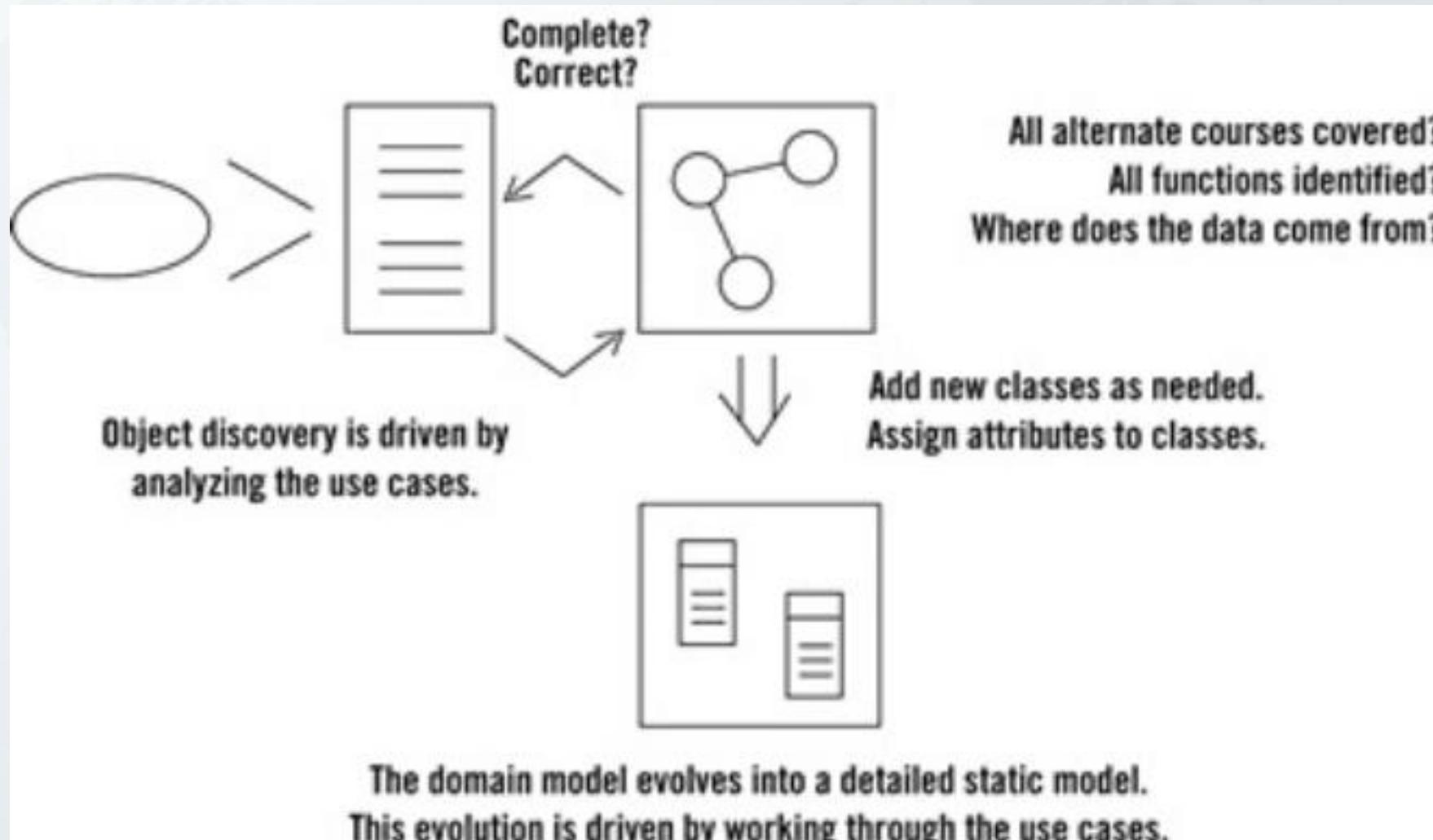




## 第六章 关键设计

# 健壮性分析作用示意图 >>>

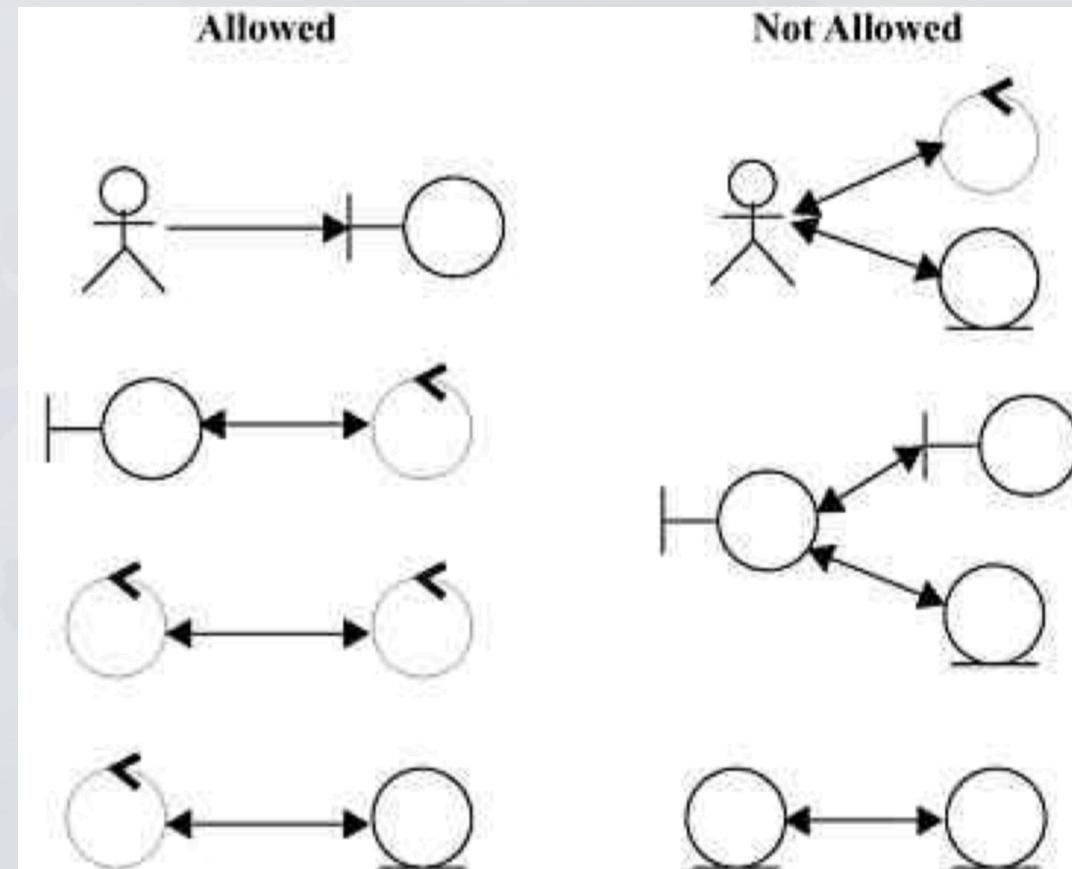


# 健壮性分析中的基本概念 ➤

- 健壮性分析中的三种元素：
  - **边界类[Boundary objects]**与用户交互的对象，系统和外部世界的界面，如窗口，对话框等等。
  - **实体类[Entity objects]**是现实世界存在的实体对象，域模型中的类，它常对应于数据库表和文件。有些实体对象是“临时”对象（如搜索结果），当用例结束后将消失。
  - **控制器类[Controller objects]**边界和实体间的“粘合剂”，将边界对象和实体对象关联起来，它包含了大部分应用逻辑，它们在用户和对象之间架起一座桥梁。控制对象中包含经常修改的业务规则和策略。

# 健壮性分析中的基本概念 »»

- 健壮性分析中三种元素的交互规则：
  - 执行者只可以和**边界**对象通话；
  - **边界**对象和**控制器**可以互相通话  
(**名词** <-> **动词**)；
  - **控制器**可以和另一个**控制器**通话  
(**动词** <-> **动词**)；
  - **控制器**和**实体**对象可以互相通话  
(**动词** <-> **名词**)；

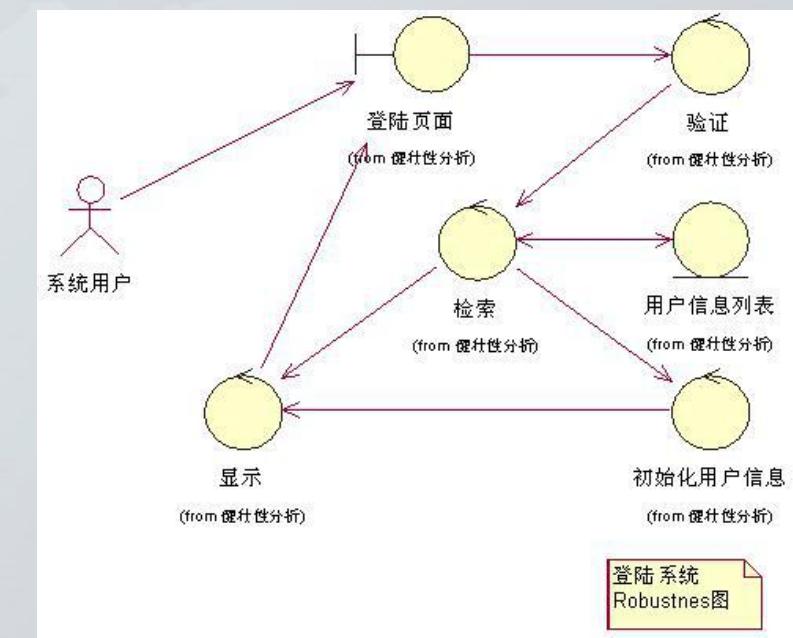
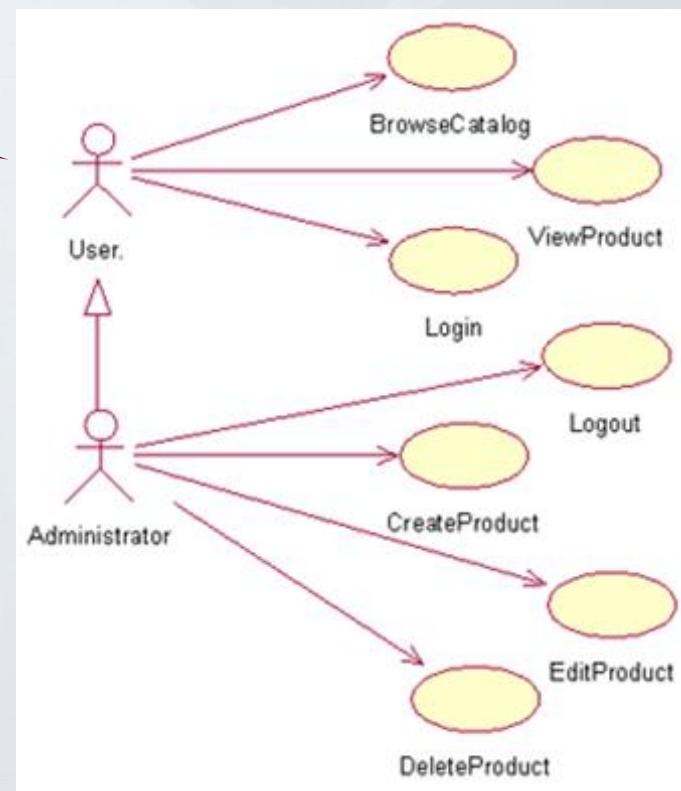
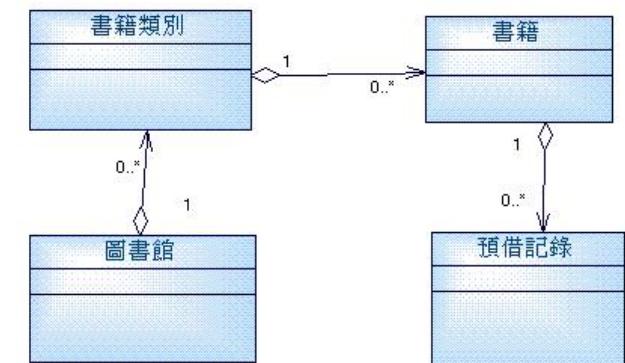


# 健壮性分析的步骤 >>>



- 第一步：创建一个空的健壮性图。
- 第二步：直接将用例文本粘贴到图上（基本路径和扩展路径）。
- 第三步：从基本路径的第一句话开始画健壮性图。
- 第四步：贯穿整个用例基本路径，一次一个句子，画执行者、适当的边界对象和实体对象以及控制器，和各元素之间的连线。
- 第五步：将每一个扩展路径画在健壮性图上，并以红色标示出。

# 域模型的迭代 >>>



登陆系统  
Robustnes图

# 目录 >>

一

关键设计的意义和方法

二

关键设计的步骤

三

关键设计的复核



# 目录 >>>

一

关键设计的意义和方法

二

关键设计的步骤

三

关键设计的复核



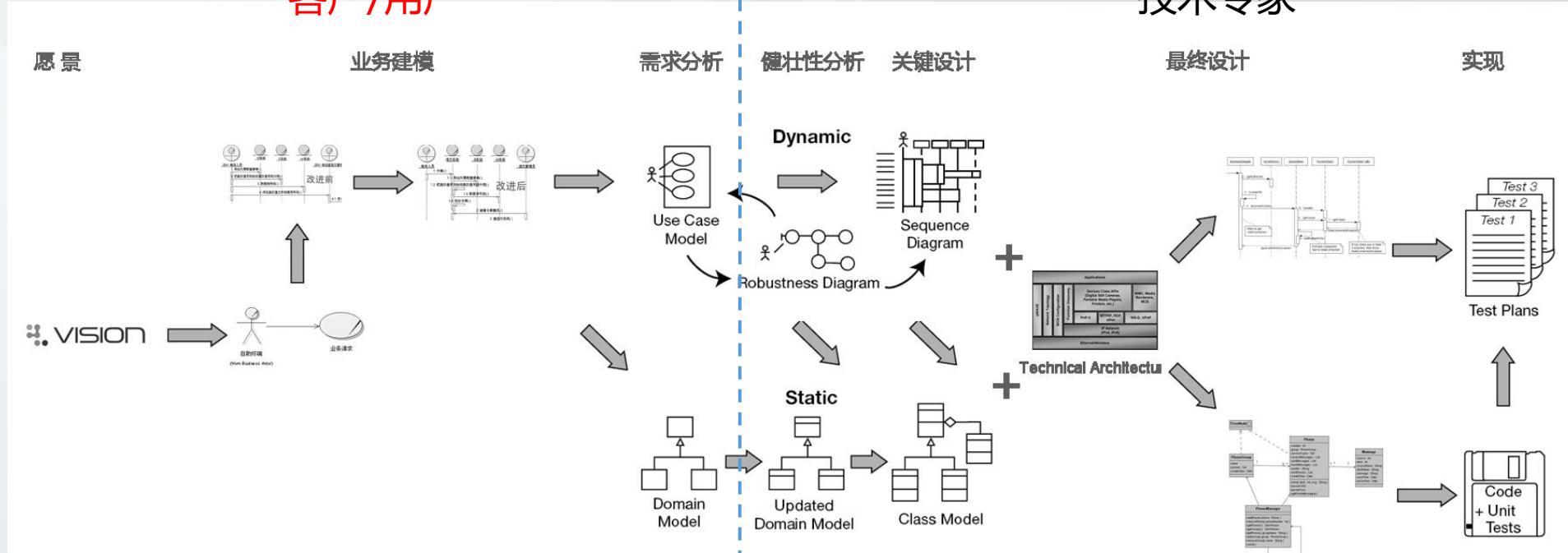
# 关键设计开始考虑如何制造的问题»»



客户/用户



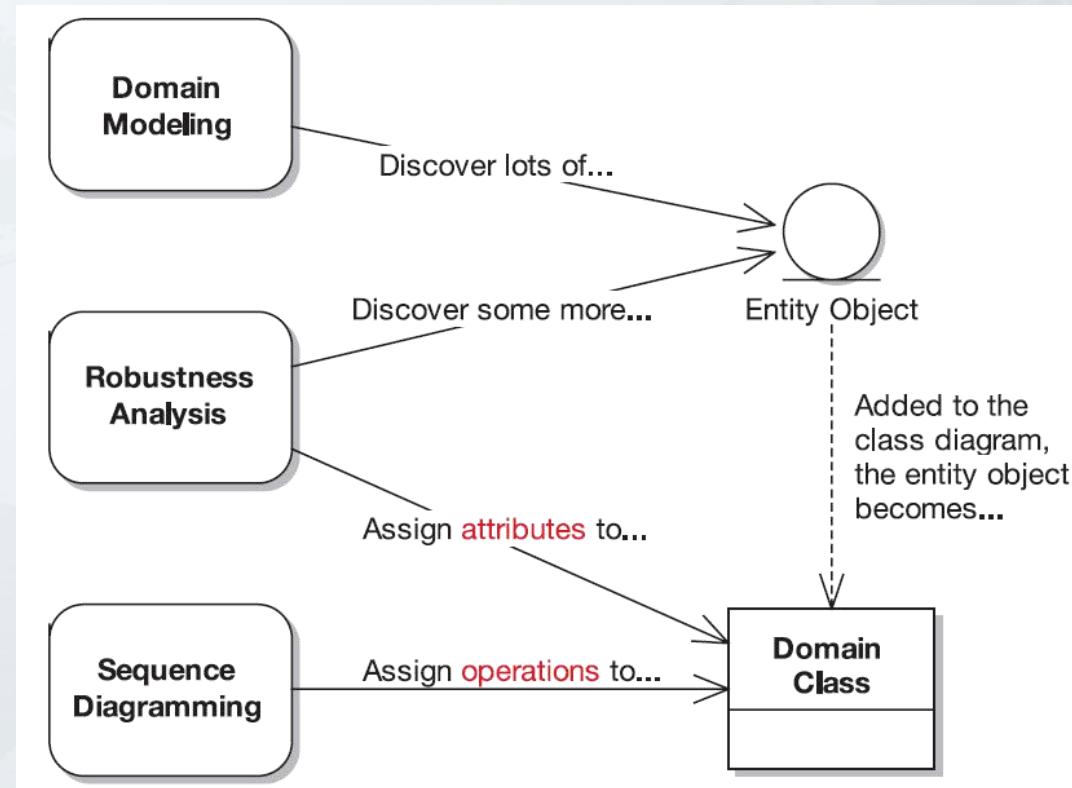
技术专家



增加收入

降低成本

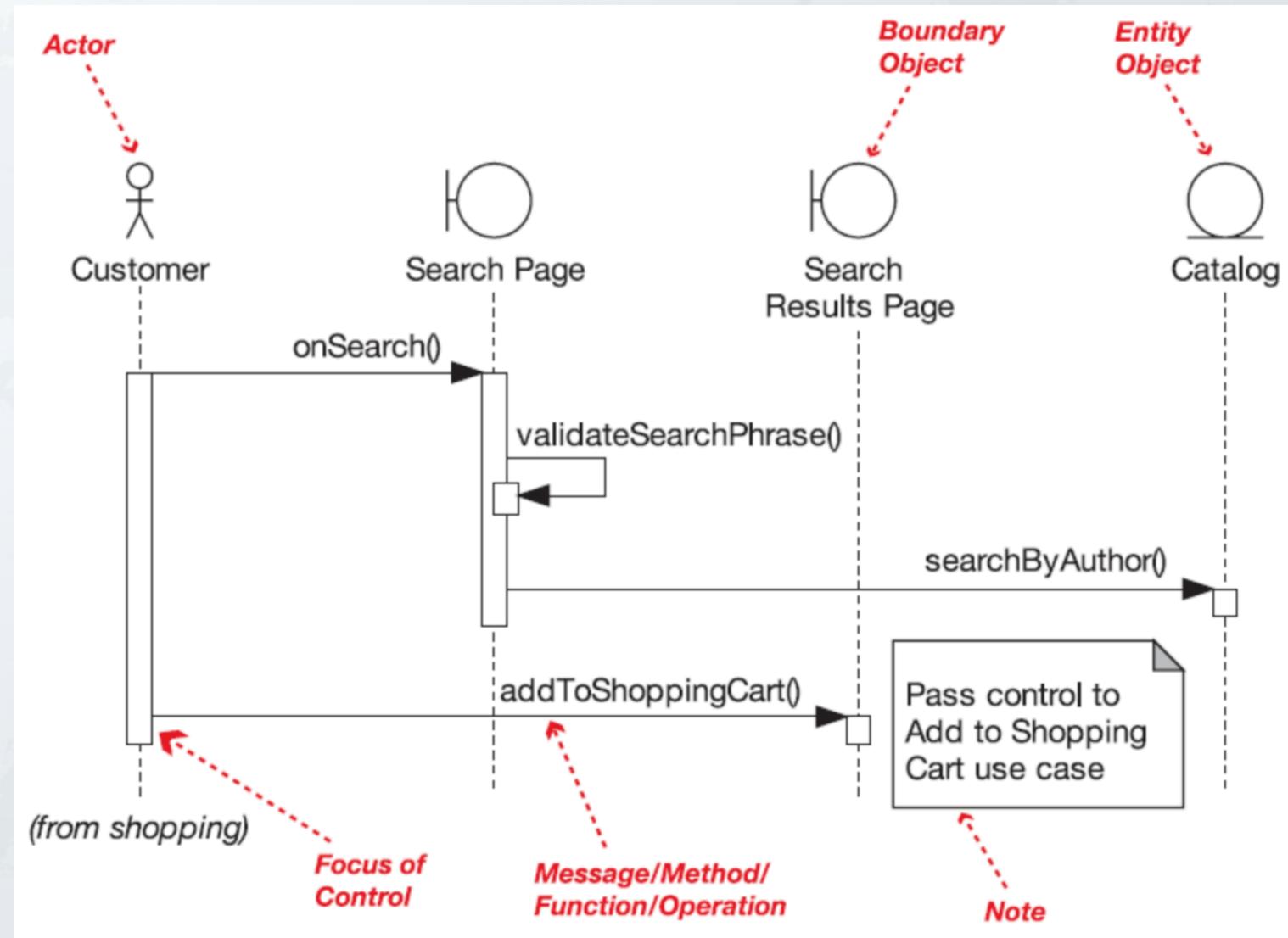
# 关键设计的方法和意义 »»



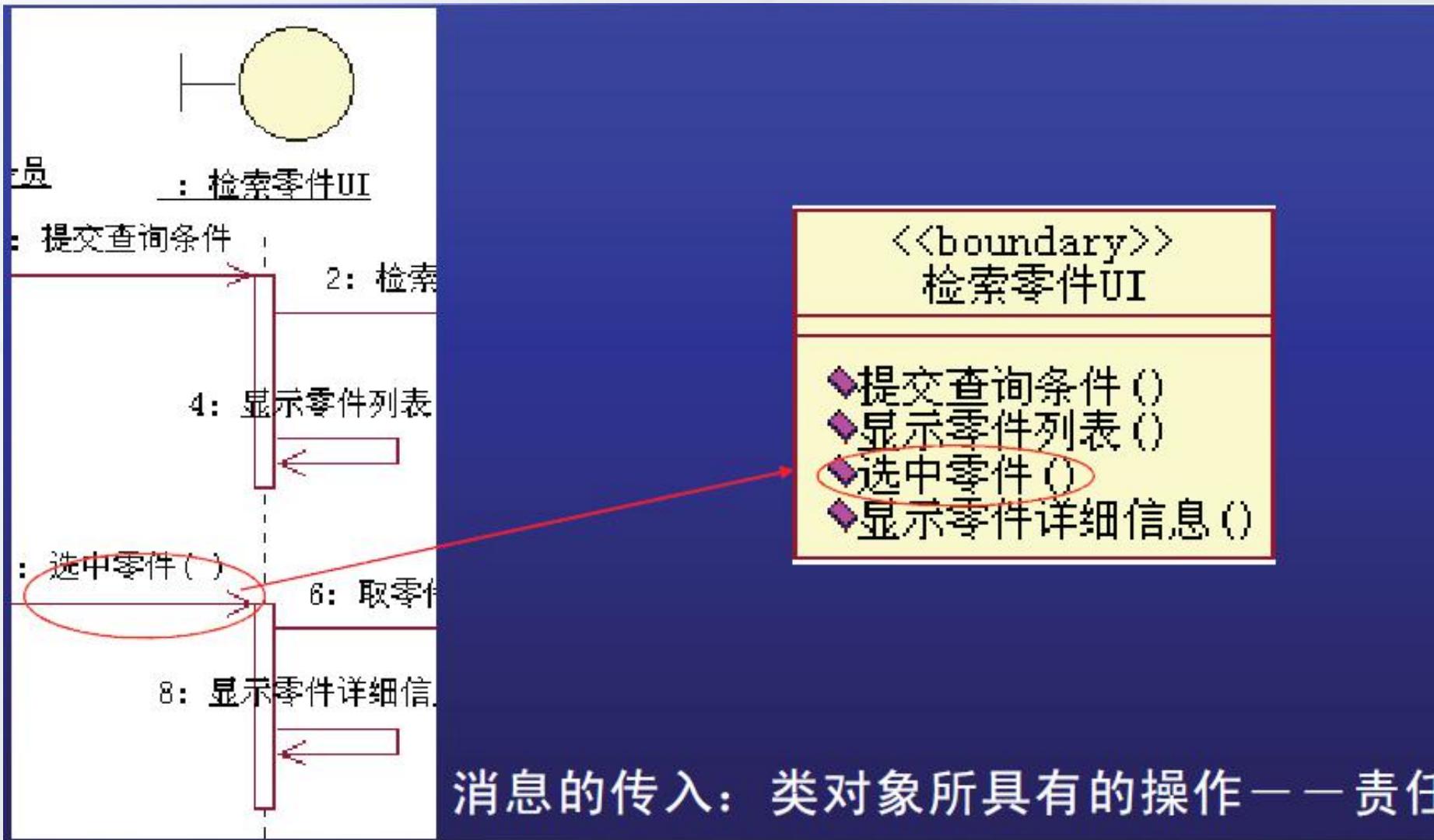
主要意义：就是要通过寻找对象之间的交互关系，进而进行方法（操作或行为）分配

基于用例图、用例描述和健壮性图，采用序列图来描述参与者、边界、实体之间的交互。

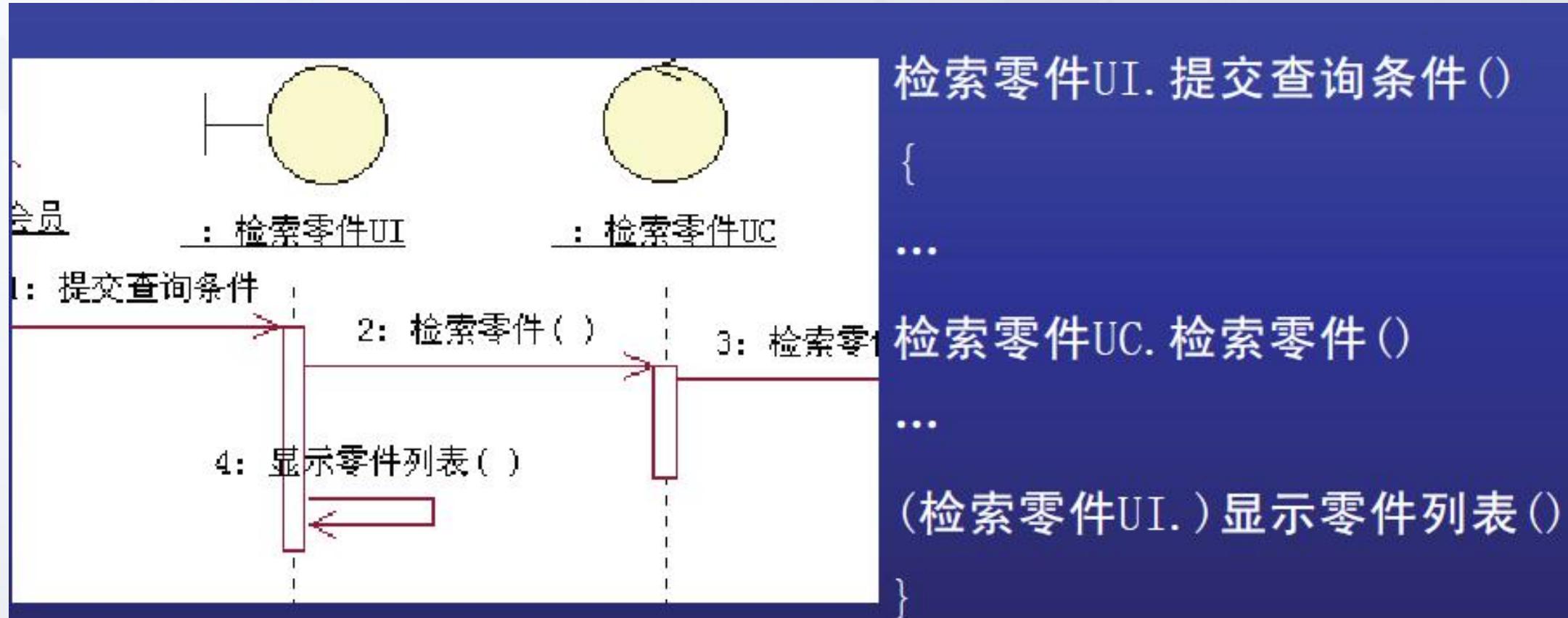
# 序列图[Sequence Diagram]的要素 »»



# 序列图和类图的映射 »»

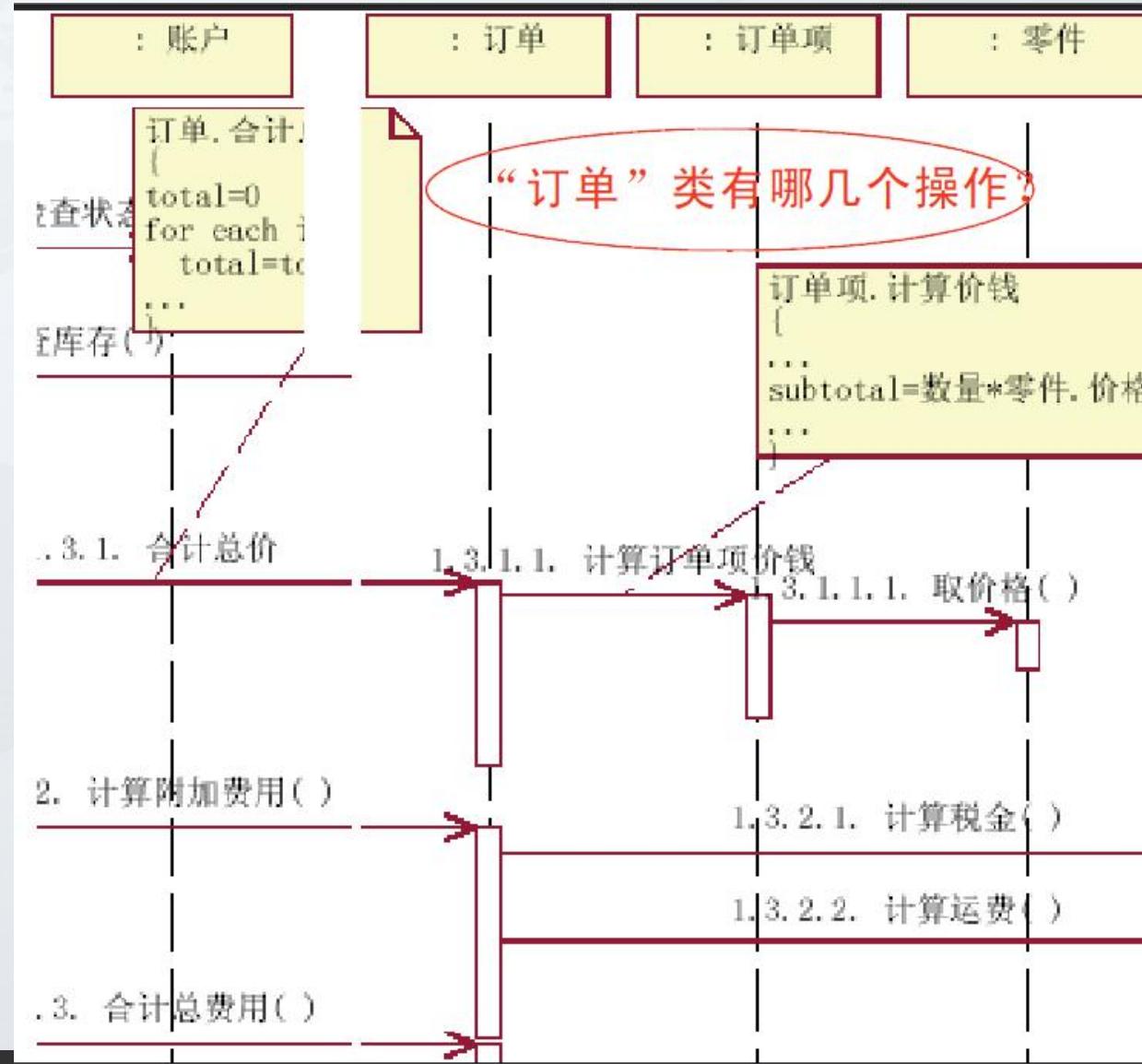


# 序列图和类图的映射 >>>



消息的传出：类对象完成操作所需合作——协作

# 练习 >>>



# 练习 >>>

请写出：“结账UC”类的“检查”操作内部的代码

: 结账UC

: 会员

: 账户

: 库存

.1. 检查( )



1.1.1. 检查账户状态( )

1.1.1.1. 检查状态( )

1.1.2. 检查库存( )

消息是否充分



.3. 结账( )

1.3.1. 合计总价

# 目录 >>

一

关键设计的意义和方法

二

关键设计的步骤

三

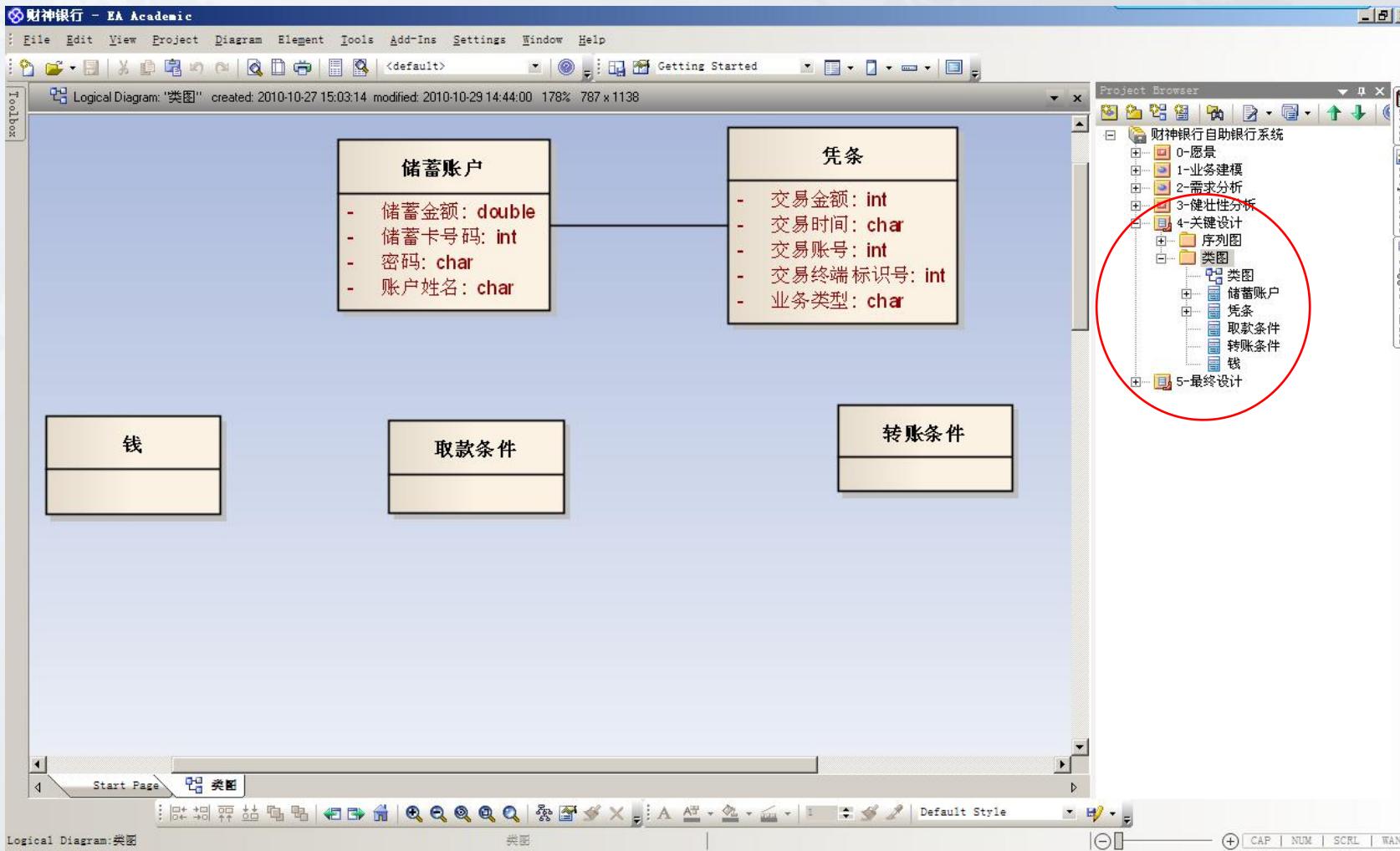
关键设计的复核



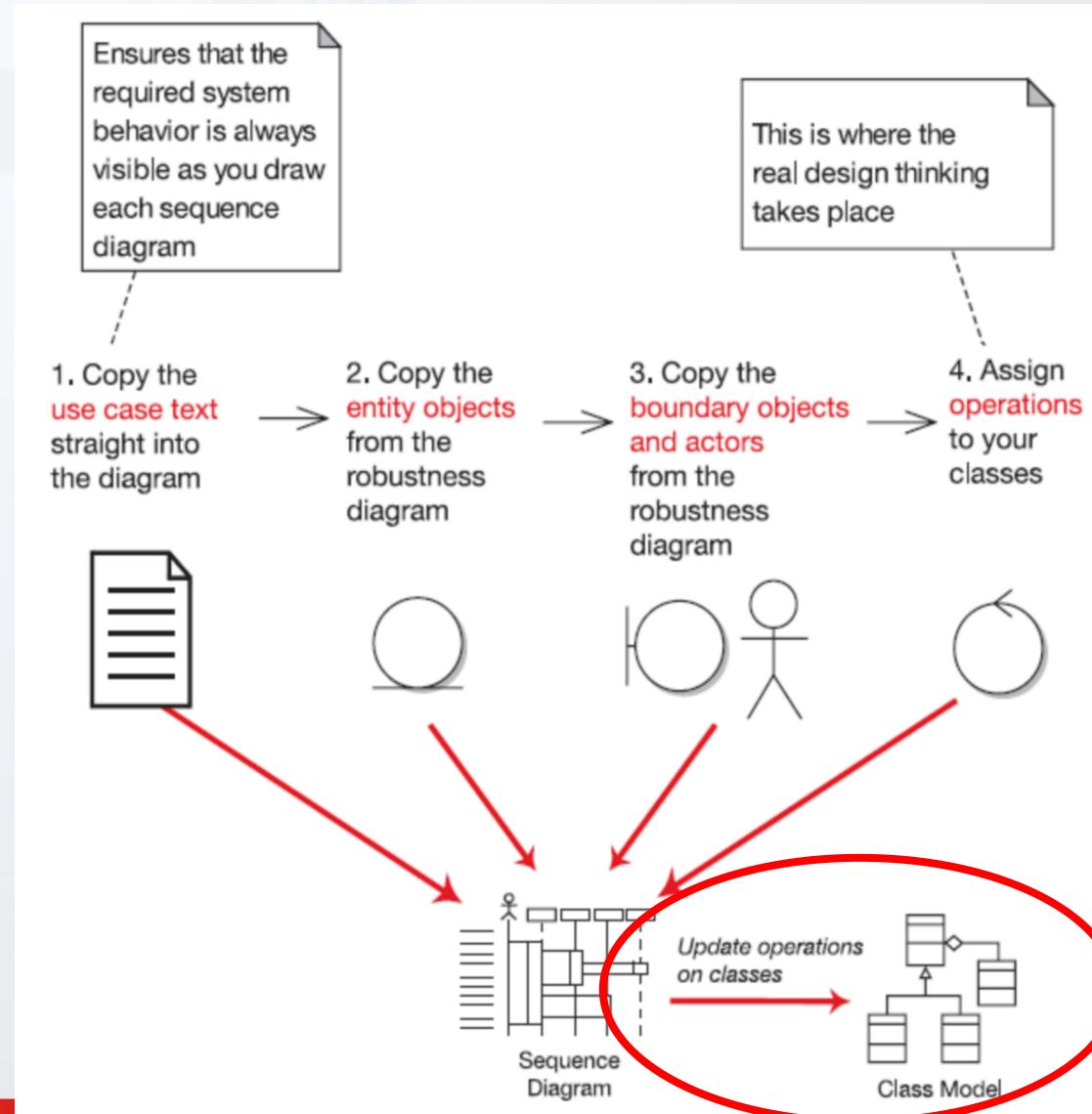
# 关键设计的步骤 »»

- 第一步：将现有的域模型直接作为第一版静态类模型；
- 第二步：基于用例描述和健壮性分析结果，画出每个用例的序列图；
  - 健壮性图中的控制类会转化为方法；
  - 如果也转化为控制类，那么就添加到类图中（**注意：边界类不添加到类图中**）；
- 第三步：整理静态类图和序列图；
- 第四步：关键设计复核，迭代更新用例图、类图和序列图；

# 第一步：形成第一版静态类图 >>>



## 第二步：画出每个用例的序列图 »»

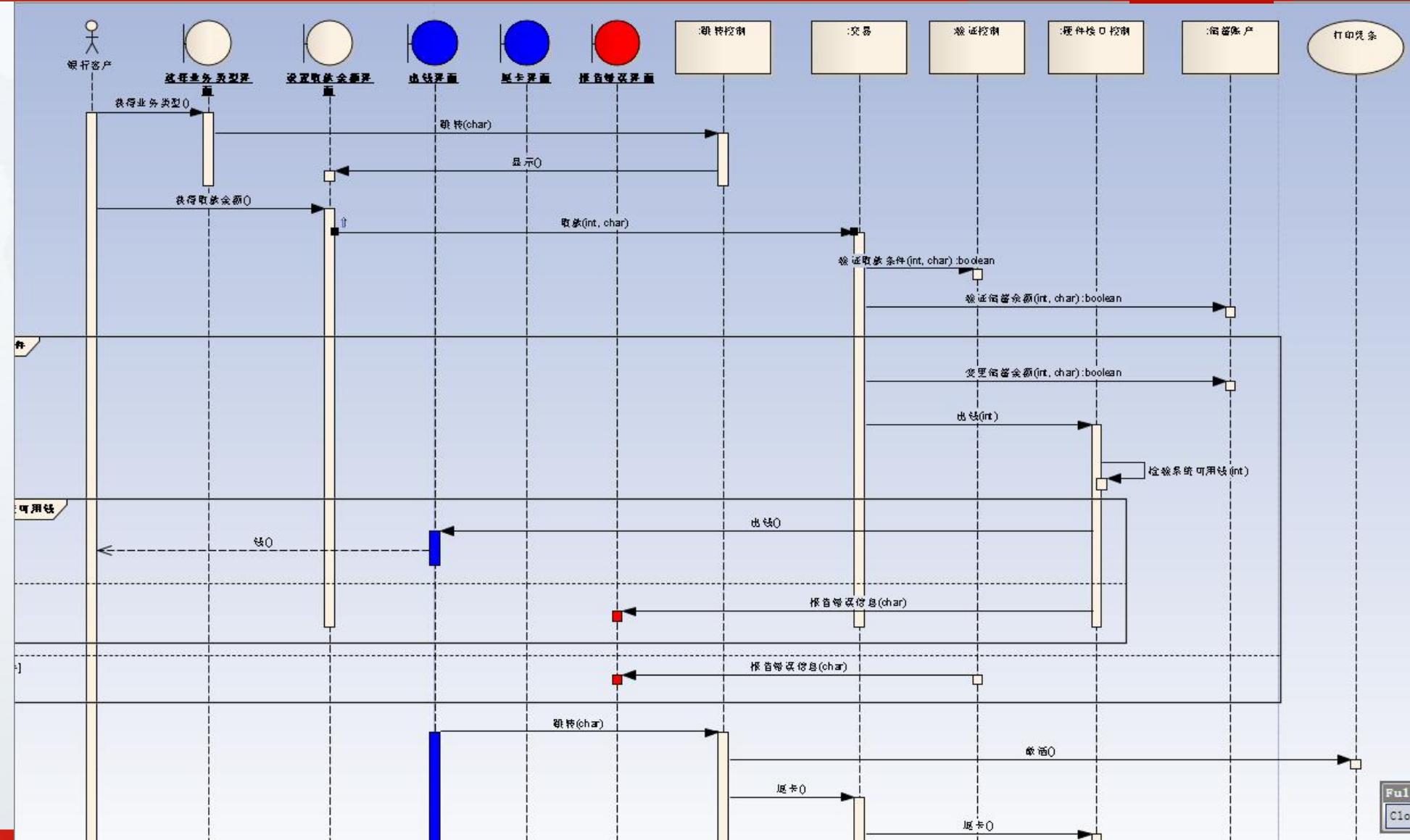


1. 逐步贯穿健壮性图上的每一个控制器，每次一个，画出序列图上相应的方法，然后核对，移至下一个控制器。
2. 控制器和方法之间并不一定是完全一对一对匹配的，也可能会转化为两个或多个方法。
3. 有时，控制器也可能会转换为一个真正的控制类。（例如：检查余额可以在账户类中，但跳转放在账户类中不合适，可以单独放到一个控制器类里）
4. 序列图会对类图做进一步的更新，完善其方法

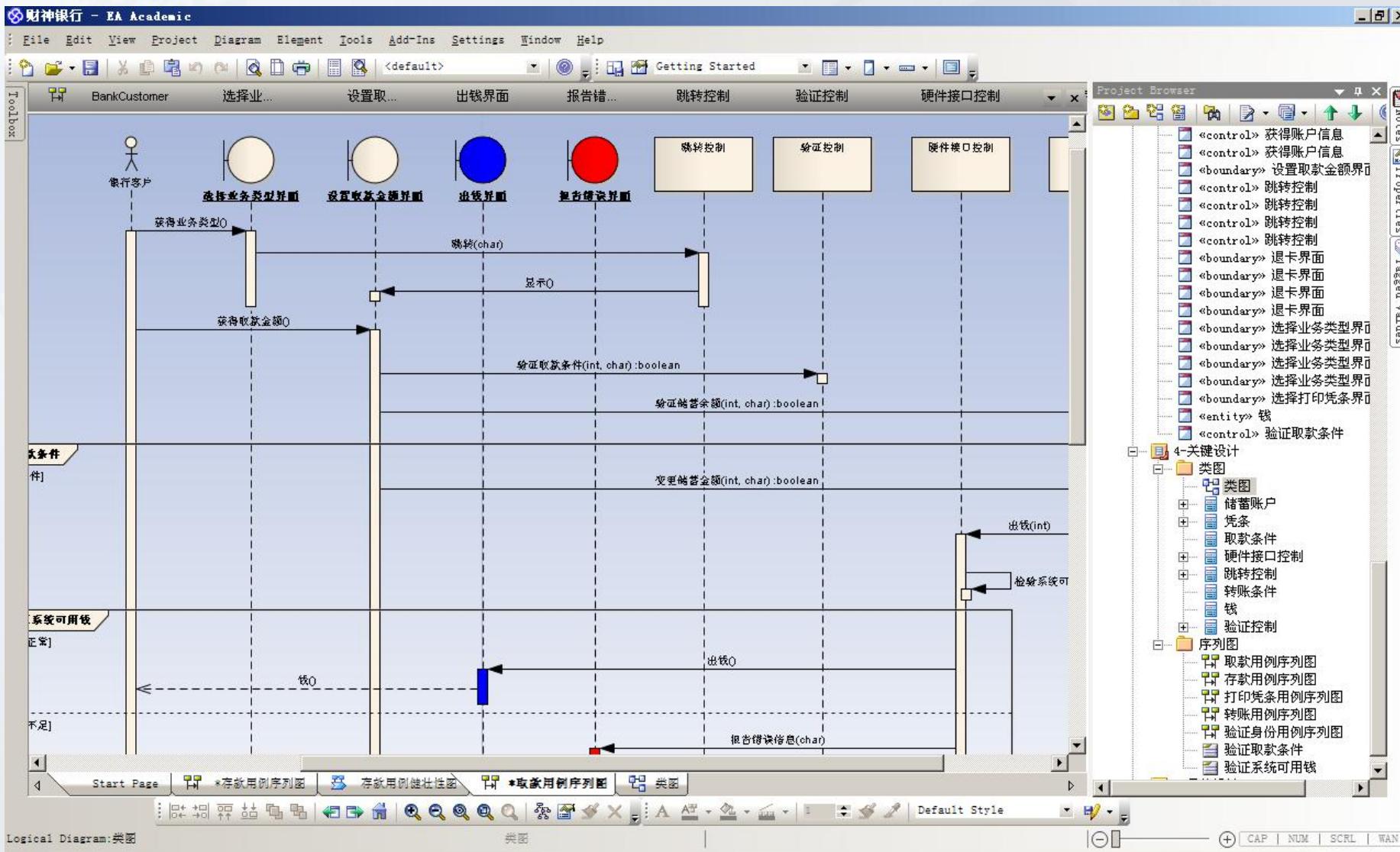
# 何决定控制器分配给哪个类？»»

- **高内聚、低耦合。**是判断设计好坏的标准。
  - 高内聚是指一个软件模块(类)是由相关性很强的代码组成，只负责一项任务，也就是常说的单一责任原则。
  - 低耦合是指一个软件模块与模块之间的接口，尽量的少而简单。如果某两个模块间的关系比较复杂的话，最好首先考虑进一步的模块划分，降低相互的依赖。这样有利于修改和组合。
  - 目的：使得模块的“可重用性”、“移植性”大大增强

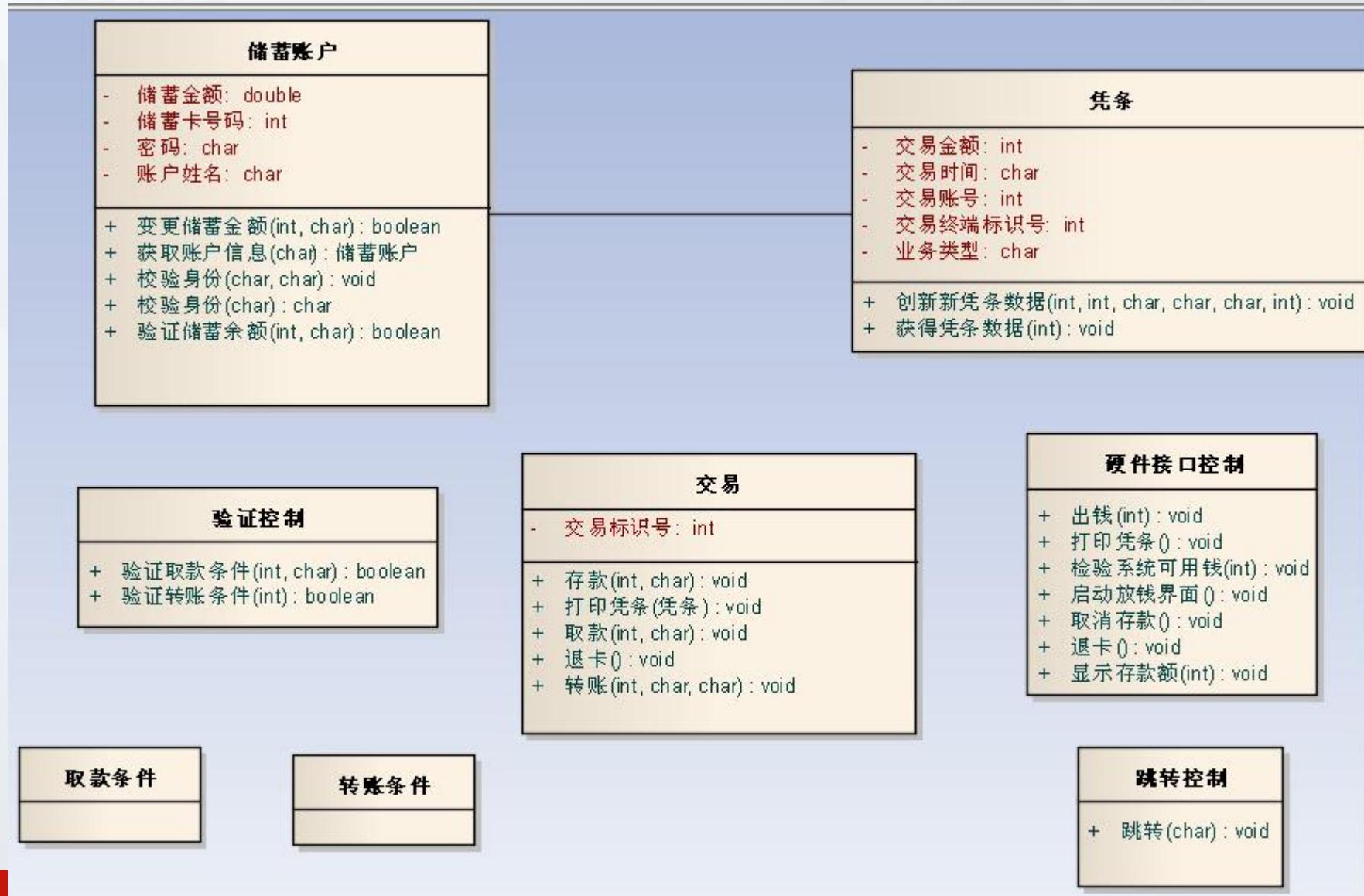
# 示例：取款用例序列图 >>>



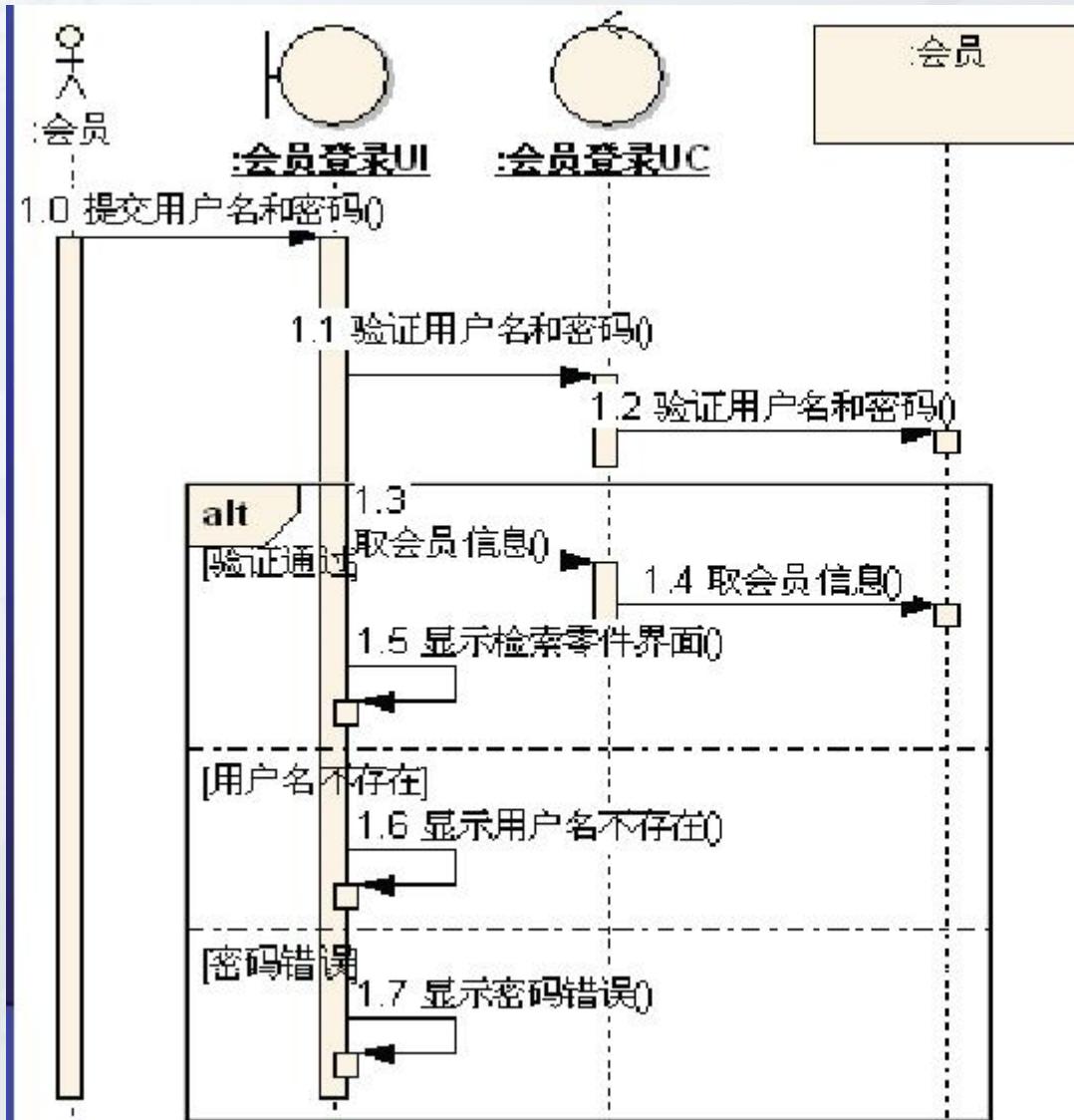
# DEMO:EA中进行序列图建模 >>>



# 示例：完成所有用例序列图后的类图

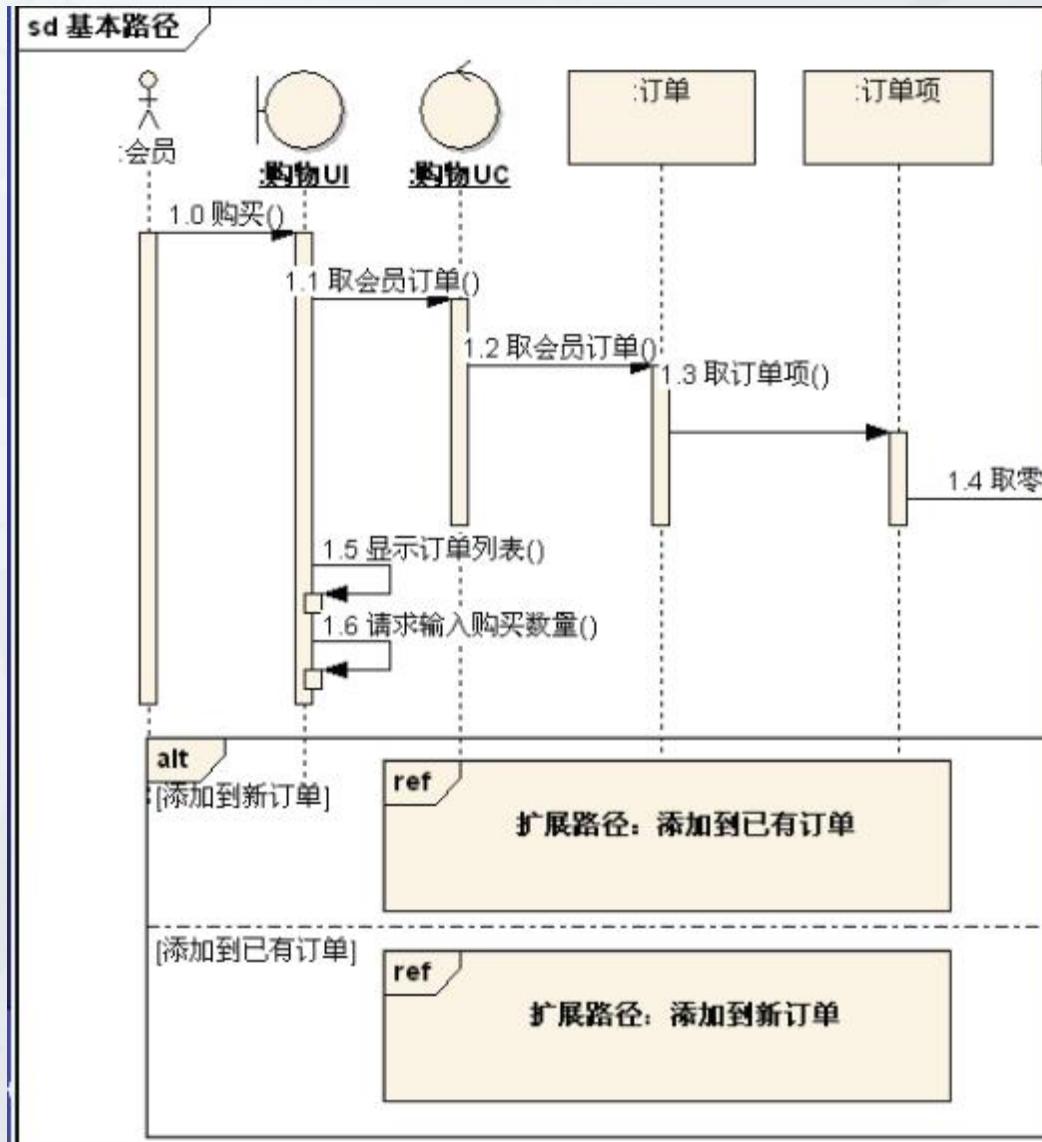


# 高级话题：画序列图的注意事项 »»



简单扩展点：可以合并到基本路径图

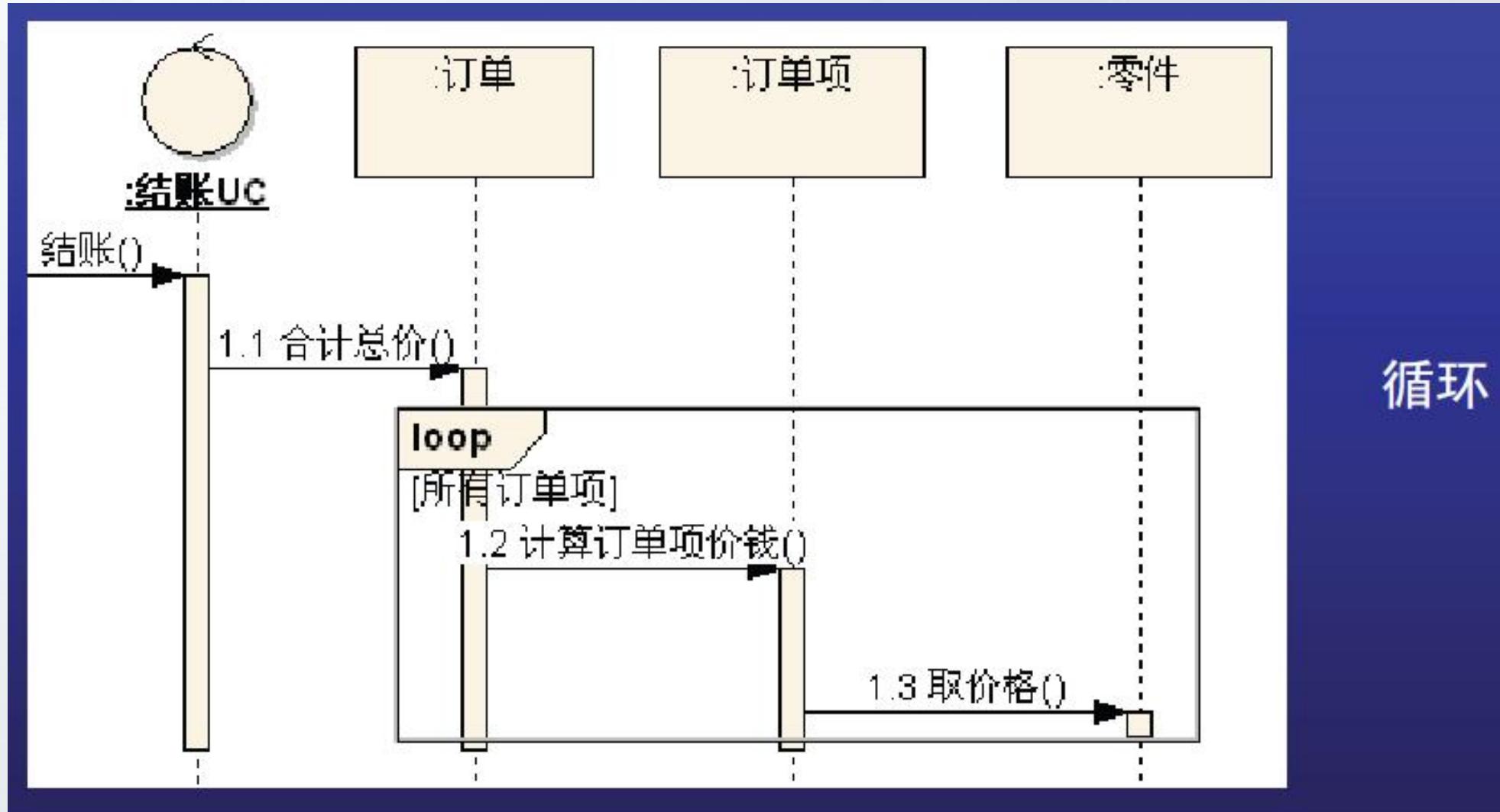
# 高级话题：画序列图的注意事项 »»



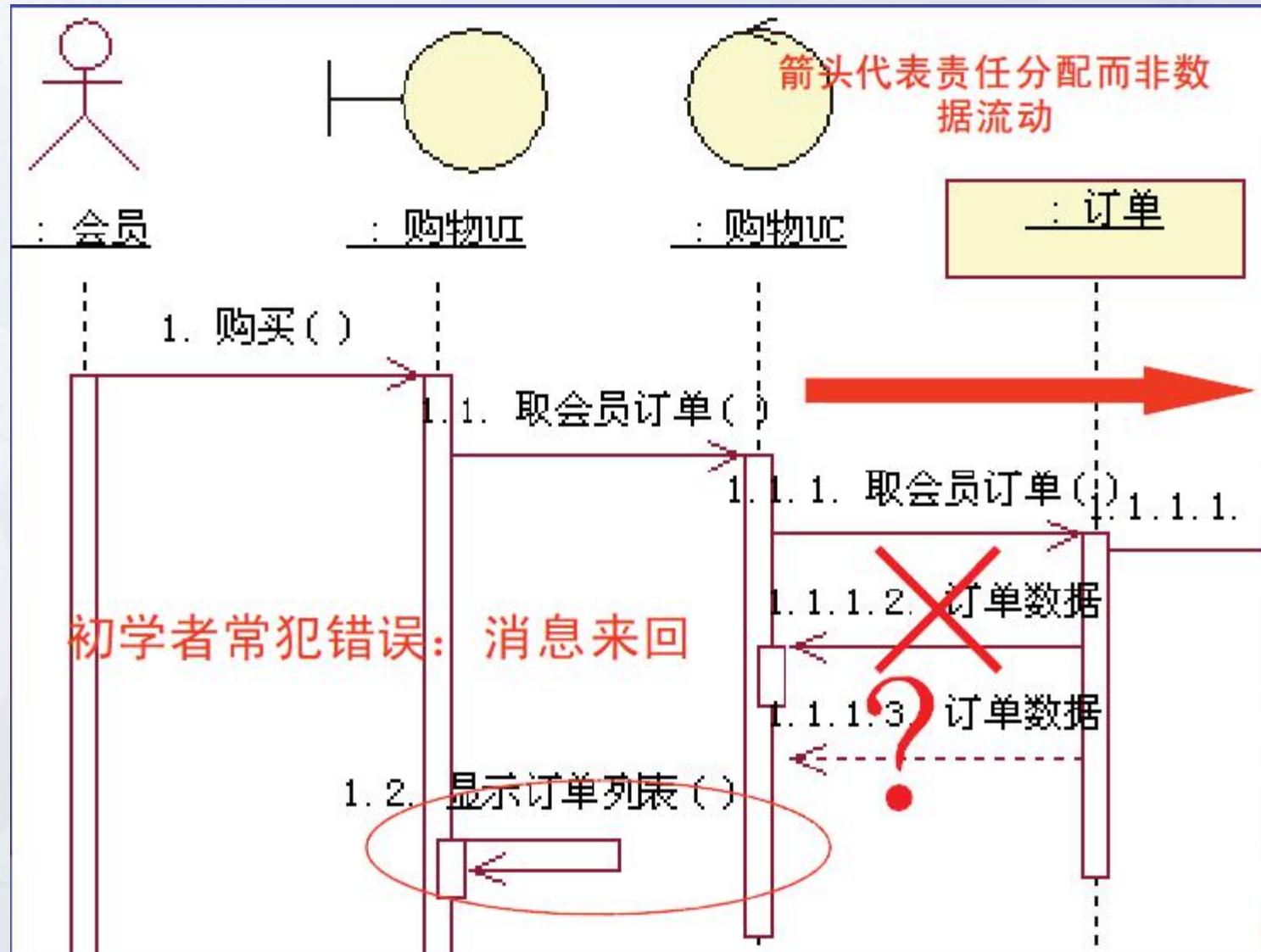
复杂扩展点：单独  
一张图，在基本路径图  
引用

Include、Extend用例也  
适用

# 高级话题：画序列图的注意事项 »»



# 高级话题：画序列图的注意事项»»



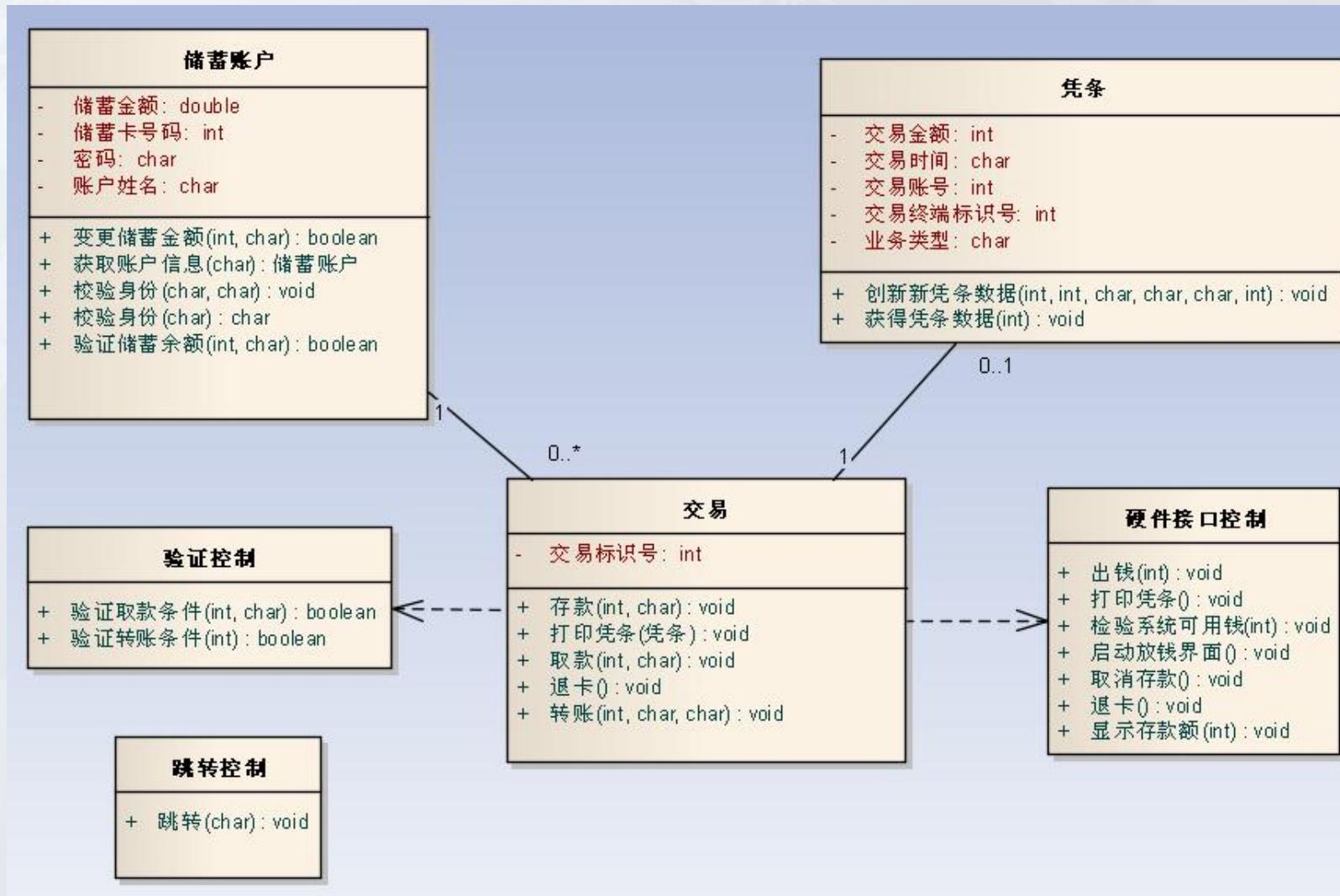
# 高级话题：画序列图的注意事项 »»

- 画方法[method]的同时将方法分配给类，反复核对类图，确保所有方法分配给了适当的类。
- 不要花费太多时间在控制焦点上。
- 不要把序列图画成了流程图。

## 第三步：整理静态类图和序列图 >>>

- 明确“钱”属于系统外实体对象，我们统一采用“硬件接口控制”来与外部实体交互，因此不会直接操作它，把它去掉；
- 验证控制器类完成了取款验证和转账验证，去掉单独的“取款条件”和“转账条件”；
- 结合序列图，定义类与类之间的关系（关联、聚合、组合、泛化、依赖），以及关系的多重性；

# 整理后的类图 >>>



# 目录 >>

一

关键设计的意义和方法

二

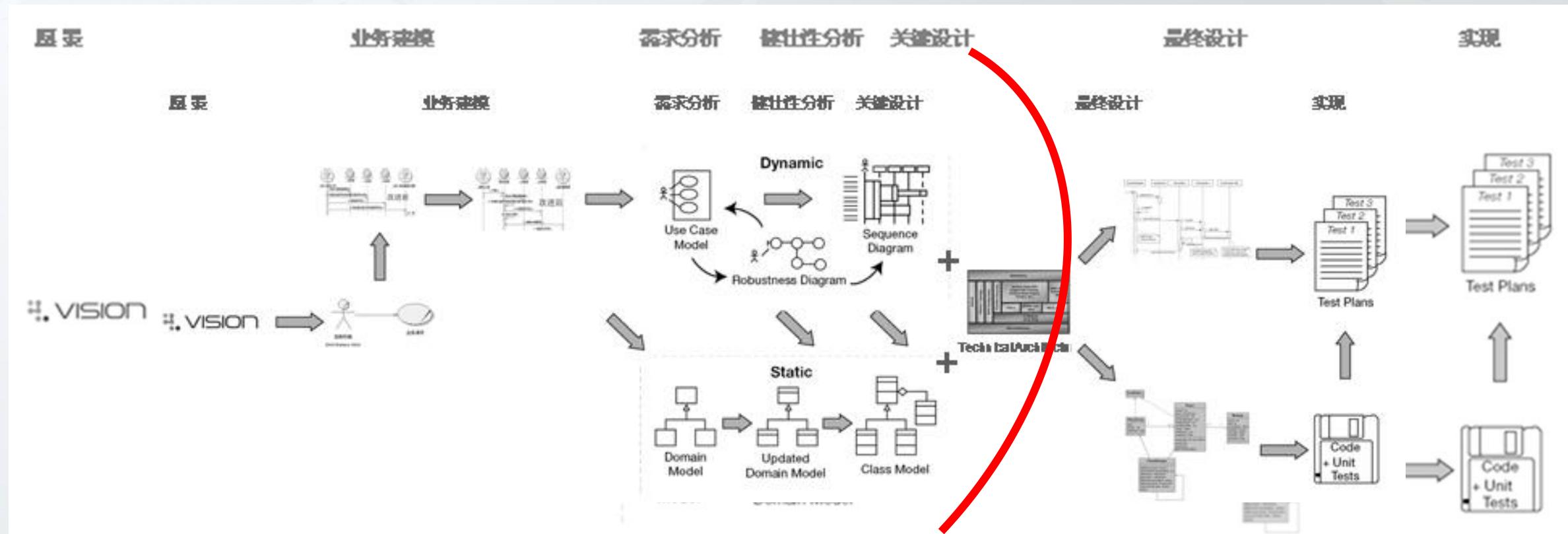
关键设计的步骤

三

关键设计的复核



# 第四步：关键设计复核 >>>



# 关键设计复核方法 »»

- 形式：面对面会议。**(可能多次、每次会很久 ☺)**
- 参会人：分析设计师、专家（分析、设计、开发）。
- 被审材料：用例图、用例描述、类图、序列图（为什么没有健壮性图？）；
- 过程：需求分析师主持，介绍需求分析成果，所有参与者交流讨论，达成**统一理解**和**确认**。
- 结论：所有参与者**签字**确认。（当然，也有可能是未达成共识，需要返工。）
- **注意：已不需要甲方人员参与。**  
**这是一个技术性的复核对话，**  
**因此，需要的是拥有技术思想的人员。**



# 关键设计复核的指导建议 >>>

- 确保关键设计的“如何做”和需求阶段的“做什么”匹配。也就是说每个用例都要和序列图匹配，包含了用例的基本流程和分支流程。
- 复核设计的品质。应该至少有一个设计专家在场。
- 检查消息的连贯性。检查时序图上消息箭头的指向，有时我们会发现对象之间缺少消息而造成跳跃，我们必须消除这些逻辑跳跃。
- 确保方法分配给了适当的类，类视图中的每一个类拥有适当的方法和属性。

# 示例：关键设计复核的迭代更新 »»

- 在关键设计复核时，通过重新审查用例图，我们发现关键设计中忽略了与银行系统交互的接口。原因是在取款、存款、转账的用例描述中就忽视了这一环节的描述（而用例图上已准确表达了）。因此，我们进行如下更新：
  1. 更新用例描述，增加关于与银行系统交互的描述；
  2. 可选：调整健壮性图，表达这个变化；
  3. 调整序列图和类图，表达这个变化；

# 示例：用例描述调整结果 >>>

## 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. 系统请求银行系统变更储蓄账户的储蓄金额；  

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

## 干系人利益

银行：防假钞

银行客户：安全、数钱准确

## 基本路径

1. 银行客户选择“存款”业务类型；
2. 系统提示放入要存的钱；
3. 银行客户将钱放入存钱槽，并确定；
4. 系统点钞；
5. 系统显示所点的数额，提示确认；
6. 银行客户确认；
7. 系统请求银行系统变更储蓄账户的储蓄金额；  

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

## 干系人利益

银行：安全、准确、节约运营成本

银行客户：安全、快捷

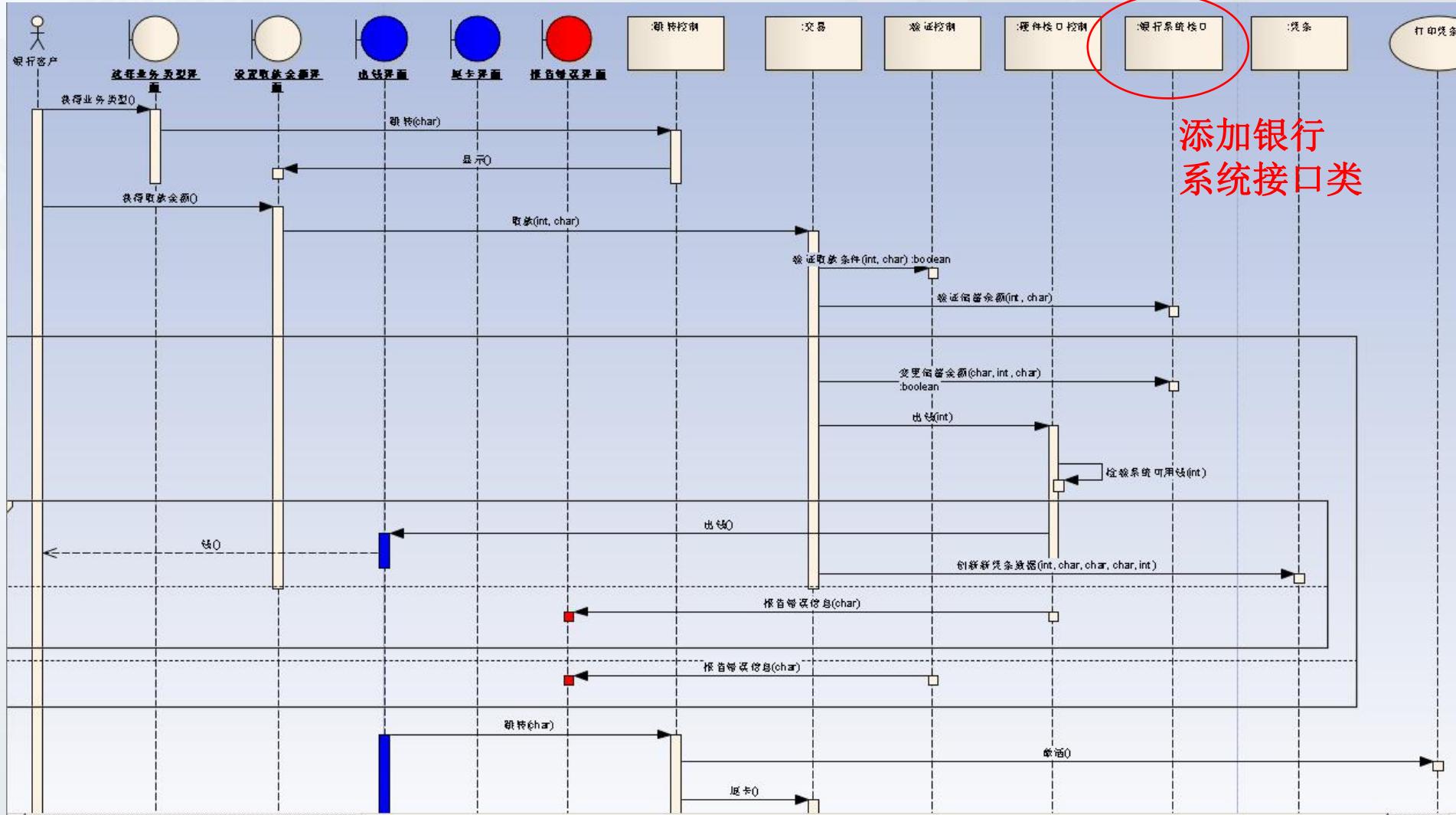
## 基本路径

1. 银行客户选择“转账”业务类型；
2. 系统提示输入转入账户；
3. 银行客户输入转入账户；
4. 系统请求银行系统校验转入账户的有效性；  

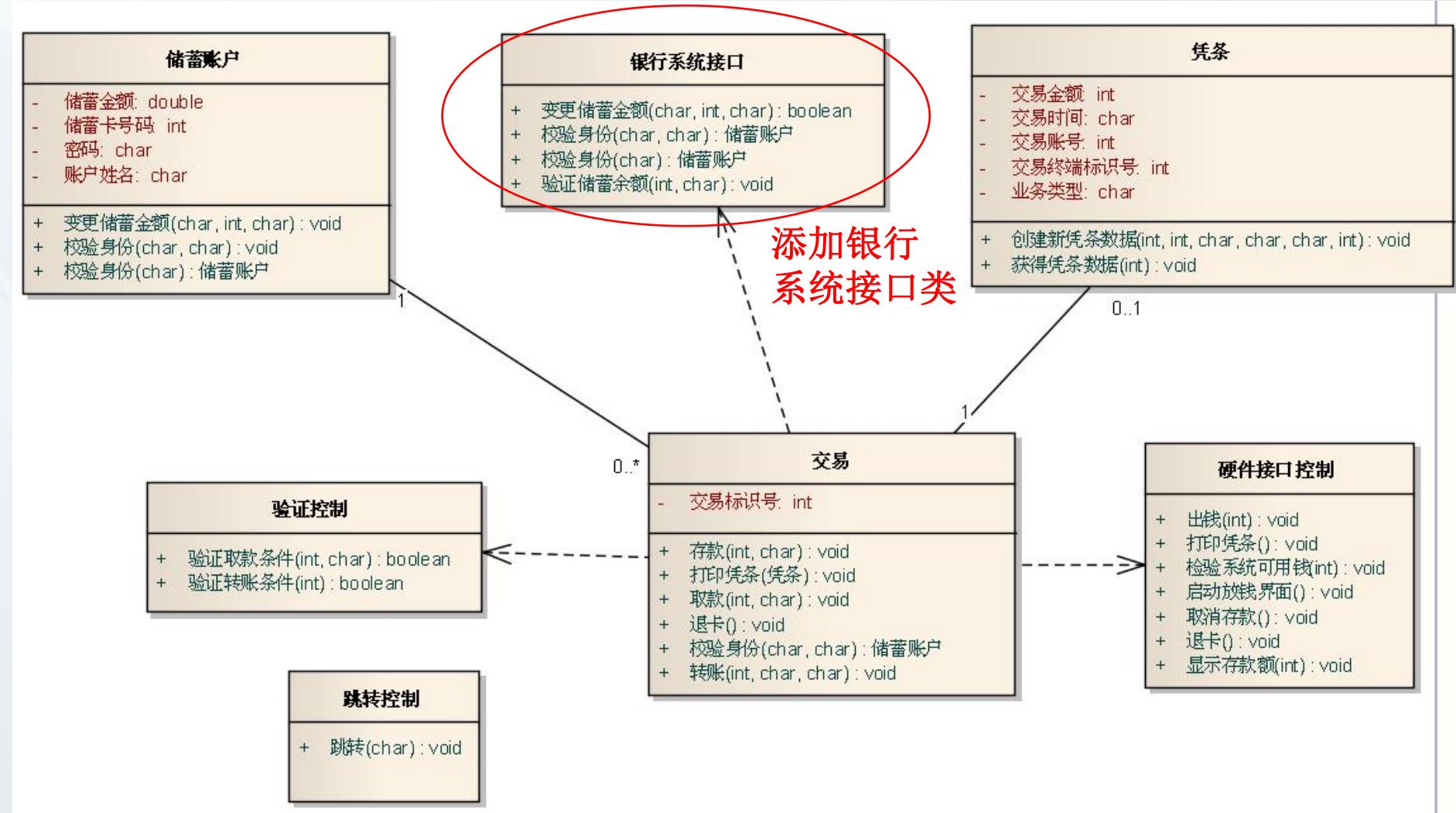
5. 系统提示转入账户开户姓名信息，提醒银行客户确认；
6. 银行客户确认；
7. 系统提示输入转账金额；
8. 银行客户输入转账金额并确认；
9. 系统校验本次转账是否符合转账条件（是否超过转账金额限制，转出账户金额是否足够）；  

10. 系统请求银行系统变更转出储蓄账户和转入储蓄账户的储蓄金额；
11. 系统激活“打印凭条”用例的扩展点；

# 示例：序列图调整结果 >>>



# 示例：类图调整结果»»



# 思考：关键设计结果的实用价值是什么？»»

- 跨平台。
- 外包。
- .....





**THANKS**