

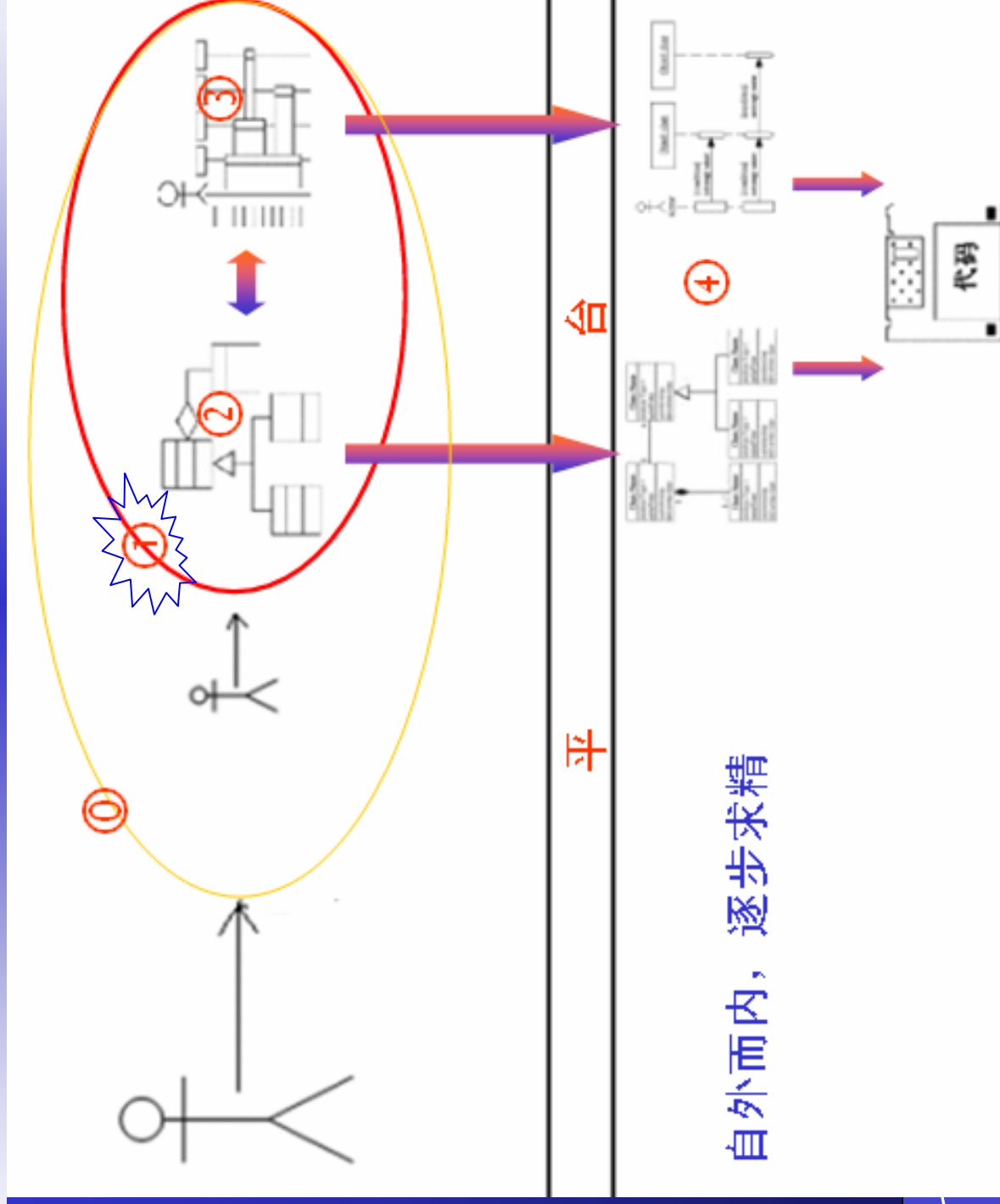
UMLChina训练 (中阶)

用例



Think

开发过程--①用例



本节目标

得到有价值的需求文档
不但形式正确，更要内容正确



告别鸡肋软件



本节线索

需求问题和用例总览

实作：执行者

实作：用例

实作：用例文档

用例关系辨析

业务用例、排序、分包



需求——石头问题

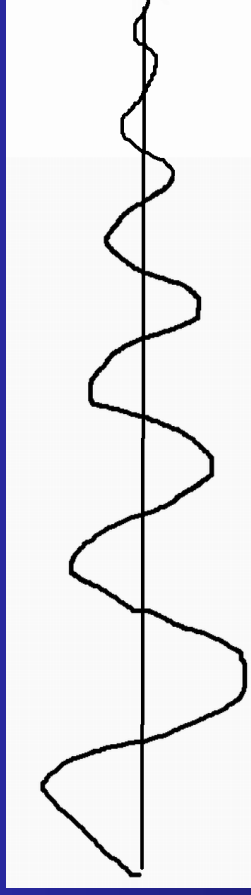
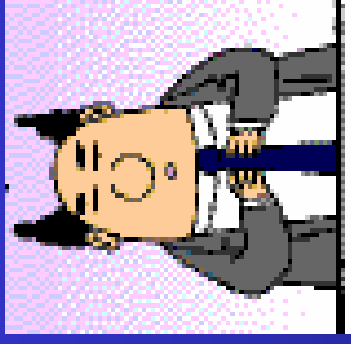
我要一块石头....

差不多，但我要小一点的....

很好，不过我要蓝色的....

啊，没有那么小....

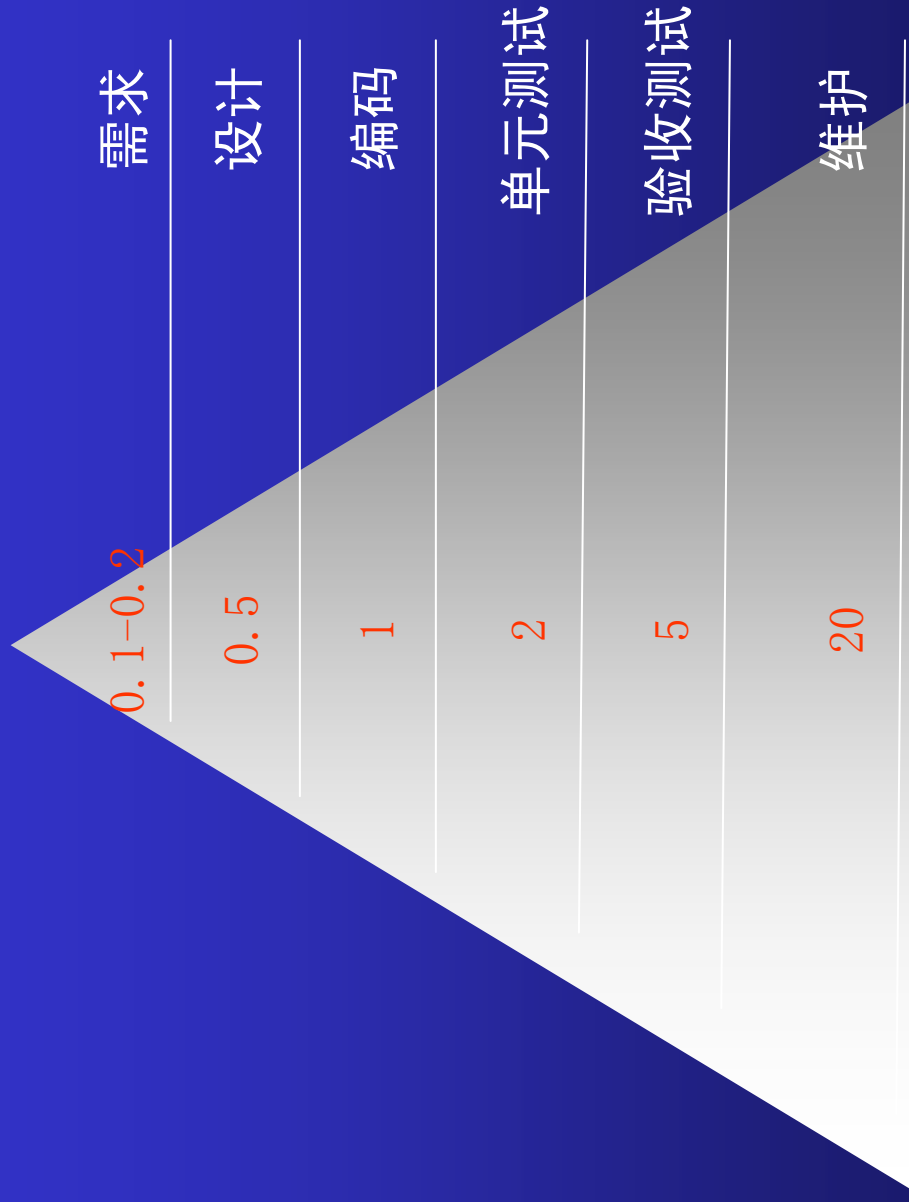
咳，还是原来那个好了....



难捕获、易变



需求问题的代价



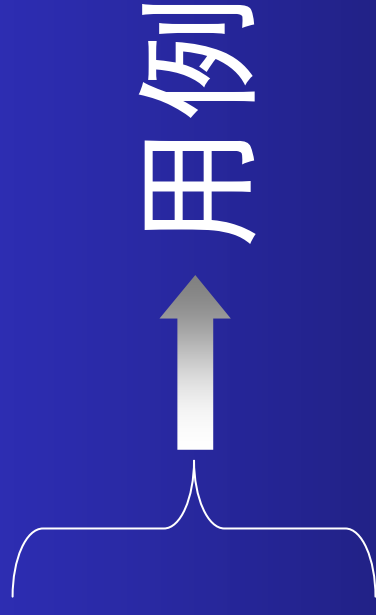
修复错误的成本200:1



需求问题——对策

难捕获→从用户视角看问题

易变→合理的结构



用例：既简单又复杂



如此多的人跌倒在这个简单的概念上



用例：用户视角的需求组织形式

立足用户视角

用户插入ATM卡

系统要求输入合法的密码

用户输入密码.

系统提示用户选择“存款”或者“取款”

用户选择“取款”

系统提示用户输入取款金额

用户输入(合理)取款金额并确认

系统从帐户中扣除取款金额

系统提示用户“打印收据”或者“不打印收据”

用户要求不打印收据

系统显示“交易结束”

立足开发者视角

系统要求用户输入合法的密码

系统提供“存款”或“取款”两种选项

系统能够接受用户录入取款金额

系统能够从帐户中扣除取款金额

系统允许选择“打印收据”或者“不打印收据”

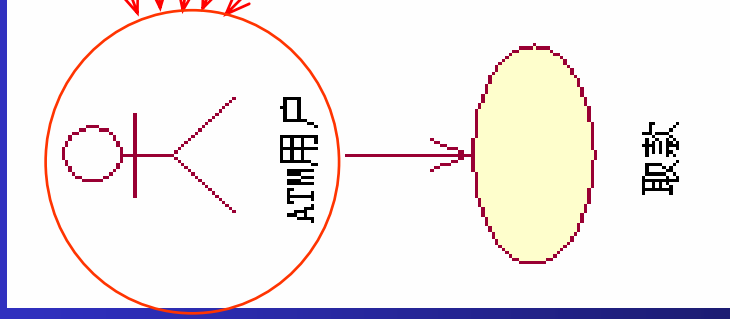
系统能够显示交易结束信息



用例：需求按目标组织

基本路径

- 用户插入ATM卡
- 系统要求输入合法的密码
- 用户输入密码.
- 系统提示用户选择“存款”或者“取款”
- 用户选择“取款”
- 系统提示用户输入取款金额
- 用户输入(合理)取款金额并确认
- 系统从帐户中扣除取款金额
- 系统提示用户“打印收据”或者“不打印收据”
- 用户要求不打印收据
- 系统显示“交易结束”



用例：有层次的需求组织形式

❖ 用例（取款）

❖ 路径（基本路径）

❖ 步骤（用户输入密码）

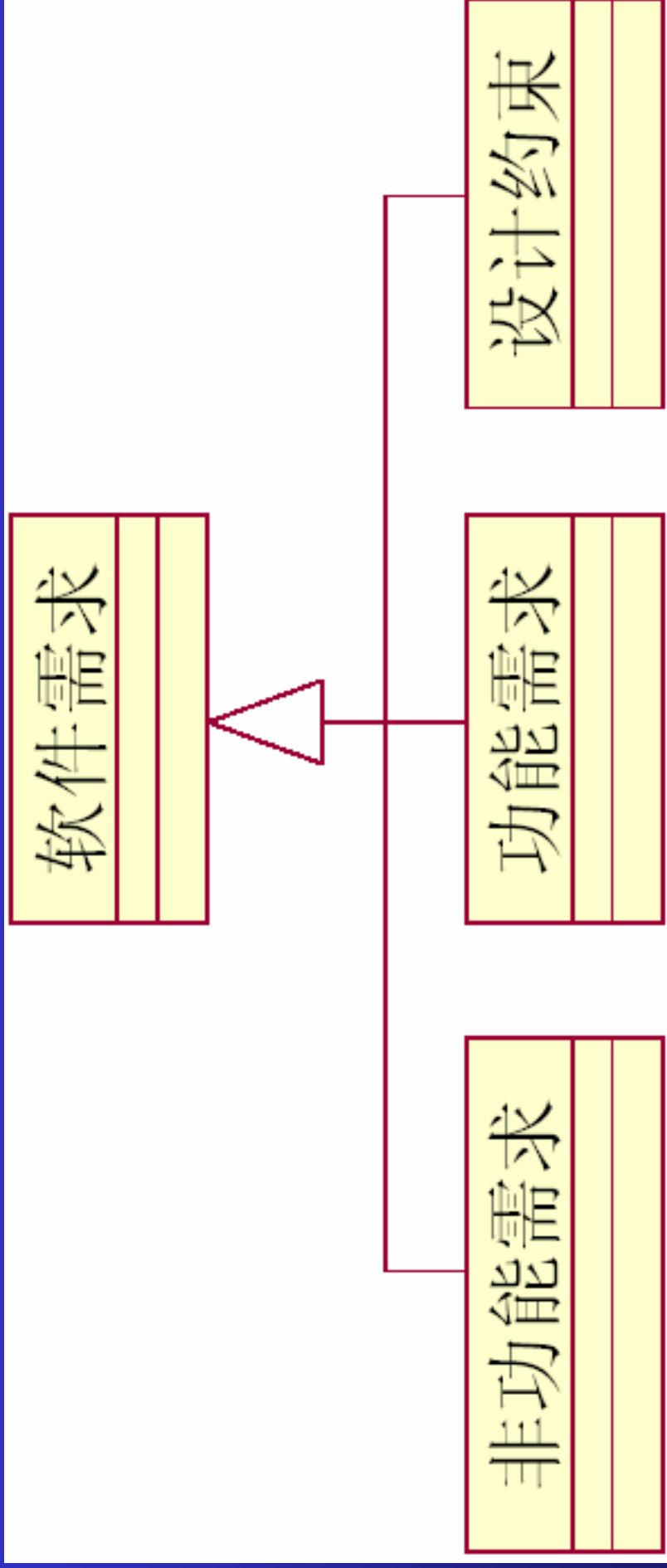
❖ 补充约束（密码由6位数字组成）

低精度，稳定

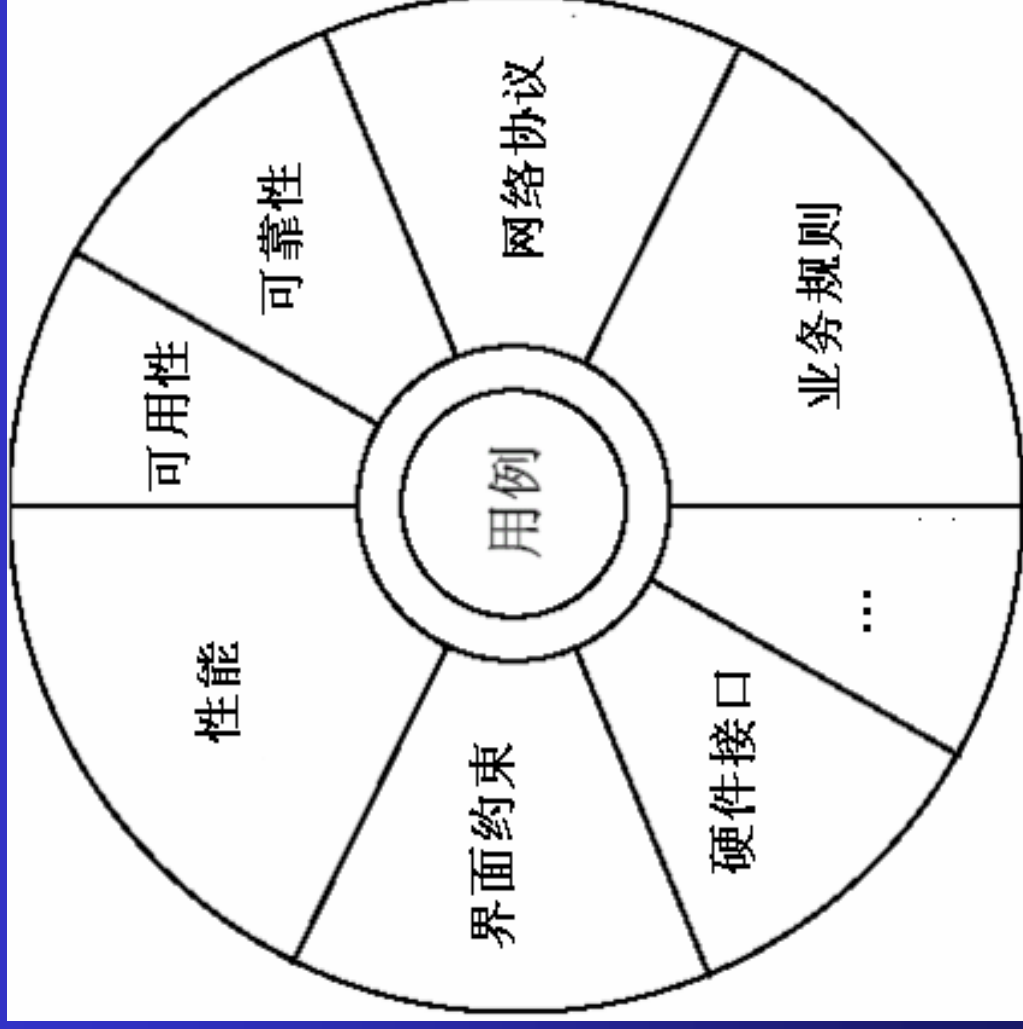
高精度，不稳定



三类需求



以用例为核心组织需求



使用用例组织的需求文档—示例

UCL, 注册

主执行者

潜在会员

前置事件

潜在会员访问系统

后置事件

系统记录注册信息

触发问题

潜在会员——希望注册成为会员

约束——希望确保对可能来自未验证用户的信息，使用最联系方法

基本流程

1. 潜在会员提交注册。
2. 系统显示注册界面。
3. 潜在会员提交注册信息。
4. 系统验证注册信息充分。
5. 系统生成用户名和密码，保存在注册信息。
6. 系统显示“注册成功，等待开发邮件”。

扩展

4a. 潜在会员提供的信息不充分。

4a1. 系统提示输入剩余信息

4a2. 返回 1

详细流程

2. 注册信息包括：公司名、联系人、电话、传真、Email、以及其他十个联系地址、联系地址、其它以下信息：州、城市、邮编、邮编。

业务规则

4. 公司名、联系人、电话是必需的。

非功能需求

设计的流

特殊问题

3. 用户名和密码生成规则特定

uctext.pdf

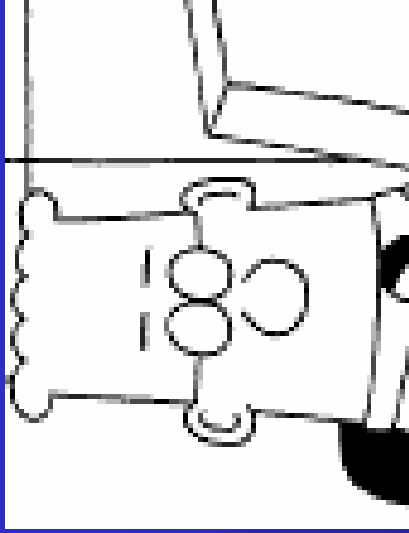


项目叙述

- ❖ 为什么要开发这个系统？
- ❖ 谁对系统需求有最后决定权？
- ❖ 这个系统会影响哪些人？
- ❖ 开发这个系统会有哪些困难？
- ❖ 开发这个系统能花多少钱？



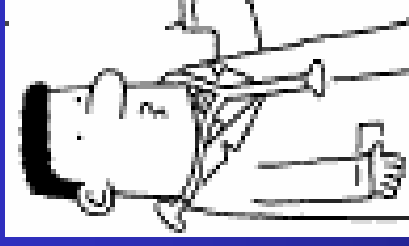
用例团队的组成



开发人员



业务专家



最终用户



步骤

- ❑ 识别执行者 (*)
- ❑ 识别用例 (*)
- ❑ 书写用例文档 (*)
- ❑ 识别用例的关系
- ❑ 用例的排序和分包



识别执行者

——关键词：边界



→ 专辑名称：边界

→ 歌手：齐秦



识别执行者

——执行者（Actor）

- 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



引入执行者帮助确定系统边界



识别执行者

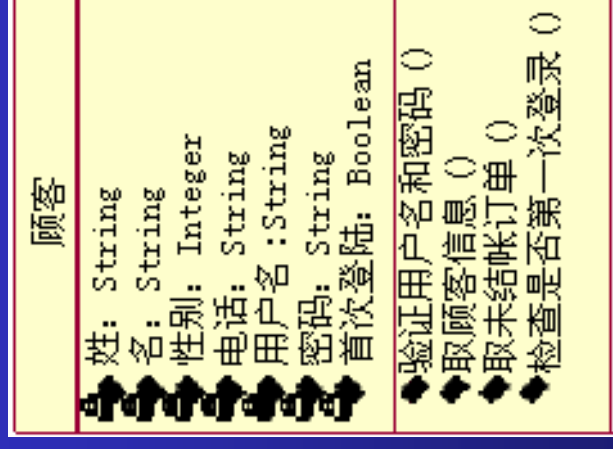
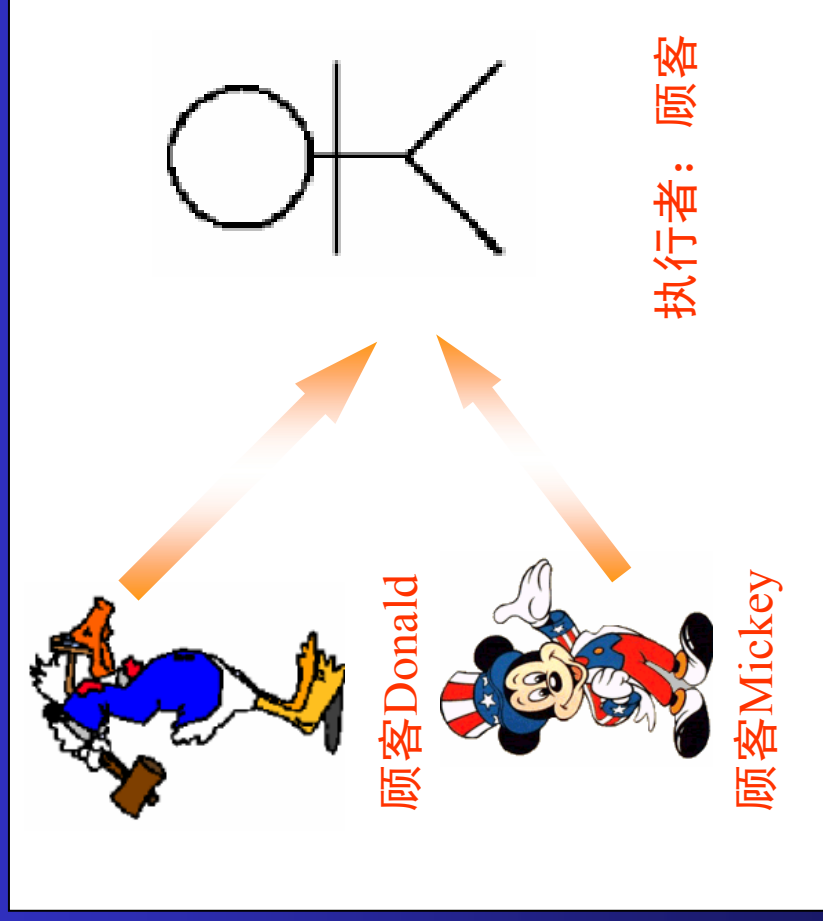
——执行者要点

- ❏ 系统外——执行者不是系统的成分
- ❏ 系统边界——责任边界，非物理边界
- ❏ 系统边界——直接与系统交互
- ❏ 有意义交互——属于目标系统的责任
- ❏ 任何事物——人、外系统、外部因素、时间



识别执行者

——在系统外：不在里面

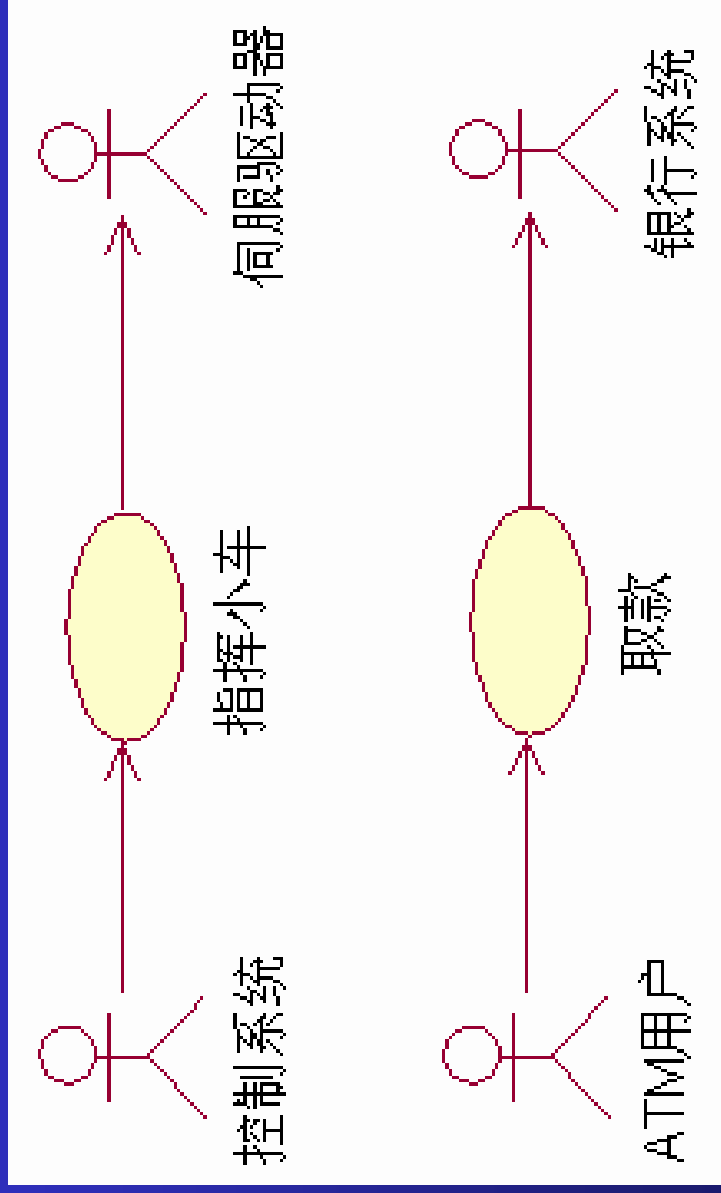


类：顾客



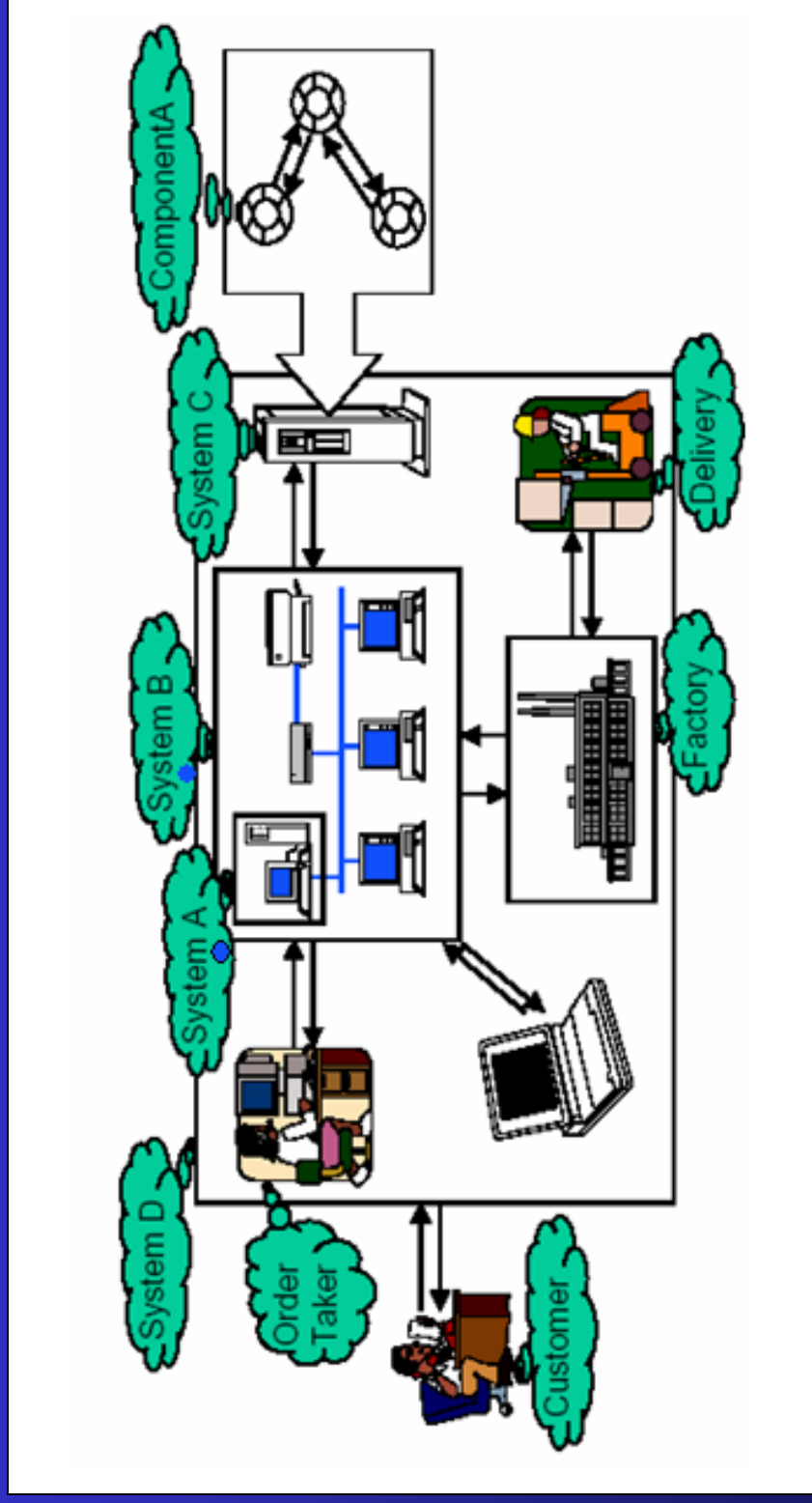
识别执行者

——系统外：已经存在，无可选择



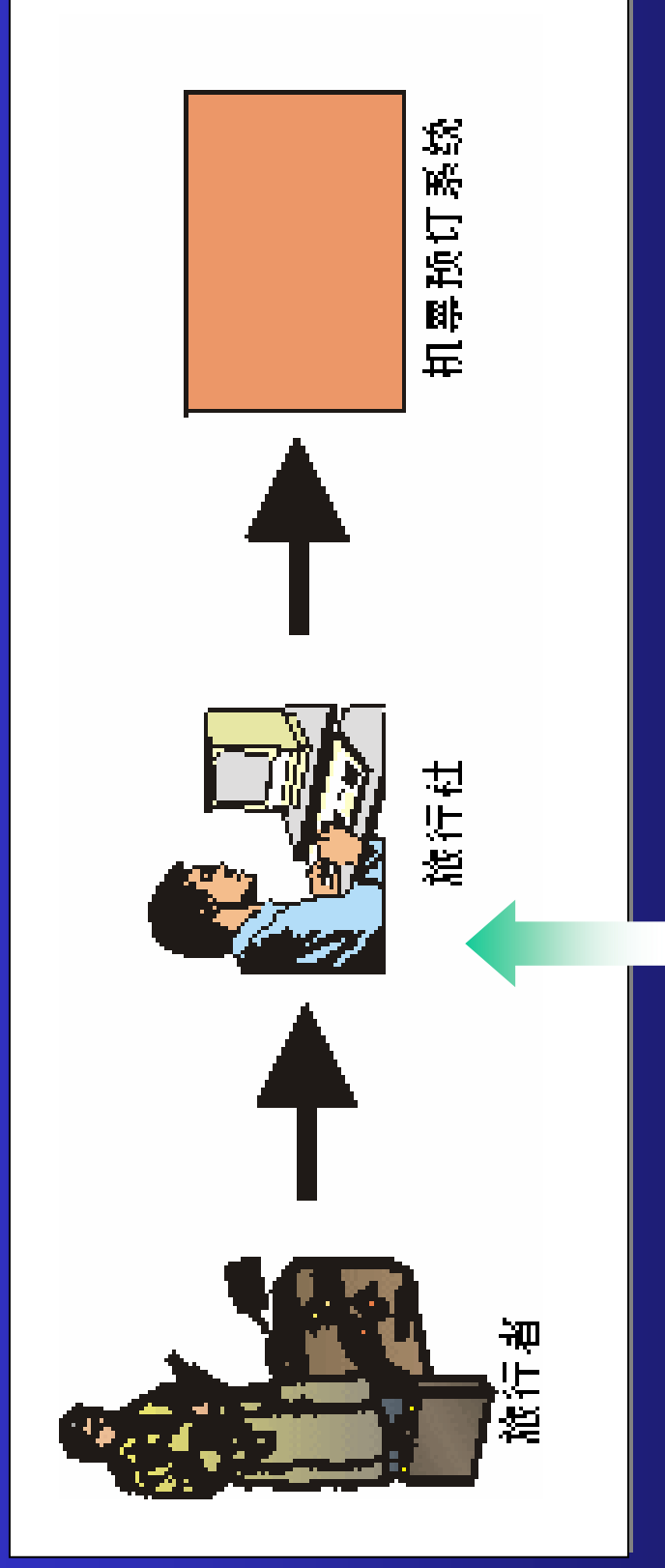
识别执行者

—— 责任的边界，不是物理的边界



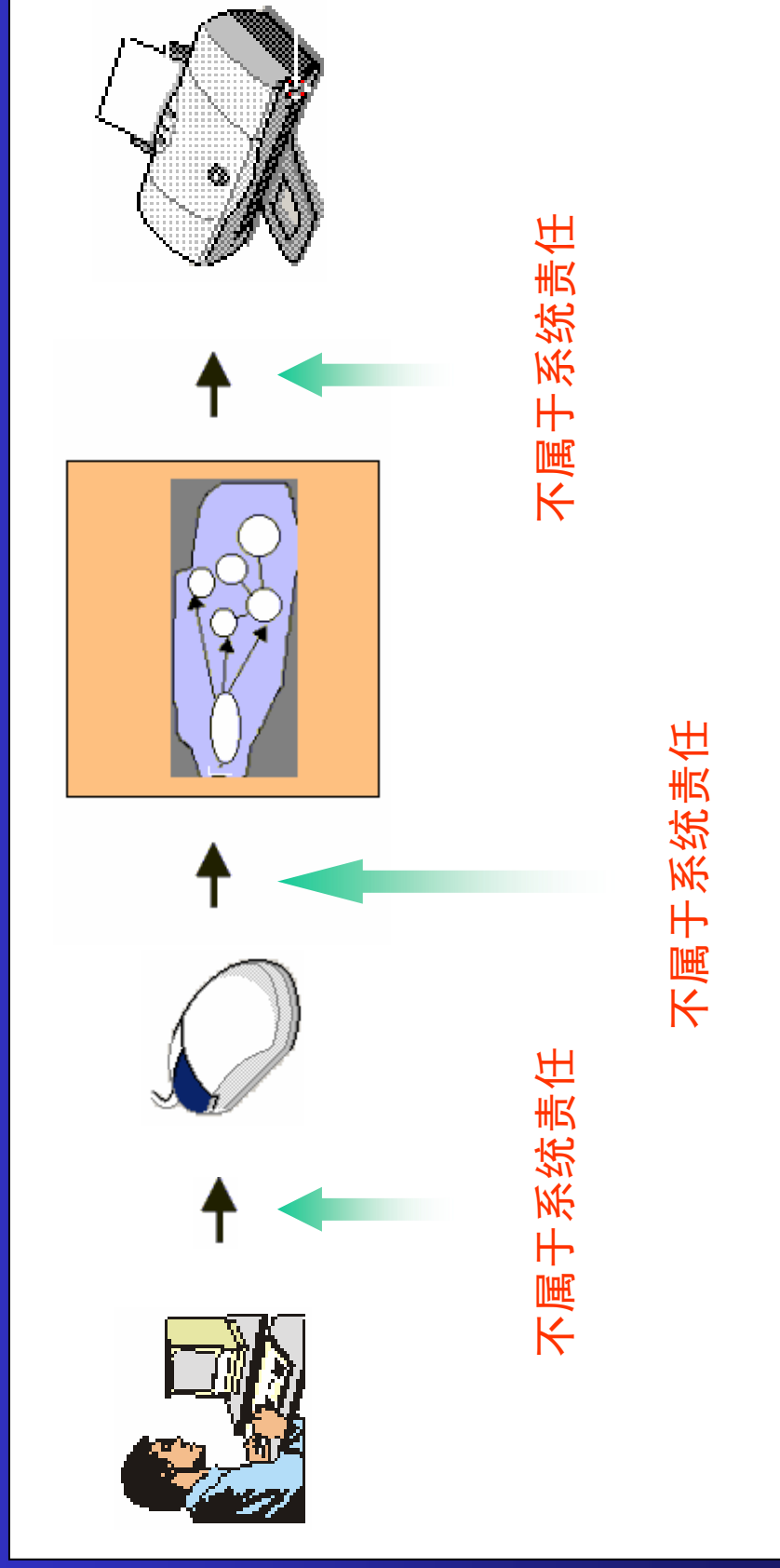
识别执行者

——直接与系统交互



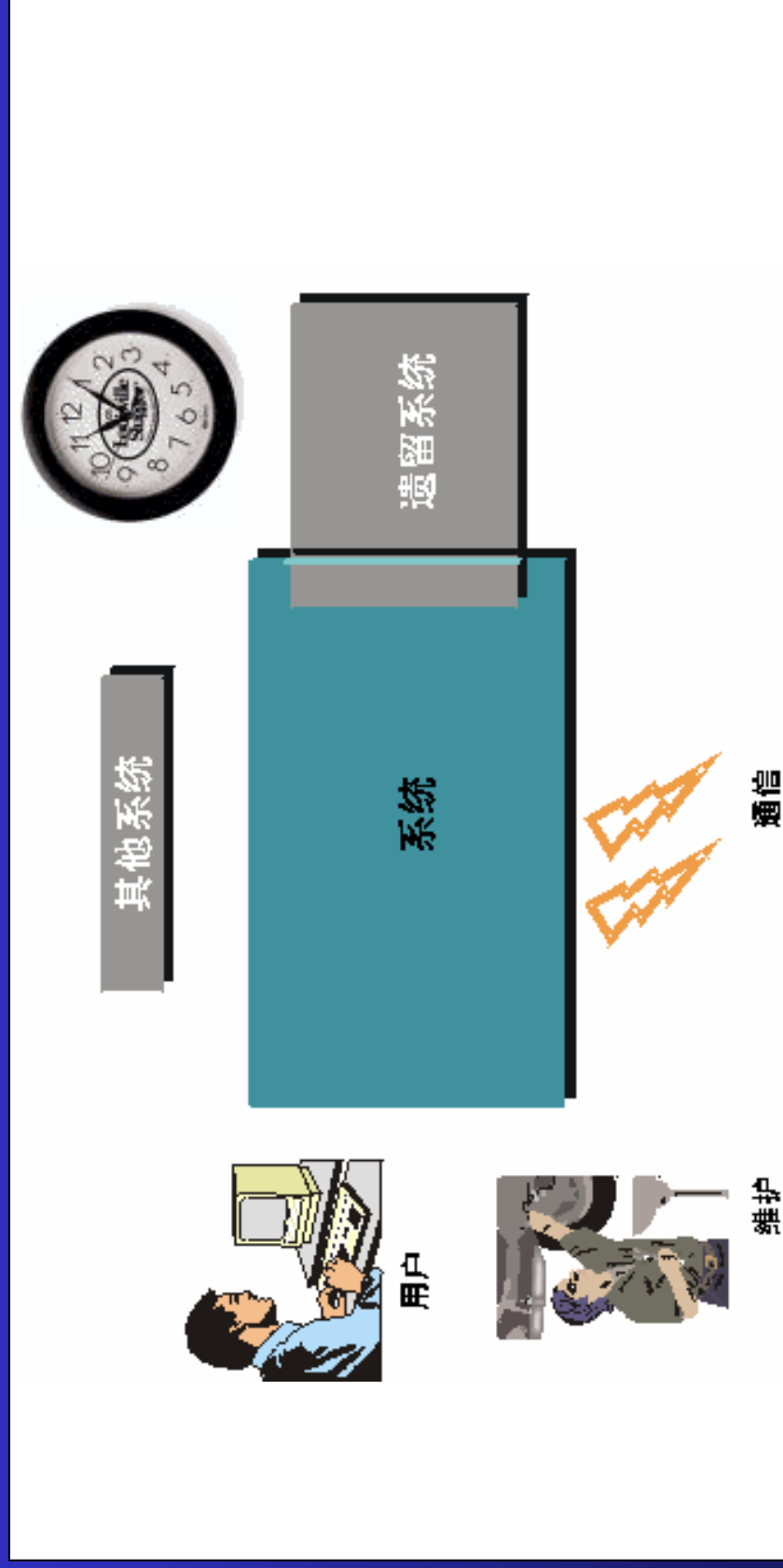
识别执行者

——有意义的交互



识别执行者

——任何事物



识别执行者

——讨论与练习



识别执行者

——识别执行者的思路

- ❖ 谁使用系统的主要功能？
- ❖ 谁改变系统的数据
- ❖ 谁从系统获取信息
- ❖ 谁需要系统的支持以完成日常工作任务？
- ❖ 谁负责维护、管理并保持系统正常运行？
- ❖ 系统需要应付（处理）哪些硬设备？
- ❖ 系统需要和哪些外部系统交互？
- ❖ 谁（或什么）对系统运行产生的结果感兴趣？
- ❖ 有没有自动发生的事件



识别执行者

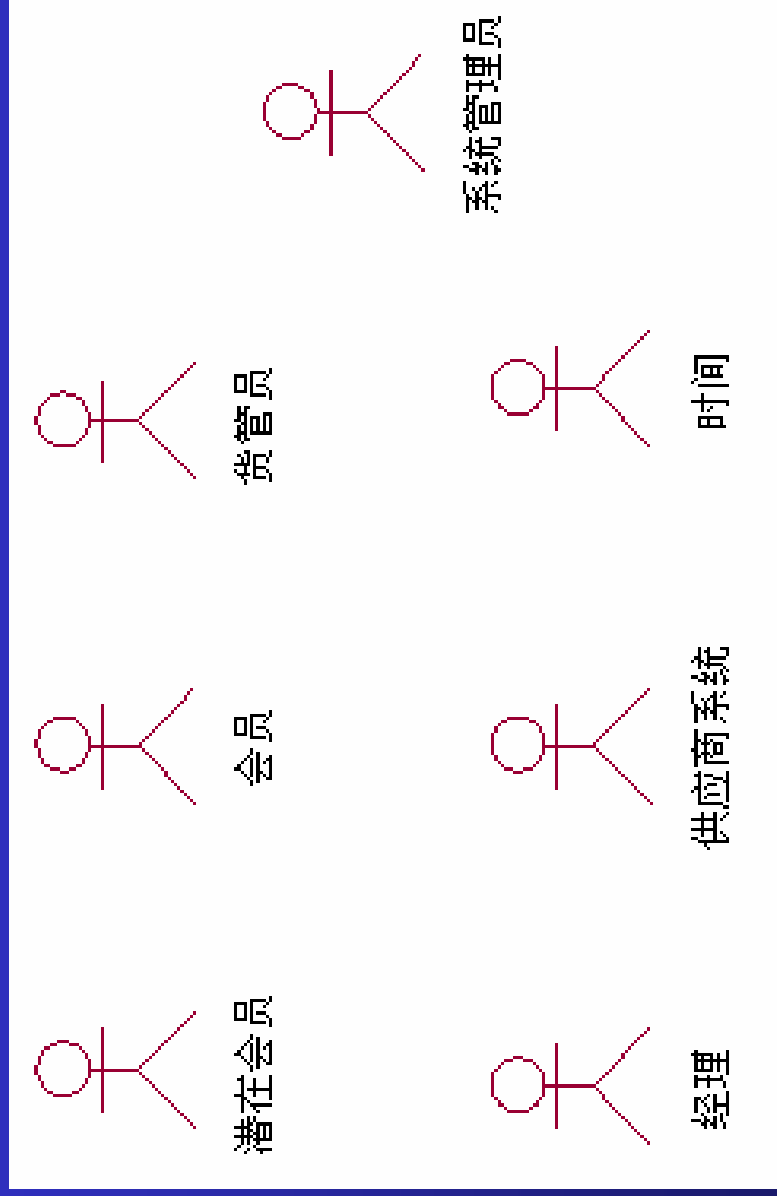
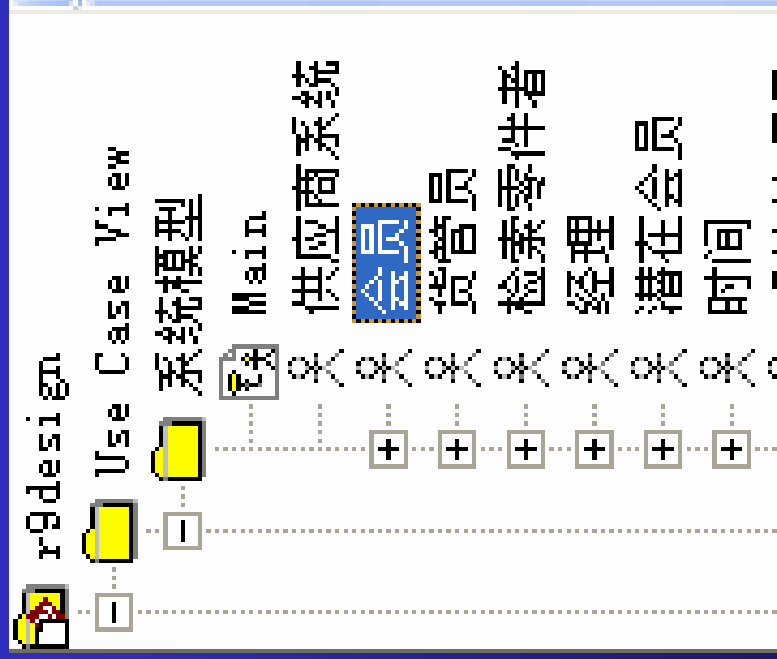
——零件销售系统

- ❖ 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

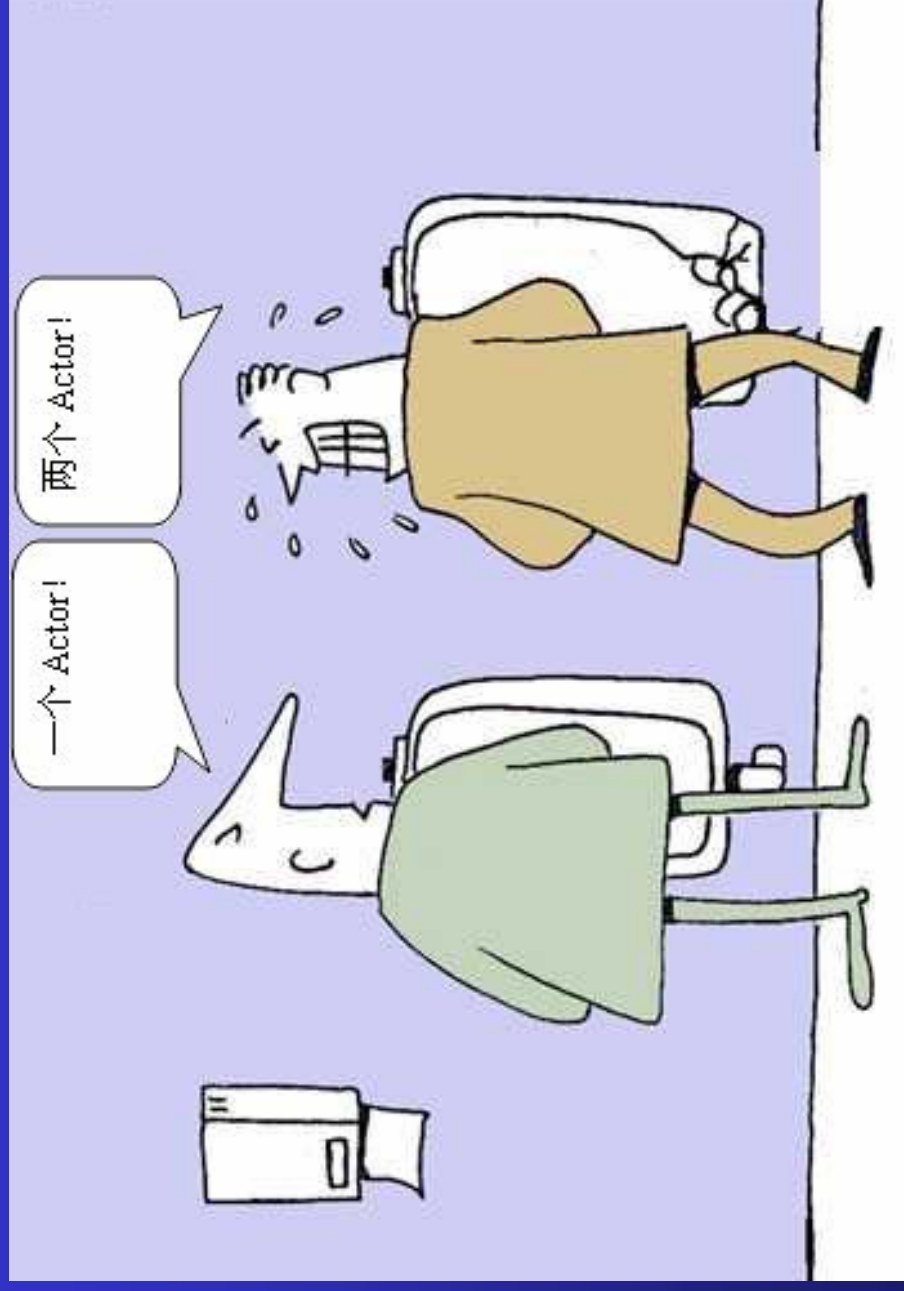


识别执行者

——候选执行者（工具演示）



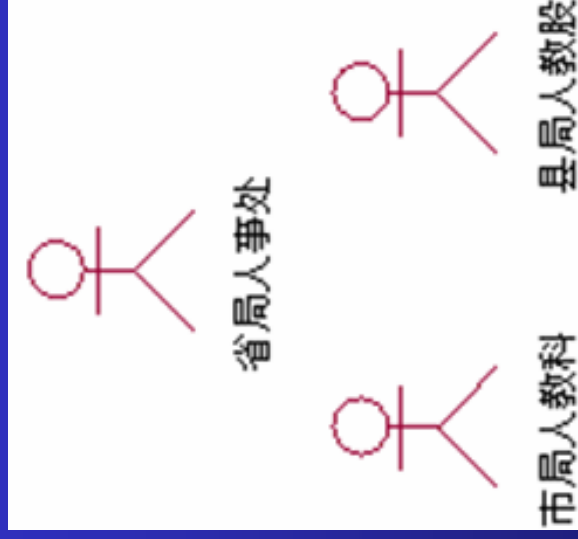
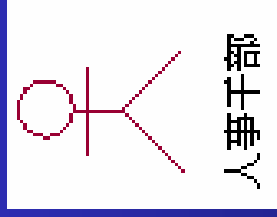
识别执行者



潜在会员？会员？多少个actor合适？



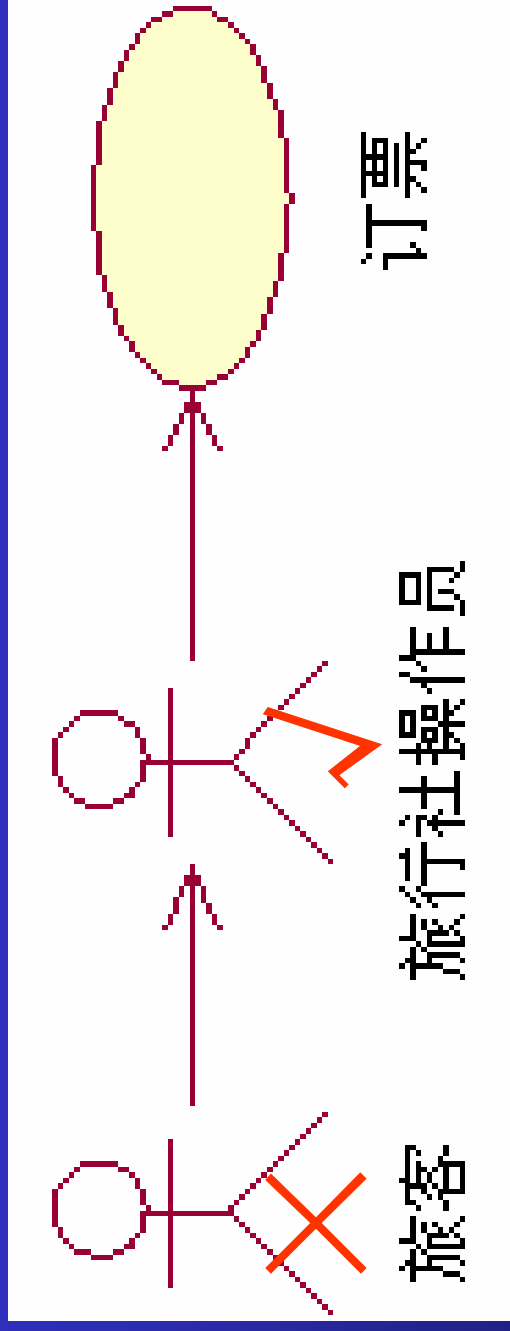
识别执行者



都对，不丢用例就行（宁多勿少）



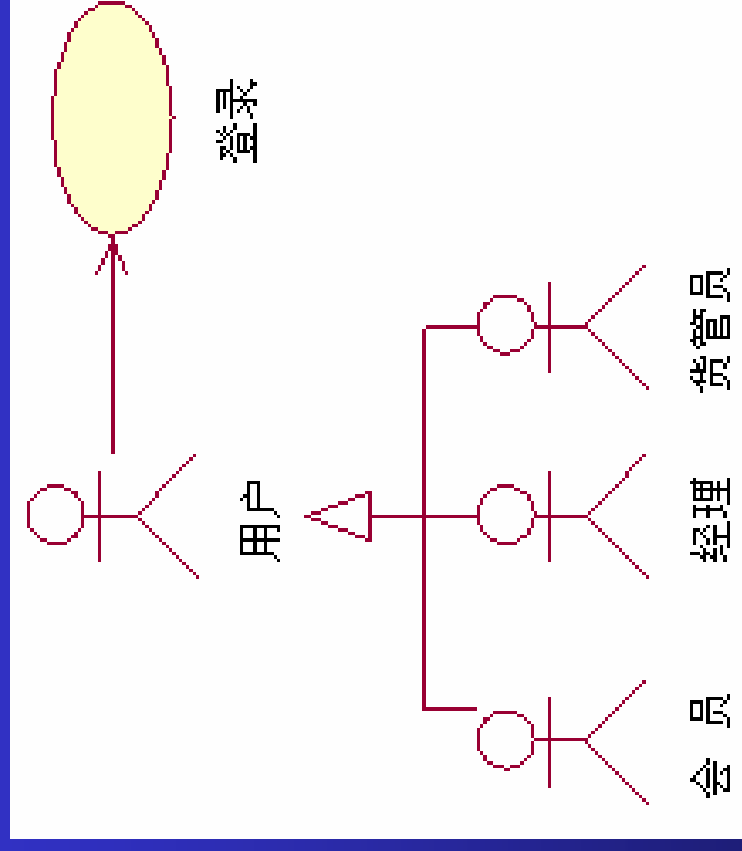
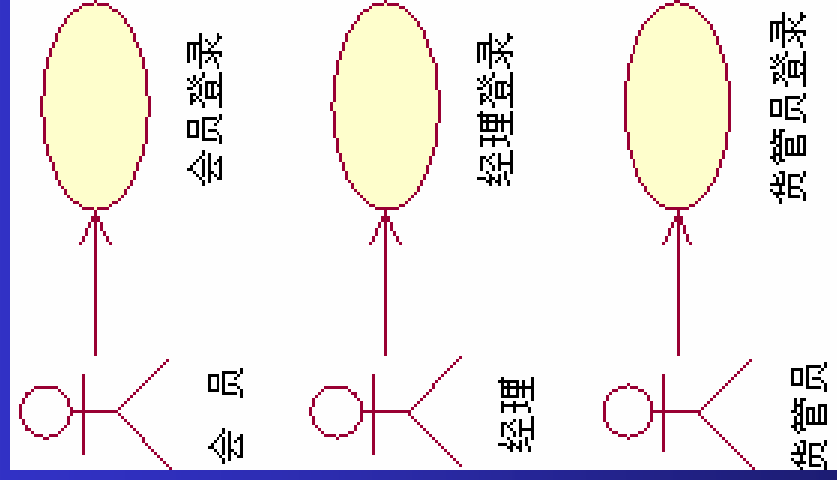
识别执行者



关键在边界，不在数量



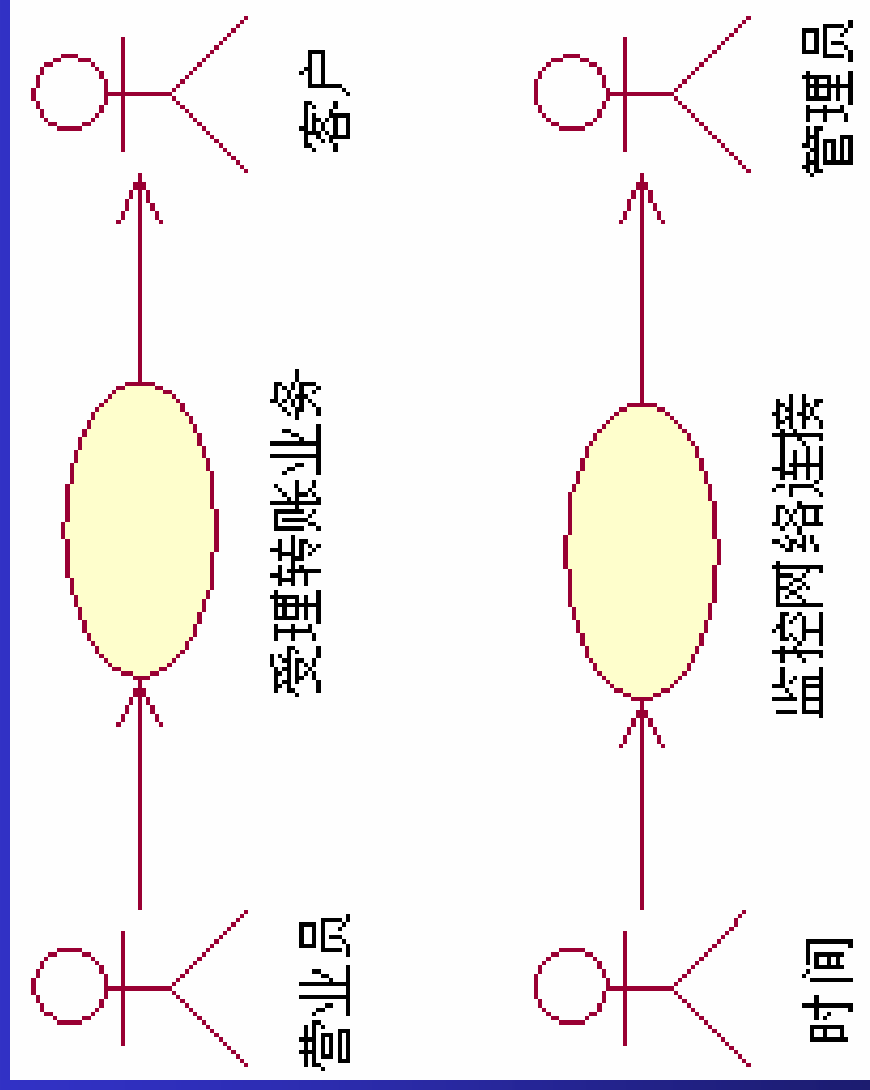
识别执行者



部分责任重叠——泛化出一个抽象角色



识别执行者



主执行者完成用例时可能需要辅助执行者



识别执行者

——项目实作

- 工作表格——列出系统的actor
- 时间：5分钟
- 记得写上自己的名字



步骤

- ❑ 识别执行者 (*)
- ❑ 识别用例 (*)
- ❑ 书写用例文档 (*)
- ❑ 识别用例的关系
- ❑ 用例的排序和分包



识别用例

——关键词：价值



识别用例

——用例

❖ 定义

用例实例是系统执行的一系列动作，这些动作将生成特定执行者可见的价值结果。一个用例定义一组用例实例。

❖ 通俗一些

执行者使用系统达到某个目标



识别用例

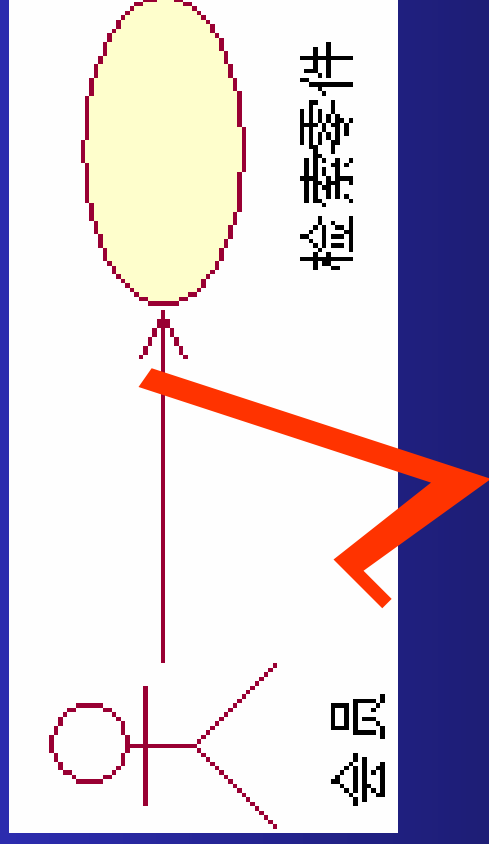
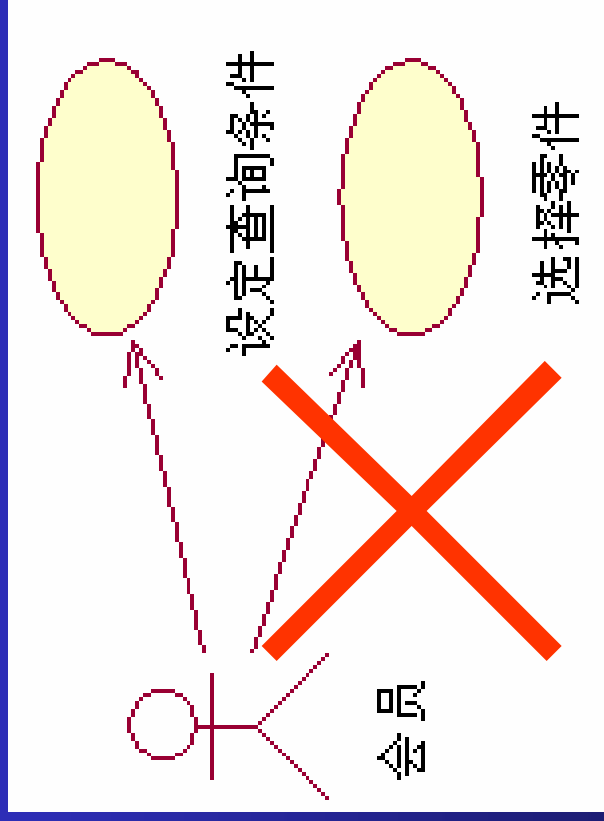
——用例要点

- 价值结果→有意义的目标
- 系统执行→价值结果由系统生成
- 执行者可见→业务语言，用户观点
- 一组用例实例→用例的粒度



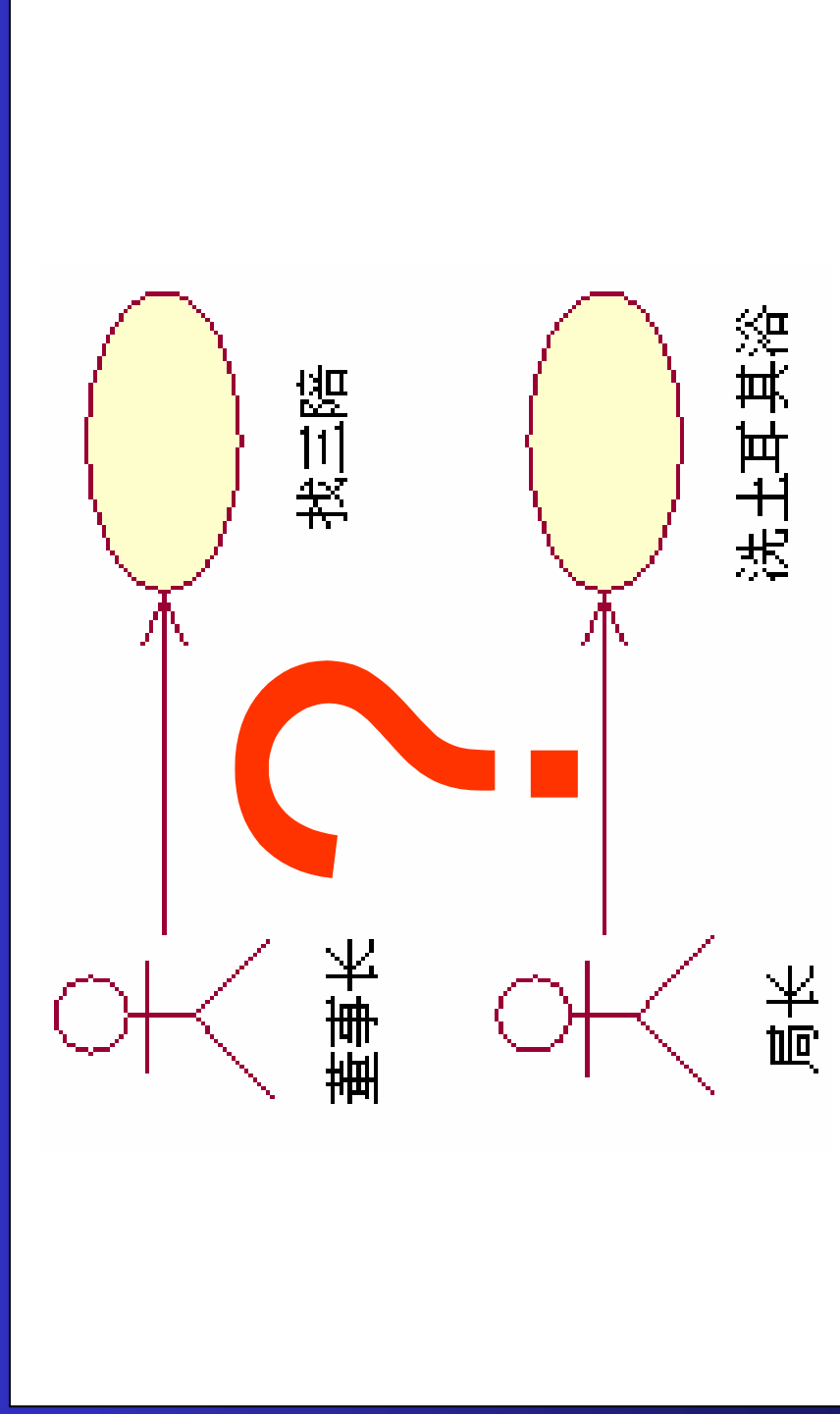
识别用例

——有意义的目标



识别用例

——价值结果由系统生成



识别用例

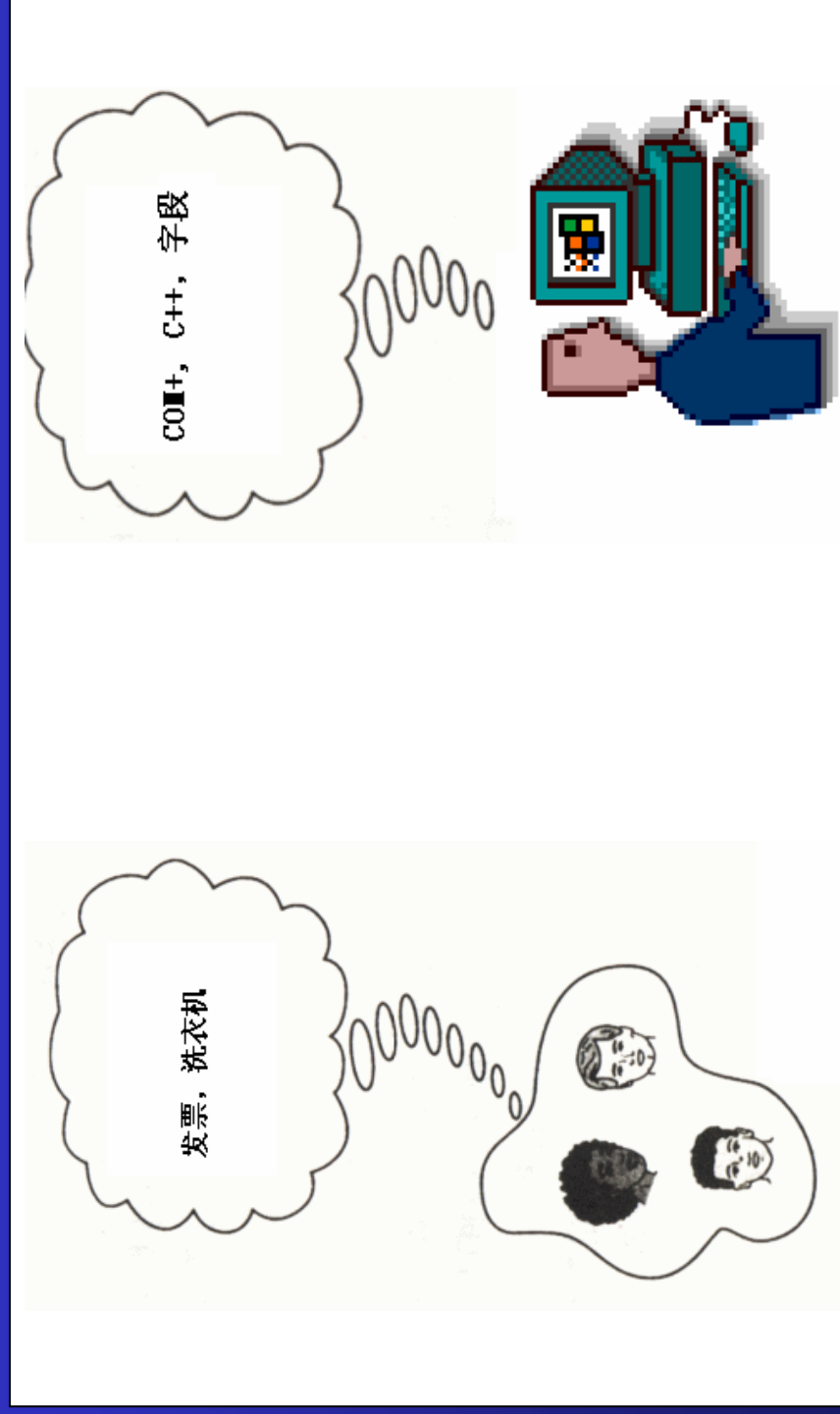
——价值结果由系统生成

局长专用：海滨市三陪小姐电话一览				
	姓名	籍贯	特征	手机
	小梅	四川	...	135012' 708
	阿宝	山西	...	139 12045466
	...			
	...			



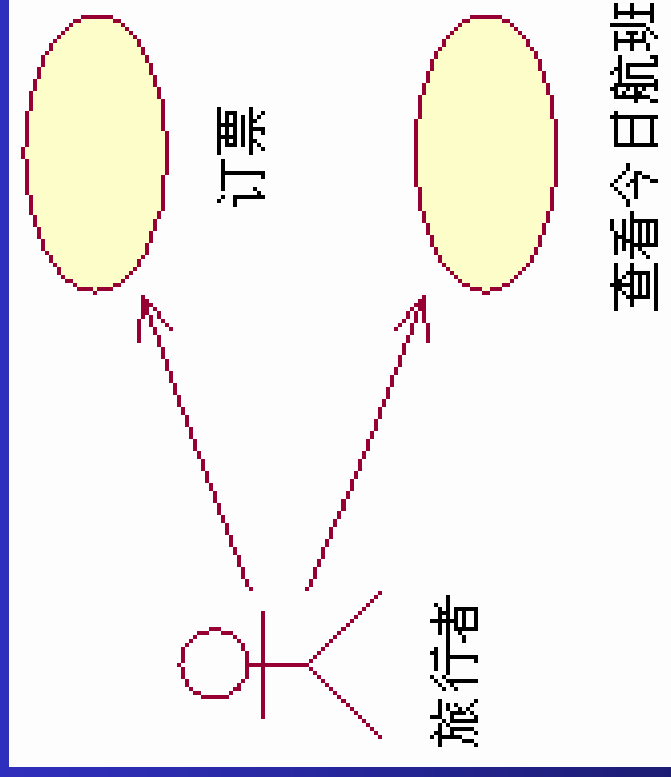
识别用例

——业务语言而非技术语言

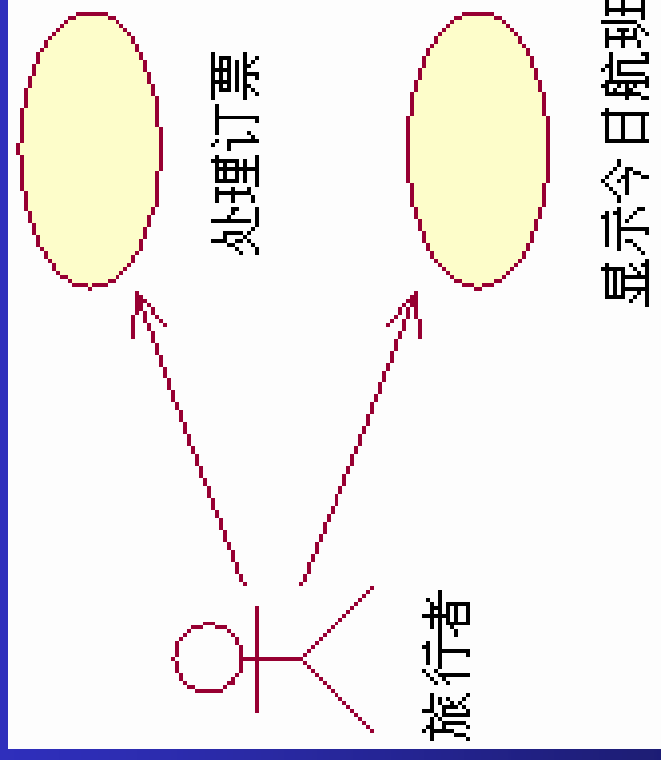


识别用例

——用户观点而非系统观点



用户观点



系统观点



识别用例

——用例命名：执行者视角

动词（+宾语）



状语



定语



识别用例

——用例命名：慎用弱动词弱名词

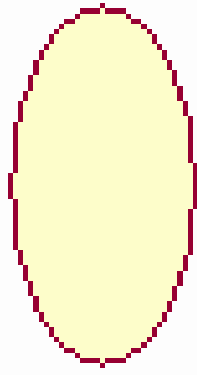
❖弱动词：进行、使用、复制、加载、重复...

❖弱名词：数据、报表、表格、表单、系统...

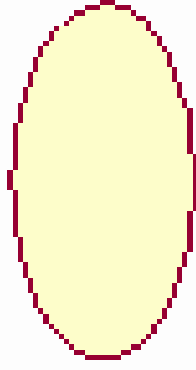
会掩盖真正的业务



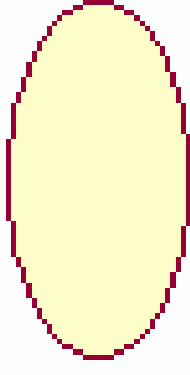
常见用例命名错误



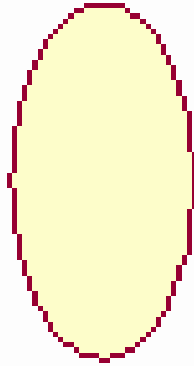
商品搜索



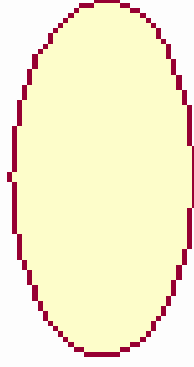
进行商品搜索



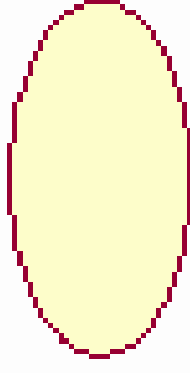
用例1



订单



调度模块

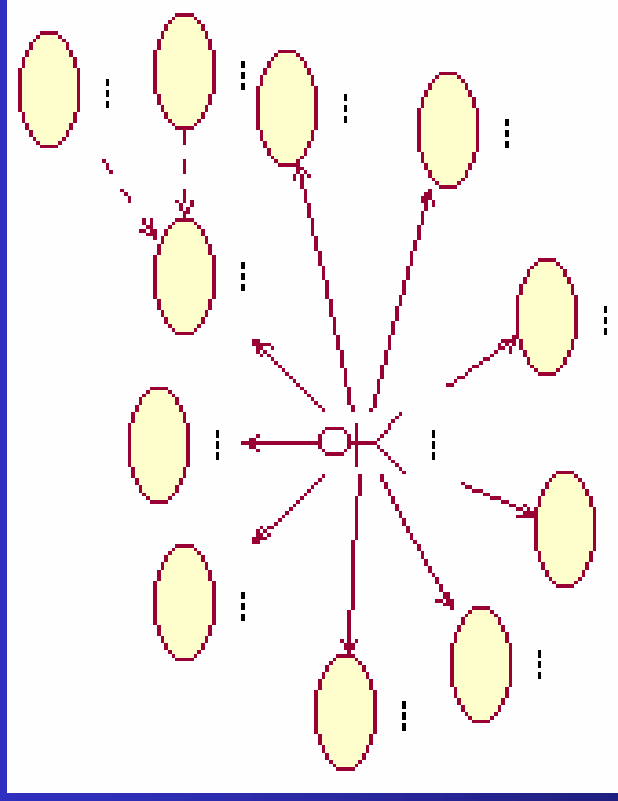


处理业务



识别用例

——用例的粒度（1）



用例要有路径，路径要有步骤。而这一切都是“可观测”的

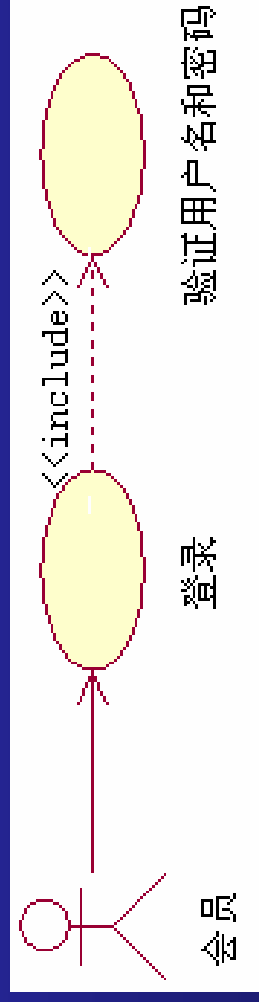
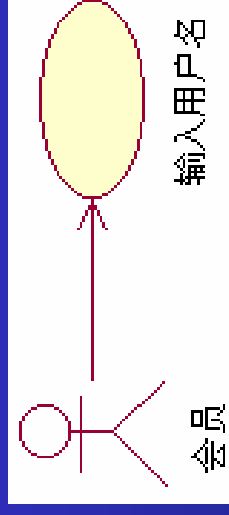
最常犯错误：把步骤当作用例



识别用例

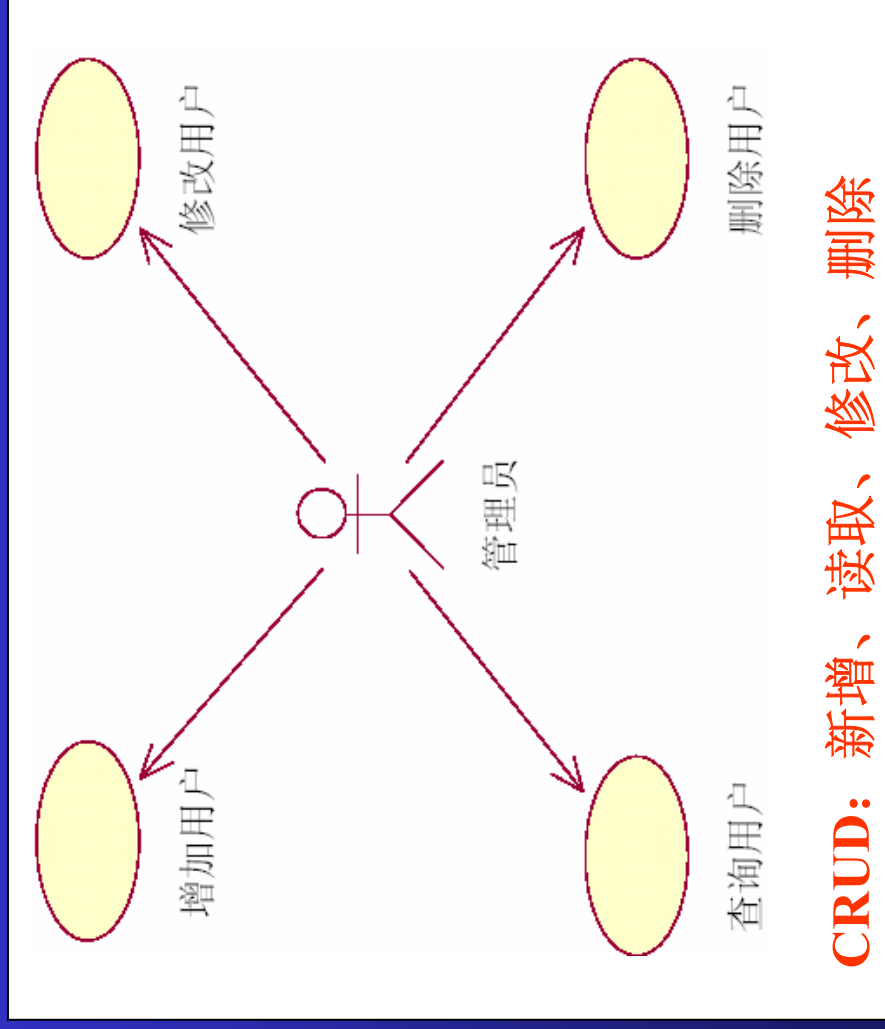
——用例的粒度（2）

- ❖ 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



识别用例

——用例的粒度（3）：四轮马车的错误



警惕！用有色眼镜看

所有业务最终都会成为CRUD

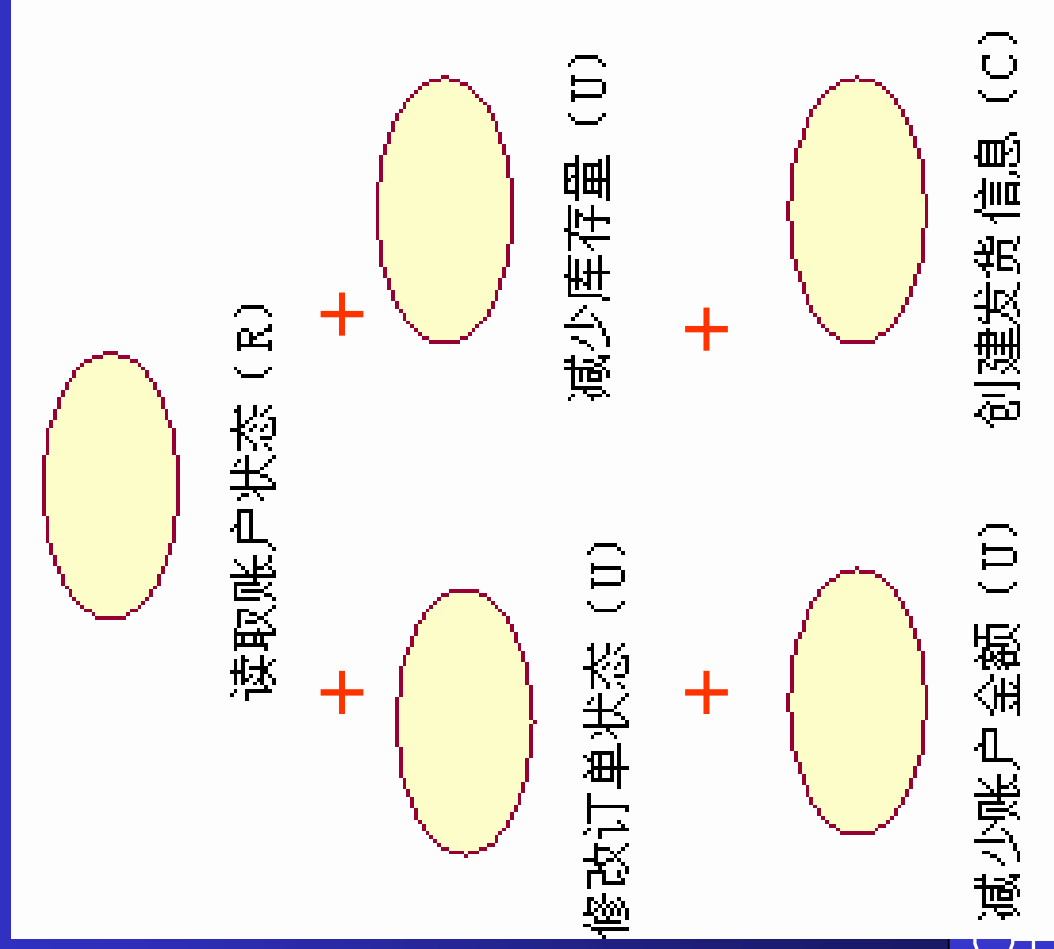
多问：为什么要CRUD？光

CRUD能为actor提供价值吗？

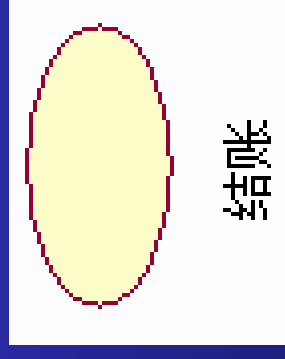


识别用例

——用例的粒度（4）：CRUD的背后



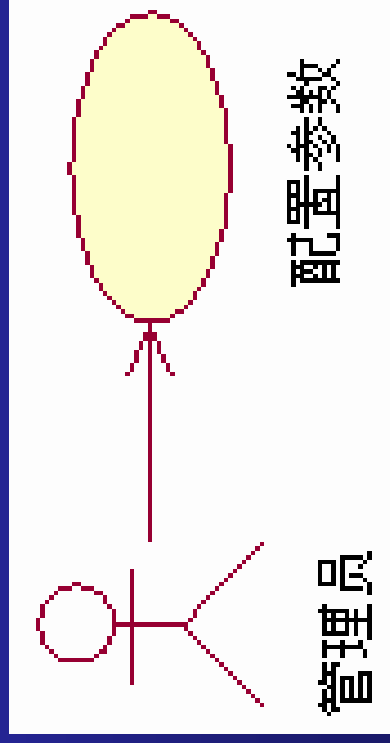
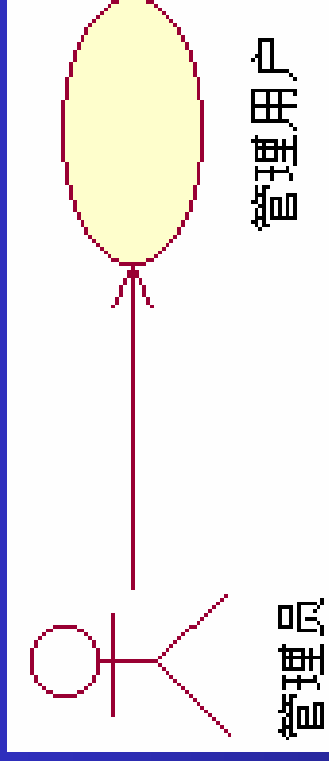
=



识别用例

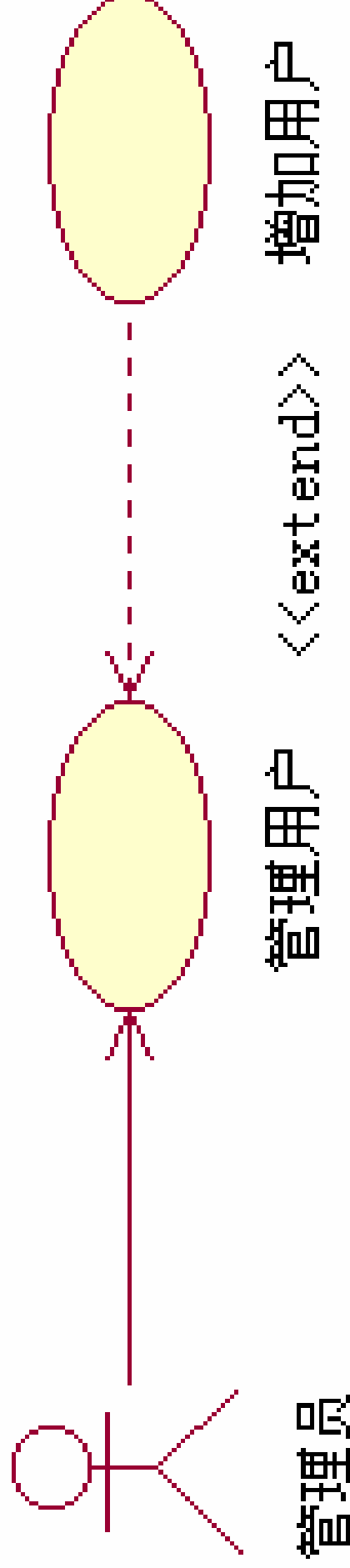
——用例的粒度（5）：如果确实是纯CRUD

- ❖ 如果CRUD不涉及复杂的交互，一个用例“管理××”即可
- ❖ 不管是C、R、U、D，都是为完成“管理”的目标
- ❖ 甚至很多种基本数据的管理都可以用一个用例表示



识别用例

——用例的粒度（6）：灵活处理CRUD



也可以把包含复杂交互的路径独立出去形成用例



识别用例

——讨论与练习



识别用例

执行者使用这个系统达到什么目标？



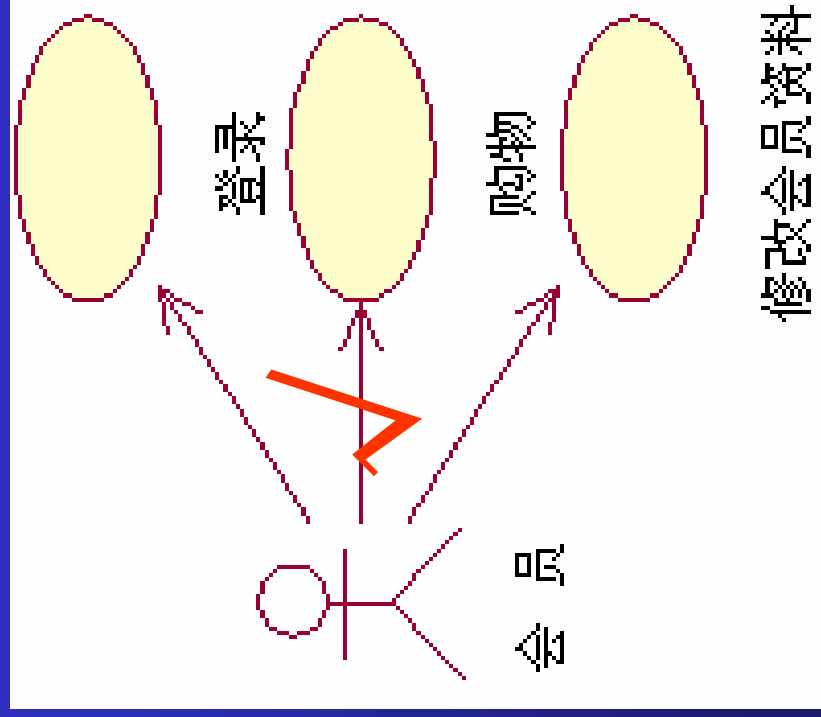
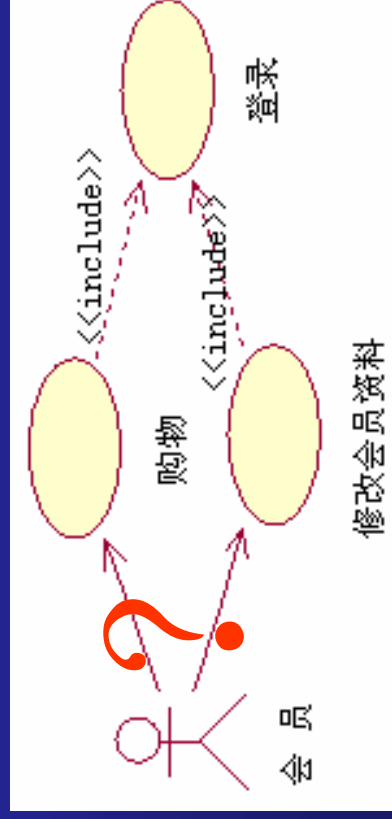
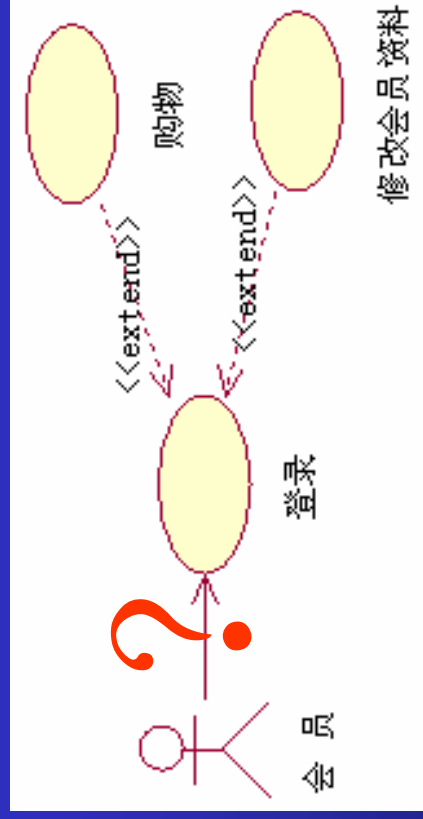
识别用例

——这些用例形式上对了，内容呢？



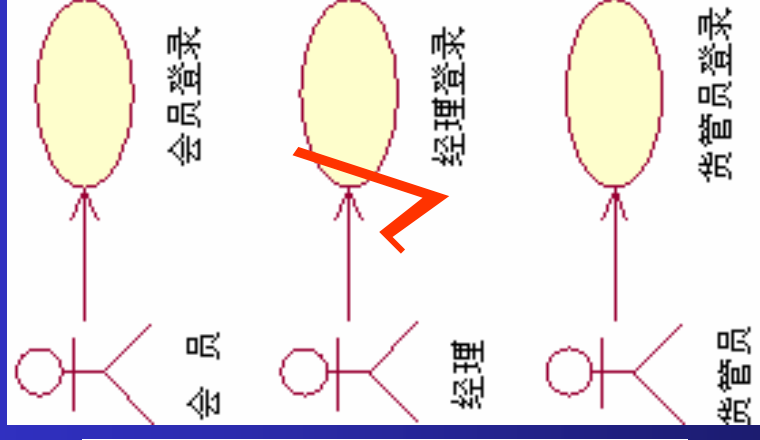
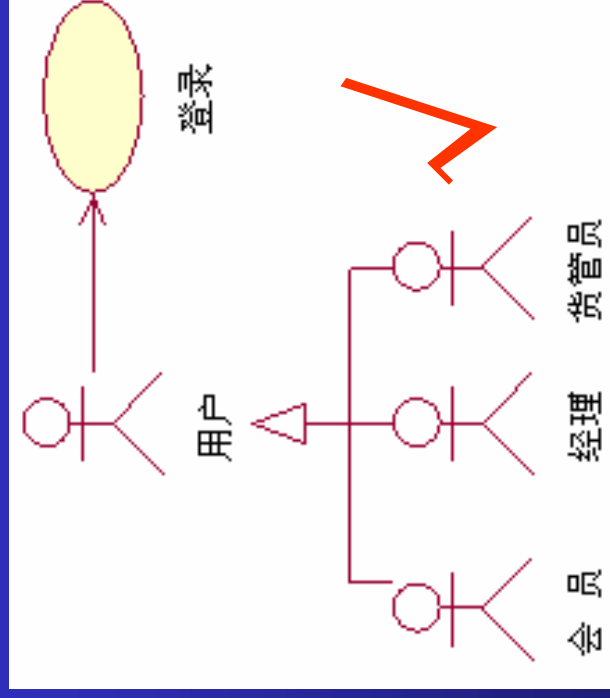
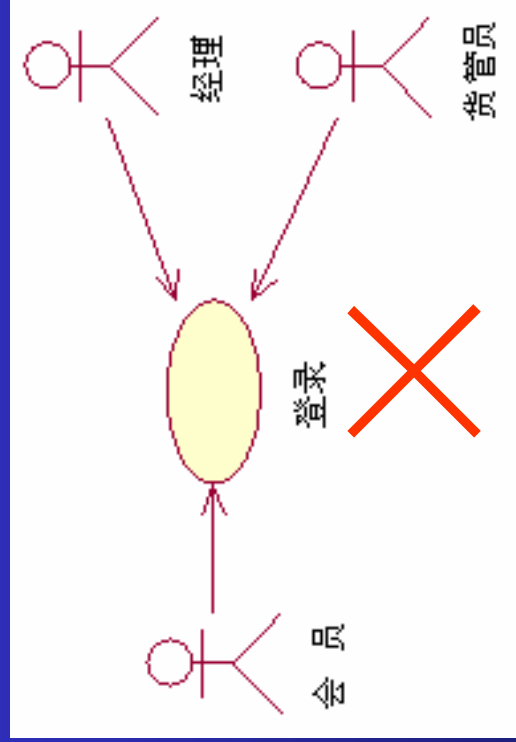
识别用例

——讨论（1）：登录怎么处理



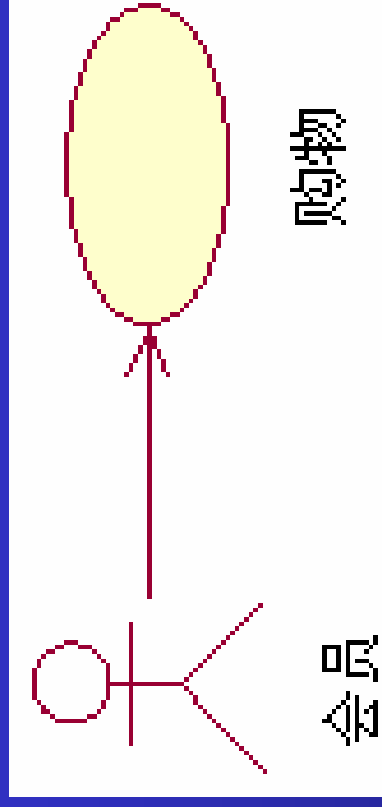
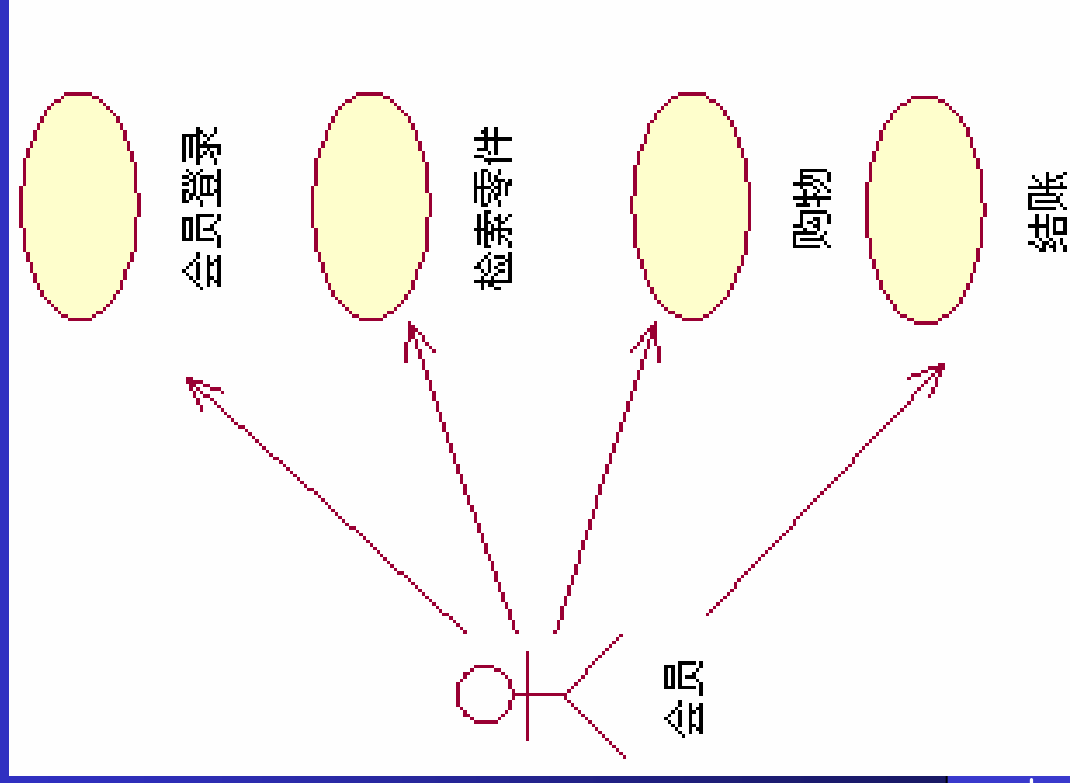
识别用例

——讨论（2）：几个登录？



识别用例

——讨论（3）：什么时候开始结束？



参考原则：

一个人，一个地方，一次完成
完成后可以愉快地离开

用户说了算，微观上是等同的

识别用例

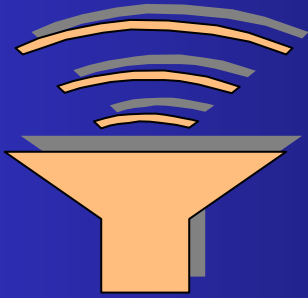
——项目实务

- 工作表格——列出系统的用例
- 时间：10分钟
- 记得写上自己的名字



识别用例

——注意



现在还在外面，不要急着乱塞东西！



识别用例

——核心！=卖点，稍安勿躁



识别用例



题外话

——为什么总是急于加入细节



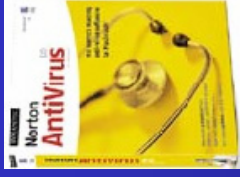
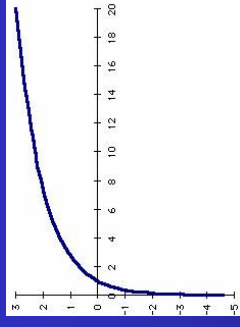
虚假的安全感



识别用例

——用例什么时候不是项目成败关键

- ❖ 系统用户类型少，接口（非内部功能）少
- ❖ 科学计算、仿真、杀毒、编译、内存管理...



- ❖ 非功能需求和设计约束占主导
- ❖ Google, Quake...



步骤

- ❑ 识别系统边界和执行者 (*)
- ❑ 识别用例 (*)
- ❑ 书写用例文档 (*)
- ❑ 识别用例的关系
- ❑ 用例的排序和分包

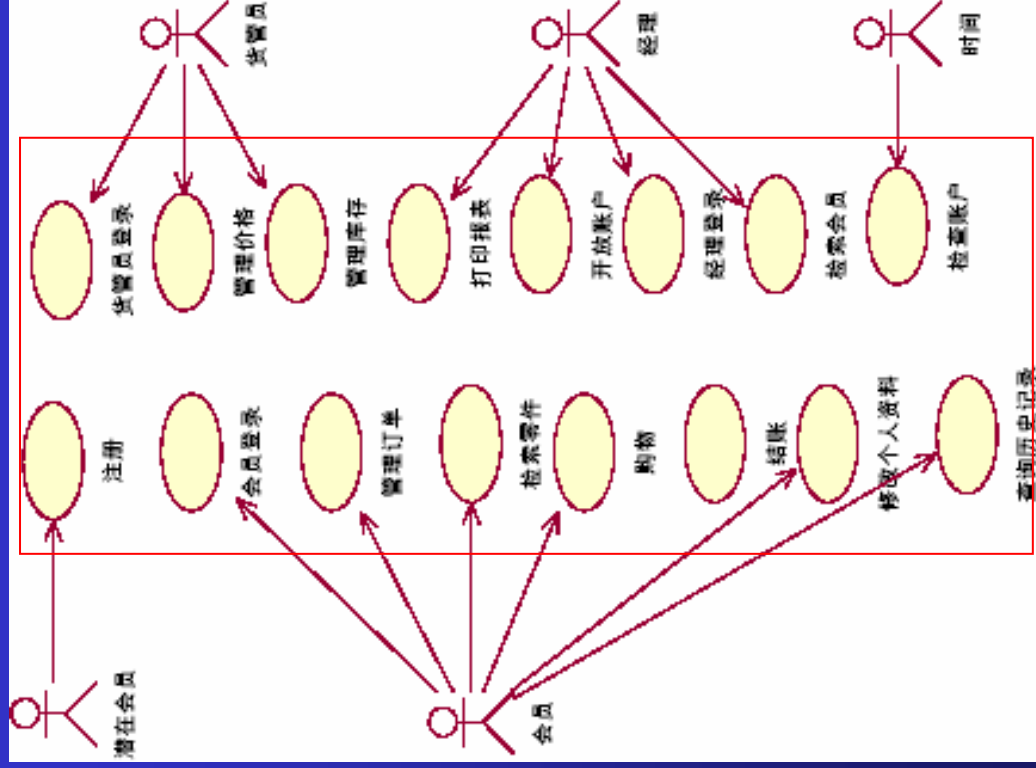


用例文档：更进一步的精度

用例图可以作为用例文档的总图

进一步的精度：有层次的文档

文档中每一句话都有其价值



书写用例文档

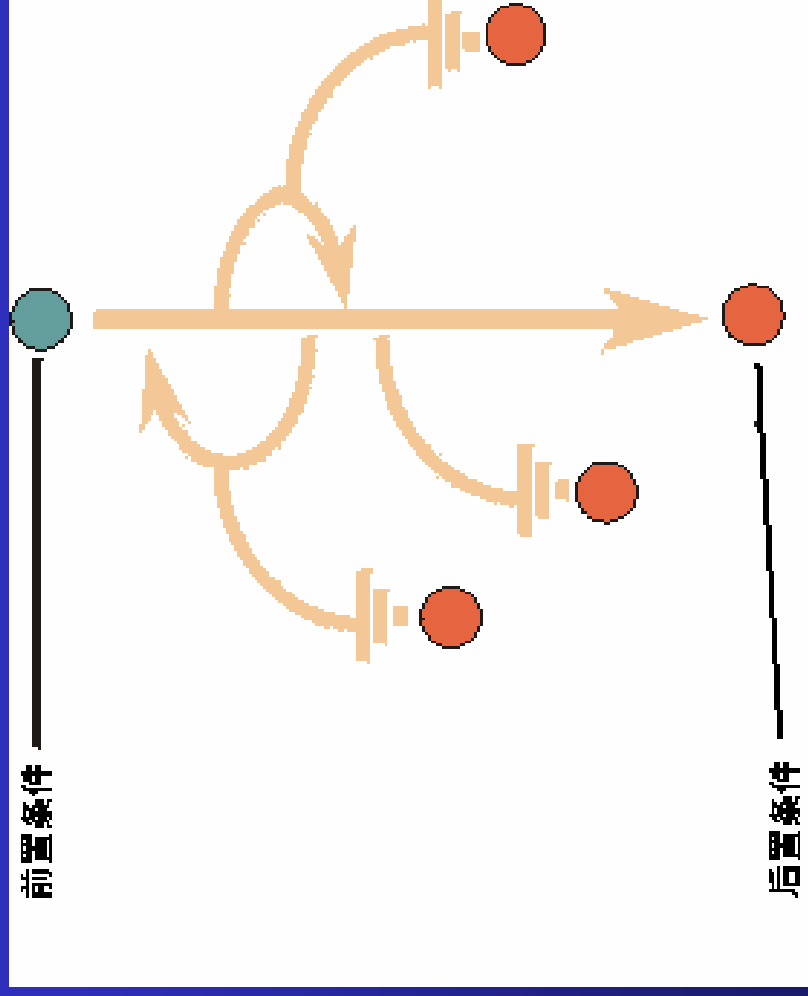
——用例模板

- ◆ 用例编号：用例名
- ◆ 执行者
- ◆ 前置条件
- ◆ 后置条件
- ◆ 涉及利益
- ◆ 基本路径
 - ◆ 1..... ×××××
 - ◆ 2..... ×××××
 - ◆ 3..... ×××××
- ◆ 扩展
 - ◆ 2a. ×××××
 - ◆ 2a1.... ×××××
- ◆ 字段列表
- ◆ 非功能需求
- ◆ 设计约束
- ◆ 业务规则
- ◆ 待解决问题



书写用例文档

——前置、后置条件（起点和终点）



开始用例前所必需的系统及其环境的状态

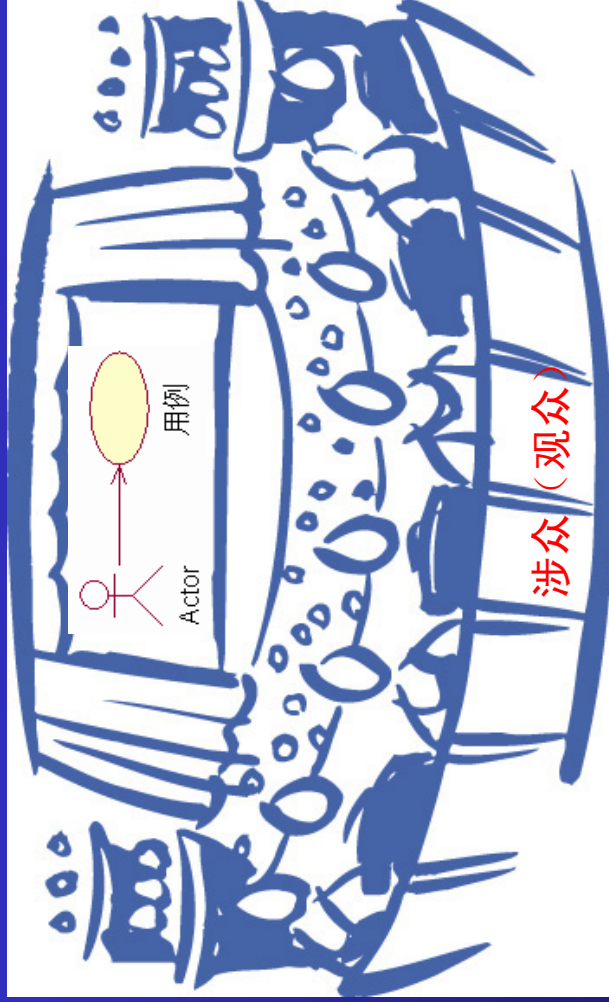
注意：必须是系统能检测到的

用例成功结束后系统应该具备的状态



书写用例文档

——涉众利益（保证内容的正确）



看戏

西门庆和潘金莲？



书写用例文档

——涉众利益（保证内容的正确）

- 最终用户
- 客户
- 政府、法律
- 开发人员
- ...

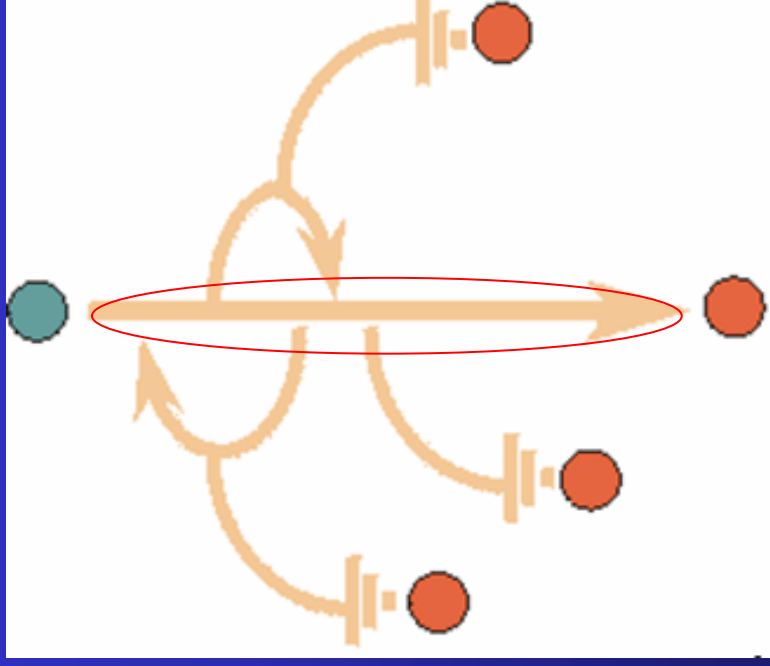


受系统影响的人——用例是平衡他们利益的契约



书写用例文档

——基本路径



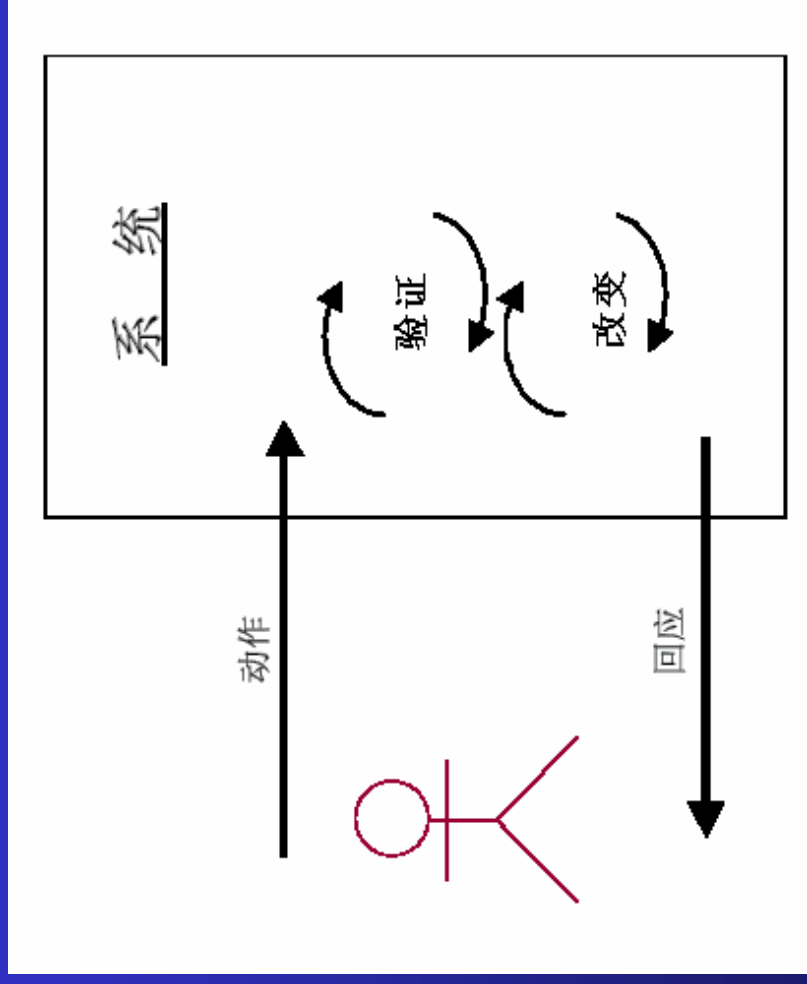
“想看到”不等于“无错”

核心的核心：客户最想看到、最关心的路径



书写用例文档

——写什么：用例交互四步曲



这些才是“需求”

写：可观测的、体现客户利益的文字



书写用例文档

——路径交互步骤的描述

- ❖ 只书写“可观测”的（说人话）
- ❖ 使用主动语句
- ❖ 句子必须以执行者或系统作为主语
- ❖ 每一句都要朝目标迈进
- ❖ 分支和循环
- ❖ 不要涉及界面细节



书写用例文档

——路径交互步骤的描述（1）

- ❖ 系统通过ADO建立数据库连接，~~发送SQL查询语句，从“零件”表查询...~~
- ❖ 系统按照查询条件搜索零件

只书写“可观测”的



书写用例文档

——路径交互步骤的描述（2）




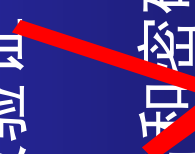
- ❖ 欧文从贝克汉姆处得到传球，守门员...
✗
- ❖ 贝克汉姆传球给欧文，欧文射门，守门员扑救....
✓

主动语句——球在谁那里？



书写用例文档

——路径交互步骤的描述（2）

- ❖ 系统从会员处获取用户名和密码 
- ❖ 会员提交用户名和密码 
- ❖ 用户名和密码被验证 
- ❖ 系统验证用户名和密码 

使用主动语句



书写用例文档

——路径交互步骤的描述（3）

- ❖ 执行者 × × × × × × × ×
- ❖ 系统 × × × × × × × ×
- ❖ 系统 × × × × × × × ×
- ❖ 执行者 × × × × × × × ×

句子必须以执行者或系统作为主语



书写用例文档

——路径交互步骤的描述（4）

- ❖ 执行者填写姓名
- ❖ 执行者填写电话
- ❖ 执行者填写联系地址
- ❖ 执行者提交
- ❖ ×××××××



每一句都要朝目标迈进



书写用例文档

——路径交互步骤的描述（5）

❖ 分支：放到扩展路径

❖ 循环：直接描述

分支和循环



书写用例文档

——路径交互步骤的描述（6）

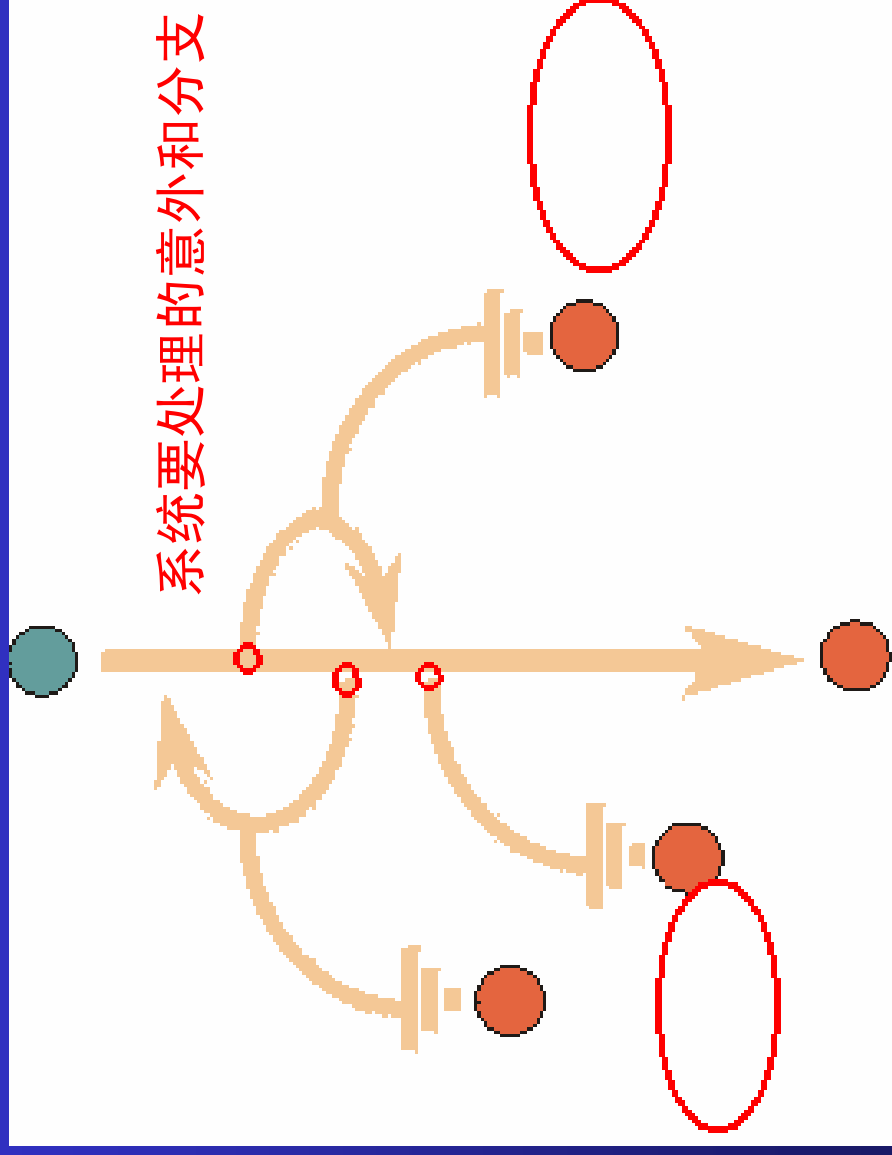
- ❖ 会员从下拉框中选择类别
- ❖ 会员在相应文本框中输入查询条件
- ❖ 会员点击“确定”按钮
- ❖

不要涉及界面细节



书写用例文档

——扩展



书写用例文档

——识别扩展点的思路

- ❖ 执行者的选择
- ❖ 系统验证
- ❖ 步骤失败



uctext.pdf

注意：必须是系统能感知的



书写用例文档

——字段列表

- 
1. 会员提交查询条件
 2. 系统.....



1. 查询条件可以是零件的类别、编号、几何特征或它们的组合



书写用例文档

——业务规则

3. 潜在会员提交注册信息。

?

4. 系统验证注册信息充分。

4. 公司名、联系人、电话是必需的。



书写用例文档

——非功能需求（URPS）

1. 会员提交查询条件

2. 系统按查询条件检索零件

?

可用性
可靠性
性能
可支持性

2. 检索时间不能超过5秒。



书写用例文档

——设计约束

5. 系统合计订单总价

6. 系统显示收费明细

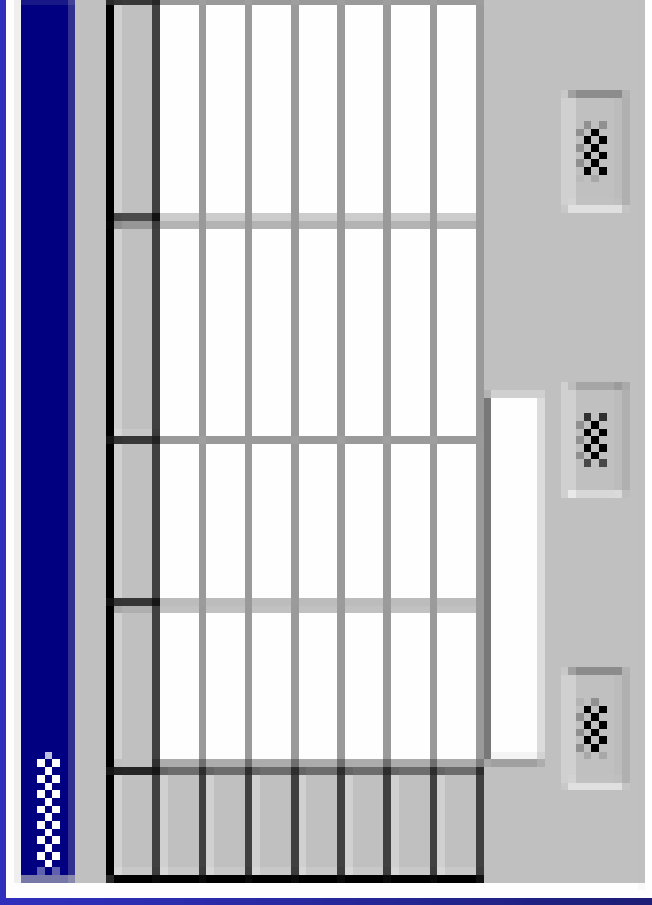


6. 以下图片为客户要求的界面布局



书写用例文档

——用例交互步骤的验证



可通过界面原型来帮助用户验证交互步骤



书写用例文档

——请大家动手做

- 我先带大家写一个
- 然后大家再自己写一个
- 写上名字，12分钟以后交上来



书写用例文档

——讨论与练习

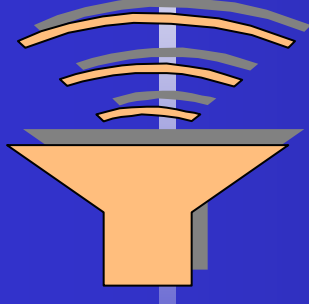


步骤

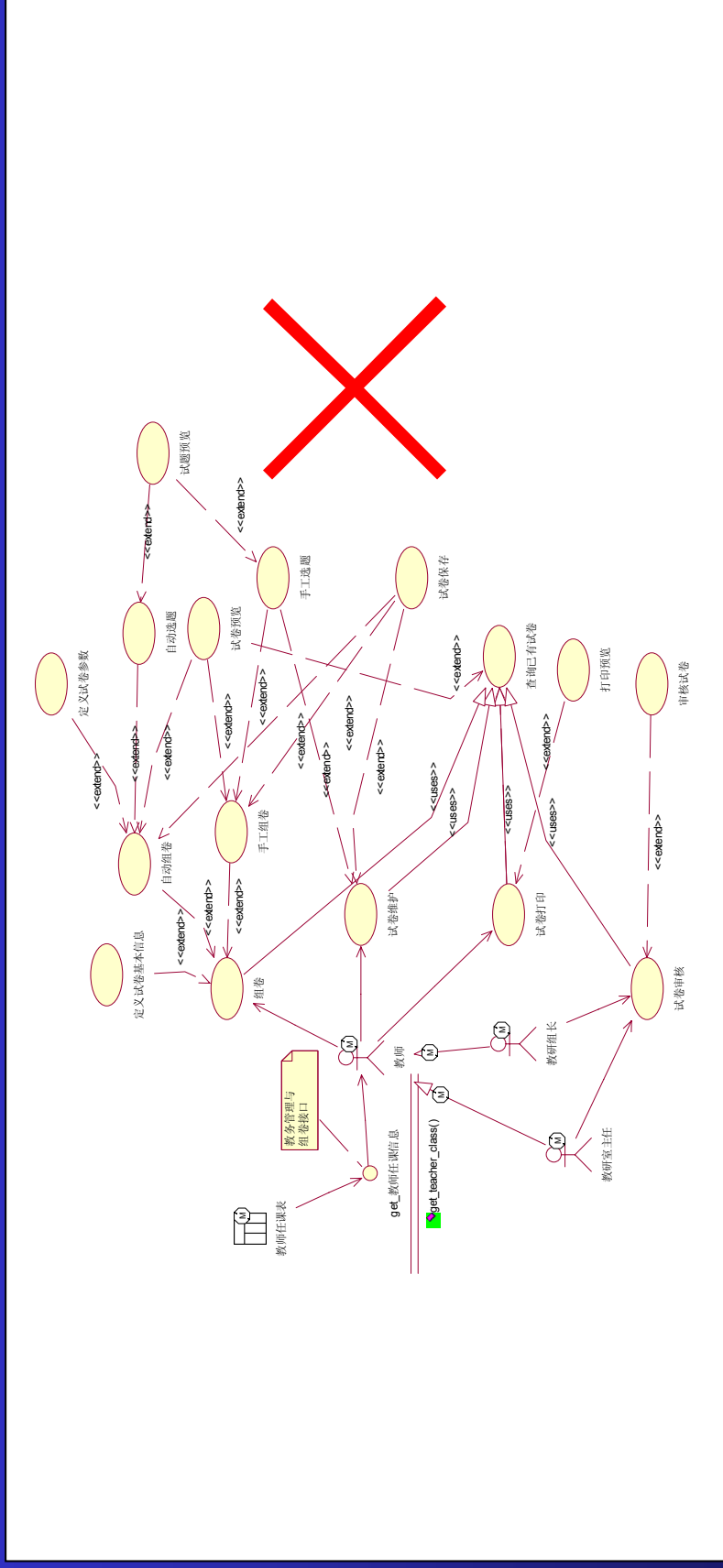
- 识别系统边界和执行者 (*)
- 识别用例 (*)
- 书写用例文档 (*)
- 识别用例的关系
- 用例的排序和分包



识别用例的关系



——注意

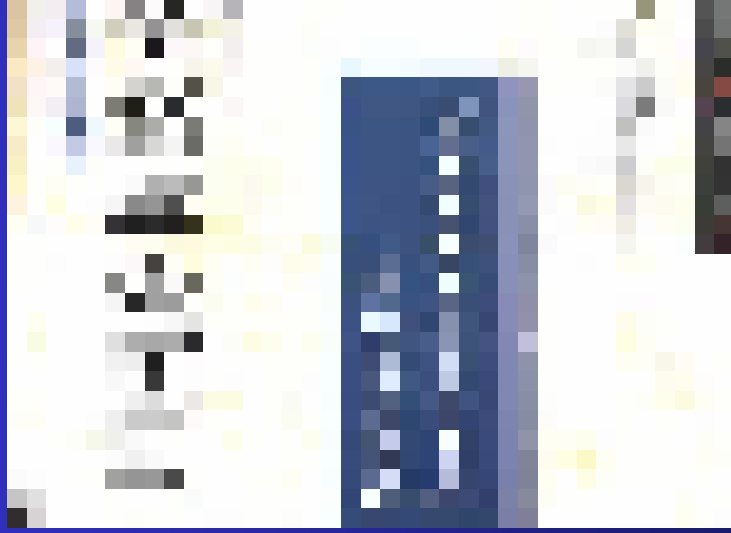


不要滥用用例关系!!!



识别用例的关系

——某些书也有误导

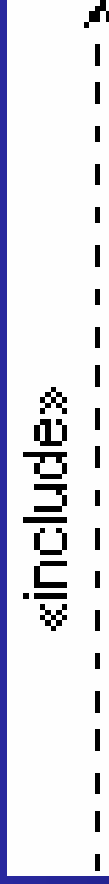


识别用例的关系

——用例的关系



扩展



包含



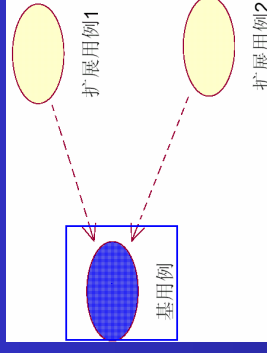
泛化



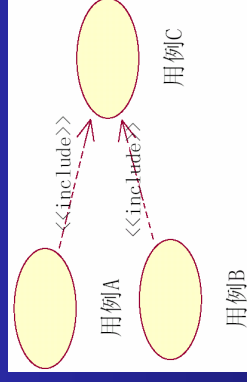
识别用例的关系

——通过关系整理用例文档

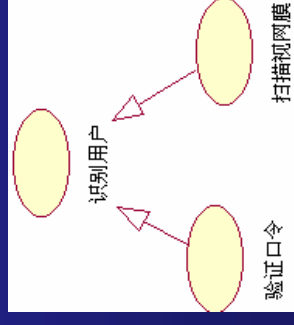
❖ 扩展：分离扩展路径



❖ 包含：提取公共步骤，便于复用



❖ 泛化：同一业务目的的不同技术实现



识别用例的关系

——扩展

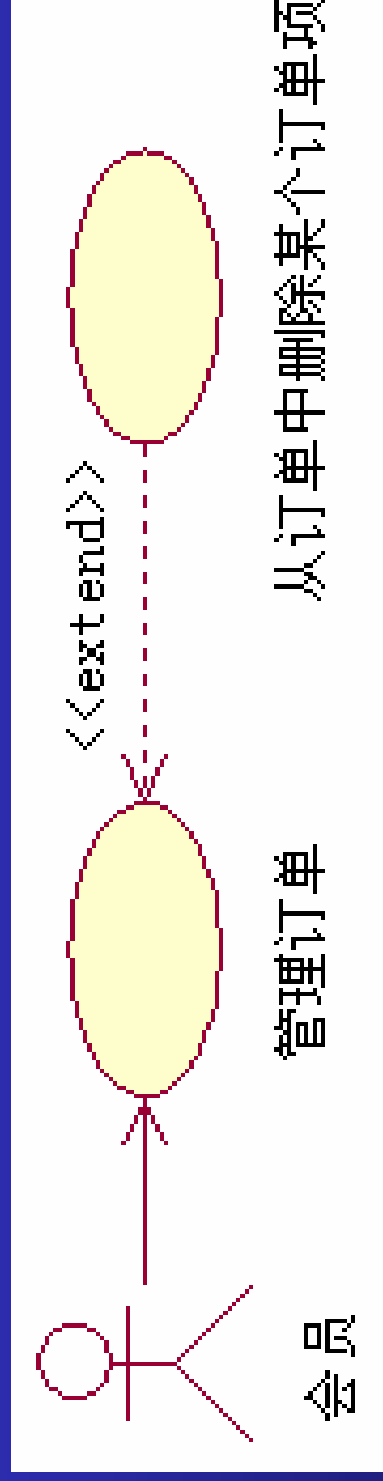


不上厕所，也能赶路



识别用例的关系

——扩展关系举例



识别用例的关系

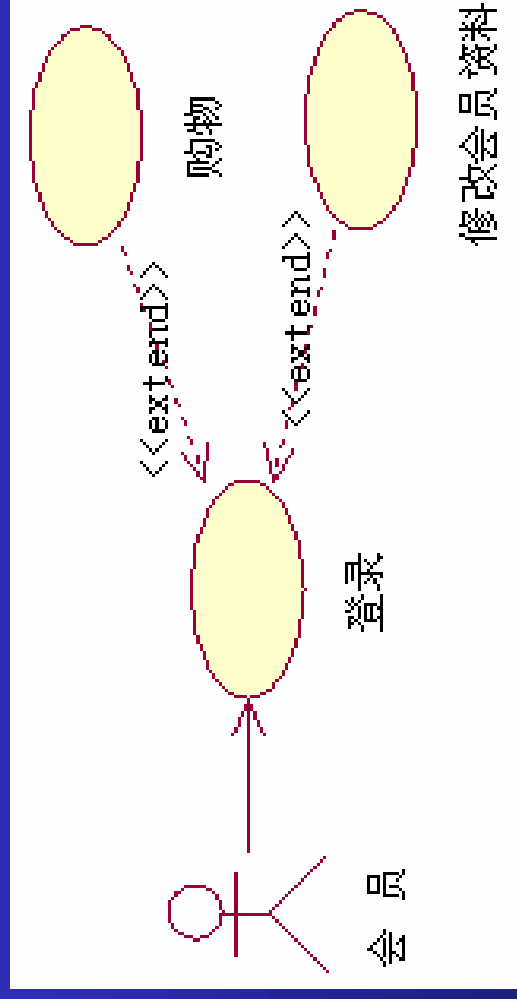
——何时使用扩展关系

- ❖ 扩展路径步骤多
- ❖ 扩展路径内部还有扩展点——扩展之扩展
- ❖ 扩展路径未定或容易变化——分离以“冻结”基用例



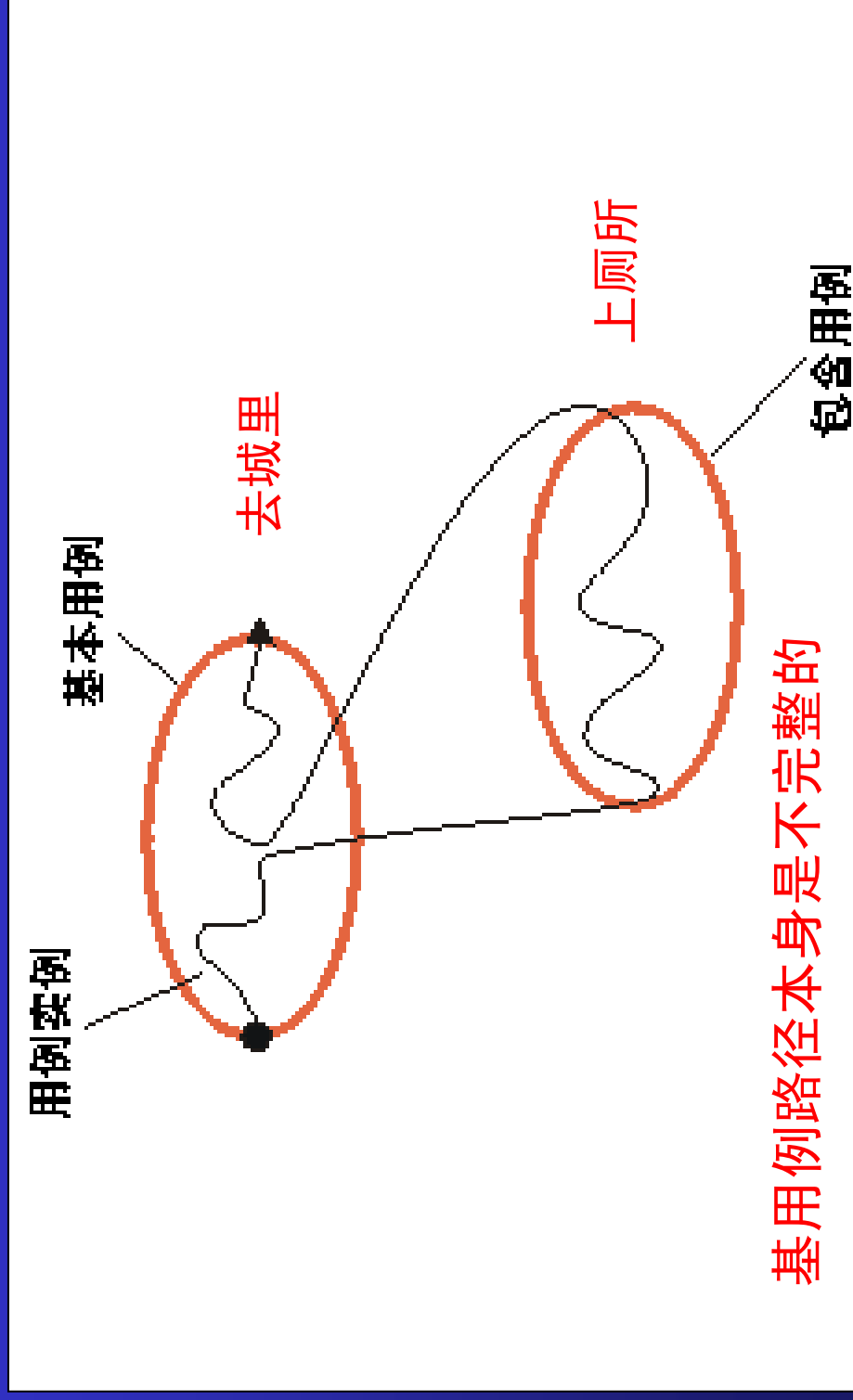
识别用例的关系

——扩展关系的误用



识别用例的关系

——包含



识别用例的关系

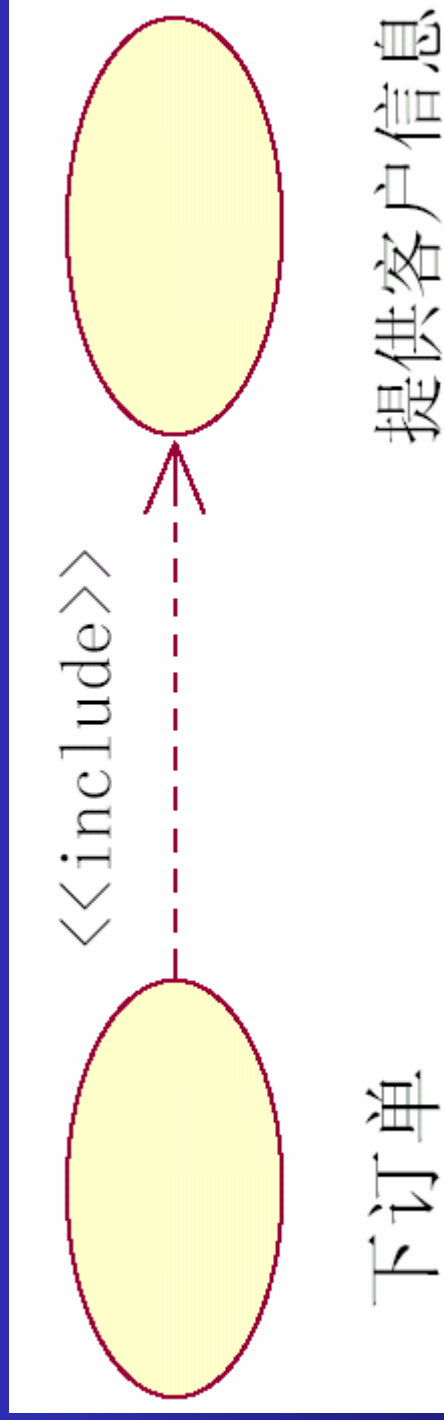
——包含



不上厕所，这段路不完整

识别用例的关系

——包含关系举例



识别用例的关系

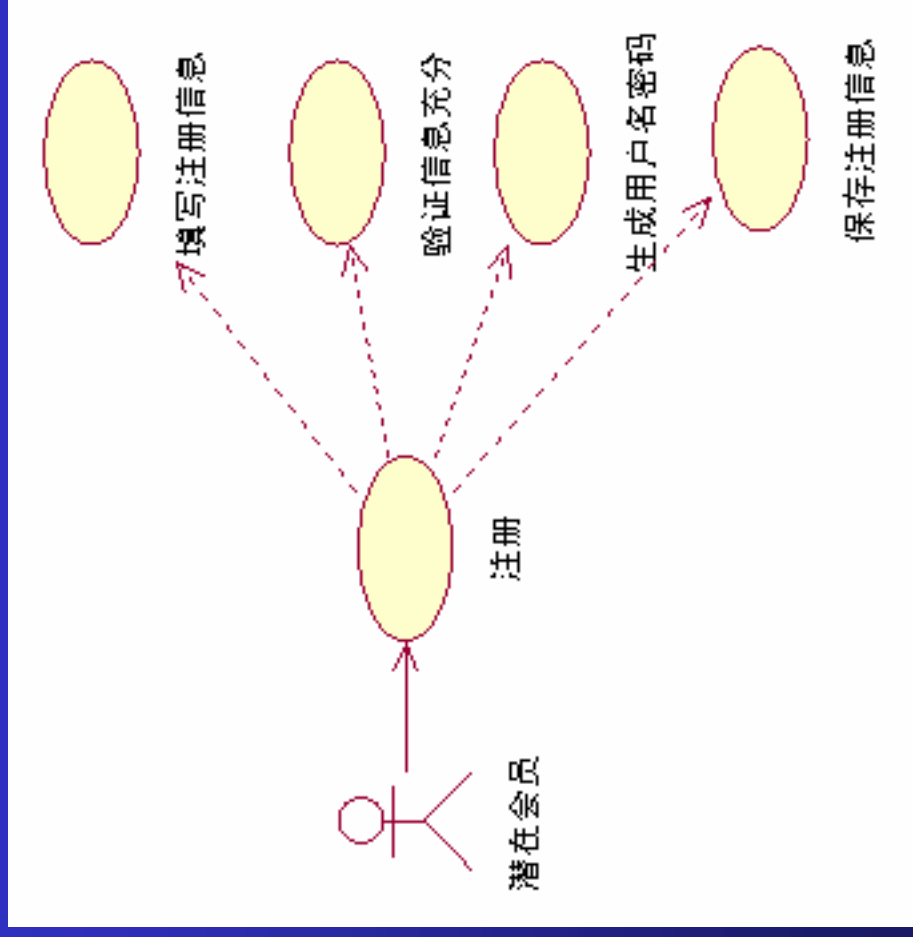
——何时使用包含关系

- ❖ 某些步骤在多个用例重复出现，且单独形成价值
- ❖ 用例的步骤较多时，可以用Include简化（慎用）



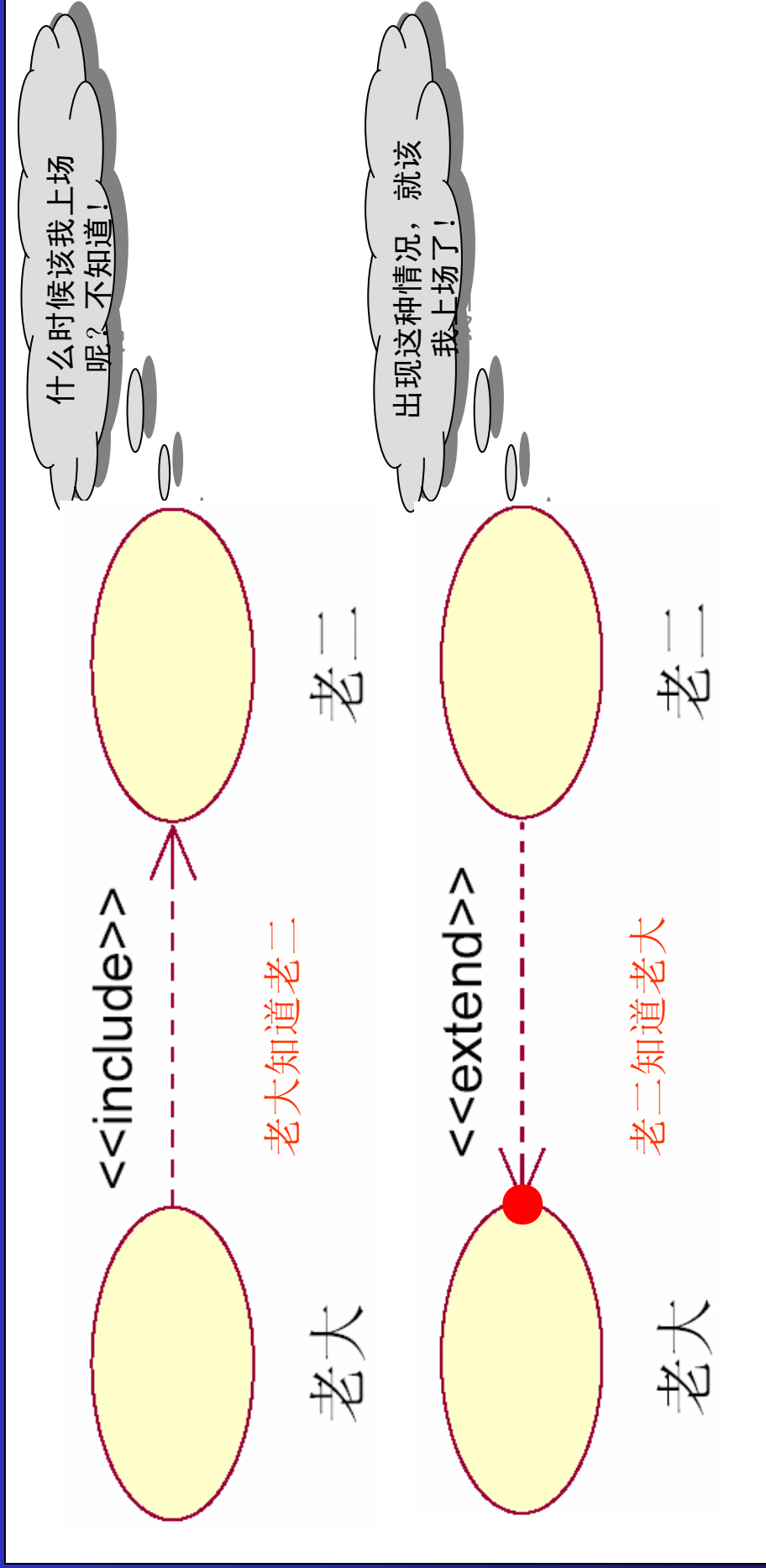
识别用例的关系

——包含关系的误用



识别用例的关系

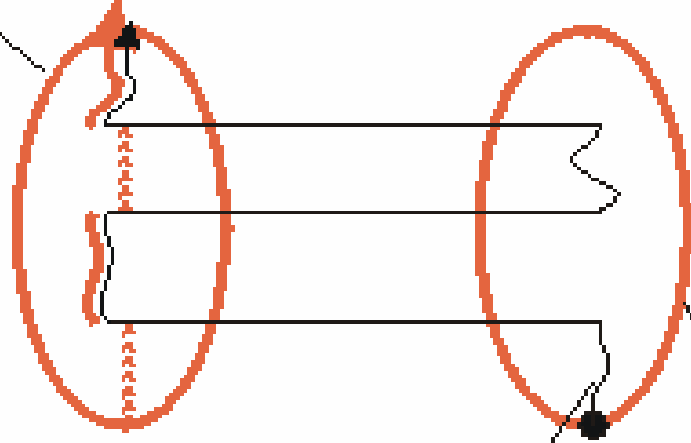
——扩展 vs. 包含的可见性



识别用例的关系

——泛化

父用例



用例实例

子用例

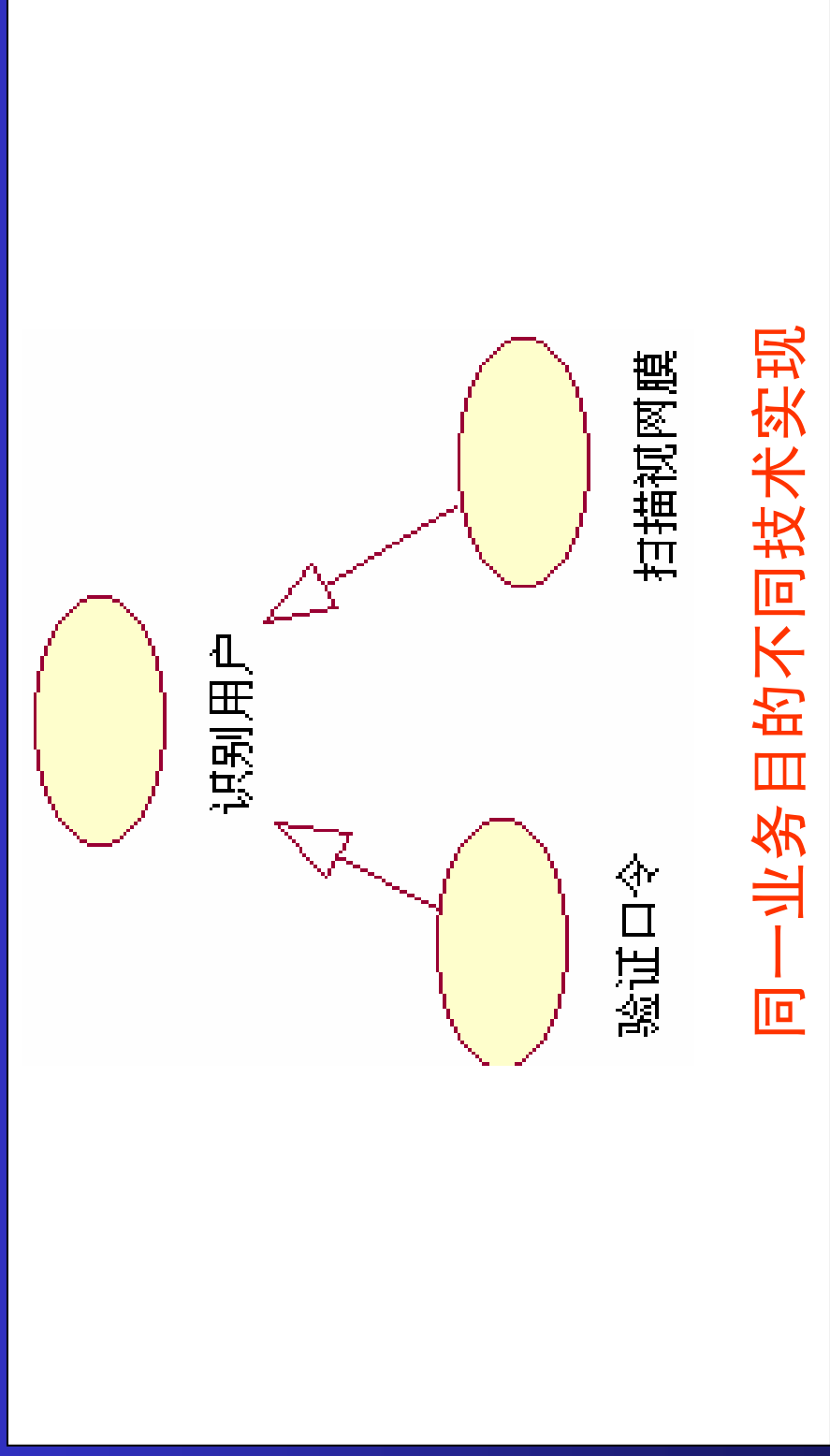
子用例继承父用例的一切:

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



识别用例的关系

——泛化关系举例



同一业务目的不同技术实现



识别用例的关系

——泛化关系文档示例

UC20 检索螺钉

父用例

UC03

主参与者

潜在会员

前置条件

参与者访问系统

后置条件

参与者查询到所要的零件

基本路径

1. 参与者提交查询条件
2. 系统按查询条件检索螺钉
3. 系统显示搜索到螺钉的编号、类别、价格
4. 参与者选中某个螺钉
5. 系统显示该螺钉的详细信息

扩展点

- 2a. 系统没有检索到所需螺钉:
2a1. 系统显示“没有找到适合条件的螺钉”
2a2. 用例结束

补充说明

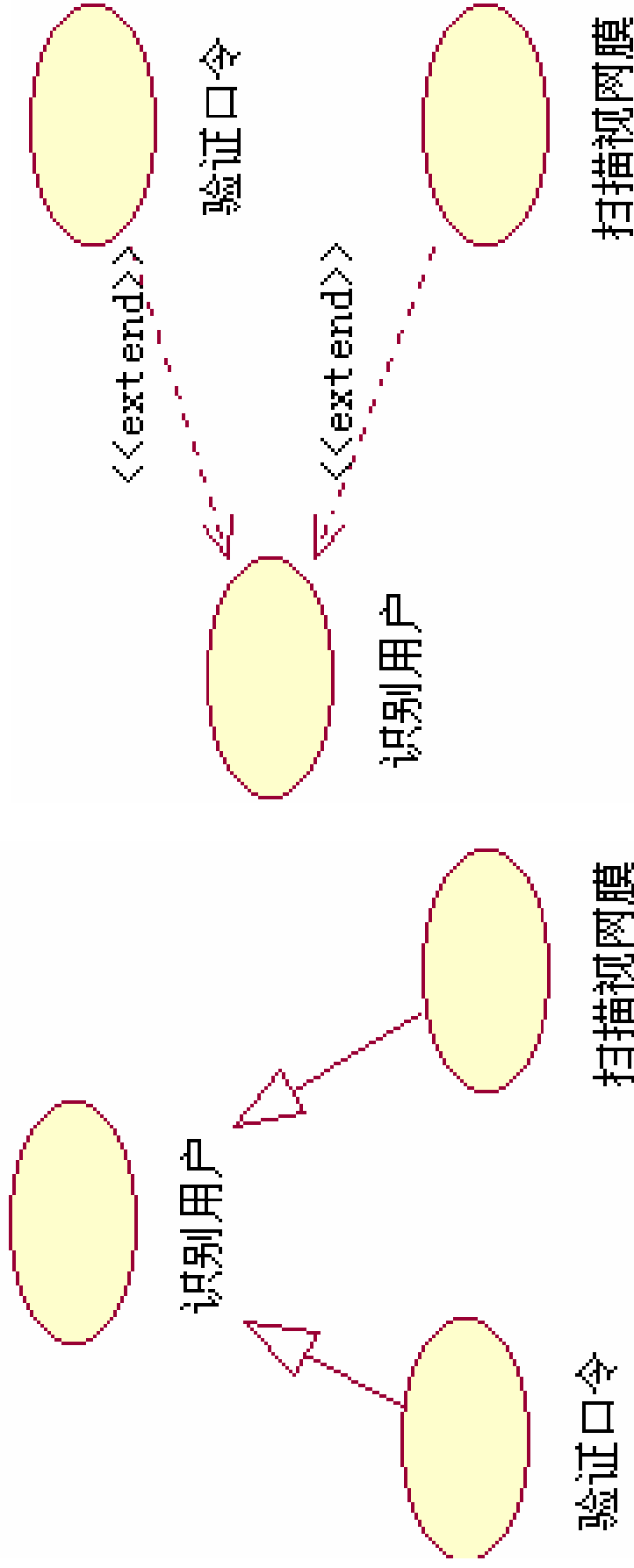
.....

} 斜体区分覆盖文本



识别用例的关系

——泛化？扩展？

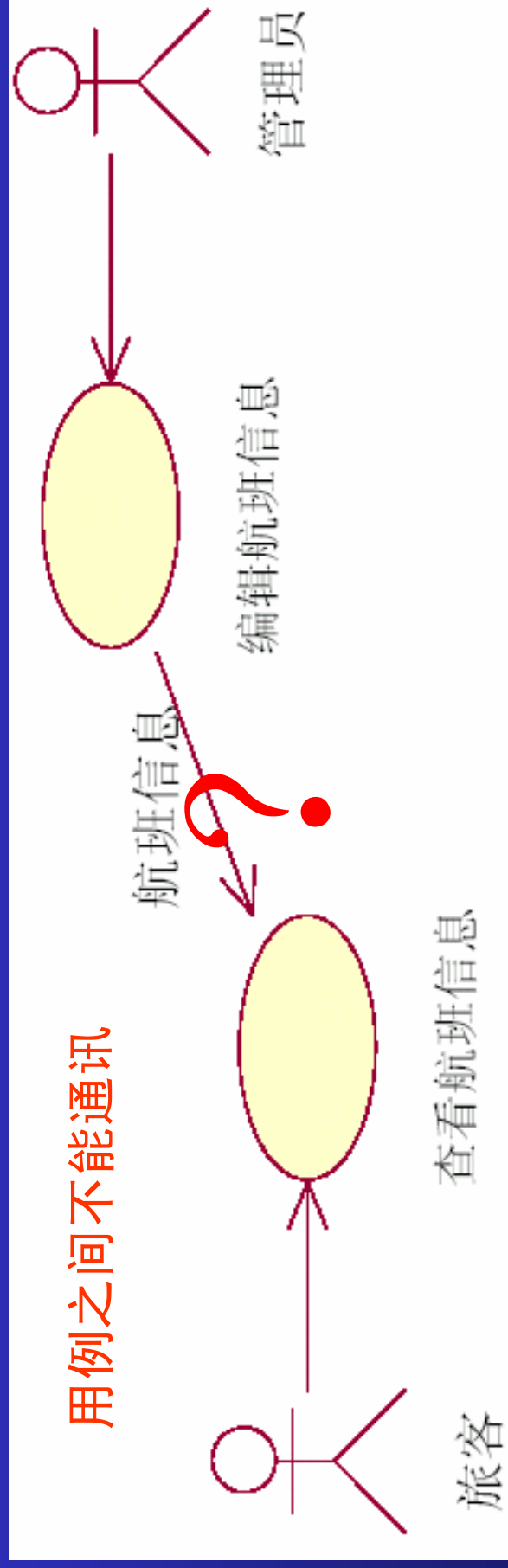


采用关系不同，文档结构不同



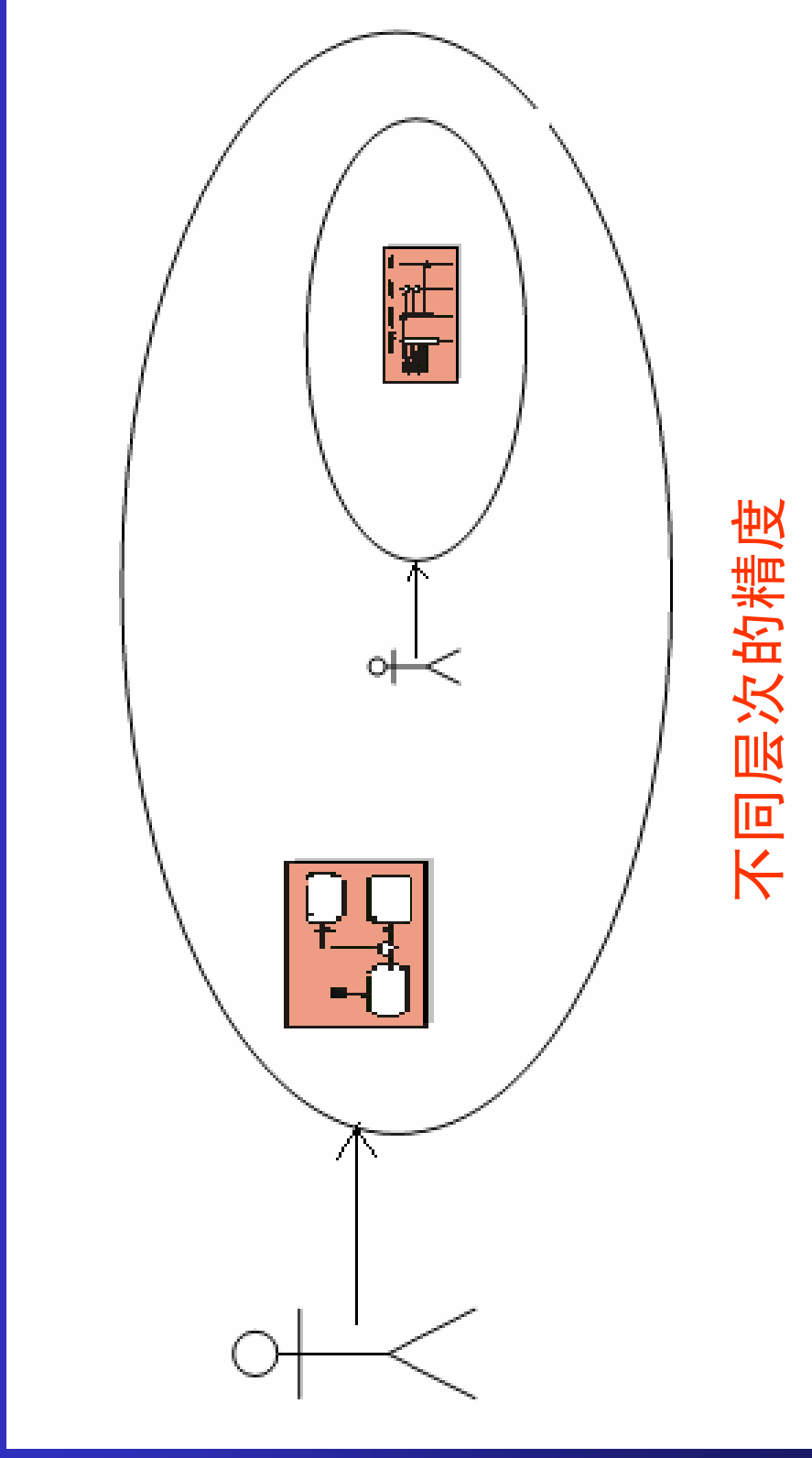
识别用例的关系

除此之外，不能有别的关系

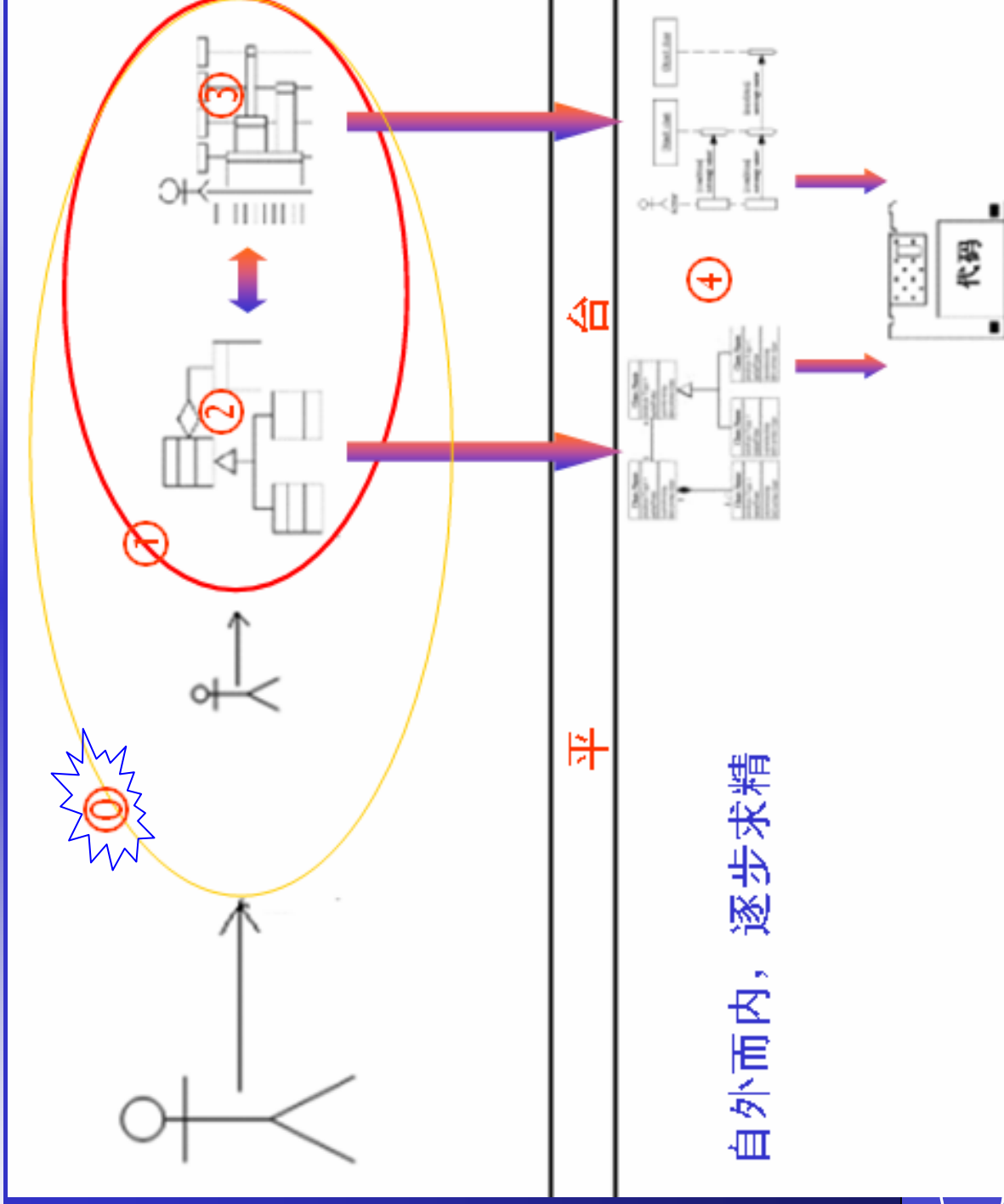


识别用例的关系

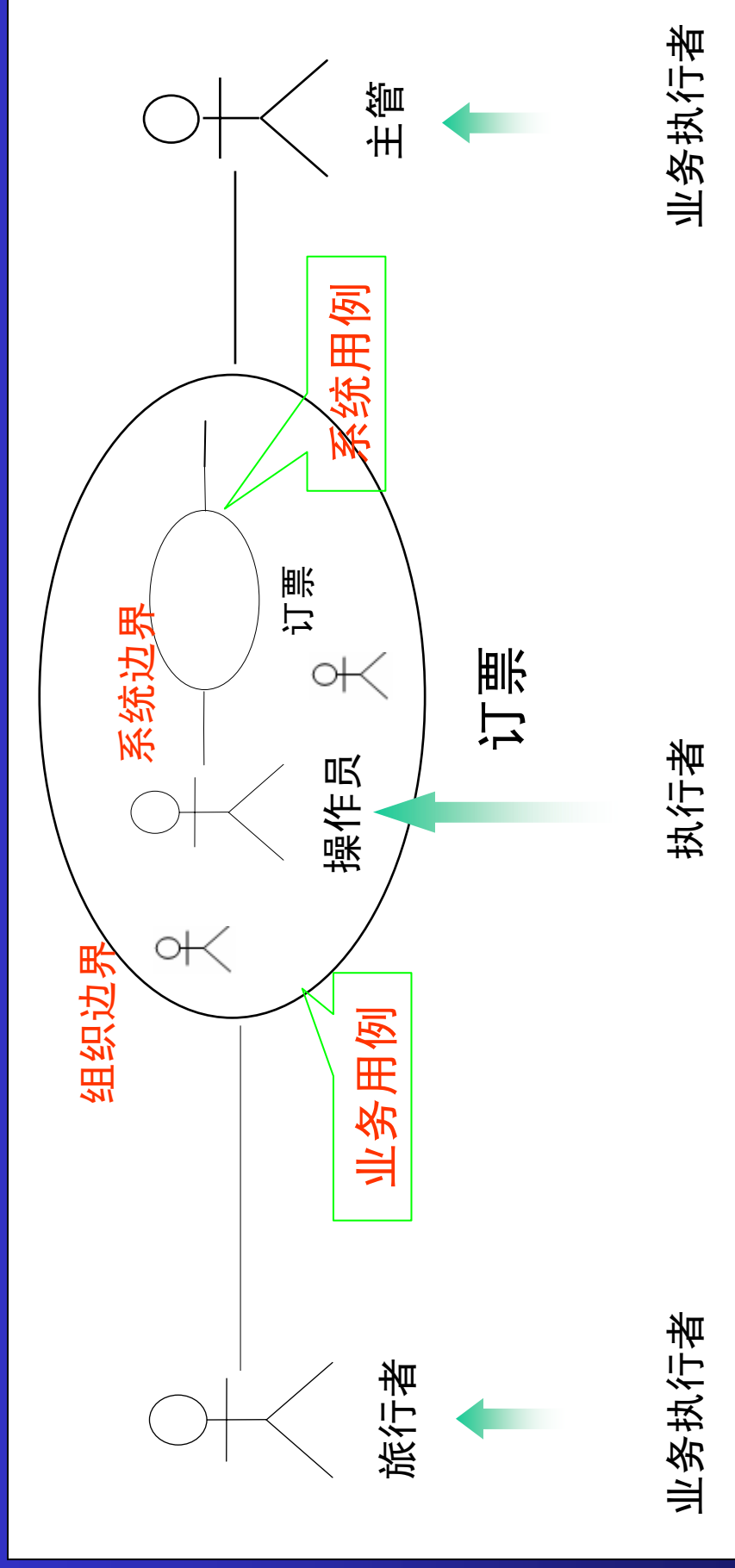
——用例是断的，不爽？

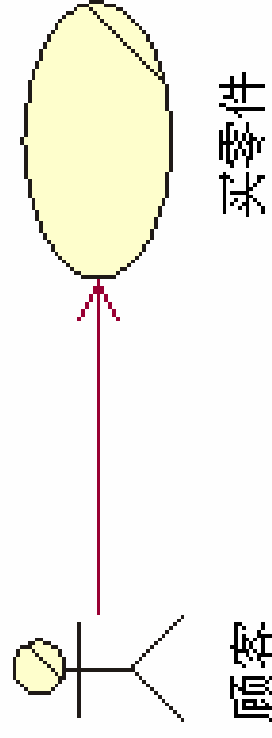
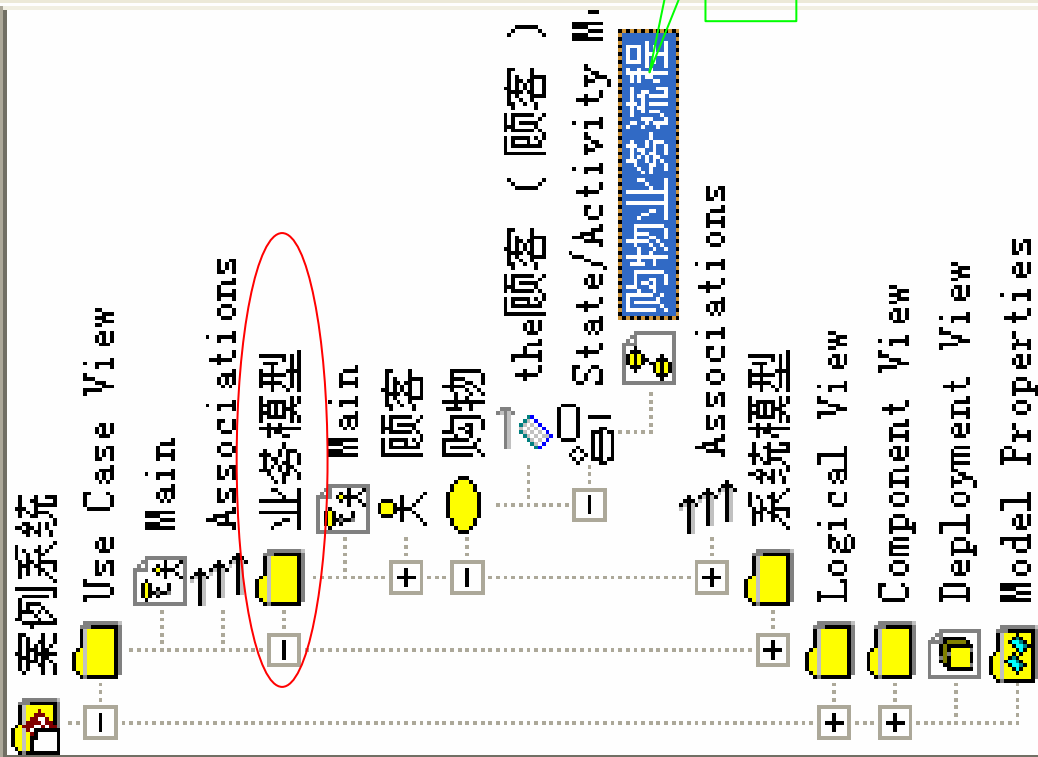


开发过程--①业务用例



业务用例：把边界向外扩





也可以不用文字
改用活动图表示流程

步骤

识别系统边界和执行者 (*)

识别用例 (*)

书写用例文档 (*)

识别用例的关系

用例的排序和分包



对用例进行优先级排序

——排序原则

- ❖ 以下情况的用例优先级最高
 - a) 对类图有重要影响
 - b) 包含丰富的业务过程信息和线索
 - c) 有开发风险、时间紧迫或功能复杂
 - d) 涉及到重要核心技术或新技术
 - e) 能直接产生经济效益或降低成本
 - f) 代表本系统的核心流程



对用例进行优先级排序

——排序方法

用例	A	B	C	D	E	F	总计
结账	5	3	4	0	5	4	21
...							



大量用例时的组织

- ❖ 按执行者分包
- ❖ 按主题分包 
- ❖ 按开发团队分包
- ❖ 按发布情况分包

可以先按主题分包，主题内再按开发团队和发布情况分包

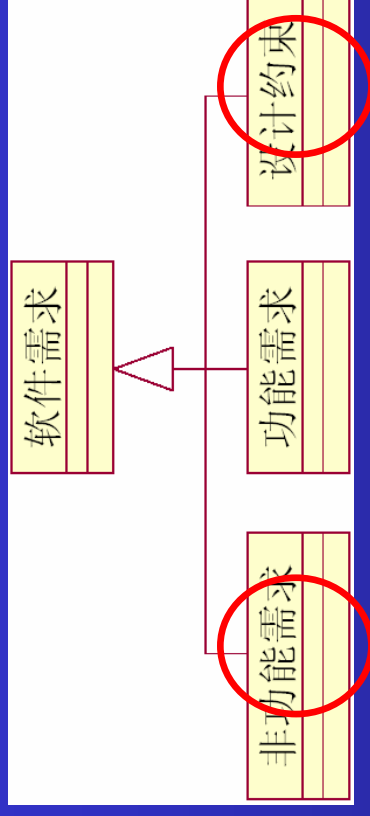


补充需求规约

- ❖ 非功能需求 (URPS)
 - ❖ 可用性 (Usability)
 - ❖ 可靠性 (Reliability)
 - ❖ 性能 (Performance)
 - ❖ 可支持性 (Supportability)

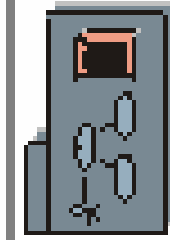
- ❖ 设计约束

- ❖ 用Oracle数据库平台, 用PB开发....
- ❖ 软件必须符合ISO××××标准....
- ❖ ...
- ❖ 本质上不是需求, 只是从商业、行政、技术上的约束



RUP的SRS包

工件：软件需求规约



软件
需求
规约

软件需求规约 (SRS) 记录对系统或系统的一部分的完整软件需求。使用用例建模时，本工件由一个包组成，该包包含用例模型的用例和适用的补充规约。

角色：

用例阐释者

模板：

- Microsoft Word 模板（一个带有用例，另一个没有用例）
- HTML 模板（一个带有用例，另一个没有用例）

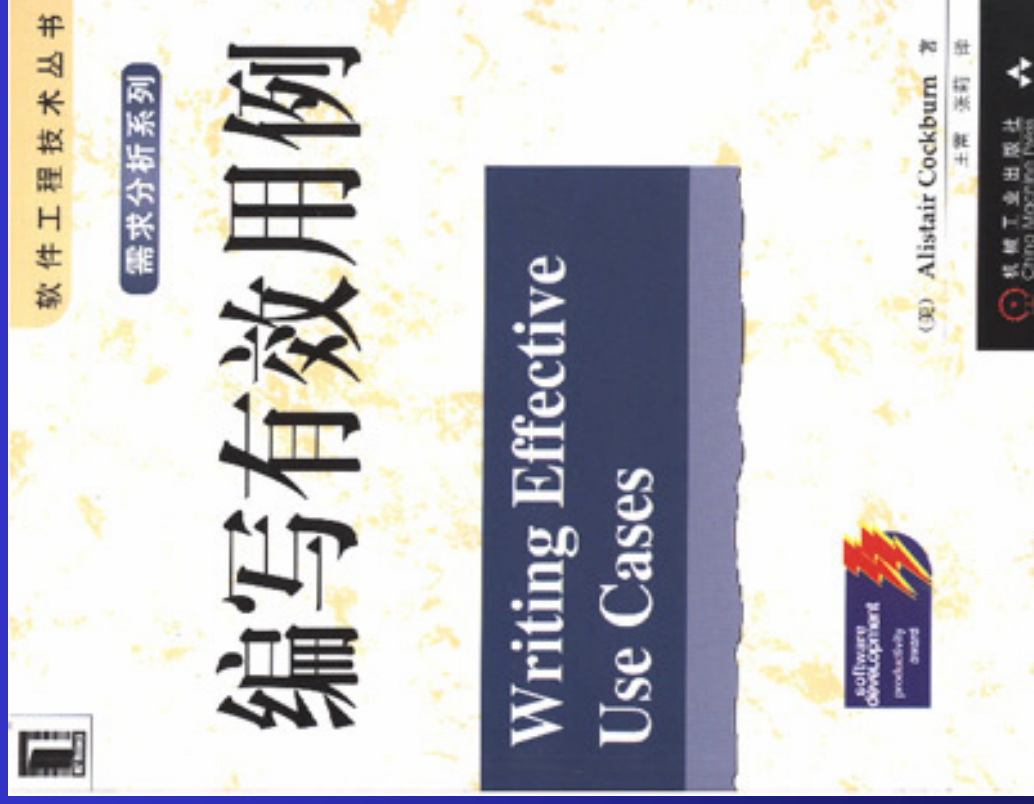


用例参考资料——《非程序员》

- 《非程序员》第1期，用Use Cases捕获需求
- 《非程序员》第4期，用例的使用误区、构造用例过程、创建有用的用例
- 《非程序员》第5期，怎样避免用例陷阱
- 《非程序员》第8期，Ivar Jacobson访谈
- 《非程序员》第10期，返璞归真：通过简化用例来简化用户界面
- 《非程序员》第11期，面向对象开发中的本质用例及其职责、利用角色扮演和用例卡片进行需求复审、针对用户界面设计用例结构和式样
- 《非程序员》第13期，为什么用例如此难用？
- 《非程序员》第14期，Alistair Cockburn：选择你所需要的
- 《非程序员》第19期，尤克滨：简单正是用例的价值
- 《非程序员》第22期，Alistair Cockburn：用例，十年风雨
- 《非程序员》第24期，让历史告诉未来
- 《非程序员》第21、22、25期，使用用例组织需求
- 《非程序员》第26期，《有效用例模式》中译本样章
- 《非程序员》第27期，误用例：带敌对意图的用例



用例书籍



UMLChina训练指定教材



<http://www.umlchina.com>

用例的将来

有效用例模式

对于这点，我们可以预言不久的将来的一些事情。

- (1) 作为一种主流技术，用例将在本科课程中、全世界的标准软件开发和法学课程中被讲授。
- (2) 工具厂商和理论家将继续寻求用例所隐含的形式主义。
- (3) 工具厂商将生产和理论匹配得更好的工具，使交叉关联的用例更容易书写、维护和集成到 CASE 工具套件中。
- (4) 人们将继续寻找这项当前主流技术的替代品，然后建议各种不用用例来进行软件开发的方法。
- (5) 一些事情依然不变：人们继续在他们的用例中书写太多的界面细节，不管他们的工具和模板是什么；公司将继续使用用例来避免人们面对面交流。
- (6) 人们将误用例，不正确地教授用例，曲解用例，然后把随之而来的混乱归结到用例身上。

用例&需求部分

——讨论和练习



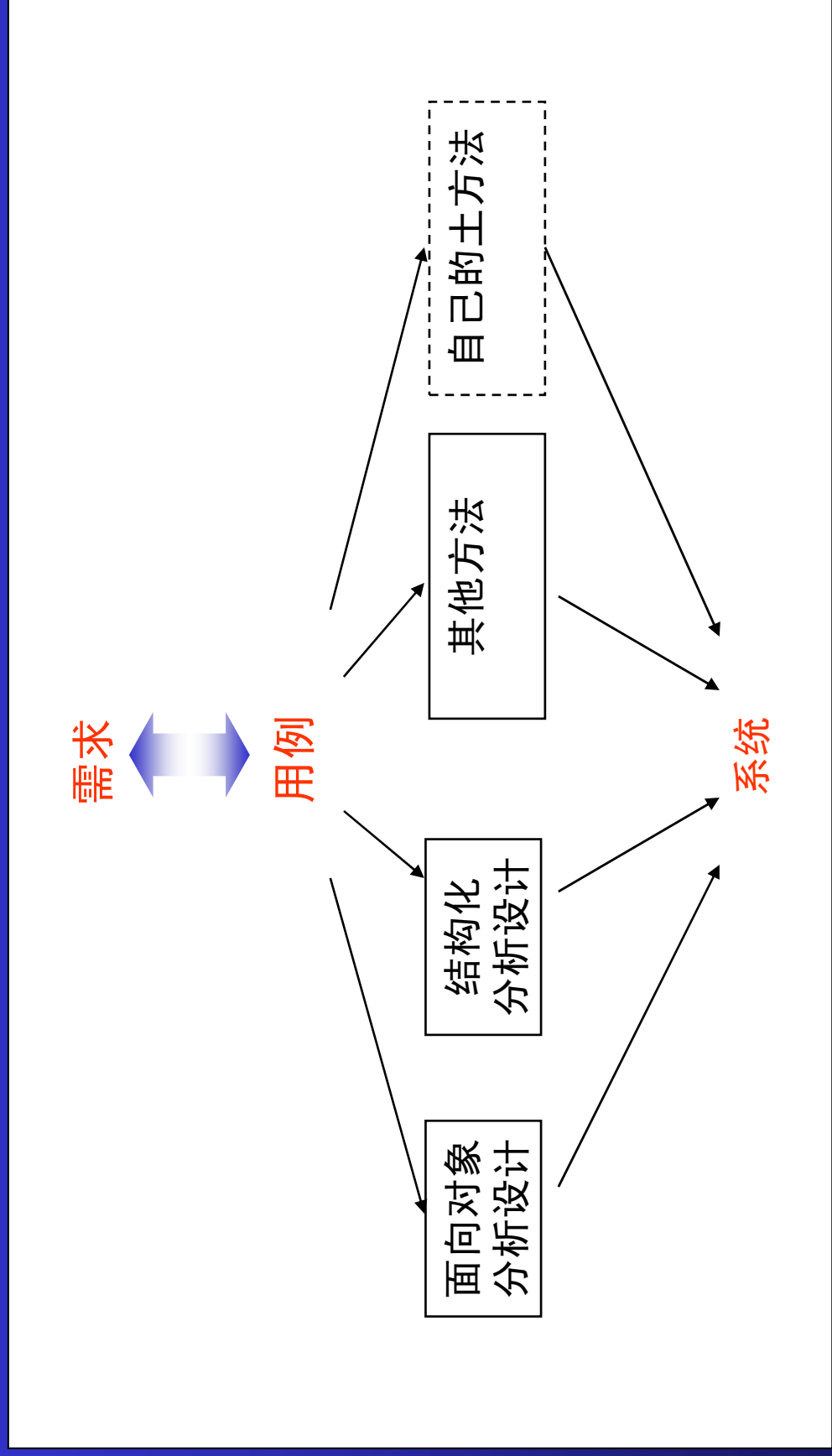
下面呢？



从前，有一个太监...



下面—可以有多种走法



用例和OO

- ❖ “发明”于OO的环境（Ivar Jacobson）
- ❖ 从外部Actor的角度描述系统功能（和对象的消息类
似）
- ❖ 不暴露内部结构
- ❖ OO设计的最好开始，是OO技术进入第二代的标志

