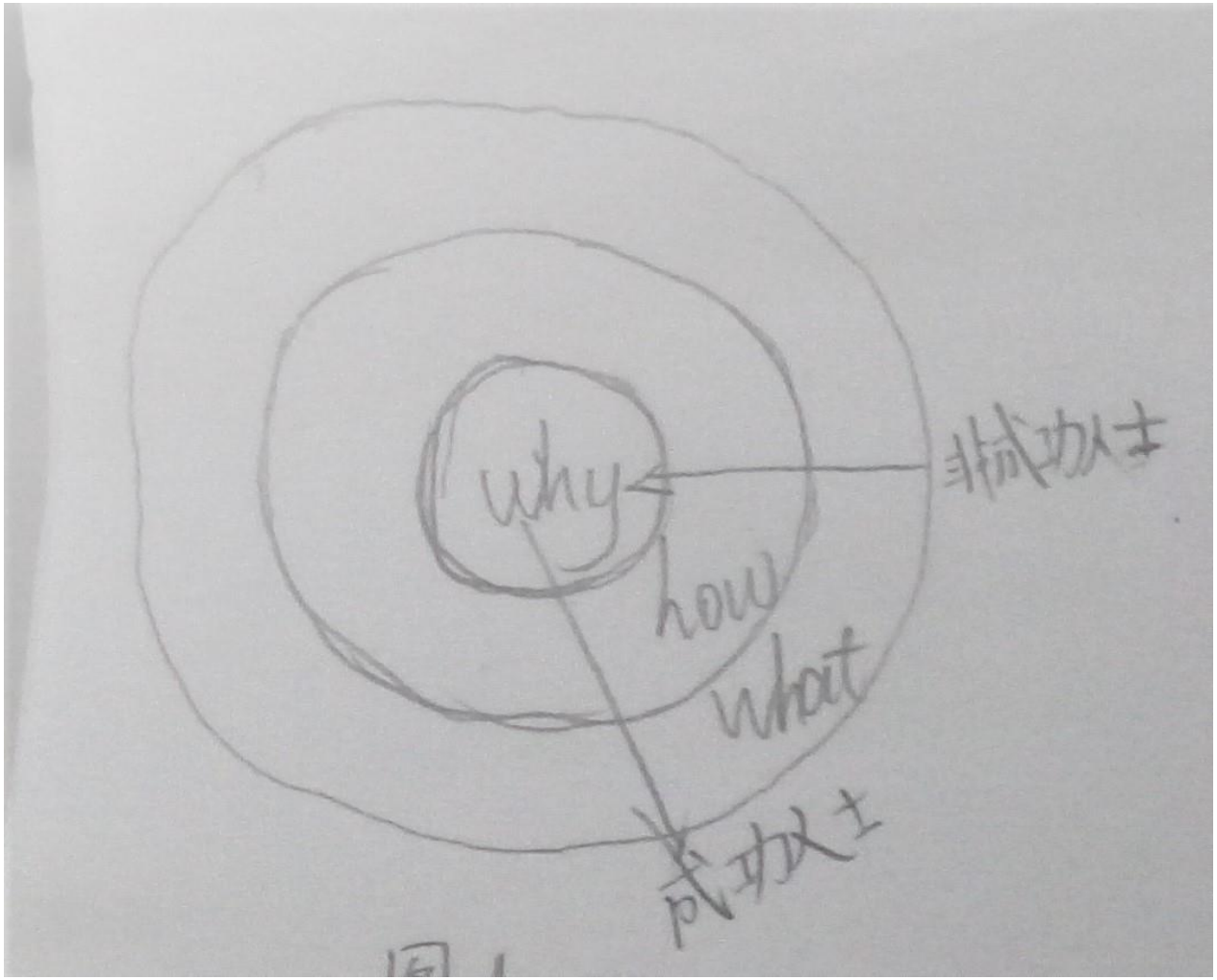
用户故事培训

2019-04

www.zhizhuochina.com

❖目录

❖成功者的思维方式



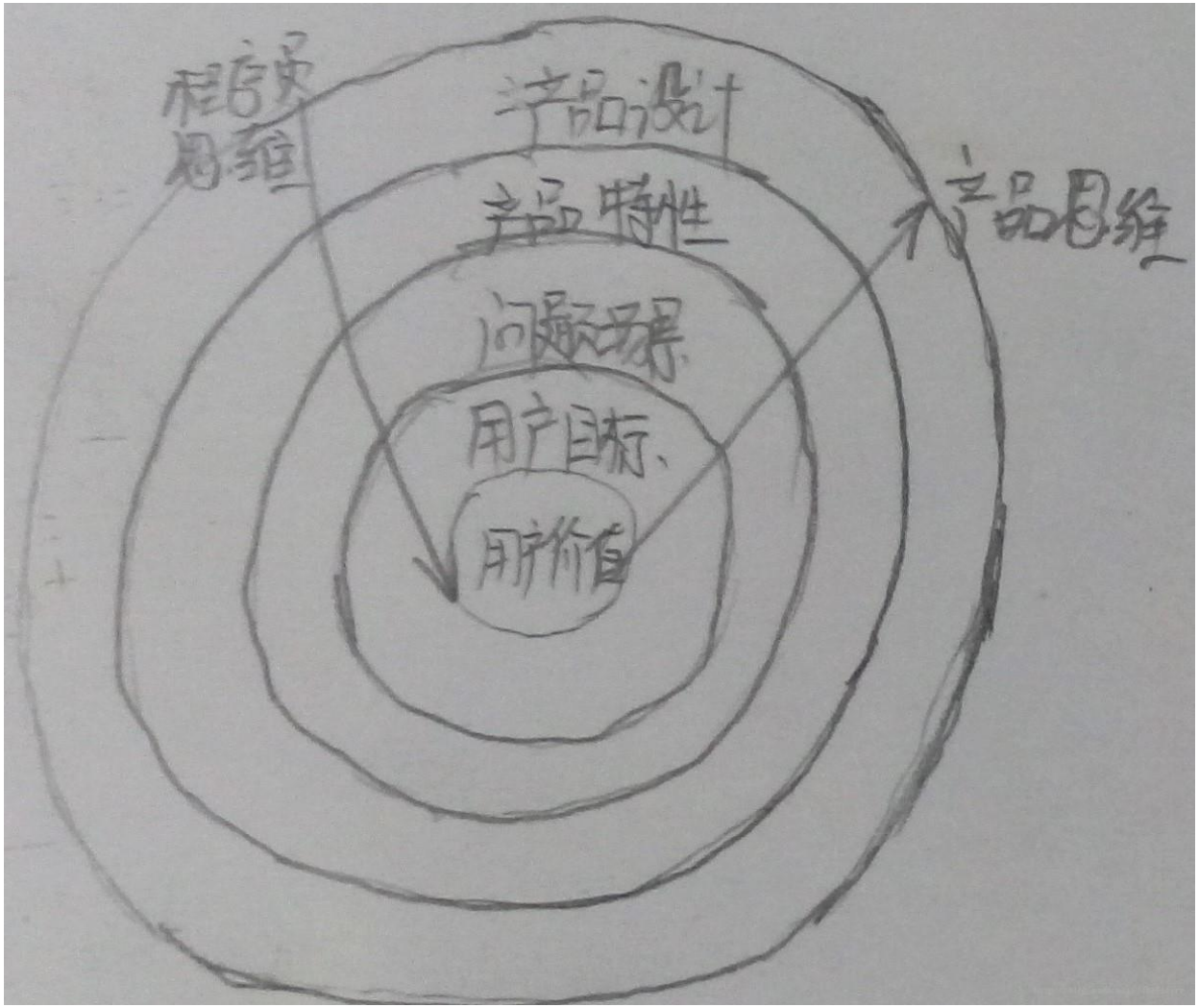
❖系统角度的需求分解与描述

业务领域	业务模块	业务单元	业务场景（操作）
产品管理	需求管理	变更管理	变更申请
			变更审批
			…
		…	
	…		
	…		
项目管理	…		
…	…		

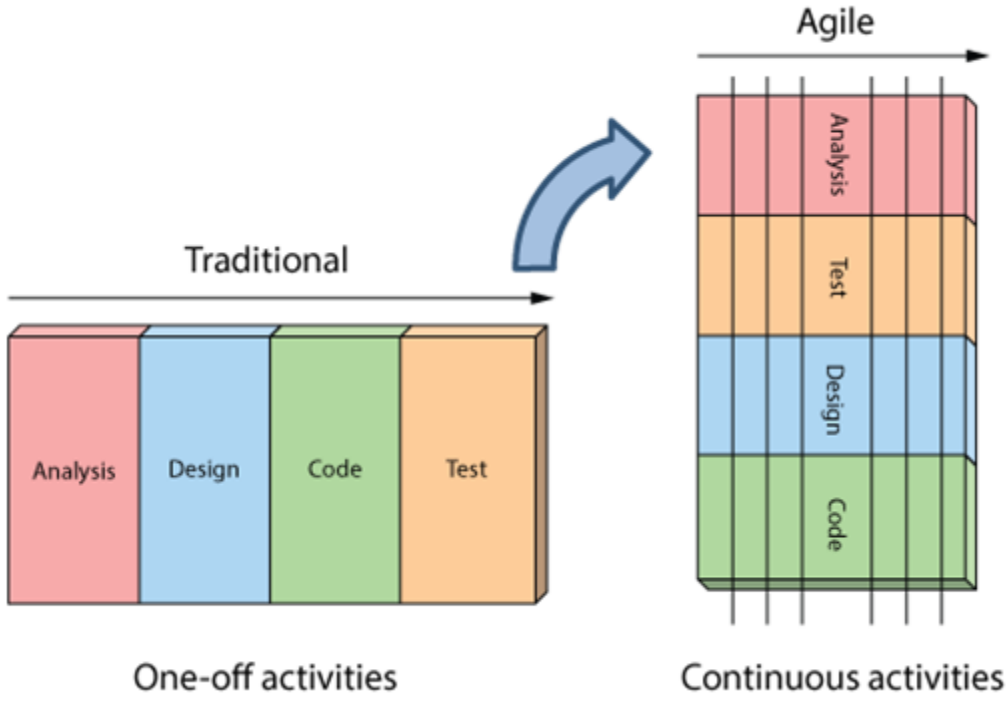
为什么要做？能给客户带来什么价值？

“我们XXX模块当前的处理流程是什么样的，这个需求就是要增加YYY结构和字段，ZZZ流程和操作”。

❖两种思维方式



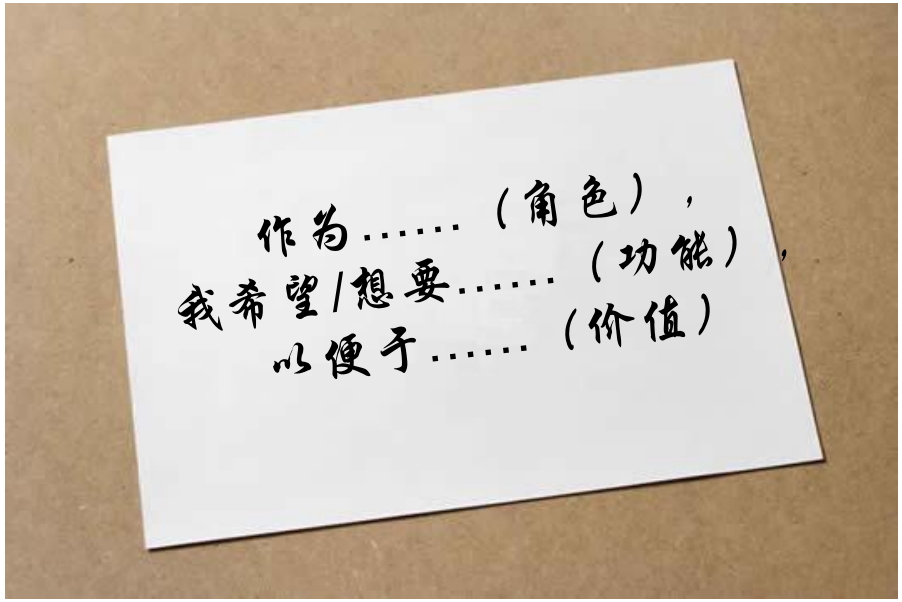
❖为什么是用户故事



- 真正的敏捷开发必须是基于用户故事的开发过程。
- 用户故事就是含有一定业务价值的端到端交付。
- 通过它更早地获取用户反馈，从而确定产品的正确性。

❖STORY的定义

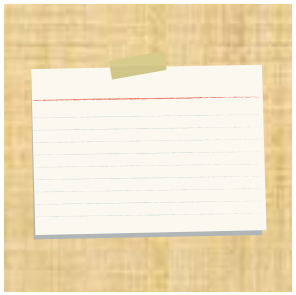
- 一个用户STORY描述了一个对客户有价值的、简单的、端到端的功能点。
- STORY可以理解成能独立交付，能够感知的最小需求。



❖Story是什么?

3C

- Story描述了对于系统或软件的客户或用户有价值的功能片段
- Story由三个方面组成
  - 一个**书面**的Story简短描述，用来做计划并在开发过程中起到提醒的作用
  - 针对Story描述进行的**交流**，用来澄清Story的细节
  - 记录和传递细节的测试信息，用来**确定**Story是否开发完成



卡片（Card）

交流（Conversation）

确认（Confirmation）

❖STORY的特征

❖STORY的特征-独立性

Independent独立的——INVEST

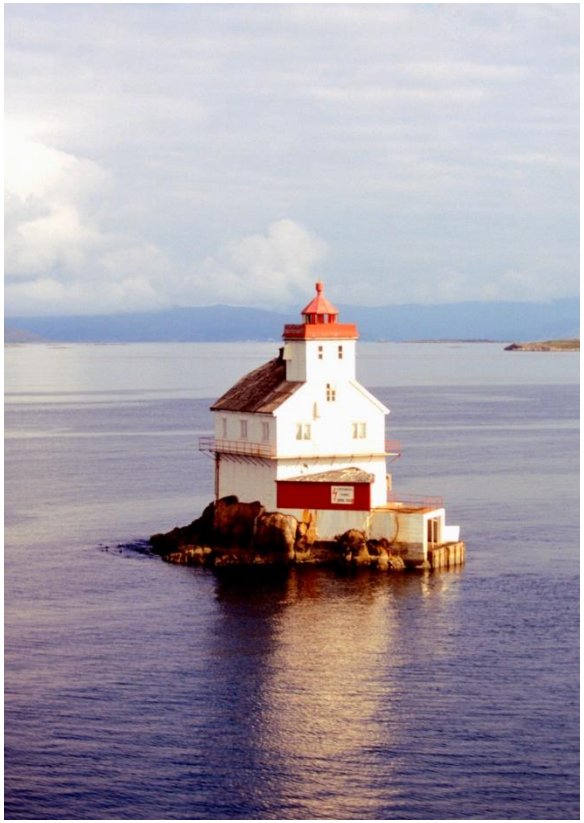
- 故事间的依赖关系会给优先级划分和计划带来问题。
- 故事间的依赖关系也会使估计变得更加困难。

❖STORY的特征-独立性

- 如果出现这类依赖关系，可以用以下两种方法规避该问题：
  - **将具有依赖关系的故事合并为一个独立的故事**
  - **根据另外一种方法划分故事**
- 如果需要编写关于用户如何支付在网站上购买的图书的故事，可以编写出以下故事：
  - **用户使用Master Card支付**
  - **用户使用Visa Card支付**
  - **用户使用Amex Card支付**
- 对于上述故事，可以采用按另一种方法划分为：
  - **用户可以使用一种类型的信用卡支付**
  - **用户可以使用两种附加类型的信用卡支付**

❖为什么Story应该是相互独立的

- 尽可能的避免在Story之间引入依赖关系，因为
  - 依赖关系会导致优先级和计划的问题
  - Story之间的依赖还会使估计工作更加困难
- 复杂系统的依赖关系不可避免
  - 独立性的引申意义：好的Story划分应该使开发人员在安排计划的时候，可以更加灵活，方便地进行调整

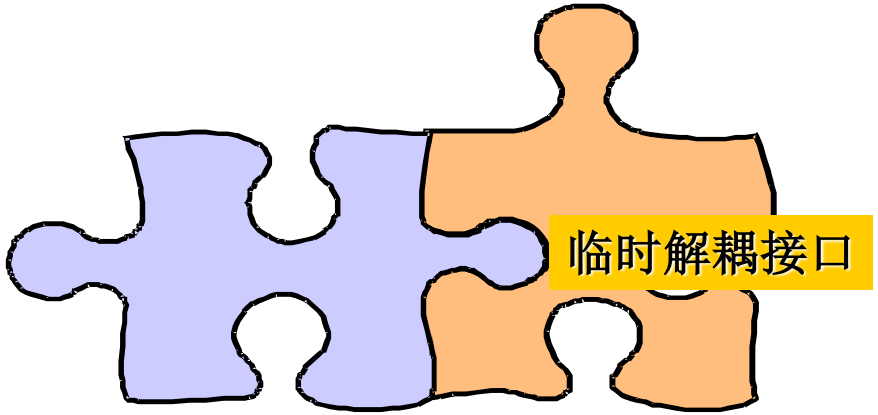


耦合谱系

数据	印记	控制	外部	共用	内容

❖形成独立的Story的方法 – 临时解耦

- 对于相对比较松散的数据依赖，可以采用临时解耦的方式来暂时解除依赖
- 临时解耦：对于共享数据的不同Story，以临时的数据配置和读取接口来临时解除Story之间数据的依赖关系。
- 注意：当这些Story都完成后，需要去除临时接口，并更新相关测试
- 以少量附加工作量来换取更大灵活性



❖STORY的特征-可协商

Negotiable可协商的——INVEST

- 故事是可协商变化的，而不是固定不变的。
- 故事卡是对功能的简要描述，功能的细节需要通过开发团队和客户协商来确定。
- 故事卡不需要包含细节化的需求描述，它是用来提醒开发团队与客户进行交流的。

❖STORY的特征-可协商

用户可以使用信用卡支付

接受Visa Card，Master Card和Amex Card。

我们是否接受Discovery Card?

对于用户界面：不需要提供选择信用卡类型的字段，系统可以根据卡号的前两位确定信用卡的类型。

❖回归本源：Story是支持交流的工具

- Story的内容应该包括：
  - **少量的简洁的语句用作关键内容的提示**
  - **关于需要在交流中解决的问题的注释**
- 细节->测试信息
  - Story卡的背面的测试点
  - 临时记录（白板，笔记，IT工具，Wiki）

正面

用户可以登录Web管理页面

注释：
- 用于金融系统，极高安全性
- 密码可以为空吗？
- 是否允许admin以外的用户登录？
- 是否允许并发的登录？

反面

- 在密码合法和不合法情况下，测试局域网登录方式
- 用空密码测试局域网登录方式
- 测试无线网络登录方式（允许/不允许）
- 测试Internet登录方式（允许/不允许）
- 用缺省密码测试Internet登录

❖STORY的特征-有价值

Valuable对客户或用户有价值的——INVEST

- 避免制造出只有开发人员认为有价值的故事，例如以下故事：
  - **所有和数据库间的连接都通过连接池建立。**
  - **通过一些通用的类来完成所有的错误处理和记录。**
- 应改写以上故事，让故事明显地体现出给客户或用户带来的好处，这样做，用户便可以根据这些好处思考应如何为故事设定优先级，按优先级将故事划分到开发日程中。以下为改写后的故事：
  - **不超过50位用户可以通过5个license使用该应用程序。**
  - **所有的错误都呈现给了用户并持续地进行了记录。**

❖STORY的特征-可估计

Estimatable可估计的——INVEST

- 故事不可估计通常是由以下3类原因造成的：
  - **开发人员欠缺业务知识**

    如果开发人员欠缺领域知识，他们应与客户讨论，不断更新故事。
  - **开发人员欠缺技术知识**

    如果开发人员欠缺技术知识，他们可以通过样例或者简单的小程序快速体验该技术——在极限编程中，这种行为称为探究（spike）。在这种情况下，应将故事划分为两部分——一部分针对快速探究，另一部分针对实际的工作。
  - **故事过大**

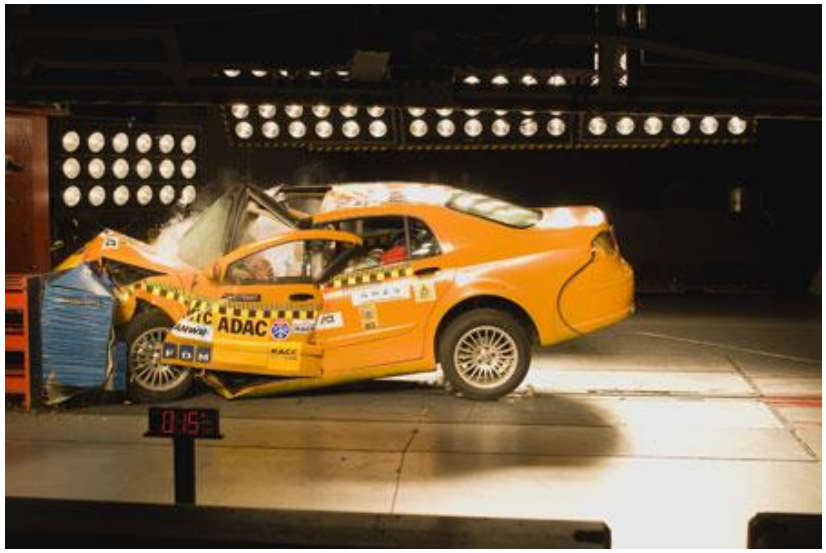
    如果故事过大，开发人员需要将故事分解为小一些的故事。

❖STORY的特征-短小的

Small短小的——INVEST

- 如果故事过大或过小，就无法用它们做计划。
- 在一个旅行预定系统中，“一个用户可以制定度假计划”是一个粗粒度的故事。制定度假计划是旅行预定系统的一个重要功能，但它包含了多项活动，因此应将该其分解为多个较小的故事。
- 粗粒度的故事分为两类：复合故事和复杂故事。
- 复合故事是一个包含多个短小的故事的粗粒度故事。
- 复杂故事比较大但不易分为多个短小的故事。如果故事所带有的不确定性增加了其复杂性，应将故事分解为两个故事，一个是为了调查研究，一个是为了开发新特性。

❖STORY的特征-可测试



Testable可测试的——INVEST

- 故事的编写必须遵循可测试的原则。故事能够成功地通过测试证明故事开发成功；如果故事不能被测试，开发人员就无法得知何时才算完成了代码编写。
- 无法测试的故事通常为针对非功能需求的故事，这些故事是关于软件的需求，但是又不是功能方面的需求。以下为一些无法测试的故事的示例：
  - **用户必须觉得软件容易使用。**
  - **用户永远不必长时间等待任何界面的出现。**

❖小结

1. 理想情况下，Story之间应该相互独立。有时候这是不可能的（特别是大型复杂系统），但是从其引申意义来说，好的Story划分应该使开发人员在安排计划的时候，可以更加灵活，方便地进行调整
2. Story的细节是通过客户和开发人员之间的沟通得到的
3. Story对于客户或用户的价值应该是清楚的。达到这个目的最好的方式是客户来编写这些Story
4. Story可以包含一些关键点的注释，但是太多细节会反而会导致Story的意图变得模糊，并且会在开发人员和客户沟通的时候造成没必要进行沟通的印象
5. 针对Story的验收测试点是对Story进行注释的最好方式之一
6. 太大的Story不利于迭代计划和工作分配，混合和复杂的Story可以拆分为多个更小的Story
7. 太小的Story可能会降低效率，所以多个微小的Story可以合并为一个更大的Story
8. Story必须是可测试的

❖Story不能包打天下

- 你需要采用不同于Story的方式来表达某些需求
  - 用户界面指导原则经常用包含了大量屏幕截图的文档方式进行描述
  - 重要的系统之间的接口说明

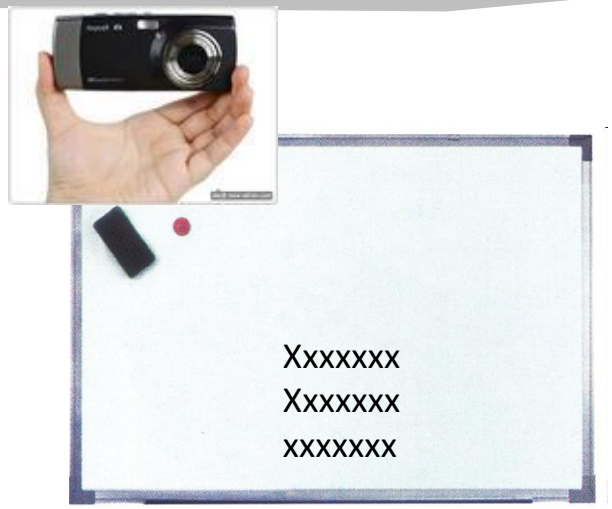
- 如果你发现系统某方面的内容采用其他的方式描述好处更多，那就采用这种方式。

❖STORY的形式——卡片

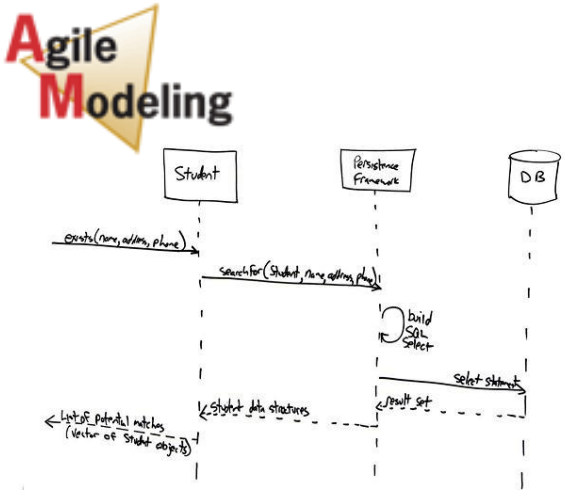
❖STORY的形式——表单

Story ID	Story Title（Story标题）	Story Description（Story 描述）	Dependency（依赖）	Non-func constraint（非功能约束）	Acceptance Test Description（可接受性测试描述）		
					ID	Conditions/Inputs/Triggers（条件/输入/触发）	Expected behavior/output（预期行为/输出）
JOB_SEARCH_04	用户可以搜索相关工作职位	用户可以根据各种搜索条件，显示出搜索到的工作职位的信息。 备注：搜索条件应该包含工作职位的描述，薪酬。 职位信息包括：公司名称，工作地，薪酬，职责，经验要求。			ATC_JOB_SEARCH_01	搜索工作职位描述为空的，	结果为空
					ATC_JOB_SEARCH_02	搜索一个很长的工作职位描述，	提示输入超长
					ATC_JOB_SEARCH_03	搜索没有具体薪酬的工作职位，	显示符合工作描述的工作。
					ATC_JOB_SEARCH_04	搜索六位数的薪酬，	显示符合要求的工作。

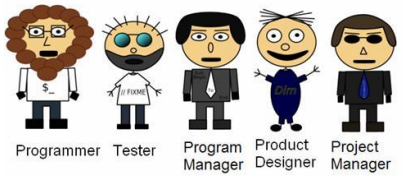
❖要是讨论完了忘掉了怎么办?



白板 + 拍照



敏捷建模



Wiki文档

细节在Story讨论的非正式的记录中!

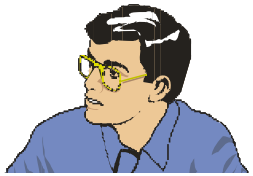
真的不写文档吗?!

❖STORY和其它需求文档的区别

开发团队

沟通

客户

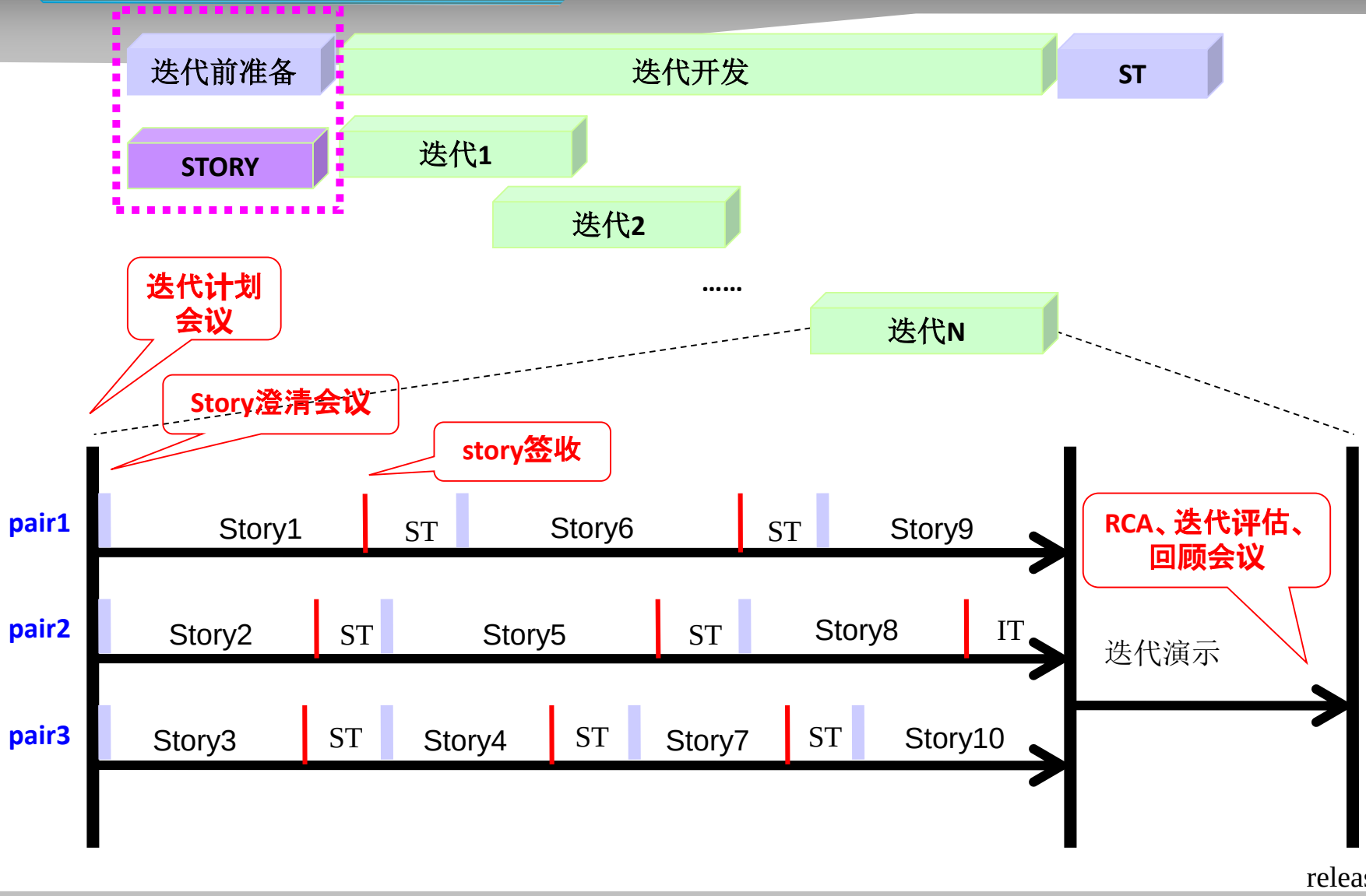


STORY

不断与客户沟通，取得双赢的结果

❖目录

❖写作STORY的时机



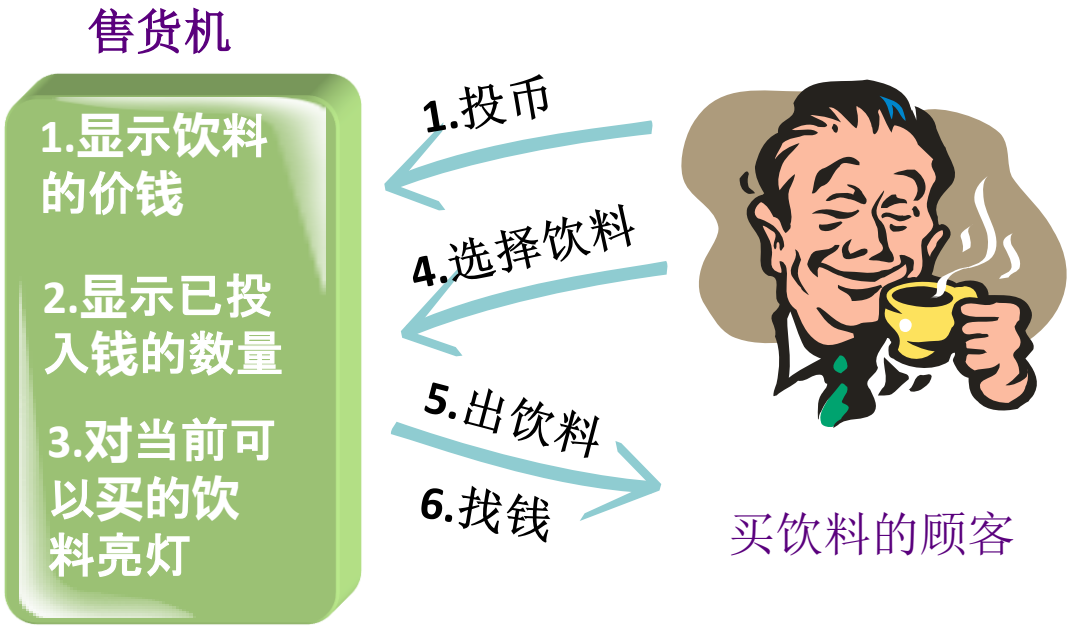
❖Story划分准则

- 专注于功能（即用户可感知的需求），不应该过分注重用户界面等细节；
- 相对独立；
- 只是用来简单的描述系统功能，供开发人员进行估计开发进度，在开发过程中开发人员和用户会不断的交流以讨论细节问题；
- 一般一个Story要求500行以内，但是不应该局限于这个数字；
- 区分外部Story(同客户交流)和内部Story（涉及底层功能的）；

❖STORY的写作步骤

1. 识别客户
2. 整理业务流程
3. 将业务流程中的功能点对应到STORY

❖STORY的写作步骤——售货机软件

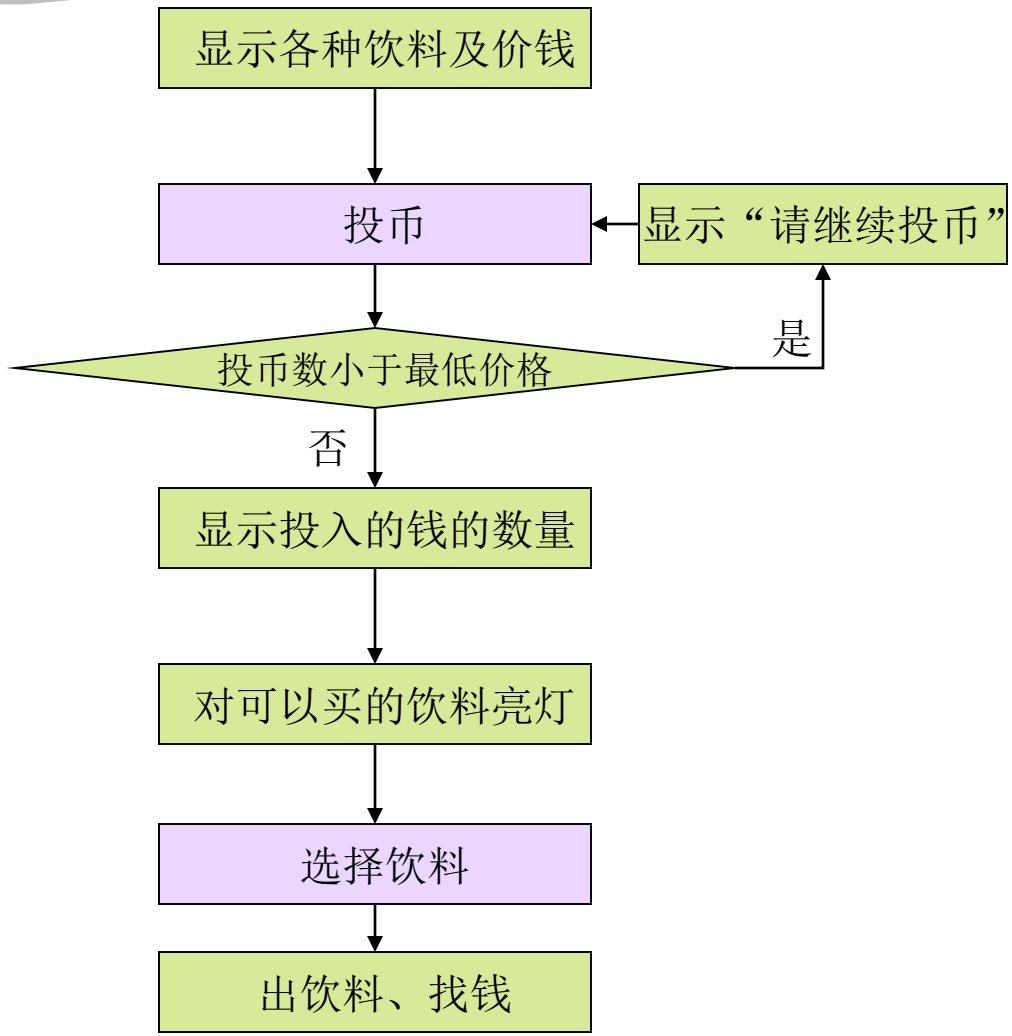


❖STORY写作步骤——第一步：识别客户

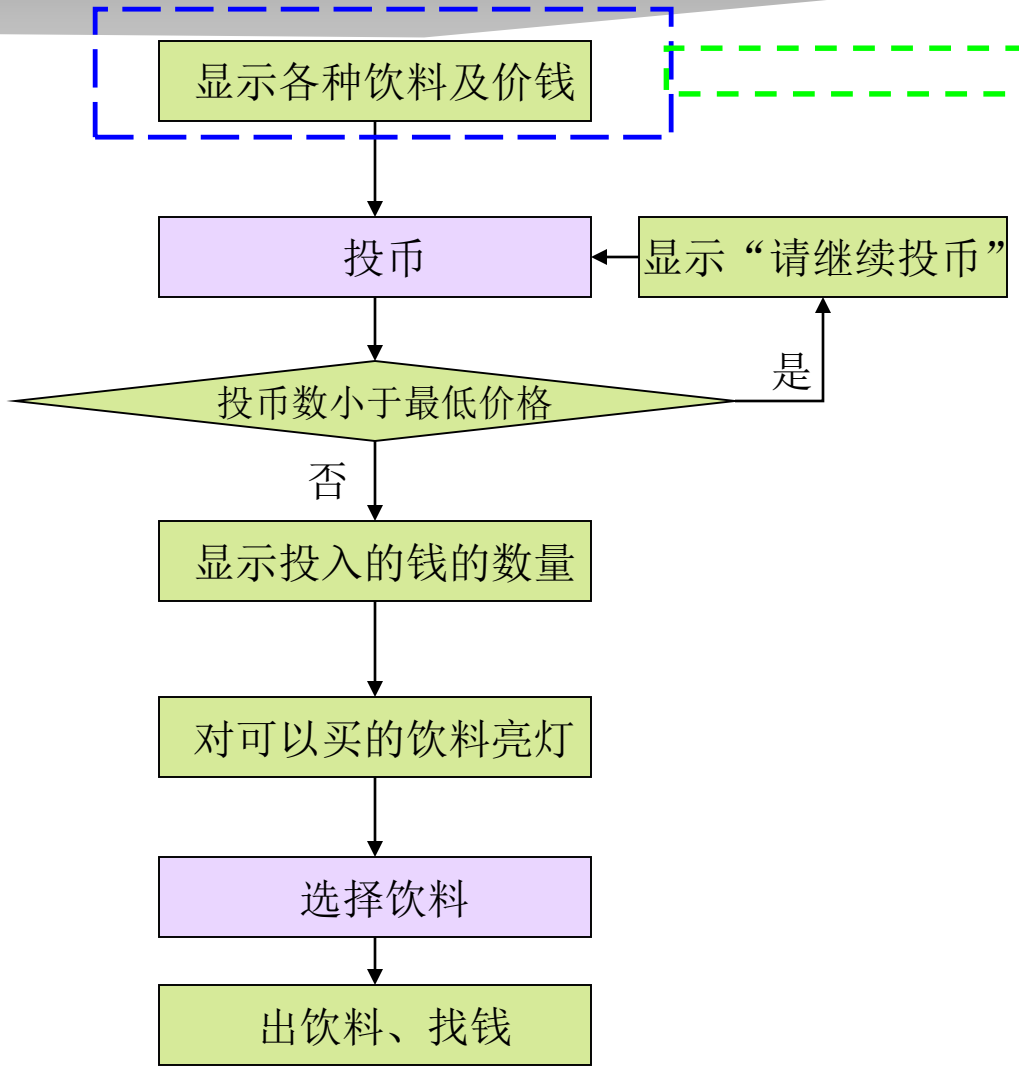
买饮料的顾客

饮料自动售货机制造商的市场经理

❖STORY写作步骤——第二步：整理业务流程

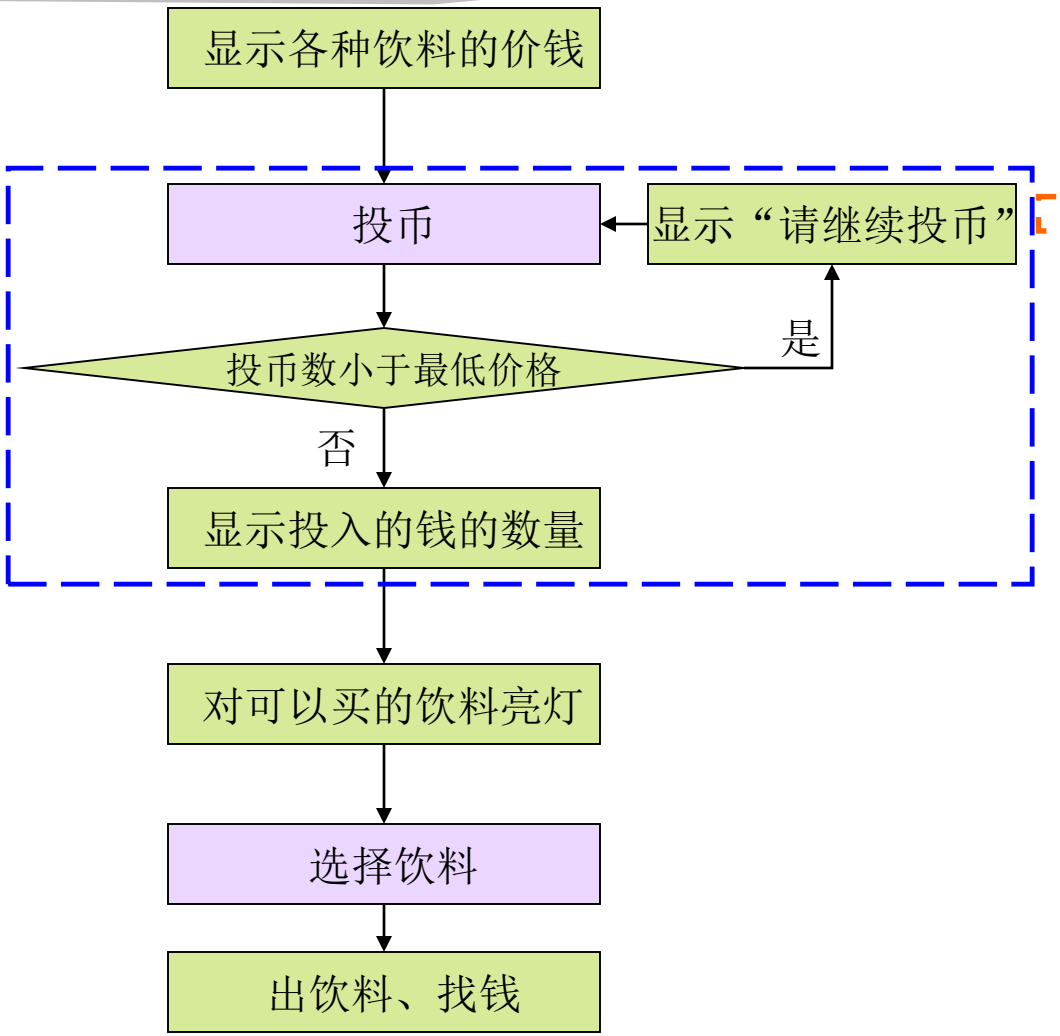


❖STORY写作步骤
——第三步：将业务流程中的功能点对应到STORY



作为：顾客
我要：看到饮料及价格
从而：为我提供指示和方便

❖STORY写作步骤
——第三步：将业务流程对应到STORY

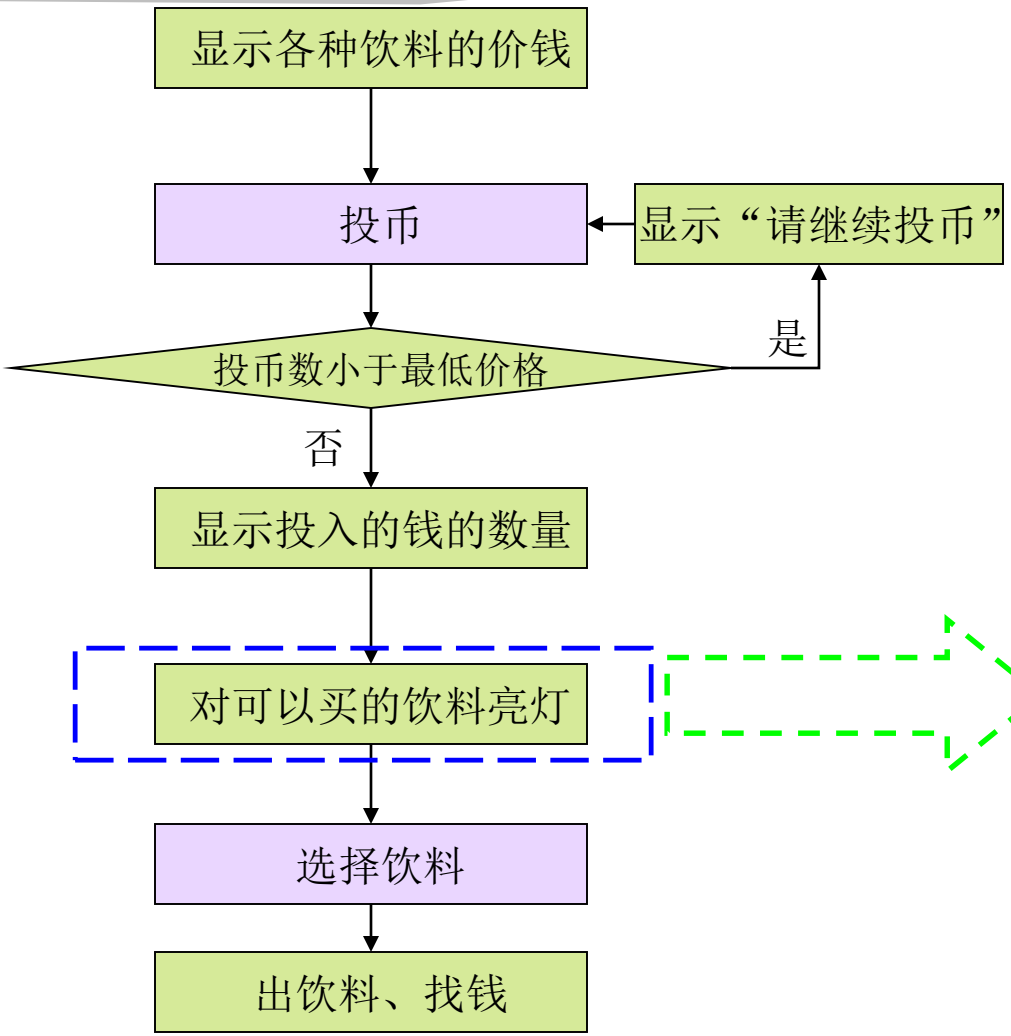


作为：投资方

我要：让顾客可以向售货机中投币

从而：顾客有渠道买我的饮料，就可以赚钱

❖STORY写作步骤
——第三步：将业务流程对应到STORY

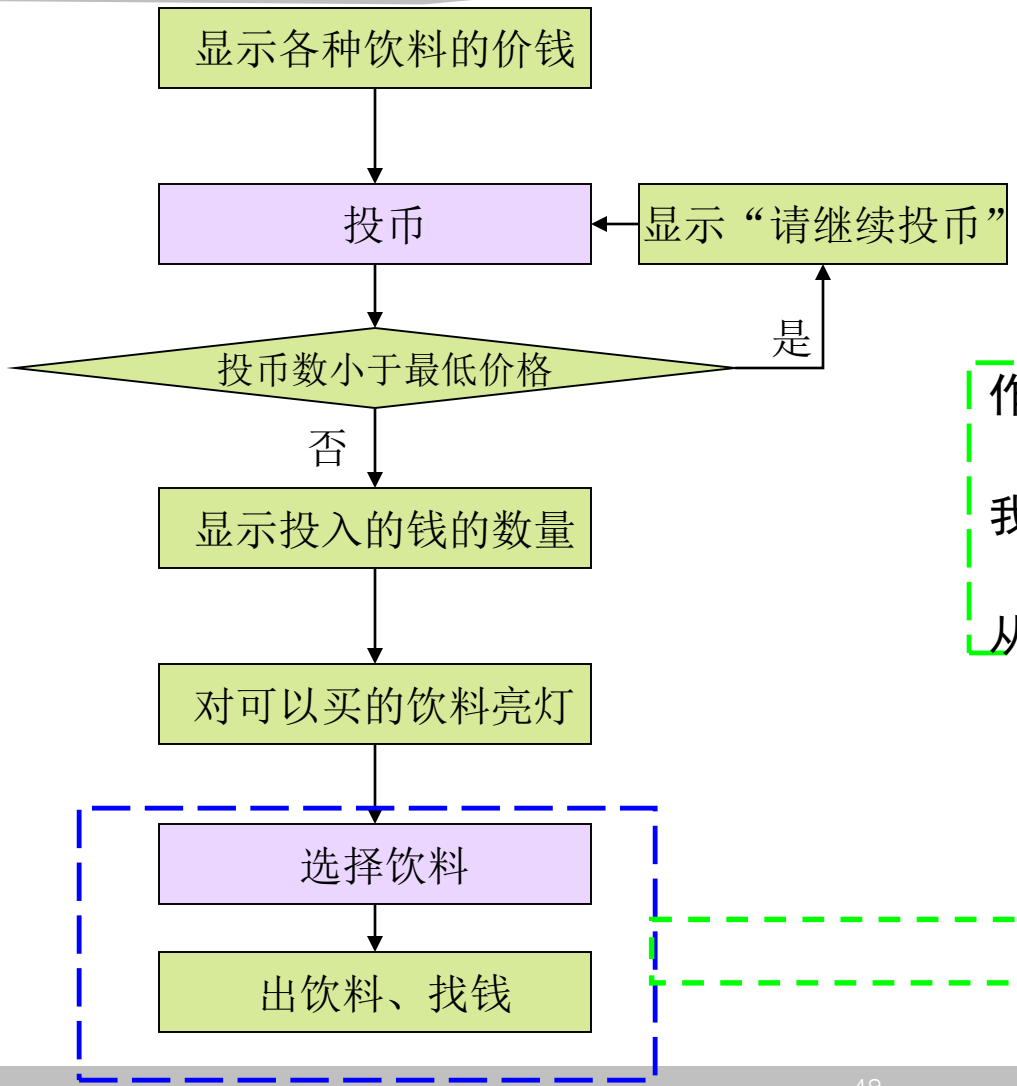


作为：顾客

我要：投币之后，能显示付得起的饮料

从而：我会觉得很方便

❖STORY写作步骤
——第三步：将业务流程对应到STORY



作为：顾客

我要：售货机可以给我出饮料并找钱

从而：我就可以解渴并不会损失钱了

示例

- 作为一个手机用户，我希望手机导航能进行语音播报，从而使我更方便地使用导航

标准化格式

- 作为……，我希望……，从而……
- 文字游戏?
- 三要素：用户角色、用户需求、用户价值

用户角色

- 作为……（Who Need It？）
- Really Need It?
- Need More?

示例

- **作为一个手机用户，我希望能手机导航能进行语音播报，以便我更方便地使用导航**
- **作为一个驾驶员，我希望能手机导航能进行语音播报，以便我更方便地使用导航**

❖利用该方法的两个例子

用户价值

- 以便……（Why Need？）
- Another Way?
- Need More?

示例

- 作为一名驾驶员，我希望手机导航能进行语音播报，以便我更方便地使用导航。

Bad

- 作为一个驾驶员，我希望手机导航能进行语音播报，以便我在驾驶的过程中更安全地使用手机导航。

Better

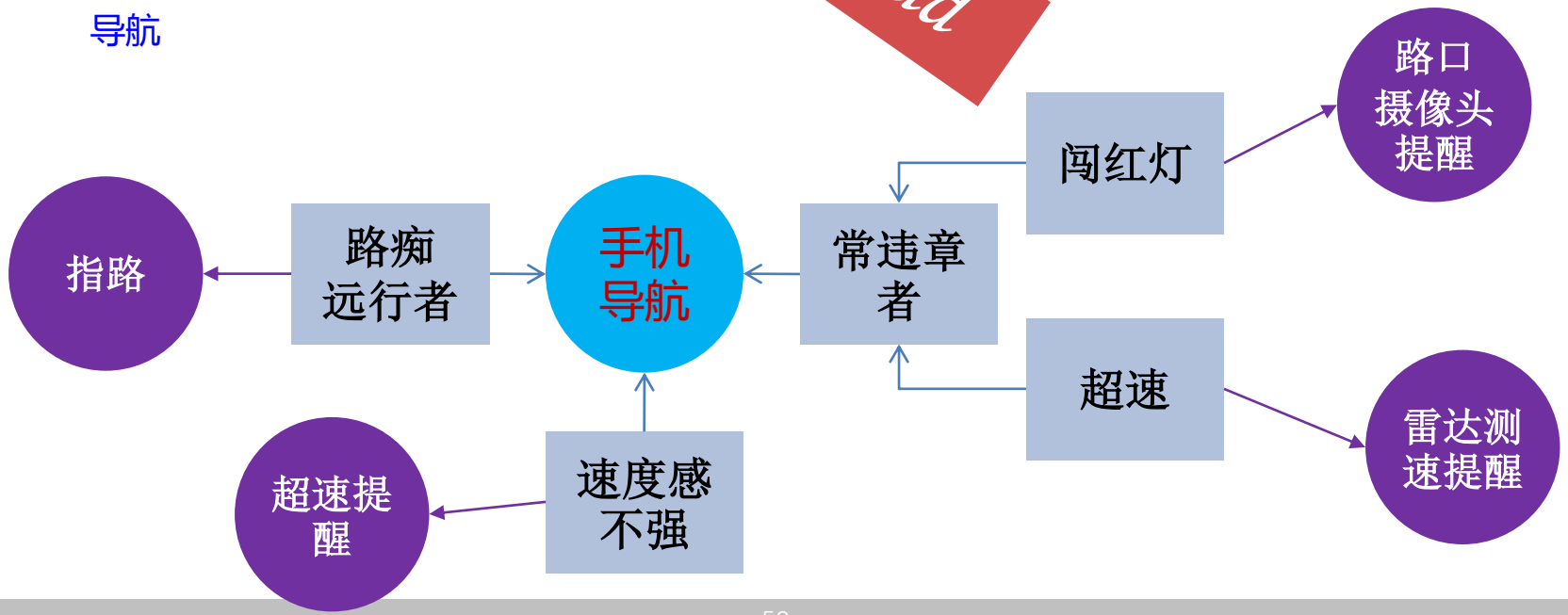
单一价值

- Why Do Them Need?
- How To Meet Them?

示例

- 作为一个驾驶员，我希望手机导航能进行语音播报，以便我在驾驶的过程中**更安全**地使用手机导航

Bad



示例

① 作为一个**路痴（远行）**的驾驶员，我希望手机导航能用语音播放方向指示，以便在更**安全地**通过手机导航**了解**下一步**行驶方向**

② 作为一个**新手（对车速不敏感）**的驾驶员，我希望手机导航能在我超速的时候及时提醒我减速，以便我能随时将车速**控制**在**安全时速**范围内

③ 作为一个**爱闯红灯**的驾驶员，我希望手机导航能用语音进行闯红灯拍照提醒，以便我能够有选择性地闯红灯，**减少被记闯红灯违章**

④ 作为一个**爱超速行驶**的驾驶员，我希望手机导航能用语音提醒前方有雷达测速装置，以便我能够及时减速，**减少被记超速违章**

Better

Better

❖如何将用户故事拆分成任务

Why

- 设计
- 计划

How

- 架构
- 依赖
- ……

示例

- 描述

  作为一个彩民我希望有一个能自动生成彩票号码，并自动通过短信投注的功能。以便提高机选中奖几率，并方便地进行短信投注。

- 验收条件

  1. 能显示往期开奖号码（每次显示20期，通过点击“下一组”附加显示后20期，按照时间先后排序显示）
  2. 能手动输入投注号码、注数，完成输入后可点击“自选投注”向购彩中心发送短信
  3. 能点击“机选号码”，自动生成随机彩票号码（蓝球在往期（最近100期）蓝球概率前十的数字中随机生成），如果往期蓝球总数少于10个大于5个，则在实际产生的数量中生成随机数，如果小于等于5个，则在实际产生的蓝球外生成随机数。红球先在奇数区全随机生成3个，再在偶数区全随机生成3个。
  4. 点击“机选号码”后能显示生成的号码，并出现注数输入框，可输入投注注数，再点击“机选投注”向购彩中心发送短信。也可点击“机选号码”重新生成投注号码。
  5. 点击“全自动投注”可直接自动生成一注彩票，并发送购彩短信。自动生成方式同“机选号码”一致。

MVC

- 视图(View)代表用户交互界面，不包括在视图上的业务流程的处理
- 模型(Model)就是业务流程/状态的处理以及业务规则的制定

  业务模型还有一个很重要的模型那就是数据模型。数据模型主要指实体对象的数据保存（持续化）
- 控制(Controller)可以理解为从用户接收请求, 将模型与视图匹配在一起，共同完成用户的请求

❖如何将用户故事拆分成任务

结果

M:（数据、规则）

- 设计一个类，用于描述投注号码
- 设计一个类，用于描述开奖号码
- 设计一个用于保存往期开奖号码的数据库
- 设计一个类，用于描述投注短信的内容

V：（视图、界面）

- 设计主界面（“查看往期”“自选输入框”“自选投注”“机选号码”“全自动投注”）
- 设计往期号码查看界面（“往期号码列表”“下一组”“返回”）
- 设计点击“机选号码”后的界面（“生成的号码”“注数输入框”“机选投注”）

C：（控制逻辑）

- 设计一个类用于从收到的短信中提取开奖号码，并将号码保存到数据库中。
- 设计一个类用于从数据库中读取往期号码，创建往期号码列表，并能将列表中的数据显示在屏幕上。
- 设计一个类，用于生成自选号码与注数，并生成一条投注短信。
- 设计一个类，用于生成机选号码与注数，并生成一条投注短信。
- 实现机选投注功能。
- 实现全自动投注功能。

非功能需求的范围

性能需求：

资源占用：内存、CPU；

操作时间：服务起停时间、具体操作时间；

管理规模：网元数、用户数；

架构、实现约束：

遵循C/D/S架构；

业务实现与通讯总线分离；

❖非功能需求STORY

Title

需求简单标题

Target

需求描述或实现约束

Guideline

为实现上述需求，需要做哪些工作

Verification method / Frequency

验证是否满足上述非功能需求的方法，多长时间进行一次验收

Who owns this

谁负责确保需求实现、验收

❖非功能需求STORY样例

Title：支持500个并发活动会话

Target：支持500个用户登录到网管系统中（不要求同时登录），并保持活动状态（即500个用户都处于工作状态）

Guideline

1、会话心跳事件每2s发送一次对系统冲击大，将心跳延长到5s；

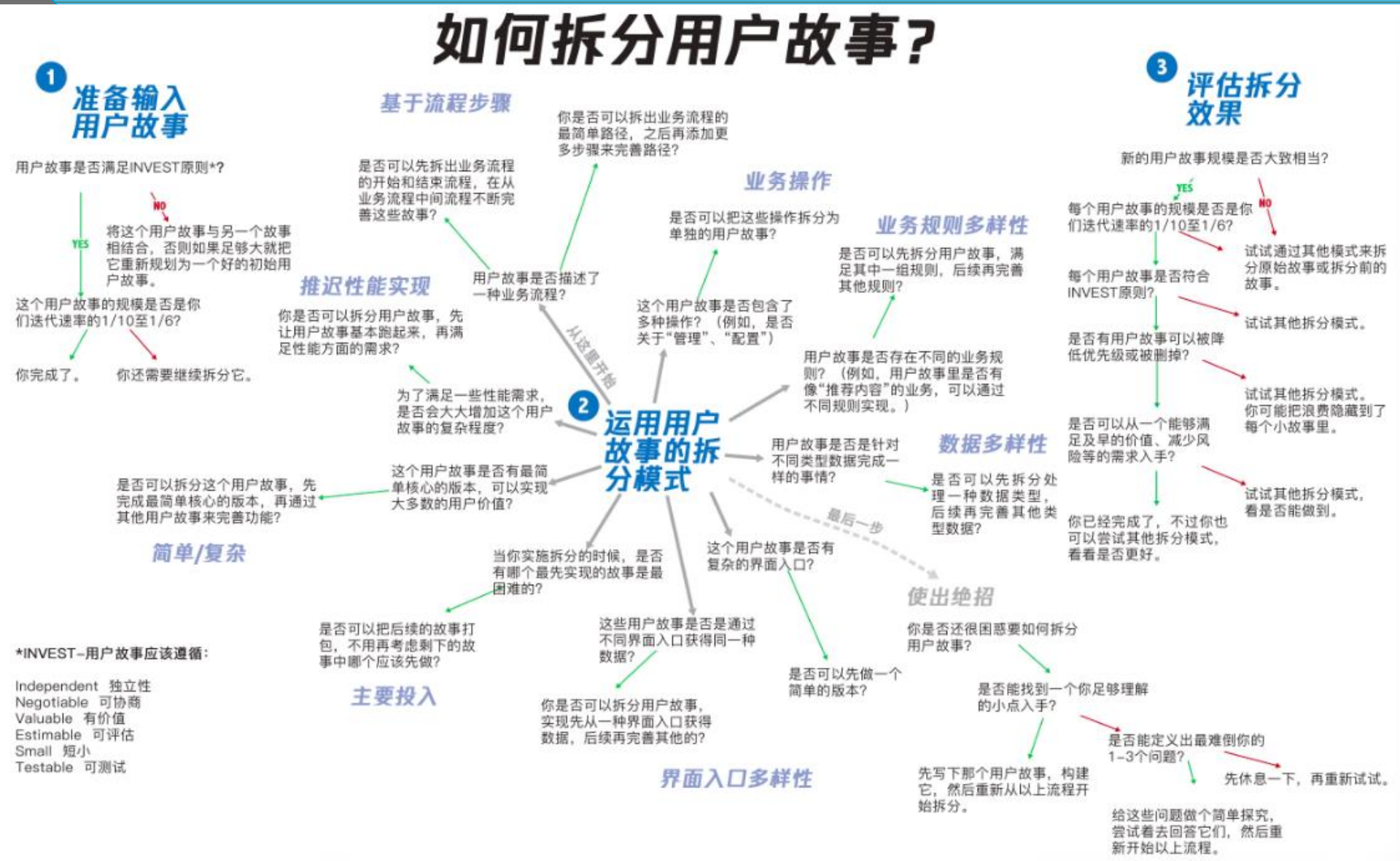
2、在DS层对心跳进行收敛， DS上每5s对Client发来的心跳进行汇总成一个心跳，发送到AS；

Verification method / Frequency

1、系统中保持500个会话活动，监视系统处于空闲状态时C/D/S各部分的CPU占用比例小于10%；频率：每个迭代验证一次；

2、测试心跳间隔在5s，AS上每5s收到一次心跳，包含所有客户端的心跳信息；频率：每个story验证一次；

Who owns this：TECH Leader



❖STORY拆分模式1-基于工作流

原则：

识别出用户为完成具体工作流采取的特定步骤，然后通过一些增量阶段实现工作流。

该方法也叫作最简路径法，即先拆出最简路径，再基于最简路径添加步骤，直到覆盖完整路径。

例子：

作为内容管理员，我可以在企业网站上发布新闻故事。

- 我可以直接在企业网站发布新闻故事。
- 我可以发布经过编辑评审的新闻故事。
- 我可以发布经过法律顾问评审的新闻报道。
- 我可以查看测试站点上的新闻故事。
- …我可以将测试站点评审通过的新闻报道发布到产品站点。

❖STORY拆分模式2-业务操作

原则：

有些用户故事使用了“管理”、“控制”等词汇，它掩盖了对故事执行的多种操作，大的用户故事可以基于不同类型的操作进行故事拆分。

例子：

作为作为系统管理员，我希望能够管理使用系统的用户

- 作为系统管理员，我希望能够添加新用户，使其能够使用系统。
- 作为系统管理员，我希望能够查询当前系统都有哪些用户。
- 作为系统管理员，我希望能够修改用户的信息，方便我管理用户。
- 作为系统管理员，我希望能够删除用户，保证只有必要的人在使用系统。

作为一个用户,我可以管理我的帐户。

- ...我可以注册一个帐户。
- ...我可以编辑我的帐户设置。
- ...我可以取消我的帐户。
- ……

❖STORY拆分模式3-业务规则变化

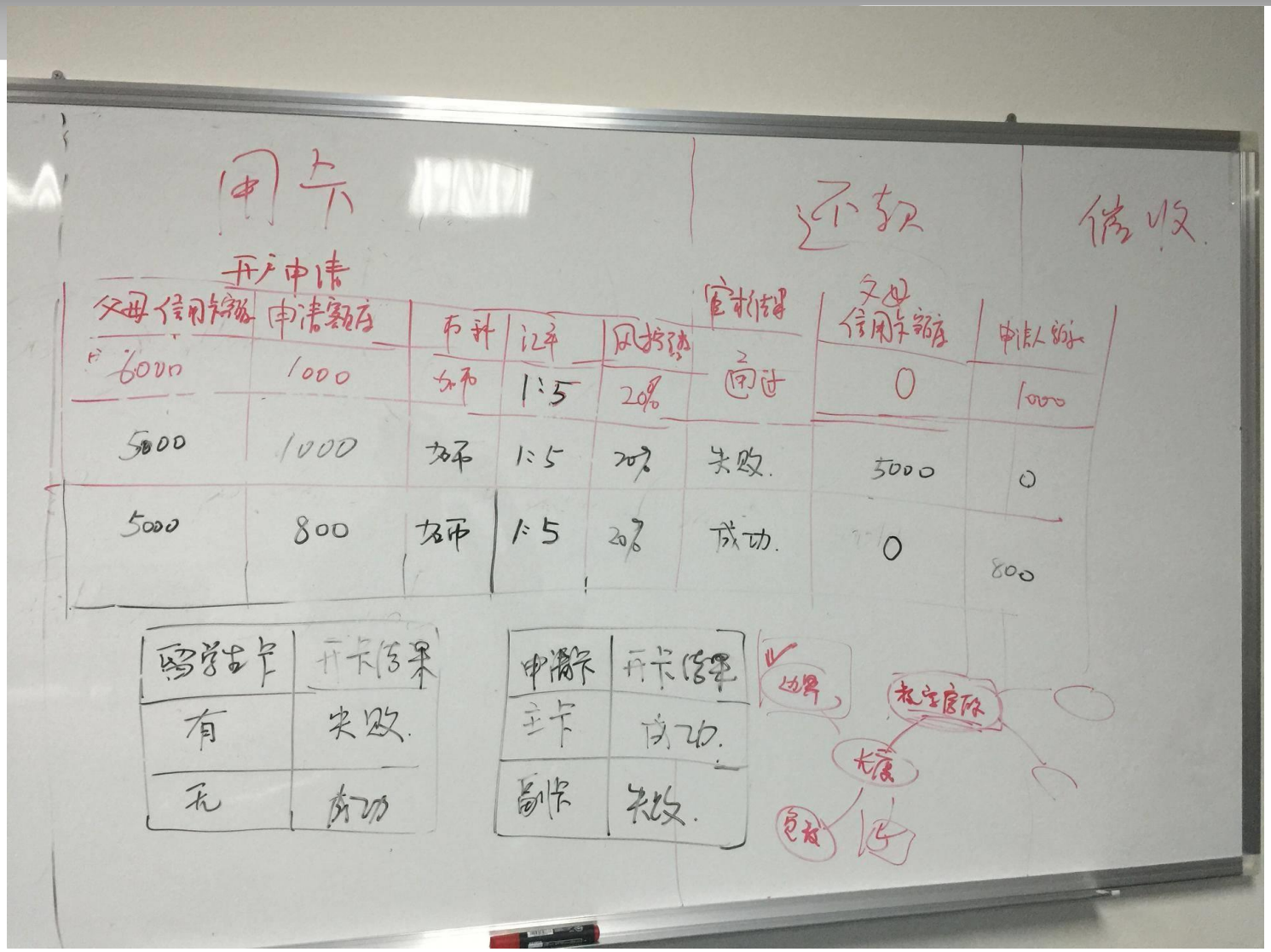
原则：

针对需求中固定的流程加载不同业务规则的情形，我们按照业务规则拆分故事。

例子：

作为网站用户，我希望网站能提供热门推荐以便我可以更快找到感兴趣的内容。

- 能根据帖子数量给出热门频道推荐……
- 能根据发帖数给出热门作者推荐……
- 能根据回帖数量给出最多评论推荐……。



❖STORY拆分模式4-数据多样性

原则：

有多种数据类型，可先实现其中一种类型，后续再完善其他。

例子：

作为用户，我希望能够查看系统的警告通知。

- 作为用户，我希望能够查看系统的异常流量警告通知;
- 作为用户，我希望能够查看系统的恶意代码警告通知;
- 作为用户，我希望能够查看系统的僵尸网络警告通知。……

❖STORY拆分模式5-接口/界面多样性

原则：

对于需求中业务流程和逻辑规则相同，仅涉及接口不同，也就是获取数据的渠道和方式不同，我们可以基于接口的差异进行故事拆分。

例子：

作为微信用户，我可以添加好友以便扩大朋友圈。

- ……我可以通过摇一摇方式添加好友……
- ……我可以通过扫二维码方式添加好友……
- ……我可以通过手写输入方式添加好友……
- ……

❖STORY拆分模式6-主要投入

原则：

根据主要投入或工作量来拆分故事。有时一个故事可以分为几部分,大部分的工作将用于实现第一个故事。

例子：

作为一个用户,我可以用维萨、万事达、大来卡或美国运通支付航班费用。

- ……我可以使用一种信用卡付费(维萨、万事达或美国运通中的一种)……
- ……我可以使用所有三种信用卡付费(假定其中一种已经支持了)……

❖STORY拆分模式7-简单/复杂

原则：

假如团队正在讨论的某个故事变得越来越大，我们可以停下来并提问：
“可以工作的最简单版本是什么?”

捕捉这一简单版本作为一个单独故事，然后把所有变体和复杂性拆解到它们各自的故事中；

首先完成核心、基本的业务价值，然后再完成其它的价值。

例子：

作为一个用户，我可以搜索两个目的地之间的航班，以便我选择预订。

- 我可以选择灵活的日期搜索……；
- 我可以选择直达或经停的方式搜索……；
- 系统默认离我最近的机场做为出发地搜索……；
- ……

❖STORY拆分模式8-推迟性能实现

原则：

不要太早考虑性能要求，记住“让它工作，然后让它更快”这个原则。

但也不要太晚。

例子：

作为用户，我希望能够在1秒内获取查询结果。

- 作为用户，我希望能够获取查询结果。
- 作为用户，我希望查询结果能够在1秒内获取。……

❖STORY拆分模式9-故事穿刺

原则：

一个故事可能比较大不一定因为它多复杂，而是由于对实施知之甚少。

我们可以在一定时间内针对怎么实施，先做个探针试验。

试验过后，知道了深浅，揭开了面纱，我们往往就可以知道如何拆分它了。

例子：

作为用户，我可以用信用卡支付。

- 调查信用卡数据处理机制;
- 实施信用卡处理。……

欢迎您的提问……

谢谢！