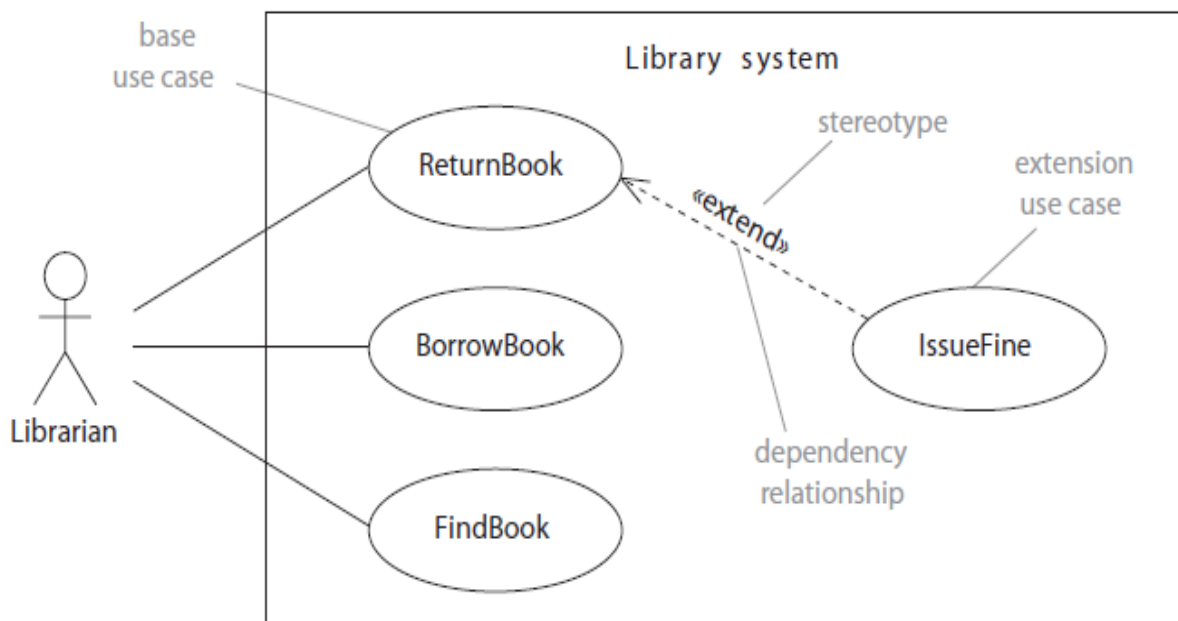
定义用例之间的关系——用例建模的步骤



- 用例之间的**扩展**
 - 用例的行为在一定条件下被另一个用例扩展
 - 基本用例
 - 扩展用例
 - 扩展点
 - 条件扩展

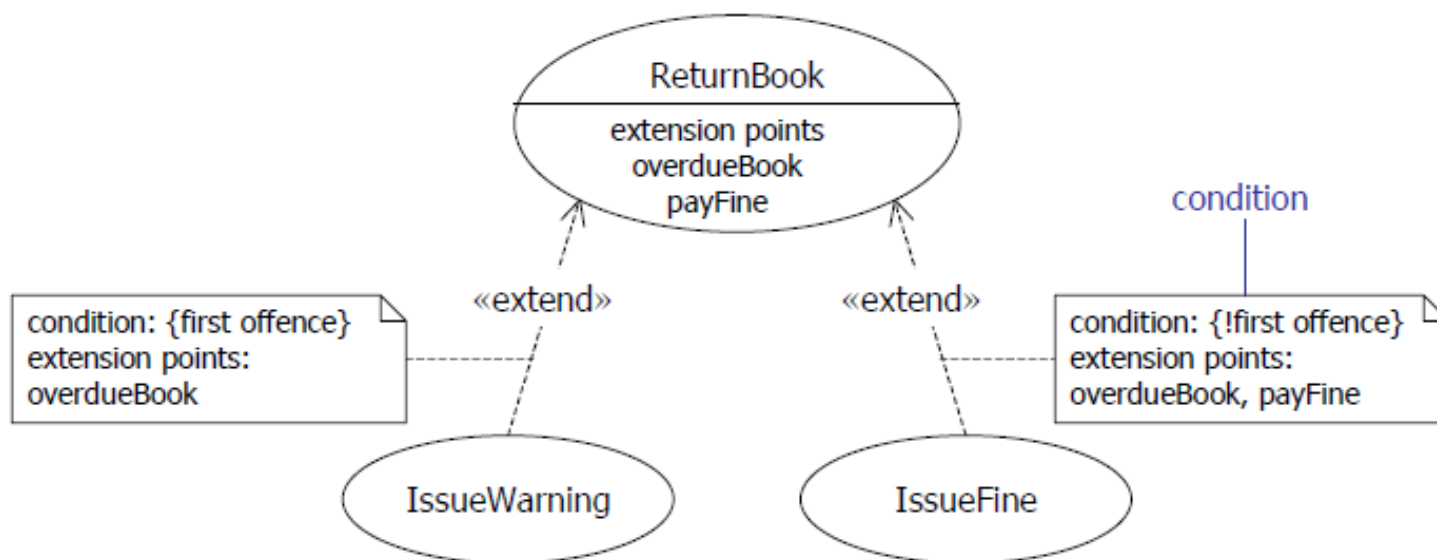
定义用例之间的关系——用例建模的步骤

• 扩展关系示例



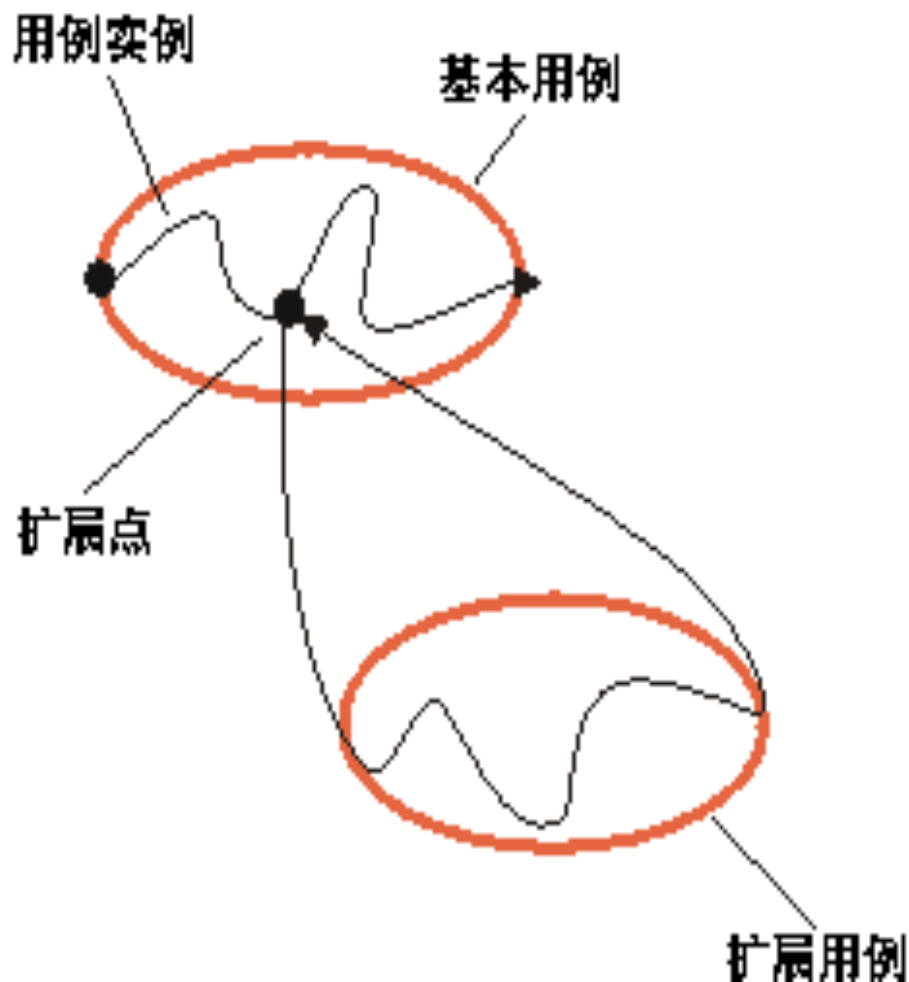
定义用例之间的关系——用例建模的步骤

- 扩展关系示例



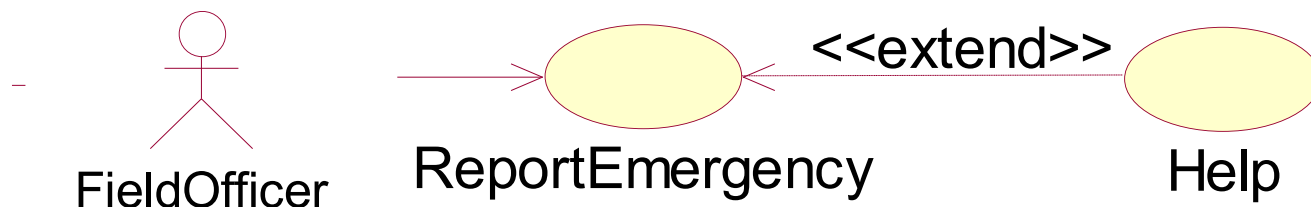
定义用例之间的关系——用例建模的步骤

- 用例扩展



定义用例之间的关系——用例建模的步骤

- 何时使用**扩展**关系？
 - 面临的问题
 - 事先无法预知所有可能的扩展
 - 解决方案
 - 从A指向B的<<extend>>指明A是B扩展行为
 - 举例
 - 报告紧急情况的用例



定义用例之间的关系——用例建模的步骤

- 用例扩展

- 适合应用的情况

- 对用例中可选的系统行为部分建模
 - 描述一个只在给定条件下可执行的子流
 - 对通过与参与者显式交互而控制的流建模
 - 区分可实现系统的可配置部分

- 为处理异常或需要灵活框架的情况提供方法

- 扩展用例并非用于捕捉系统中所有可能发生的错误条件

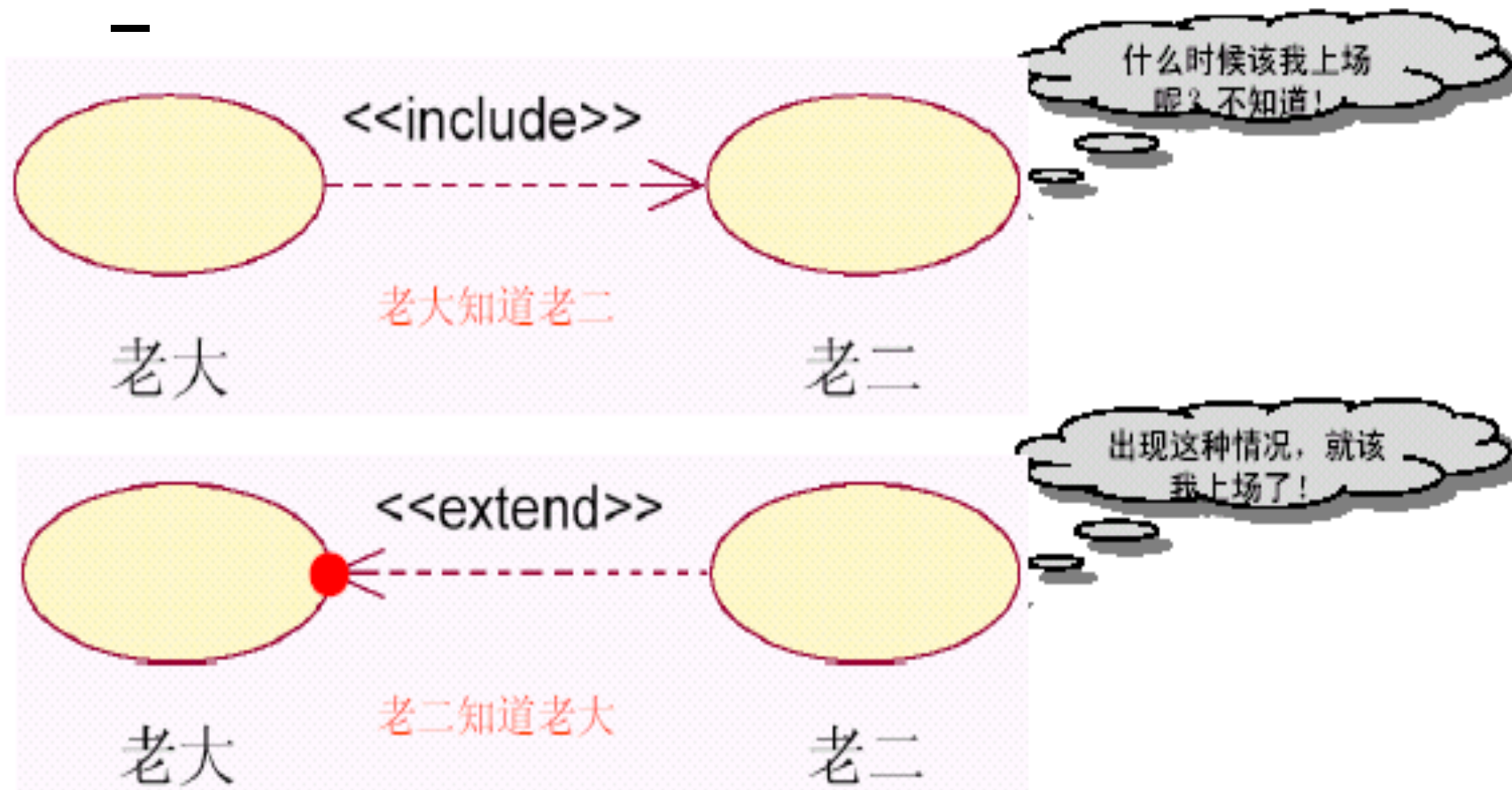
定义用例之间的关系——用例建模的步骤

• 包含VS扩展

	包含用例	扩展用例
这个用例是可选的吗？	N	Y
若缺少这个用例，基用例完整吗？	N	Y
用例的执行需要满足某种条件吗？	N	Y
基用例是否知道其细节？	Y	N

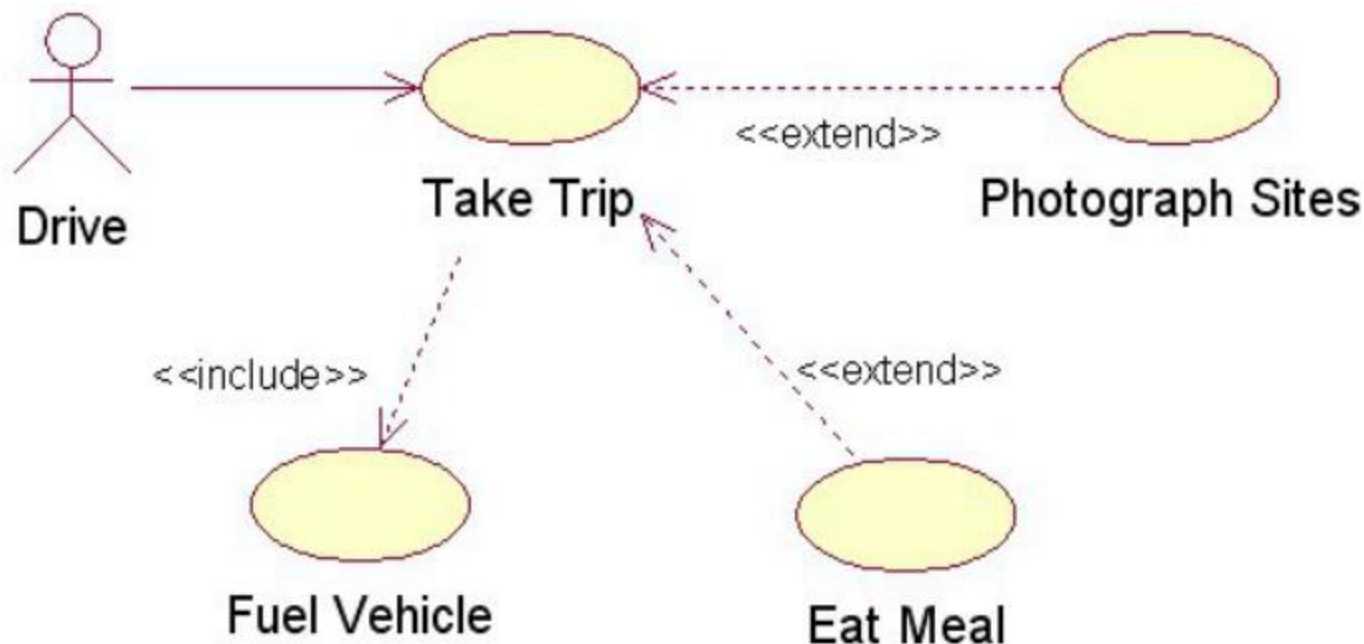
定义用例之间的关系——用例建模的步骤

- 包含VS扩展的可见性



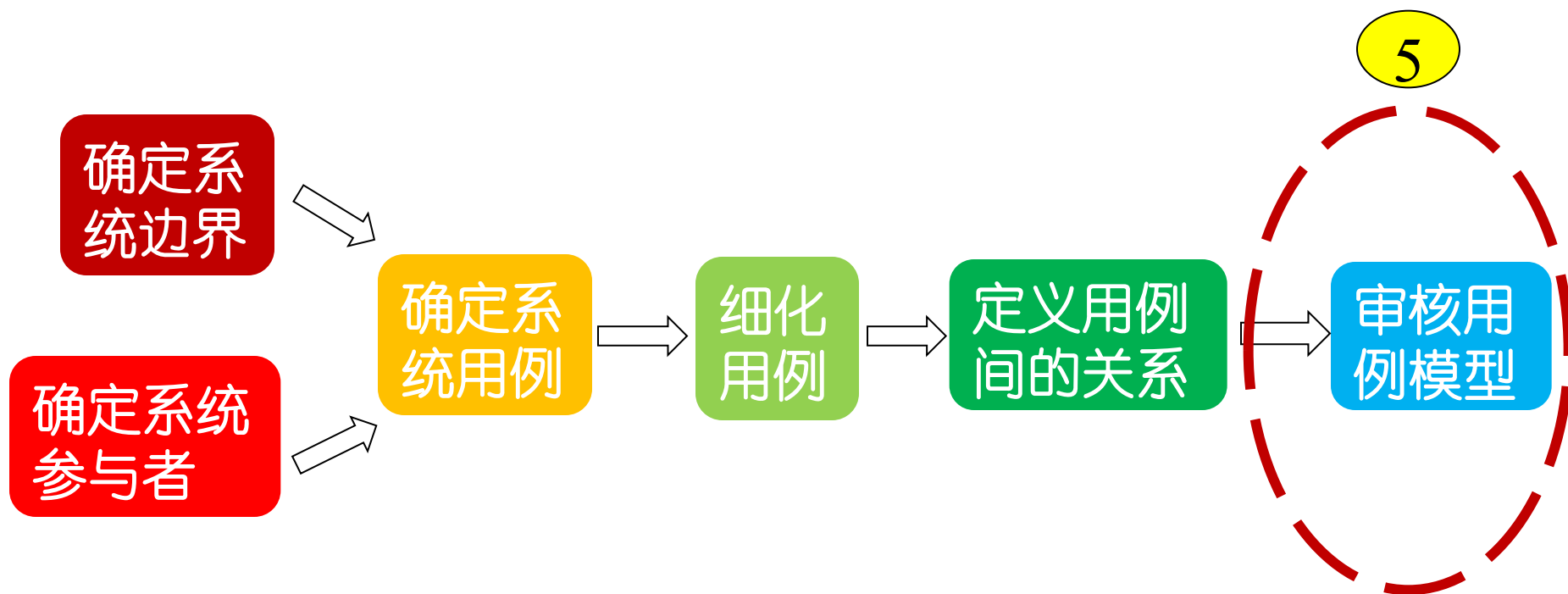
定义用例之间的关系——用例建模的步骤

- 开车旅行的用例
 - 给汽车加油——必选行为
 - 就餐、景点拍照——可选行为





用例建模的步骤

- 用例建模的步骤



检查用例模型——用例建模的步骤

用例的层次

- 用例是目标驱动的
 - 用例建模过程从期望的“概要目标”（白色）开始——白云（顶级）
 - 细化生成多个用户目标（蓝色）——这是用例模型最重要的层次，应该尽量详尽地编写文档。（风筝）
 - 接下来派生出子功能层（靛蓝色）。如《include》的用例，需要为这个用例编制文档。
- 考虑用户实现目标所要花费的时间
 - 如果花费时间是可观的（比如，超过半个小时），则很可能是一个白色层次的目标；
 - 如果花费时间教少，则可以作为很好的用户层目标。

检查用例模型——用例建模的步骤

用例模型上的关系的比较

- 参与者、用例间的关系类型

关系类型	说明	表示符号
关联(association)	actor 和 use case 之间的关系	_____
泛化 (generalization)	actor 之间或 use case 之间的关系	_____>
包含(include)	use case 之间的关系	<<include>> ----->
扩展(extend)	use case 之间的关系	<<extend>> ----->

检查用例模型——用例建模的步骤

定义用例关系的原则

- 遵循用例划分的基本原则

- 关注WHAT而不是HOW

- “WAVE” 测试

W 用例是否描述了应该做什么(What to do), 而不是如何做?
A 用例的描述是否采取了参与者的视点(Actor's point of view)?
V 用例是否对参与者有价值(Value for the actor)?
E 用例描述的时间流是否是一个完整场景(Entire scenario)?

- 保持用例简短

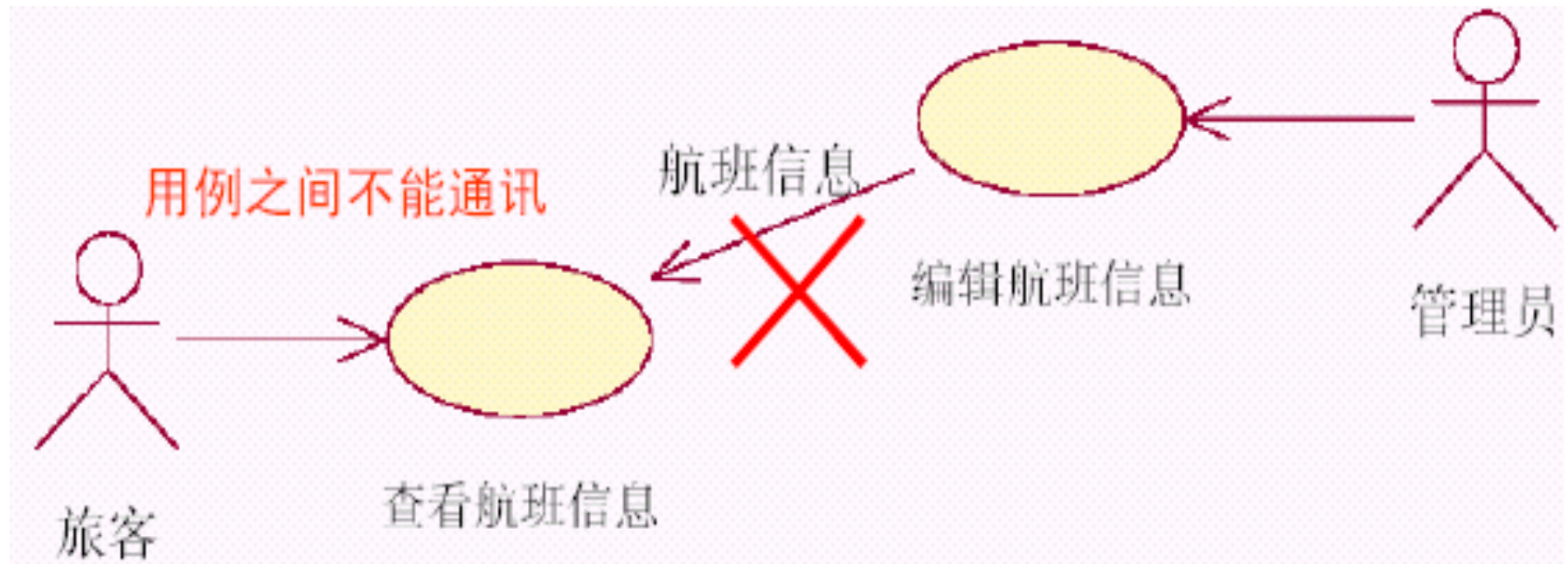
- 简化和澄清用例模型

- 在提高用例模型整体清晰度时才使用

- “纸张崇拜 (paper envy)”

检查用例模型——用例建模的步骤

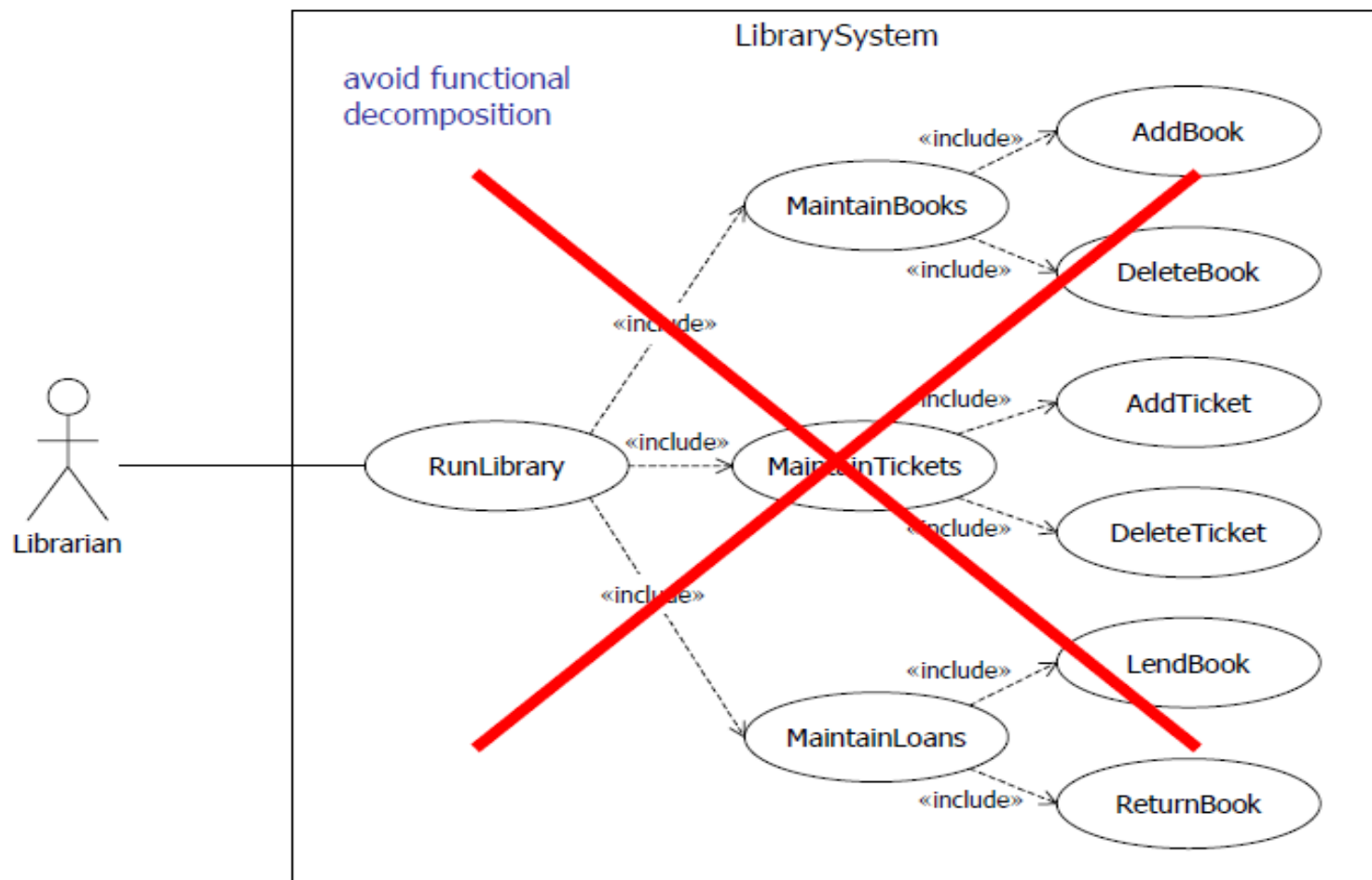
Structure the Use-Case Model



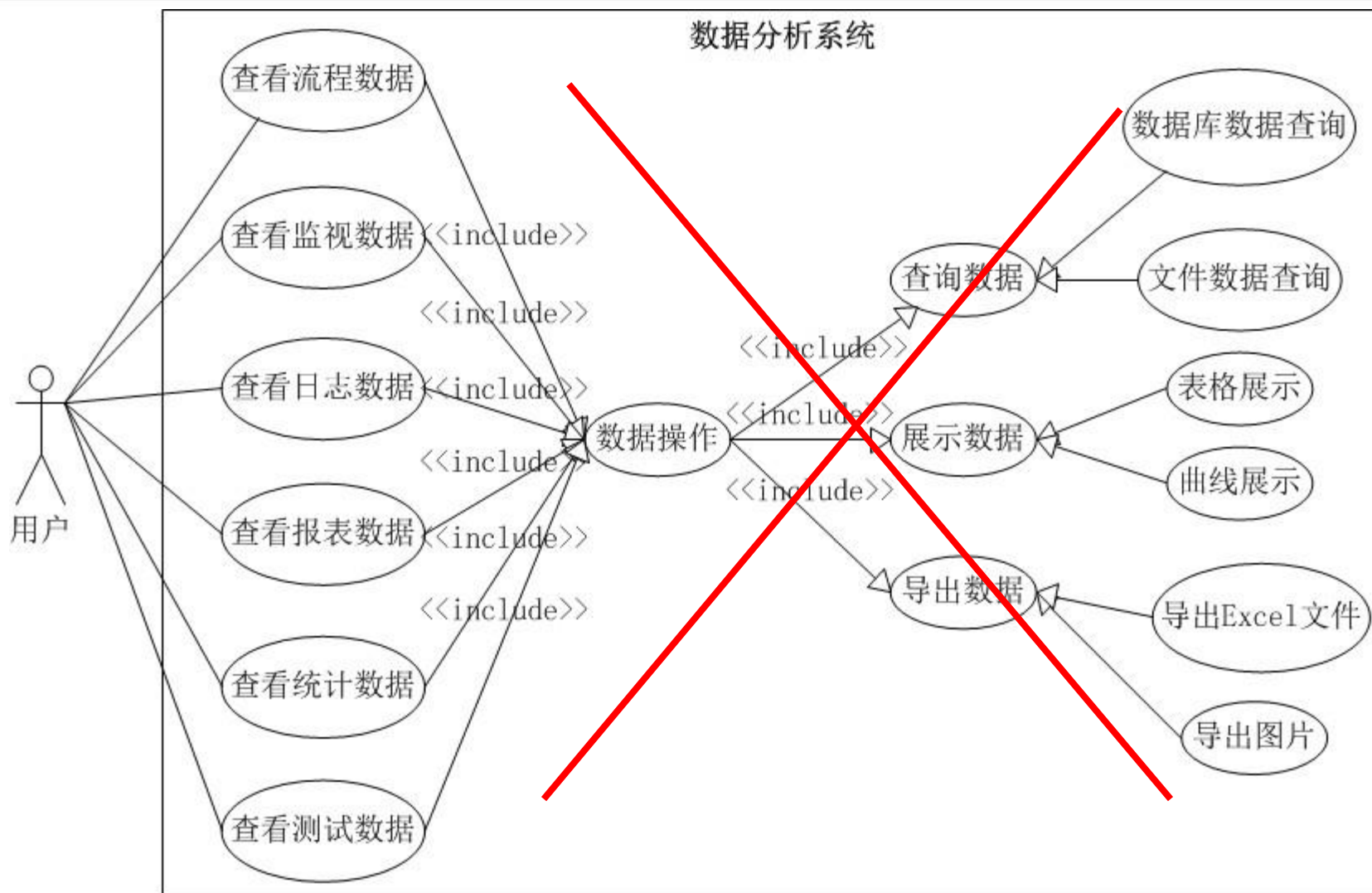
检查用例模型——用例建模的步骤

• 避免功能分解

定义用例关系的原则



用例图不能随意嫁接组合



检查用例模型——用例建模的步骤

定义用例关系的原则

- 避免功能分解
 - 检查任何具有深度层次的用例
 - 自然的层次通常不能超过一级深度
 - 整个模型不要植根在单一用例中

小结



- 重点
 - 用例建模的重要性
 - 用例建模的步骤
 - 用例规则说明
 - 用例间的关系
 - 建模用例的原则