第三部分 软件需求工程

- 3.1 软件需求与需求工程
- 3.2 需求获取与建模
- 3.3 需求规格说明

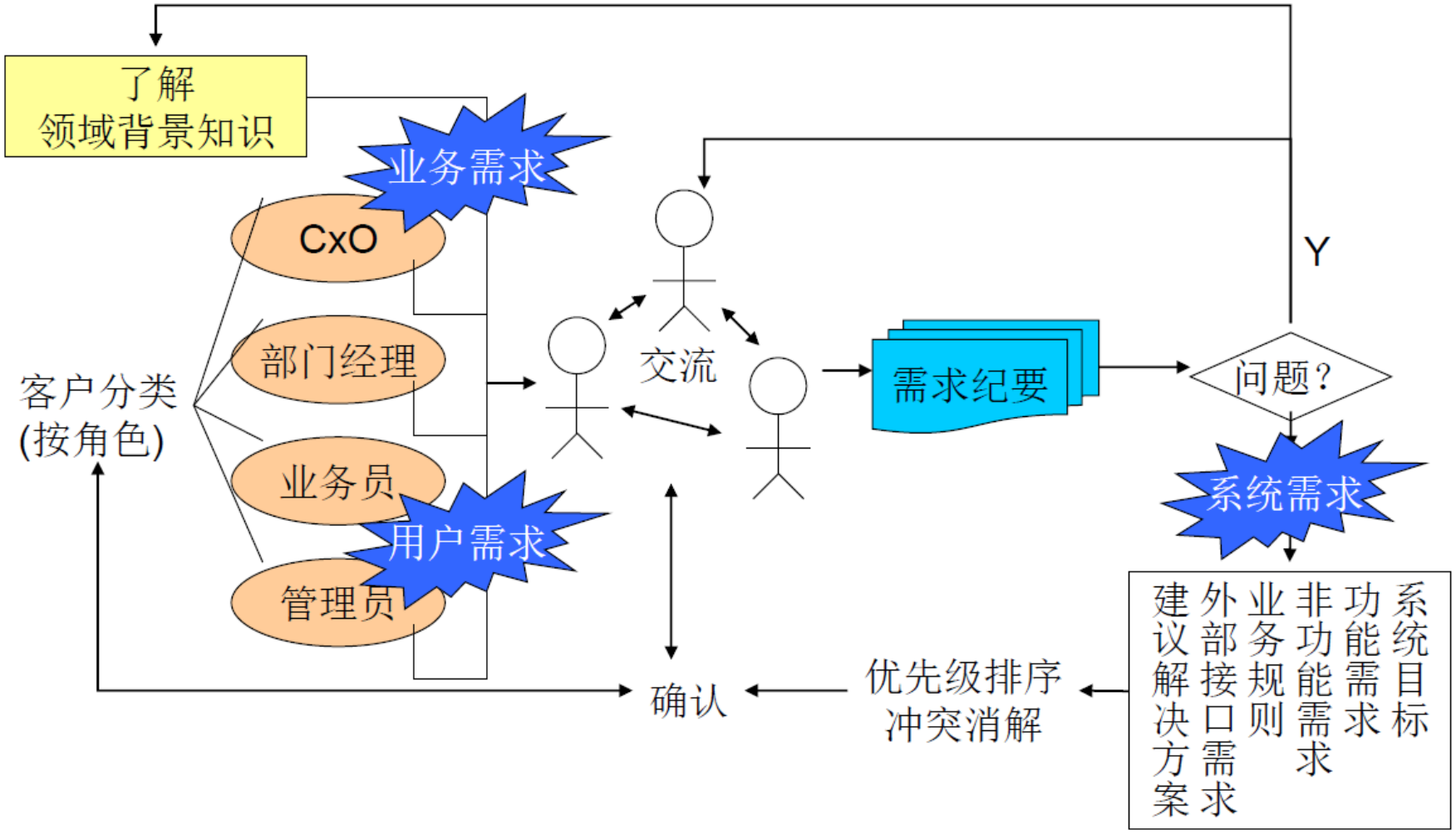
3.2 需求获取与建模

- 需求获取的挑战和途径
- 需求建模
  - 基于场景的方法
  - 基于类的方法
  - 基于模式的方法

需求获取的目标

- **需求获取(requirement elicitation)的目标：**
  - 收集准备建立的系统和正在使用的系统的信息，并从这些信息中提取用户和系统需求。
  - 为下一步的需求分析提供素材。

需求获取的基本步骤


了解
领域背景知识
业务需求
CxO
部门经理
客户分类
(按角色)
业务员
用户需求
管理员
交流
需求纪要
问题？
Y
系统需求
系统目标
功能需求
非功能需求
业务规则
外部接口需求
建议解决方案
优先级排序
冲突消解
确认

需求获取的基本步骤

- 第1步：了解相关背景和领域/行业的知识，确定产品所期望的用户类；
- 第2步：与客户企业或组织的高层人员进行交流，了解实际用户任务和目标以及这些任务所支持的业务需求；
- 第3步：与客户企业或组织的底层人员进行交流，获取每个用户类的详细的用户需求；
- 第4步：整理需求纪要，发现新问题，并重复1-3步；
- 第5步：需求分类和组织，以区别功能需求、非功能需求、约束条件、业务规则、外部接口需求、建议解决方案和附加信息；
- 第6步：优先排序和冲突解决；
- 第7步：得到最终需求清单，并与客户做最终签字确认。

“看似简单，实际却很难...”

“需求获取？不就是问问题吗？这有什么难的？”

“需求获取？客户要神马就开发神马？”

(1) “Yes, But”综合症

- 当你把新开发的系统展示给用户时...
  - Wow，太酷了！这正是我们想要的，你做了一个了不起的系统！”
  - ...五分钟后...，“Yes, but, 嗯....这个模块是怎么回事？...如果你把它修改成这样岂不是会变得更好？...如果那样的话，我觉得我会更喜欢它...”
- 不管之前他多么认同你的设计，在没有看到真正的系统之前，用户决不可能完全理解你的设计；
- 软件本质上的“无形性”造成的必然结果

(2) “Undiscovered Ruins”综合症

- “我知道的就这些，你们还想知道什么？”
- “我们已经发现了所有的需求，现在让我们开始着手开发吧…”

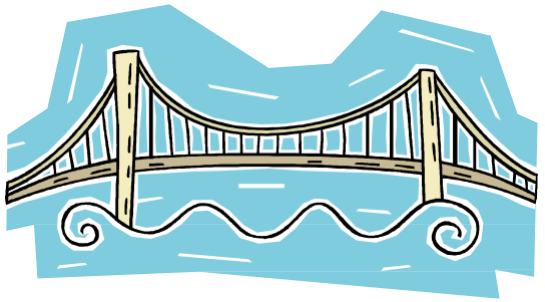
- “知道的越多，不知道的也越多”
- “**Undiscovered Ruins**”：请问尚未被发现的废墟有多少呢？

- 需求是永无止境的，变化的！

(3) “User and Developer”综合症

- 中国猫：喵喵~喵~
- 日本猫：涅呀~涅呀~

- 软件开发中，开发人员与用户处于不同的知识、技术层面，所关注的目标不同，双方在沟通时必然存在**communication gap(**交流的鸿沟**)**。



小结

问题	解决方案
“Yes, But”综合症：直到开发人员将用户描述的东西交给他们，用户才认为他们知道自己要什么	尽早提供可选择的启发技术 原型开发、迭代方法等
“Undiscovered Ruins”综合症 用户不知道自己需要什么，或知道但不知如何表达	详细调研，将用户当作领域专家来认识和激励，尝试其他交流和启发技术
“User and Developer”综合症 双方在沟通时存在交流的鸿沟	熟悉应用领域，把分析员放在用户的位置上，采用用例分析方法

需求获取的手段

- 面对面访谈(face-to-face interviewing)
- 问卷调查(Survey Research Methods)
- 专题讨论会(workshop)
- 头脑风暴(brainstorming)

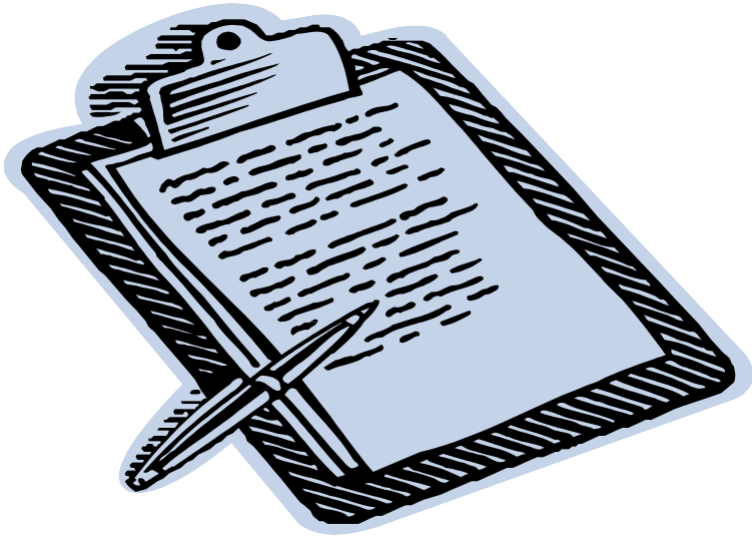
- 重点注意业务单据等资料的收集、整理！！---分析的依据
- 多种方法要复合在一起使用，效果更好

面对面访谈(Face-to-face Interviewing)

- 类型
  - 有组织的面谈通常事先制定议程，问题可以是开放的
  - 自由面谈则无既定议程
- 面谈的优点
  - 可以获得大量丰富的数据
    - 有助于发现观点、感受、目标及事实
  - 可以进行深入探讨，根据面谈前一阶段获知的内容调整后续的问题
- 面谈的缺点
  - 大量的数据是定性的，难以分析
  - 难于对问题的回答者进行比较
  - 面谈技巧难于掌握

问卷调查(Survey Research Methods)

- 预先定义的问题
- 面对广泛涉众的时候使用
- 统计分析的结果看起来更科学
- 什么时候使用问卷调查
  - 大基数的被访者
  - 需要关于良好定义的特定问题的答案
  - 验证有限次面谈得出的结论
  - 当你需要一个特定的结果的时候



Use a Questionnaire

- Prepare interview questions in advance
- Ask open–ended questions
- Ask essential questions
  What? Why? When? Who? How? Where?

问卷调查(Survey Research Methods)

优点

- 可快速获得大量反馈
- 可以远程执行
- 可搜集关于态度、信念及特性的信息

缺点

- 简单的分类导致对上下文的考虑较少
- 留给用户自由表达其需要的空间较小
- 选择样本时可能存在偏见
- 样本规模太小
- 自由发挥问题难于分析

专题讨论会(Workshop)

目标

- 介绍项目成员和干系人
- 收集需求列表
- 在会议过程中，运用头脑风暴、故事板、角色扮演、现有系统评估等方法获需求

指导原则

- 主持人组织研讨:
  - 给每个人发言的机会
  - 保持研讨话题的相关性
  - 定义需求属性
  - 记录需求发现
  - 总结并作出结论

头脑风暴(Brainstorming)

- 目标
  - 通过群组效应，激发对新产品和系统的新想法
  - 在需求不完全明确的情况下比较有用
- 指导方针
  - 采用有组织的研讨会形式
  - 百花齐放，不评价、不争论、不批评
  - 不受现实可行性限制
  - 新观点多多益善
  - 抛砖引玉
  - 互相启发

IDEA GENERATION

Broad ideas may yield good requirements or result in requirements creep

Set expecta;ons for par;cipants

3.2 需求获取与建模

- 需求获取的挑战和途径
- 需求建模
  - 基于场景的方法
  - 基于类的方法
  - 基于模式的方法

需求建模

- 基于场景的方法
  - 用户故事（User Story）
  - 用例图（User Case Diagram）
  - 活动图（Activity Diagram）
- 基于类的方法
- 基于模式的方法

用户故事（User Story）

用户故事（User Story）是从用户角度对功能的简要描述。用户故事卡片格式:

格式：作为一个<角色>，可以<活动>，以便于<价值>。

谁要使用这个功能?

需要执行什么操作?

完成操作后带来什么好处?

作为一个“网站管理员”，我想要“统计每天有多少人访问了我的网站”，以便于“我的赞助商了解我的网站会给他们带来什么收益。”

好的用户故事应具备的特征：INVEST

- 独立性（Independent）：尽可能避免故事之间存在依赖关系，否则会产生优先级和规划问题。
- 可协商（Negotiable）：故事是可协商的，不是必须实现的书面合同或者需求。
- 有价值（Valuable）：确保每个故事对客户或者用户有价值的，最好是让用户编写故事。
- 可估算（Estimatable）：开发者应该能够预测故事的规模，以及编码实现所需要的时间。
- 短小的（Small）：故事尽量短小，最好不超过10个理想人天，至少在一个迭代中完成。
- 可测试（Testable）：所编写的故事必须是可测试的。

用户故事的3个C

关于用户故事，Ron Jeffries用3个C来描述它：

- 卡片（Card） - 用户故事一般写在小的记事卡片上。卡片上可能会写上故事的简短描述，工作量估算等。
- 交谈（Conversation）- 用户故事背后的细节来源于和客户或者产品负责人的交流沟通，用户如何与系统交互。
- 确认（Confirmation）- 如何通过验收测试确认用户故事被正确完成。

用户故事

作为顾客可以使用信用卡从而购买购物车中的商品。

注释：接受Visa、 Master和American Express信用卡。

Conversation

- 用 Visa、MasterCard和American Express进行测试 （通过）
- 用 Visa借记卡测试（通过）
- 用 Diner‘s Club（大莱卡）测试（失败）
- 用正确的、错误的和空的卡号测试
- 用过期的卡测试
- 用不同限额的卡测试（包括超出银行卡的限额）

Confirmation

用户故事

作为一个玩家，可以通过显示排名，让自己在服务器中的地位获得认可。

- 激发玩家“斗志”，鼓励购买道具；
- 实现有技术问题：实时查看不现实；
- 小玩家对自己的排名不太关心，不会为了提升排名去购买道具，只有少数顶级大佬才会受此蛊惑。

作为一个排名靠前的付费玩家，可以通过显示排名，让自己在服务器中的地位获得认可（以刺激消费）。

系统每周重新排名一次，而且只显示前xxx名玩家。

用户故事类型

不要只写成作为一个用户，而是要把用户区别对待，这样才能更好地理解他们如何使用和为何使用

作为一名维基用户，我希望上传一个文件到维基，以便可以和同事进行分享。

作为一名客服代表，我希望为客户问题创建一个记录卡，以便记录和管理客户支持请求。

作为一名网站管理员，我想要统计每天有多少人访问了网站，以便赞助商了解这个网站会给他们带来什么收益。

用户故事类型

用户故事除了描述功能以外，有时也可以用于描述非功能需求。

系统必须支持IE8、IE9、Safari5、Firefox7 和Chrome15浏览器。

作为开发人员，我想为新的过滤引擎做两个参考原型，以便知道哪一个更合适。

- 两个原型要执行性能测试、规模测试和类型测试
- 编写简短的备忘录以描述试验、结果和建议

用户故事类型

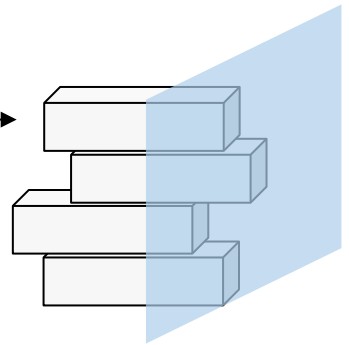
系统必须支持IE8、IE9、Safari5、Firefox7 和Chrome15浏览器。

作为开发人员，我想为新的过滤引擎做两个 参考原型，以便知道哪一个更合适。

修复缺陷跟踪系统的缺陷#256，这样可以使客户在搜索项中输入特殊字符不会出现异常

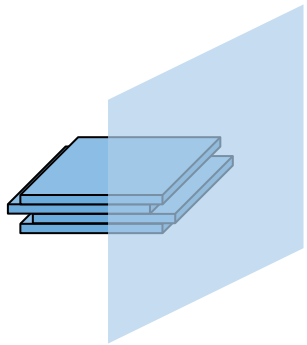
产品订单

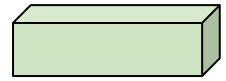
高优先级



产品订单
(粗粒度功能)

迭代订单
(细粒度功能/任务)





增量功能

产品负责人

产品故事的市场价值来决定故事的优先级

迭代计划会议

开发团队

进一步细化成开发任务，最后由开发团队实现

每次迭代时会对功能和功能的优先级进行调整

优先级	名称	用户故事描述	估算
1	浏览商品	作为一名顾客想购买商品而不确定型号时，我希望能浏览网站在售的商品，按照①商品类型和②价格范围进行过滤。	
2	搜索商品	作为一名顾客在查找某种商品时，我希望能进行不限格式的文本搜索例如按照短语或关键字。	
3	注册账户	作为一名新顾客，我希望注册并设置一个帐户，包括用户名、密码信用卡和送货信息等。	
4	维护购物车	作为一名顾客，我希望能将指定商品放入购物车（稍后购买）、查看我的购物车内的商品以及移除我不想要的物品。	
5	结账	作为一名顾客，我希望能完成我购物车内所有商品的购买过程。	
6	编辑商品规格	作为一名工作人员，我希望能够添加和编辑在售商品的详细信息（包括介绍、规格说明、价格等）。	
7	查看订单	作为一名工作人员，我希望能登录并查看一段时间内应该完成或已经完成的所有订单。	

一个完整的用户故事

- Card [优先级：xx 工作量估算：]
  - 作为一个视频制作者，我希望上传视频到网页，以便于任何其他用户能够在网页上看到这个视频。
- Conversation
  - “上传按钮”会出现在这个网站的每一个网页上
  - 视频不能大于100MB或者长于10分钟
  - 视频的格式包括.flv,.mov,.mp4,.avi和.mpg
  - 能够实时显示上传进度

一个完整的用户故事

- Confirmation
  - ✓检查“上传视频”按钮
  - ✓上传视频的具体操作
    1. 检查.flv,.mov,.mp4,.avi和.mpg视频文件是否能上传
    2. 检查其它视频文件是否不能上传
    3. 检查视频大于100MB时是否会报错
    4. 检查视频长于10分钟时是否会报错
  - ✓检查是否能够实时显示上传进度

User Story注意事项

- 要对用户进行分类，倾听每种用户的需求
  1. 注册用户，非注册用户（游客），开发人员
  2. 注册用户可以看报表，产生报表
  3. 非注册用户（游客）可以看报表，但不能产生报表
  4. 注册用户可以选择加载从xx年-xx年的数据进系统
  5. 非注册用户（游客）不能加载数据，只能使用系统中默认加载的数据
  6. 开发人员可以选择系统支持哪个系统平台，而其他人员不能选择
  7. 开发人员可以选择爬虫爬取xx年-xx年数据
  8. 非开发人员不能操作爬虫，只能选择系统加载xx年-xx年数据
  9. 开发人员还可以再继续细分。。。

User Story注意事项

- 给出来的user story没有价值/好处
  - 例子：作为一个广告商，我希望有一个过滤选项
    - 作为一个广告商，他为什么想要一个过滤选项？他的目的是什么？
  - 例子：作为一个商业用户，我希望ATM能够输入密码
    - 作为一个商业用户，他为什么希望ATM能够输入密码？
- 如果没有接受标准或满意条件，可能会使故事由于缺乏理解而失败，后续的测试或者测试用例也将被改变。

User Story注意事项

- User Story的目标太多了
  - 导致User Story的功能太多了
- User Story的目标和功能不匹配
  - 目标是做搜索，Conversation里有分享功能
- 描述User Story的文字太主观
  - 作为一个xx用户，我希望看到xx领域的发展趋势，以便我能对xxx项目做投资
  - 作为一个xx用户，我希望能够正常的登陆自己的账号，并得到我想看到的那部分数据
  - 希望做一个漂亮的界面
  - 希望xx按钮出现在清晰可见的位置

需求建模

- 基于场景的方法
  - 用户故事（User Story）
  - 用例图（User Case Diagram）
  - 活动图（Activity Diagram）
- 基于类的方法
- 基于模式的方法

UML

UML语言-面向对象的建模语言



Booch Rumbaugh Jacobson

- 借助面向对象的概念和机制来表示系统
  - 系统：软件系统，现实系统
  - 需要表示机制和语言
- 1980s-1990s出现了大量面向对象（Object Oriented）建模语言
  - 多达几十种
  - 代表性有Booch方法、OMT方法和OOSE方法
- 期望
  - 各有千秋，却又有很多类似之处，往往让使用者无所适从,需要吸收各种技术的优点
  - 寻求一种概念清晰、表达能力丰富、适用范围广泛的面向对象的建模语言

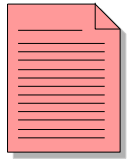
UML

UML: Unified Modeling Language

- Unified(统一)
  - 提取不同方法中的最好建模技术，如OMT(James Rumbaugh)，Booch method(Grady Booch)和OOSE(Ivar Jacobson)
  - 统一表示方式，具有标准化的表示手段
- Modeling(建模)
  - 对现实系统和软件系统进行可视化描述
  - 建立系统模型
- Language(语言)
  - 图形语言：语法、语义和语用
  - 包括规则，约束 扩展机制

UML

- 用来可视化(visualize) 、描述(specify)、构造(construct)和文档化(document)软件密集型系统的各种产品
- 支持不同人员之间的交流(Communication)



Specifying

Visualizing

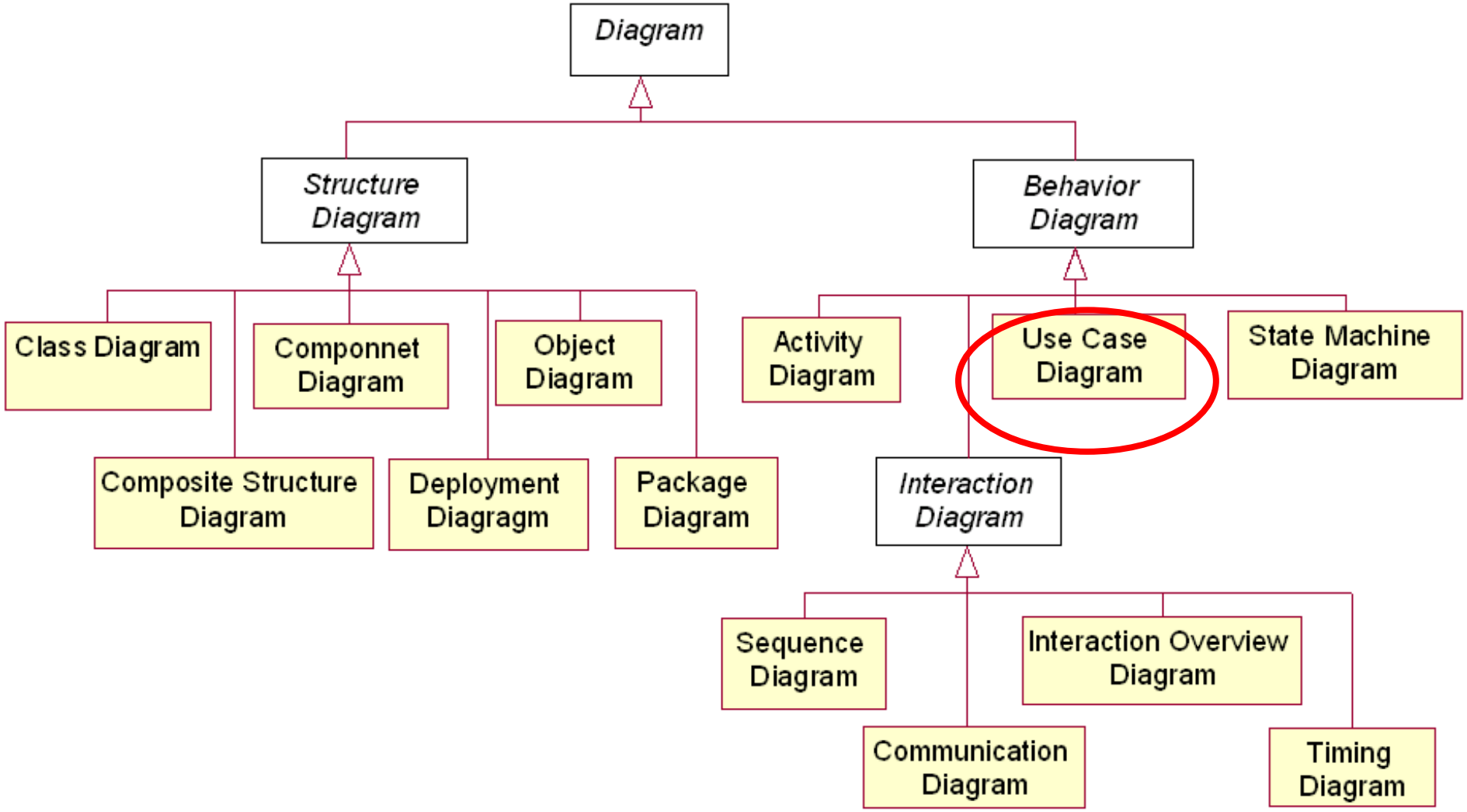
Constructing

Documenting

Business Modeling

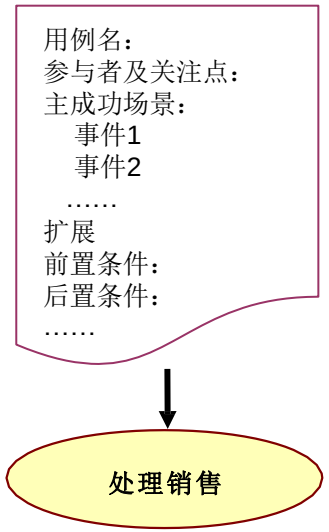
Communications

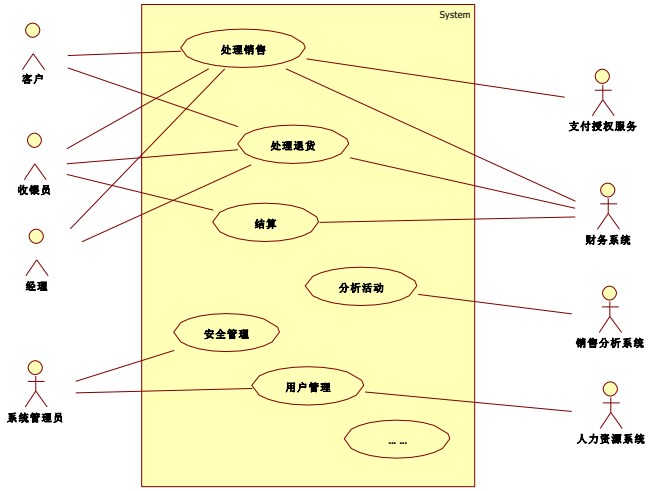
UML2.0的图的分类-用例图



基于用例的需求获取

- **什么是用例**
  - 用例描述了使用者使用系统完成用户某一目标的过程
  - 用例把系统描述为实现特定客户目标所要做的事
  - 一个用例聚焦于**单一的目标**
  - 一个用例可能**包含多个功能**（需求）
- **什么是用例图**
  - 用例图包含系统的所有用例
  - **用例图是系统的蓝图**

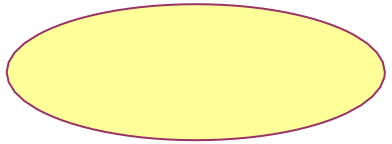




用例图基本概念

- 参与者(actor)：是某些具有行为的事物，可以是人、计算机系统或者组织，且对系统本身而言参与者是外部的。例如收银员
- 场景(scenario)：是参与者和系统之间的一系列特定的活动交互，也称为用例实例(use case instance)
- 用例(use case)：就是一组相关的成功和失败场景集合，用来描述参与者如何使用系统来实现其目标

用例的特征



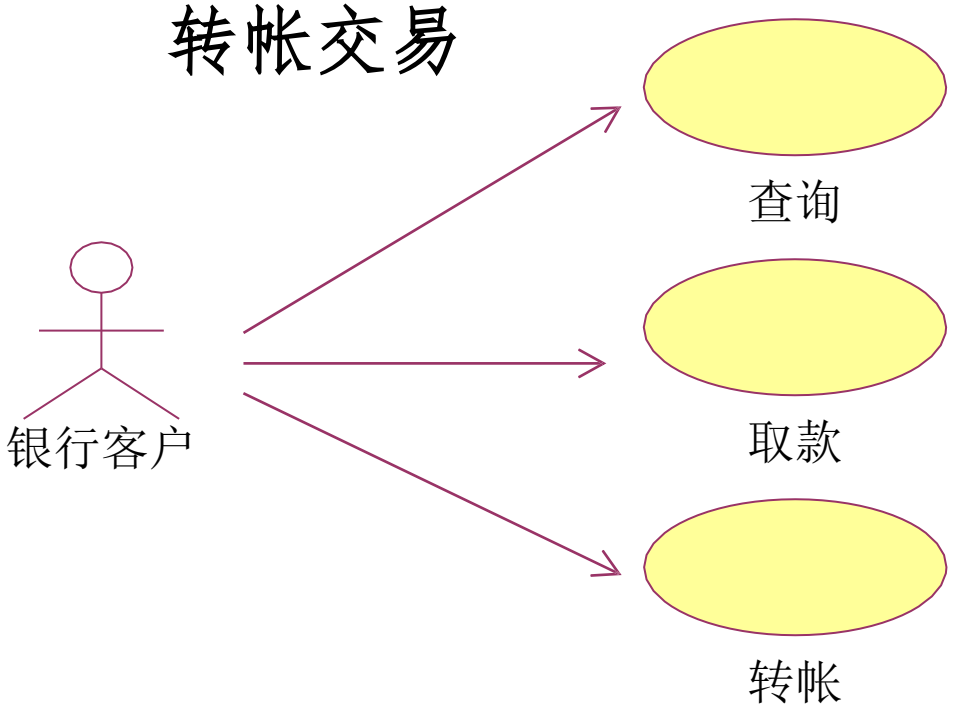
- 用例：站在用户角度定义软件系统的外部特征
- 四大特征：
  - 行为序列(sequences of actions)：一个用例由一组可产生某些特定结果的行为构成，这些行为是不可再分解的(接收用户输入、执行、产生结果)
  - 系统执行(system performs)：系统为外部角色提供服务；
  - 可观测到的、有价值的结果(observable result of value)：用例必须对用户产生价值；
  - 特定的角色(particular actor)：某人、某台设备、某外部系统、等等，能够触发某些行为。

用例方法的基本思想

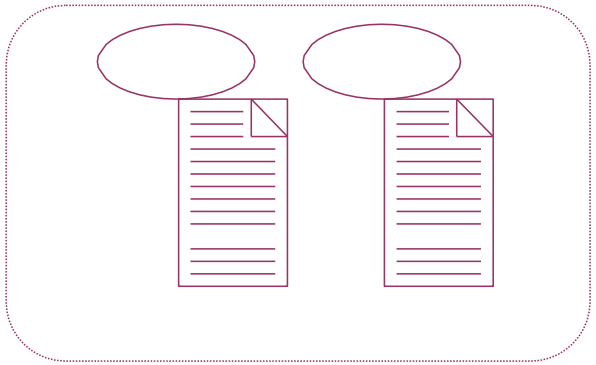
- **用例方法的基本思想：从用户的角度来看，他们并不想了解系统的内部结构和设计，他们所关心的是系统所能提供的服务，也就是被开发出来的系统将是如何被使用的。**
- **用例模型主要由以下模型元素构成：**
  - 参与者(Actor)：存在于被定义系统外部并与该系统发生交互的人或其他系统，代表系统的使用者或使用环境。
  - 用例(Use Case)：用例从较高层次上给出参与者和系统之间交互的故事。
  - 通讯关联(Communication Association)：用于表示参与者和用例之间的对应关系，它表示参与者使用了系统中的哪些服务(用例)、系统所提供的服务(用例)是被哪些参与者所使用的。

示例：ATM系统的用例

- 参与者：银行客户
- 用例：银行客户使用自动提款机来进行银行帐户的查询、取款和转帐交易

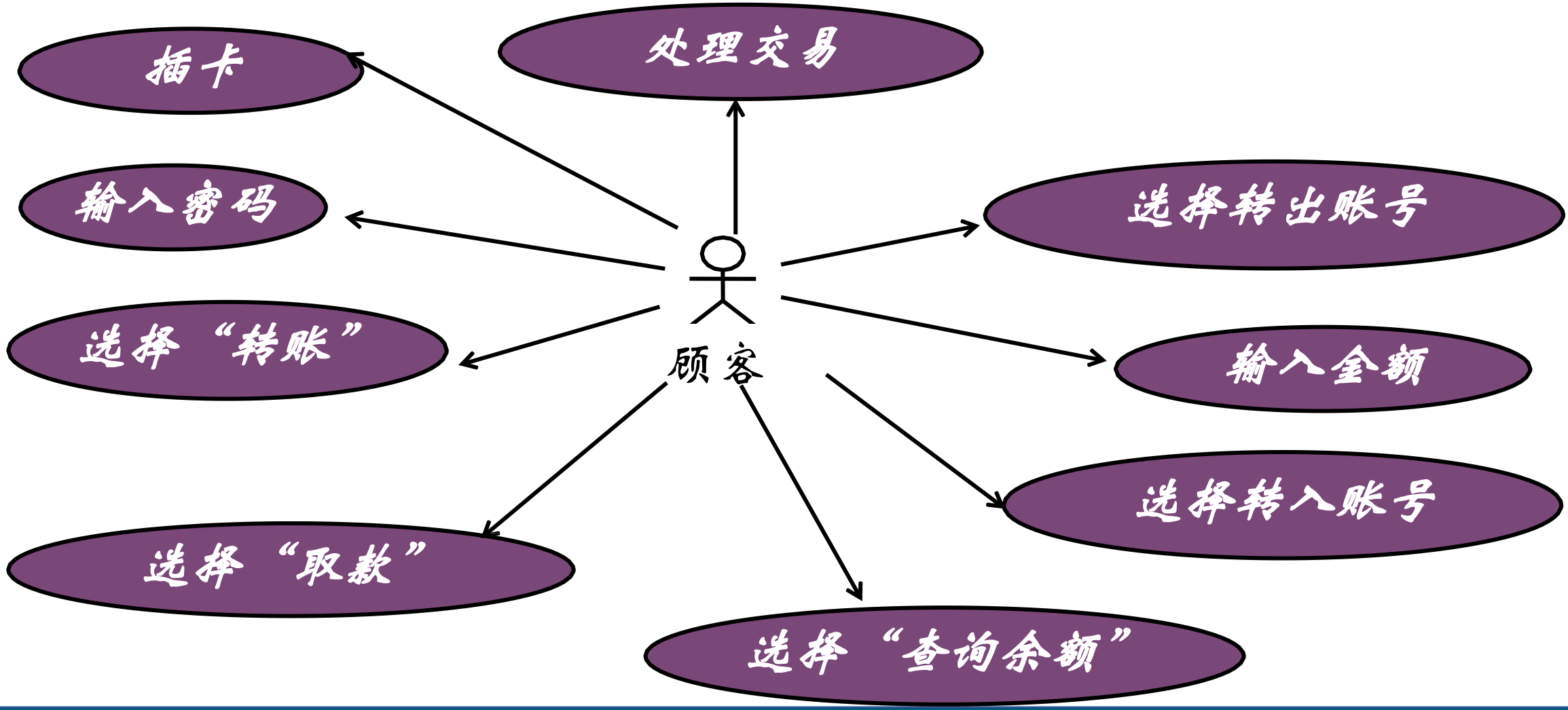


用例规约



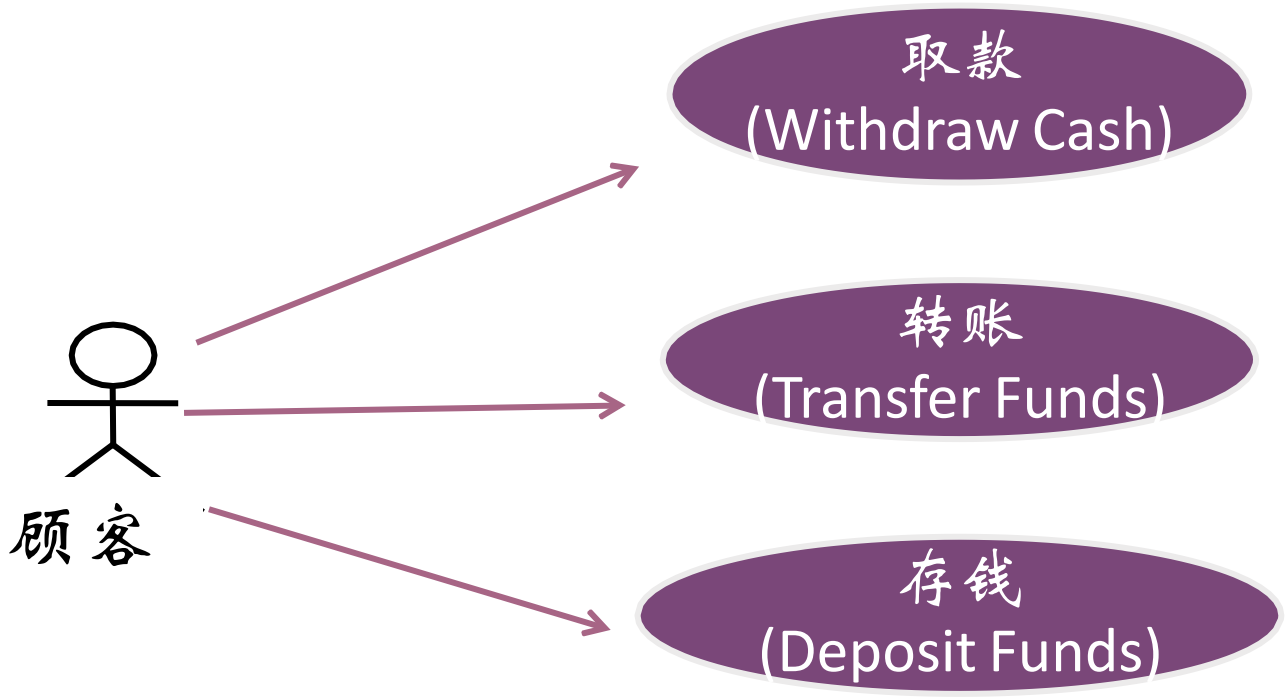
用例名：
参与者及关注点：
主成功场景：
事件1
事件2
……
扩展
前置条件：
后置条件：
……

用例常见错误-功能分解



用例常见错误-功能分解

走出功能分解：正确的用例建模



关于“通讯关联”的几点说明

- **通讯关联表示的是参与者和用例之间的关系：**
  - 箭头表示在这一关系中哪一方是对话的**主动发起者**，箭头所指方是对话的被动接受者；
  - 如果不想强调对话中的主动与被动关系，可以使用不带箭头的关联实线。
  - 通讯关联**不表示**在参与者和用例之间的**信息流**，并且信息流向是双向的，它与通讯关联箭头所指的方向没有关系。

用例方法的优点

- 系统被看作是一个黑箱，并不关心系统内部是如何完成它所提供的功能的。
- 描述了被定义系统有哪些外部使用者(抽象为Actor)、这些使用者与被定义系统发生交互。
- 针对每一参与者，又描述了系统为这些参与者提供了什么样的服务(抽象成为Use Case)、或者说系统是如何被这些参与者使用的。

如何绘制用例模型

- **Step 1：** 确定系统边界
- **Step 2：** 识别并描述参与者**(actor)**；
- **Step 3：** 确定每个参与者目标，识别用例**(use case)** ；
- **Step 4：** 识别参与者与用例之间的通讯关联**(Association)**；
- **Step 5：** 给出每一个用例的详细描述
- **Step 6：** 细化用例模型

Step 1：确定系统边界

- 系统目标
- 系统范围

例：学生成绩管理系统

•目标：

大学？中小学？

•范围：

单机、网络？学籍？课程？

Step 2：识别并描述参与者

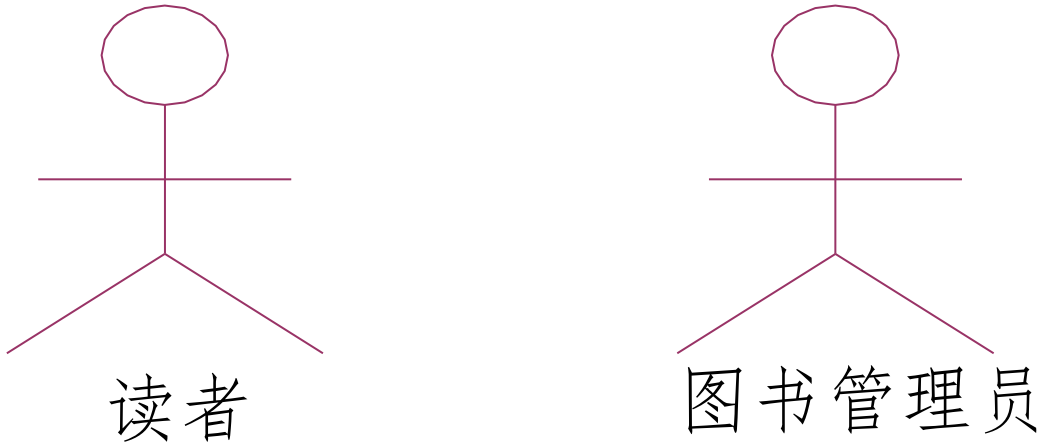
- **通过以下问题来识别Actor：**
  - 谁使用这个系统的功能？
  - 谁从该系统获得信息？
  - 谁向该系统提供信息？
  - 该系统需要访问(读写)那些外部硬件设备？
  - 谁来负责维护和管理这个系统以保证其正常运行？
  - 该系统需要与其他系统进行交互吗？

Step 2：识别并描述参与者

- 问1：对一个图书管理系统来说，有哪些参与者？

Step 2：识别并描述参与者

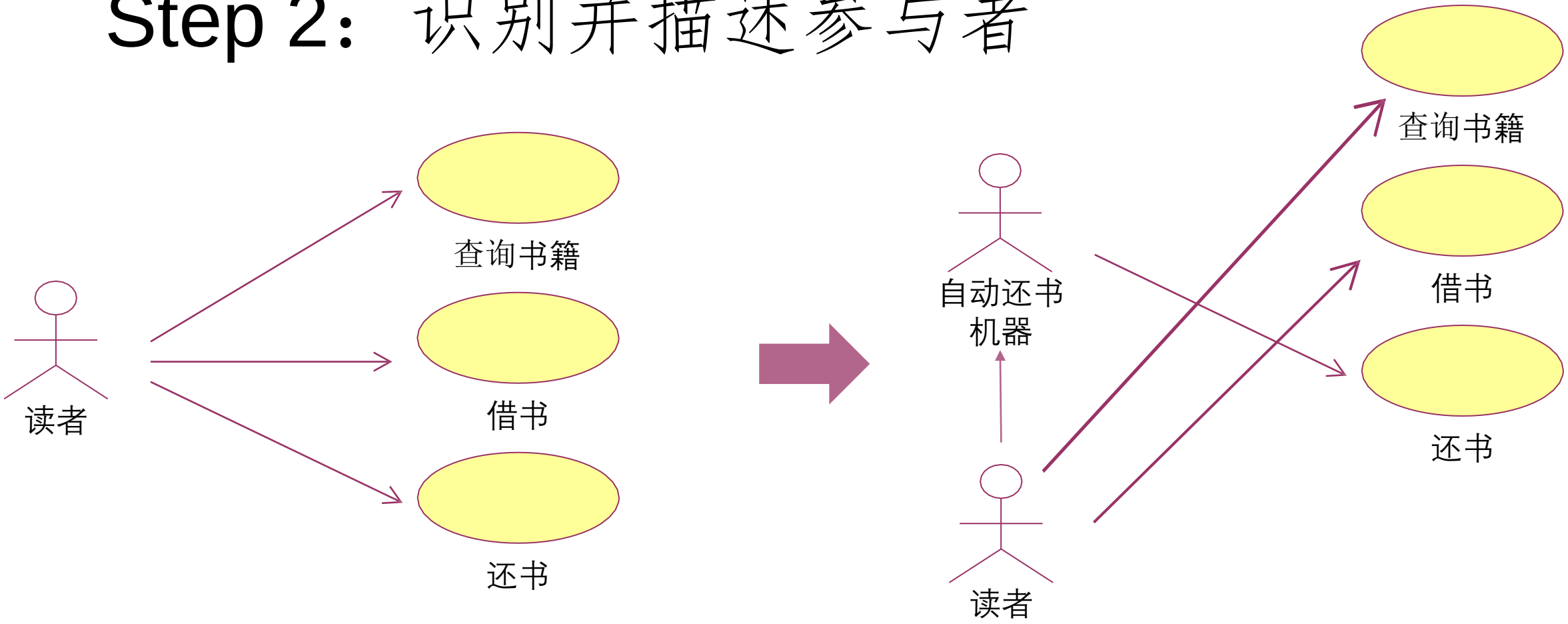
- 问1：对一个图书管理系统来说，有哪些参与者？
  - 读者
  - 图书管理者



Step 2：识别并描述参与者

- 问2：现在图书馆做了升级，增加了一个自动还书机器，请问这个图书管理系统有哪些参与者？

Step 2：识别并描述参与者



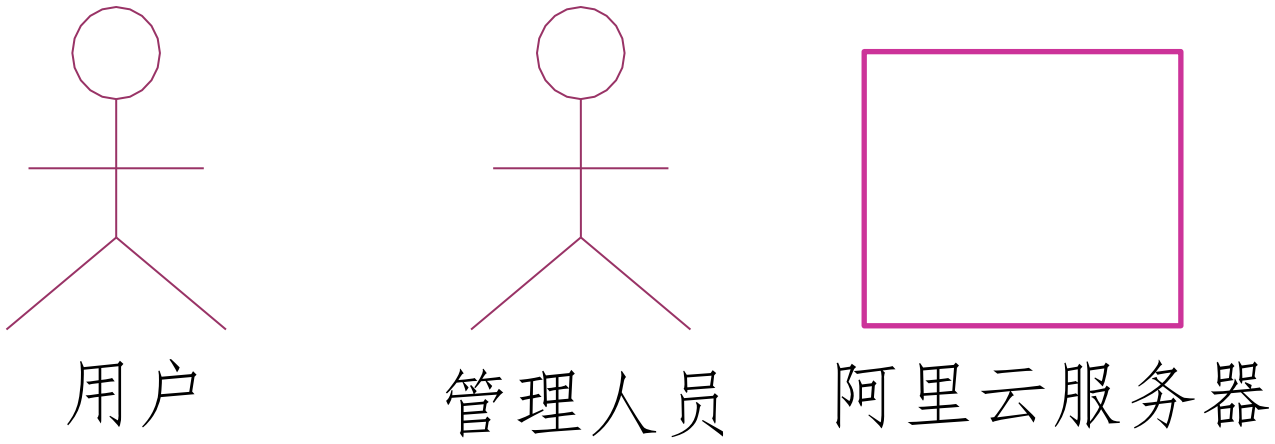
读者自己去图书馆找到书，并刷卡借书
同时他可以在有还书机的地方刷卡还书

Step 2：识别并描述参与者

- 问2：对一个电影可视化系统来说，有哪些参与者？

Step 2：识别并描述参与者

- 问2：对一个电影可视化系统来说，有哪些参与者？
  - 用户
  - 管理人员
  - 阿里云服务器



特殊的参与者：系统时钟

- 有时候需要在系统内部定时的执行一些操作，如检测系统资源使用情况、定期生成统计报表等等；
- 但这些操作并不是由外部的人或系统触发的；
- 对于这种情况，可以抽象出一个系统时钟或定时器参与者，利用该参与者来触发这一类定时操作；
- 从逻辑上，这一参与者应该被理解成是系统外部的，由它来触发系统所提供的用例对话。

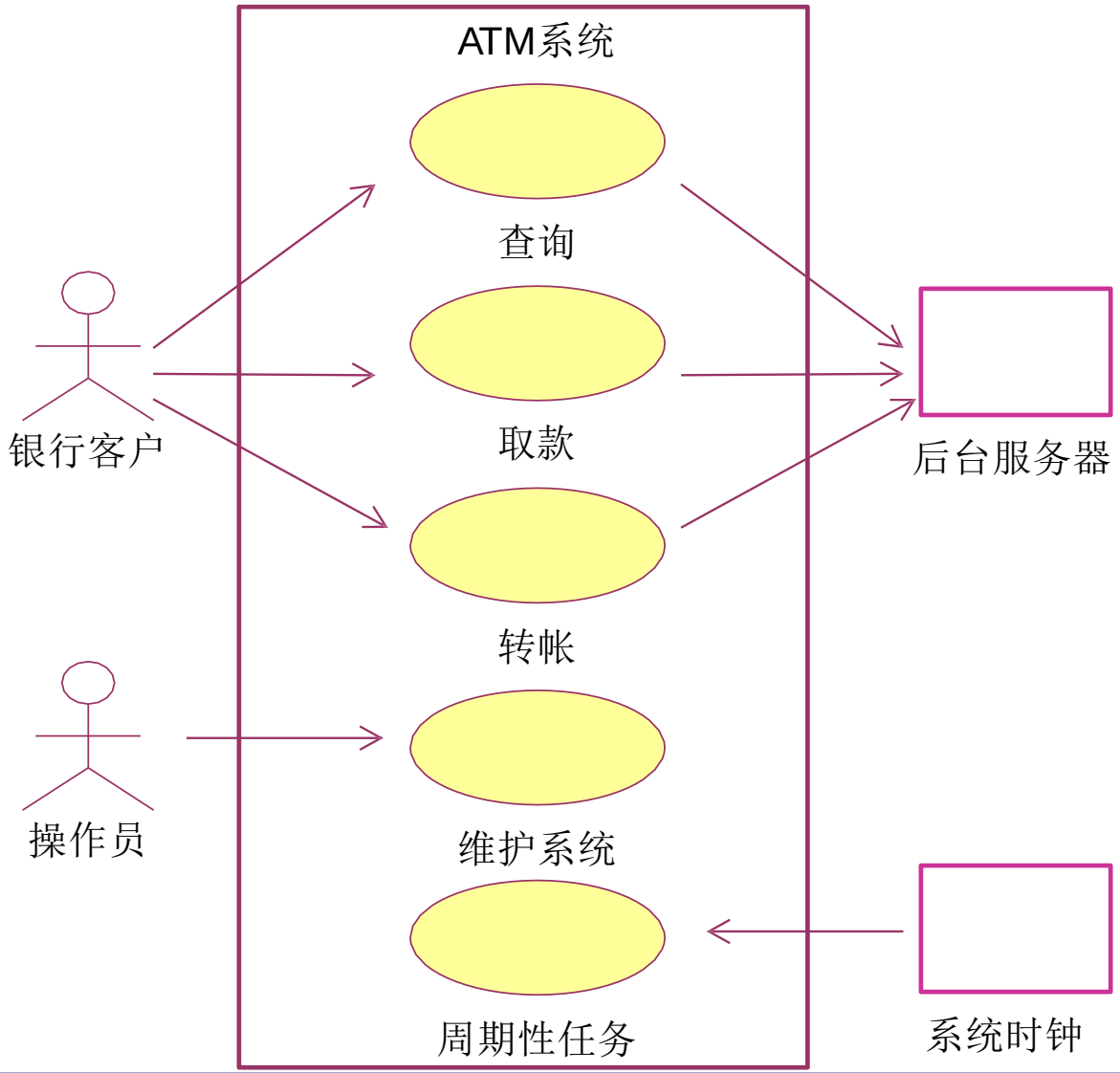
Step 3：识别用例(use case)

- 找到参与者之后，据此来确定系统的用例。主要分析各参与者目标，需要系统提供什么样的服务，或者说参与者是如何使用系统的。
- 寻找用例可以从以下问题入手(针对每一个参与者)：
  - 参与者使用该系统执行什么任务？
  - 参与者是否会在系统中创建、修改、删除、访问、存储数据？如果是的话，参与者又是如何来完成这些操作的？
  - 参与者是否会将外部的某些事件通知给该系统？
  - 系统是否会将内部的某些事件通知该参与者？

Step 3：识别用例(use case)

- **例1：对图书馆管理系统来说，有哪些参与者和用例？**
  - 图书管理员
    - 管理读者信息
    - 管理图书信息
    - 登记借书
    - 登记还书
  - 普通读者：
    - 预订图书
    - 取消预订
    - 查询浏览图书信息
- **例2：对ATM系统来说，有哪些参与者和用例？**
  - 银行客户
    - 查询
    - 取款
    - 转账
  - ATM维护人员
    - 维护系统
  - 后台服务器
    - 周期性操作

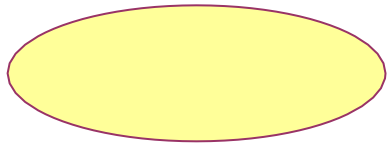
Step 4：识别参与者与用例之间的通讯关联



Step 5：给出用例的详细描述

Name of the Use Case (用例的名字)

Description (描述)
Actor(s) (参与者)
Flow of events(事件流)
  Basic flow(常规流)
    Event 1 (事件)
    Event 2
  ……
  Alternate flow(备选流)
Pre-conditions (前置条件)
Post-conditions (后置条件)
……



Use case

Step 5：给出用例的详细描述

- **单纯的用例图并不能描述完整的信息，需要用文字描述不能反映在图形上的信息。**

编写用例文本的准则

- 以无用户界面约束的本质风格编写用例
- 编写简洁的用例
- 编写黑盒用例
- 采用参与者和参与者目标的视角

用例名：
参与者及关注点：
主成功场景：
　事件1
　事件2
……
扩展
前置条件：
后置条件：
……

事件流（场景）

退货处理

主要成功场景（常规流）：

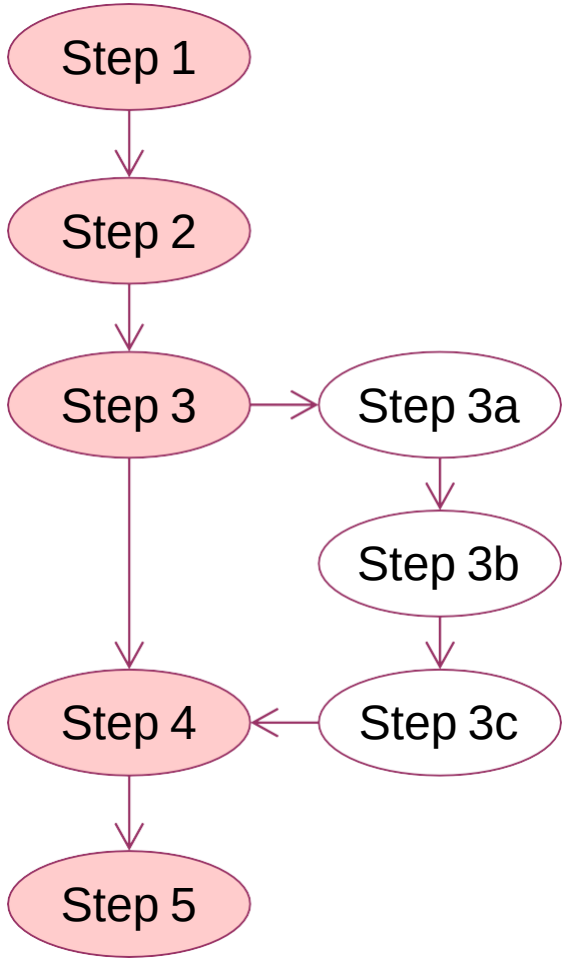
- 客户带着要退的货物到达收款处，出纳员使用POS系统记录每一个要退货的货物,...

候选场景（扩展流）：

- 若信用验证失败，通知客户并要求使用其他付款方法
- 若系统检测到与外界计税系统通信失败，...

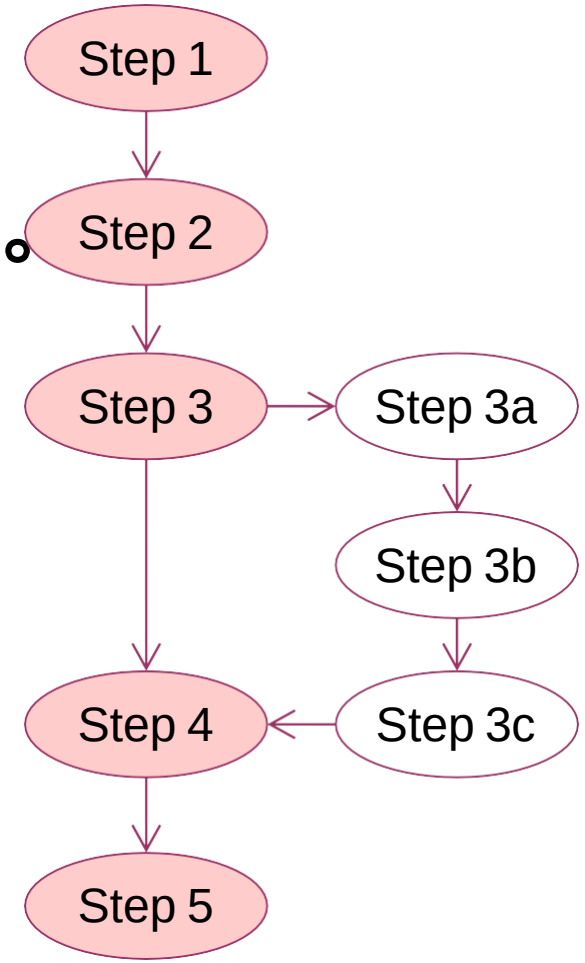
常规事件流（场景）

- 每一个步骤都需要用数字编号以清楚地标明步骤的先后顺序
- 用一句简短的标题来概括每一步骤的主要内容
- 对每一步骤，从正反两个方向来描述
  - 参与者向系统提交了什么信息
  - 对此系统有什么样的响应



扩展事件流（场景）

- **扩展流的描述格式可以与基本流的格式一致，也需要编号并以标题概述其内容。**
  - 起点：该扩展流从事件流的哪一步开始；
  - 条件：在什么条件下会触发该扩展流；
  - 动作：系统在该扩展流下会采取哪些动作；
  - 恢复：该扩展流结束之后，该用例应如何继续执行。



例

用例：登记借书

1. 目标：

本用例允许图书管理员登记普通读者的借书记录

2 事件流：

1. 常规流程

当读者希望借书、图书管理员准备登记有关的借书记录时，本用例开始执行。

(1) 系统要求管理员输入读者的注册号和所借图书号；
(2) 图书管理员输入信息后，系统产生一个唯一的借书记录号；
(3) 系统显示新生成的借书记录；
(4) 图书管理员确认后，系统增加一个新的借书记录

2. 扩展流程

(1) 读者没有注册

在主流程中，如果系统没有读者的注册信息，系统将显示错误信息，用例结束；

(2) 所借图书不存在

在主流程中，如果所借图书已被借出或者系统中无该图书，系统将显示错误信息，用例结束。

3 前置条件：用例开始前，图书管理员必须在系统登录成功；

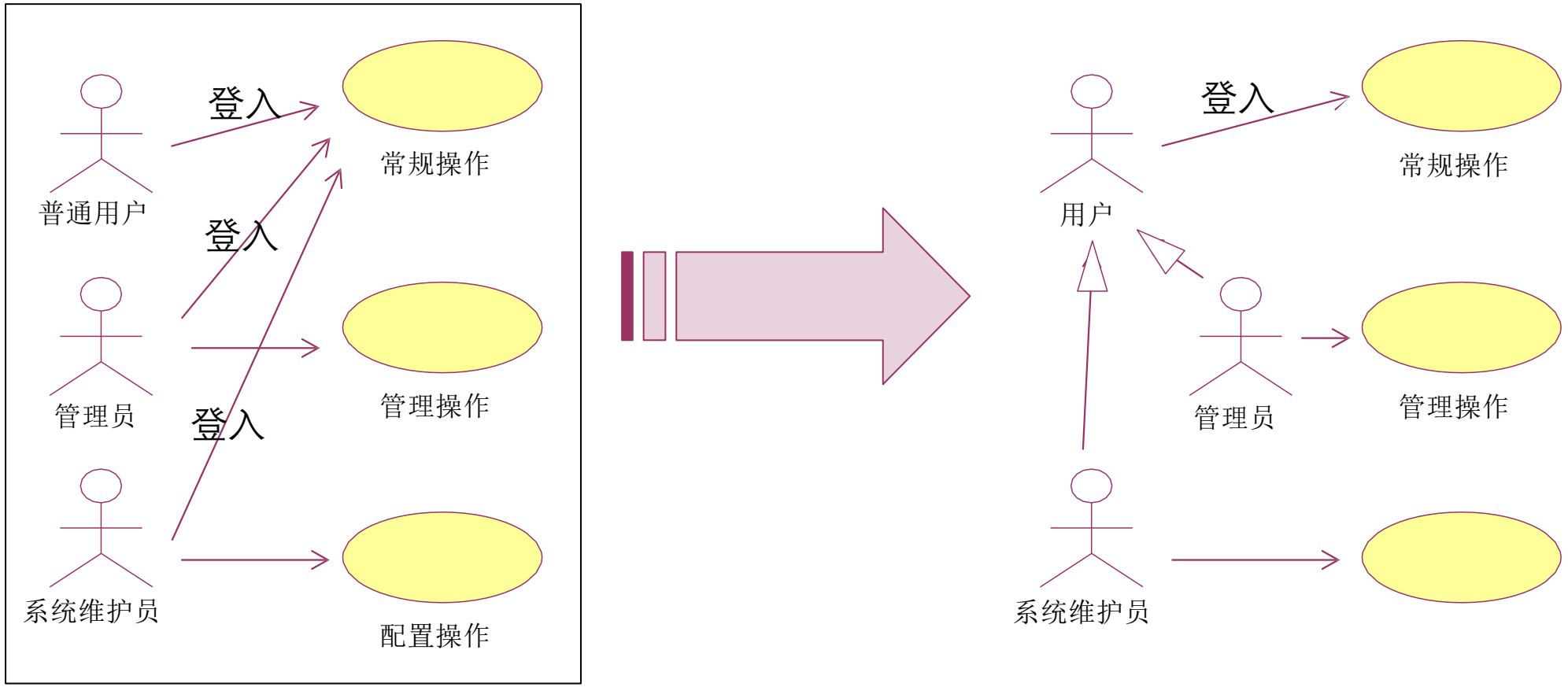
4 后置条件：如果用例执行成功，该读者的借书记录被更新，否则，系统状态不变。

Step 6：细化用例模型

1. **在一般的用例图中，只需表述参与者和用例之间的通讯关联，除此之外，还可以描述：**
   - 参与者与参与者，用例与用例之间的泛化(generalization)
   - 用例和用例之间的包含(include)
   - 用例和用例之间的扩展(extend)
   - 利用这些关系来调整已有的用例模型，把一些公共的信息抽取出来复用，使得用例模型更易于维护。
2. **根据用例描述绘制用例图**
3. **补充非功能性需求**

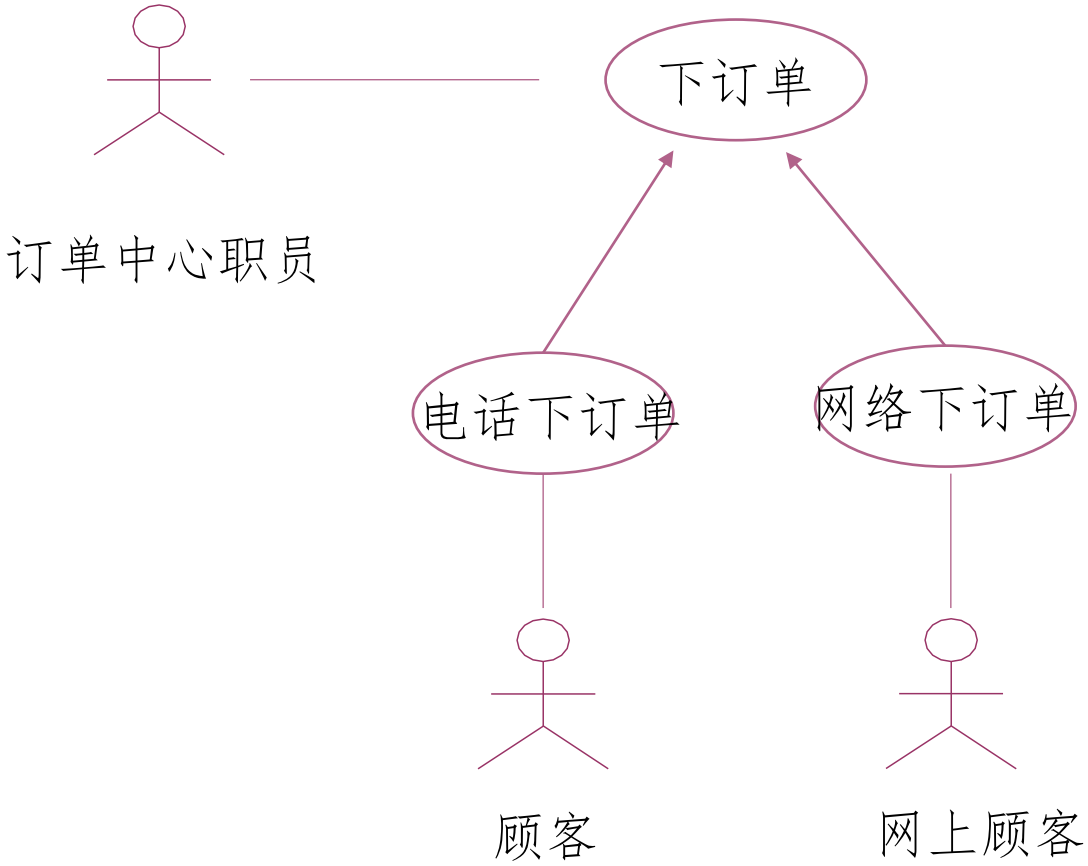
泛化(Generalization)

- 参与者之间可以有泛化**(Generalization)**关系。

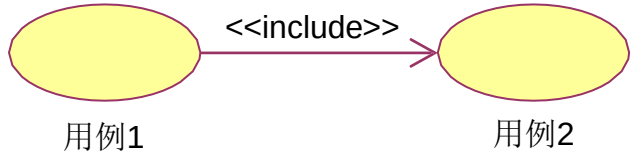


泛化(Generalization)

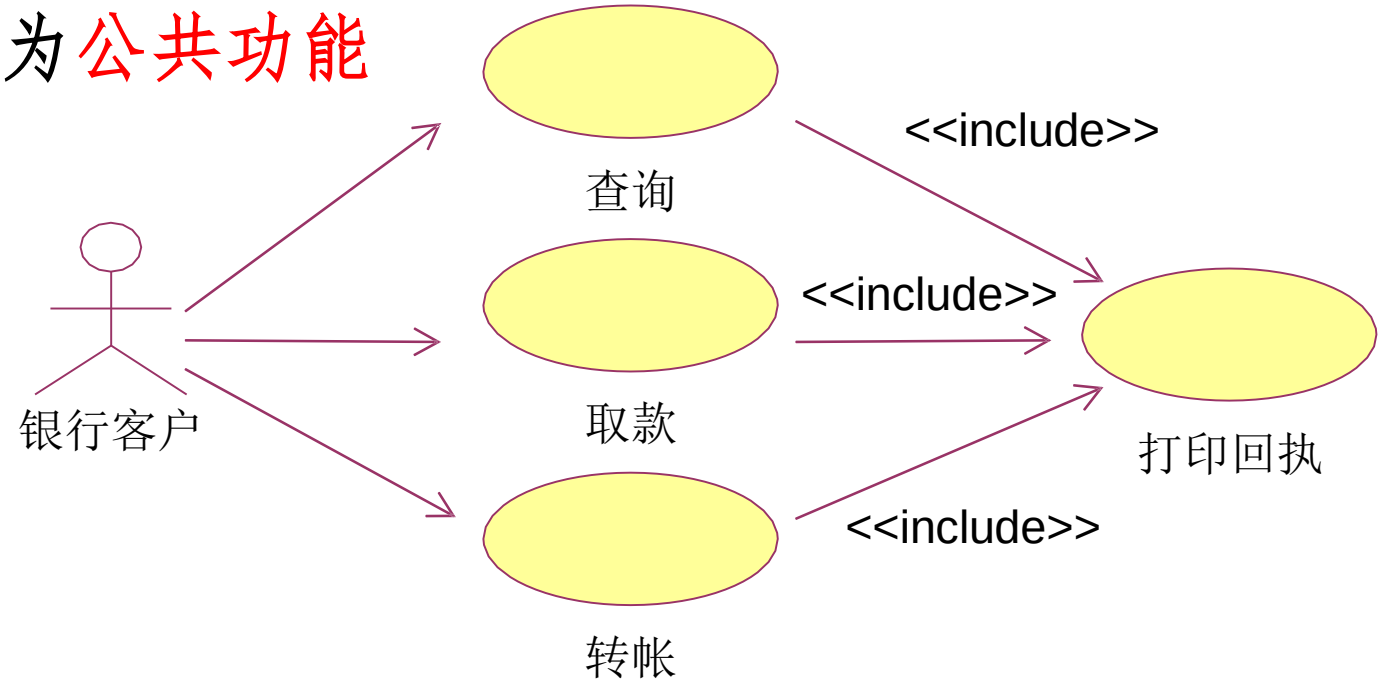
- Case之间也有泛化**(Generalization)**关系。



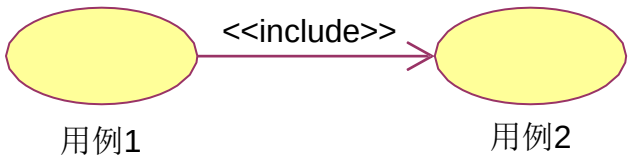
包含（Include）



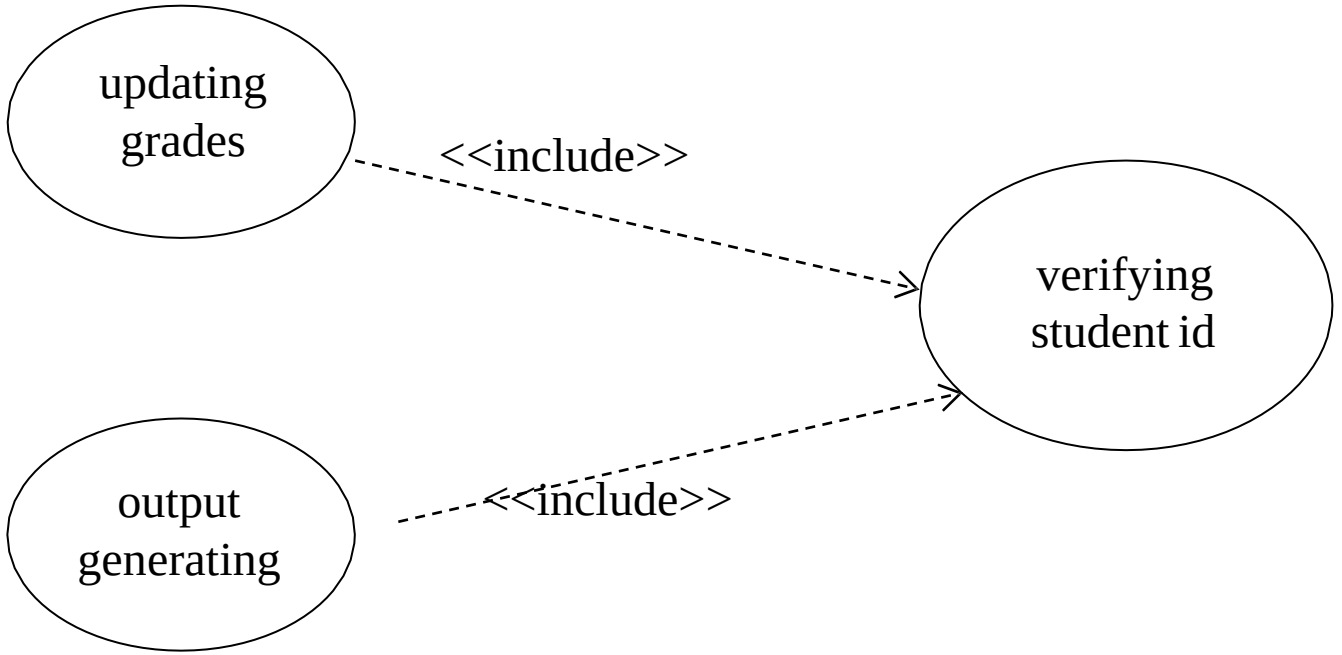
- “包含关系”是通过在关联关系上加入<<include>>标记来表示；
- 语义：用例1会用到用例2（无条件执行），用例2的事件流将被插入到用例1的事件流中
- 一般表示为公共功能



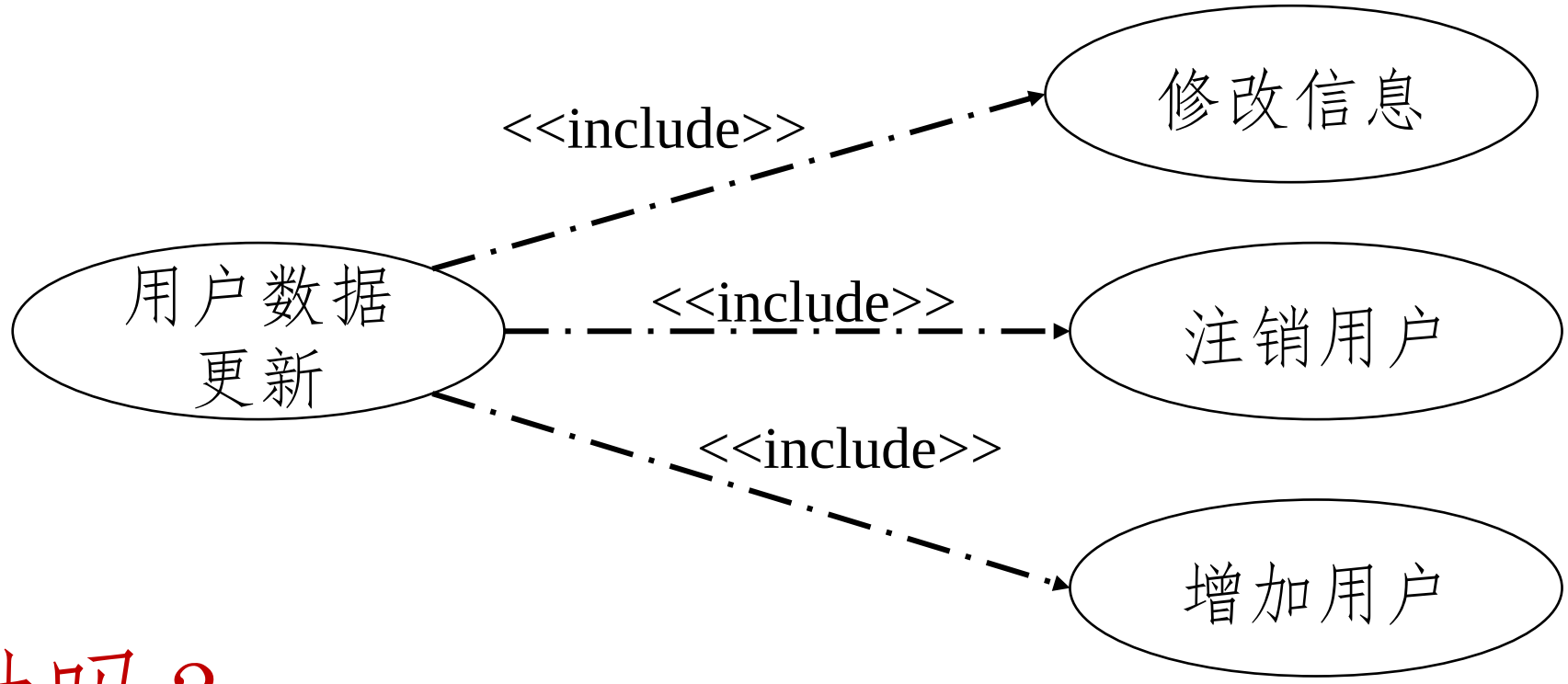
包含（Include）



- 通过将公共行为放在用例中，我们可以避免多次描述相同的事件流。

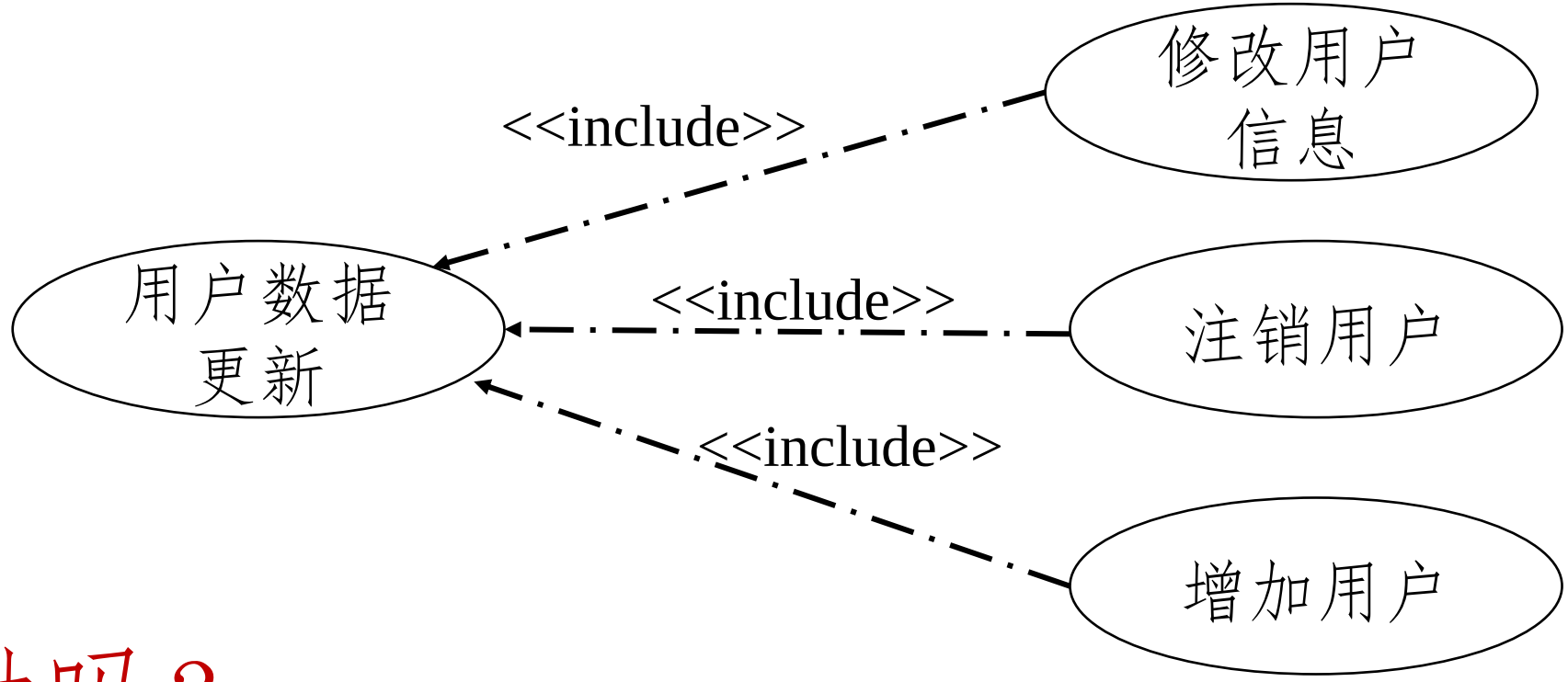


包含（Include）



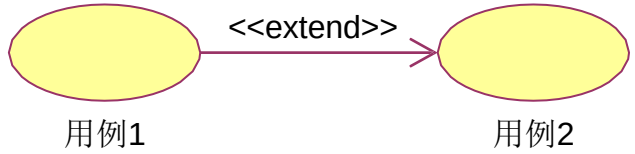
对吗？

包含（Include）

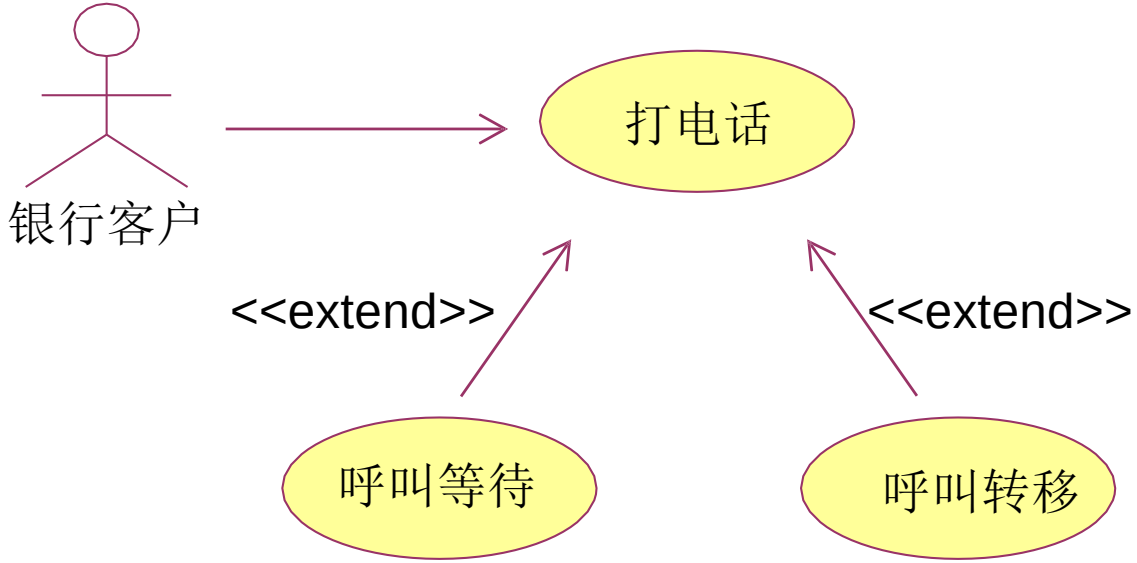


对吗？

扩展(extend)

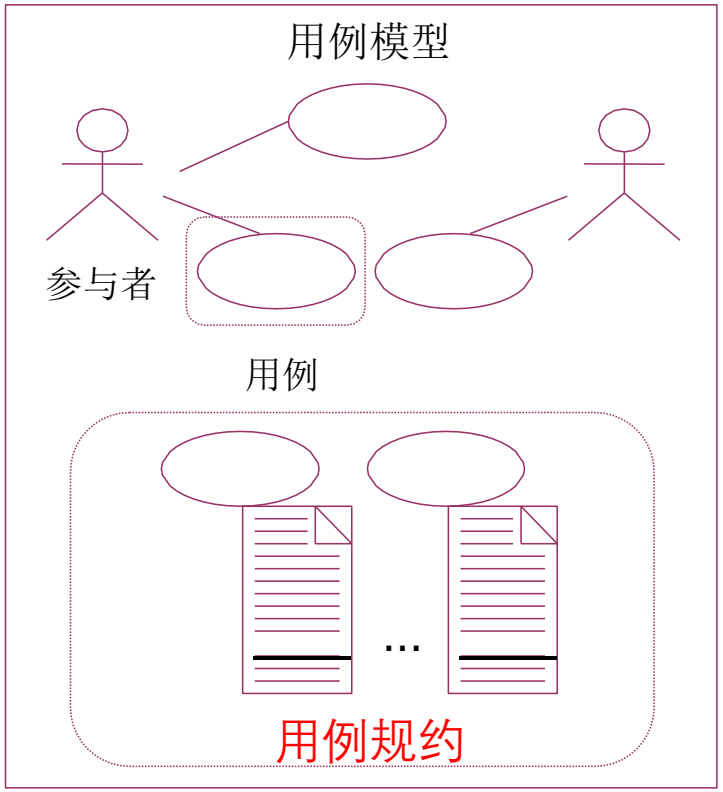


- “扩展关系”是通过在关联关系上加入**<<extend>>**标记来表示；
- 语义：用例**2**在某些特定情况下（有条件执行）会用到用例**1**，此时，用例**1**的事件流将被插入到用例**2**的事件流中。
- 一般表示为异常功能，大多是扩展流程



最终用例模型的提交物

- 1 用例模型
- 2 每个用例的详细描述
- 3 术语表：所用到的术语说明
- 4 补充规约：非功能性需求的说明



图书管理系统

建立图书信息管理系统，系统要求实现以下功能：

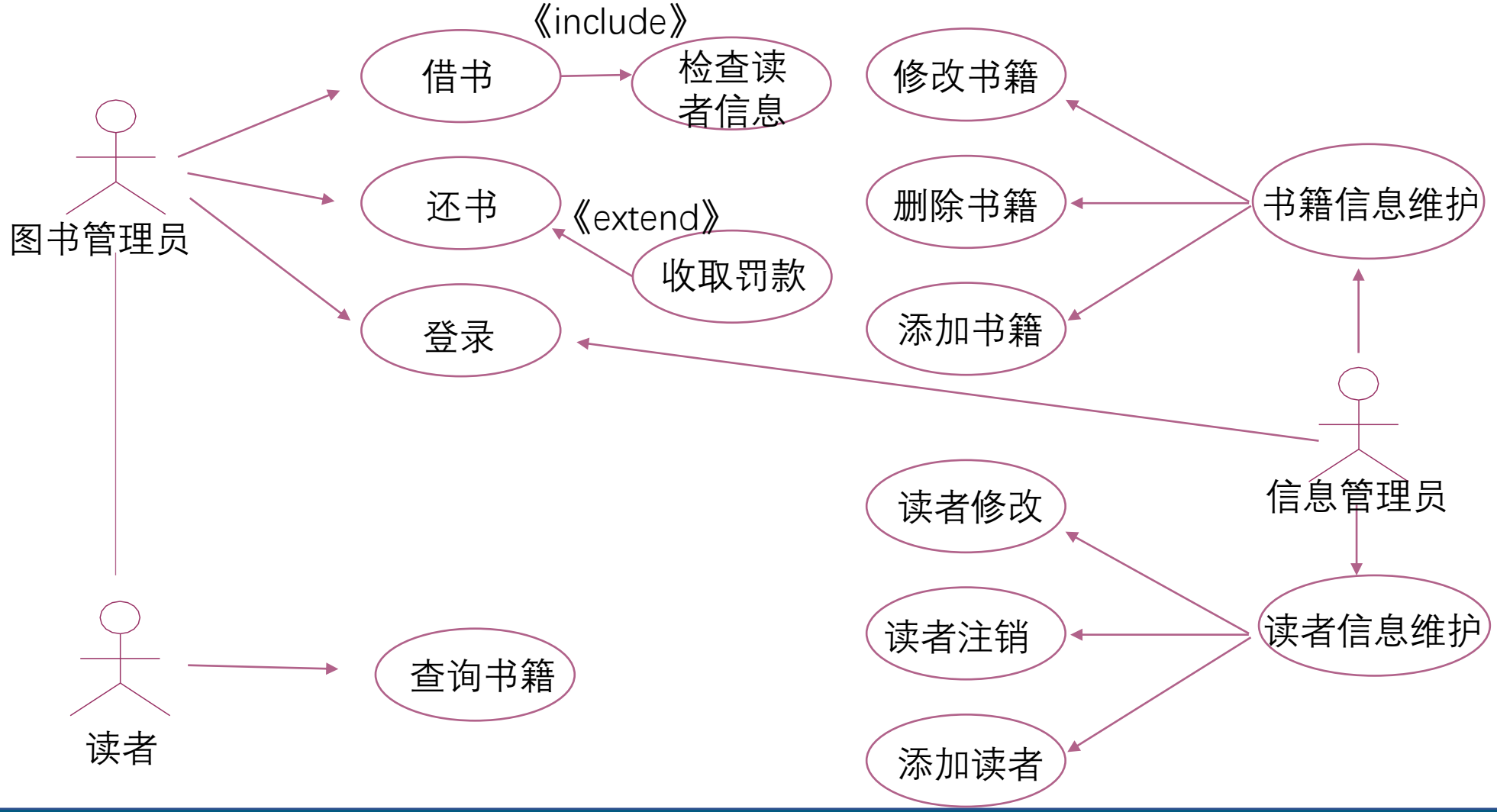
- ✓用户管理功能，包括读者信息的录入、修改、更新以及登录
- ✓书籍管理功能，如书籍的添加、修改、更新、删除等数据维护功能，还可以根据读者借阅书籍的要求随时更新图书馆的书籍数据库
- ✓书籍的借阅、归还管理、如借还进行详细登记，更新书籍数据库。同时提供图书预定功能
- ✓信息查询功能，如图书信息查询、用户借书、还书信息查询、书籍库存情况查询等

请根据以上描述，确定执行者及用例，建立系统的用例模型。

图书管理系统

- 参与者：管理员和读者；管理员=后台信息维护管理员+图书管理员
- 读者可以查询书籍获得书籍信息。
- 读者可以通过管理员来进行图书的借、还操作
- 图书管理员可以执行借书和还书操作，需要用管理员账号进行登录。
- 信息维护管理员可以进行书籍信息维护和读者信息维护，需要用管理员账号进行登录。
- 书籍信息维护主要包括添加书籍、修改书籍和删除书籍
- 读者信息维护主要包括添加新读者、读者信息修改和读者注销

图书管理系统用例图


《include》
借书
检查读者信息
还书
《extend》
收取罚款
登录
图书管理员
修改书籍
删除书籍
添加书籍
书籍信息维护
信息管理员
读者修改
读者注销
添加读者
读者信息维护
查询书籍
读者

需求建模

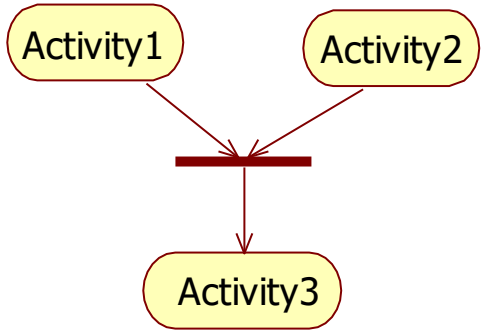
- 基于场景的方法
  - 用户故事（User Story）
  - 用例图（User Case Diagram）
  - 活动图（Activity Diagram）
- 基于类的方法
- 基于模式的方法

活动图

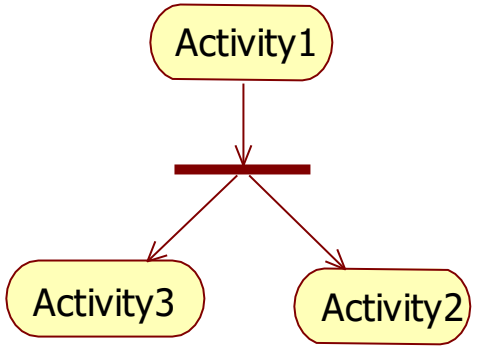
- **什么是活动图？**
  - 显示了组成复杂过程的步骤序列，例如算法或工作流
  - 活动图用于详细描述用例
  - 是状态图的一个变种
  - 活动图的目的是描述动作及动作的结果
  - 活动图中的动作可以放在“泳道中”，泳道聚合一组活动，通常根据活动的功能来组合活动。

活动图图形元素

初始　　终止　　判断

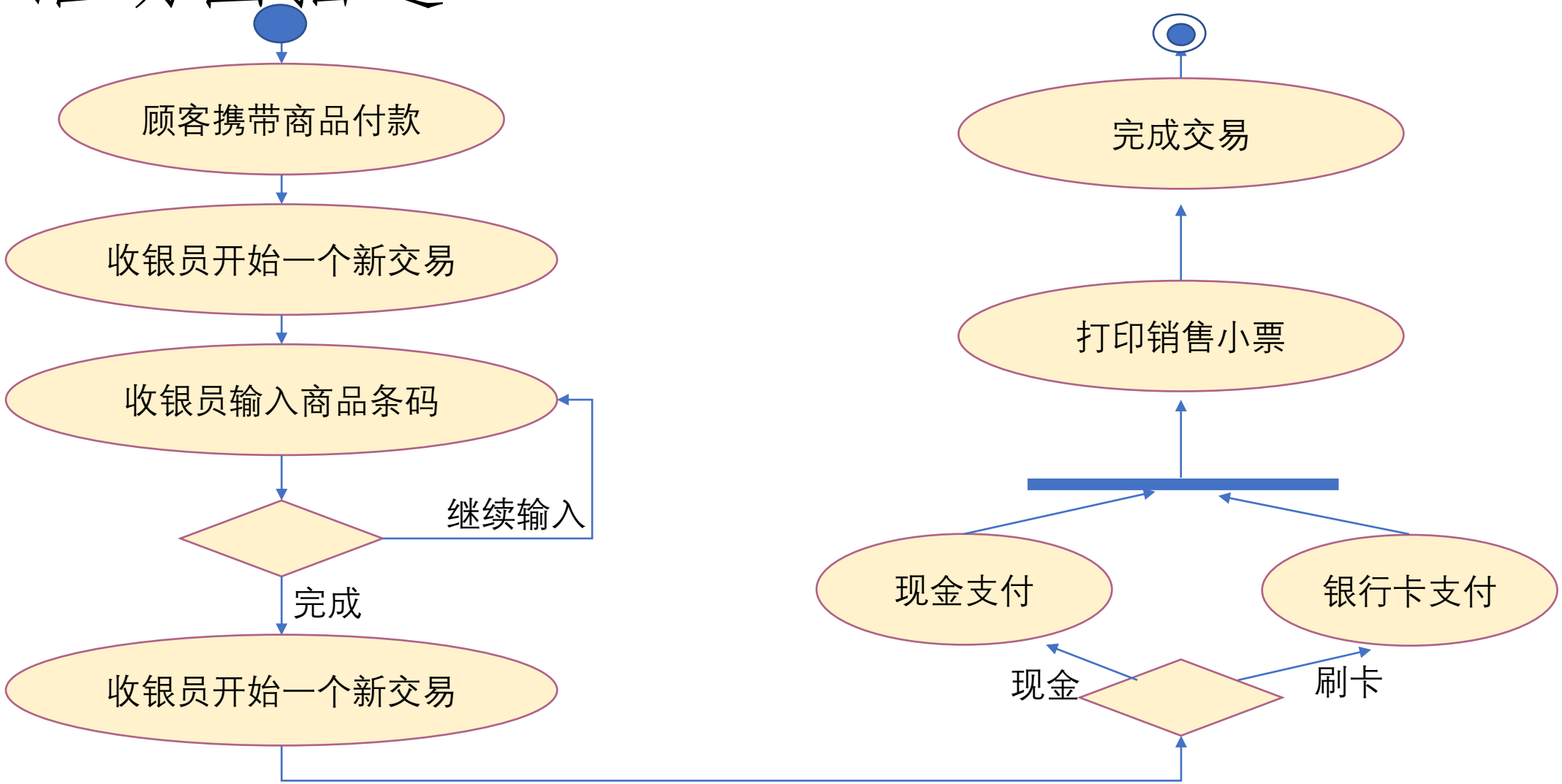


并发合并

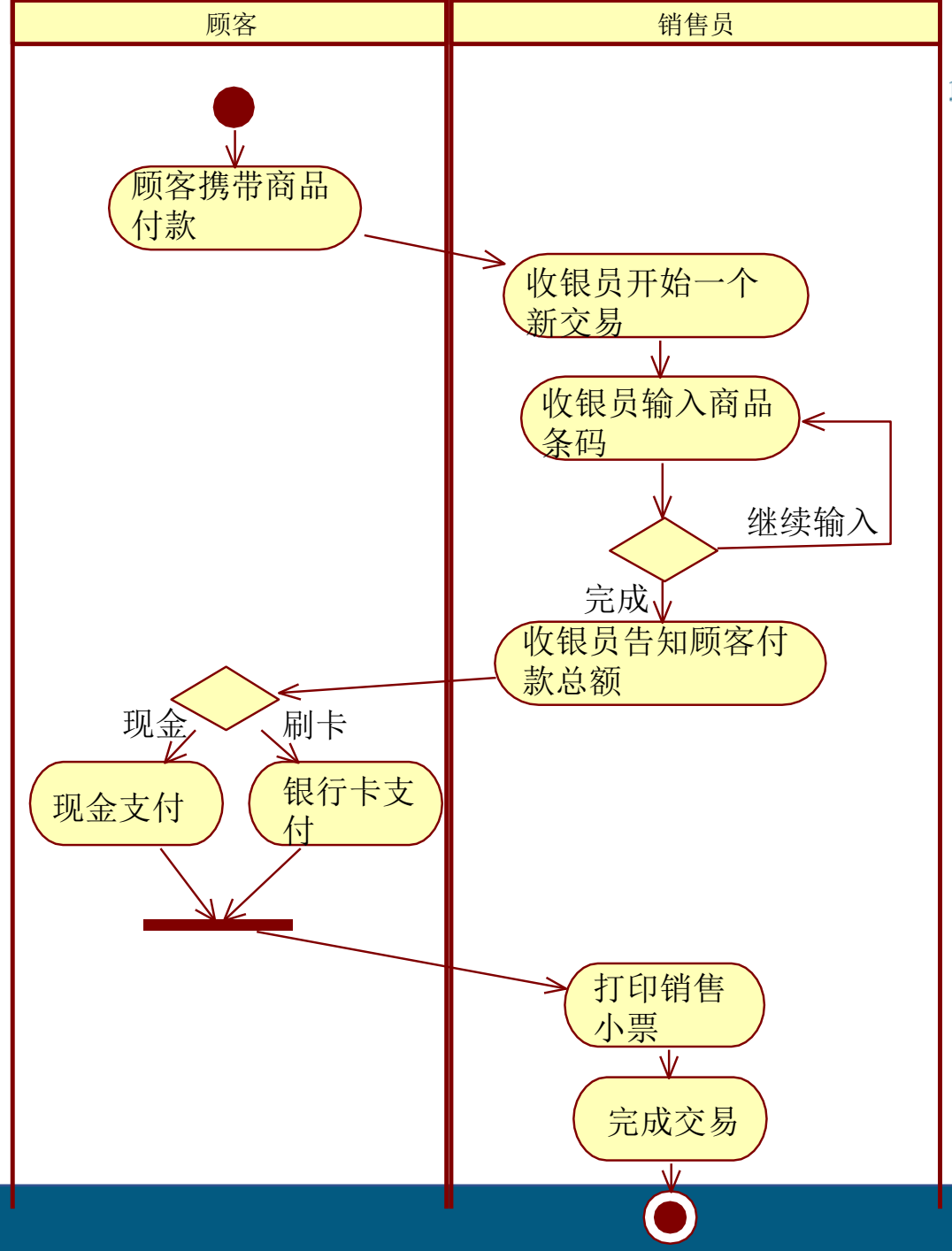


并发分叉

活动图描述


顾客携带商品付款
收银员开始一个新交易
收银员输入商品条码
继续输入
完成
收银员开始一个新交易
完成交易
打印销售小票
现金支付
银行卡支付
现金
刷卡

泳道图


顾客
销售员
顾客携带商品付款
收银员开始一个新交易
收银员输入商品条码
继续输入
完成
收银员告知顾客付款总额
现金
刷卡
现金支付
银行卡支付
打印销售小票
完成交易

小结

- 需求获取的挑战和途径
- 需求建模
  - 基于场景的方法
  - 基于类的方法
  - 基于模式的方法