L2 需求管理(1)

内容提要

- 需求的基本概念
- 需求工作坊
- 用例分析方法
- 用户故事方法

内容提要

- **需求的基本概念**
- 需求工作坊
- 用例分析方法
- 用户故事方法

需求

- 需求：关于系统应当提供的服务以及对其运行的约束的描述
  - 主观因素：反映了客户和用户对于一个系统的需要，这种需要服务于一定的目的，例如购买所需要的商品
  - 客观因素：对系统开发决策有影响同时又不可改变的客观现实，例如一个学校的学生人数和校园网络状况
- 不同层次的需求
  - 用户需求：客户和用户表达的高层抽象需求
  - 系统需求：关于系统应当做什么的详细描述，又称为系统规格说明（System Specification）

软件需求

功能性需求

- 系统应提供哪些服务
- 系统如何响应特定的输入
- 系统在特定情形下的行为（即应当做什么、不应当做什么）

非功能性需求

- 待开发系统的质量属性，例如系统的性能、可靠性、稳定性等
  - 可以是系统整体、全局性的定义
  - 也可以是特定的服务、功能或系统组件的定义

约束

- 针对开发过程或待开发系统自身属性的一种限制。通常来自其他组织过程或系统运行的上下文环境

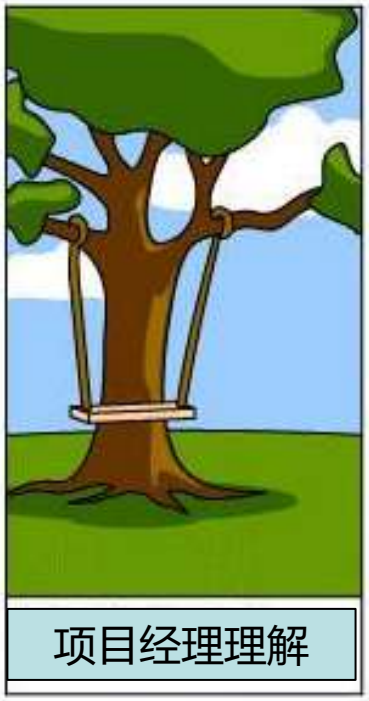
需求与项目成功

Project Challenged Factors	% of Responses
1. Lack of User Input	12.8%
2. Incomplete Requirements & Specifications	12.3%
3. Changing Requirements & Specifications	11.8%
4. Lack of Executive Support	7.5%
5. Technology Incompetence	7.0%
6. Lack of Resources	6.4%
7. Unrealistic Expectations	5.9%
8. Unclear Objectives	5.3%
9. Unrealistic Time Frames	4.3%
10. New Technology	3.7%
Other	23.0%

需求不清晰、易变是突出问题

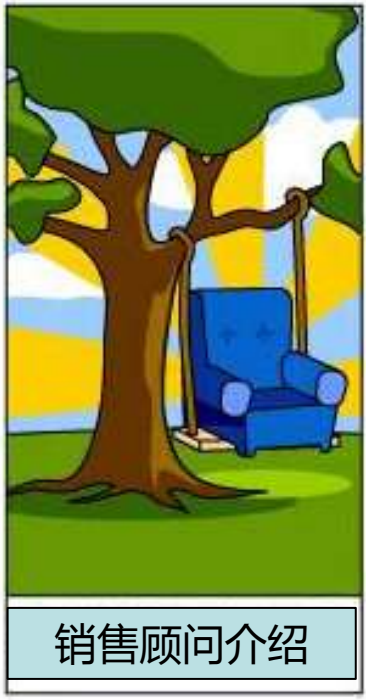
The Standish Group Report: Chaos, 1995

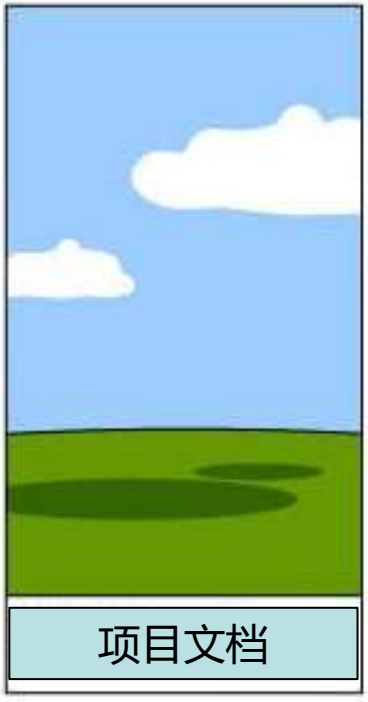

客户描述

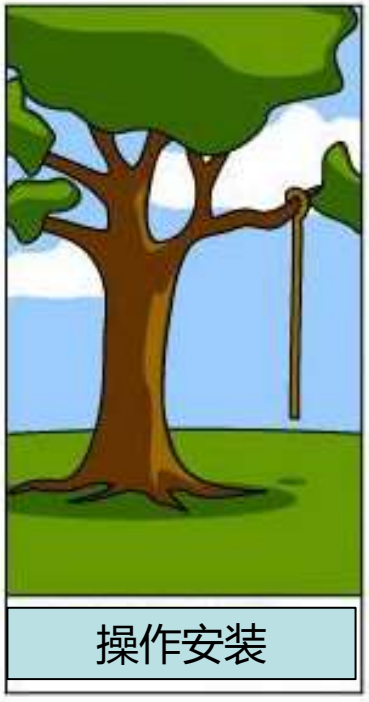

项目经理理解

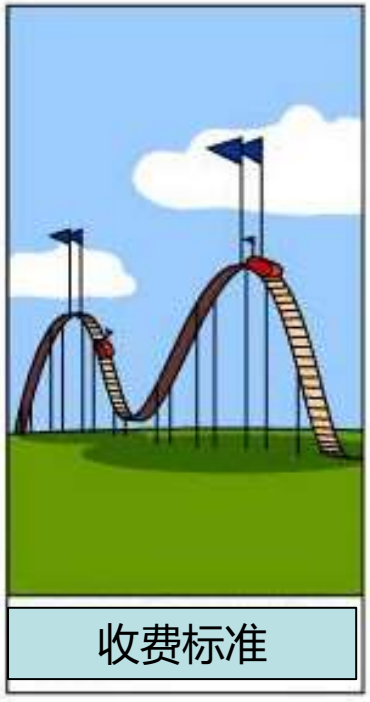

分析师的设计

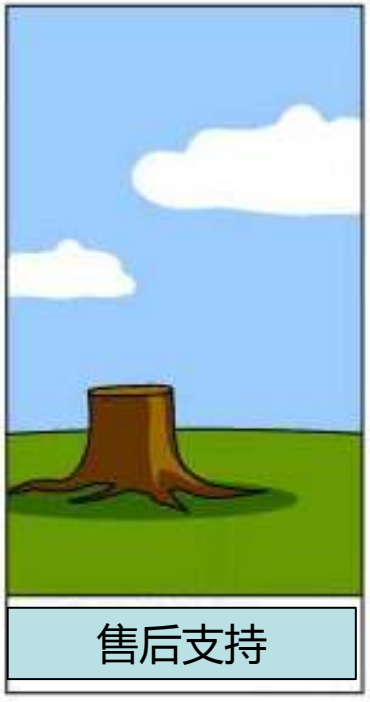

程序员的实现


销售顾问介绍


项目文档


操作安装


收费标准


售后支持


客户实际想要的

需求以及需求管理的一些问题

- 需求是否清楚?
- 需求是否表达和满足了业务目标?
- 需求之间是否存在关联?
- 需求之间的优先级是怎样的?
- 需求是什么状态？怎样才是完成了需求?

试图用足够的规范化文档……

- 写下来，传递下去！
  - 系统应当……
  - 系统应当……
  - 系统应当……
  - 系统应当……
  - 系统应当……
  - ……

需求描述不是软件开发的目的！

交付可运行的有用软件才是！

- 也许华丽、完整，然而……

需求描述的不准确性

Make it warmer

Too cold?

Too hot?

内容提要

- 需求的基本概念
- **需求工作坊**
- 用例分析方法
- 用户故事方法

需求工作坊

- 需求工作坊是一个协作活动。目标在于：
  - 澄清需求开发的愿景
  - 协调不同利益和开发视角
  - 建立通用的词汇和概念（统一语言）
  - 产生有关系统功能的具体想法（idea），创建待办事项列表
  - 完成沟通、澄清和说明

内容提要

- 需求的基本概念
- 需求工作坊
- **用例分析方法**
- 用户故事方法

UML——统一建模语言

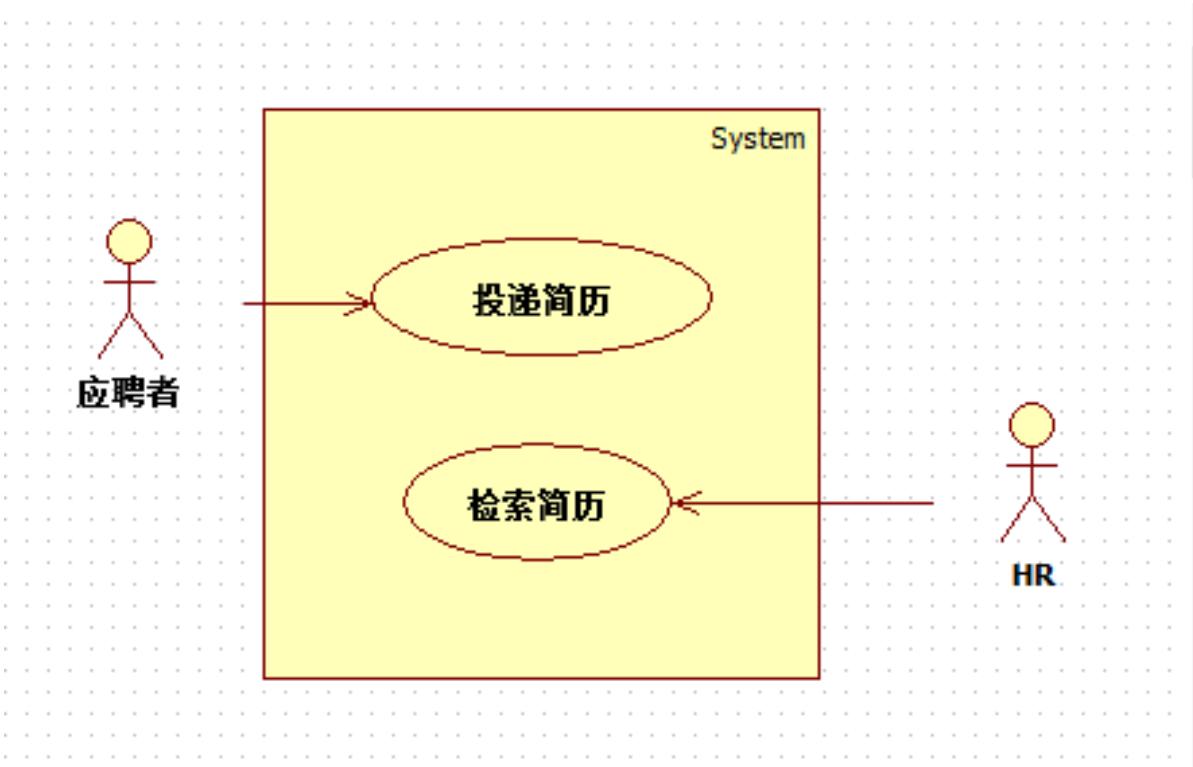
- 用例
- 用例图

- 类
- 类图

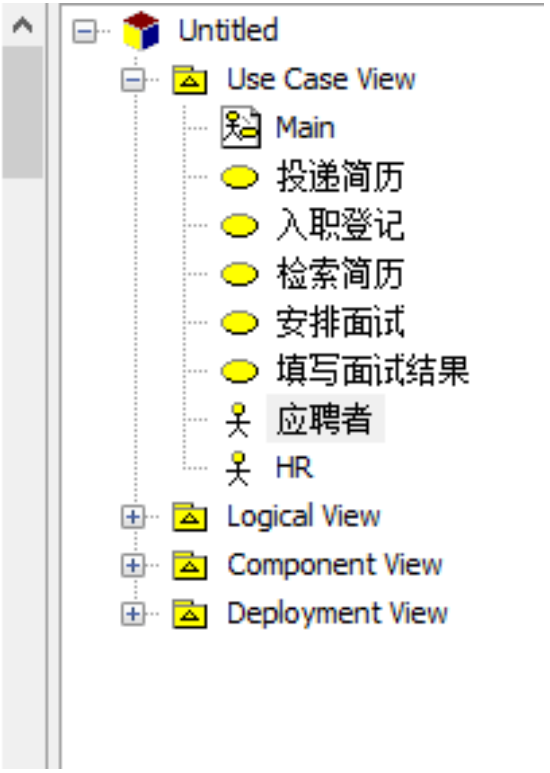
- 顺序图
- 活动图
- ……

- **现实世界可以看作对象和对象的协作**
- 协作具有清晰的业务目标和涉众(stakeholders)
- 可以使用UML图来描述业务过程
- 对软件系统而言，需要定义清晰的系统边界和操作者

用例和用例图



Diagram



Model

用例和用例图

- **执行者(actor)**——与系统交互的实体
- **用例(use case)**——执行者使用系统的一种方式，描述一个或多个参与者和系统之间的交互序列
- **系统边界(boundary)**——描述了系统的范围

系统边界和执行者

- 对**系统边界**的思考有助于认识系统与外部世界的接口和通过这些接口与系统进行交互的执行者
- **识别执行者**、分析他们与系统交互的情况，是定义用例的主要手段
- 考虑系统外部的执行者是启发分析人员发现对象的主要策略之一

用例不仅仅是图

描述项	说明
用例名称	简短、易理解的名称，一般以动词开头。
主执行者	参与当前用例交互过程的外部参与者
涉众及关注点	与当前用例相关的涉众、以及这些涉众对该用例的不同关注点和目标
前置条件	当前用例可执行前应该满足的条件
后置条件	当前用例成功结束后应该满足的条件
触发事件及条件	触发当前用例执行的事件
发生频率	当前用例发生的频率。该属性对和用例相关的性能、可用性等质量属性的设计具有较大的影响。
主场景	当前用例的（唯一）主场景。应通过顺序编号的方式描述场景的每个交互步骤。
可替换场景	当前用例的（零个或多个）可替换场景。可以类似主场景的方式逐一描述每个可替换场景的交互步骤，也可在主场景基础上描述每个可替换场景的差异部分。
例外场景	当前用例的（零个或多个）例外场景。对于每个例外场景都要描述例外条件和相应的例外处理步骤。
质量属性	与当前用例执行过程相关的质量属性及邀请。例如，安全保密性需求、易用性需求等。
杂项	待解决问题，需要进一步澄清或协商的地方等。

用例示例

- **用例名称：**企业商品采购
- **执行者：**采购发起人（请求者、企业员工）
- **涉众及关注点：**
  - 请求者：希望得到需要采购的东西，并且操作要简单。
  - 公司（审批者）：希望控制花费，但允许必要的购买
  - 供货商：希望得到任何已发货物的付款
  - 采购员：希望方便确认库存，并在需要时向供货商发出订单
- **前置条件：**无
- **后置条件：**请求者得到货物，并将货款从预算额度中扣除。
- **触发事件及条件:**请求者决定购买东西

用例示例

- **主场景：**

1. 请求者：发起一个请求。
2. 审批者：检验预算资金，检查货物价格，批准请求
3. 采购员：检查仓库存货，找到供应商
4. 采购员：完成订购请求，向供货商发出订单

…

扩展(可替换场景）：

1a. 请求者不知道供货商和货物价格，不填写这些内容，然后继续。

2a. 审批者拒绝申请，发回给请求者，要求其修改或删除

…

用例场景

用例场景包含一系列为满足目标而执行的交互步骤，提供了业务用例的具体细节

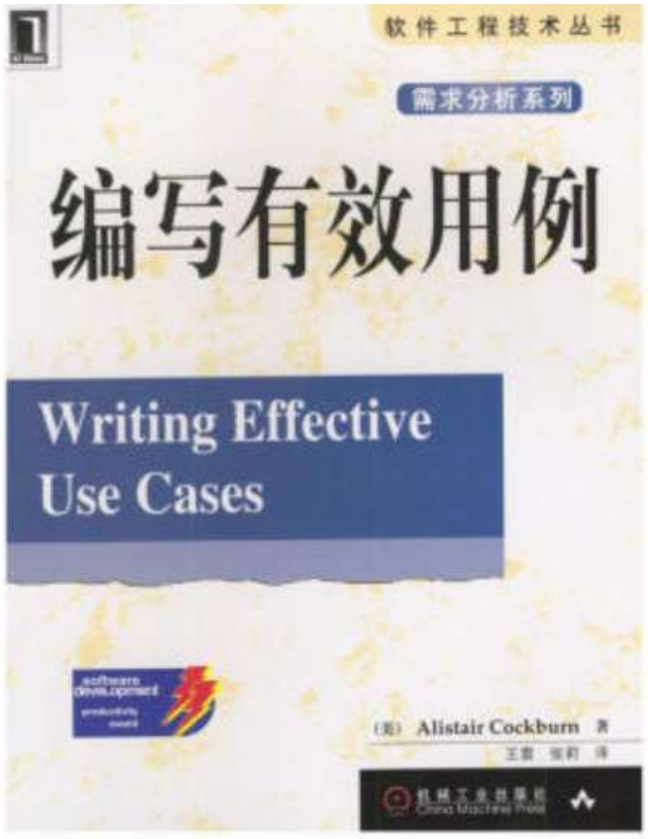
一个用例由一个**主场景**、若干**可替换场景**和若干**例外场景**组成

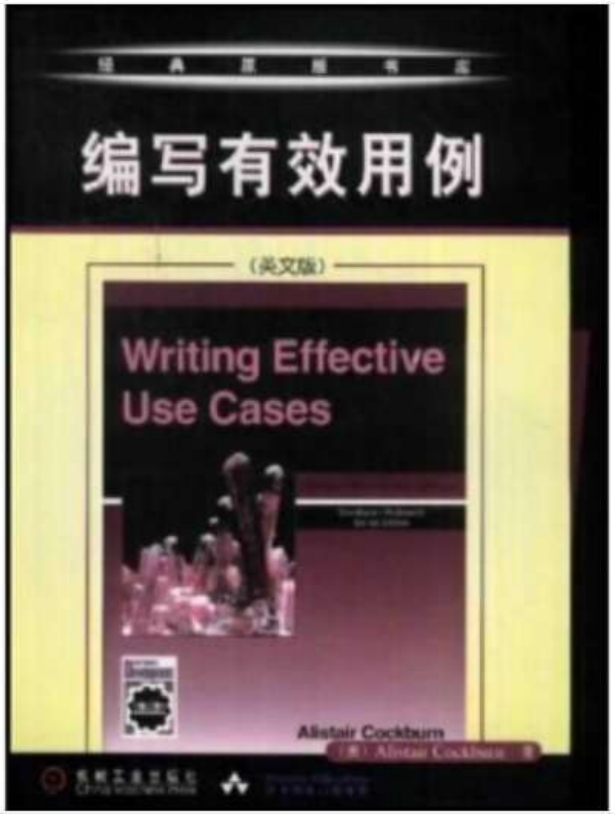
- 主场景：代表相关涉众期望发生的一般情况下的交互序列
- 可替换场景：与主场景相比在交互步骤上有所差异的可替换场景
- 例外场景：在此过程中可能发生的例外情况的例外场景

使用用例的优势和劣势

- 优势
  - 用户的角度，体现价值
  - 系统的角度，完善细节
  - 层次性、整体性
  - 可使用UML贯穿开发过程（用例驱动）
- 劣势
  - 用例有的时候过于复杂，细节过多
  - 任务需要进一步拆分，以便管理和推进开发
  - 变更相对困难，影响也可能更大
    - 早期识别的细节可能需要补充或者修改

- Alistair Cockburn，Writing Effective Use Cases





内容提要

- 需求的基本概念
- 需求工作坊
- 用例分析方法
- **用户故事方法**

用户故事 User Story

- **为什么使用用户故事**
- 什么是用户故事
- 用户故事的优势与不足
- 如何编写故事
- 优秀故事的准则

为什么使用用户故事

- 需求是一个沟通问题
  - 业务方（客户和用户）
    - 包括分析人员、领域专家和其他从业务或组织视角来审视软件的人
  - 技术团队或开发团队
- 任何一方把持绝对地位，项目就会遭受损失
  - 业务方关注软件功能和交付日期
    - 很少关注开发人员能否**同时**满足这两个目标
    - 很少关注开发人员是否**确切地**了解需求（也许自己也不完全清楚）
  - 开发团队会关注技术实现
    - 技术术语代替业务语言，难以倾听**实际**需求

为什么使用用户故事

- 协同的方法
  - 业务方和开发团队共同面对资源分配、需求澄清问题
- 根据手头的信息来做决策
  - 各个决策分散在项目过程中
- 确保有一个获取信息的过程
  - 越早越好
  - 越频繁越好

- 用例是不错的方法
  - 但有时候太大、内容太复杂了……
  - 要另外想办法安排开发进度、分解任务

为什么使用用户故事

- 站在客户和用户的视角，描述客户需求
  - 逐步精化的细节
- 可独立交付的单元，规模小，适合于迭代开发，以获取快速反馈
- 强调编写验收测试用例作为验收标准
  - 准确把握需求背后的实际含义
  - 而不仅仅用文字记录需求本身

用户故事 User Story

- 为什么使用用户故事
- **什么是用户故事**
- 用户故事的优势与不足
- 如何编写故事
- 优秀故事的准则

什么是用户故事?

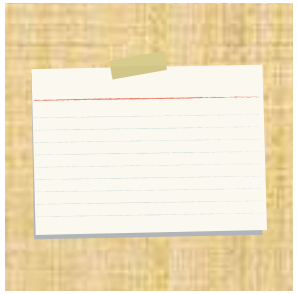
- 理解用户故事
  - **用户故事描述了对系统或软件的客户或用户有价值的功能片段**
  - 用户故事中的“卡片**代表**客户需求但不是**记录**需求”
  - **卡片**包含故事的文字描述
  - 需求的细节需要在**对话**中获得
  - 在**确认**中以测试的方式得以记录

Ron Jefferies

Rachel Davies

什么是用户故事?

- 故事由三方面组成（3C)
  - **卡片（Card）**：一个书面的、简短的故事描述，用来做计划并在开发过程中起到提醒的作用
  - **对话（Conversation）**：针对故事描述进行的交流，用来澄清故事的细节
  - **确认（Confirmation）**：测试，用于表达和归档故事细节且可用于确定故事何时完成



卡片（Card）

交流（Conversation）

确认（Confirmation）

故事的描述模板

Rachel Davies

作为（As a） X [什么用户角色]，
我要（I want） Y [能做什么]，
以便（so that） Z [达到什么目的或实现什么价值]

- 如果为了更加突出价值，也可以把so that放在前面作为“目的”
  - 为了Z，X要Y
- 如果角色、目的非常明显，也可以省略它们
- “I want”应该是简洁、用户可感知的功能描述

故事的描述模板

- 用户故事并没有必须遵循的标准模板
- 任何能够**准确表达**下述内容的**简单**文字，都是好的方式
  - Who is the user? (Stakeholder, person, role)
  - What do they want to do? (Action)
  - Why do they want to do it? (Value or justification)

* https://resources.scrumalliance.org/Article/anatomy-user-storyte

故事的描述模板——例子

- 在一个在线招聘网站中

作为（As a） 求职者，
我要（I want） 搜索工作，
以便（so that） 获取可申请的职位信息

或者

求职者可以搜索工作

- 对用户角色而言，价值非常清楚，无需多言

故事的描述模板——例子

- 在一个在线招聘网站中

> 作为（As a） 求职者，
> 我要（I want） 快速找到符合我的技能和薪资水平的工作，
> 以便（so that） 向合适的公司投递简历

- 明确什么是功能，什么是目的或价值
- 用户故事是对**问题**的描述，而不是解决方案

故事的描述模板——不理想的故事

- 在一个在线招聘网站中

> 这个软件用C++语言编写

或者

> 程序将通过连接池连接到数据库

- 若不是用户关注或需要感知的功能，那么即使它是一个功能，也不应该作为用户故事

故事的描述模板——例子

- 在一个在线招聘网站中

> 求职者可以搜索工作

  - 细节在哪里?
    - 搜索哪些值？省份？城市？职位？关键字?
    - 必须是注册用户吗?
    - 要显示哪些匹配职位的信息?
    - 搜索参数或结果可以保存吗?
    - ……

故事的描述模板——例子

- 在一个在线招聘网站中

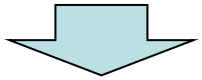
> 求职者可以搜索工作

> 公司可以发布工作信息

- 庞大的用户故事（史诗故事，Epic）
  - 拆分为多个小的故事
  - 每个故事半天到几天可以有单个程序员或者一对程序员完成（包括编码和测试）

故事的描述模板——例子

求职者可以搜索工作



求职者可以通过地区、薪水范围、职位、公司名称和发布日期之类的属性来搜索工作

求职者可以查看搜索结果中的每个工作的信息

求职者可以查看发布工作的公司的详细信息

- 但不需要过度拆分
  - “每个工作的信息”？包括描述，薪水，地点?

故事的描述模板——例子

求职者可以查看搜索结果中的每个工作的信息

- “每个工作的信息”
  - 可以包括工作描述、薪水范围、工作地点……
  - 但是仍然可以包含别的东西
  - 每个已经识别出的属性也可以有更为详细的内容
- 与其写下所有这些故事细节，不如在这些细节变得重要时（比如开发时）才讨论
  - 可以加上一些注释，用来形成初步的共识
  - 达成的共识由 **测试** 来记录

关键在于讨论

故事的描述模板——例子

- 带有注释的故事卡片

故事卡正面

求职者可以查看搜索结果中的每个工作的信息

小张：要显示描述、薪酬以及工作地点

- 故事卡片背面写下用户的期望（测试故事的提示语句）

故事卡背面

试试空的工作描述
试试很长的工作描述
试试没有薪资会怎样
试试6位数的薪资

验收测试 Acceptance Tests

- 验收测试用来验证所实现的用户故事是否符合客户团队的期望
  - 客户团队：确保软件满足用户需求的所有人，比如测试人员、产品经理、实际用户和交互设计师
- 测试工作
  - 在沟通的基础上，写下测试描述
  - 编写测试代码
  - 将测试代码放到自动化测试工具中
- 团队中要包含一个专业的测试人员（来完成偏技术的工作）
- 尽早开始测试工作
  - 与开发人员开始编写代码同时开始

验收测试 Acceptance Tests

故事卡正面

用户可以用信用卡为购物车中的商品付款

故事卡背面

- VISA, MasterCard, UnionPay的信用卡测试(通过)
- 运通卡不支持(失败)
- 用UnionPay的借记卡测试(通过)
- 用有效、无效或者缺失卡背面校验码测试
- 用过期卡测试
- 用不同金额测试（包括超出信用卡额度）

写下当时的预期，如果发现还有更多细节，可以在这些细节变得重要时（比如要开始开发时）才进一步细化（创建新的故事）

用户故事 User Story

- 为什么使用用户故事
- 什么是用户故事
- **用户故事的优势与不足**
- 如何编写故事
- 优秀故事的准则

用户故事的优势

- 用户故事强调对话交流而不是书面沟通
- 用户故事没有技术术语，能被客户和开发人员同时理解
- 用户故事的大小适合于做计划
- 用户故事适合于迭代开发
- 用户故事鼓励推迟考虑细节
- 用户故事支持随机应变的开发
- 用户故事鼓励参与性设计
- 用户故事传播隐性知识

用户故事的不足

- 面对大型项目，用户故事数量庞大，相互关系错综复杂、难以捉摸
  - 面对大量的需求时，用例（use case）的固有层级性会使需求收集比较方便
  - 用例可以通过其核心成功场景（main success scenario）以及扩展（extensions）把大量的用户故事汇集成一个实体
- 需求的可追溯性（traceability）
  - 用户故事卡片用完即扔
  - 利用一份额外的文档把**用户故事**与**验收测试**关联起来
- 大型团队中隐性知识如果不记录下来，难以得到必要的共享——书面文档与面对面交流的平衡

用户故事 User Story

- 为什么使用用户故事
- 什么是用户故事
- 用户故事的优势与不足
- **如何编写故事**
- 优秀故事的准则

基本原则——INVEST

- 独立的 Independent
- 可讨论的 Negotiable
- 有价值的 Valuable to Purchasers or Users
- 可估计的 Estimatable
- 小的 Small
- 可测试的 Testable

独立的 Independent

- 避免在故事之间引入依赖关系
  - 依赖会影响判定优先级和排定计划
  - 依赖会让故事的估计更加困难

- 然而，依赖关系往往不可避免……

独立的 Independent——例子

用户可以用VISA信用卡为商品付款

用户可以用银联信用卡为商品付款

用户可以用MasterCard信用卡为商品付款

- 看似独立的三个故事
- 第一种信用卡类型需要3天时间，但此后每添加一种信用卡，只需要1天时间
- 如果不排计划，就无法给这些故事确定估计

独立的 Independent——例子

用户可以用VISA信用卡为商品付款

用户可以用银联信用卡为商品付款

用户可以用MasterCard信用卡为商品付款

- 处理方法1：合并成一个大的、独立的故事
  - 如果合并后仍然可以在短时间内（例如几天）完成，则可以用合并的方法
- 处理方法2：用不同的方式去分割故事

用一种信用卡支付（3天）
用另外两种信用卡支付（2天）

- 处理方法3：给出两个不同的估计

独立的 Independent——例子

用户可以用VISA信用卡为商品付款

用户可以用银联信用卡为商品付款

用户可以用MasterCard信用卡为商品付款

- 一些典型的依赖关系往往是技术相关的
  - 确定数据结构后才能建数据库?
  - 有数据文件后才能做数据读写?
  - 有后台数据才能进行前台展示?
- 用户故事是非技术语言，是直接对用户有用的功能
  - 有优先级，但可以不相互依赖
- 实现中的依赖关系可以通过技术手段解决（比如mock）

可讨论的 Negotiable

- 故事不是必须要遵守的书面的合同或需求文档
- 故事是关于功能特性的简短描述
  - 沟通->细节->测试
- 故事卡仅仅是提醒将来需要交流的内容，不需要（不应该，也不可能）包括所有相关的细节
  - 一些已知的重要细节可以作为注释记录在故事卡片上

用户可以用信用卡为购物车中的商品付款

支持Visa、MasterCard、银联信用卡；美国运通暂不支持

可讨论的 Negotiable

- 为什么不是这样？？

用户可以用信用卡为购物车中的商品付款

支持Visa、MasterCard、银联信用卡；美国运通暂不支持；支付金额超过1000元或等值人民币时需要提供手机验证；用户不需要手工选择卡类别，而是自动通过卡号前两位来自动确定；需要记录卡号，但是只能记录卡号后四位，且必须由用户确认；要记录卡的有效期；用户可以选择已经付款成功的卡付款，无需再提供卡号

可讨论的 Negotiable

- 为什么不是这样？？

> 用户可以用信用卡为购物车中的商品付款
>
> 支持Visa、MasterCard、银联信用卡；美国运通暂不支持；支付金额超过1000元或等值人民币时需要提供手机验证；用户不需要手工选择卡类别，而是自动通过卡号前两位来自动确定；需要记录卡号，但是只能记录卡号后四位，且必须由用户确认；要记录卡的有效期；用户可以选择已经付款成功的卡付款，无需再提供卡号

  - 过多的细节给人以错觉，需求就在这里，不需要再沟通了
  - 涵盖所有细节是非常困难的

可讨论的 Negotiable

- 客户团队负责提供故事
  - 故事不是验收指标，而是讨论细节的起点
  - 确定的细节要变成测试（把测试要点写在卡片背面或者电子文档的特定地方）
    - 在开始开发故事代码的同时，开始测试工作

用户可以用信用卡为购物车中的商品付款

支持Visa、MasterCard、银联信用卡；美国运通暂不支持

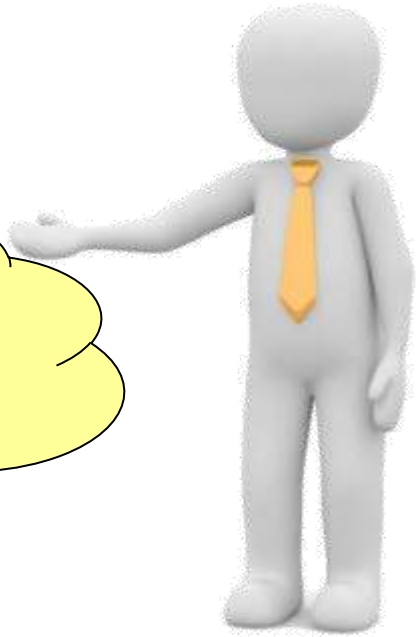
- 输入62xx xxxx xxxx xxxx时，识别为银联卡
- 当输入卡可用余额为100，商品金额为101元时，支付失败，提示余额不足
- ……
- 要记录已经支付过的卡号？ 我们再建一个故事吧~

有价值的 Valuable

- 与客户团队密切合作，由客户来写故事

  故事卡片只是记录了期望，而非验收标准或具体描述

  开发与客户：非对抗



可估计的 Estimatable

- 每个故事需要花费多少时间完成
  - 心里大概有个数，便于形成共识
  - 便于安排计划（Fit in the time box）
- 但估计往往困难
  - 开发人员缺少领域知识
    - 与业务人员沟通
  - 开发人员缺少技术知识
    - 技术穿刺 （spike），限定探索的时间上限
  - 故事太大了
    - 根据需要拆分故事，或者给一个粗略的整体估计作为占位符

小的 Small

- 合适的小
  - 过大的故事：史诗故事（Epics）
    - 复合故事
      - 数据对象的多个部分
      - 对数据的多种处理（例如增删改）
    - 复杂故事
      - 整体庞大复杂且不容易分解
  - 对过大的故事进行拆分……

小的 Small

- 故事拆分方法
  - 根据数据边界拆分
    - 填写个人信息时的：教育情况、工作情况、家庭情况、获奖信息、社会兼职
  - 根据逻辑顺序拆分
    - 填写个人信息、修改个人信息、删除个人信息
  - 根据不确定性或技术障碍拆分
    - 技术穿刺
      - 技术准备的故事+用该技术进行开发的故事
    - 架构穿刺
      - 关键架构元素搭建和验证的故事+具体的业务功能故事
  - “切蛋糕”（slice the cake）——**端到端**的完整功能

小的 Small

- 故事的合并
  - 过小的故事
    - 填写个人信息时的：教育情况中每个教育阶段的起止日期的填写和修改
  - 编写故事的人（比如客户）并不知道这个故事过小了
  - 在看到相关的几个过小的故事时，可以进行合并
    - 在纸质卡片情况下，可以把几个卡片叠起来夹在一起

可测试的 Testable

- 故事完成的标准：通过了测试
  - 测试点注释用于捕获开发人员未考虑或者不知道的标准
  - 不排斥其他测试：用于明确当前故事边界到哪里为止
  - 还漏了啥？撕掉完成的那个，再来一个（或者一打）新的故事
- 尽量使用自动化测试
- 不可测试的故事往往发生在非功能需求上
  - “很长时间” “很容易使用” “界面要看上去舒服”
  - 请考虑换一种表达
  - 如果不能自动化测试，那么就用人工方式进行测试

用户故事 User Story

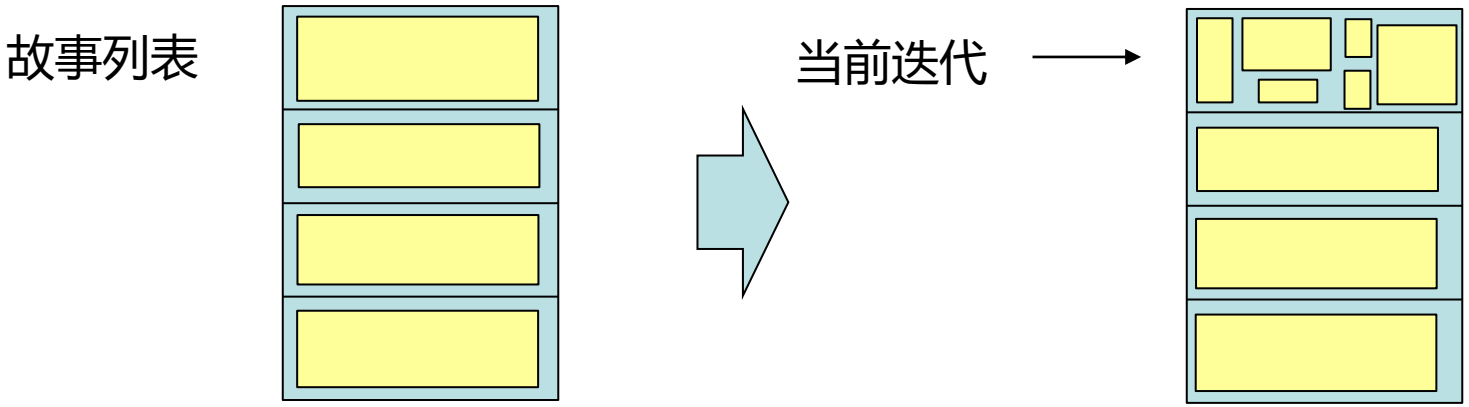
- 为什么使用用户故事
- 什么是用户故事
- 用户故事的优势与不足
- 如何编写故事
- **优秀故事的准则**

优秀用户故事的准则

- “切蛋糕”
  - 提供某种程度的完整的功能
  - 而不是按照技术路线来分
    - 比如“界面填写履历” “后台保存履历数据”两者只有合在一起才能实际使用
- 编写封闭的故事
  - 封闭：实现一个目标就可以结束；让用户觉得完成了某个任务
  - 不好的例子：销售职员管理所有的客房预定信息（酒店预订系统）
- 不要过早涉及界面
  - 避免把需求和解决方案混到一起

优秀用户故事的准则

- 根据实现进度确定故事的大小
  (Size the Story to the Horizon)
  - 聚焦于即将发生的迭代
  - 近期的故事需要细化拆分，远期的故事可以是更高层次、更粗的粒度
  - 利用故事灵活性的优势



优秀用户故事的准则

- 用户故事里包含角色
- 只为一个用户编写
  - 针对性强，可读性强，代入感强
- 用主动语态编写
- 由客户编写
- 不要故事卡编号
  - 如果一定要编号，那么顺序编号足够，并且需要一个简短的标题
- 不要忘记意图
  - 故事需要简洁，便于沟通讨论，仅记载备忘的关键细节，表明给客户或用户带来的价值

用户故事编写练习

- 在线购物系统
  - 用户？客户？角色？价值?
  - 为谁做？卖什么？谁卖？谁买？怎么买？……
  - 考虑下面这些描述：
    - 运营方能添加商品，设定价格(商品可以是实体的，也可以是虚拟的)
    - **消费者能购买商品**
    - 消费者能浏览商品信息，查看每个商品的详情
    - 消费者能把选中的商品加入购物车，并结账购买
    - 消费者能直接购买商品并支付
    - 消费者能查看所购买商品的运送情况

Next Lecture

- 需求管理(2)