5-2-1 用户故事与用例建模

2019年5月8日 22:04

1 “用户故事” (User Story)

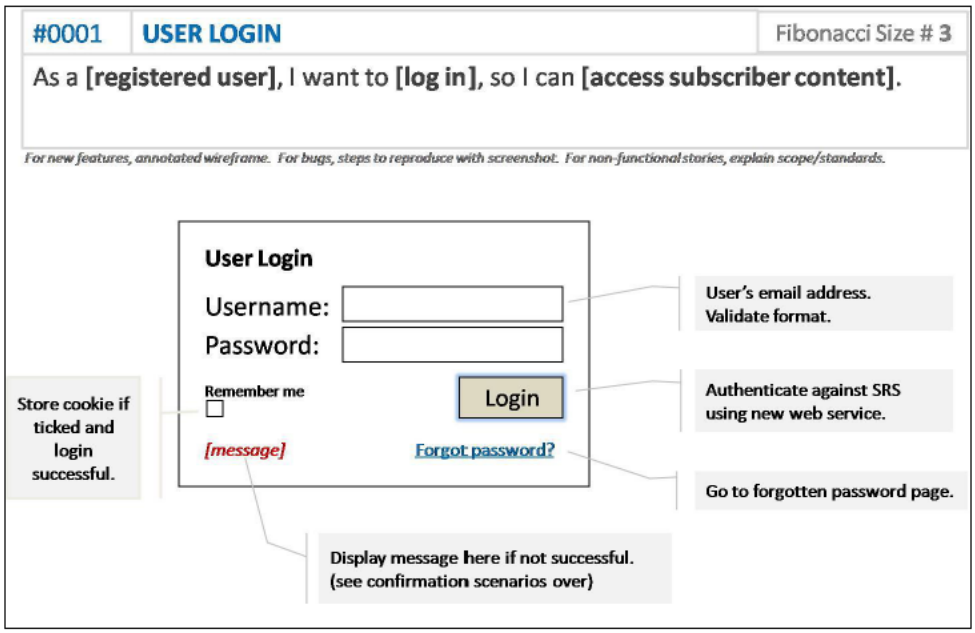
何谓“用户故事”？

- 用户故事： A concise, written description of a piece of functionality that will be valuable to a user (or owner) of the software
- 从用户的角度来描述用户渴望得到的功能：角色(谁要使用这个功能)、目标/活动(需要完成什么样的功能)、商业价值(为什么需要这个功能，这个功能带来什么样的价值)。
- 三个组成部分：
  - 简要的文本陈述
  - 用户如何与系统交互
  - 如何验证和测试

用户故事的描述

- As a[**user role**] I want to [**goal**] so I can [**reason**]
- 作为一个<**角色**>,我想要<**活动**>,以便于<**商业价值**>
- Who(user role)
- What(goal)
- Why(reason)
- 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“**赞助商了解我的网站会给他们带来什么收益**”。

用户故事卡的正面:Conversation

- 

用户故事卡的反面：Confirmation

- **Confirmation**

  1. Success – valid user logged in and referred to home page.
     a. 'Remember me' ticked – store cookie / automatic login next time.
     b. 'Remember me' not ticked – force login next time.

  2. Failure – display message:
     a) "Email address in wrong format"
     b) "Unrecognised user name, please try again"
     c) "Incorrect password, please try again"
     d) "Service unavailable, please try again"
     e) Account has expired – refer to account renewal sales page.

好的用户故事应具备的特征：INVEST

- Independent尽可能独立；
- Negotiable可讨论的。它不是一个合同，没有详细的规约，后续开发阶段可以不断协商和改进；
- Valuable对用户/客户有价值的，以用户可理解的语言书写，是系统的"特性"而非"开发任务"；
- Estimatable其工作量可以估计；
- Small小，而不是大；
- Testable可测试的、可验证的。

分割用户故事(1)

- 当故事非常大时，很难对它进行估计。如果故事预计在N次迭代后才进行，那么大的故事很正常。但如果估计预计在接下来的迭代中进行，那么我们就可能会对大的故事进行拆分。可参考如下准则：
  - 按照用户故事所支持数据的边界来分割大型用户故事（例如导入GBQ文件、Excel

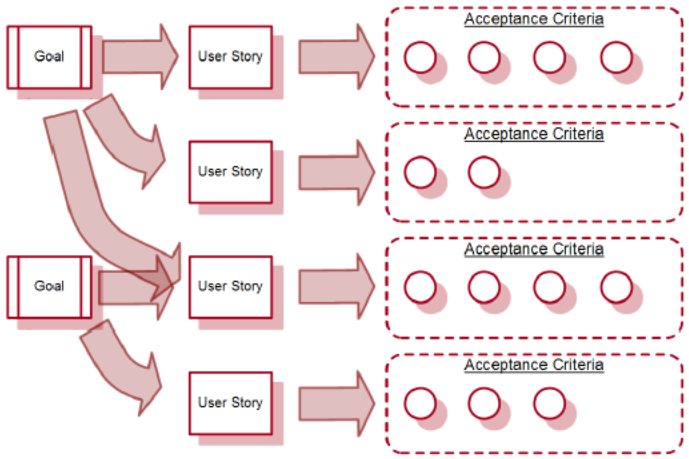
等）。
    - 从主用户故事中除去对例外或错误条件的处理（相当于用户的基本路径和扩展路径），从而把一个大型用户故事变小许多。
    - 按照操作边界分割，把大型用户故事分割成独立的建立、读取、更新和删除操作（例如预算二次导入，或者新增时需要向导、规则而比较复杂时也可以单独成一个故事来描述）。
    - 考虑去除横切考虑（例如安全处理、日志记录、错误处理等），为用户故事建立两个版本：一个具备对横切考虑的支持，另一个不具备这种支持。
    - 考虑功能性需求和非功能性需求隔离到不同的用户故事，从而分割大型用户故事（性能）。
- 在拆分故事时，我们有时也需要考虑组合故事的场景，如把bug列入产品backlog时，可以把多个类似的bug组合成一个故事。

优秀的用户故事准则

- 试着让故事的大小能够在使用后让用户感到可以去喝杯咖啡休息一下
- 不要让故事过早涉及用户界面；
- 实际编写故事时，要包括用户角色；
- 用主动语态编写故事；
- 为单个用户编写故事；
- 让客户编写故事，而不是开发人员；
- 用户故事要简短，它们只是提醒开发人员和客户进行对话；
- 不要给故事卡添加顺序号。

用户故事支持验收测试

- 在每个用户故事背后，列出未来用户测试时可能使用的“测试用例”，作为该故事是否已被完整实现的基本标准。
- 对应于敏捷开发的一个基本思想：在写代码之前先写测试(TestDrivenDevelopment,TDD)

- 

用户故事支持敏捷迭代计划

- 针对识别出的每一个故事，使用StoryPoint估算其工作量；
  - 故事点：一个达到共识的基本时间单位，例如1天；
  - 使用预定的值：1/2、1、2、3、5、8、13、20、40、80；
- 团队成员分别估计(而不是由项目经理一人决定)，差异较大时面对面讨论，发现分歧，形成共识；
- 形成估算清单。

评定用户故事的优先级

- 对用户故事排列优先级：最简单的方法就是问问客户最希望在下一个迭代中最想看到的是哪一些功能。可以从以下几个因素来考虑：
  - 获取这些功能带来的经济价值，价值越高的优先级越高。
  - 开发成本带来的影响。例如可能2个月后由于使用新技术只需要2周，而现在做需要2个月，这时可以考虑把优先级放低一些。
  - 获取新知识的重要性。在开发中会不断的产生一些项目和产品的新知识，及早了解和开发这些新知识可以减少不确定性，所以这类功能优先级会高些。
  - 故事之间会存在依赖关系，这时候被依赖的优先级会更高，需要先完成
  - 开发这些功能所减少的风险。在开发过程中，会出现进度风险、成本风险、技术风险等，对于风险越高价值越大的我们需要首先处理，对风险高价值低的要尽量避免，可以通过以下图查看确定功能优先级时综合考虑风险和价值的关系。

用户故事支持敏捷迭代计划

- 对用户故事排列优先级；
- 安排责任人；
- 汇聚为迭代计划并发布；

- 开发过程中监控进度；

用户故事举例

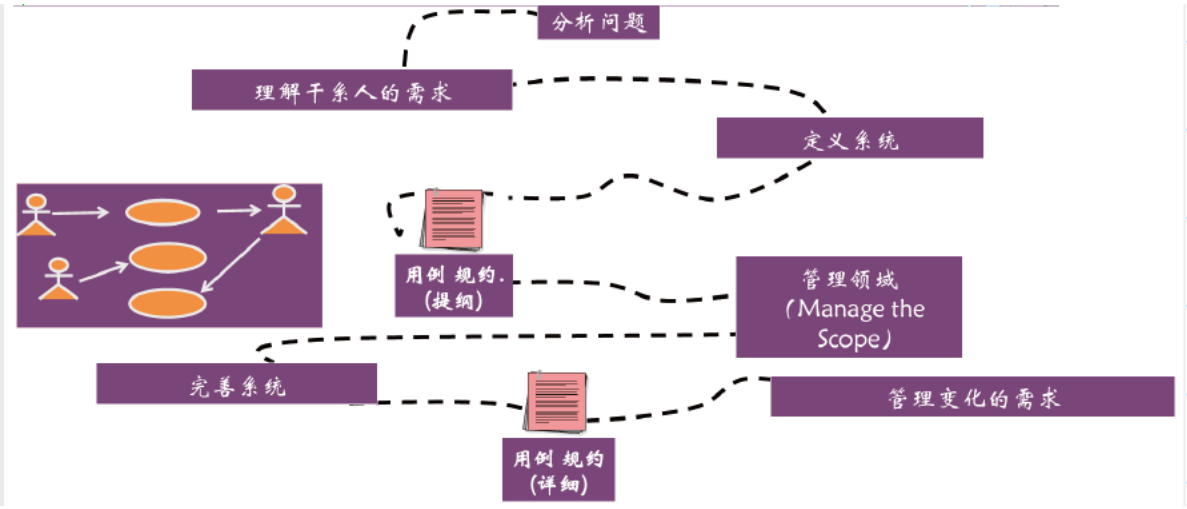
优先级	名称	用户故事描述	估算
1	浏览商品	作为一名顾客想购买商品而不确定型号时，我希望能浏览网站在售的商品，按照①商品类型和②价格范围进行过滤。	
2	搜索商品	作为一名顾客在查找某种商品时，我希望能进行不限格式的文本搜索例如按照短语或关键字。	
3	注册账户	作为一名新顾客，我希望注册并设置一个帐户，包括用户名、密码、信用卡和送货信息等。	
4	维护购物车	作为一名顾客，我希望能将指定商品放入购物车（稍后购买）、查看我的购物车内的商品以及移除我不想要的物品。	
5	结账	作为一名顾客，我希望能完成我购物车内所有商品的购买过程。	
6	编辑商品规格	作为一名工作人员，我希望能够添加和编辑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

说明

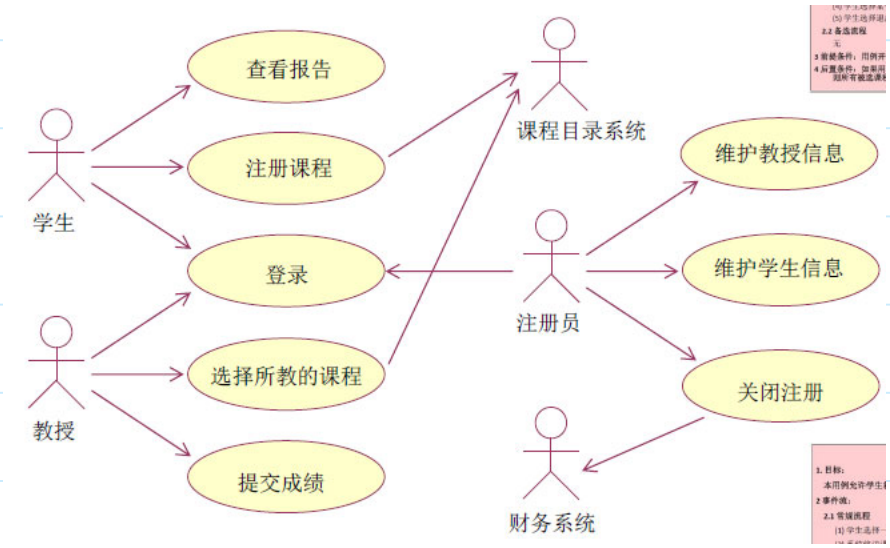
- 在敏捷开发中，需求表述为一组“用户故事”；
- 在传统的OO分析与设计方法中，需求被表述为一组“用例”。
- 二者都是对需求的陈述形式，都是尽量从业务人员和开发人员双方面角度阐述；
- 区别在于大小不同、具体形式不同。
- 对初学者来说，可以将二者看作等价之物，在实践中汇集二者的优点

2 面向对象方法中的“用例”(UseCase)

用例在需求管理过程中的作用



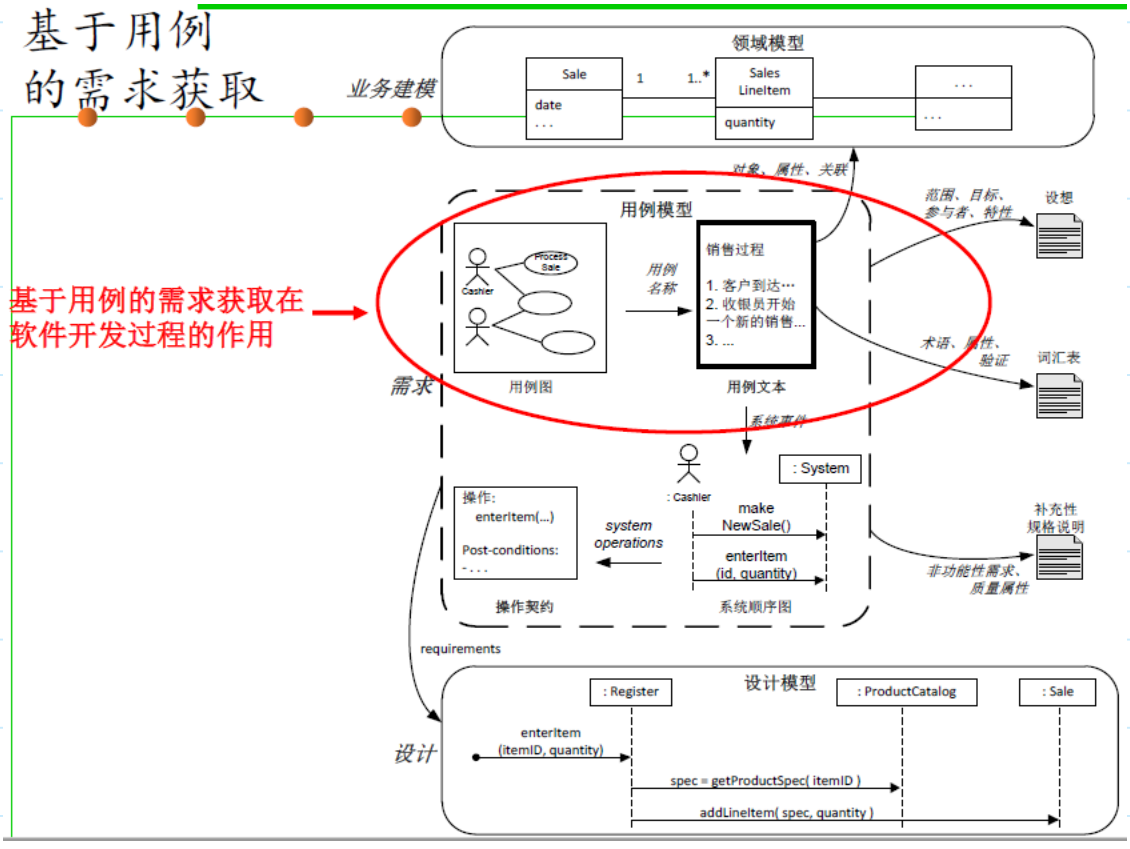
需求分析技术：用例

- 

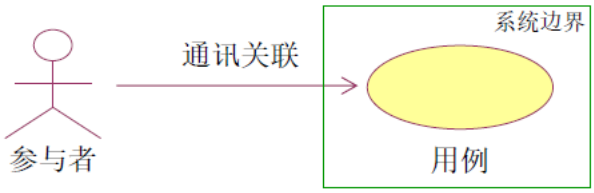
什么是用例(UseCase)?

- 用例(UseCase)：表示系统所提供的服务或可执行的某种行为
  - 定义了系统是如何被参与者所使用的，描述了参与者为了使用系统所提供的某一完整功能而与系统之间发生的一段“对话”（活动交互）
  - 用例的概念在1986年由IvarJacobson正式提出之后被广泛接受，迅速发展，已成为OO、UML、RUP的标准规范和方法。

基于用例的需求获取

- 

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(UseCase)
  - 系统边界(SystemBoundry)：明晰时可以省略。
  - 通讯关联(CommunicationAssociation)：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。
  - 

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(sequencesofactions)：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(systemperforms)：系统为外部角色提供服务；
  - 可观测到的、有价值的结果(observableresultofvalue)：用例必须对用户产生价值；
  - 特定的角色(particularactor)：某人、某台设备、某外部系统、等等，能够触发某些行为。

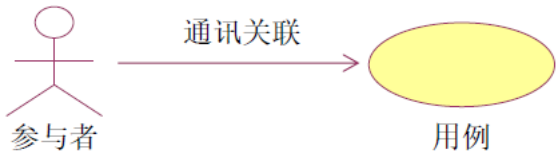
示例：ATM系统的用例

- 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

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。

- 

用例的内部剖析

- 用例=椭圆+名字？——NO！
- 用例是文本文档，而非图形，用例图是系统的蓝图；
- 用例建模主要是编写文本的活动，而非制图。
- 用例模型：一张或几张用例图+若干用例描述文本
- **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(扩展流)
  Pre-conditions (前置条件)
  Post-conditions (后置条件)

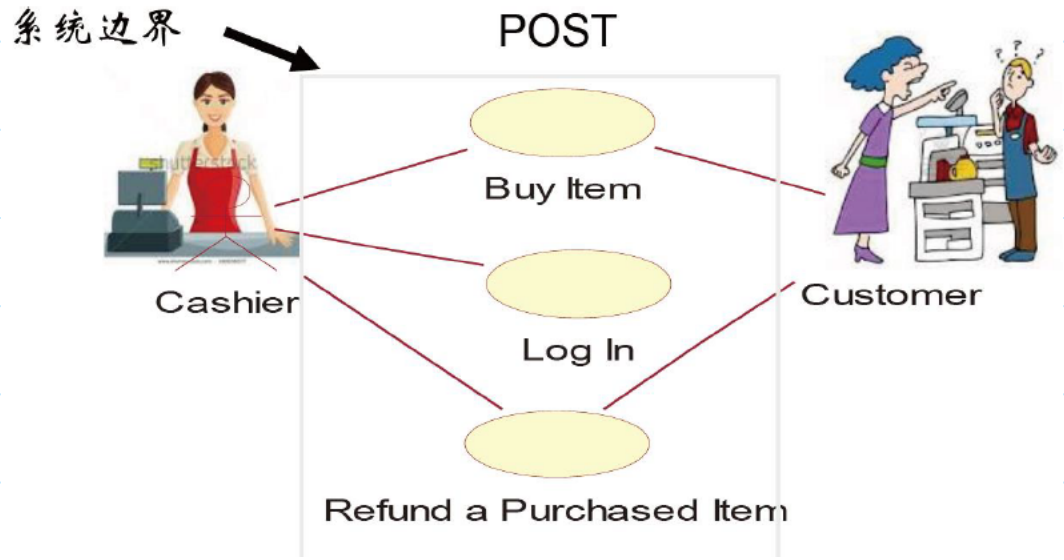
用例方法的优点

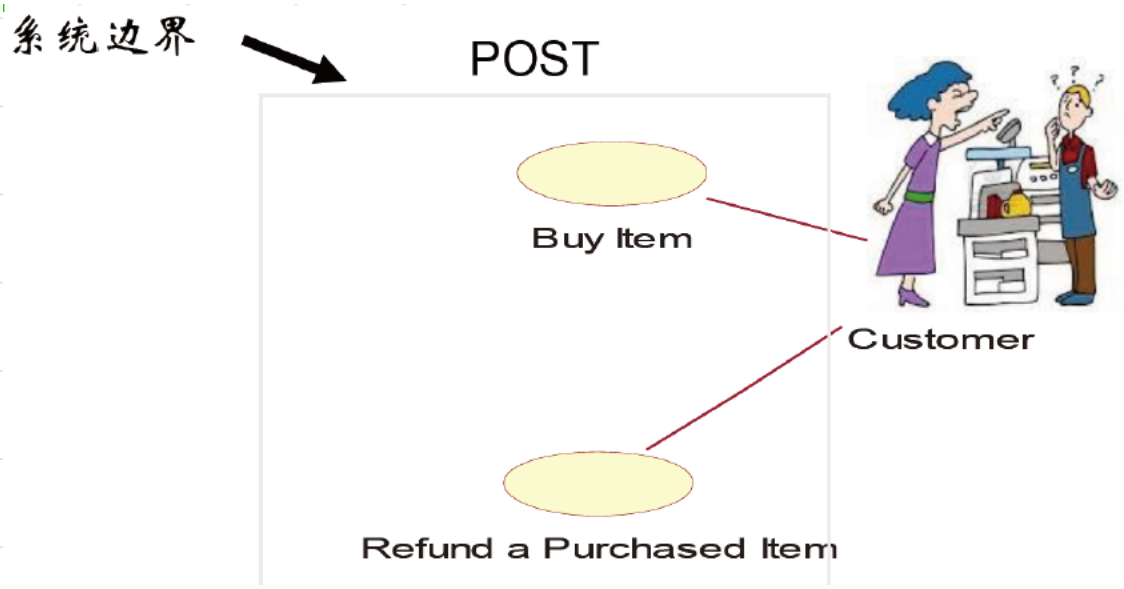
- 系统被看作是一个黑箱，并不关心系统内部是如何完成它所提供的功能的。
- 首先描述了被定义系统有哪些外部使用者(抽象为Actor)、这些使用者与被定义系统发生交互；
- 针对每一参与者，又描述了系统为这些参与者提供了什么样的服务(抽象成为UseCase)、或者说系统是如何被这些参与者使用的；
- 用例模型容易构建、书写简单、容易阅读；
- 完全站在用户的角度上，从系统外部来描述功能；
- 帮助系统的最终用户参与到需求分析过程中来，其需求更容易表达出来
- 用户能给出系统需求的情景，可以使人们理解需求的原因及系统如何实现它的目标。

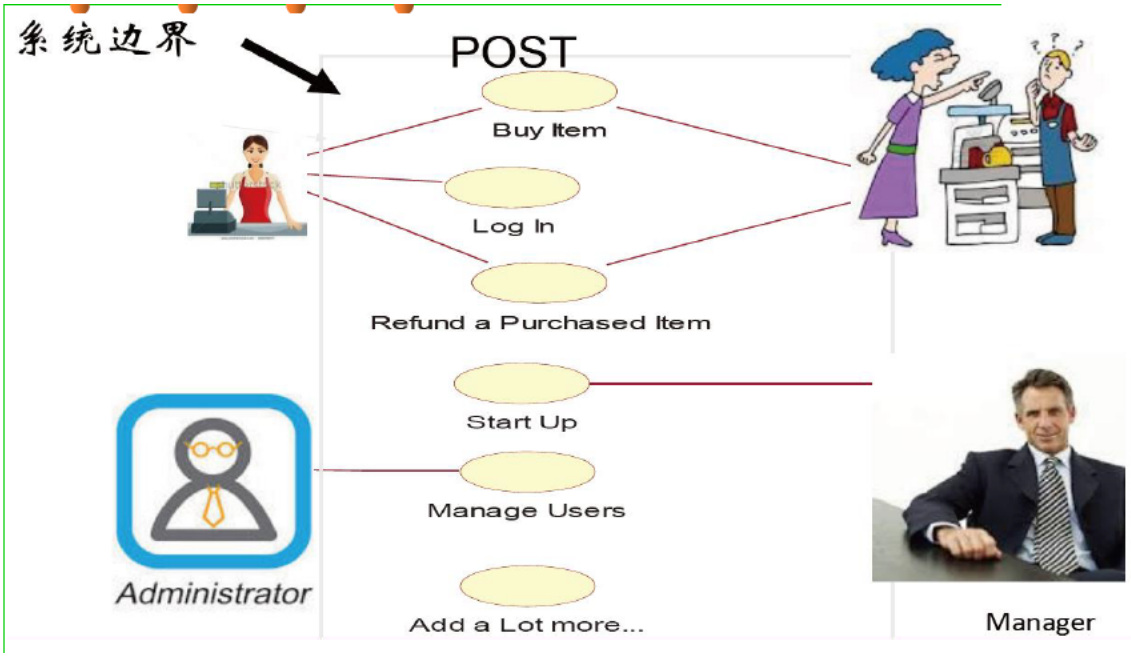
3 用例建模的基本过程

- 3.6 细化用例模型

3.1 确定系统边界

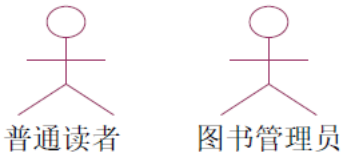
- 确定系统边界是为了识别出什么在系统之内，什么在系统之外，进而识别出什么是系统的职责。
- 典型的系统边界包括：一个组织中的部门或整个组织；硬件设备或硬件/软件边界；
- 确定系统边界时要明确
  - 系统目标
  - 系统范围
- 例：学生成绩管理系统
  - 目标：
    - 大学？中小学？
  - 范围：
    - 单机、网络？
    - 学籍？课程？
    - 短信/邮件通知？
- 考虑用于零售店销售管理的系统的用例图：
  - 记录销售及付款情况的软硬件集成系统
  - 包括硬件设备，如计算机、条码扫描装置
  - 包括运行在系统上的软件
  - 系统目标包括：
  - 



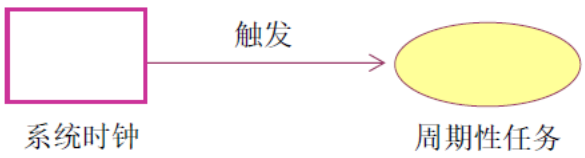


3.2 识别并描述参与者

- 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)哪些外部硬件设备?
  - 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?
- 例1：对一个图书管理系统来说，有哪些参与者?
  - 普通读者
  - 图书管理员

- 

- 例2：对ATM系统来说，有哪些参与者？如果是“银行系统”，“后台服务器还是参与者吗？
  - 银行客户
  - ATM维护人员
  - 后台服务器
  - 

- 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。
  - 

- “参与者”与“最终用户”
  - “参与者”与“最终用户”并非一回事
  - 典型用户在使用系统时可能会扮演多个不同的角色，在用例中他们仅扮演一种角色（参与者）。
  - 某些情况下，每个参与者的角色可能由不同的人员扮演。

3.3 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?

识别用例

- 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

识别用例的几点注意事项

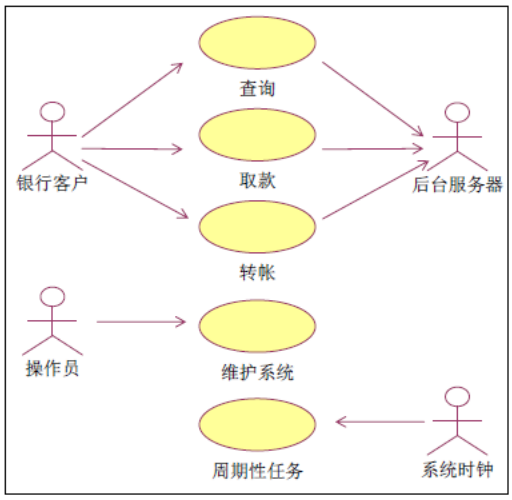
- 用例必须是由某一个actor触发而产生的活动，即每个用例至少应该涉及一个actor。
- 如果存在与actor不进行交互的用例，需要将其并入其他用例，或者是检查该用例相对应的参与者是否被遗漏。
- 反之，每个参与者也必须至少涉及到一个用例，如果发现有不与任何用例相关联的参与者存在：
  - 仔细考虑该参与者是如何与系统发生对话的；
  - 由参与者确定一个新的用例；
  - 该参与者是一个多余的模型元素，应该将其删除。

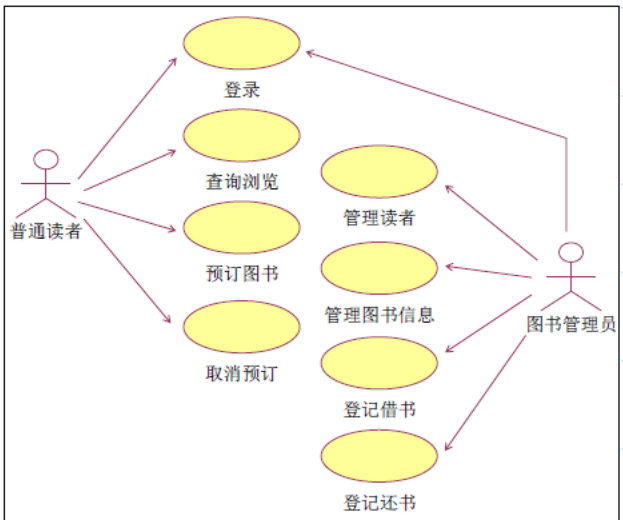
发现有用的用例

- 老板测试
- 基本业务过程测试
  - 基本业务过程：一个人于某个时刻在一个地点所执行的任务，用以响应业务事件。该任务能够增加可量化的业务价值，并且以持久状态留下数据。
- 规模测试
  - 用例由一系列相关联的步骤组成，不能是单独的活动!
- 下面哪个是有效用例?

- 就供应者合同进行协商
- 处理退货
- 登录
  - 将商品进行条码扫描
  - 保存学生成绩
  - 销售管理
  - 数据维护
  - 需要具体问题具体分析

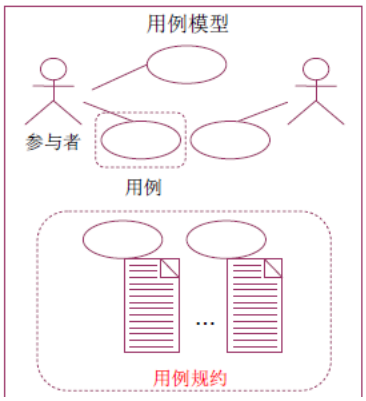
3.4 识别参与者与用例之间的通讯关联

- 



3.5 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。
- 用例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
- 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(usecaseinstance)

- 

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：
……

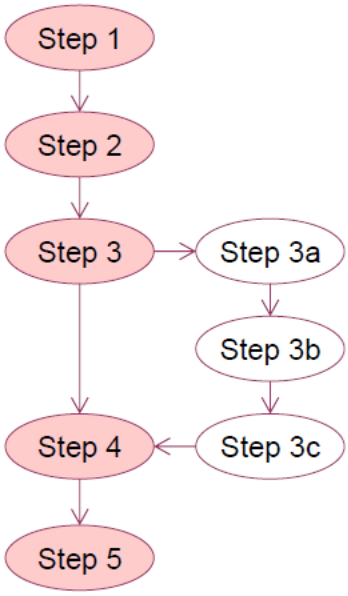
事件流（场景）

- 用例的事件流：

- 说明用例如何启动，即哪些参与者在何种情况下启动用例?
- 说明参与者与用例之间的信息处理过程;
- 说明用例在不同条件下可以选择执行的多种方案;
- 说明用例在什么情况下才能被视作完成;

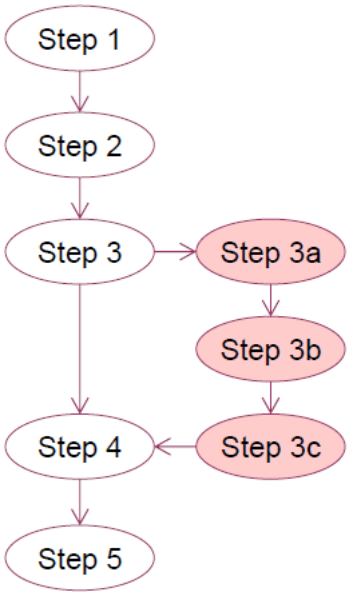
- 分为常规流和扩展流两类:
  - 常规流：描述该用例最正常的一种场景，系统执行一系列活动步骤来响应参与者提出的服务请求;
  - 扩展流/备选流：负责描述用例执行过程中异常的或偶尔发生的一些场景。

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，从正反两个方面来描述
  - 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。

- 

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

[案例]用例描述1

- **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：

2.1 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

2.2 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。

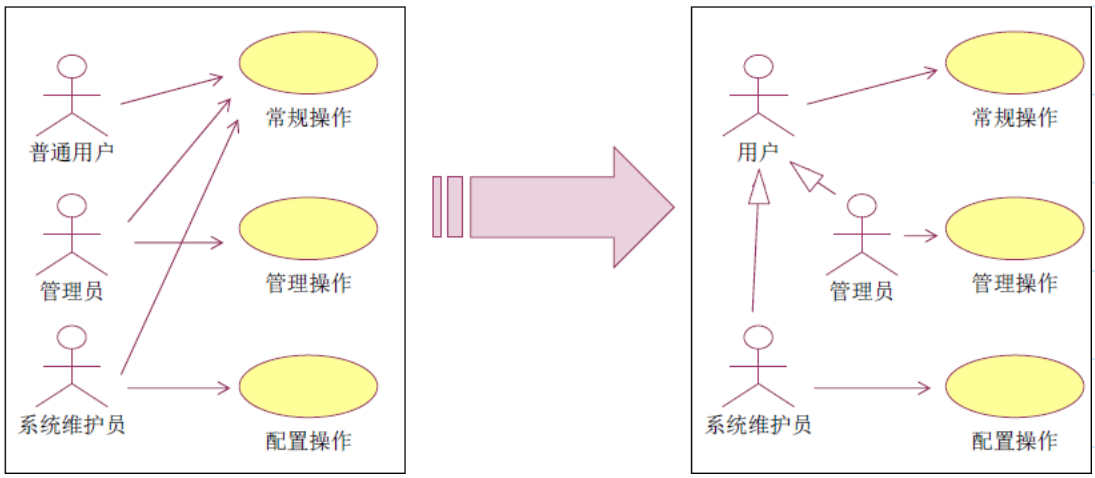
[案例]用例描述2

- 用例名称：处理销售
  参与者与关注点：
  ●收银员：希望准确、快速地输入，而且没有支付错误，因为如果少收货款，将从其工资中扣除。
  ●... ...
  前置条件：收银员必须经过确认和认证
  成功保证（或后置条件）：存储销售信息。准确计算税金。更新账务和库存信息。
  主成功场景（或基本流程）：
  1.顾客携带所购商品或服务到收银台通过POS机付款。
  2.收银员开始一次新的销售交易。
  3.收银员输入商品条码
  扩展（或替代流程）：
  3a.无效商品ID（在系统中未发现）：
  系统提示错误并拒绝输入该ID。
  收银员响应该错误
  特殊需求：
  ●使用大尺寸平面显示器触摸屏，文本信息可见距离为1米
  ●。。。
  发生频率：可能会不断地发生
  未解决问题：
  提成处理规则不确定
  收银员换班时如何处理

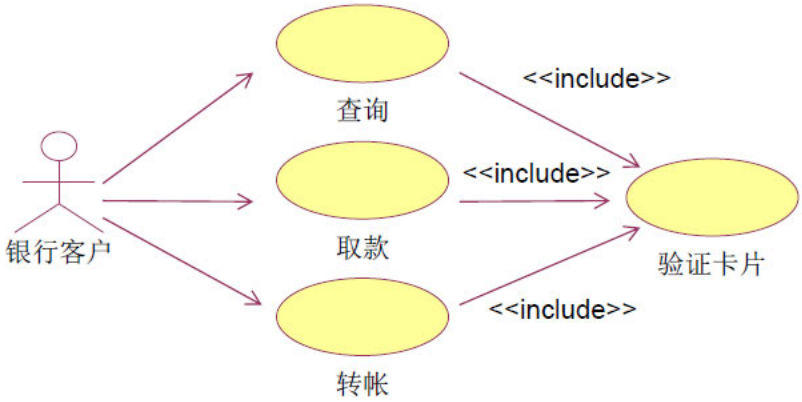
3.6 细化用例模型

- 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)
  - 用例和用例之间的泛化(generalization)关系
- 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。
- 根据用例描述绘制活动图
- 补充非功能性需求

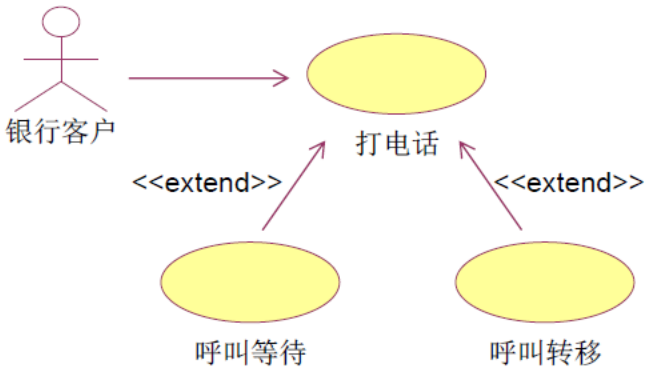
参与者之间的关系

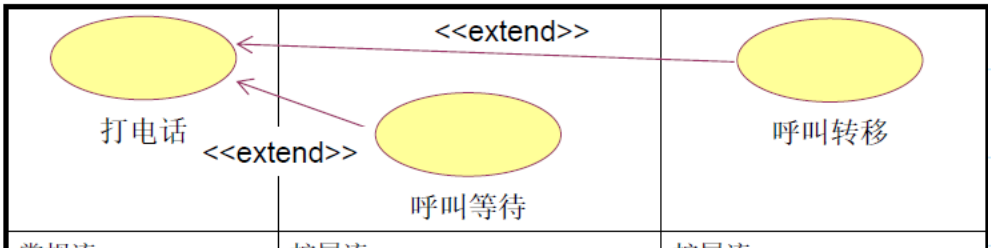
- 参与者之间可以有泛化(Generalization)关系。
- 

用例之间的关系：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
- 

用例之间的关系：扩展(extend)

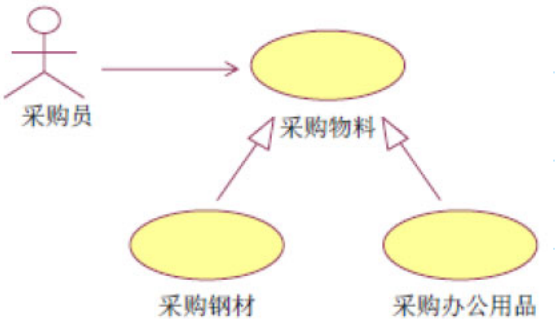
- “扩展关系”是通过在关联关系上加入<<extend>>标记来表示；
- 语义：用例2在某些特定情况下（有条件执行）会用到用例1，此时，用例1的事件流将被插入到用例2的事件流中。
- 一般表示为异常功能，大多是扩展流程
- 

- 

常规流：	扩展流：	扩展流：
1 拨号 2 建立通话链路 3 通话 4 挂机	1a 如果应答方正忙，用铃声提示应答方并保持拨号呼叫	1b 如果应答方无应答，进行呼叫转移

实际上相当于第一个用例的“扩展流”

用例之间的关系：泛化(generalization)

- 当多个用例共同拥有一种类似的结构和行为的时候，可将它们的共性抽象成为父用例，其他的用例作为泛化关系中的子用例。
- 子用例继承了父用例所有的结构、行为和关系。
- 

标识非功能性需求（URPS+）

- 可用性
  - 人性因素：顾客将能够看到POS大屏幕显示器的显示。因此：
  - 应该在1米外轻松看到文本
  - 避免使用一般色盲人群难以辨认的颜色
  - 快捷、无错的销售处理极为重要，因为购买者希望快速离开，否则会给他们的购买体验（和对销售员的评价）带来负面影响。
  - 收银员的视线通常停留在顾客或商品上，而不是计算机显示器上。因此，提示和告警应该通过声音传递而不仅仅是通过图像传递。
- 可靠性
  - 可恢复性
    - 如果正在使用外部服务（支付授权、账务系统、...）时出现故障，为了完成销售交易，需要尝试采用本地方案（如存储和转发）加以解决。对此需要更深入的分析
  - 长时间运行：当商品标识有破损或不能识别，应能够给予提示。
- 性能
  - 购买者希望非常快速地完成销售处理过程，数据本地存储的时间不能超过1秒，90%的授权服务在20秒内完成。
  - 系统应能够保证历史数据存储3年，查询当天数据相应时间<3秒，当月数据<5秒，当年数据<10秒。
- 可支持性
  - 可适应性
  - 不同型号的票据打印机打印的效果可能存在差异，软件能够支持市场上主流的票据打印机。

- 可配置性
    - 人员的权限会根据企业的变化而调整，系统应该能够方便配置调整。还存在一些其他的配置要求，如打印格式、查询项目等，对此需要进一步分析。
- 实现
    - 采用Java技术解决方案。Java易于开发，远期具有可移植、便于扩展的能力。
    - 持久化数据存储采用开源数据库方案，减少投资。
- 接口
    - 重要硬件和接口
        - 触摸屏（可视为鼠标输入）
        - 条码激光扫描仪（可视为键盘输入）
        - 票据打印机
        - 银联信用卡/借记卡读卡器
    - 软件接口
    - 银联支付系统
    - 财务账务系统