



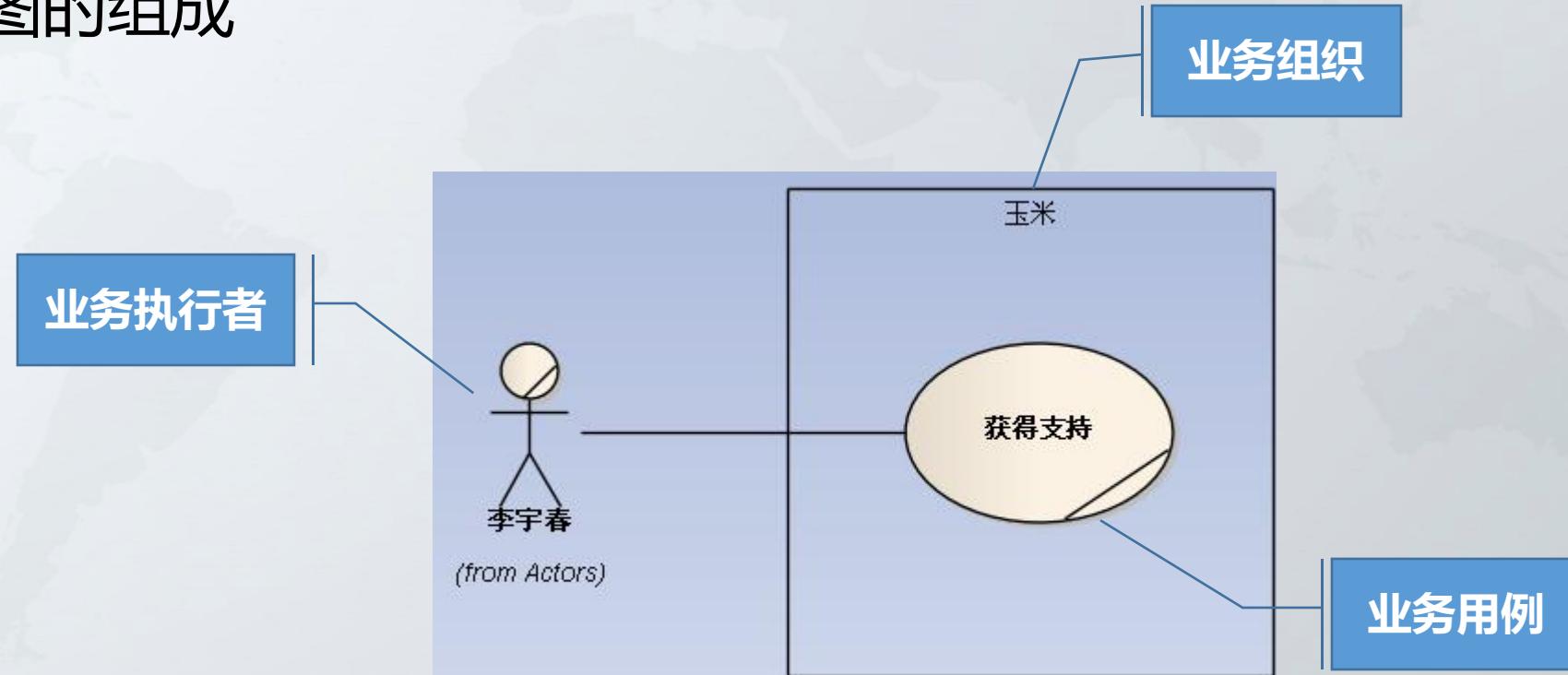
第四章 需求分析，用例分析法

回顾：业务建模的步骤 >>>

1. 明确我们为谁服务（选定愿景要改进的组织）。
2. 要改进的组织是什么现状（业务用例图、现状业务序列图）。
3. 我们如何改进（改进业务序列图）。

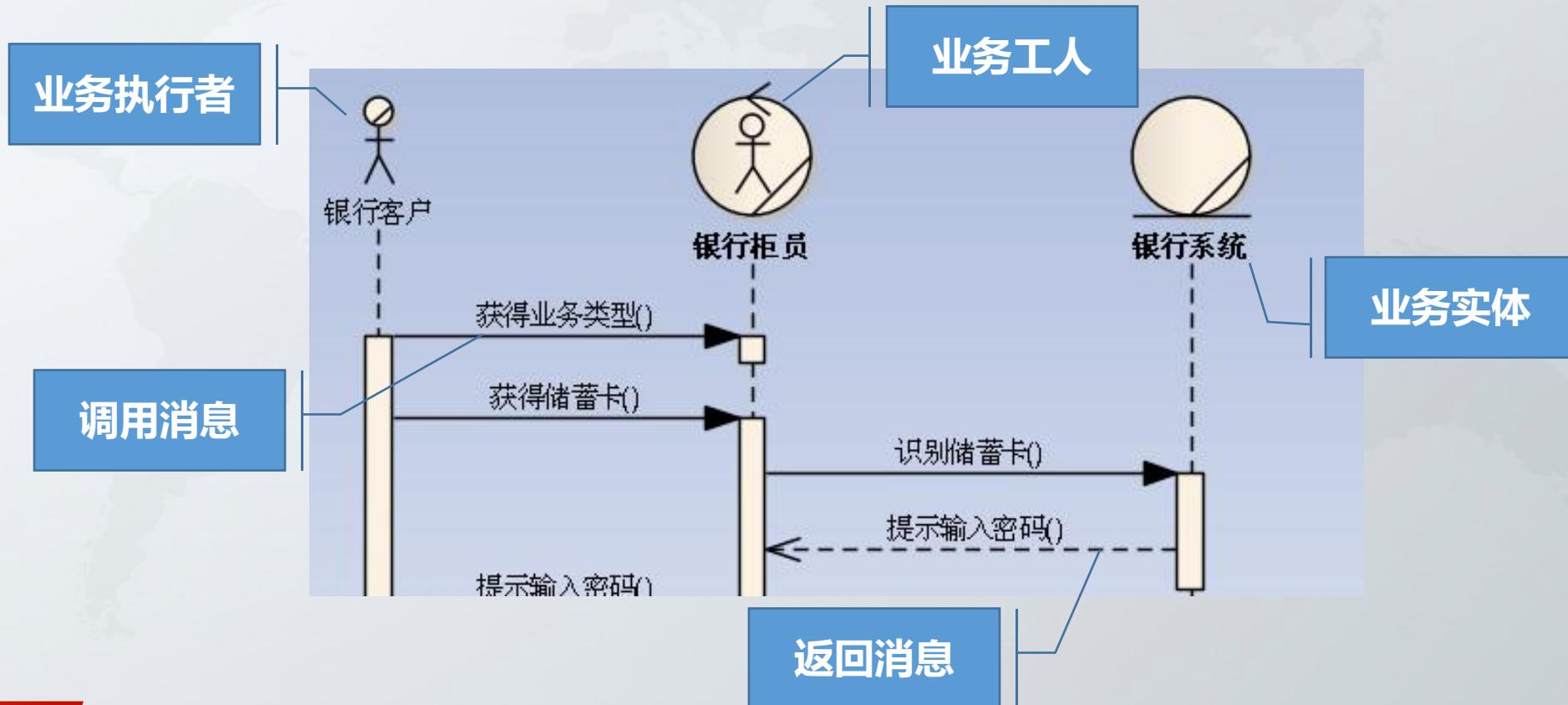
回顾：业务用例图 >>>

- 业务用例图帮助我们从高层次了解组织的业务构成。
- 业务用例图的组成



回顾：业务序列图的组成 »»

- 业务序列图详细描述业务执行者、业务工人、业务实体之间如何交互，以完成某个业务用例的实现流程。

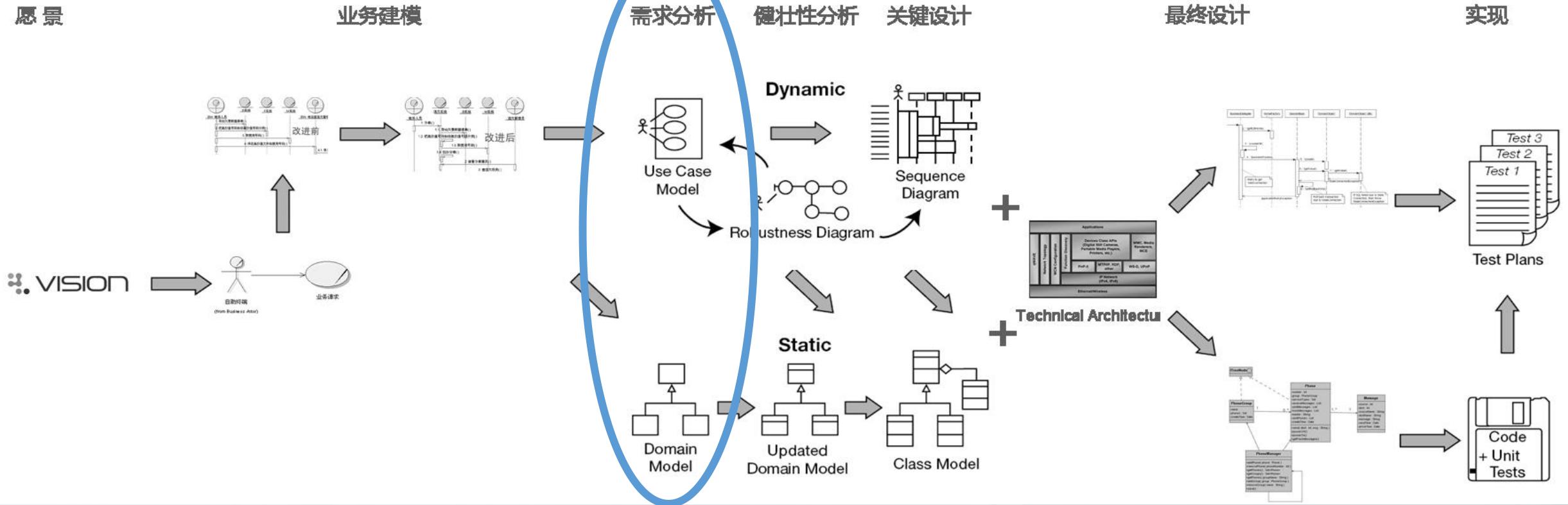


回顾：采用改进业务序列图改进业务流程

1. 将新系统作为一个业务实体；
2. 将其引入组织现有业务流程；
3. 查看其可以改进的流程；
4. 评估改进结果。

需求分析

愿景



目录 >>>

一

需求分析的几种主流方法

二

域建模：以OO思想构建术语表

三

用例分析：系统功能性需求分析的好工具

四

非功能性需求分析及需求定义与评审



目录 >>>

一

需求分析的几种主流方法

二

域建模：以OO思想构建术语表

三

用例分析：系统功能性需求分析的好工具

四

非功能性需求分析及需求定义与评审

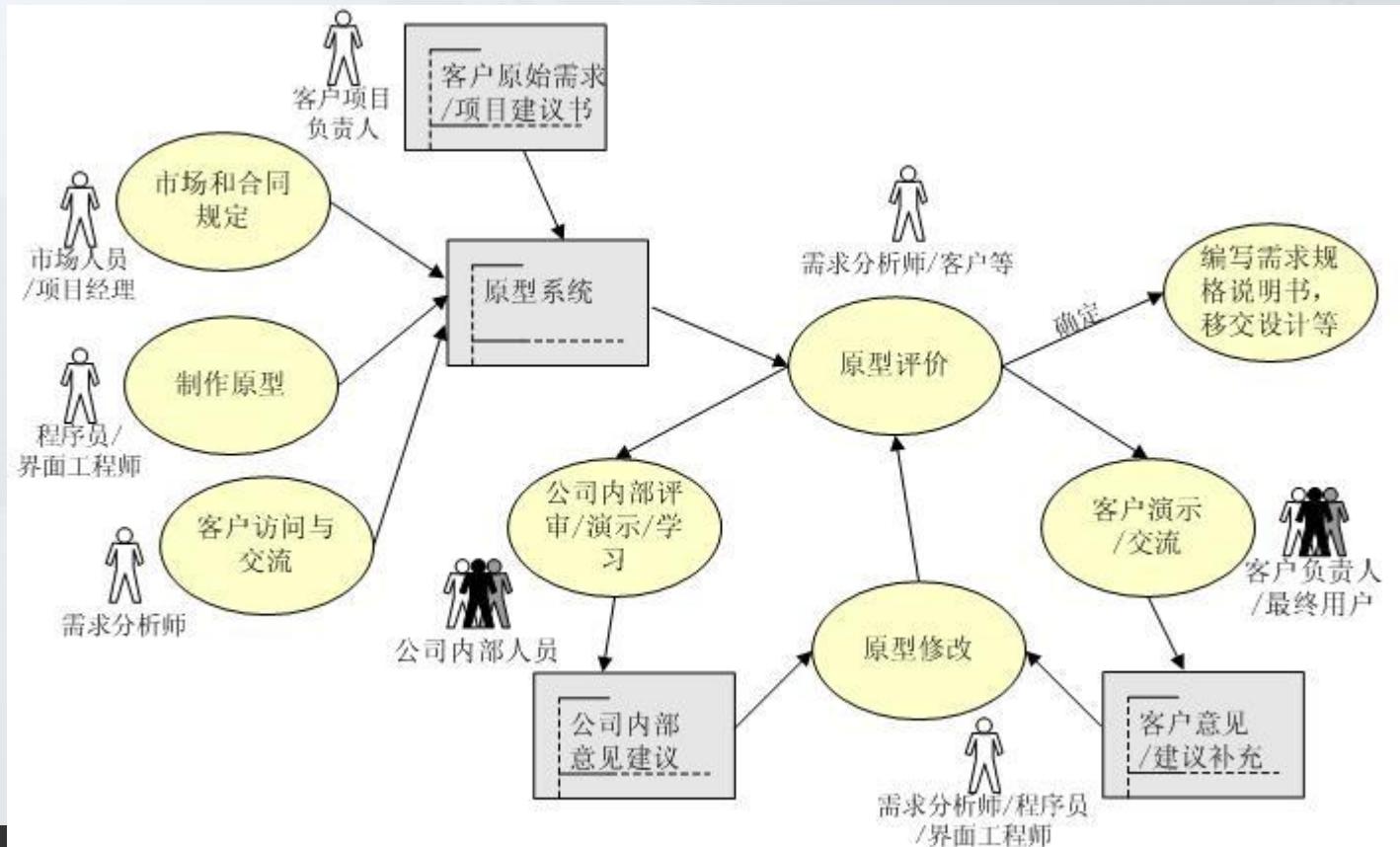


需求分析方法演进简史 ➤➤➤

时间	方法	描述	工具	缺点
1970年前	传统分析	需求分析人依据个人习惯进行一些建模和分析工作，完全依赖个体才智	自然语言	缺乏结构、不可重复、不可测量、具有主观臆断性
70-80年	结构化分析	把现实世界描绘为数据在信息系统中的流动，以及在流动过程中，数据向信息的转化。	数据流图、实体关系图、状态转移图等	分析向设计过渡的难度高 对于复杂系统分析困难
80-90年	信息工程	对结构化方法的改进。从信息角度来开发系统，客观世界被描述为数据和数据属性及其相互关系。	功能分解图，过程依赖图	仅适用于信息系统的开发
90年后	面向对象	将系统看作对象的集合，对象间的互相协作，共同完成系统任务。	用例分析，域模型、交互图、状态图、活动图等	

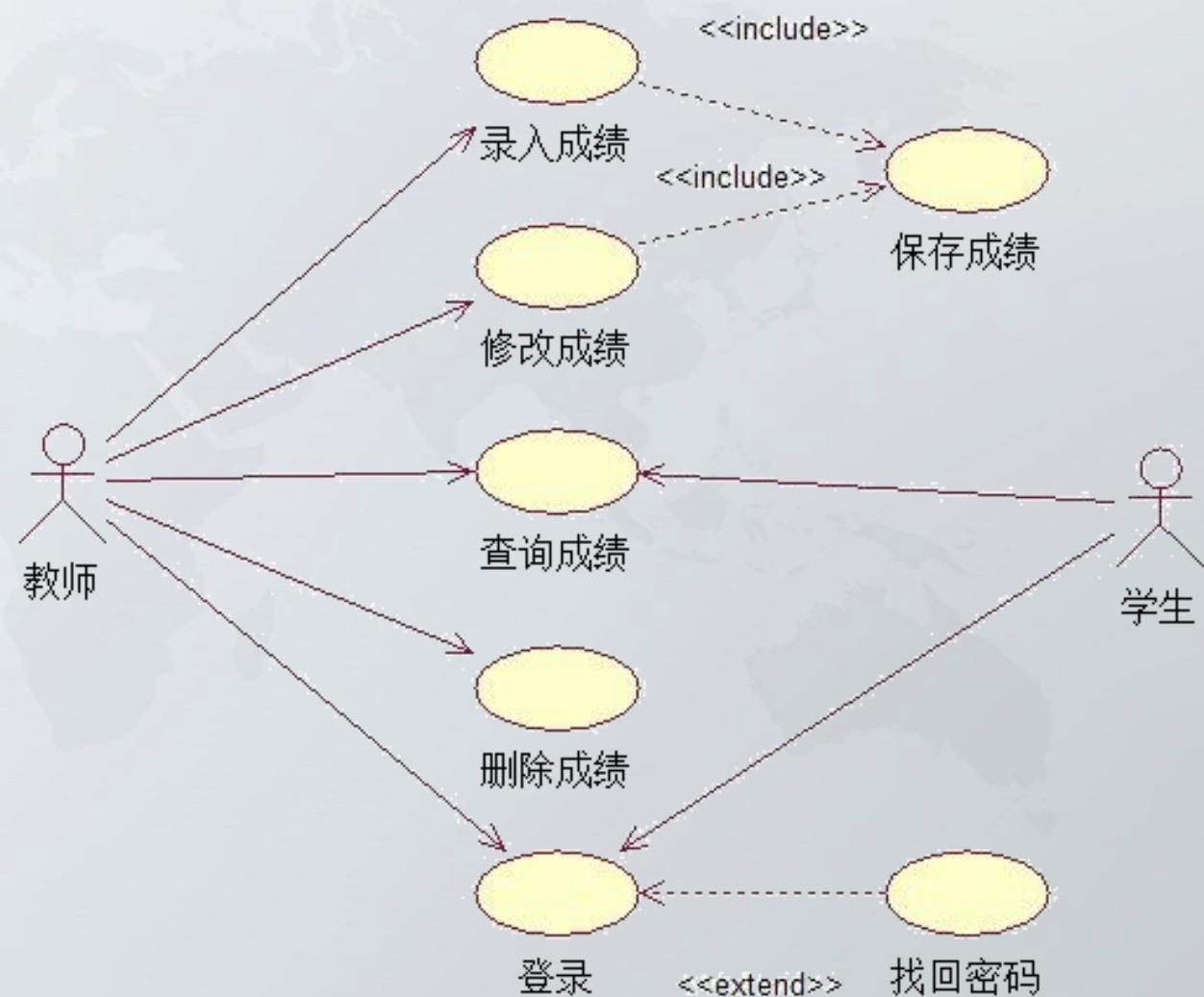
主流分析方法之原型法 ➤

- 原型法是指在获取一组基本的需求定义后，利用高级软件工具可视化的开发环境，快速地建立一个目标系统的最初版本，并把它交给用户试用、补充和修改，再进行新的版本开发。反复进行这个过程，直到得出系统的“精确解”，即用户满意为止的一种方法。



主流分析方法之用例法 ➤

- 用例图(Use Case Diagram)是由软件需求分析到最终实现的第一步，它描述人们如何使用一个系统。用例视图显示谁是相关的用户、用户希望系统提供什么样的服务，以及用户需要为系统提供的服务，以便使系统的用户更容易理解这些元素的用途，也便于软件开发人员最终实现这些元素。用例图在各种开发活动中被广泛的应用，但是它最常用来描述系统及子系统。



目录 >>>

一

需求分析的几种主流方法

二

域建模：以OO思想构建术语表

三

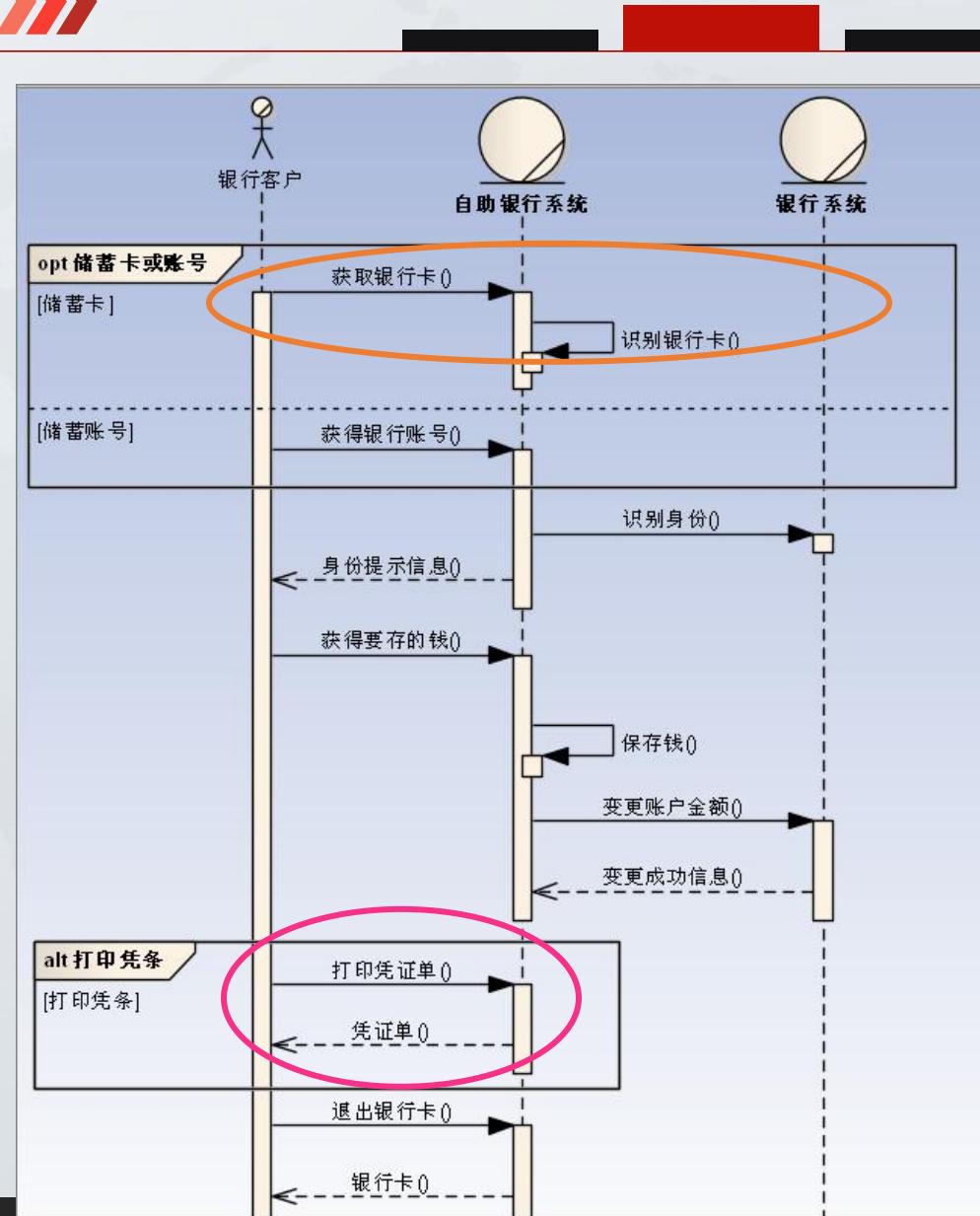
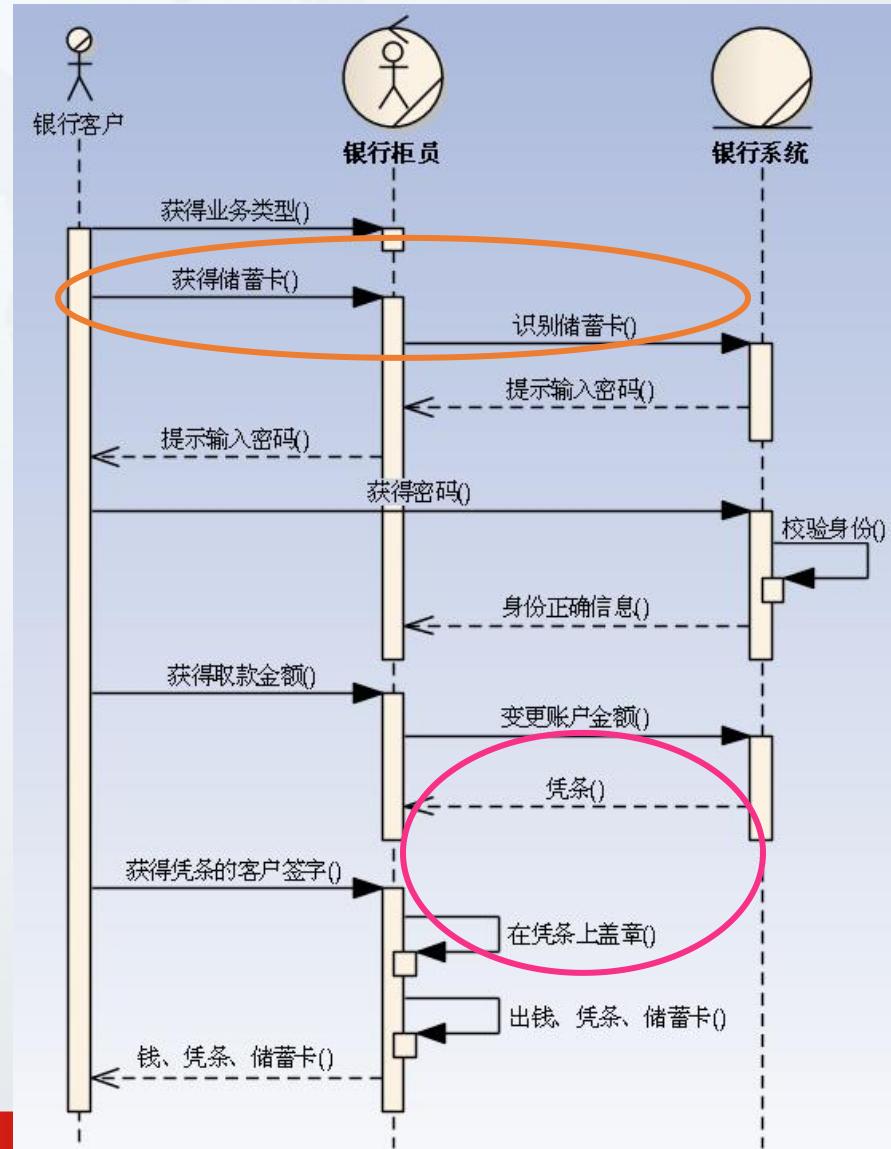
用例分析：系统功能性需求分析的好工具

四

非功能性需求分析及需求定义与评审



思考:下面图中有什么问题吗? >>>



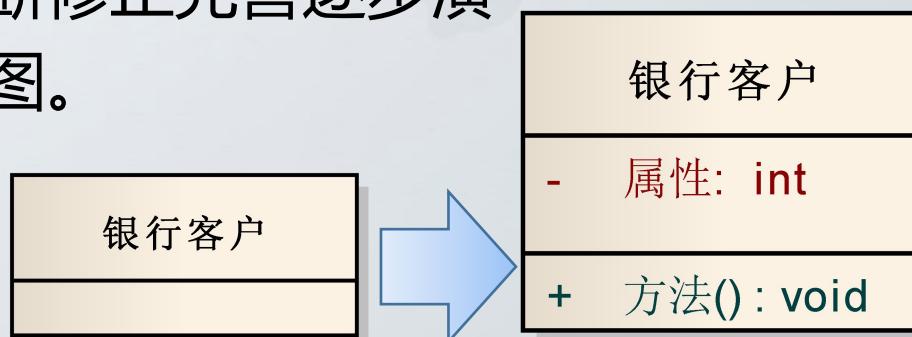
需求描述中术语不统一 ➤➤

- 在实际项目中，原始需求的描述形式可能是文字、活动图、序列图或其它形式，不管使用哪种形式，术语不统一的现象非常常见。
- 通常一个完整的项目需求都有较多的篇幅，想象一下，大量不统一的术语对交流和后续工作会造成多大的混乱，即使文字统一，也可能造成理解偏差。

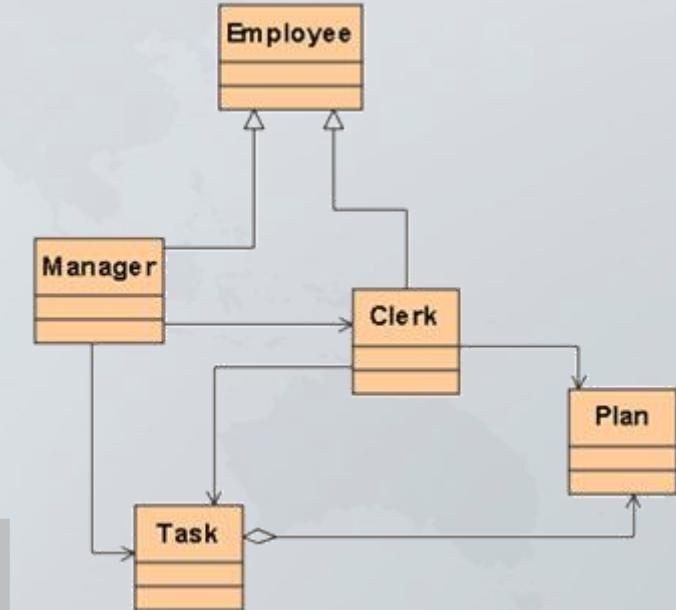


域建模的意义 »»

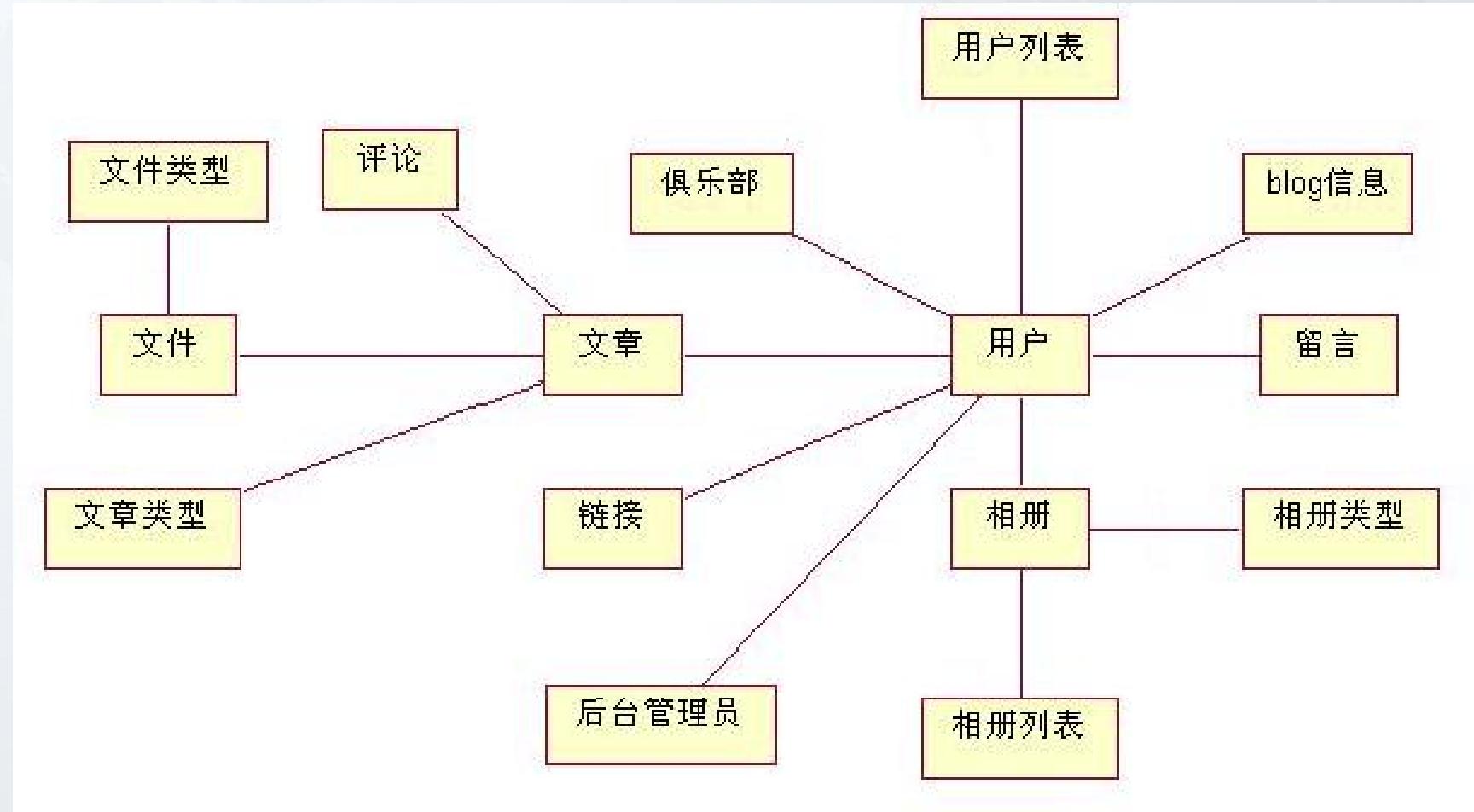
- 域建模[Domain Modeling]
 - 为项目创建一个**术语表**。确保项目中的每个人都能以**清晰一致**的术语来理解和交流问题领域。
 - 域建模比普通的项目术语表优良的地方体现在：以图的方式清晰地显示出不同术语间的**关系**（减少理解偏差）。
 - 域模型图将通过不断修正完善逐步演化为最终的静态类图。



雇员、经理、职员、计划、任务



某社交系统域建模结果 >>>



域建模的步骤 >>>

Step 1
仔细阅读需求文档，提取出名词和名词短语

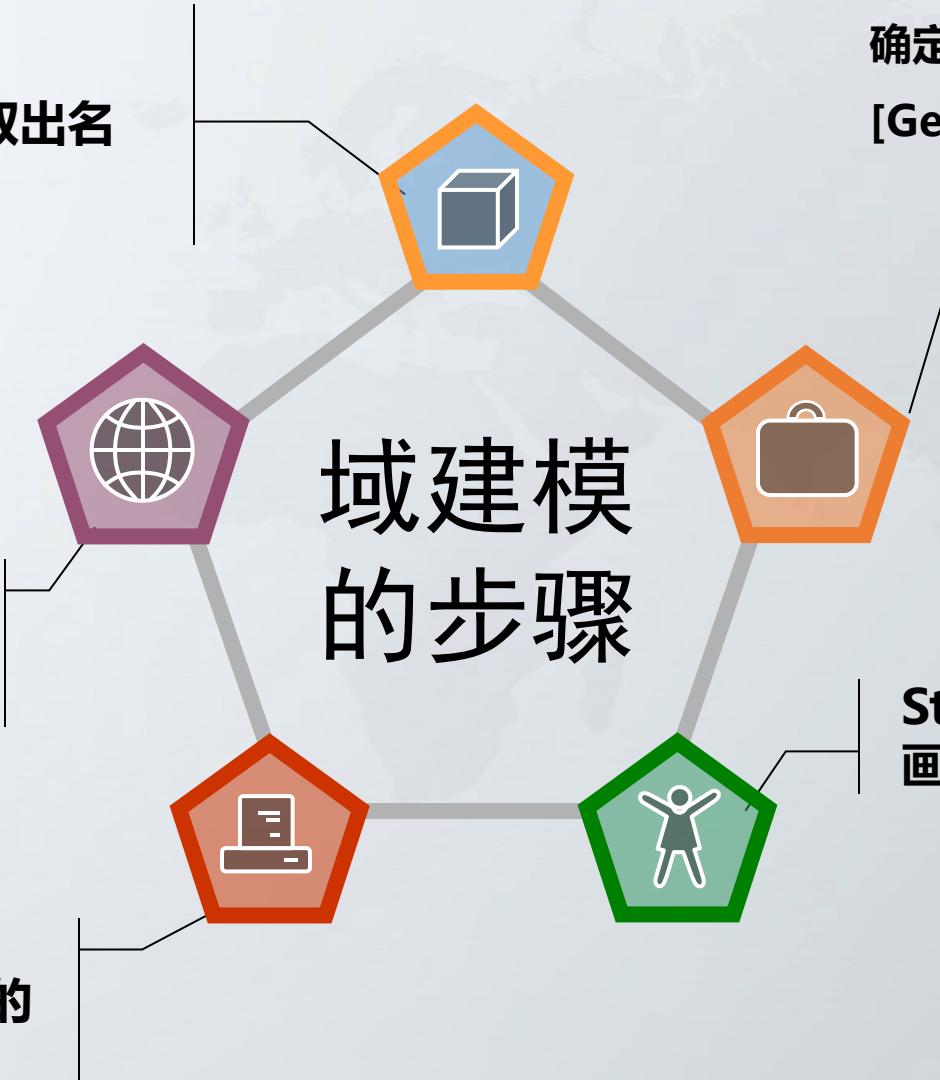
Step 2
排除列表中重复、相似的术语

Step 3
排除超出系统范围的术语

确定域模型之间的关系：泛化
[Generalization]和关联[Association]

Step 5
整理第一版域模型

Step 4
画出第一版域模型图



示例：基于文字需求进行域建模 >>>

- 取款：银行客户将储蓄卡插入自助银行系统，系统提示用户输入密码。用户输入正确的密码，系统验证成功后，提示用户选择业务类型，用户选择“取款”。系统提示用户输入取款金额，用户输入取款金额后，系统变更用户账户金额，然后给用户输出相应的钱。用户如果选择“打印凭条”，系统会为用户打印出凭证单。用户选择“退卡”，系统退出用户的银行卡。

第一步：提取名词或名词短语 >>>

- 取款：银行客户将储蓄卡插入自助银行系统，系统提示用户输入密码。用户输入正确的密码，系统验证成功后，提示用户选择业务类型，用户选择“取款”。系统提示用户输入取款金额，用户输入取款金额后，系统变更用户账户金额，然后给用户输出相应的钱。用户如果选择“打印凭条”，系统会为用户打印出凭证单。用户选择“退卡”，系统退出用户的银行卡。

第二步：排除重复、相似 ➤

排除前	重复、相似	排除后
银行客户 储蓄卡 自助银行系统 系统 用户 密码 业务类型 取款金额 用户账户金额 钱 凭条 银行卡 凭证单	银行客户 用户 储蓄卡 银行卡 自助银行系统 系统 凭条 凭证单	银行客户 储蓄卡 自助银行系统 凭条 密码 业务类型 取款金额 用户账户金额 钱

选择无歧义、符合业务习惯的术语

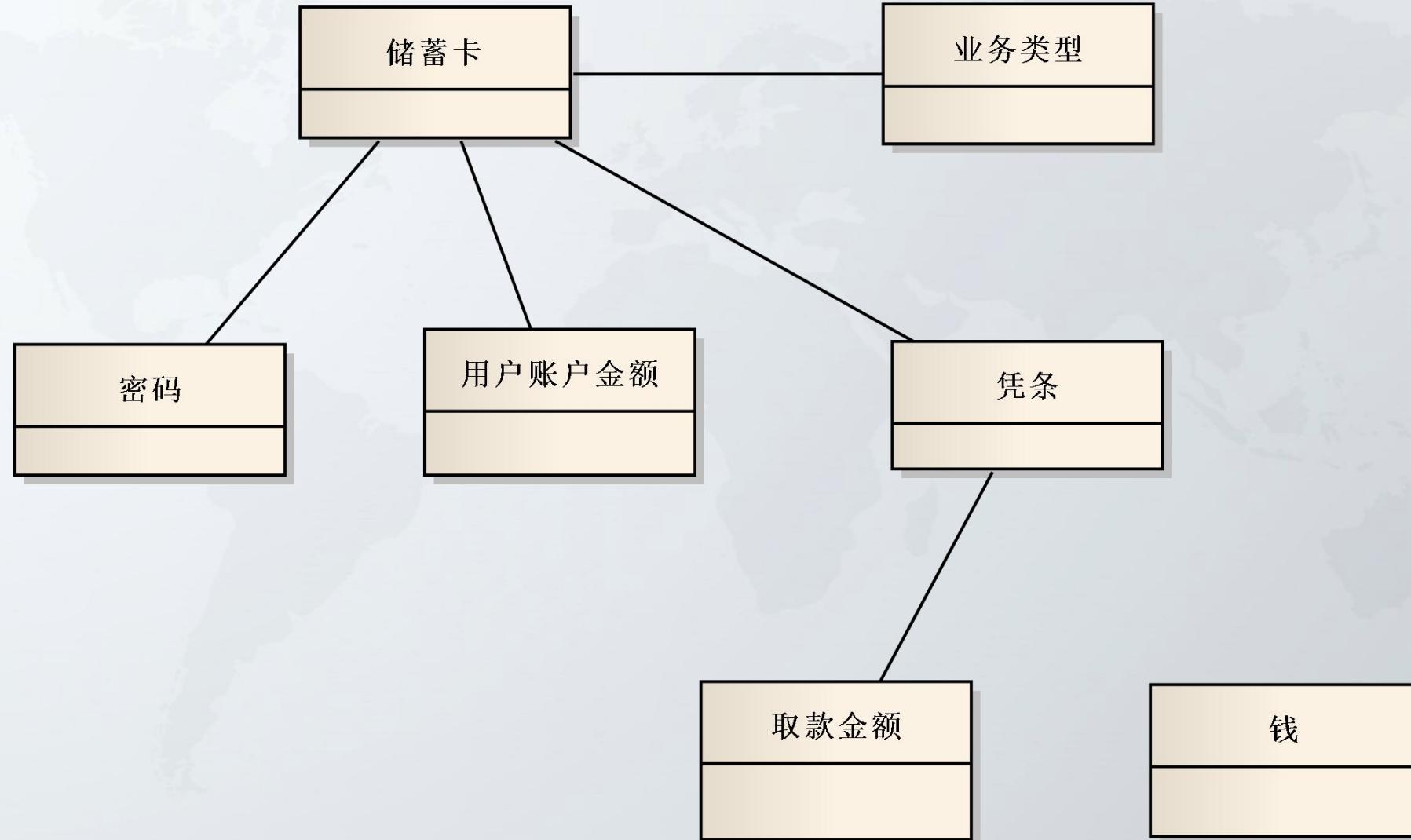
第三步：排除系统范围外和系统本身»»

排除前	系统外	排除后
银行客户	业务执行者（也是系统执行者）	×
储蓄卡		储蓄卡
自助银行系统	系统本身	×
凭条		凭条
密码		密码
业务类型		业务类型
取款金额		取款金额
用户账户金额		用户账户金额
钱		钱

第四步：画出第一版域模型图 ➤

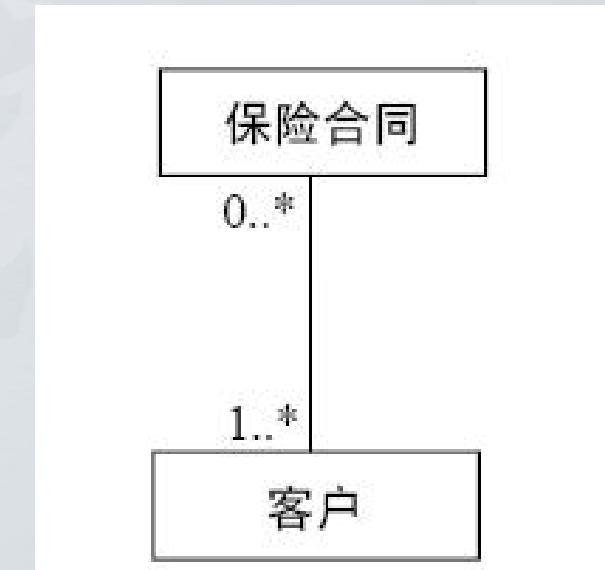
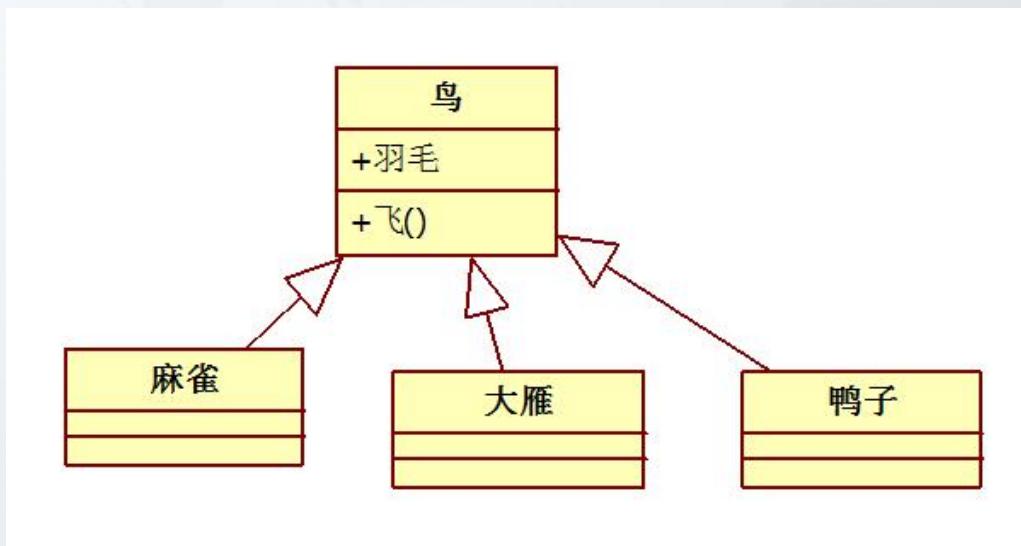


第五步：整理第一版域模型 >>>

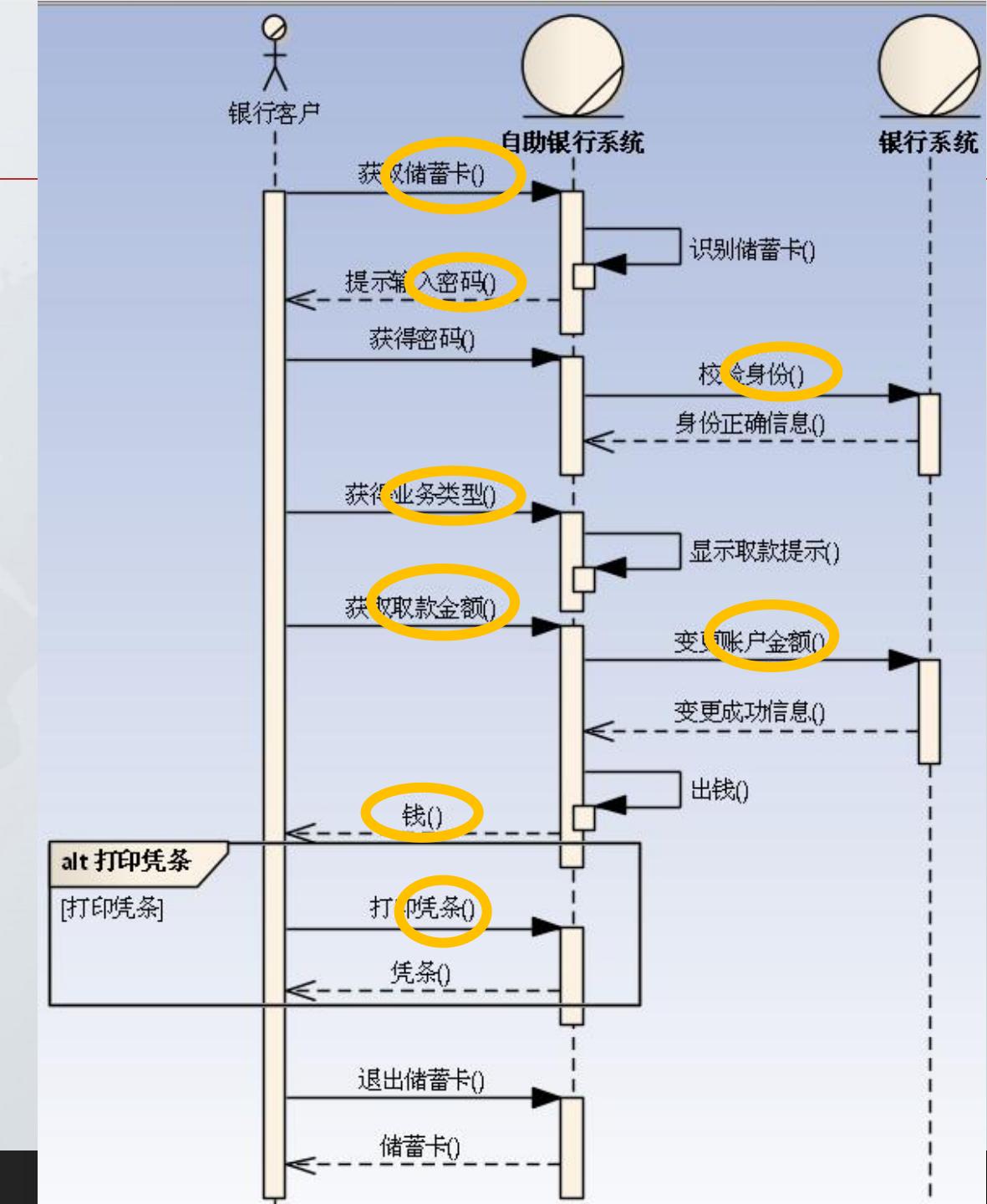


域模型之间的关系 ➤➤➤

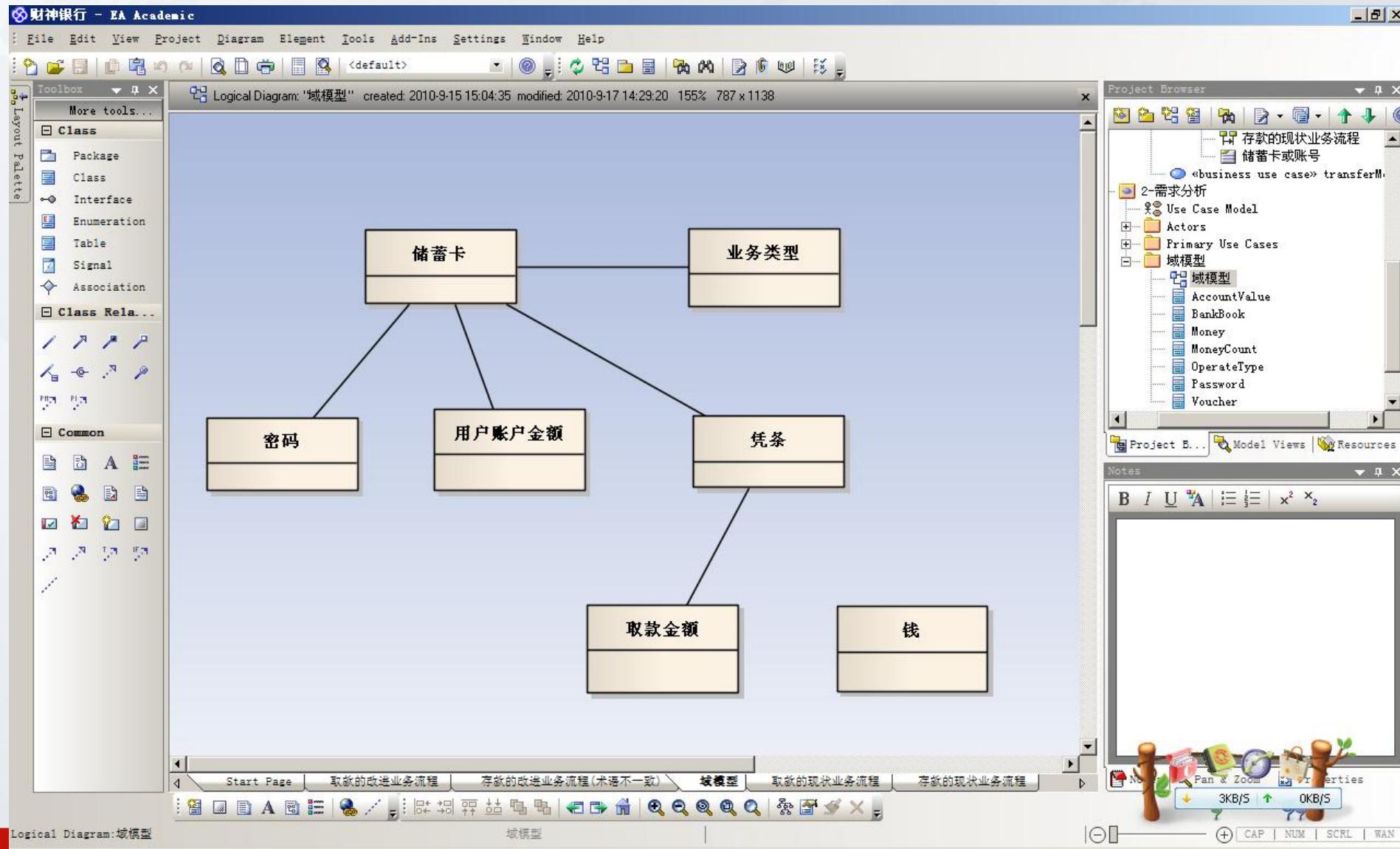
- 泛化[Generalization]，一般元素和特殊元素的关系。
- 关联[Association]，两个类之间存在着某种语义上的联系。



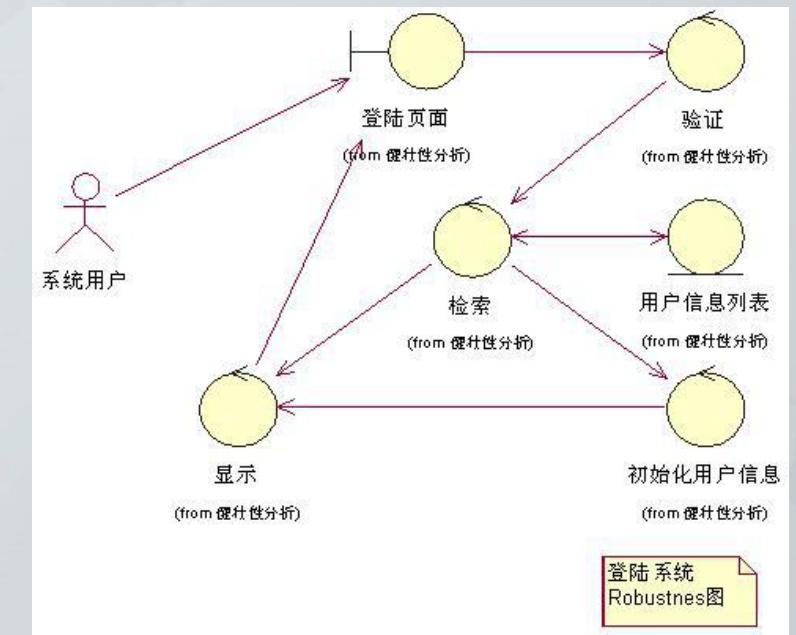
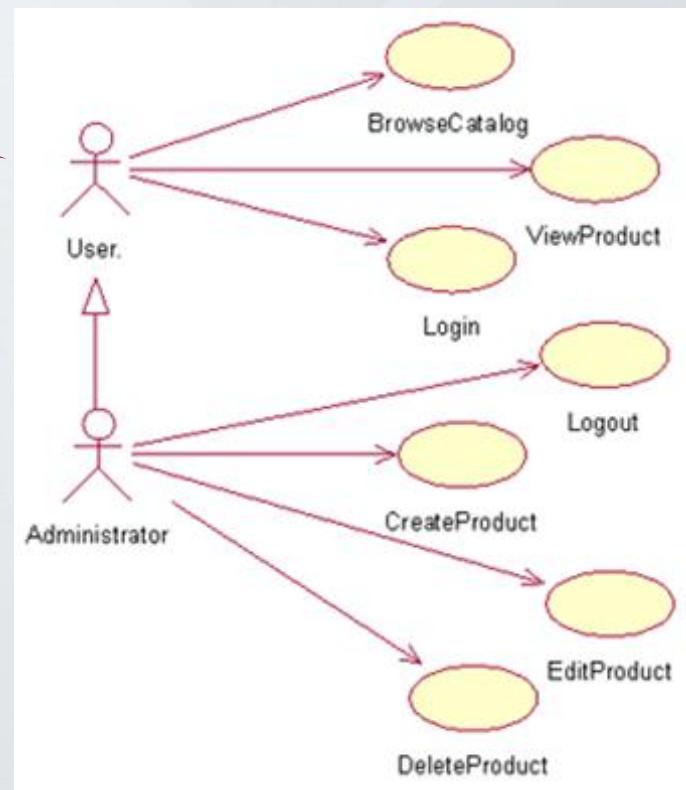
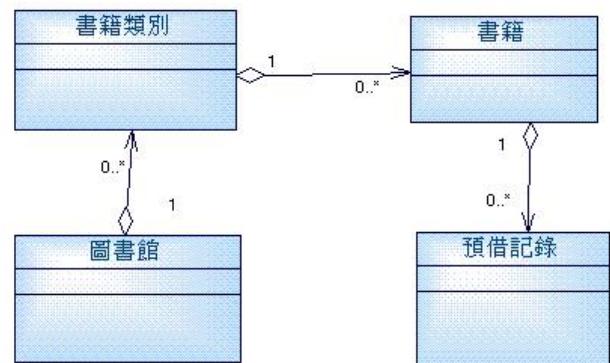
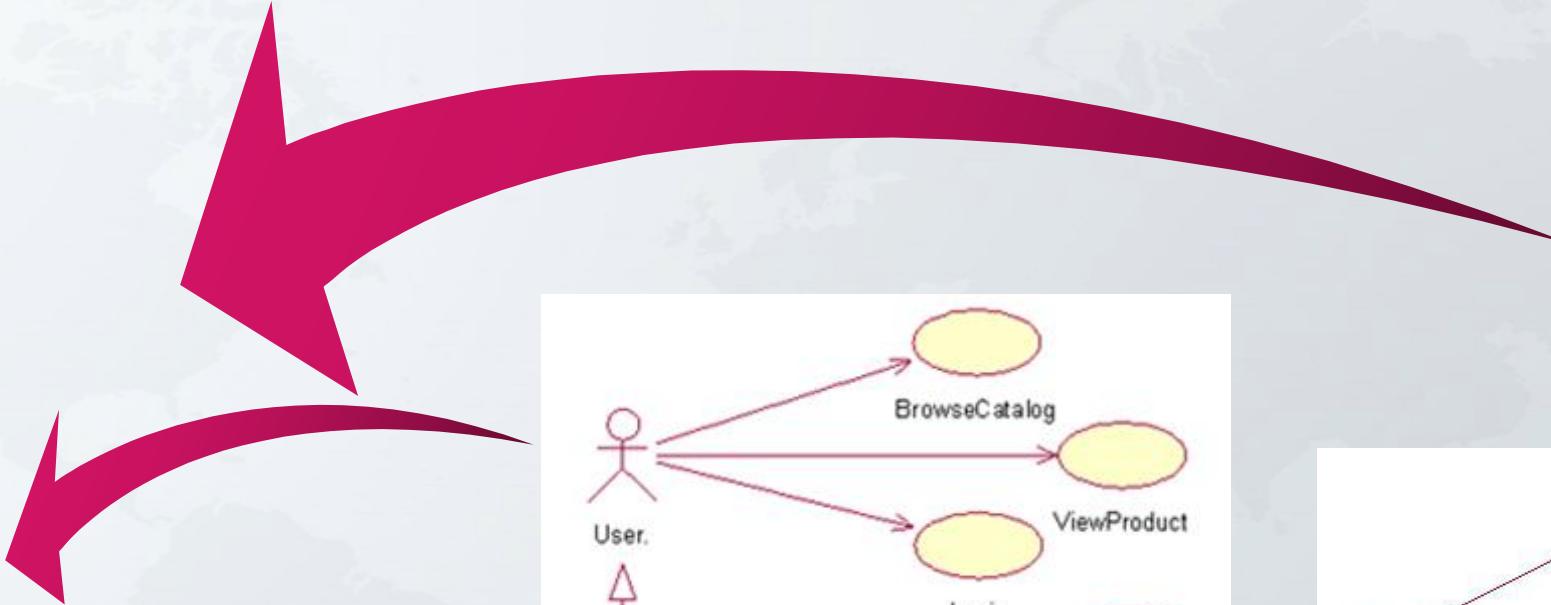
示例：基于模型图进行域建模 ➤



DEMO:EA中进行域建模 >>>



高级话题：域模型的迭代 »»



登陆系统
Robustness图

高级话题：域模型≠数据模型 >>>

域模型

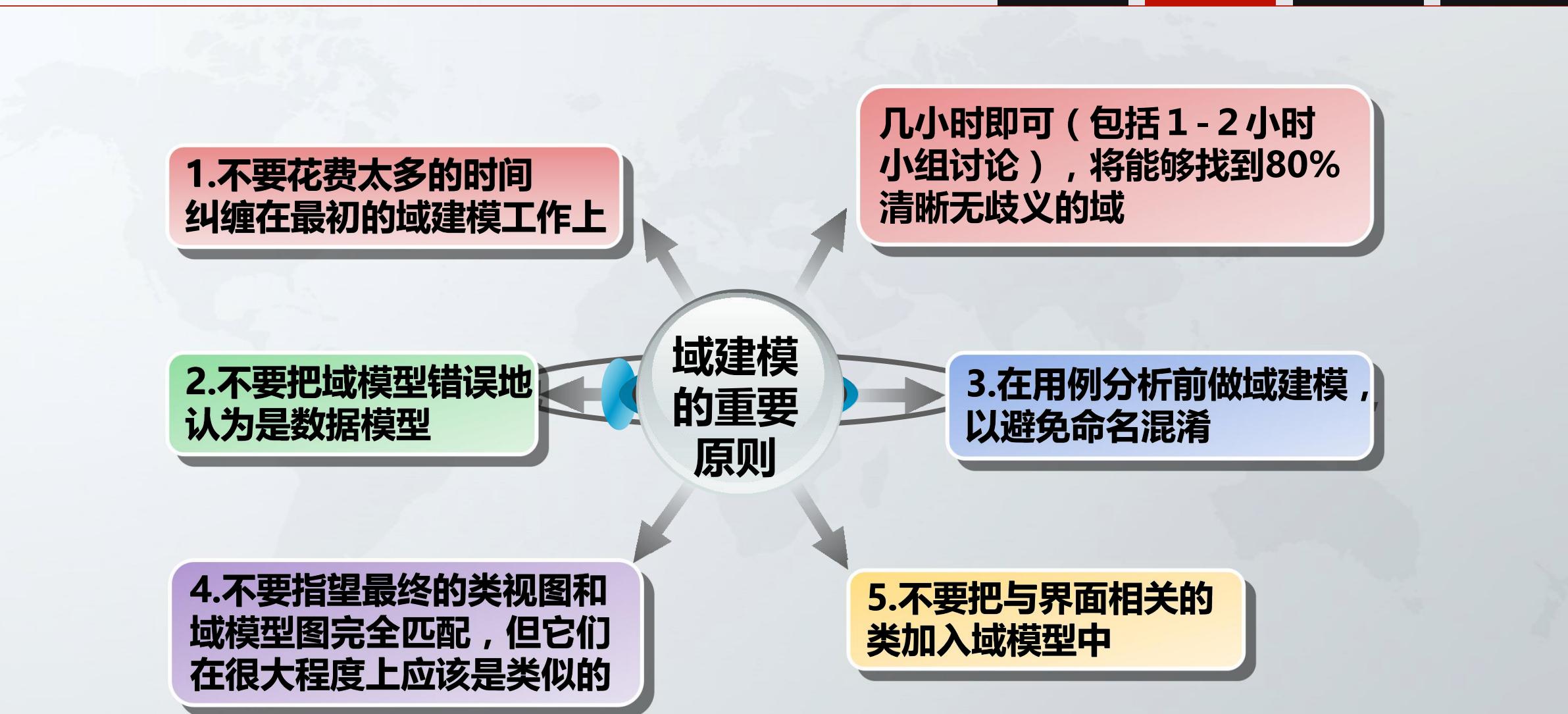
- 分析模型
- 系统分析员、用户认识现实业务的工具
- 描述业务实体及相互之间的关系

数据模型

- 系统设计、实现的一部分
- 描述的是对用户需求在数据结构上的实现

域模型设计期间不用考虑数据的存放问题，只考虑业务描述中涉及的实体以及实体之间的关系。

高级话题：域建模的重要原则



目录 >>>

一

需求分析的几种主流方法

二

域建模：以OO思想构建术语表

三

用例分析：系统功能性需求分析的好工具

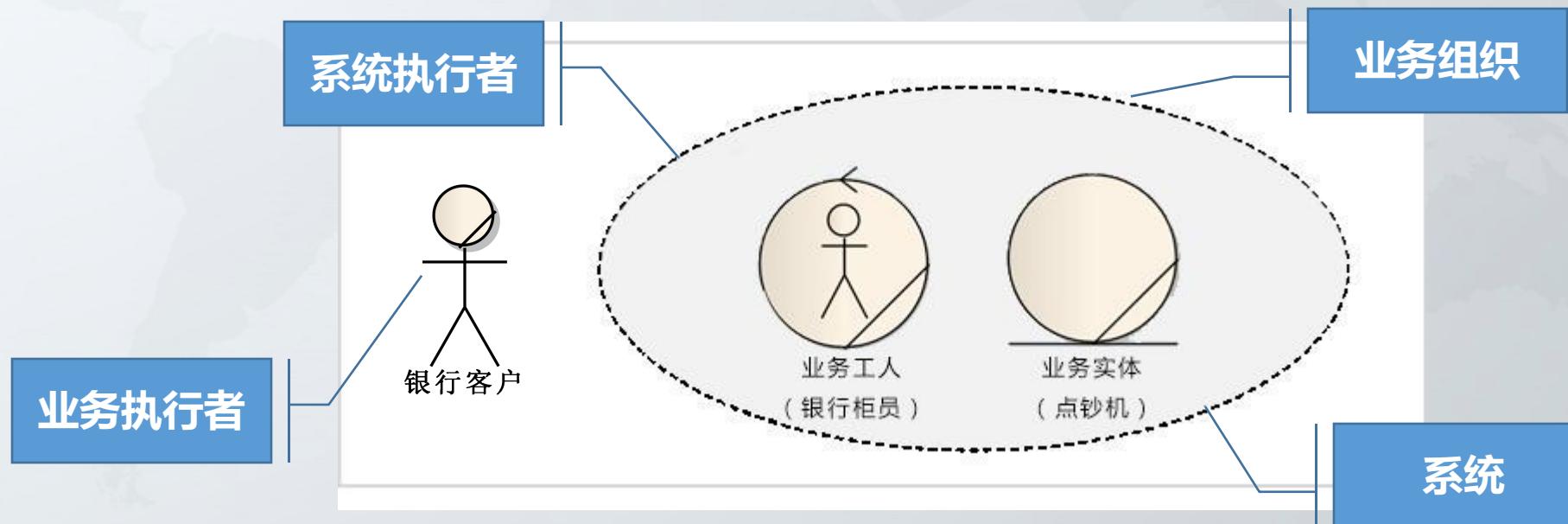
四

非功能性需求分析及需求定义与评审



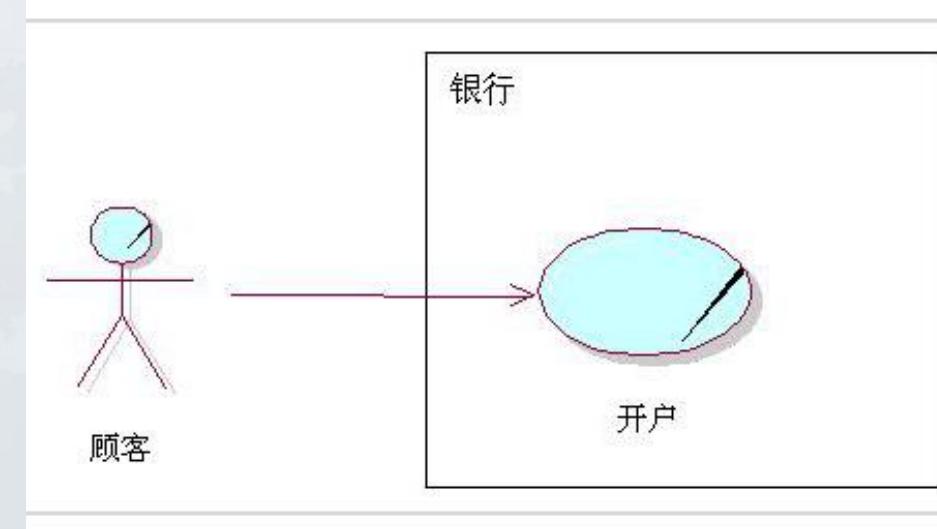
系统用例建模的意义 »»

- 系统需求分析的目的是把视角从**业务组织**转向**新系统**，站在**最终用户**及其它干系人的角度看问题。
- 系统用例是对（新）**系统**为系统执行者提供的**价值**的建模。

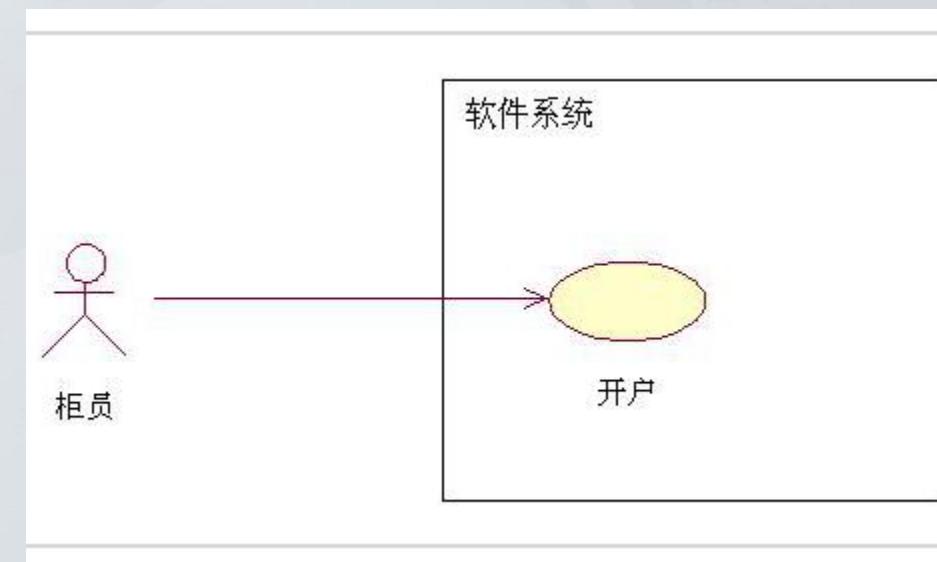


业务用例VS系统用例 >>>

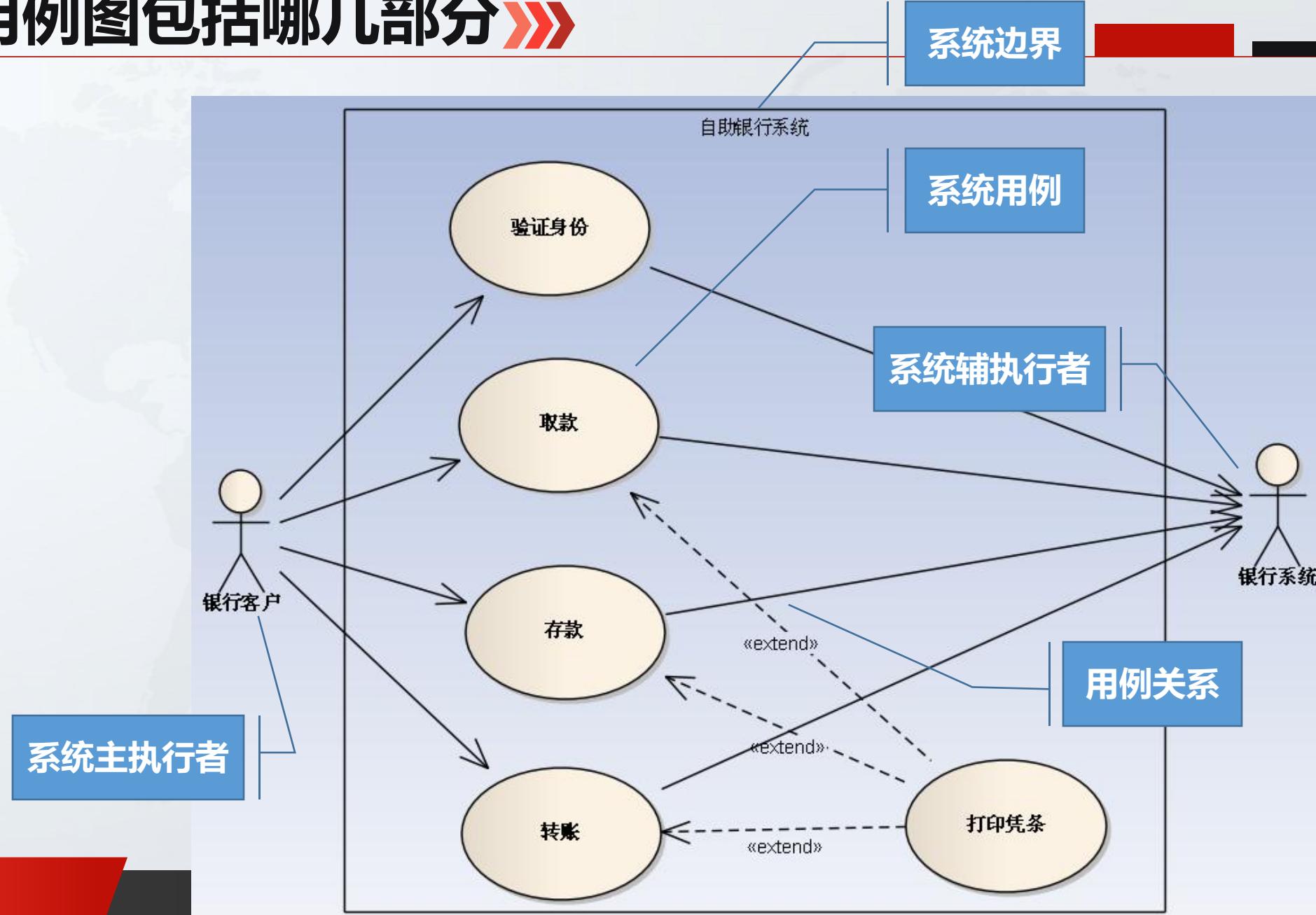
- 对银行进行**业务建模**，研究对象是**银行**。
- 用于分析现有业务的利与弊。



- 对银行的软件系统进行**系统建模**，研究对象是银行的**软件系统**。
- 用于分析新系统所带来的价值。



系统用例图包括哪几部分»»»



系统用例建模步骤 >>>

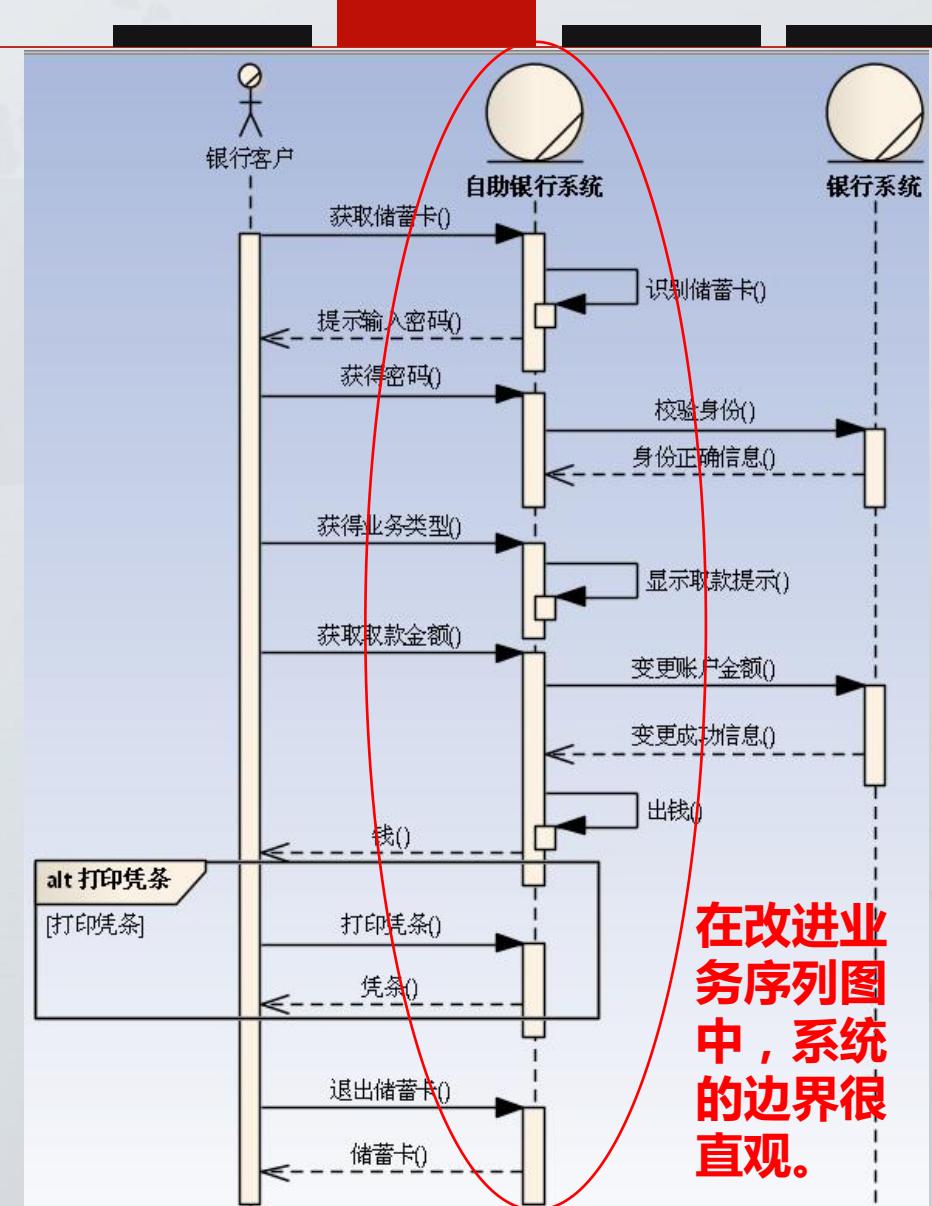
1. 绘制系统用例图
2. 编写系统用例描述
3. 更新域模型

绘制系统用例图 ➤

1. 确定系统边界
2. 识别系统执行者
3. 识别系统用例
4. 确定用例间的关系

第一步：确定系统边界»»

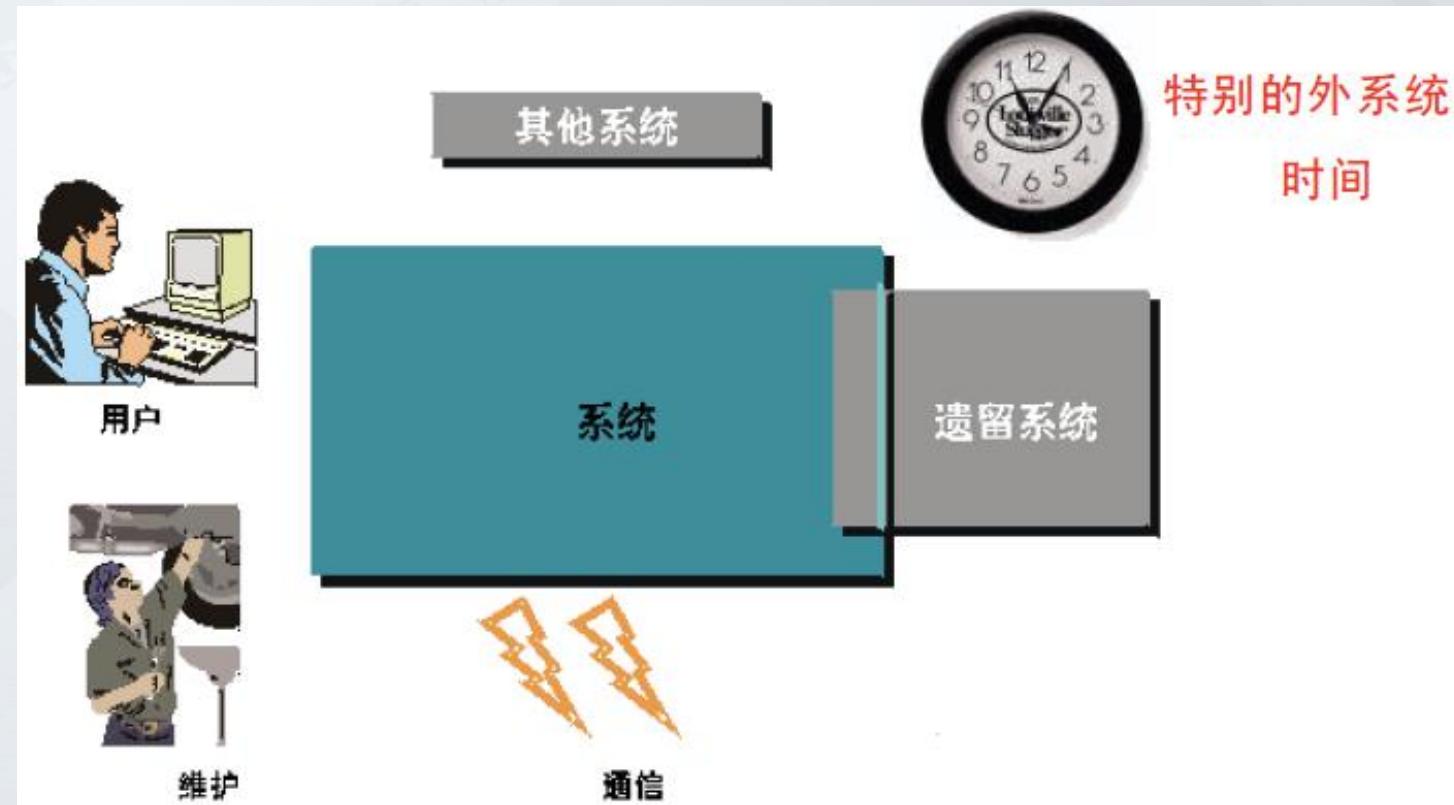
- 系统边界，即系统包含的功能与不包含的功能之间的界限。
- 通俗来说就是分割出系统内和系统外。
 - 系统内，需要我们投入大量的精力进行建设。
 - 系统外，需要我们考虑与它们的接口。



第二步：识别系统执行者 >>>

- 执行者[actor]是在**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**。

- 人
- 系统
- 硬件设备
- 时间
-



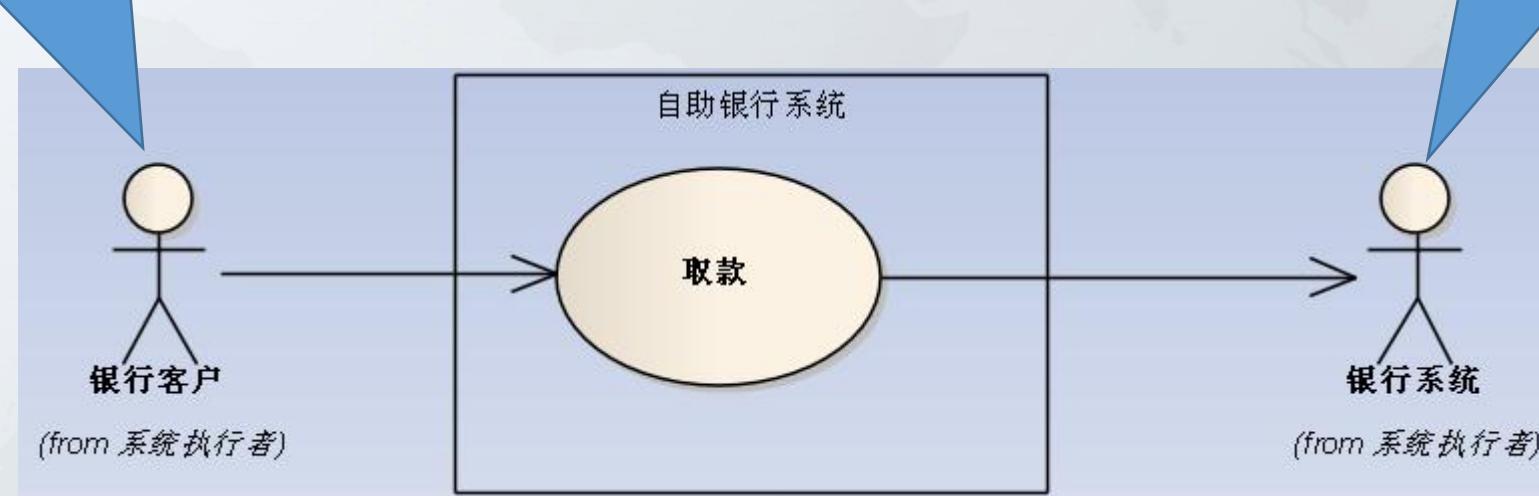
主执行者 VS. 辅执行者 »»

主执行者：

- 1.用例发起者；
- 2.用例为其实现有价值的目标；

辅执行者：

- 1.用例支持者；
- 2.用例的完成需要与其交互，得到其支持；



识别执行者的方法(思考+头脑风暴) >>>

- 谁使用该系统？
- 谁改变系统的数据？
- 谁从系统获取信息？
- 谁负责维护、管理并保持系统正常运行？
- 系统需要应付哪些硬件设备？
- 系统需要和哪些外部系统交互？
- 谁对系统运行产生的结果感兴趣？
- 有没有自动发生的事件？

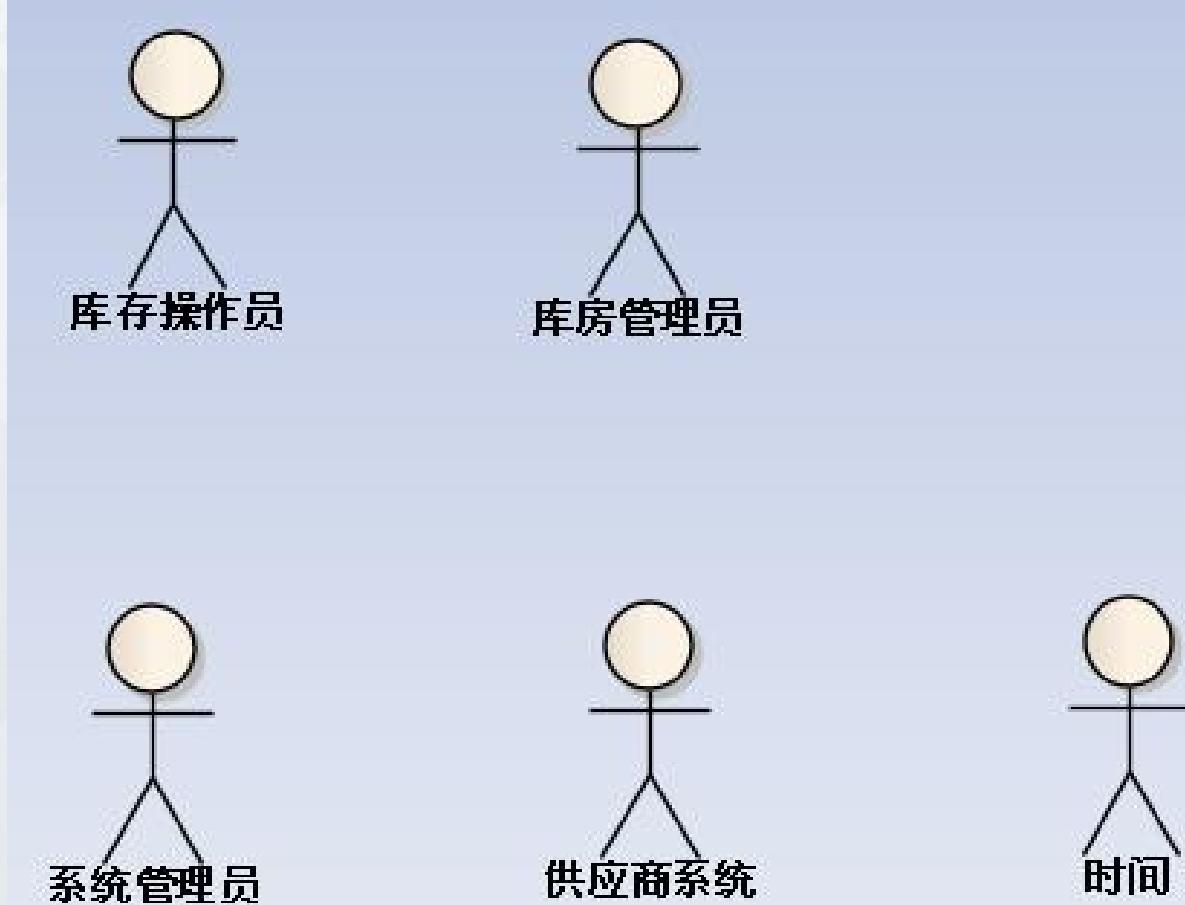
示例：识别执行者 >>>

- 某汽车制造厂需要一套物料管理系统，该系统实现的部分功能如下：
 - 工人领取物料，库存操作员交付物料给工人并在系统中记录物料变更情况；
 - 工人退还余料给库房，库存操作员接收物料并在系统中记录物料变更情况；
 - 库房管理员定期盘点库存；
 - 缺货时，库房管理员通过系统通知供应商系统供货；
 - 对积存的货物，库房管理员通过系统向供货商系统申请退货；
 - 每日晚11点系统自动备份数据；
 - 系统管理员负责维护系统正常运行；

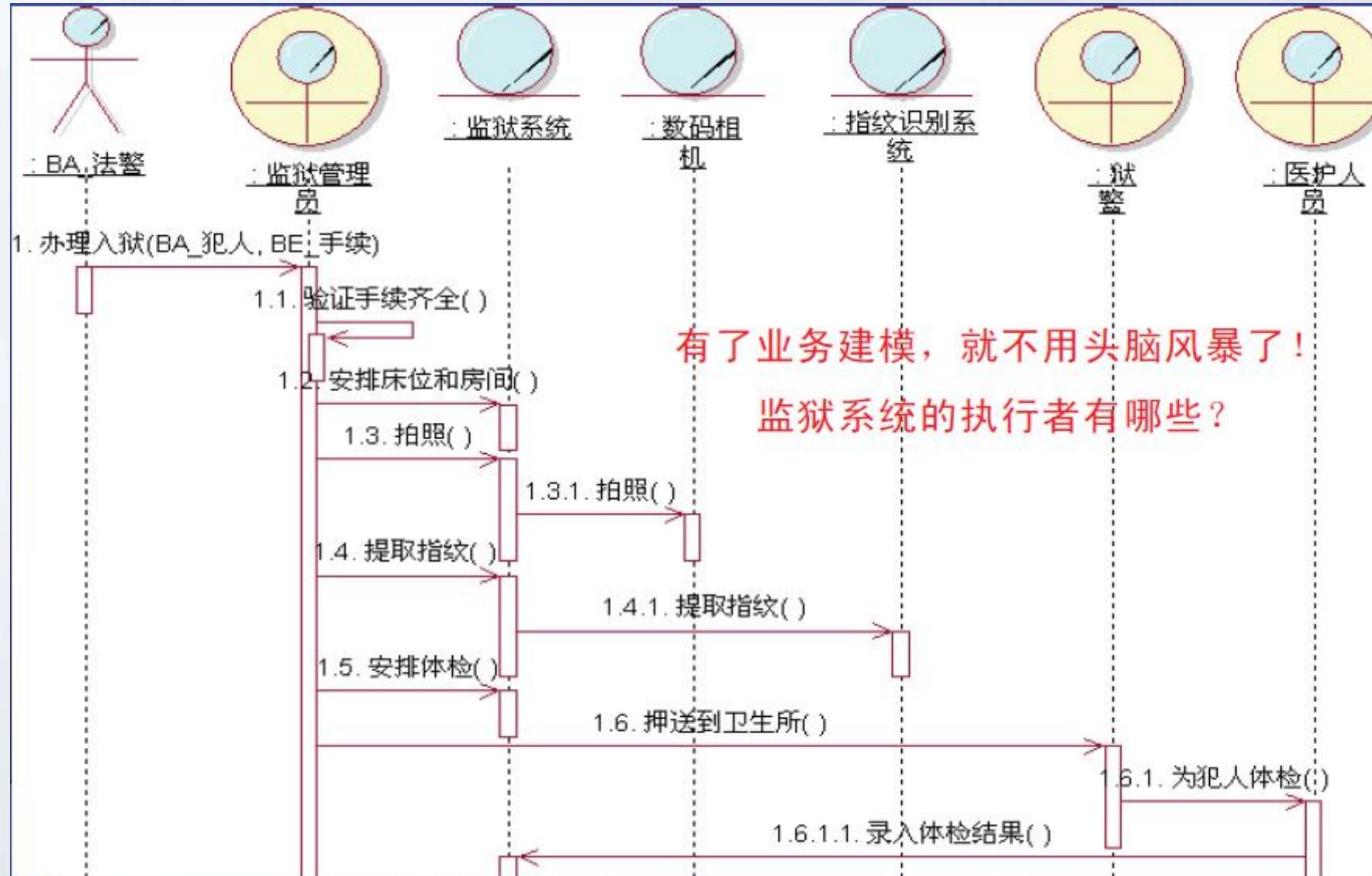
示例：识别执行者 >>>

- 谁使用该系统？ 库存操作员，库房管理员
- 谁改变系统的数据？ 库存操作员，库房管理员
- 谁从系统获取信息？ 库存操作员，库房管理员
- 谁负责维护、管理并保持系统正常运行？ 系统管理员
- 系统需要应付哪些硬件设备？ ✗
- 系统需要和哪些外部系统交互？ 供应商系统
- 谁对系统运行产生的结果感兴趣？ 库存操作员,库房管理员,供应商系统
- 有没有自动发生的事件？ 时间

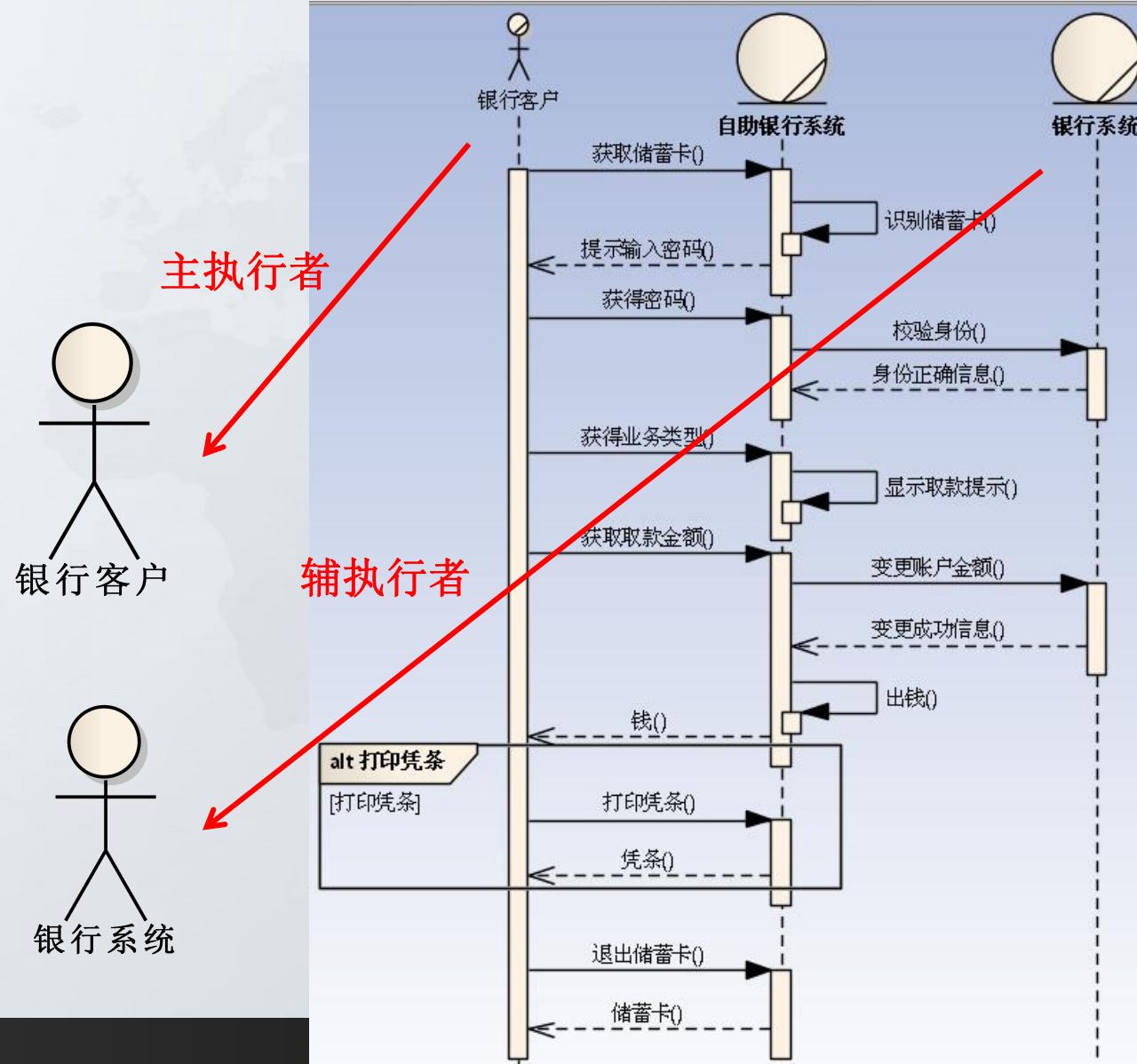
示例：识别执行者 ➤➤➤



如何在业务模型基础上找执行者 >>>



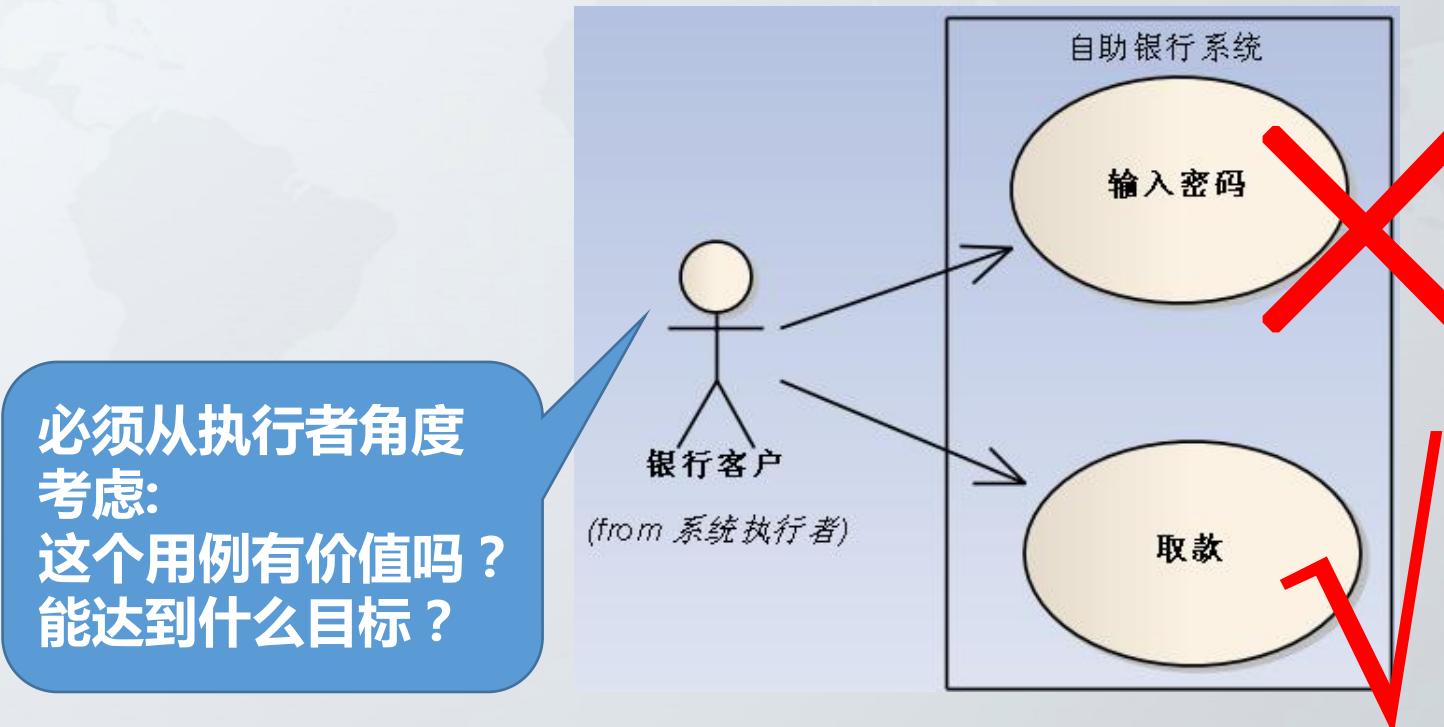
思考：自动银行系统的执行者？»»



第三步：识别系统用例»»

• 再谈用例（注意：这里是指**系统用例**，关注点由业务组织转向系统）

- **系统用例**是系统执行的一系列动作，这些动作可以生成“执行者”可观测的**有价值**的结果。
- 通俗讲：系统用例是执行者通过系统可以**达到的某个目标**。



识别用例的方法(思考+头脑风暴) >>>

- 执行者想要从系统获得什么有价值的功能？
- 系统存储信息吗？哪些执行者将建立、读取、更新或删除这些信息？
- 当系统内部状态有变化时，系统需要通知参与者吗？
- 系统需要定期执行什么操作吗？
- 当发生了某些外部事件时，系统需要自动执行什么操作吗？

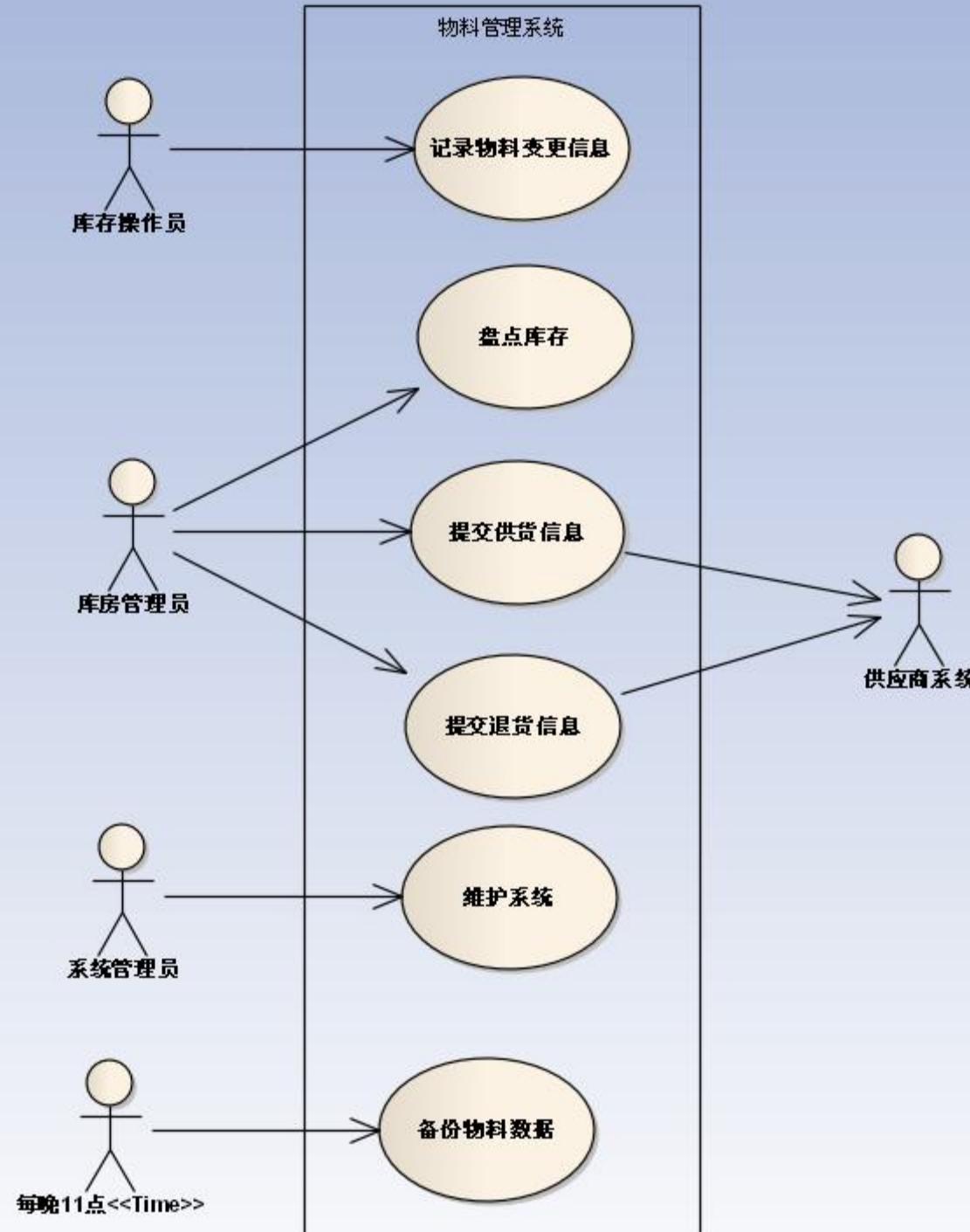
示例：识别系统用例

- 某汽车制造厂需要一套物料管理系统，该系统实现的部分功能如下：
 - 工人领取物料，库存操作员交付物料给工人并在系统中记录物料变更情况；
 - 工人退还余料给库房，库存操作员接收物料并在系统中记录物料变更情况；
 - 库房管理员定期盘点库存；
 - 缺货时，库房管理员通过系统通知供应商系统供货；
 - 对积存的货物，库房管理员通过系统向供货商系统申请退货；
 - 每日晚11点系统自动备份数据；
 - 系统管理员负责维护系统正常运行。

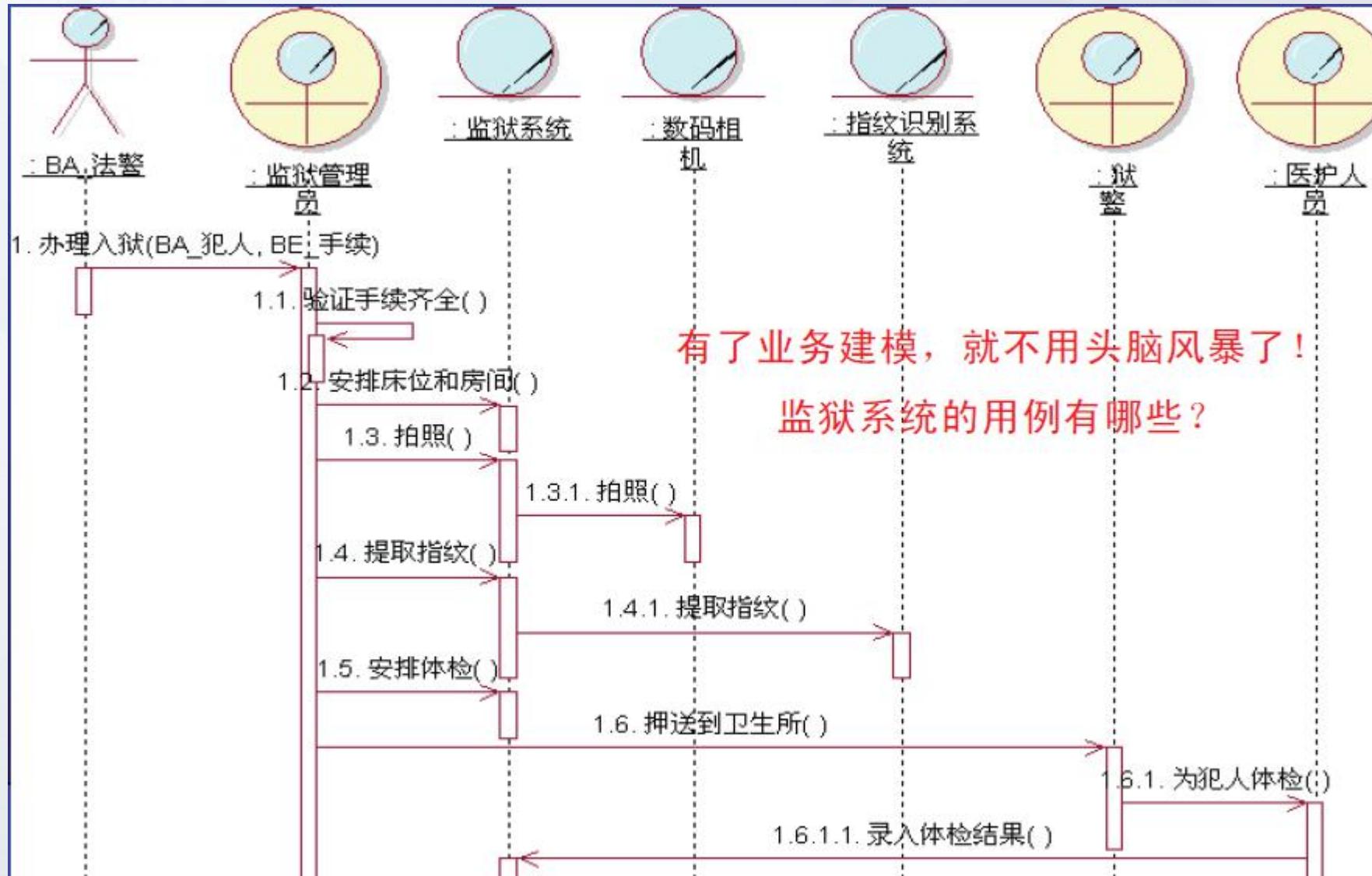
示例：识别系统用例 »»

- 工人领取物料，库存操作员交付物料给工人并在系统中**记录物料变更情况**；
- 工人退还余料给库房，库存操作员接收物料并在系统中**记录物料变更情况**；
- 库房管理员定期**盘点库存**；
 - 缺货时，库房管理员通过系统**通知供应商系统供货**；
 - 对积存的货物，库房管理员通过系统向供货商系统**申请退货**；
- 每日晚11点系统自动**备份数据**；
- 系统管理员负责**维护系统**正常运行。

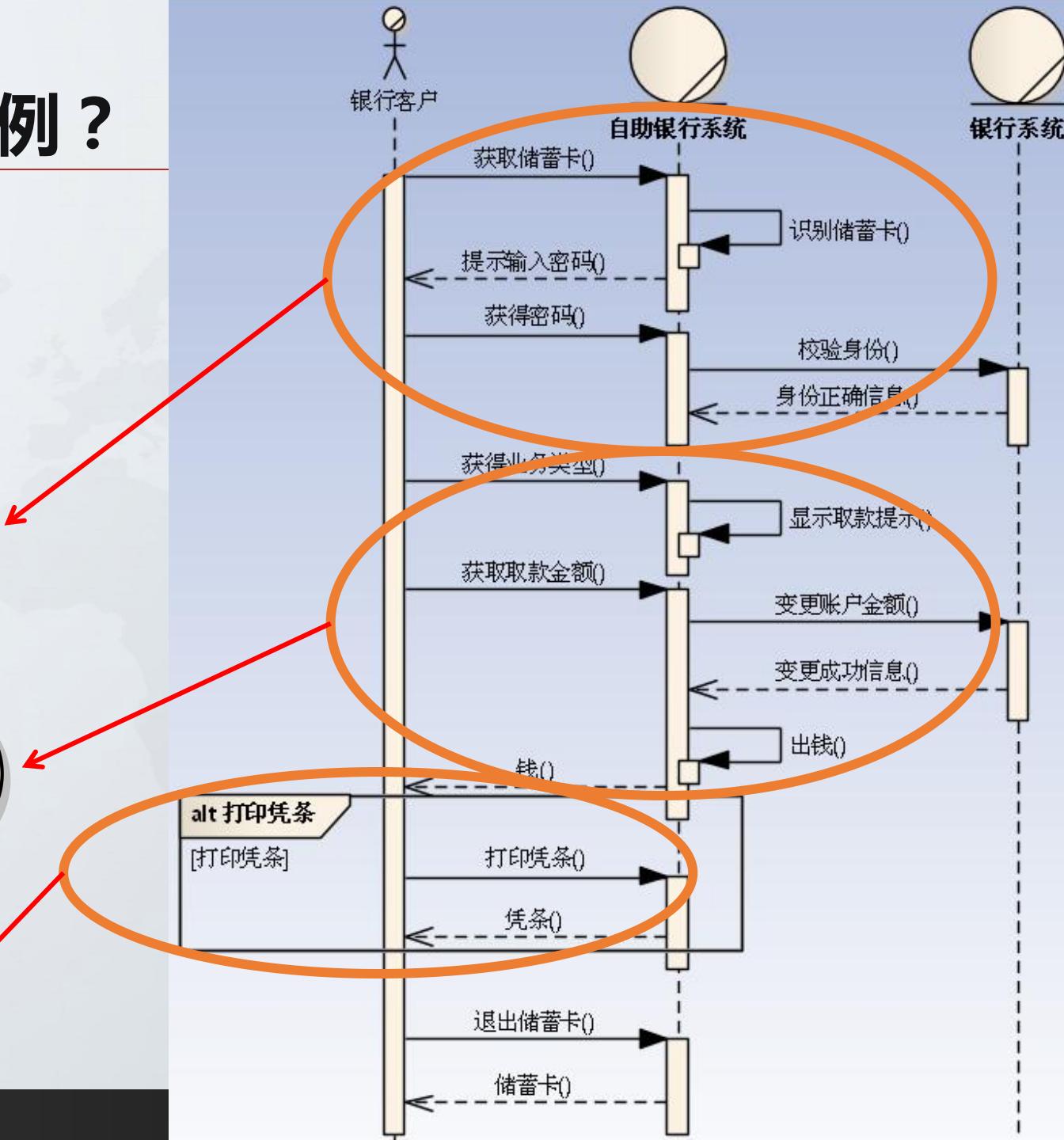
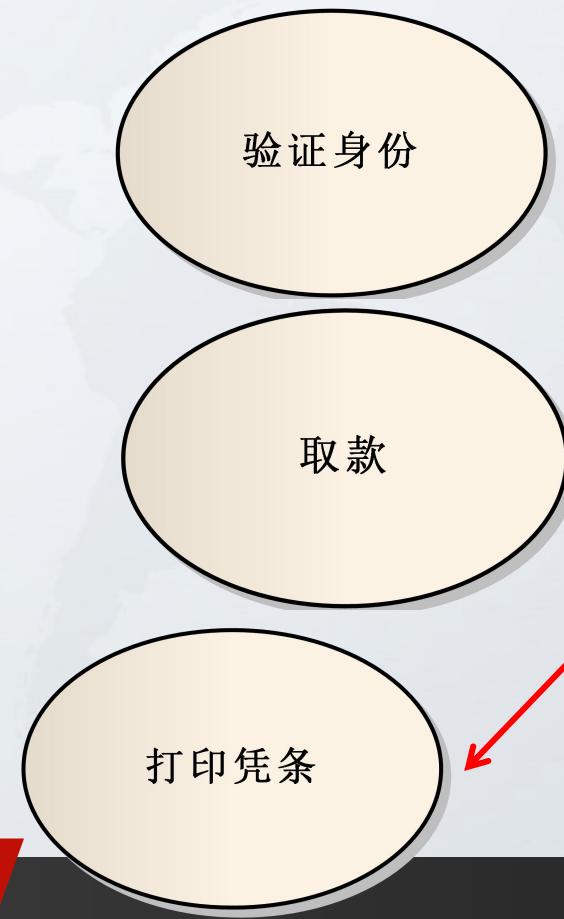
示例：识别系统用例



如何在业务模型基础上找用例

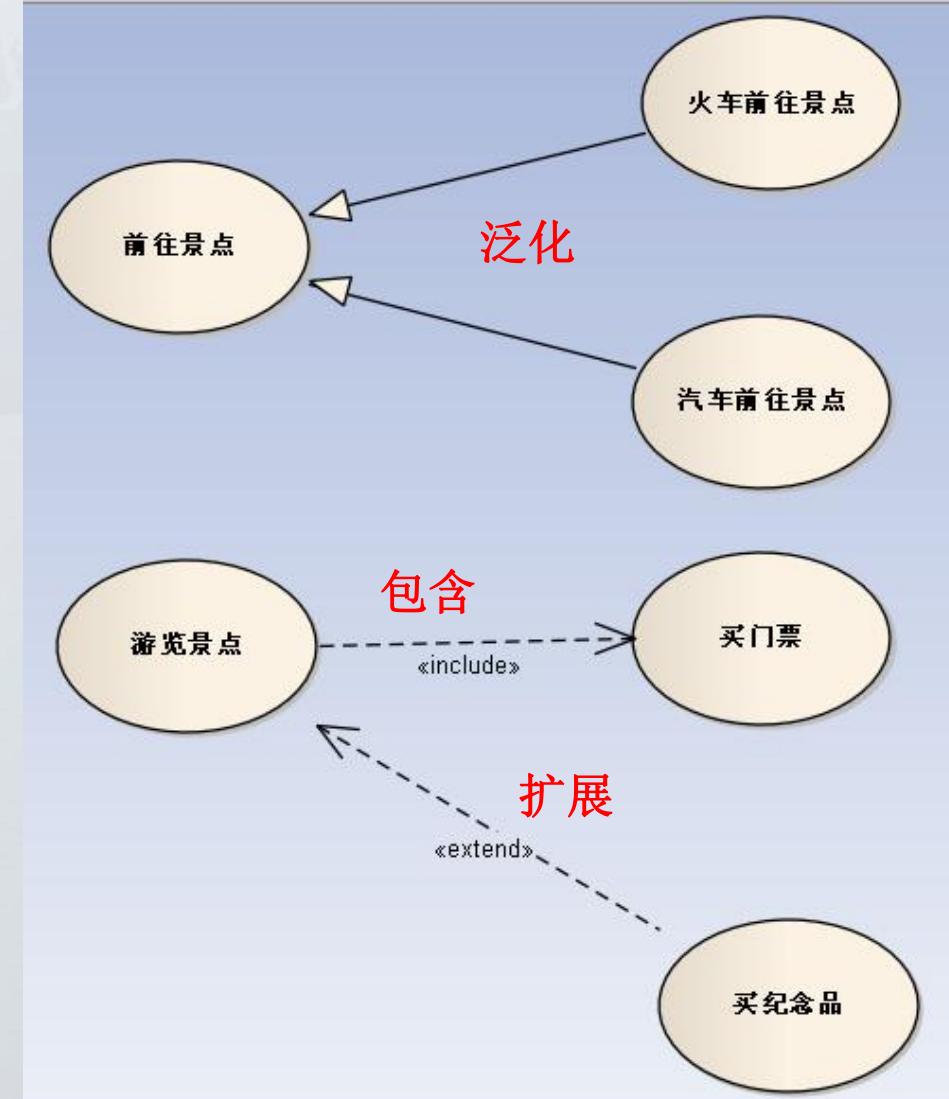


思考：自动银行系统的用例？

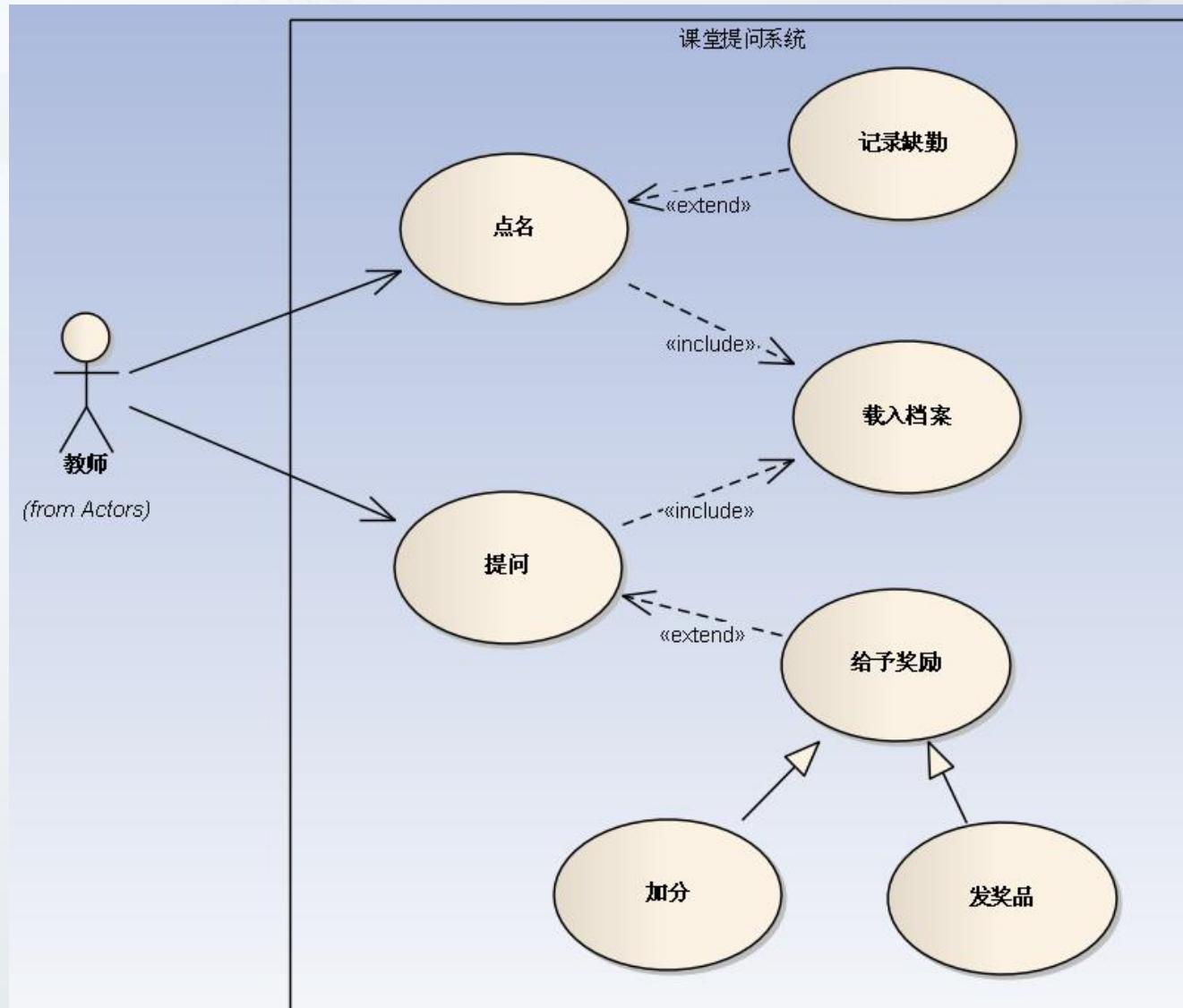


第四步：确定用例间的关系 ➤➤➤

- 用例之间的关系：
 - 泛化[Generalize]。
 - 包含[Include]。
 - 扩展[Extend]。
- 思想：从现有的用例中抽取出公共的那部分信息，作为一个单独的用例。然后通过不同的方法来**重用**这个公共的用例，以减少模型维护的工作量。

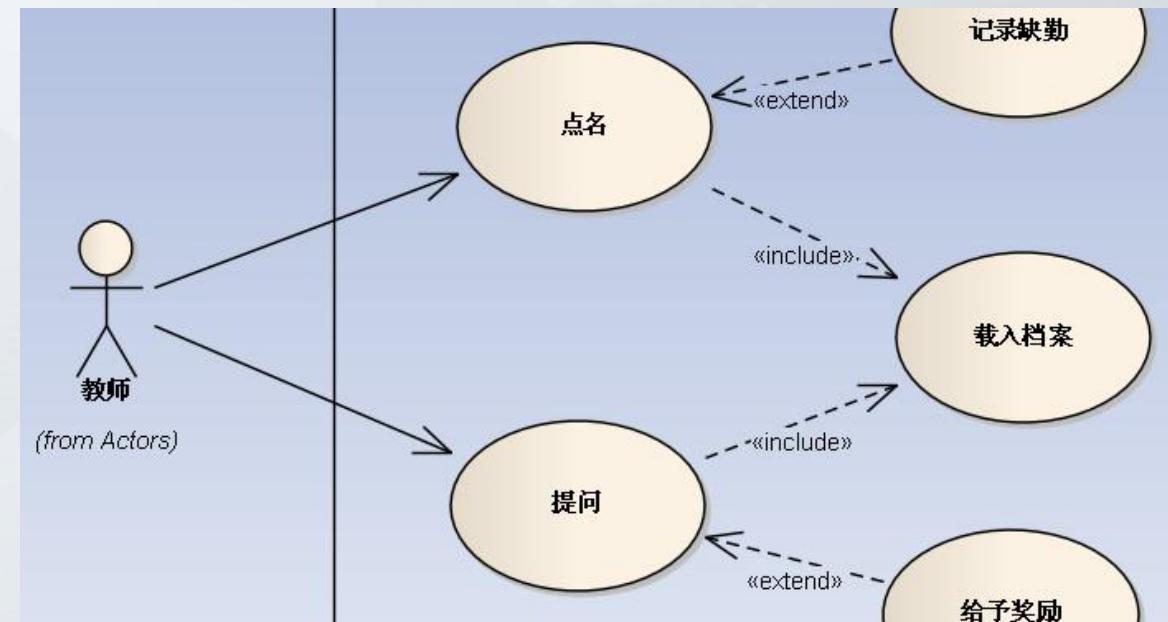


示例：“课堂提问系统”的用例关系 »»



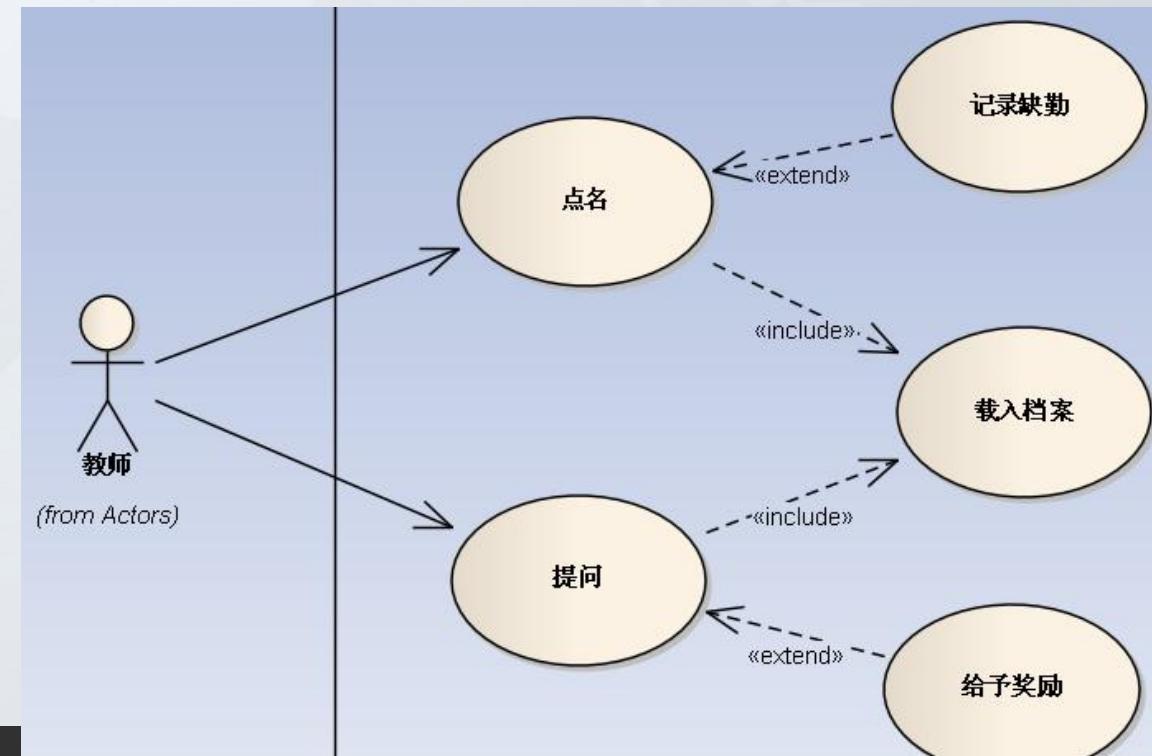
思考：包含关系的适用场景 ➤➤➤

- 包含关系：使用包含用例来封装一组跨越多个用例的相似动作（行为片段），以便多个基用例复用。



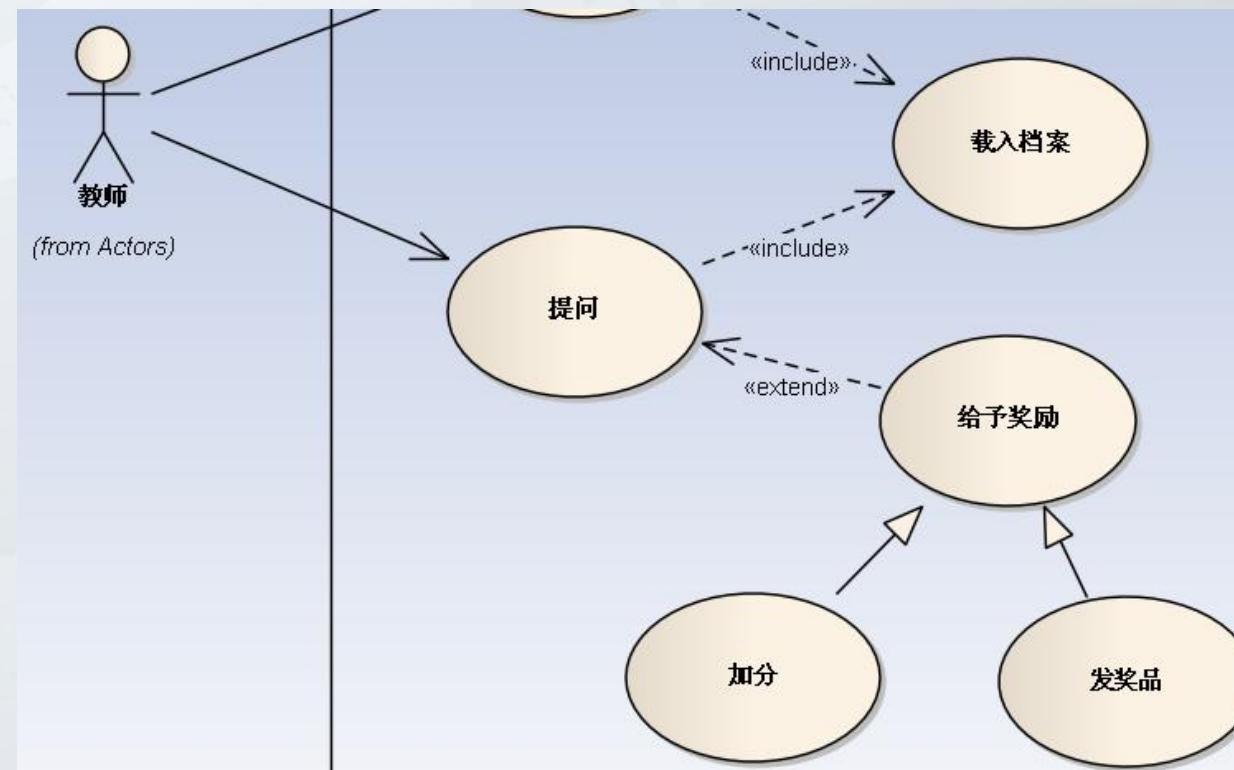
思考：扩展关系的适用场景 »»

- 扩展关系：将基用例中一段相对独立并且可选的动作，用扩展用例加以封装，再让它从基用例中声明的扩展点上进行扩展，从而使基用例行为更简练和目标更集中。

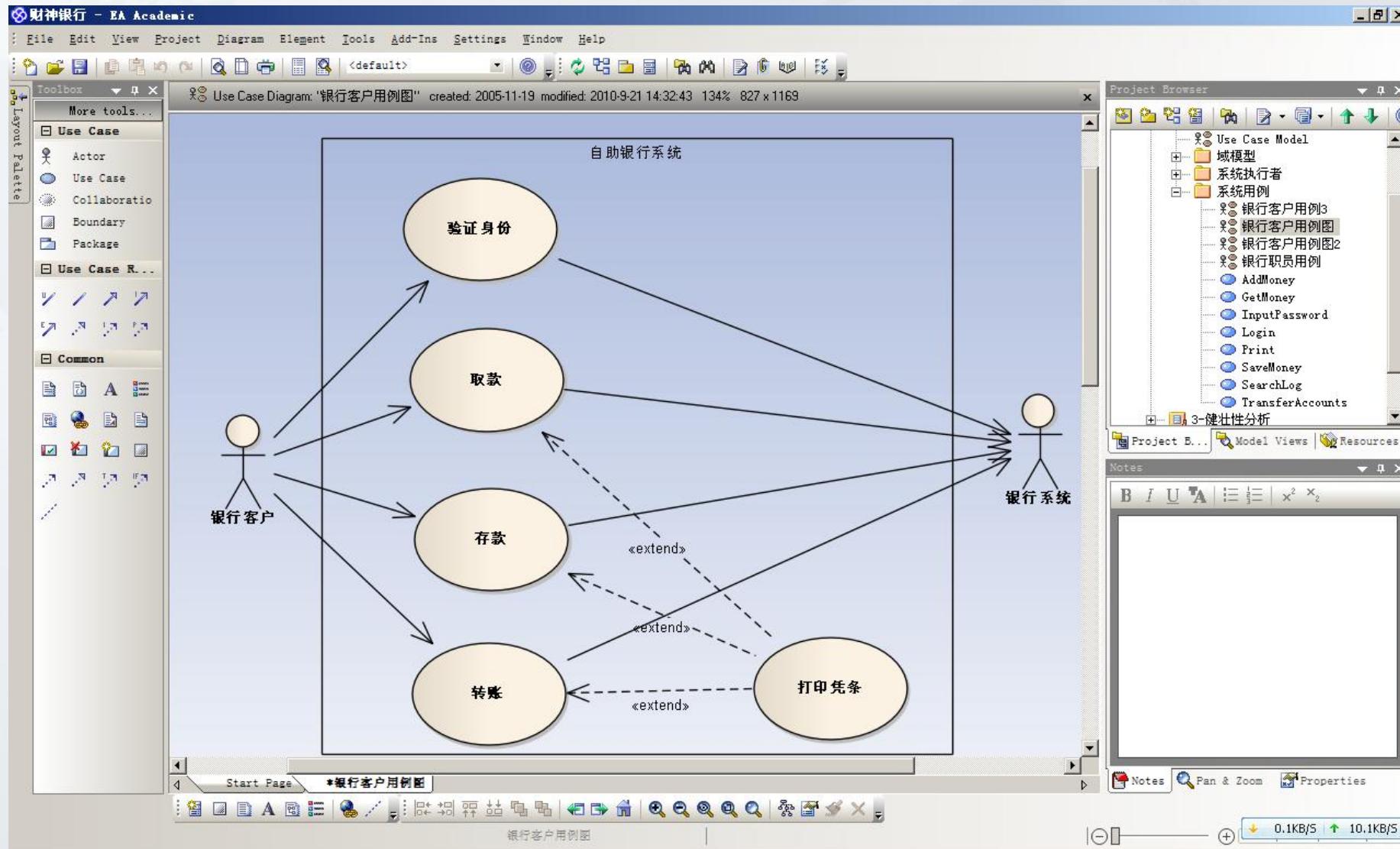


思考：泛化关系的适用场景 »»

- 泛化关系：子用例和父用例相似，但表现出更特别的行为；子用例将继承父用例的所有结构、行为和关系。



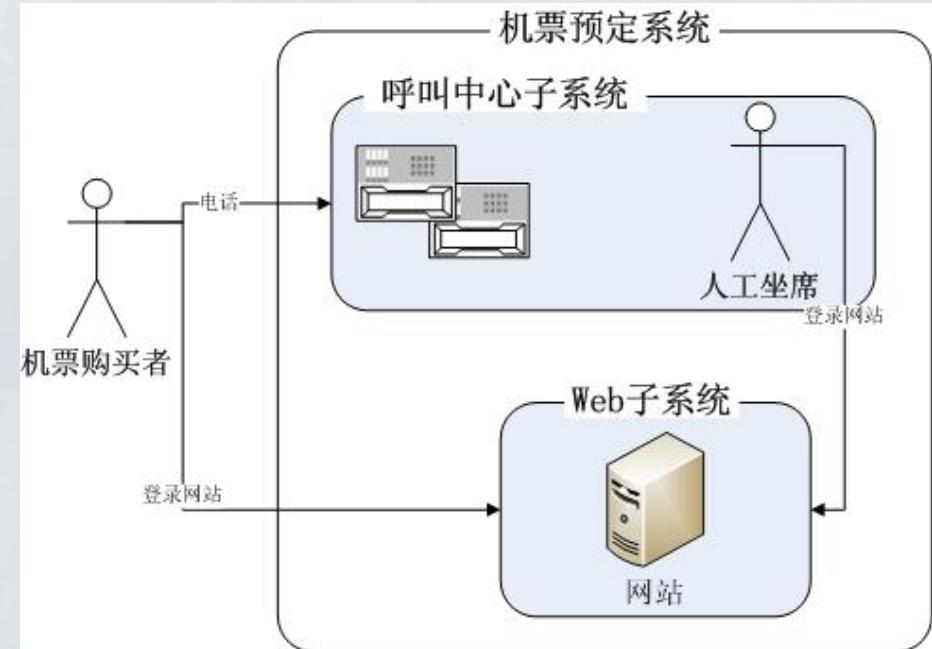
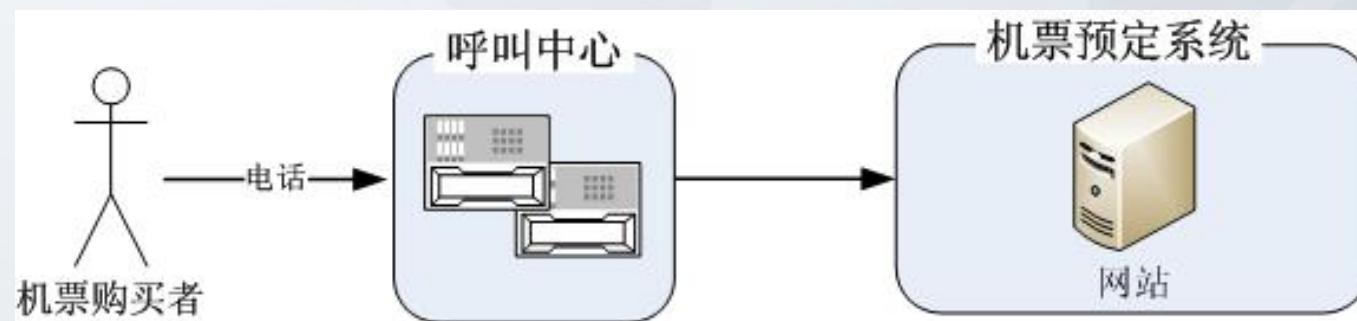
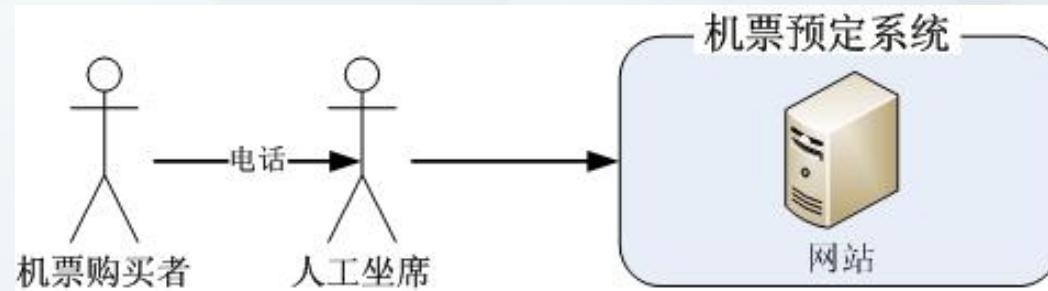
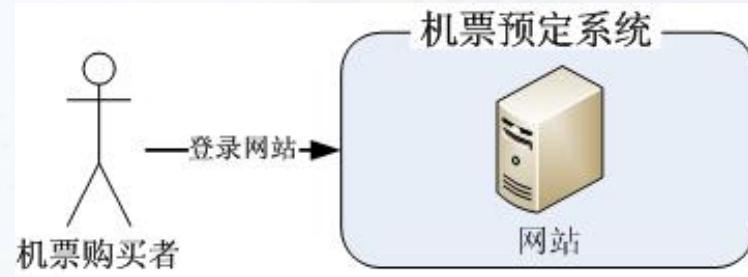
DEMO:EA中进行系统用例建模 >>>



高级话题:先发现执行者还是先发现用例? >>>

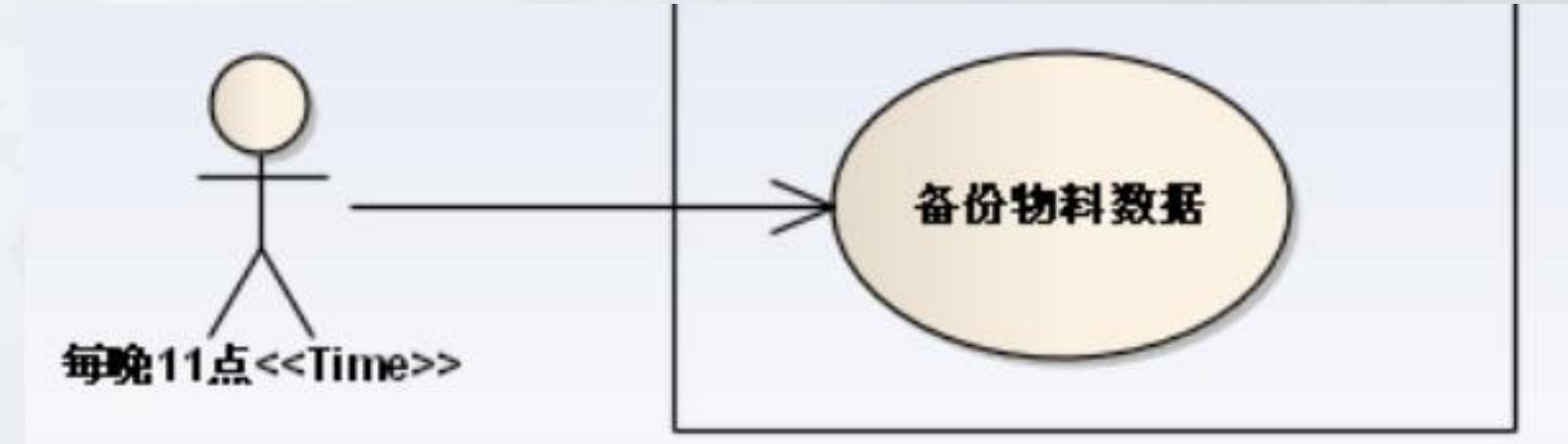
- 执行者比用例明显。
- 执行者的个数远比用例的个数少。
- 找到一个执行者，就可以找到一堆用例。
- 执行者是系统外部人物的代表，所以当然是要先找到执行者，才能够从执行者的角度去寻找用例。

高级话题:执行者与重要性无关



高级话题：‘时间’ 执行者 »»

- 用 ‘时间’ 执行者来标示预定的事件。



高级话题:用例的命名

- 用例名称必须是动宾短语。
- 采用域建模中的名词术语。
- 慎用弱动词弱名词——会掩盖真正的业务。
 - 弱动词：进行、使用、复制、加载、生成.....
 - 弱名词：数据、报表、表格、表单、系统.....



高级话题：用例≠功能 ➤➤➤

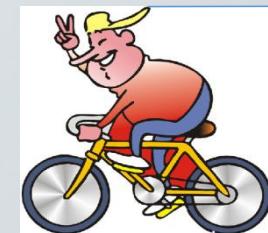
- 描述一个事物的三个出发点
 - 这个事物是什么？——**结构**
 - 这个事物能做什么？——**功能**
 - 人们能够用这个事物做什么？——**价值**



刹车系统



前进



上班代步

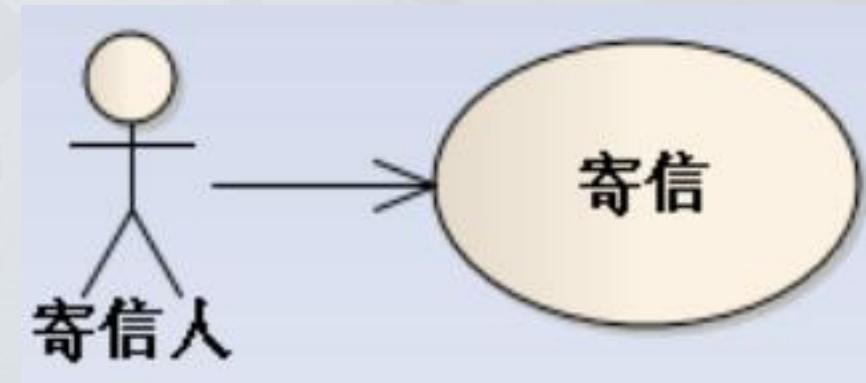
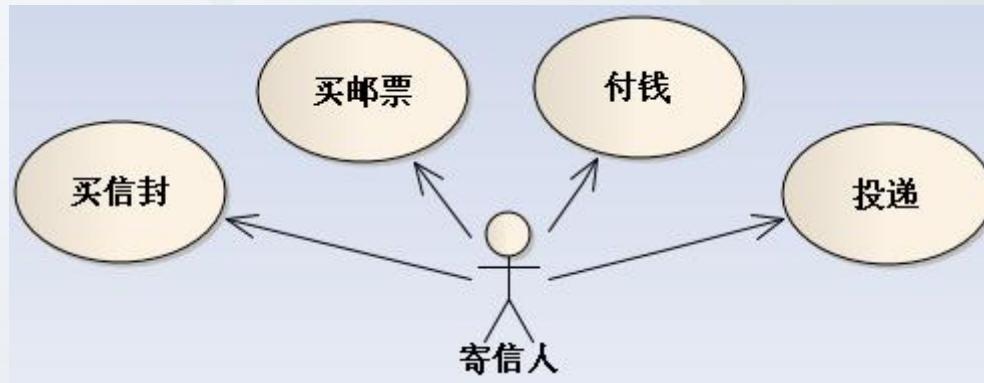
传动系统

转弯

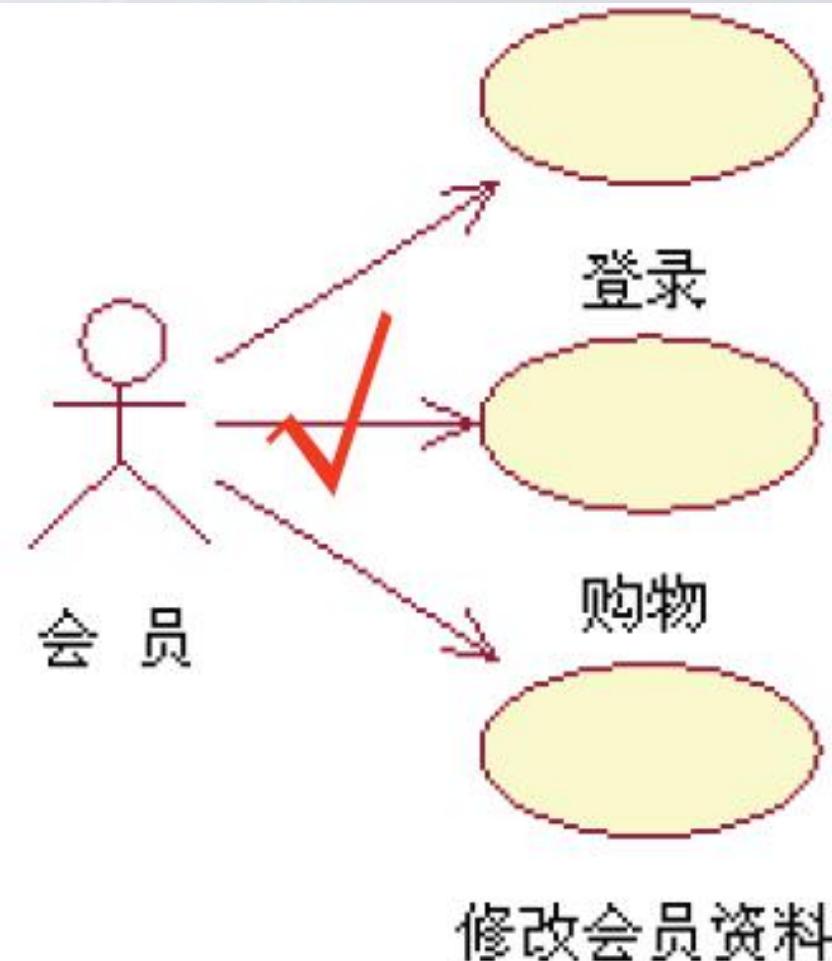
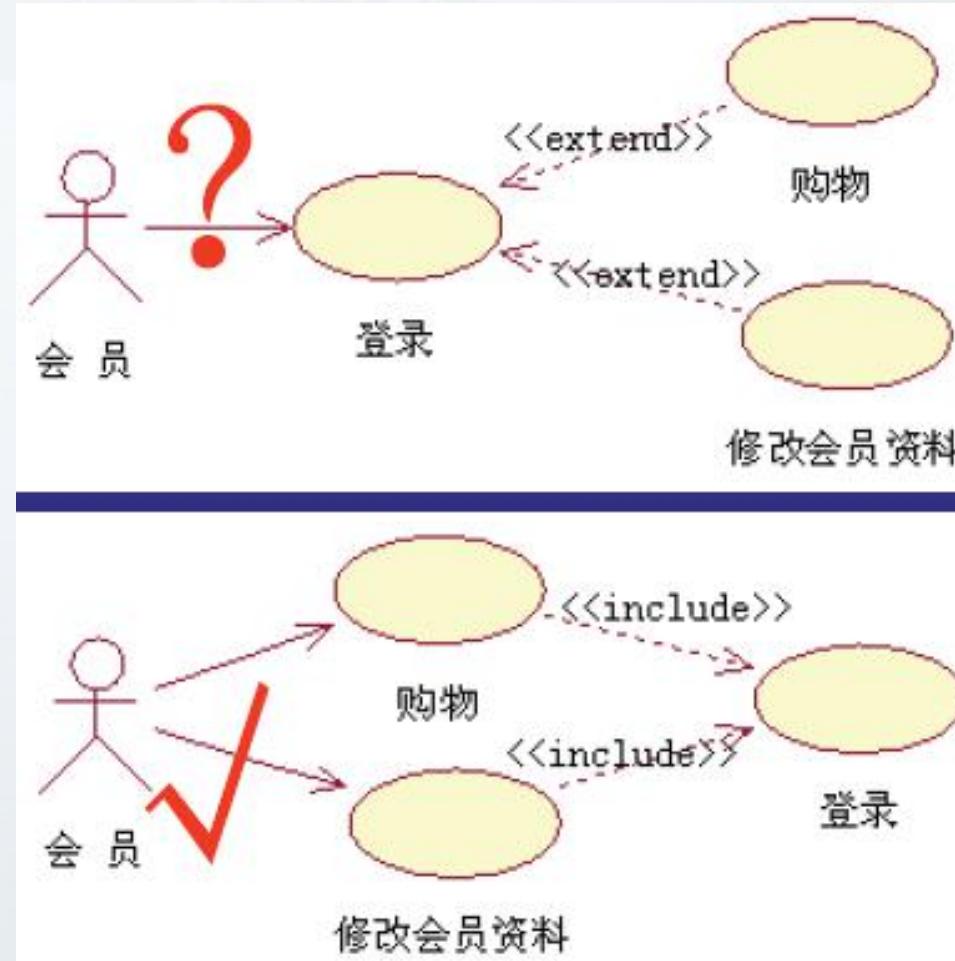
接送女朋友

高级话题：用例≠步骤 »»

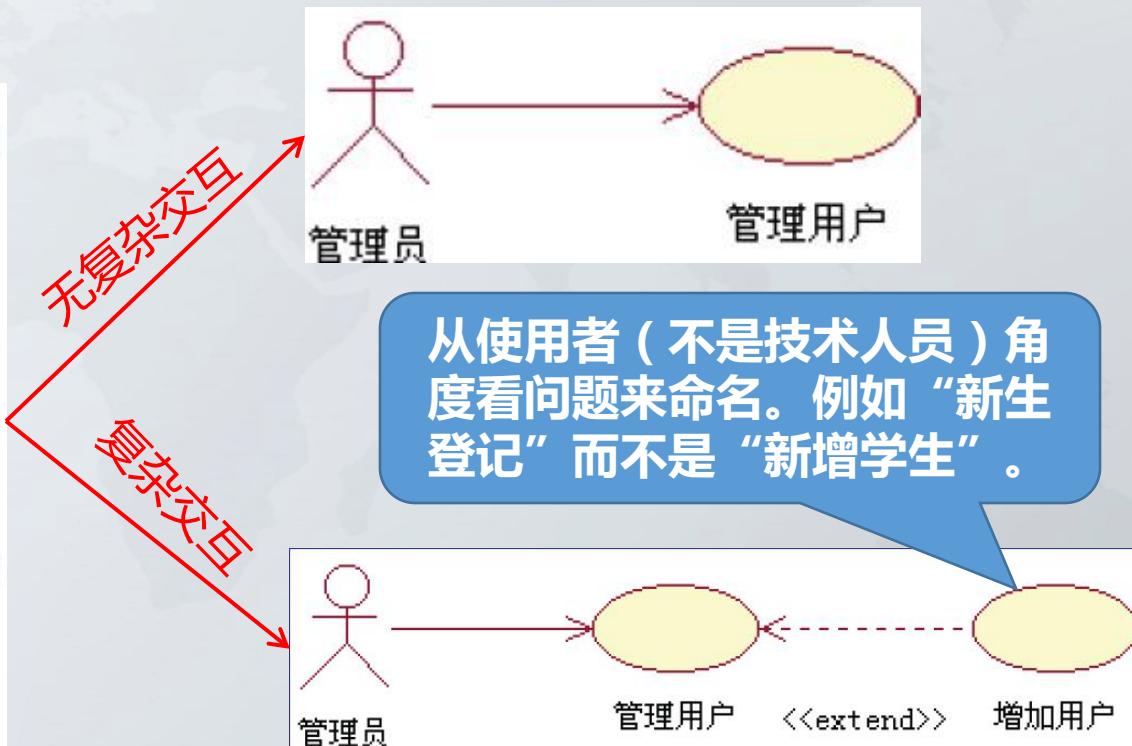
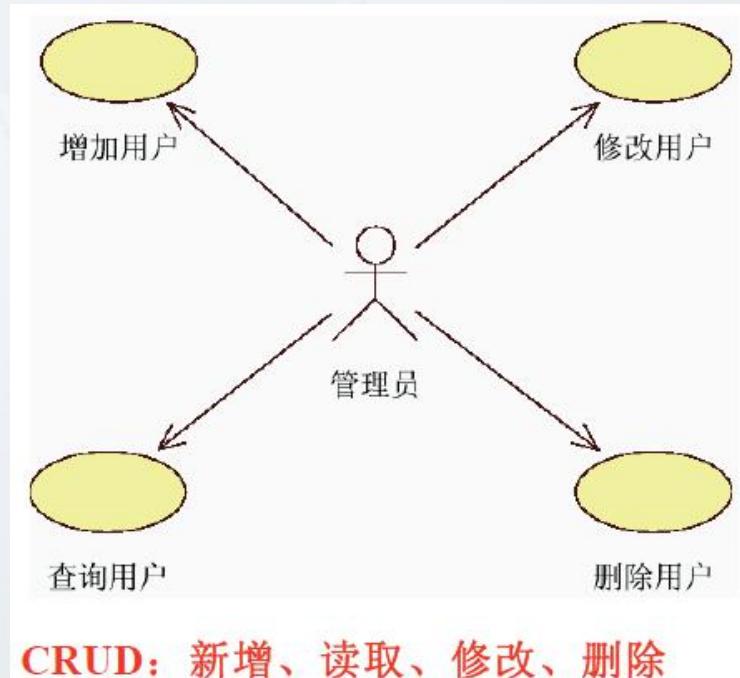
- 用例是执行者对系统的一个愿望。
- 完成这个愿望可能需要经过很多步骤，但任何步骤不能够完整的反映执行者的目标。



高级话题：怎么处理登录 >>>



高级话题：用例的“四轮马车”»»



高级话题：用例与愿景目标 »»

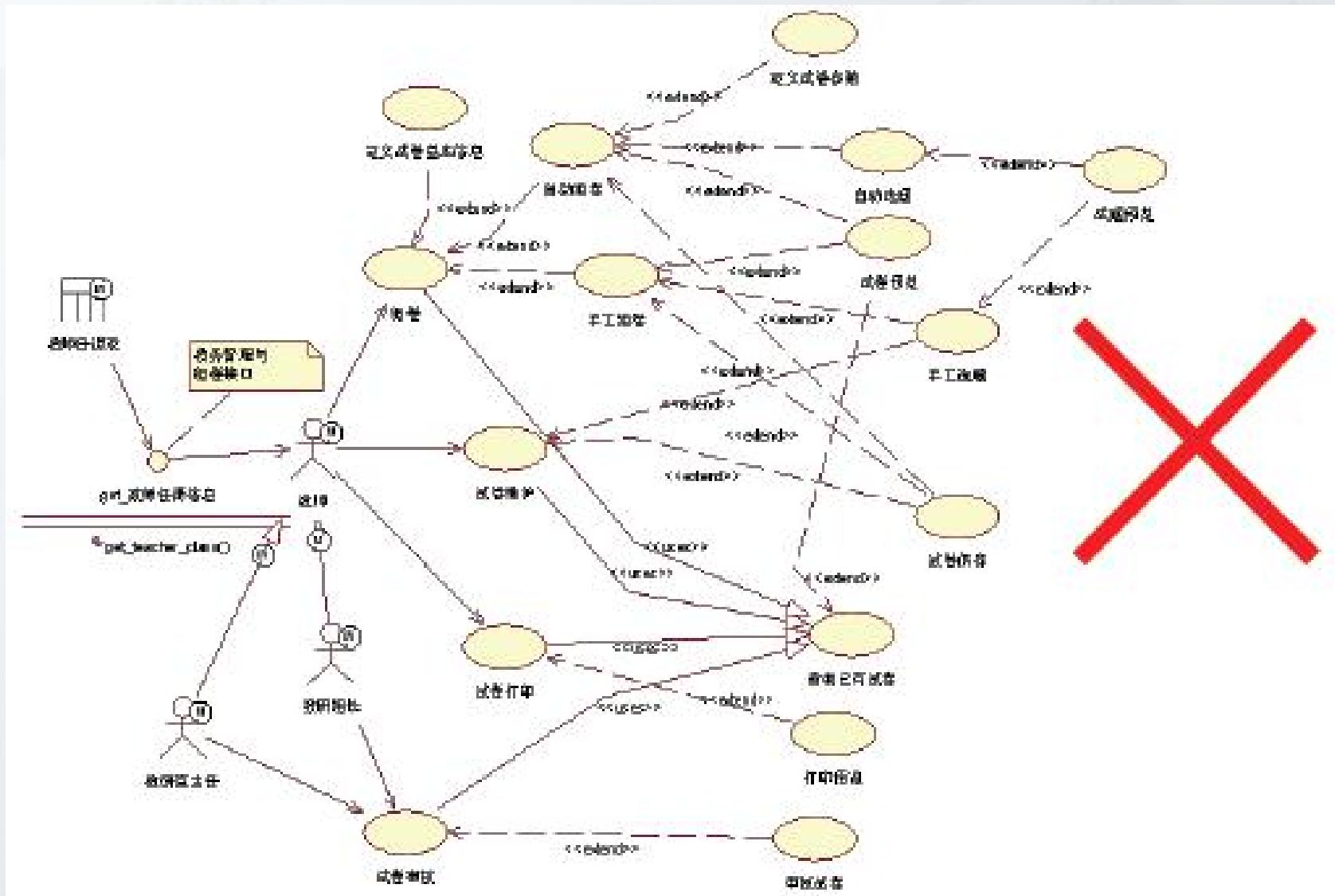
- 所有用例应该都能追溯到愿景目标。
- 所有的愿景目标都应有对应的用例实现。

	用例	用例	用例	用例	用例
目标		●	●		
目标	?	●			?
目标	?		●		?
目标			●	●	
目标			?	●	

高级话题：用例分包 ➞

- 当存在大量用例时可以对用例进行分包
 - 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
 -

高级话题：不要滥用用例关系

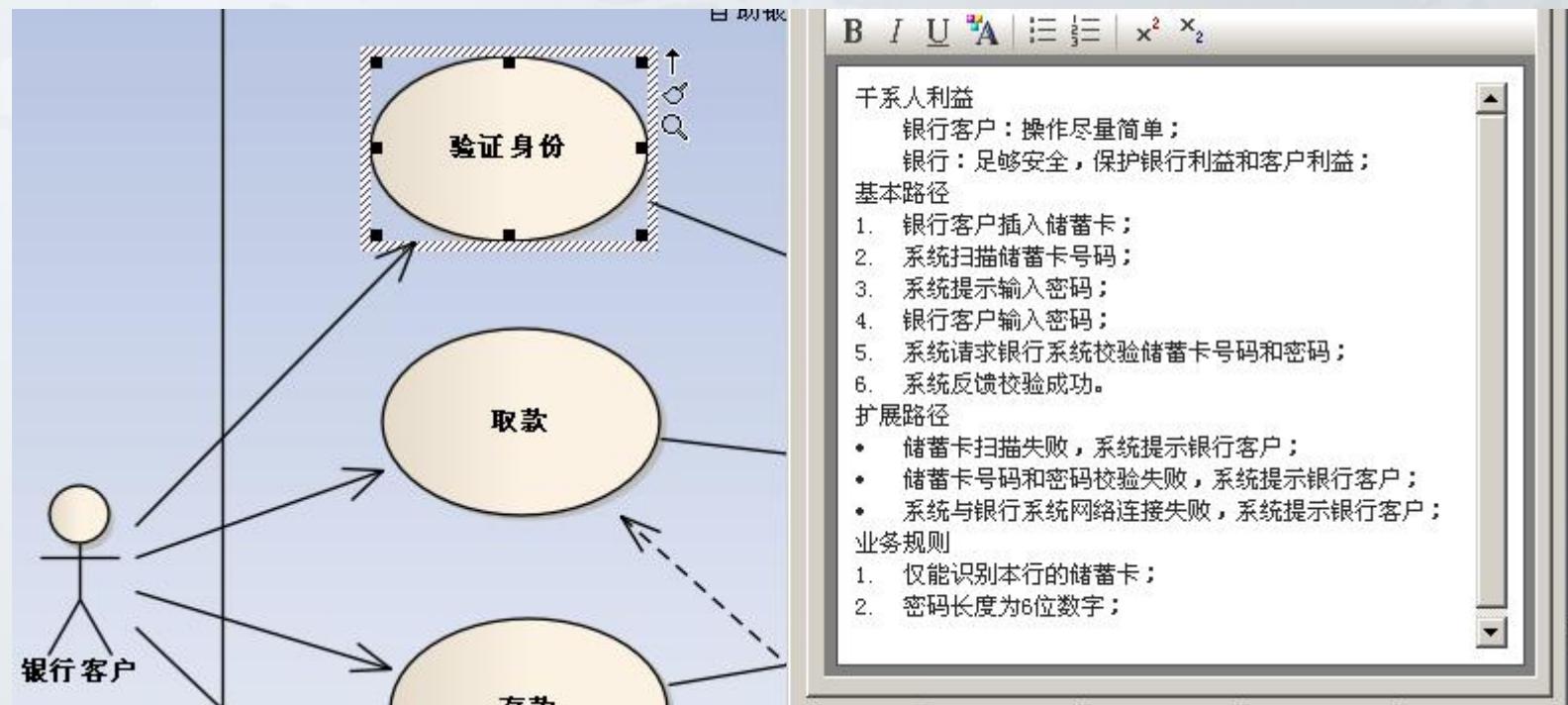


系统用例建模步骤 >>>

1. 绘制系统用例图
2. 编写系统用例描述
3. 更新域模型

用例描述的作用 »»

- 系统用例图描述总体，系统用例文档描述细节。
- 每个系统用例必须对应有系统用例描述。



用例描述的基本组成 »»

- 干系人利益
- 基本路径
- 扩展路径
- 业务规则

干系人利益

银行客户：操作尽量简单；

银行：足够安全，保护银行利益和客户利益；

基本路径

1. 银行客户插入储蓄卡；
2. 系统扫描储蓄卡号码；
3. 系统提示输入密码；
4. 银行客户输入密码；
5. 系统请求银行系统校验储蓄卡号码和密码；
6. 系统反馈校验成功。

扩展路径

- 储蓄卡扫描失败，系统提示银行客户；
- 储蓄卡号码和密码校验失败，系统提示银行客户；
- 系统与银行系统网络连接失败，系统提示银行客户；

业务规则

1. 仅能识别本行的储蓄卡；
2. 密码长度为6位数字；
3. 3次以上密码输入错误，系统将吞掉储蓄卡；

为什么要考虑干系人利益？»»»



同样是“取钱”：
为什么家里的抽屉不用密码，取款机要用？
为什么银行不放一个箩筐在门口？
为什么取了钱以后要写“系统扣除账户金额”？

同样的目的，不同的涉众群体

系统用例描述中的干系人及其利益 »»

- 用例体现干系人利益的平衡点。
 - 谁关心这个系统？
 - 涉及他的什么利益？
 - 如何平衡之间的利益？



涉众利益

用户——希望方便
银行——希望安全，希望节约运营成本。

法律——保护个人财产

基本路径

1. 用户插入 ATM 卡
2. 系统要求输入合法的密码
3. 用户输入密码
4. 系统验证密码合法 正确
5. 系统提示用户输入取款金额
6. 用户输入取款金额并确认
7. 系统检测取款金额合法
8. 系统从帐户扣除取款金额
9. 系统吐钞
10. 系统提示用户“打印收据”或者“不打印收据”
11. 用户要求不打印收据
12. 系统显示“交易结束”，退卡

业务规则

1. 密码为 6 位数字
2. 金额必须为 100 元的倍数
3. 当日取款金额不能超过 5000 元

利益的冲突

银行的

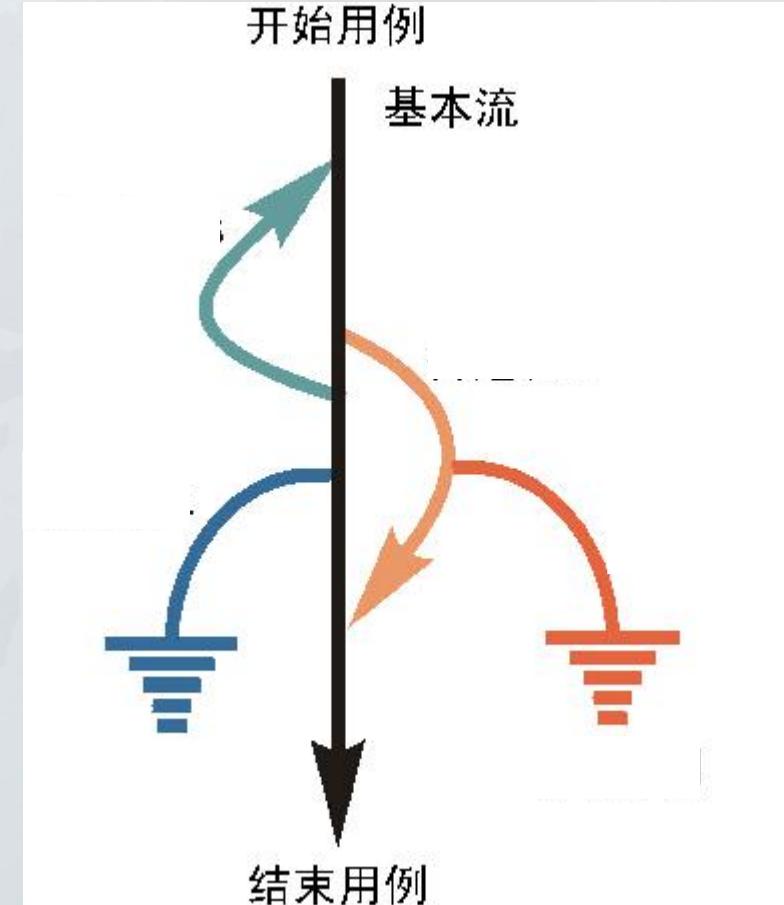
用户的

法律的

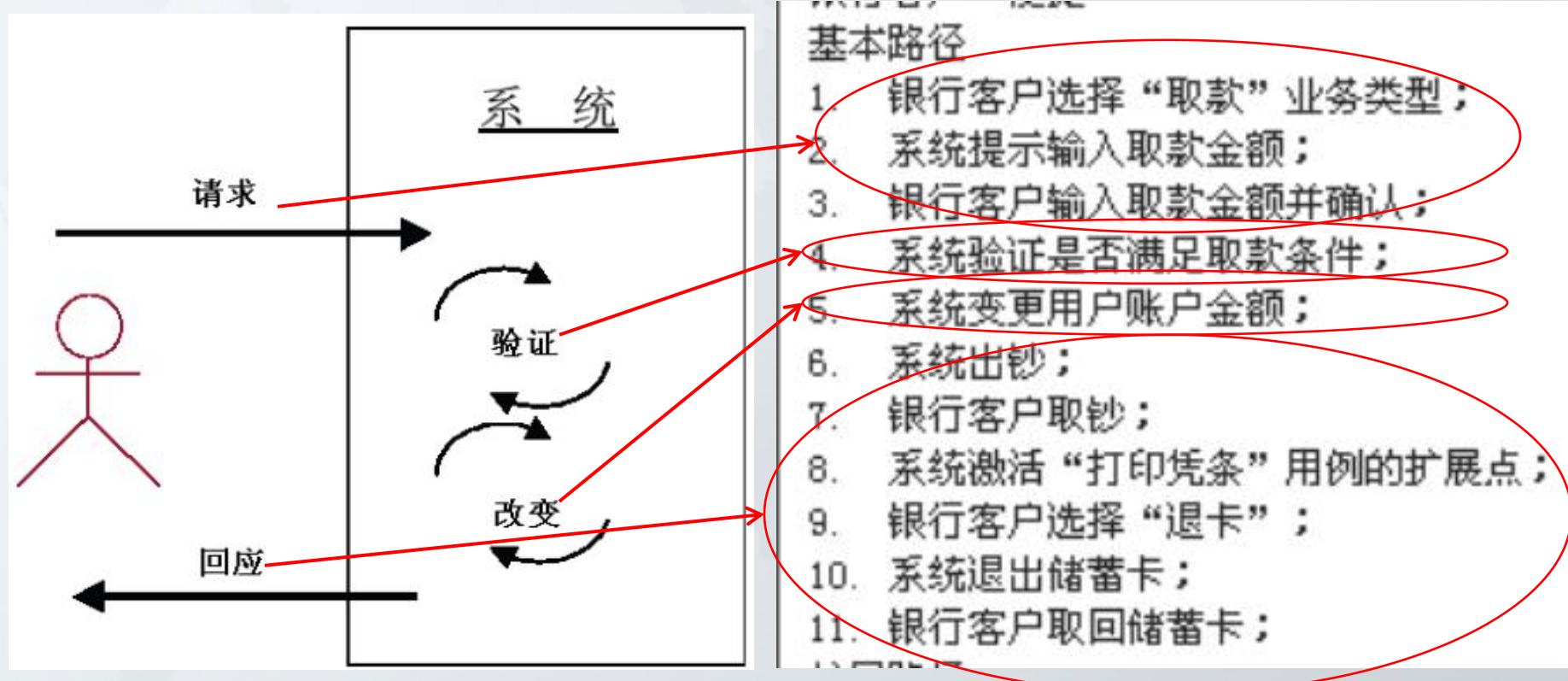
谁的？

基本路径 >>>

- 客户**最想看到的、最关心**的路
径。——核心的核心
- 把基本路径单独分离，凸显用例的
核心价值。
- 与扩展路径相比，更为有序。

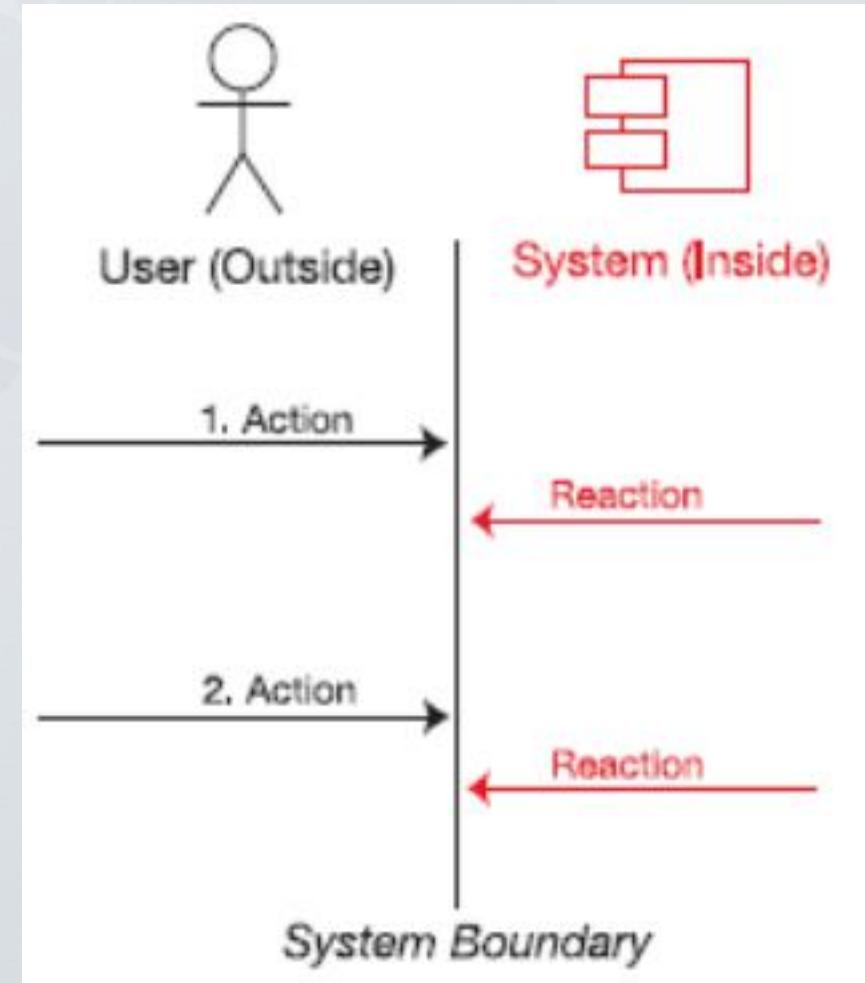


基本路径的典型案例和步骤 >>>



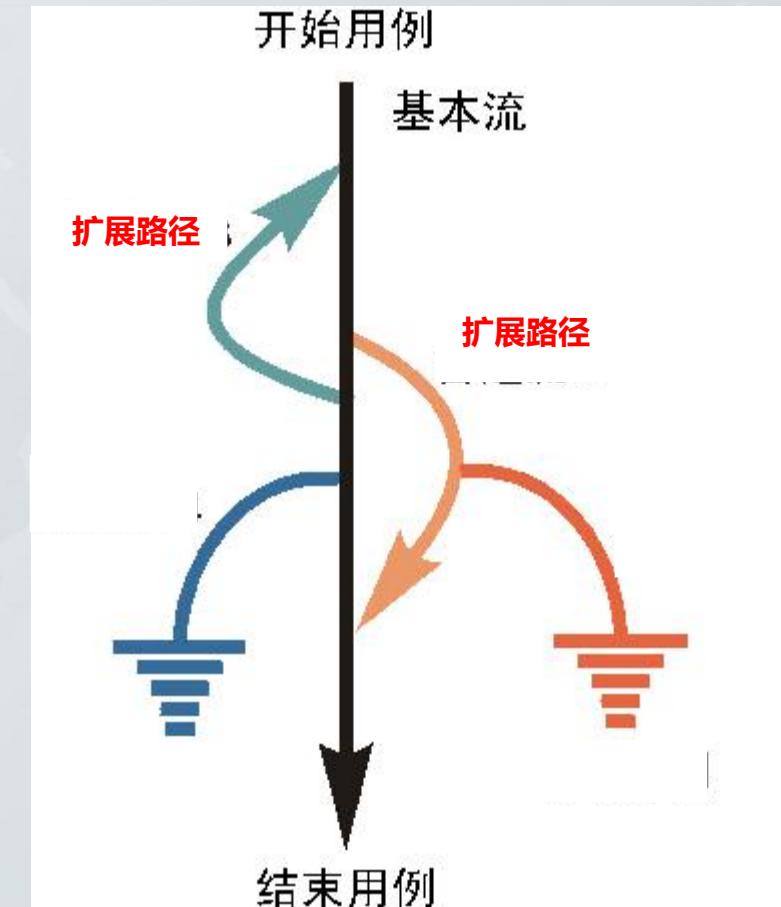
注意：基本路径的书写要求 »»

- 以主动语态、“名词-动词-名词”格式来书写。
- 如：银行客户-输入-密码；
银行自助系统-弹出-储蓄卡
- 主语只能是执行者或系统。



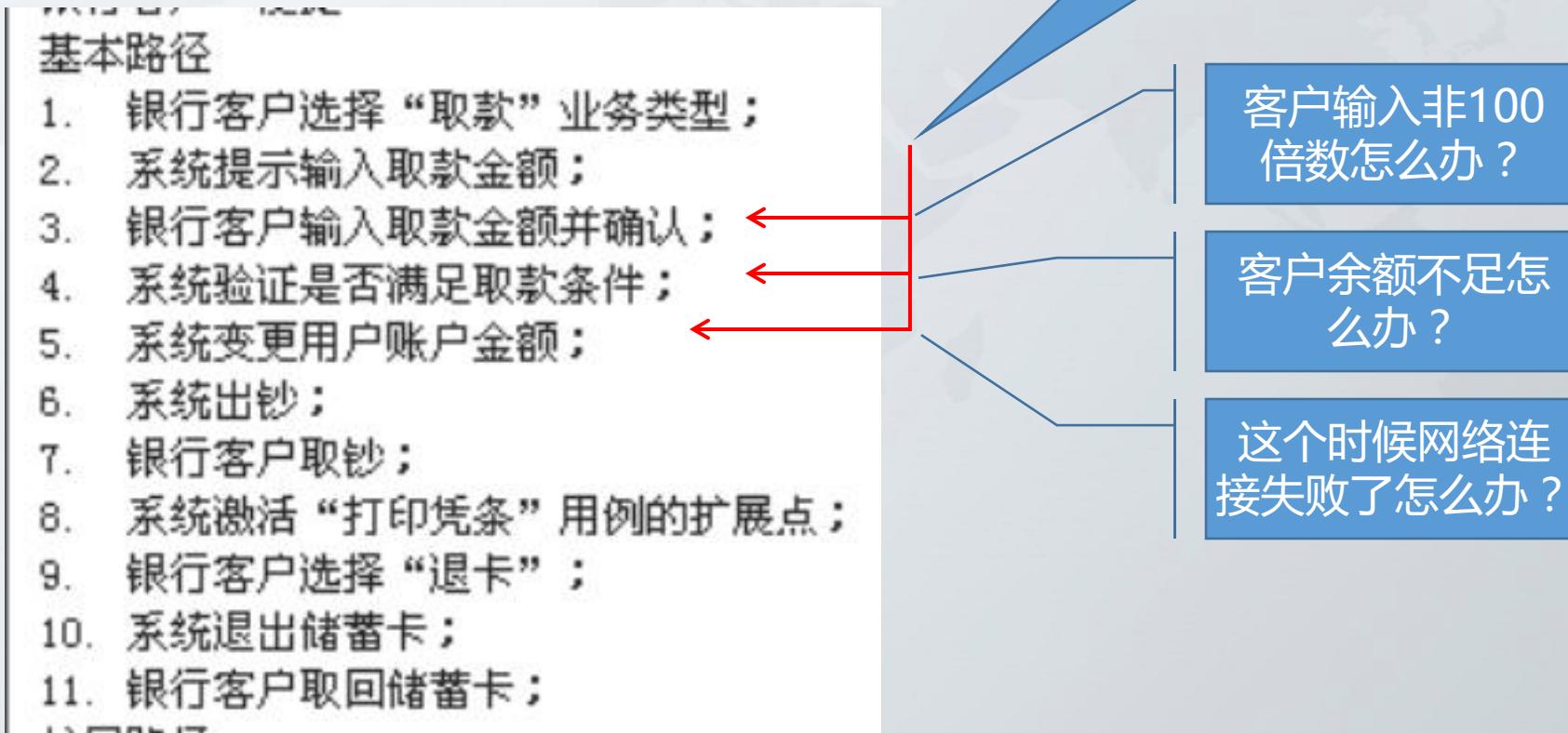
扩展路径 ➞

- 系统要处理的意外和分支。
- 与基本路径相比，更为无序。



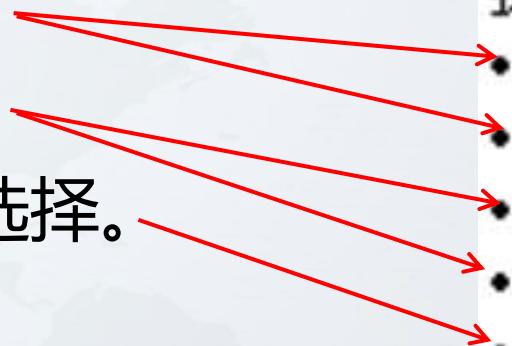
识别扩展路径的方法 ➤➤➤

- 从基本路径的第一句开始，不断地问
“还可能发生别的事情吗？”



常见的扩展点 »»

- 系统验证。
 - 步骤失败。
 - 执行者的选择。
 -
- 扩展路径**

 - 银行客户输入的取款金额不符合业务规则，系统提示；
 - 银行客户的账户金额不足，系统提示；
 - 系统可用货币不足，系统提示；
 - 系统与银行系统网络故障，系统提示；
 - 如果用户选择“打印凭条”，系统进入“打印凭条”用例；
- 

注意：扩展路径描述 >>>

- 每一种扩展尽量用一句（段）话描述；

扩展路径

- 银行客户输入的取款金额不符合业务规则，系统提示；
- 银行客户的账户金额不足，系统提示；
- 系统可用货币不足，系统提示；
- 系统与银行系统网络故障，系统提示；
- 如果用户选择“打印凭条”，系统进入“打印凭条”用例；

DEMO:EA中进行用例描述 >>>

EA Academic

File Edit View Project Diagram Element Tools Add-Ins Settings Window Help

Toolbox Layout Palette More tools... Use Case Actor Use Case Collaboratio Boundary Package Use Case R... Common

Use Case Diagram: '银行客户用例图' created: 2005-11-19 modified: 2011-03-22

UseCase : GetMoney

General | Require | Constraints | Links | Scenario | Files

Name: GetMoney
Stereotype: Abstract
Author: zhaosheng Status: Proposed
Scope: Public Complexity: Easy
Alias: 取款 Language: <none>
Keywords:
Phase: 1.0 Version: 1.0 Advanced
Notes:

干系人利益
银行: 安全、准确、节约运营成本
银行客户: 便捷
基本路径
1. 银行客户选择“取款”业务类型；
2. 系统提示输入取款金额；
3. 银行客户输入取款金额并确认；
4. 系统验证是否满足取款条件；
5. 系统变更用户账户金额；
6. 系统出钞；
7. 银行客户取钞；
8. 系统激活“打印凭条”用例的扩展点；
9. 银行客户选择“退卡”；
10. 系统退出储蓄卡；
11. 银行客户收回储蓄卡；
扩展路径
• 银行客户输入的取款金额不符合业务规则，系统提示：
• 银行客户的账户金额不足，系统提示；
• 系统可用货币不足，系统提示；
• 系统与银行系统网络故障，系统提示；
• 如果用户选择“打印凭条”，系统进入“打印凭条”用例；
业务规则
• 系统只提供100元面额的钞票；
• 系统单次最大取款额为2000元，当日最高总取款额为20000元；
• 30秒内无人取出钞或储蓄卡，系统自动吞回；

project Browser

- Use Case Model
 - 银行客户用例3
 - 银行客户用例图
 - 银行客户用例2
 - 银行职员用例
 - AddMoney
 - GetMoney
 - InputPassword
 - Login
 - Print
 - SaveMoney
 - SearchLog
 - TransferAccounts
- 3-健壮性分析

Project E... Model Views Resources

Notes

干系人利益
银行: 安全、准确、节约运营成本
银行客户: 便捷
基本路径
1. 银行客户选择“取款”业务类型；
2. 系统提示输入取款金额；
3. 银行客户输入取款金额并确认；
4. 系统验证是否满足取款条件；
5. 系统变更用户账户金额；
6. 系统出钞；
7. 银行客户取钞；
8. 系统激活“打印凭条”用例的扩展点；

确定 取消 应用(A) 帮助 Notes Pan & Zoom Properties 0.1KB/S 0.7KB/S PAN

高级话题：不要涉及页面细节 ➞

- ❖ 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
- ❖

高级话题：基本与扩展分开 ➤➤➤

基本路径

1. 银行客户选择“取款”业务类型；
2. 系统提示输入取款金额；
3. 银行客户输入取款金额并确认；
4. 系统验证是否满足取款条件；
5. 系统变更用户账户金额；
6. 系统出钞；
7. 银行客户取钞；
8. 系统进入“打印凭条”用例；
9. 银行客户选择“退卡”；
10. 系统退出储蓄卡；
11. 银行客户收回储蓄卡；

扩展路径

- 银行客户输入的取款金额不符合业务规则，系统提示；
- 银行客户的账户金额不足，系统提示；
- 系统可用货币不足，系统提示；
- 系统与银行系统网络故障，系统提示；
- 如果用户选择“打印凭条”，系统进入“打印凭条”用例；



基本路径

1. 银行客户选择“取款”业务类型；
2. 系统提示输入取款金额；
3. 银行客户输入取款金额并确认；
4. 系统验证是否满足取款条件；
5. 系统变更用户账户金额；
6. 系统出钞；
7. 银行客户取钞；
8. 系统激活“打印凭条”用例的扩展点；
9. 银行客户选择“退卡”；
10. 系统退出储蓄卡；
11. 银行客户收回储蓄卡；

扩展路径

- 银行客户输入的取款金额不符合业务规则，系统提示；
- 银行客户的账户金额不足，系统提示；
- 系统可用货币不足，系统提示；
- 系统与银行系统网络故障，系统提示；
- 如果用户选择“打印凭条”，系统进入“打印凭条”用例；

高级话题：不要假想系统不能负责的事情 ➤

3e. 收银

- 3e1. 系统收集客层分析数据。
- 3e2. 系统显示收银界面。
- 3e3. 系统计算并显示应收总金额。
- 3e4. ~~顾客付款。~~
- 3e5. 用户录入付款方式和金额。
- 3e6. 系统计算并找零。
- 3e7. 系统打开钱箱并打印小票。
- 3e8. ~~用户找零~~并关闭钱箱。

系统请求××做某事 (不用写××系统做某事)

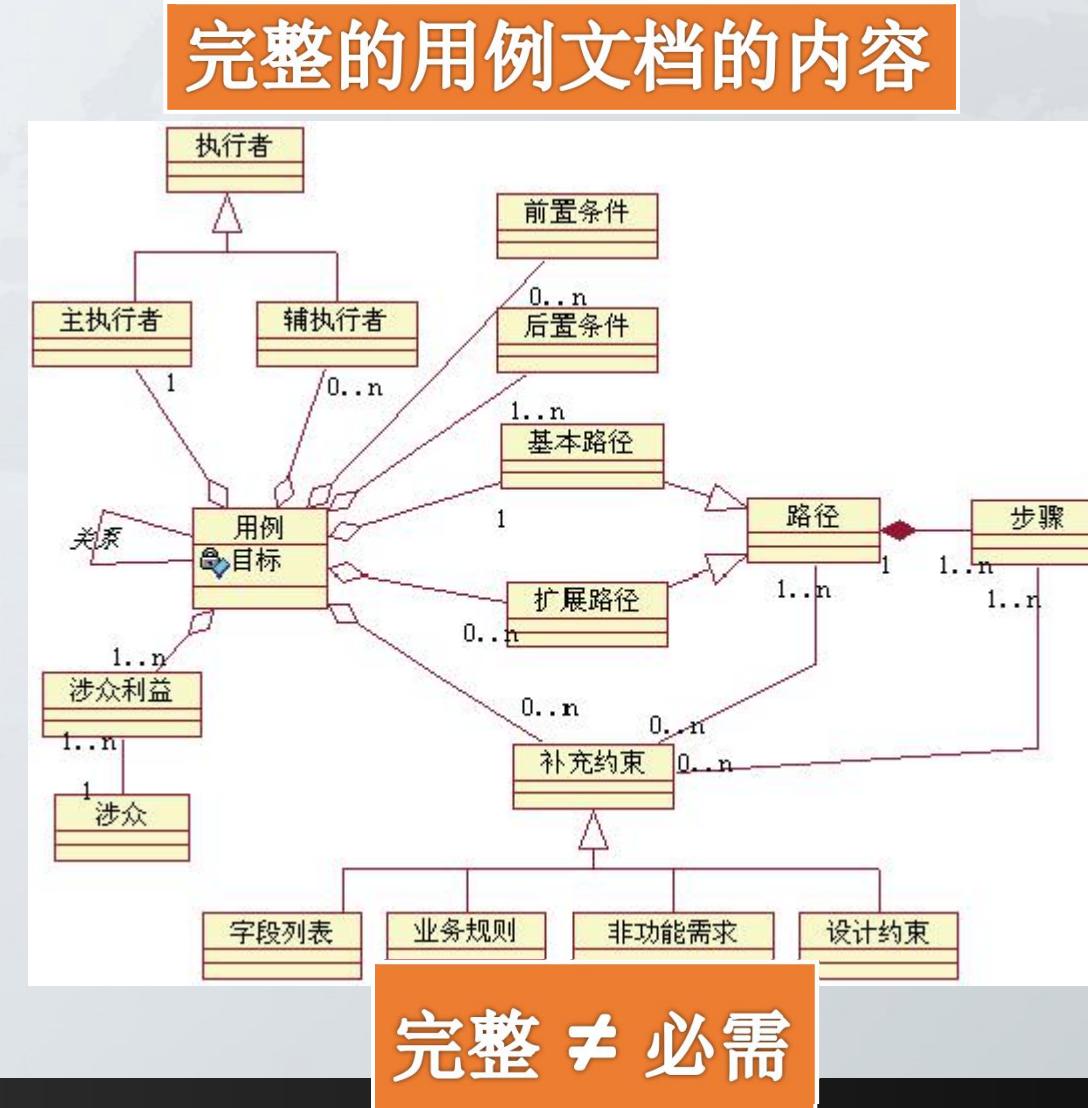
3、→痕检员编辑提交指纹特征信息、案件描述信息。

4、→痕检员提交查找任务。

5、→系统将案件信息发送给请求 AFIS 系统进行查找比对。

高级话题：完整的用例文档 ➞

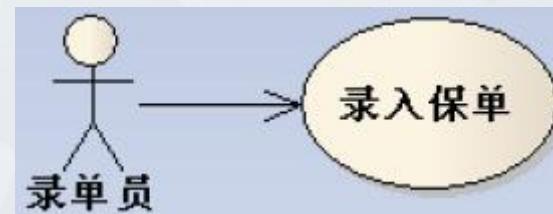
- 用例编号：用例名
- 执行者
- 前置条件
- 后置条件
- 干系人利益
- 基本路径
 - 1.....
 - 2.....
- 扩展路径
 - 2a.....
 - 2a1.....
- 字段列表
- 业务规则
- 设计约束



高级话题 >>>

- **前置条件**：开始用例前系统及其环境的状态。

- 形式：必须是系统**能检测到的**
- 内容：不涉及到**涉众的利益**



业务代表已把保单交给录单员 X

录单员已经登录 ✓



前置条件

ATM用户的账户里有足够的金额 X

高级话题 >>>

- **后置条件**：用例成功结束后**系统**应该具备的**状态**。

后置条件。

- 系统记录痕检员的鉴定结果。



后置条件。

- 系统**已**记录痕检员的鉴定结果。

高级话题 >>>

- **字段列表**：可以用**自然语言**，也可以用**表达式**

- **+ : 数据序列**
- **[] : 可选项**
- **{ }* : 多个**
- **{|||} : 可能取值**
- **A=B : 把B的结构赋给A**

- ❖ **注册信息=公司名+联系人+电话+{联系地址}***
- ❖ **联系地址=州+城市+街道+邮编**
- ❖ **保存信息=注册信息+注册时间**
- ❖ **客房状态={空闲|已预定|占用|维修中}**

高级话题 >>>

- 字段列表：
 - 细致程度取决于干系人共识
 - 不同于数据模型，只是数据模型的一部分
 - 不等于数据字典，不要过早的把时间花在细节上

高级话题 >>>

- 设计约束：
- 界面样式
- 报表
- 平台
- 语言
- 外系统接口
-



必须来自干系人！

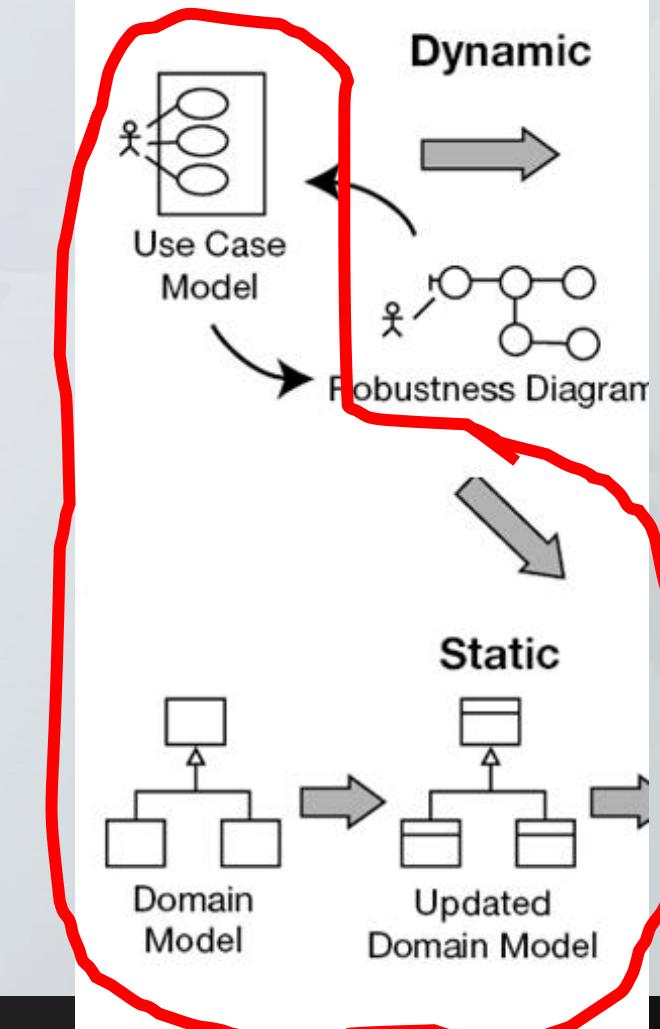
系统用例建模步骤 >>>

1. 绘制系统用例图
2. 编写系统用例描述
3. **更新域模型**

更新域模型 »»

- 在用例分析结果的基础上，对早期的域模型进一步完善和调整。
 - 以所有的用例描述为基础，采用首次域建模类似的方法，完善和调整已有的域模型；

需求分析 健壮性分析



回顾：域建模的步骤 >>>

Step 1
仔细阅读需求文档，提取出名词和名词短语

Step 2
排除列表中重复、相似的术语

Step 3
排除超出系统范围的术语

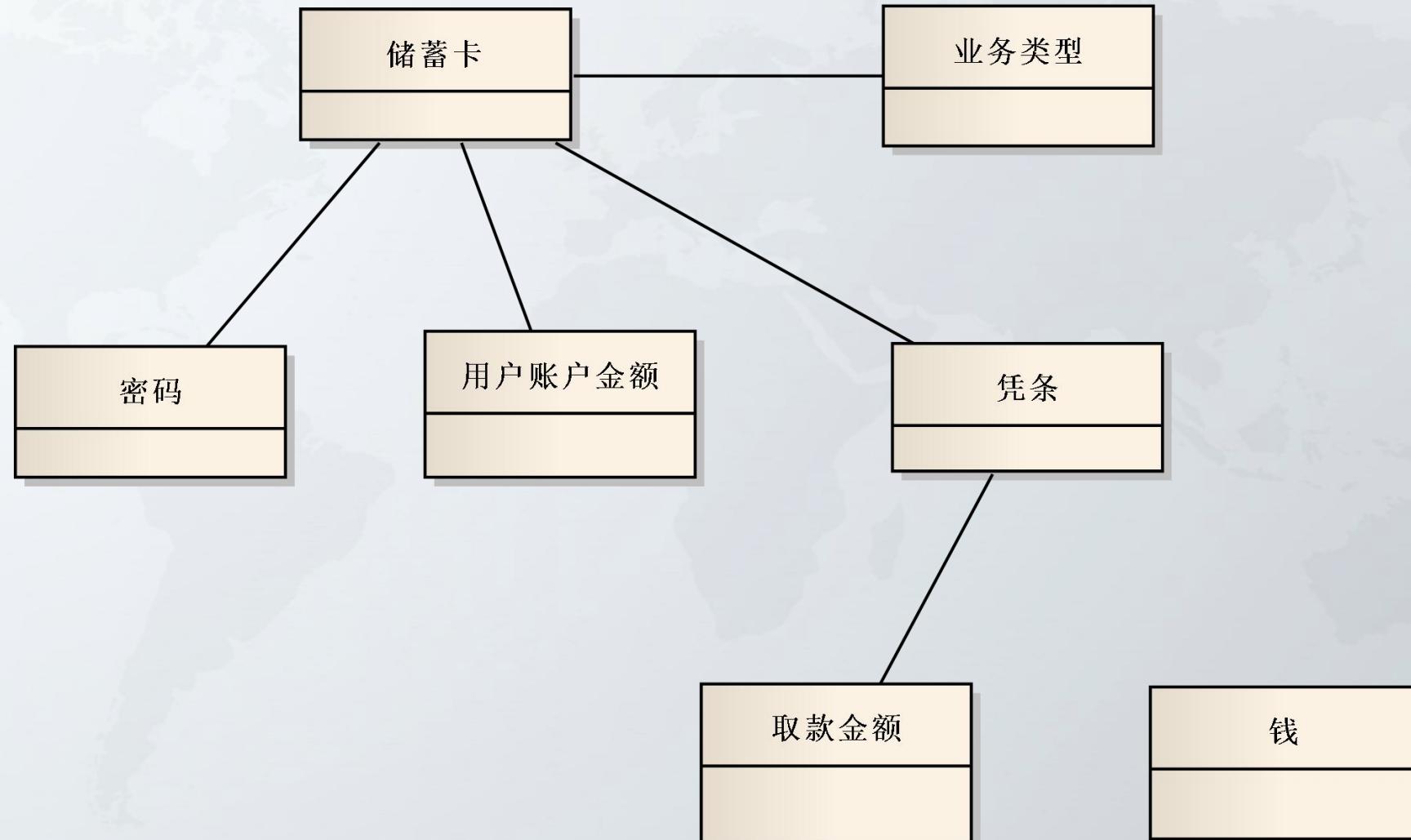
确定域模型之间的关系：泛化
[Generalization]和关联[Association]

Step 5
整理第一版域模型

Step 4
画出第一版域模型图



回顾：第一版域模型 >>>



第一步：基于用例描述更新域模型 ➤



干系人利益

银行客户：操作尽量简单；

银行：足够安全，保护银行利益和客户利益；

基本路径

1. 银行客户插入储蓄卡；
2. 系统校验储蓄卡有效性；
3. 系统扫描储蓄卡号码；
4. 系统提示输入密码；
5. 银行客户输入密码；
6. 系统请求银行系统校验储蓄卡号码和密码；
7. 系统反馈校验成功。

扩展路径

- 储蓄卡扫描失败，系统提示银行客户；
- 储蓄卡无效，系统提示银行客户；
- 储蓄卡号码和密码校验失败，系统提示银行客户；

业务规则

1. 仅能识别本行的储蓄卡（储蓄卡有效性）；
2. 密码长度为6位数字；
3. 3次以上密码输入错误，系统将吞掉储蓄卡；



干系人利益

银行：安全、准确、节约运营成本

银行客户：便捷

基本路径

1. 银行客户选择“取款”业务类型；
2. 系统提示输入取款金额；
3. 银行客户输入取款金额并确认；
4. 系统验证是否满足取款条件；
5. 系统变更用户账户的储蓄金额；
6. 系统出钱；
7. 银行客户取钱；
8. 系统激活“打印凭条”用例的扩展点；
9. 银行客户选择“退卡”；
10. 系统退出储蓄卡；
11. 银行客户收回储蓄卡；

扩展路径

- 银行客户输入的取款金额不符合取款条件，系统提示；
- 银行客户的储蓄金额不足，系统提示；
- 系统可用钱不足，系统提示；
- 如果用户选择“打印凭条”，系统进入“打印凭条”用例；

业务规则

- 系统只提供100元面额的钞票；
- 系统单次最大取款额为2000元，当日最高总取款额为20000；
- 30秒内无人取出的钞票或储蓄卡，系统自动收回；

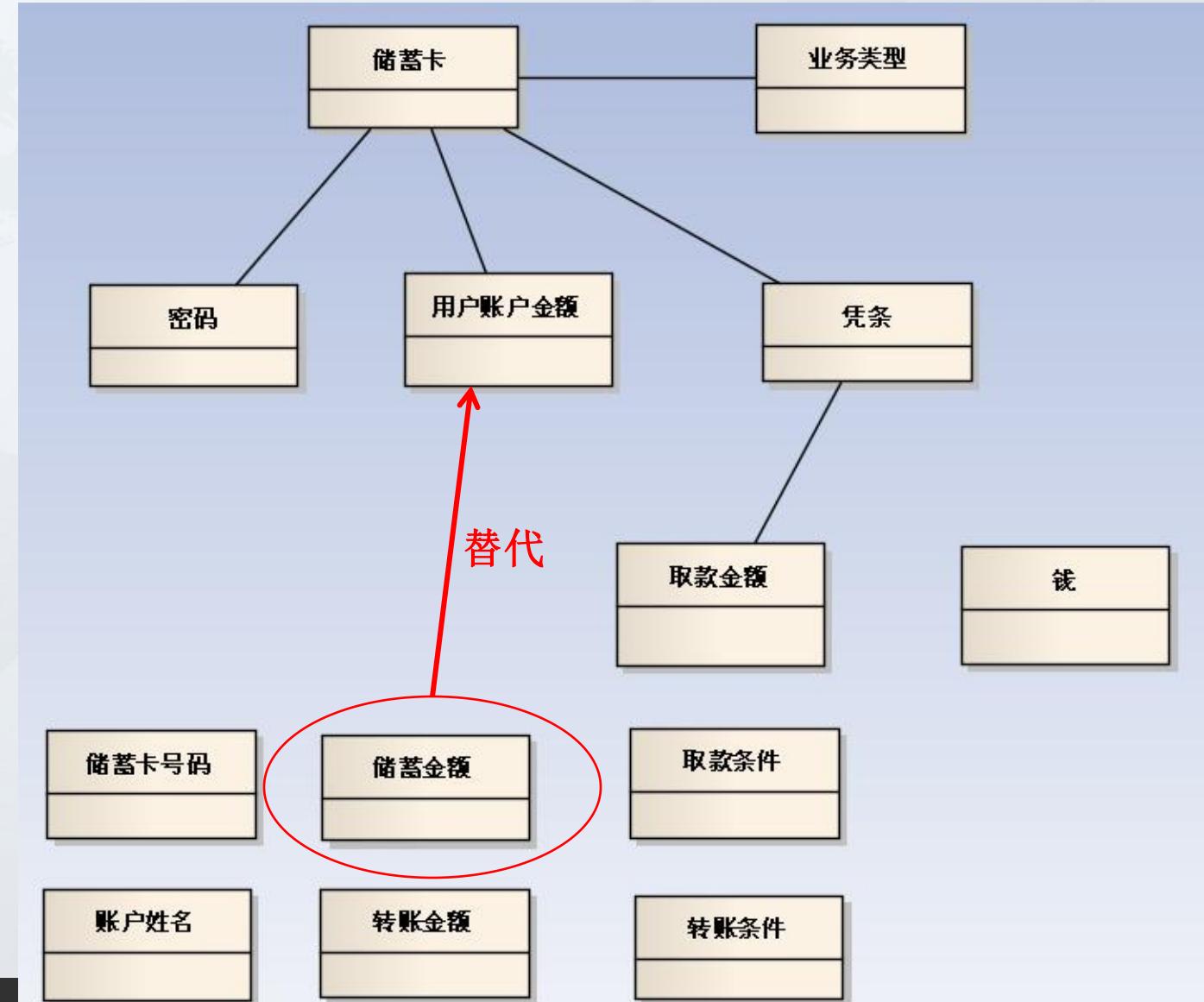
整理新发现的名词（排重）»»

排除前	重复、相似	排除后
储蓄卡号码 取款条件 用户账户 储蓄金额 存钱槽 转入账户 账户姓名 转账金额 转出账户 转账条件	储蓄卡号码 用户账户 转入账户 转入账户	储蓄卡号码
		取款条件 储蓄金额 存钱槽 账户姓名 转账金额 转账条件

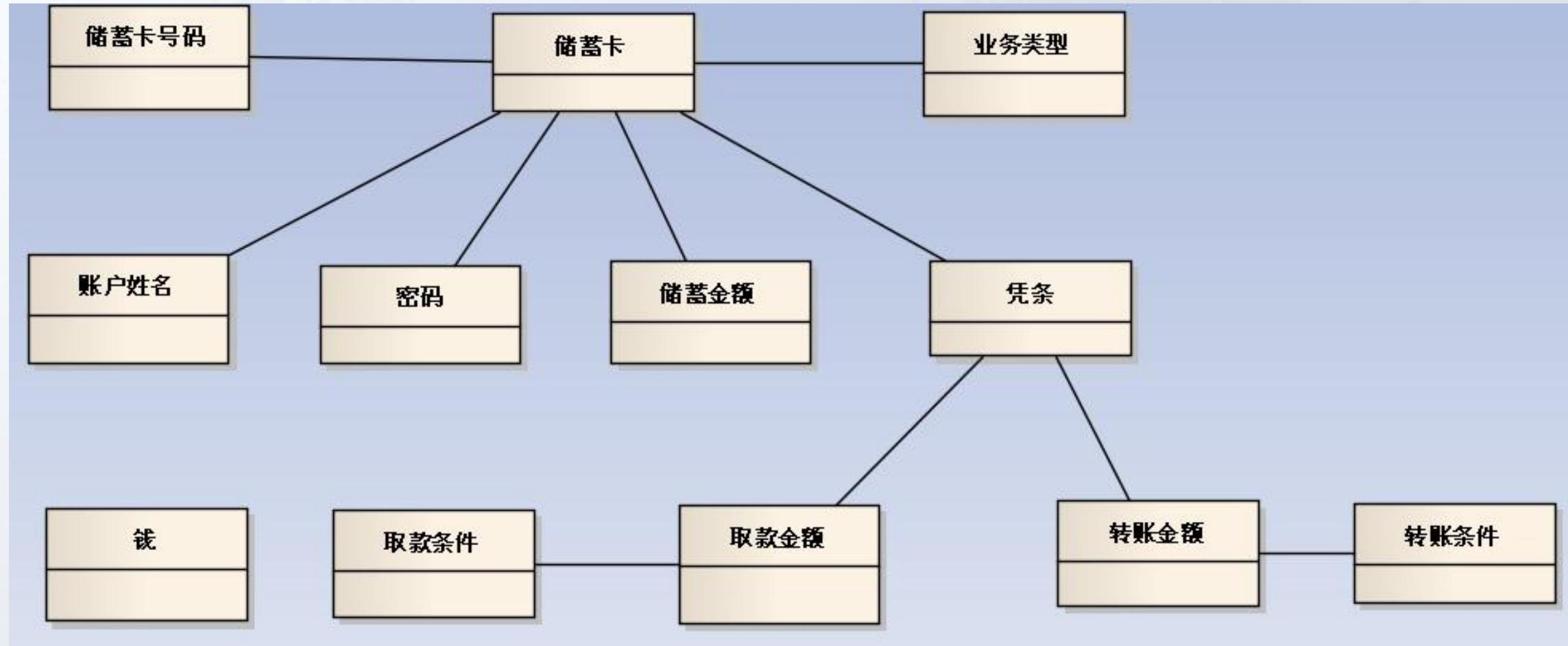
整理新发现的名词（排除系统外）»»

排除前	系统外	排除后
储蓄卡号码		储蓄卡号码
取款条件		取款条件
储蓄金额		储蓄金额
存钱槽	系统外，硬件设备的一部分	
账户姓名		账户姓名
转账金额		转账金额
转账条件		转账条件

更新域模型 >>>



整理域模型 »»



目录 >>>

一

需求分析的几种主流方法

二

域建模：以OO思想构建术语表

三

用例分析：系统功能性需求分析的好工具

四

非功能性需求分析及需求定义与评审



功能性需求 & 非功能性需求 »»

- 功能性需求，通俗讲就是系统可以做什么。
- 非功能性需求，通俗讲就是系统可以把某项功能做到什么程度。

人无我有，人有我优

功能性需求

非功能性需求

思考：下面哪些是非功能性需求？»»



1. 最高时速200公里；
2. ABS防抱死；
3. 安全气囊；
4. 双开门；
5. 百公里耗油8-10升；



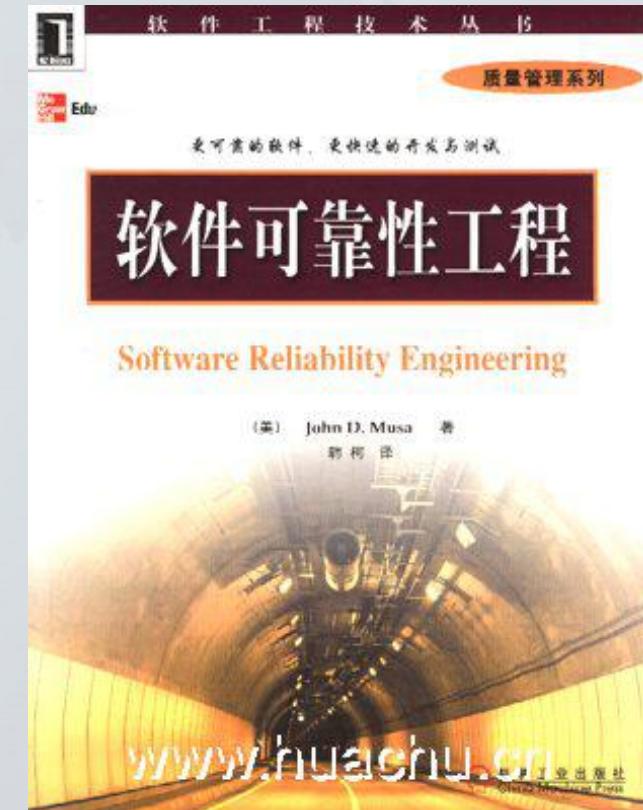
1. 7*24小时不间断服务；
2. 可存款、取款、转账；
3. 可自动识别存款中的伪钞；
4. 单次最高取款额3000元；
5. 可跨行取款；

软件产品的典型非功能性需求(RUPS) ➤

- 可靠性[Reliability]。
- 可用性[Usability]。
- 性能[Performance]。
- 可支持性[Supportability]。

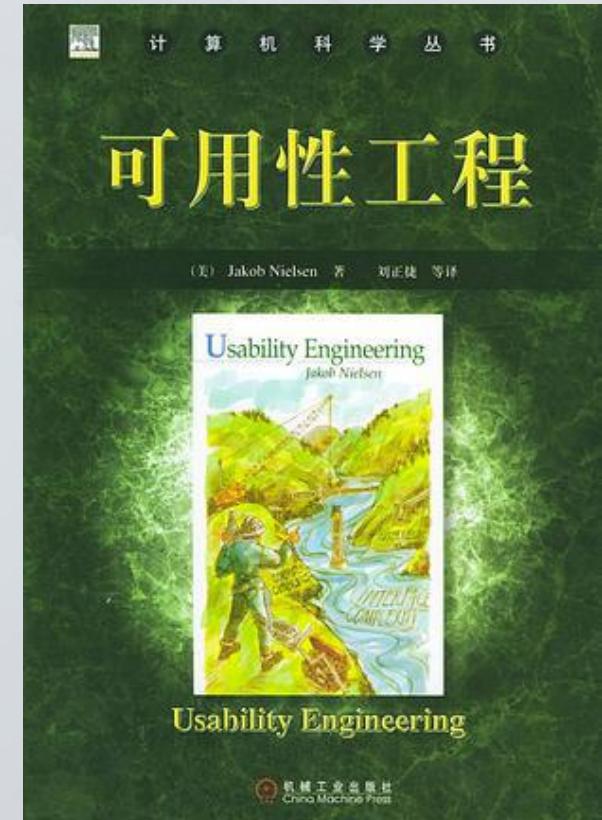
可靠性[Reliability] >>

- 系统在一定时间内、在一定条件下无故障地执行指定功能的能力。
- 自助银行系统的可靠性指标示例：
 - 7*24小时不间断服务；
 - 系统采用冗余机制保障数据安全、完整；



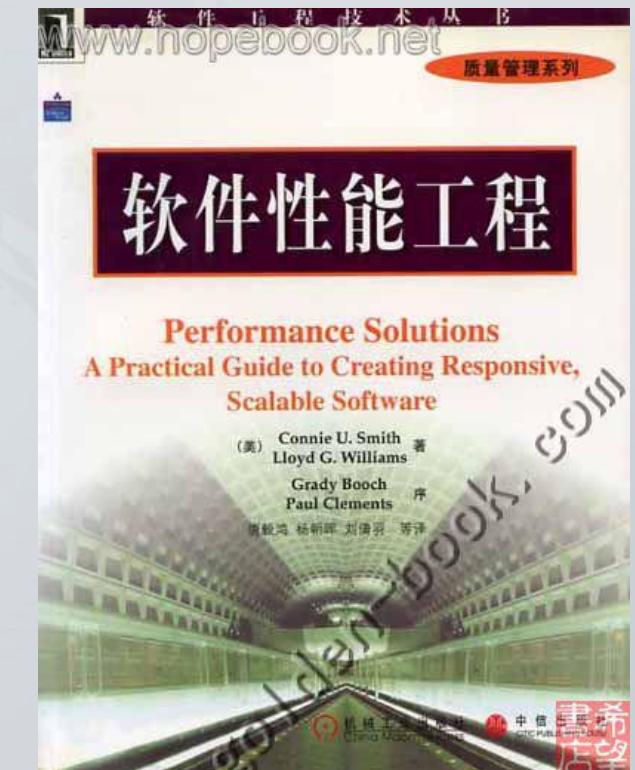
可用性[Usability] >>>

- 一个产品可以被特定的用户在特定的上下文中，有效、高效并且满意得达成特定目标的程度。
- 自助银行系统的可用性指标示例：
 - 用户完成任何一项业务，操作步骤不超过5步；
 - 任何用户操作失误或系统错误都有友好的提示信息；



性能[Performance] >>>

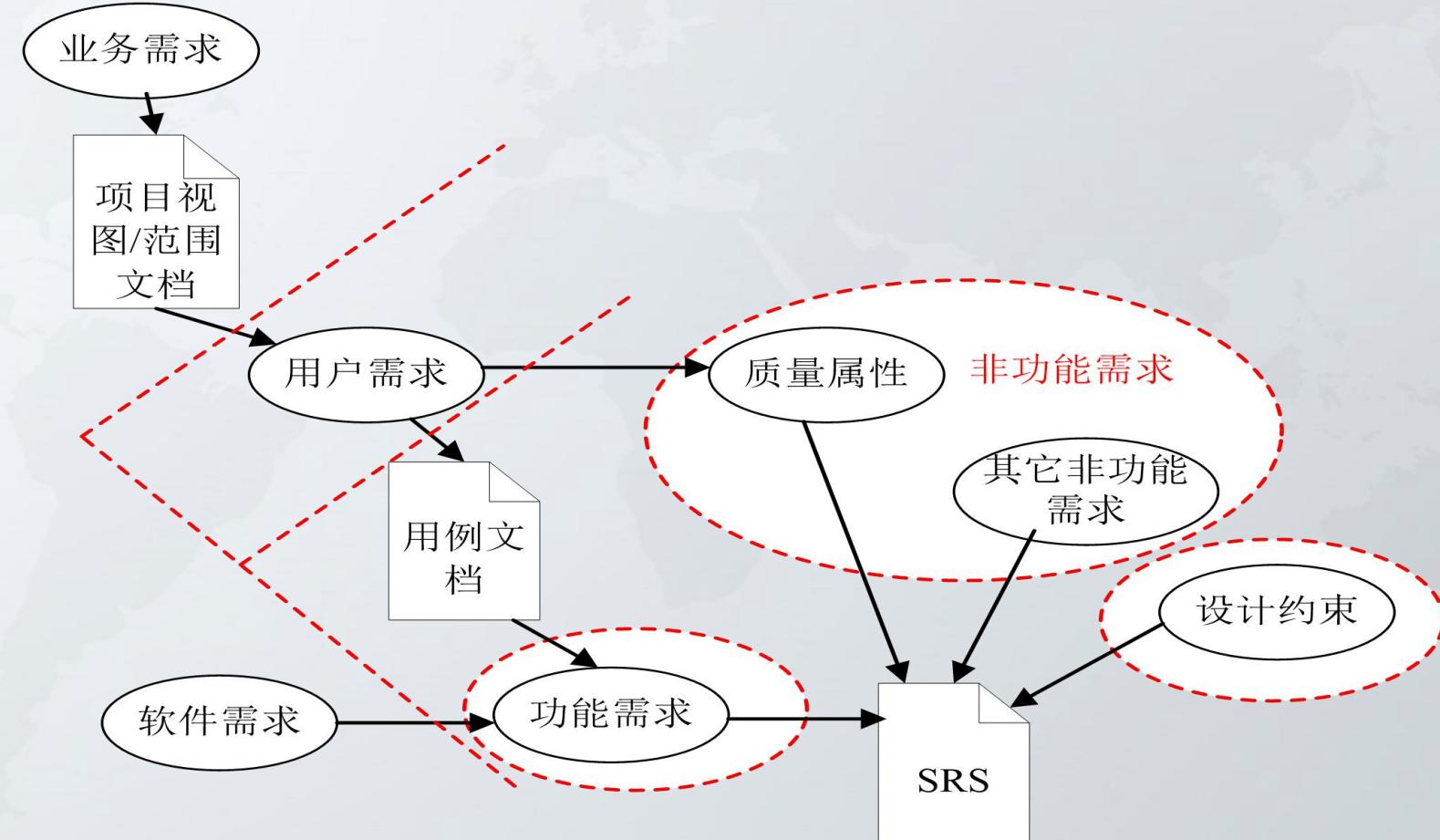
- 系统实现预期功能的能力的特性。
- 自助银行系统的性能指标示例：
 - 身份验证时间<1秒；
 - 正常操作下，系统最大支持的业务量为1000笔/日；



可支持性[Supportability] ➤

- 系统在故障解决和系统升级方面的能力。
- 自助银行系统的可支持性指标示例：
 - 95%的系统故障可在两小时内解决；
 - 系统的软件部分每年升级一次，并保持向下兼容；

需求定义



需求描述风格

- 自然语言为主，图形语言为辅
- 图形语言为主，自然语言为辅
 - 团队整体的模型标准能力较高

需求描述模板

- 标准模板
 - 国标 : GB8567-2006
 - RUP
 - 咨询商 : Volere
- 自定义模板
 - 团队整体的模型标准能力较高

业务需求写作总则

目录

第一部分·业务目标	4+
1 业务需求基本情况	4+
2 业务背景描述	4+
3 业务目标	4+
第二部分·业务需求	4+
1 业务背景分析	4+
2 业务目标	4+
3 业务流程	4+
3.1 业务相关组织结构、人员组织结构	5+
3.2 业务流程列表	5+
3.3 业务流程一描述	5+
3.4 业务流程 N 描述	6+
3.5 报表	6+
4 需求点描述	6+
5 相关业务制度、管理办法、实务	6+
7 业务术语表	6+
8 干系人列表	7+

需求描述两大原则

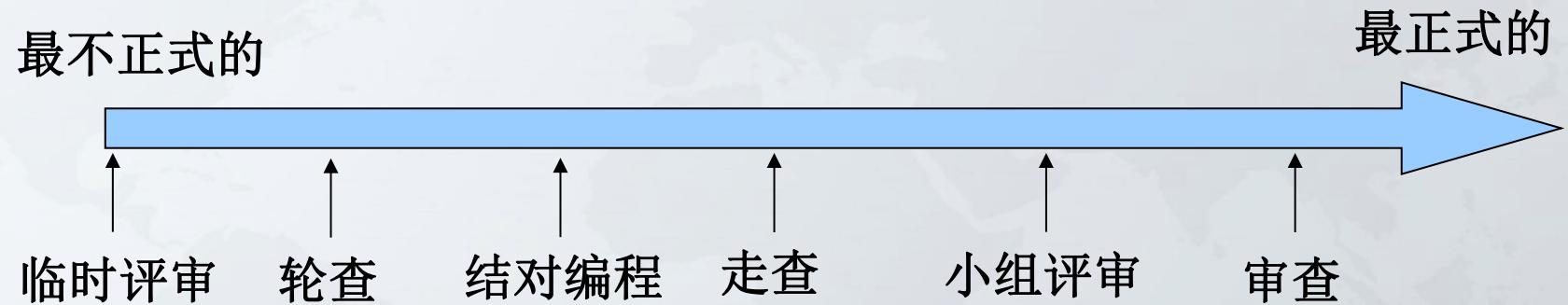
- 简洁、段落文字少：尽可能不要采用长篇大论的方法。段落越长，读者压力越大，产生歧义的可能性越大。
- 列表、图表相结合的表示法
- 示例

应急抢修：这是指对航标故障的突发性维护工作。应急抢修的发起点包括：遥测遥控系统报警、外部人员报修，系统在这些发起点生成“应急抢修单”；应急抢修的物资管理原则是：按比例建设备品，根据应急抢修单申请备品，执行完成后填写相应的执行情况（更新备品信息、更新航标档案），最后配合固定资产及物资管理系统补课备品库。

应急抢修：

- 定义：对航标故障的突发性维护工作
- 发起点：遥测遥控系统报警、外部人员报修，系统在这些发起点生成“应急抢修单”
- 物资管理原则：
 - 按比例建设备品；
 - 根据应急抢修单申请备品；
 - 执行完成后填写相应的执行情况（更新备品信息、更新航标档案），最后配合固定资产及物资管理系统补课备品库。

需求评审



Random review, Pass-round, Walkthrough, Team review, Inspection

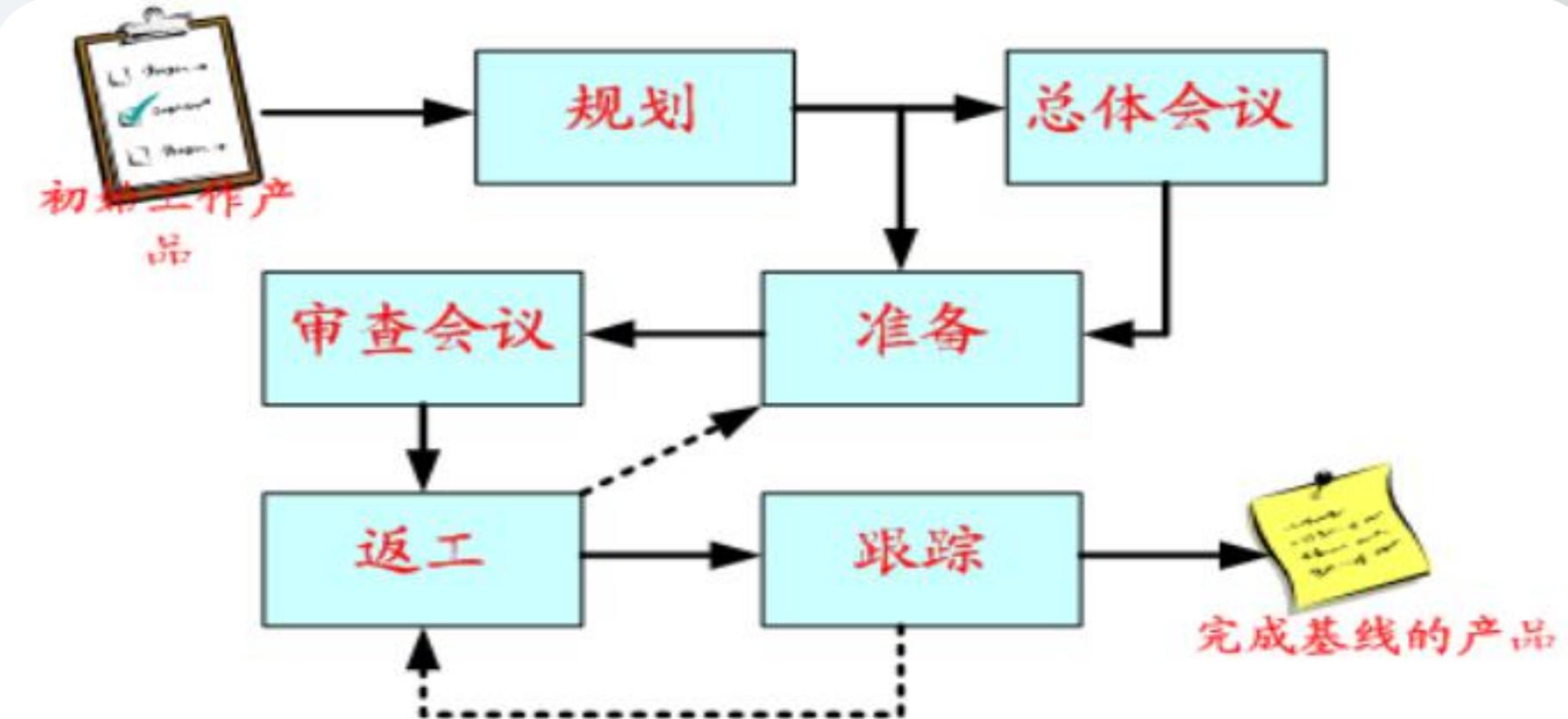
三种相对正式的评审

	审查	小组评审	走查
原文	Inspection	Team review	Walkthrough
组织者	企业级 常由EPG、QA部门组织	团队级、项目级 通常由项目经理发起	团队级、项目级 通常由项目内发起
主持人	专职主持人，不能是作者	可以是专职或作者	通常是作者
会前会	一定召开评审前会议	会前会通常很简单	通常没有会前会
检查表	一定有规范的检查表	相对简单的检查表	通常没有检查表

三种相对不正式的评审

- 结对编程（ /需求分析/设计/测试 ）。
- 轮查：交叉交换文档产物，互相提出意见。
- 临时评审：在日常沟通中做回顾确认。例如和用户沟通时，对沟通内容做总结，请用户确认理解是否一致。

需求审查：主要阶段



审查过程概述

1. 规划：谁参加？评审什么？

主要参与人	列席参与人	评审内容
需求规格说明书的作者	用户代表	《需求规格说明书》 参考材料： 《招标书》《投标书》 《合同》《需求调研报告》
评审专家	接口负责人	
甲方领导	开发代表	
[监理]	测试代表	
主持人		
记录员		

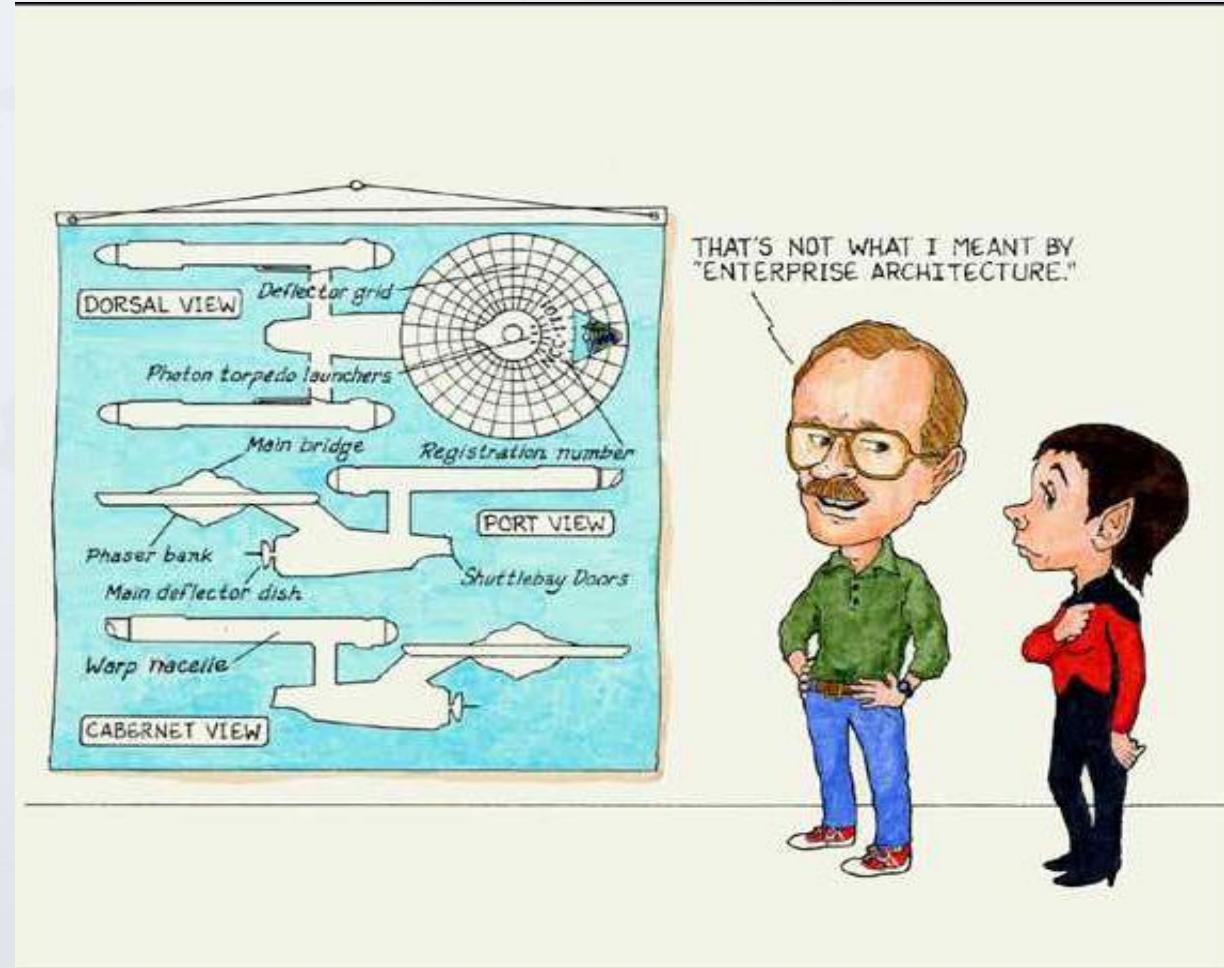
2. 总体会议（会前会）：召集参加评审会的所有成员开一个简短的会议，讨论、明确要评审的内容、评审的要点、评审时所需的资料、缺陷检查表
3. 准备：评审人员提前阅读和准备，才能提出有价值的建议

审查过程概述

4. 审查会议：暴露问题、讨论问题，待审查文档应该符合：
 1. 遵循标准模板，统一模板易于阅读和评审；
 2. 已进行了拼写检查，减少文字错误带来的问题
 3. 已检查了排版错误
 4. 所有未解决问题提前标记好，避免大家浪费精力于还存在问题的地方
5. 返工：没有返工的评审必然沦为“形式主义”。评审中发现的错误必须得到重视和回应。
6. 跟踪
 1. 解决问题：对评审提出的问题进行解决
 2. 避免问题再次出现：对问题进行分类、因果分析，找到问题的深层次原因

需求规格说明书的检查要点

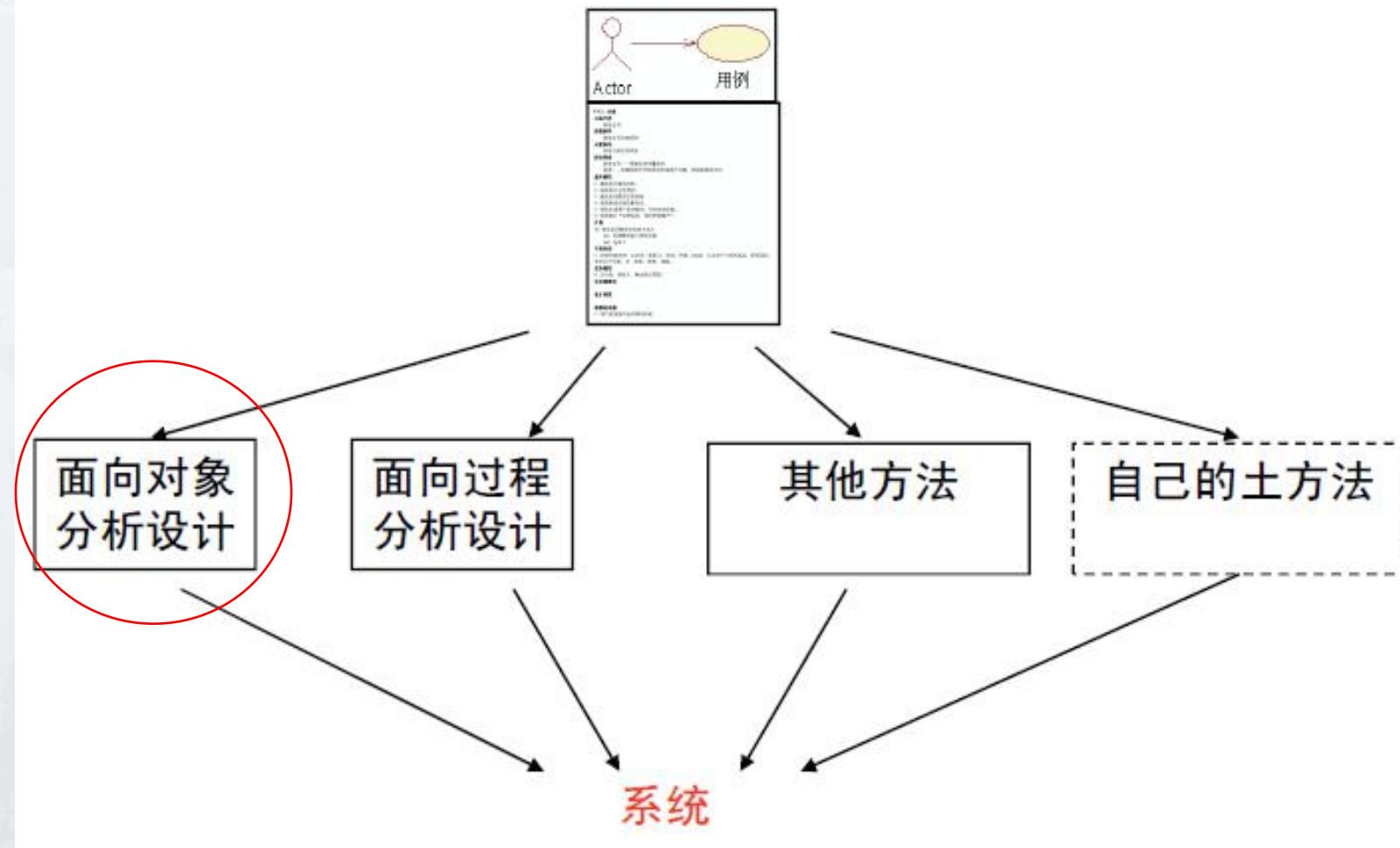
- 正确性；
- 必要性；
- 优先级；
- 明确性；
- 可测性；
- 完整性；
- 一致性；
- 可修改性；



需求测试的checklist文件(检查表示例)

序号	内容	满足/正确	问题描述
1	是否描述系统的所有输入，包括输入源、准确性、取值范围和出现频率？		
2	是否描述系统的所有输出，包括输出的目标、准确性、取值范围、出现频率和格式？		
3	是否描述所有(主要)的报表格式？		
4	是否描述所有硬件和软件的外部接口？		
5	是否描述所有的通信接口，包括握手协议、差错检测和通信协议？		
6	从用户的角度来看，是否描述了对所有必要操作的预计响应时间？		
7	是否对时间方面问题进行考虑，如处理时间、数据传输和系统的吞吐量？		
8	是否描述用户想要完成的所有(主要)任务？		
9	是否每个任务都描述了所使用的数据及产生的数据？		
10	是否描述了安全级别？		
11	是否描述了系统的可靠性，包括软件产生故障的后果、故障后重要数据的保护、错误检测和恢复？		

思考：接下来该怎么办？»»



小结 >>

- 需求分析就是确定新系统的目的、范围、定义和功能；
- 域建模用来生成统一的关键术语表；
- 用例建模用来定义系统的功能性需求；
- 系统的非功能性需求通常包括RUPS；
- 需求评审确认需求分析结果，然后才能进入设计阶段。



THANKS