

# 寻找多团队管理的最大公约数

需求管理pizza模型

曾著@胡莱游戏

# Geekbang>

## 极客邦科技

整合全球最优质学习资源, 帮助技术人和企业成长  
Growing Technicians, Growing Companies

**InfoQ**  
ueue

专注中高端技术人员的  
技术媒体



**EGO** EXTRA GEEKS' ORGANIZATION  
NETWORKS

高端技术人员  
学习型社交网络



**StuQ**  
ueue

实践驱动的  
IT职业学习和服务平台



**GiT** GEEKBANG  
INTERNATIONAL  
TRAINING  
极客邦培训

一线专家驱动的  
企业培训服务



旧金山 伦敦 北京 圣保罗 东京 纽约 上海  
San Francisco London Beijing Sao Paulo Tokyo New York Shanghai

# QCon

## 全球软件开发大会

2016年4月21-23日 | 北京·国际会议中心

主办方 **Geekbang**  **InfoQ**  
极客邦科技

**7折** 优惠 (截至12月27日)  
现在报名, 节省2040元/张, 团购享受更多优惠

[www.qconbeijing.com](http://www.qconbeijing.com)



扫描获取更多大会信息

# 自我介绍

- 曾著 互爱科技 工程技术副总裁，入行16年，历经银行、证券、保险、智能硬件、互联网金融、游戏等多个行业，2010年初加入互爱科技，《胡莱三国》主程
- 互爱科技（胡莱游戏），《胡莱旅馆》、《胡莱三国》、《斩仙》、《新神曲》、《伏魔道》、《火柴人联盟》等

# 问题的背景：一个爆款

- 《胡莱三国》 2011年,登陆腾讯开放平台第7款游戏
  - SLG + 社交玩法
  - 超过4亿注册用户
  - 长生命周期，到现在还贡献收入
  - 收入创纪录



# 问题的背景：扩张中的研发管理

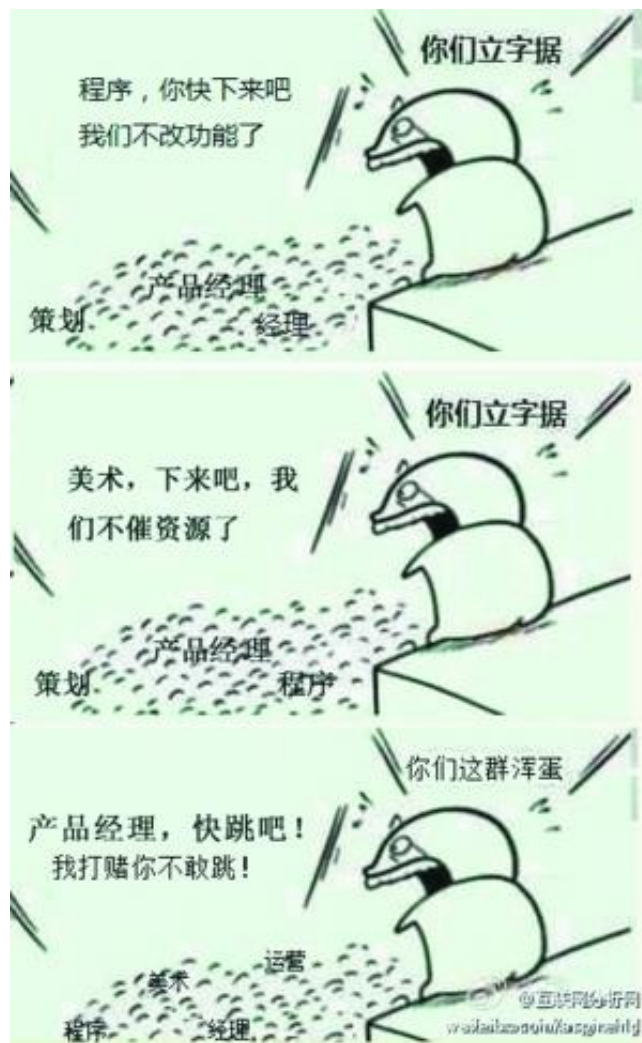
- 游戏工作室(20人) ——> 游戏制作、发行、运营的综合  
性公司(1000人)
- 业务扩张，产品类型多样化
  - 页游, 手游, 移动社交, 游戏发行平台等
- 引入了多个新团队
  - 团队风格，技术架构，研发流程各方面都有较大差别
  - 管理范围变大

# 怎么办？

- 团队风格不统一，如何推行优秀的工程实践？
- 管理范围扩大，风险失控
  - 去了解细节，疲于奔命，或者
  - 跨过细节决策，挠不到痒处

# 思路：找到研发管理的最大公约数，切入

- 共性的困难是需求管理
- 需求管理的几个目标
  - 方向及时纠错，快速失败
  - 管理优先级
  - 减少等待
  - 减少理解偏差
  - 协助估算
  - 方便进度跟踪





# 需求管理的Pizza模型

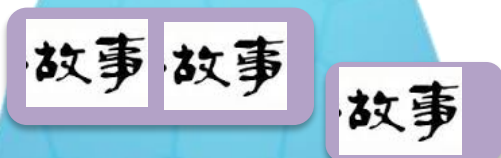
产品愿景



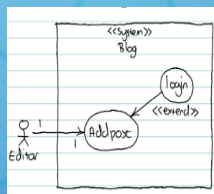
用户画像  
核心价值



模块  
用户故事



低保真模型  
系统交互  
概念模型



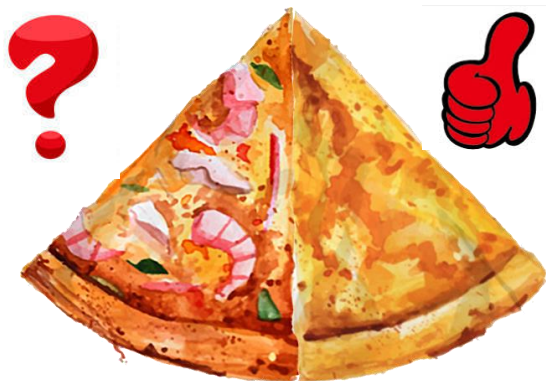
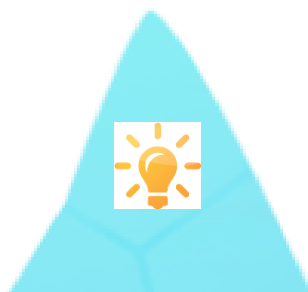
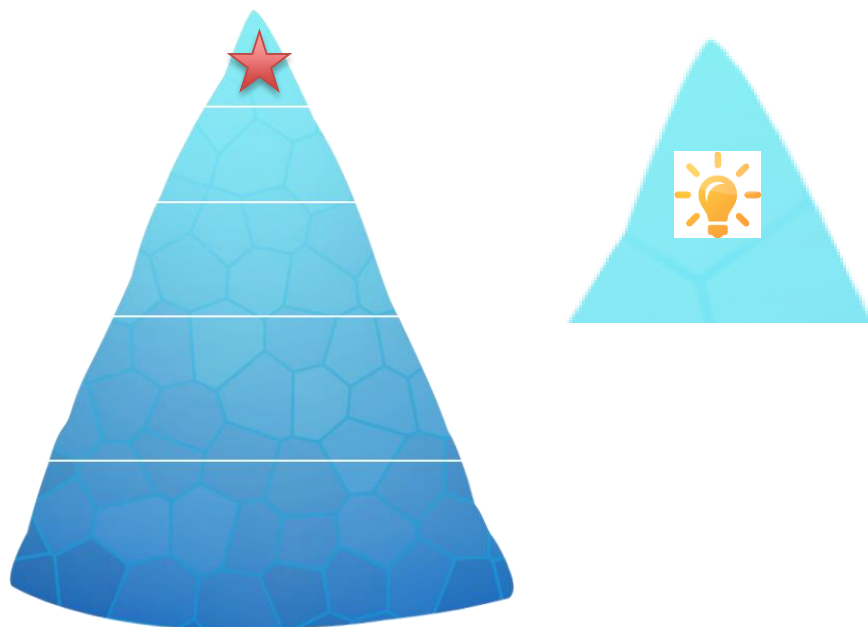
设计与验证  
细节

领域模型  
前后端协议  
测试数据实例

布局位置, 视觉  
设计, 配色风格

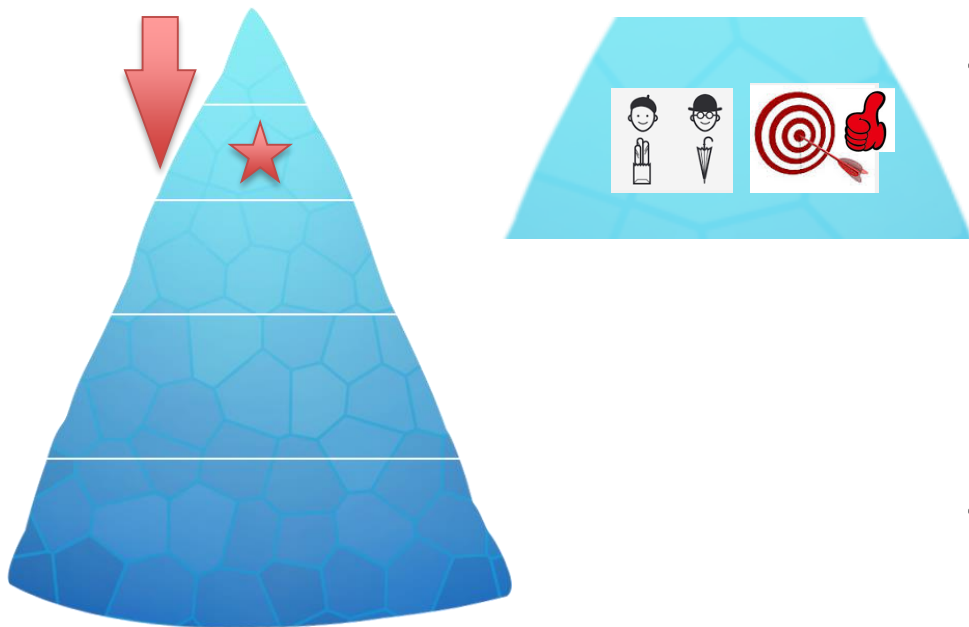


# 第一层：产品愿景，宪法



- 最根本的假设和战略意图
  - 例，微信八条：用户价值第一、鼓励有价值服务、消除中介、去中心化、消除地理限制、生态系统、动态系统、社交流量
- 每一层都要查是否“违宪”
  - 例，素食与海鲜
- 如第一层发生改变，即为放弃原产品，但快速失败是一种局部成功

## 第二层：用户核心价值



- 用户画像
  - 用户形象化
  - 代入感，设身处地考虑用户体验

例如“阿土伯”，“杜拉拉”

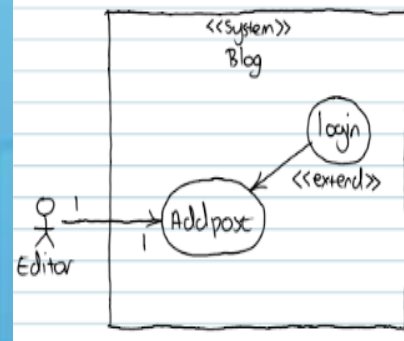
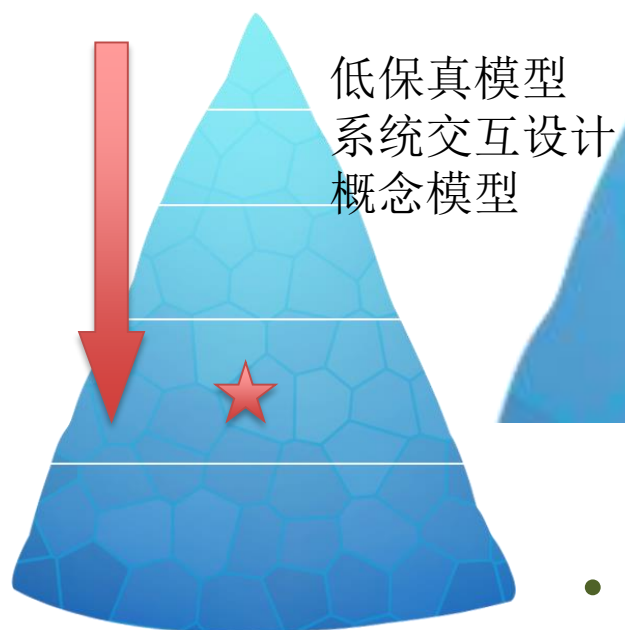
- 用户核心价值
  - 例如P2P金融对于投资者：资产增值
- 产品设计原则和常识
  - 例如“用户体验要素”

## 第三层：用户故事、模块



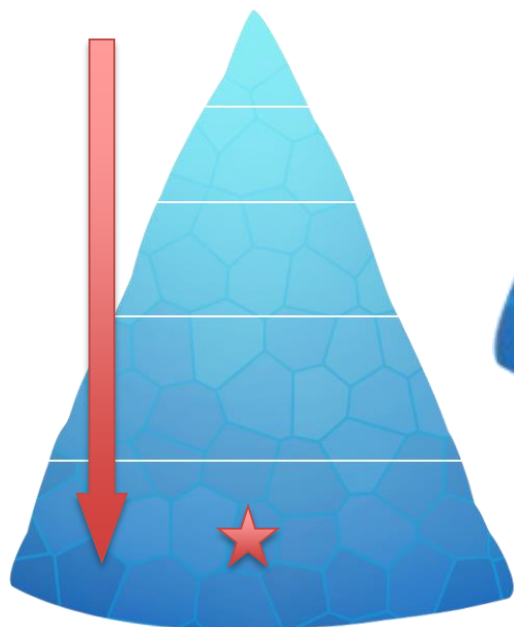
- 用户故事
  - Role , What , Why
  - 强调用户价值
  - 作为受管理的任务单元，驱动研发
- 模块 ( Epic )
  - 相关用户故事的集合
- 高层的利益相关者，可更多关注前三层，因为受控任务单元的进度跟踪会暴露风险

## 第四层：用户故事的概要设计\*\*

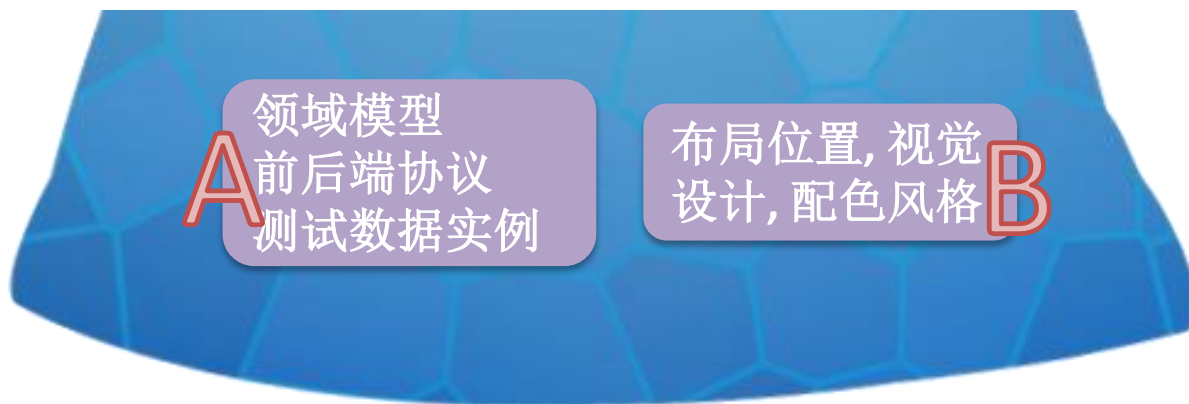


- 减少需求传递失真
- 用最小成本传递需求，活文档
- 凸显最容易反复出错的部分
- 独立验证单元，尽量竖切，快速验证

# 第五层：细化设计和验证标准



A部分容易反复出错，  
是验证重点



- 不同角色的关注重点
  - 产品，开发，测试 关注 A
  - 产品，UE，UI，原画等关注 B
- 细节的特点
  - A：逻辑的，隐藏的
  - B：直觉的，主观的



# 独立验证单元



低保真原型

数据实例

- 1, given: zhangsan vip level 10, expect: 2000
- 2, given: lisi , non-vip level 10, expect: 2000
- 3, given: zhangsan vip level 11, expect: 2200

A

用户操作	系统展示  状态变更
用户点击“签到”图标	显示本月日历。日历里有每天签到的情况、凸显当天。显示“签到”按钮。显示签到奖励的规则。
用户长按日历的某天的格子	显示按中格子对应的签到奖励
用户点击“签到”按钮	如当天已经签到，提示“已经签到了”，否则告知“签到成功，获得 xxx 奖励”，奖励规则如下：  如果是 vip,奖励 200*level 金币; 如果不是 vip, 根据 level, 如果 level 大于 10 奖励 100*level 金币, 如果 level 小于等于 10, 奖励 200*level 金币

系统交互设计

- 1, 今日已签到
- 2, 今日未签到

数据实例

分离了交互设计和数据实例，简化文档工作  
进一步还可以借助工具自动化

用户操作	系统展示  状态变更
用户点击“签到”图标	显示本月日历。日历里有每天签到的情况、凸显当天。显示“签到”按钮。显示签到奖励的规则。
用户长按日历的某天的格子	显示按中格子对应的签到奖励
用户点击“签到”按钮	<p>如当天已经签到，提示“已经签到了”，否则告知“签到成功，获得 xxx 奖励”，奖励规则如下：</p> <p>如果是 vip,奖励 <math>200 \times \text{level}</math> 金币;                  如果不是 vip, 根据 level,                      如果 level 大于 10 奖励 <math>100 \times \text{level}</math> 金币,                      如果 level 小于等于 10, 奖励 <math>200 \times \text{level}</math> 金币</p>

## 系统交互设计

## 数据实例

- 1, given: zhangsan vip level 10, expect: 2000
- 2, given: lisi , non-vip level 10, expect: 2000
- 3, given: zhangsan vip level 11, expect: 2200

## 数据实例

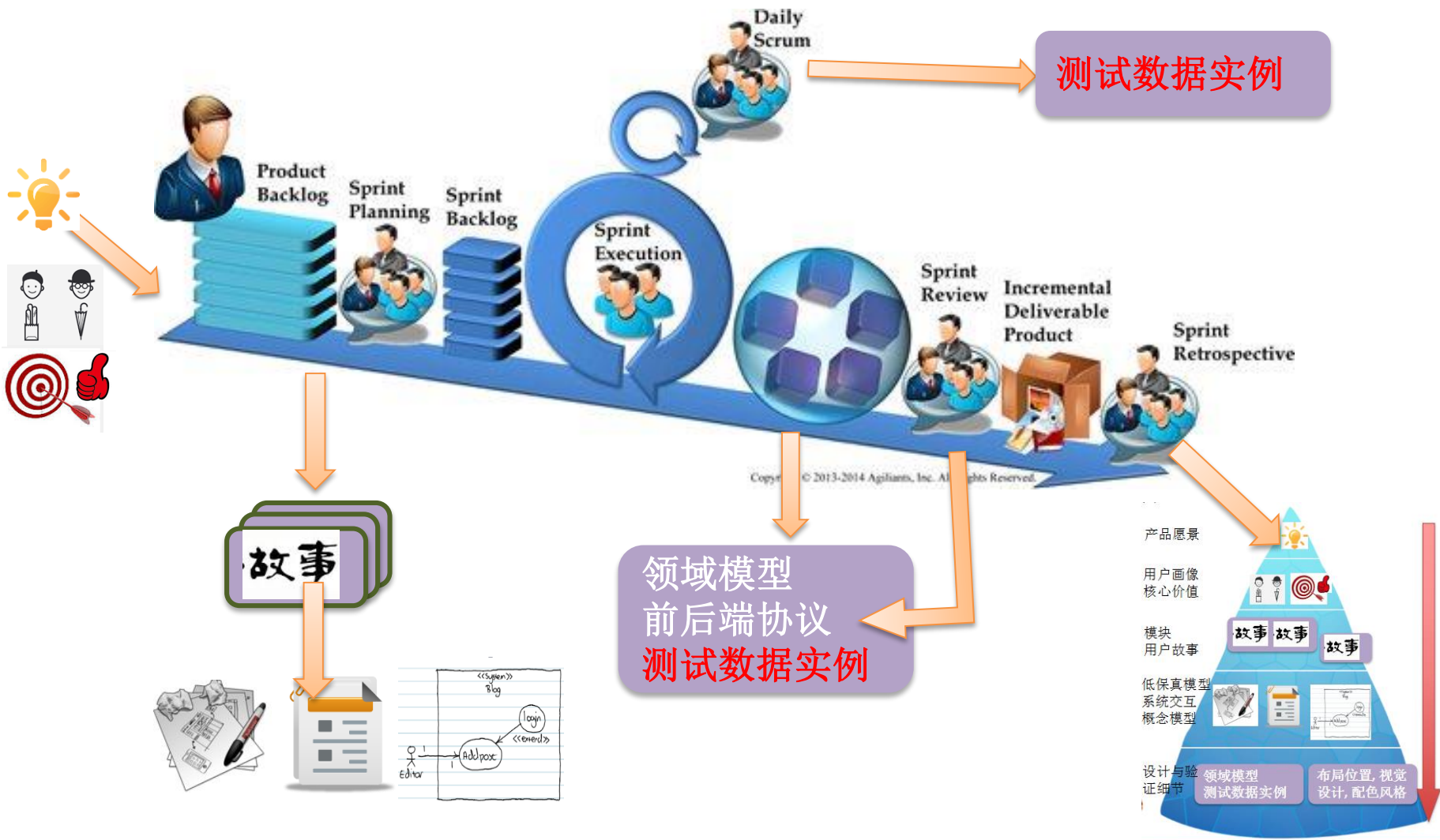
- 1, 今日已签到
- 2, 今日未签到



# 结合研发流程, 以Scrum为例

- 调研阶段
  - 产品愿景、用户核心价值、产品设计原则
- 每一个Sprint迭代前
  - 用户故事
  - 优先级
- 需求讲解会议
  - 低保真原型、概念、系统交互描述
- 估算会议
  - 一些典型的测试数据实例作为验收标准来估算
- Scrum会议
  - 标注完成的需求必须通过数据实例验证
- 产品演示会议
  - 整体回归, review上层约束

# 结合研发流程, 以Scrum为例



# 案例1,

- 现象：产品的某个模块一个方向持续做了很长时间, 上线后, 该模块数据改善了, 但产品整体数据变差了
- 原因：与产品主线冲突, (愿景, 用户偏好, 入口层次等等)
- 解决：每细化一层, 必须验证上层约束发现冲突及时调整



## 案例2 ,

- 现象：某个迭代2周时间, 预计最后3天统一调试, 结果发现工程师声称完成的任务中有很多bug, 改正这些bug花了整整一周半的时间.
- 原因：需求缺乏验收标准, 工程师估算时, 以写完程序或调试完主干业务为"完成"标准.
- 解决：
  - 用户故事为工作单元, 必须用数据实例验证后才算完成,
  - 能提前发现：需求理解偏差或工作量估算偏差.



## 案例3 ,

- 现象：某个产品, 测试工程师用excel撰写了详尽的测试用例, 但开发途中, 需求一再改变, 测试用例的修改工作量大, 有的跟不上需求的变更速度
- 原因：文档维护成本过高
- 解决：分离数据实例与交互设计描述文档
  - 数据实例更容易变化和增加
  - 去除重复描述



# 总结：Pizza模型的特点

- 每个环节盯紧用户价值
- 用户故事作为受控工作单元，驱动研发流程，发现风险
- 支持和鼓励竖切验证
  - 独立验证单元
    - 测试实例
    - 系统交互设计描述
    - 低保真原型Mockup
  - 不同特点的细节分开验证
    - 逻辑A 与 直觉B



# 最大公约数只是一个开始，接下来：

- 兼容并蓄
- 暴露风险和问题
- 推行最佳实践
- 提取可复用组件、服务、流程

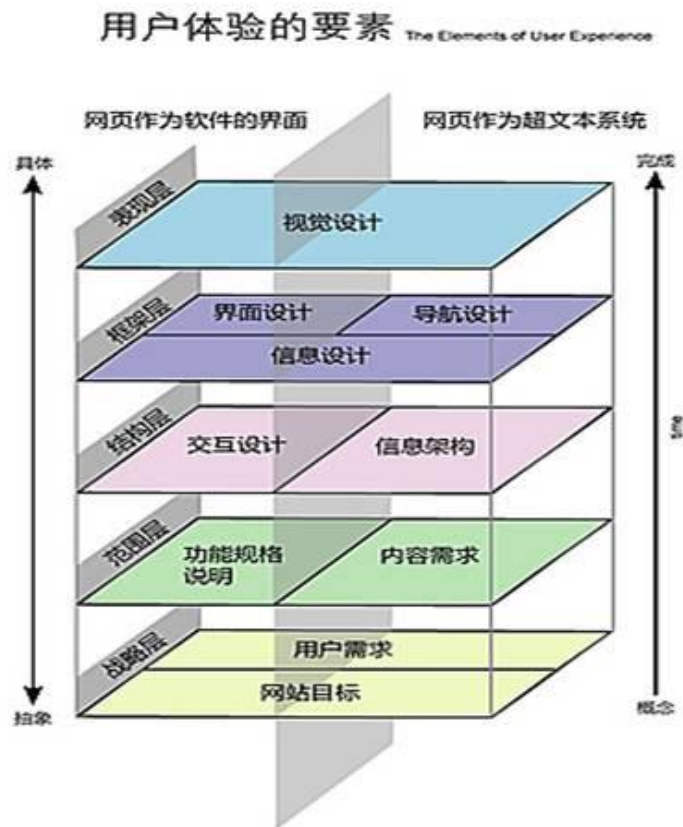


# Thanks!





## 参考资料：与用户体验要素对比



© 2000 Jesse James Garrett  
Redraw by xw  
Email: xw@xw@gmail.com  
Blog: D8in.com



## 参考资料

- 《实例化需求：团队如何交付正确的软件》 by Gojko Adzic
- 《用户体验要素》 by Jesse James Garrett