Measures

敏捷软件开发

麦哲思科技（北京）有限公司

报告人

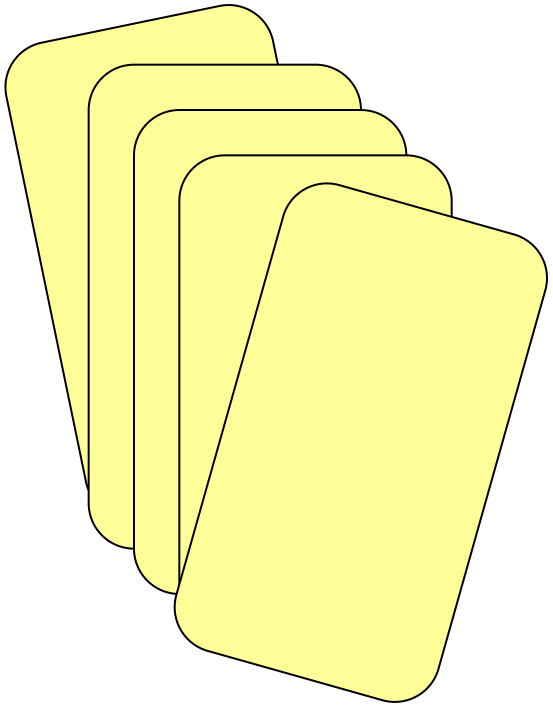
- 刘军
  - 高级咨询顾问，主要从事软件开发过程改进的培训与咨询
  - 联系方式：
    - Mobile： 13602564725
    - E-mail： liujun@measures.net.cn
  - 工程经验：
    - 从1996年开始从事软件开发，从普通的程序员迅速成长为项目骨干、项目经理、PMO项目总监，先后参与了20多个大中型项目开发和管理工作。
    - 多次成功地实施、咨询和评估CMMI3级过程改进

内容

- 敏捷的含义
- 产生背景
- 基本理念
- Scrum方法简介
- XP方法简介
- 小结

你是如何理解的”敏捷”?

请列出描述敏捷思想的5个词？



敏捷的含义是什么

Measures

神话

1. 没有文档
2. 没有规范
3. 没有计划
4. 没有预测
5. 不会变异
6. 一时的流行风尚
7. 银弹
8. RUP不敏捷
9. 没有固定的价格

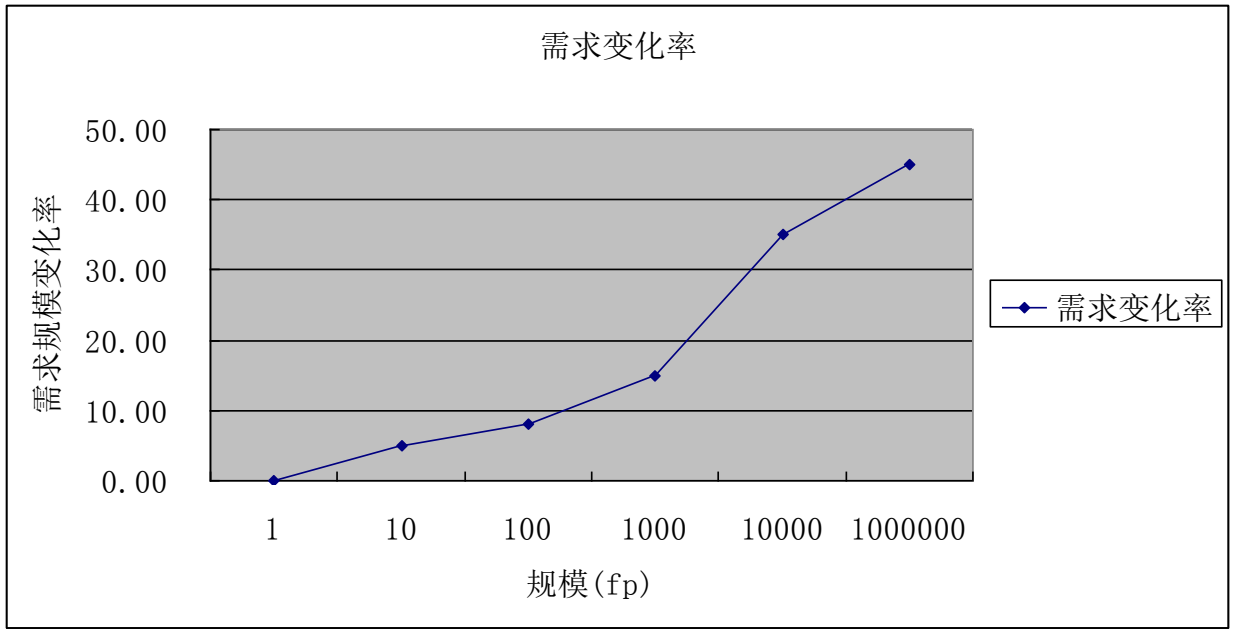
现实

1. 敏捷文档
2. 需要严格的纪律
3. Just-in-time（JIT）策划
4. 更长远的可预测性
5. 慢慢变化也是敏捷
6. 常态方法
7. 需要熟练的高技能人员
8. 如果你裁剪它，RUP也可以敏捷
9. 由利益相关者控制的预算、进度和范围

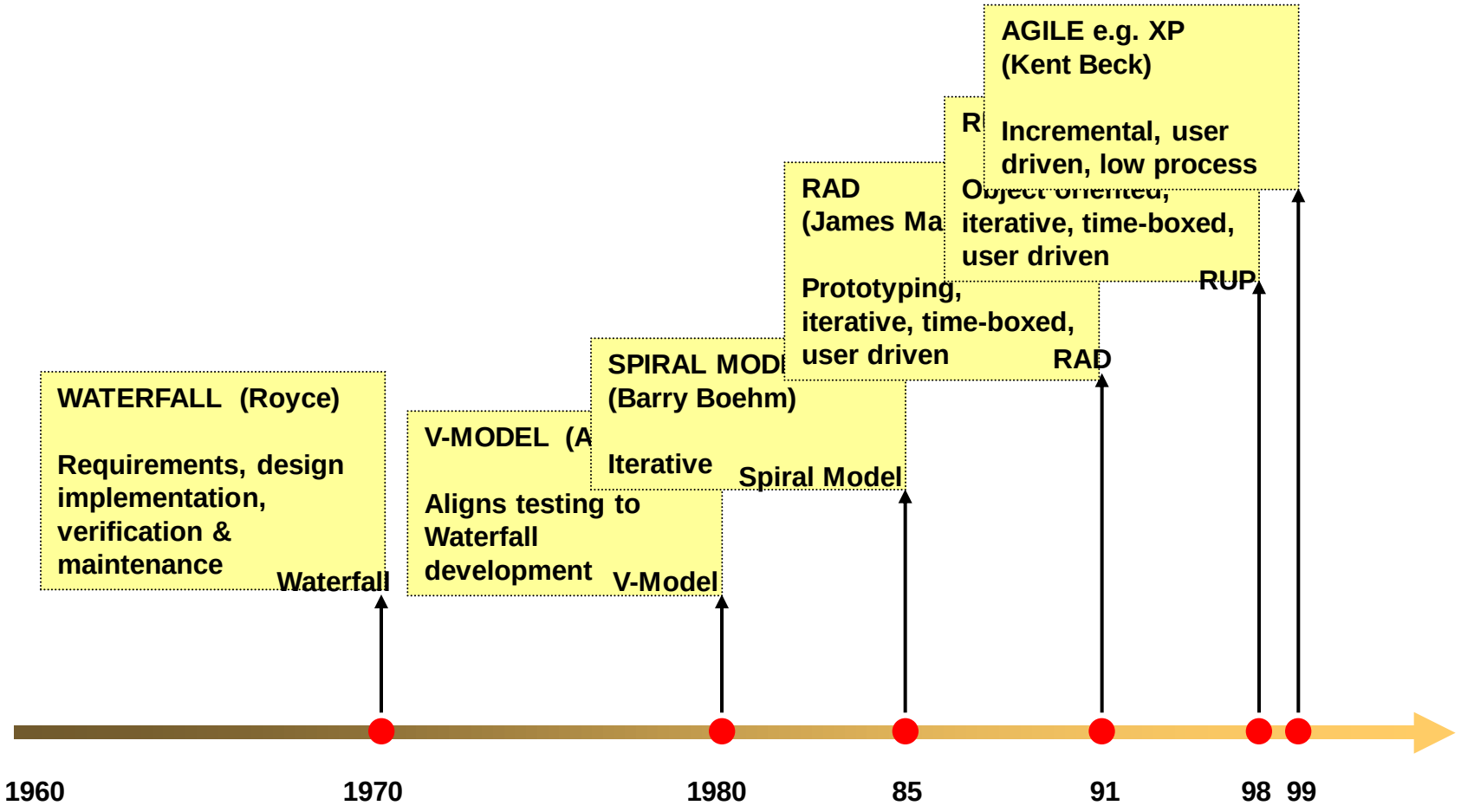
敏捷方法产生的背景

现代软件的

- 复杂性
  - 软件越来越复杂
- 可变性
  - 需求越来越多变
- 一致性
  - 过程越来越规范



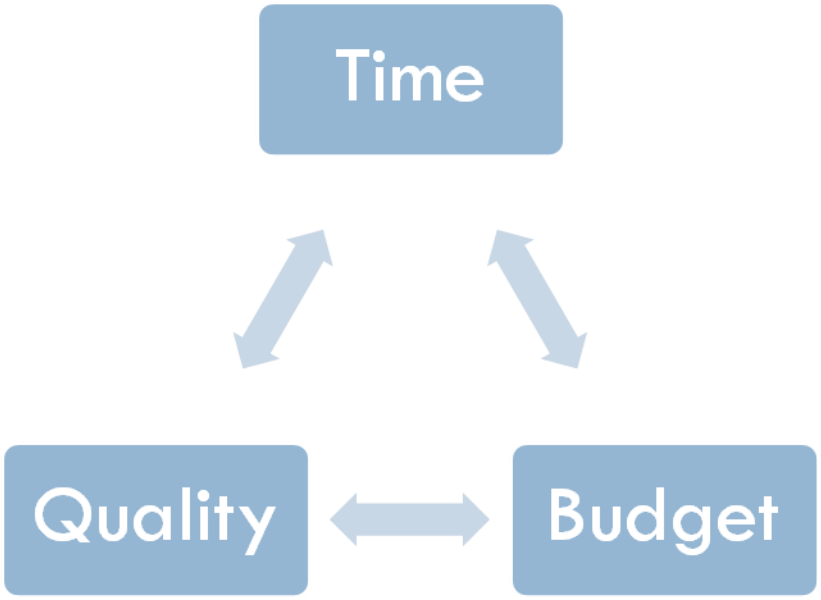
开发方法的简史


AGILE e.g. XP
(Kent Beck)
Incremental, user driven, low process
RUP
iterative, time-boxed, user driven
RAD
Prototyping, iterative, time-boxed, user driven
Spiral Model
Iterative
V-Model
Aligns testing to Waterfall development
Waterfall
Requirements, design implementation, verification & maintenance
WATERFALL (Royce)
1960
1970
1980
85
91
98
99

敏捷方法与规范方法的平衡策略的比较

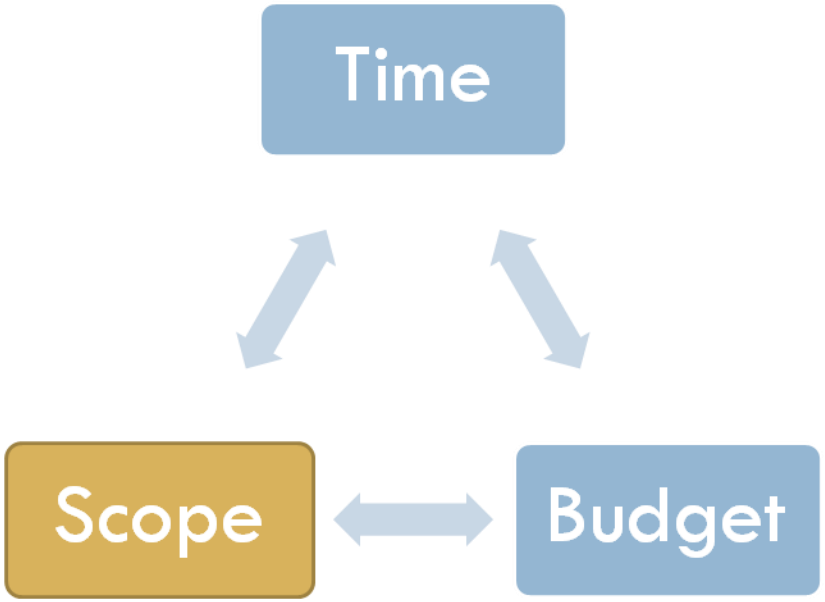
Measures

Waterfall



Scope is Fixed

Agile



Quality is Fixed to High!

产品质量的分类

Measures

- 外部质量
  - 是系统用户可以感知的。
- 内部质量
  - 是用户看不到的要素，他们对系统的可维护性具有深远的影响。可维护性包括系统设计的一致性、测试覆盖率、代码可读性和重构等。
- 产品负责人定义了外部质量，团队成员定义了内部质量。
- 外部质量是可以裁剪的，内部质量是不可以裁剪的。在敏捷方法中绝不对内部质量让步。
- 外部质量好，内部质量未必好，内部质量差，外部质量一定差。

规范的方法

Measures

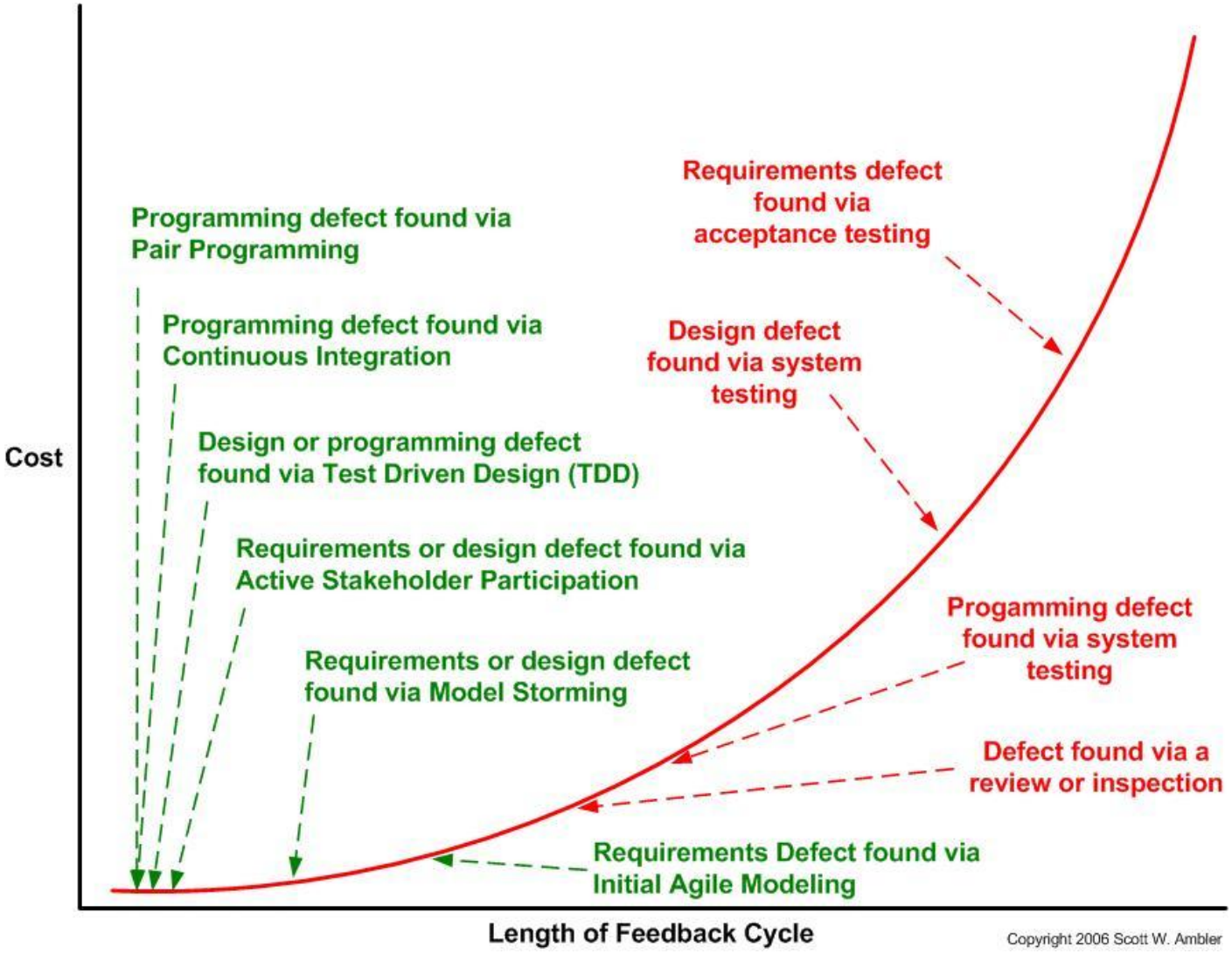
- 管理的假设：
  - 每个人都会犯错误
  - 规范的、已定义的过程可以降低犯错的概率，可以通过过程保证产品的质量
  - 通过检查过程的记录可以检查是否执行了过程
- 反思：
  - 这些工作产品是对客户、对用户有价值的吗？有直接价值吗？是最终的交付物吗？
- 应用范围：
  - 甲方驱动，甲方希望乙方规范其开发与生产的流程，甲方的高质量要求驱动了乙方管理的规范化

敏捷的方法

Measures

- 管理的假设：
  - 团队的沟通与协调可以减少错误的发生，每个员工都积极地与其他的成员合作
  - 只要告诉了成员做事的原则，他们都能够基于经验做出正确的选择，即使做错了，也能及时发现，及时纠正
  - 不需要检查执行的中间记录，而是检查交付物，只要保证了交付物的质量，而不关注中间工作产品的质量
- 反思：
  - 是否每个人都有能力胜任其本质工作？
  - 团队是否都建立了沟通与合作的良好文化？
  - 如果确保成功能够重复？
- 应用范围：
  - 乙方驱动，乙方希望甲方积极参与到项目开发中，理解开发的过程。

为什么敏捷方法可以成功？



敏捷方法宣言 2001年初

Measures

个体和交互	over	过程和工具
可以工作的软件	over	面面俱到的文档
客户合作	over	合同谈判
响应变化	over	遵循计划

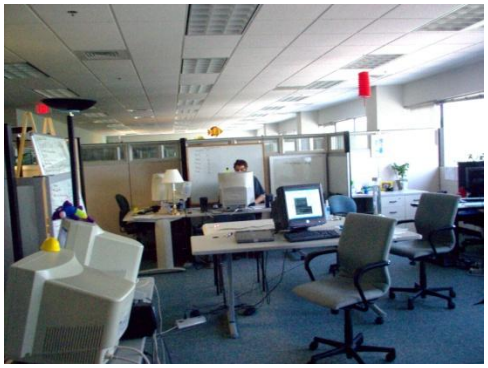
《敏捷宣言》12条原则

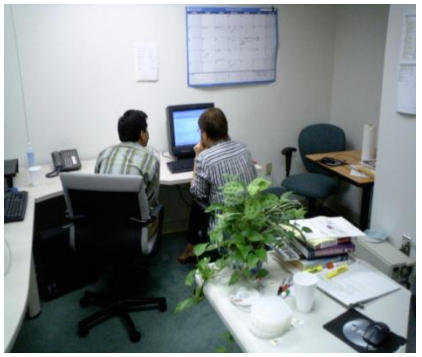
Measures

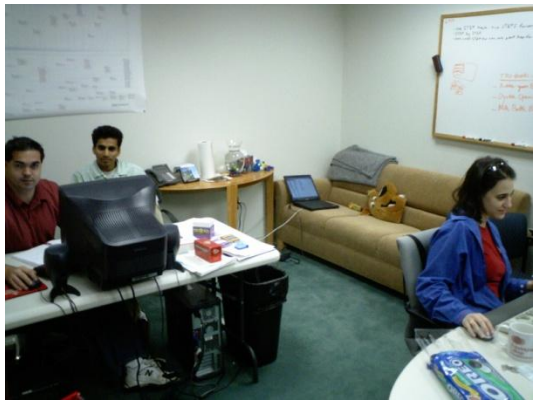
1. 最优先的目标是通过尽早地、持续地交付有价值的软件来满足客户。
2. 欢迎需求变化，甚至在开发后期。敏捷过程控制、利用变化帮助客户取得竞争优势。
3. 频繁交付可用的软件，间隔从两周到两个月，偏爱更短的时间尺度。
4. 在整个项目中业务人员和开发人员必须每天在一起工作。
5. 以积极主动的员工为核心建立项目，给予他们所需的环境和支持，信任他们能够完成工作。
6. 在开发团队内外传递信息最有效率和效果的方法是面对面的交流。
7. 可用的软件是进展的首要度量指标。
8. 敏捷过程提倡可持续开发。发起人、开发者和用户应始终保持一个长期的、稳定的开发速度。
9. 简化——使必要的工作最小化的艺术——是关键。
10. 持续关注技术上的精益求精和良好的设计以增强敏捷性。
11. 最好的架构、需求和设计产生于自我组织的团队。
12. 团队定期地对运作如何更加有效进行反思，并相应地调整、校正自己的行为。

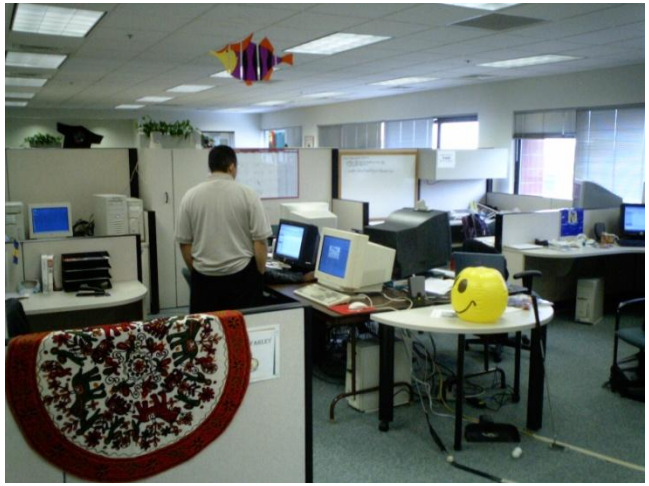
敏捷方法下的办公环境

Measures





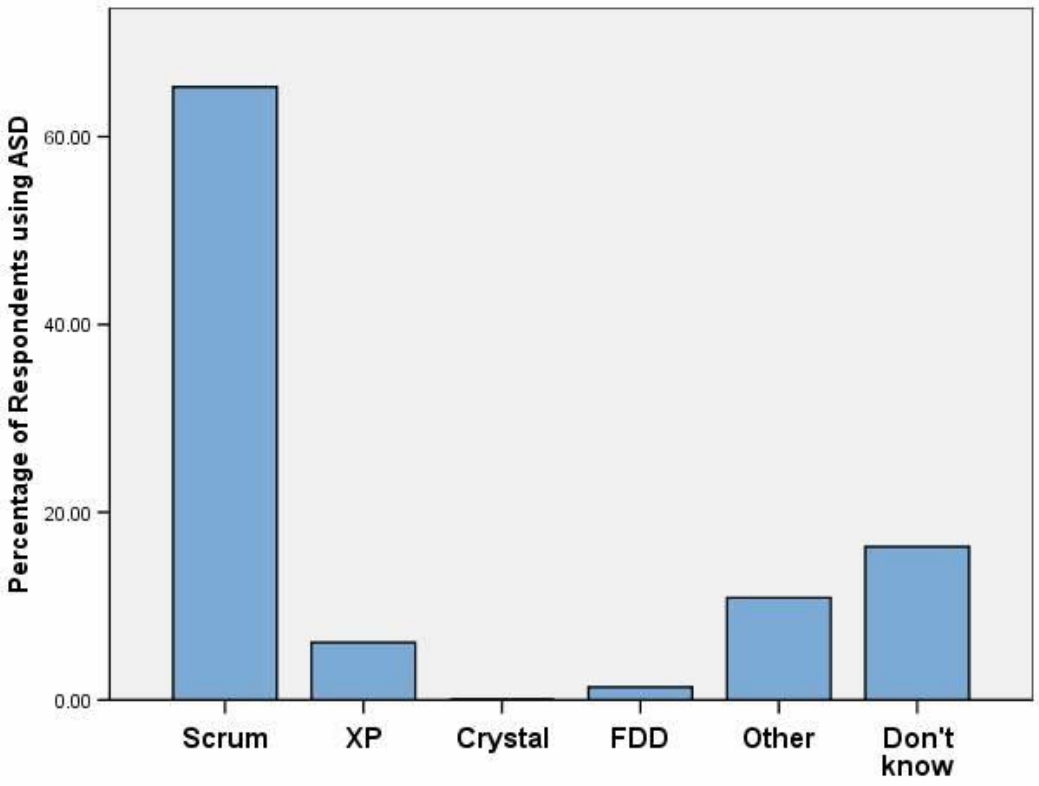




代表方法

- 瑞理统一开发过程: Rational Unified Process
- 敏捷建模: Agile Modeling
- 极限编程: eXtreme Programming
- 自适应软件开发: Adaptive Software Development
- 水晶方法体系:Crystal
- Scrum方法

  etc.



Measures

Scrum项目管理方法

内容

- 什么是scrum?
- Scrum的过程
- Scrum的框架
- Scrum的三个角色
- Scrum的三个工作产品
- Scrum的四个活动
- Scrum的演练

Scrum

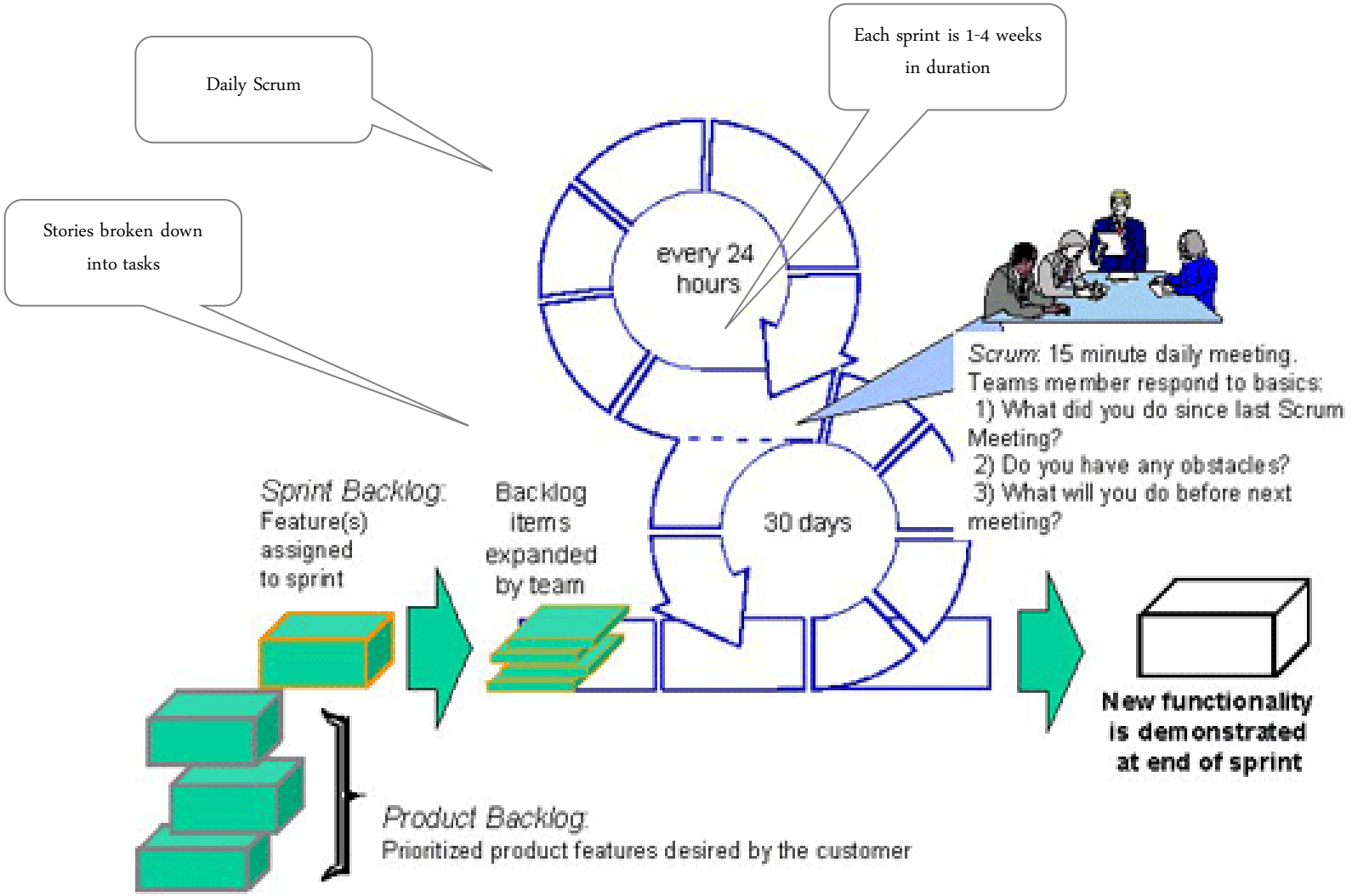
Measures

- SCRUM 是敏捷过程的一种，以英式橄榄球争球队形(Scrum)为名。

Scrum

- 基本假設是
  - 『开发软件就像是开发新产品，无法一开始就能定义最终产品的规程，过程中需要研发、创意、尝试错误，所以没有一种固定的流程可以保证项目成功。
- Scrum将软件开发团队比拟成橄榄球队
  - 有明确的最高目标，
  - 熟悉开发流程中所需具备的最佳典范与技术，
  - 具有高度自主权，
  - 紧密地沟通合作，
  - 以高度弹性解决各种挑战，
  - 确保每天、每个阶段都朝向目标有明确的推进

Scrum 过程


Daily Scrum
Each sprint is 1-4 weeks in duration
Stories broken down into tasks
every 24 hours
30 days
Scrum: 15 minute daily meeting.
Teams member respond to basics:
1) What did you do since last Scrum Meeting?
2) Do you have any obstacles?
3) What will you do before next meeting?
Sprint Backlog:
Feature(s) assigned to sprint
Backlog items expanded by team
New functionality is demonstrated at end of sprint
Product Backlog:
Prioritized product features desired by the customer

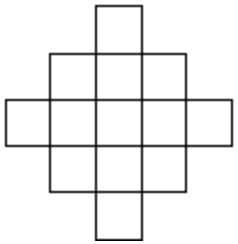
Scrum 关键元素/ 概念

Measures

- 冲刺或迭代
- 自管理团队
- 审查与调整循环
- 采用三角平衡结构（product owner, scrum master, and development team）而非层次化组织结构
- 划分了优先级的待办事项列表（backlogs）
- Sprint Backlog根据产品backlogs列表转化为系统功能所需完成的任务
- 经验型过程控制机制

练习：命令式开发管理

- 2人一组
- 一个人是经理
- 一个人是组员
- 组员受经理的领导
- 经理给组员下达命令：
  - 前进
  - 停止
  - 左转
  - 右转
- 请在90秒内完成60个正步走

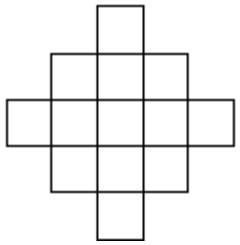


- 2人一组
- 一个人是经理
- 一个人是组员
- 组员受经理的领导
- 经理给组员下达命令：
  - 状态指令：
    - 停：笔停止移动，直到听到画的命令
    - 画：笔画画直到听到停的命令
  - 方向指令：如果是停的状态则等待下条指令，如果是画的状态则按方向指令画画，直到听到下一条指令
    - 上：
    - 下：
    - 左：
    - 右：
- 请在90秒内完成下面的图形

练习：自管理式团队

- 2人一组
- 一个人是经理
- 一个人是组员
- 经理确定目标
- 经理提供各种支持与帮助，不需要经理发命令
- 请在60秒内完成60个正步走

- 2人一组
- 一个人是经理
- 一个人是组员
- 经理确定目标
- 经理提供各种支持与帮助，不需要经理发命令
- 请在30秒内完成下面的图形



Scrum 框架

Measures

角色

- Product owner
- ScrumMaster
- Team

活动

- Sprint planning
- Sprint review
- Sprint retrospective
- Daily scrum meeting

工作产品

- Product backlog
- Sprint backlog
- Burndown charts

Measures

角色

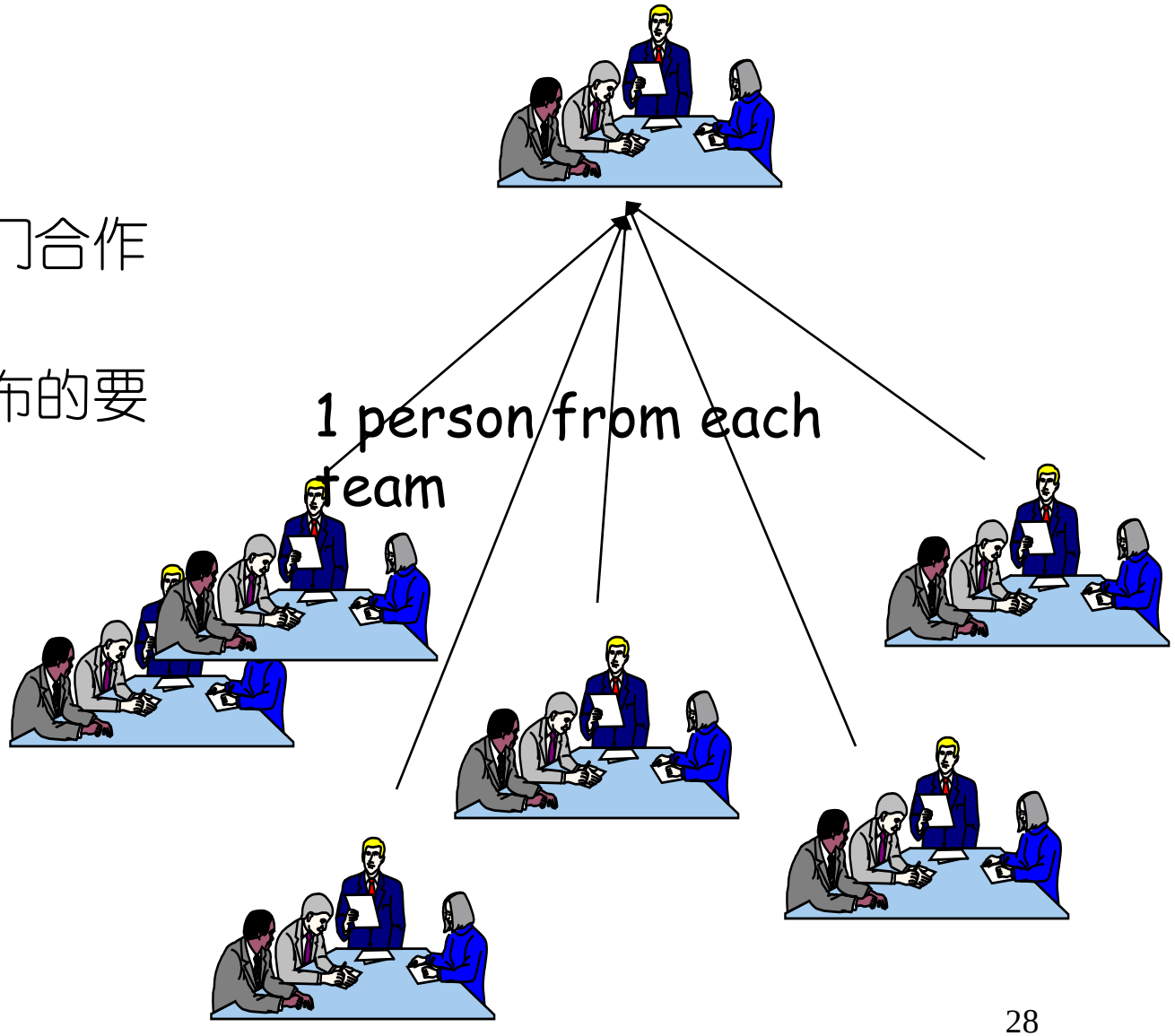
Scrum开发团队的组成

Measures

- 7人组成
- 管理者主持会议，负责对整个项目的成败
- 对于人数多于7人的项目团队，建议与其扩大团队规模不如将团队分组
- 通过Scrum会议对各个子团队的工作进行同步
- 团队不止是一个程序员队伍，它由各种背景下的不同角色组合而成，包括商业分析者，设计师，程序员和测试者等等。正确的组合决定了团队的能力和效率

大团队的组成

- 团队再分组
- 多个小团体
- 最有效的大部门合作方法
- 也适宜地域分布的要求



Scrum 角色

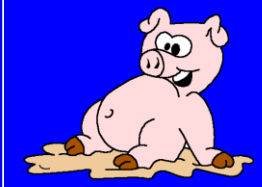
Pigs[1] are *committed* (developers, writers, etc.). They get to speak in the Scrum meeting.

Chickens are *involved*. They keep quiet.

(1) "Pigs & Chickens" is central to Scrum -- no offense meant.

角色

Pigs



Chickens

- Product Owner: 定义用户故事
- Scrum Master: 推动者
- Team: 5-9 人, 交叉功能性技能

- Users
- Stakeholders
- Managers

Scrum roles

Measures

ScrumMaster

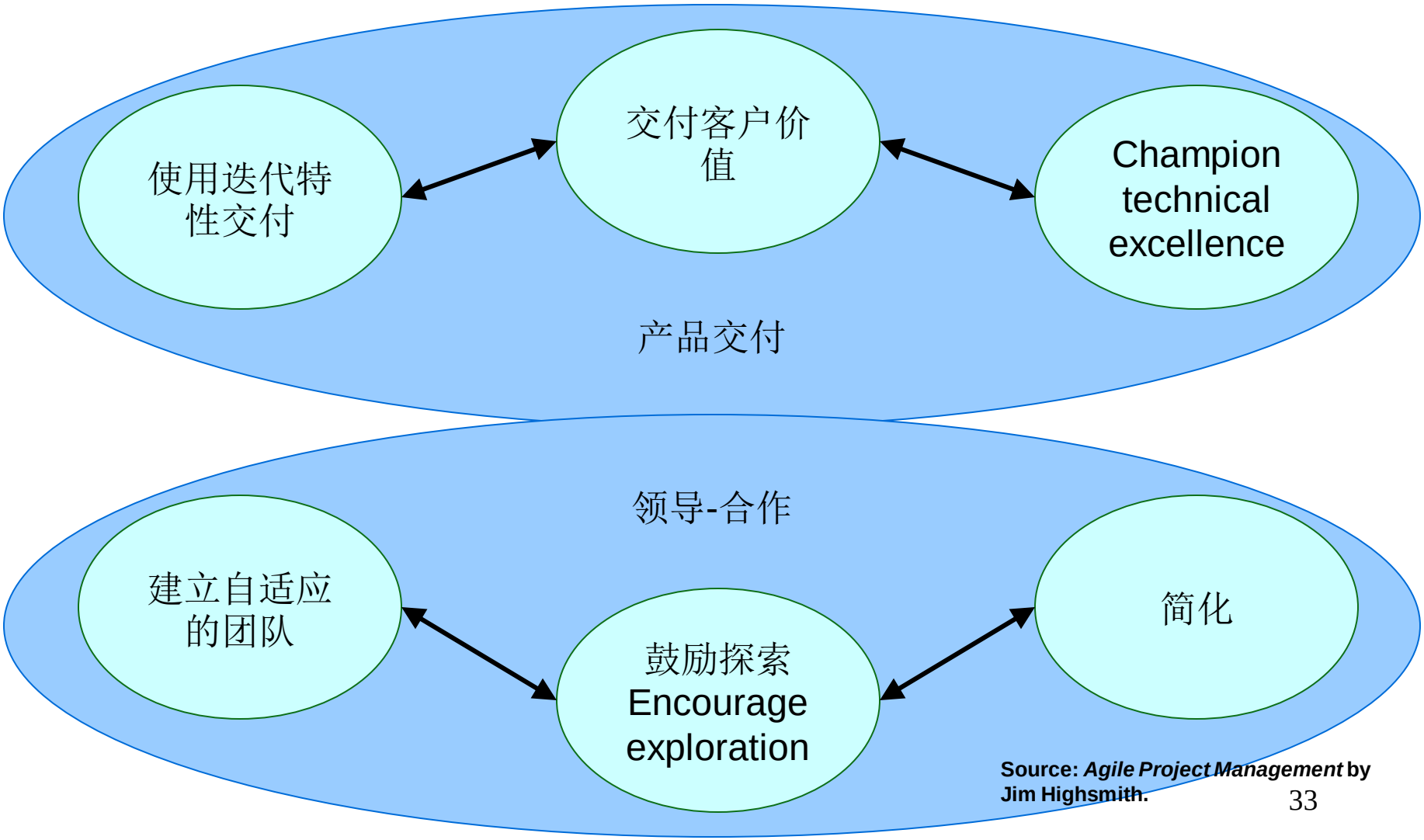
Product Owner

The Team

Scrum Master的职责

- (1) 会议主持人
  - 他负责主持四个主要的会议：策划会议、每日站立会议、迭代评审会议、迭代回顾会议。
- (2) 牧羊犬
  - 他负责屏蔽项目组外部的干扰。
- (3) 雷锋
  - 他给product owner、team提供帮助，帮助product owner确定需求、排定优先级，帮助team做估算、分解任务、完成任务。
- (4) 外交官
  - 当项目组外部有人不理解项目组的工作时，他负责去解释说明，负责对外发布项目组的信息。
- (5) 教练
  - 他指导项目组的成员按照SCRUM的原则、方法做事情，当出现偏差时，他去纠正，可以说他是精神教父、他也是警察（QA）。如果有项目组的成员不熟悉SCRUM的方法，他要去提供相关的培训。
- (6) 清道夫
  - 他负责排除在项目进展中遇到的各种障碍，如果他没有能力或资源他可以协调项目组的其他成员一起来排除障碍。

Scrum Master的6个指导原则



Source: ***Agile Project Management*** by Jim Highsmith.

产品交付原则 1–2

Measures

使用迭代特性交付

- 每次sprint确保可以交付潜在的可使用的产品
- 保持聚焦于交付特性而不是完成任务

交付客户价值

- 通过实现高价值的特性支持product owner

产品交付原则 3

Measures

技术优异的冠军

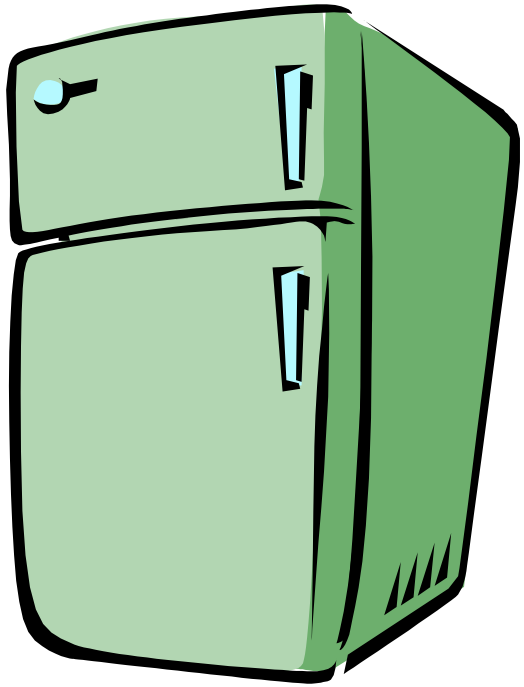
- 产品每天都要交付价值
  - 所以我们需要聚焦于能力和质量
- 不能仅仅成为进度的冠军

提倡技术卓越

Measures

- You want code you're so proud of that you hang it on your mom's fridge

- 确保编写好的代码并保证速度



When something is done well, it's only a matter of time until it is done quickly.

领导/合作原则

Measures

建立自适应的团队	▪ 团队成员需要理念和技能以响应变化
鼓励探索	▪ 在不确定的环境中，我们需要探索而非确认计划
简化	▪ 仅作“刚刚好”的事情

Scrum roles

Measures

ScrumMaster

Product Owner

The Team

Product Owner的责任

Measures

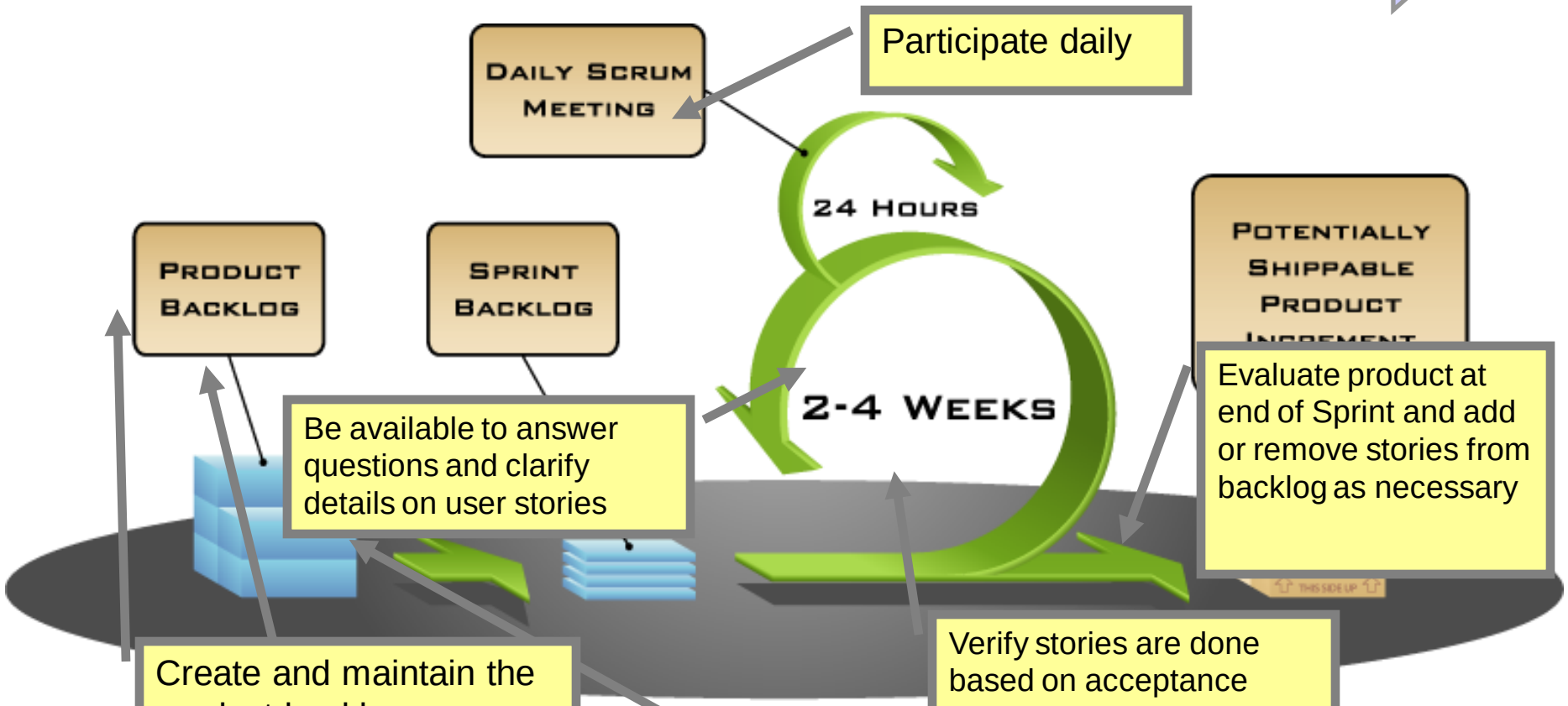
- 与其他成员一起估算Product Backlog
- 增加特性的唯一一个人
- 对特性排优先级的唯一一个人
- 对User Stories / requirements具有最终的权力
- 负责确保 Product Backlog及时更新
- 负责产品的盈利能力
- 为在每个Sprint调整特性的优先级
- 验收完成的任务
- 是产品经理，来自于市场或其他类似部门

Product Owner的责任

- (1) 领域专家
  - 他是需求方面的专家，熟悉需求。他知道客户、最终用户、以及其他利益相关者对项目的真正需求是是什么。他负责编写用户需求、维护用户需求。
- (2) 需求决策人
  - 哪个需求重要，哪个需求不重要，需求的优先级如何排列，在某次发布中要发布哪些需求是他来拍板的。他负责来平衡需求、进度与资源的关系。
- (3) 需求讲师
  - 他负责在项目进展过程中给项目组的其他成员讲解需求的含义，对需求进行答疑。
- (4) 测试员
  - 他负责编写每个需求的验收标准，功能测试用例。
- (5) 验收人
  - 当项目组成员完成某个需求后，是product owner进行功能测试，进行验收，他认可后才能认为某个需求完成了。

Pr...

- Communicate Business Goals, Customer Goals, End User Goals
- Coordinate involvement of SMEs, users, and business stakeholders
- Coordinate with other product owners to insure coherence of product and releases



41

Scrum 角色

Measures

ScrumMaster

Product Owner

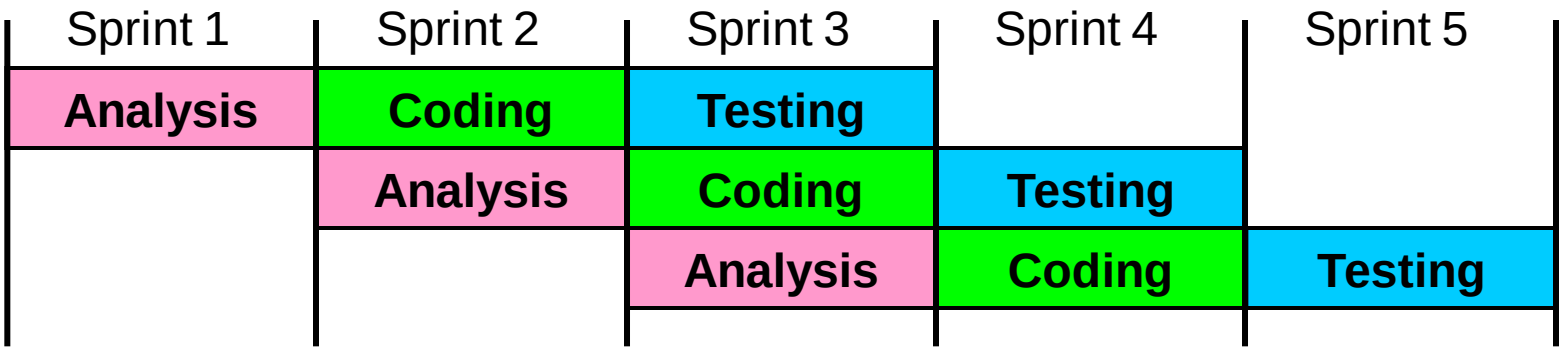
The Team

Scrum 团队

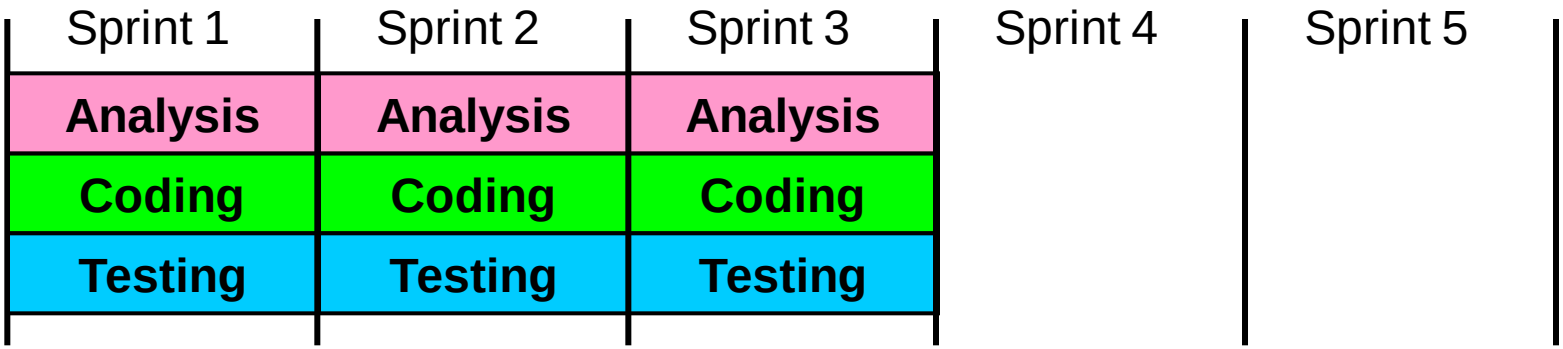
- 通常 5-10 人
- 交叉职能
  - -QA, 程序员, UI 设计者, etc.
- 应该是全职人员
  - -可能有例外（e.g., Sys Admin, etc.）
- 团队成员自我管理
- 只有在sprint之间才可以更换成员

团队成员是交叉职能

- 非交叉职能团队



- 交叉职能团队



Team 承诺

Measures

- 团队挑选在一个sprint中的工作
- 团队承诺完成他们选择的task
  - 是团队的承诺，不是个人的承诺
- 有权去做可以达成承诺的任何事情
- 是整个团队成功或失败了而非某个人

Product owner和开发人员的分工

- 业务人员确定的内容
  - 范围
  - 优先级
  - 版本的组成
  - 发布的日期
- 技术人员确定的内容
  - 估算
  - 技术风险
  - 过程
  - 详细的日程表

Measures

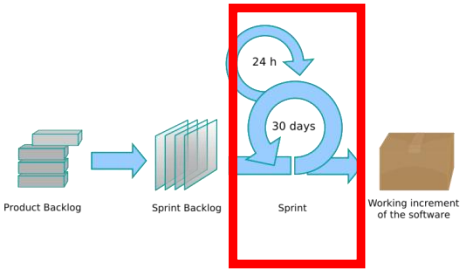
文档

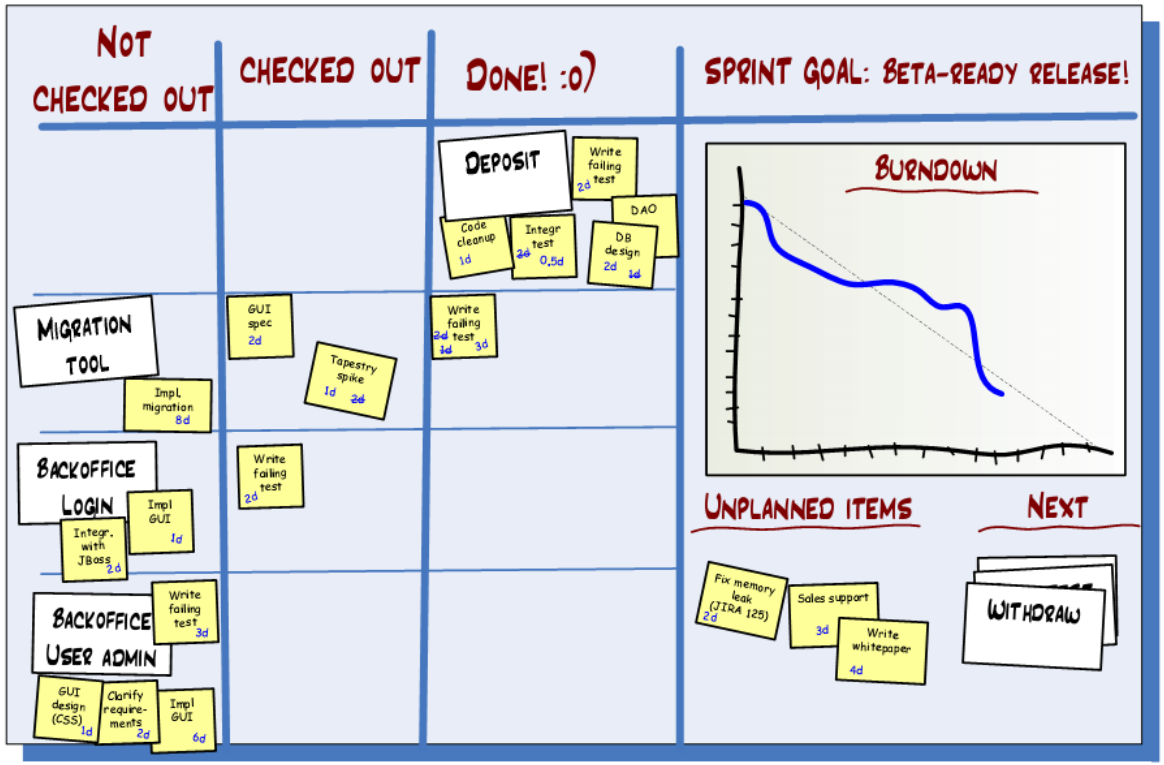
文档-Scrum看板

Measures

Scrum Board

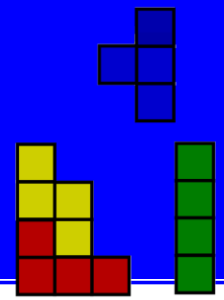
- Status
- Assignments



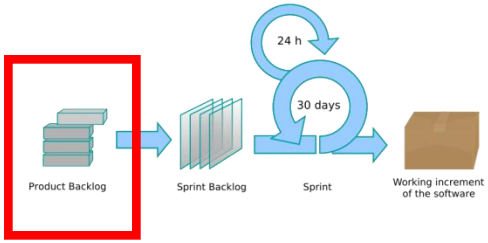


文档- Product Backlog

The product Backlog



- 高层文档
- 由用户故事构成
- 粗估计



- The users starts a game by selecting the complexity level
- Random figures fall down
- The user can rotate the falling figure
- Figures pile at the bottom
- When the pile reaches the top, the user is asked his name for the hall of fame

用户故事

Measures

- 用户故事描述了对软件（或系统）用户或客户有价值的功能。
  - 用户故事案例：用户可以在网站上发布简历
  - 非用户故事：这个软件将用C++编写

- 客户编写用户故事，而非开发人员编写。
- 开发者辅助客户编写故事，告诉客户所编写的故事是进一步讨论的引子，而不是详细的需求规范。
- 在任何项目中，需要客户团队根据故事的重要性来安排开发者的工作，回答所有开发者的问题，编写所有的故事。
- 客户团队包含多个成员（诸如测试人员、产品经理、真实用户、交互功能设计人员），确保软件满足目标用户的需求。

User stories的描述方式

Measures

As a <type of user>,
作为用户什么角色

I need <immediate goal>,
需要满足什么功能

so that <business outcome>.
说明为什么要实现这个功能

51

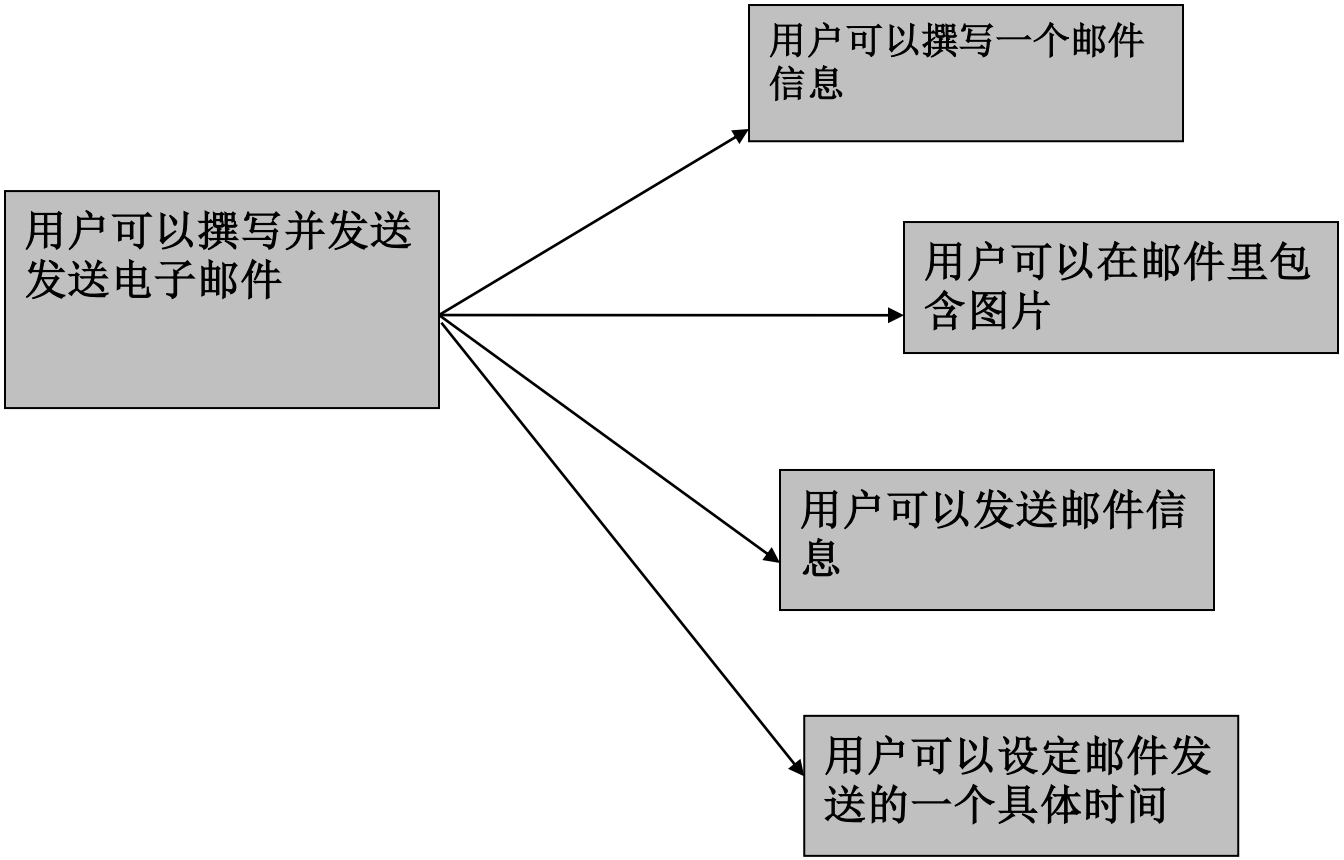
User story 样例

一个工作招聘的网站的用户故事：

- 找工作的人可以搜索职位
- 公司可以发布新职位
- 公司可以搜索简历
- 找工作的人可以限制浏览简历的人
- 网站的人可以限制公司搜索简历的联系人信息

更详细的User story 样例

多个小故事胜过一个庞大的故事（史诗故事）。



需求与验收准则

Measures

用户故事不是契约，验收准则才是契约。

作为一个家长，我需要一个餐厅，以便于我能够和朋友们在家里开一个party

- 验证这个餐厅是否可以容纳36把椅子
- 验证灯光是否明亮
- 验证餐厅是否离厨房比较近
- …

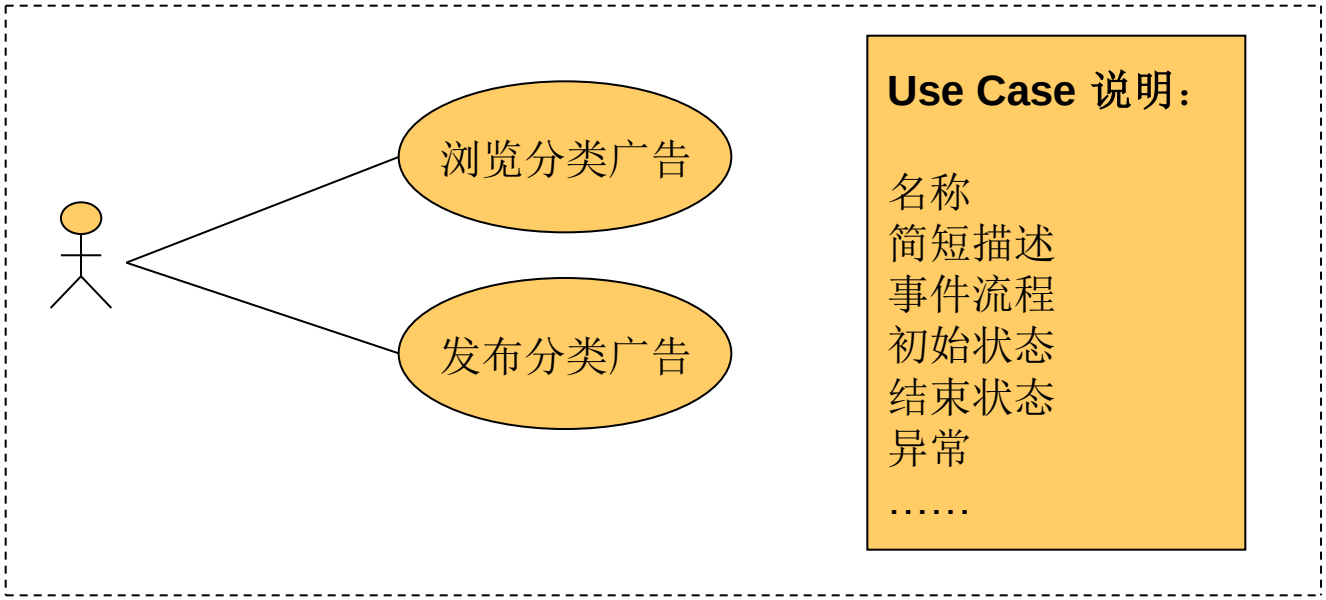
用户故事卡片上的内容

- 用户故事
- 优先级
- 风险
- 估算的工作量
- 责任人
- 验收标准

用户故事的特征

- Bill Wake使用缩写词INVEST表示6个属性[2]。
  - -1）独立性。尽可能避免故事之间存在依赖关系，故事间依赖关系会产生优先级和规划问题。
  - -2）可协商性。故事是可协商的，不是必须实现的书面合同或者需求。
  - -3）对用户或者客户有价值。确保每个故事对客户或用户有价值的最好方式是让用户编写故事。
  - -4）可预测性。开发者应该能够预测（至少大致猜测）故事的规模，以及编码实现所需要的时间。
  - -5）短小精悍。故事规模对实现有影响，何种故事规模最合适，取决于开发组规模、开发组的能力，以及技术实现等方面。
  - -6）测试性。所编写的故事必须是可测试的。

User Story 和 Use Case 的区别



术语表：

客户：　……
分类广告：　……
……

补充说明：

可用性：　……
可靠性：　……
可维护性：　……
……

Presentation Requirement

Mock-up Window

……

User Story 和 Use Case 的区别

Measures

User Story	Use Case Modeling
从用户角度看	从开发人员角度看
由用户写	由开发人员写
给开发人员看	给用户看
一张卡片上的一段描述	图
每张卡必须有商业价值、开发风险和开发时间三个评估值	每个图通常常有正规的说明、术语表等等
不包含细节流程	包含细节流程
1周 < 粒度 < 3周	无
一张卡片只有一个User Story	一个图通常有多个Use Case
开发计划以User Story为单位	无
开发速度根据User Story计算	无

练习：编写user story

- 30分钟
- 5-10人一组
- 编写家庭财务管理软件的需求，用以管理各个家庭的财务计划、收入、支出情况以及年底的各种统计。
- 输出：
  - （1）编写的用户故事
  - （2）对用户故事的验收标准
  - （3）该方法的优缺点分析

练习：user story好与不好

Measures

- 用户可以在WINDOWS XP和WIN7上运行系统
- 所有绘图和图表将用第三方类库完成
- 用户最多可以撤销50步操作
- 软件将在7月30日发布
- 软件用java编写
- 用户可以从下拉列表选择她的国籍
- 系统用LOG4J记录错误信息到文件
- 用户15分钟没有保存文档，系统提示用户进行保存
- 用户可以选择“导出到XML”特性
- 用户可以导出数据到XML文件

练习：user story好与不好

Measures

- 用户可以在WINDOWS XP和WIN7上运行系统 好
- 所有绘图和图表将用第三方类库完成 不好
- 用户最多可以撤销50步操作 好
- 软件将在7月30日发布 不好
- 软件用java编写 不好
- 用户可以从下拉列表选择她的国籍 好
- 系统用LOG4J记录错误信息到文件 不好
- 用户15分钟没有保存文档，系统提示用户进行保存 好
- 用户可以选择“导出到XML”特性 好
- 用户可以导出数据到XML文件 好

技术故事

Measures

- 或成为非功能性条目，非产品负责人所关心的，如：
  - 安装持续集成服务器
  - 编写系统设计概览
  - 重构DAO层
  - 升级JIRA
- 技术故事能否转换为用户故事？
- 技术故事能否成为某个用户故事的任务？

约束故事

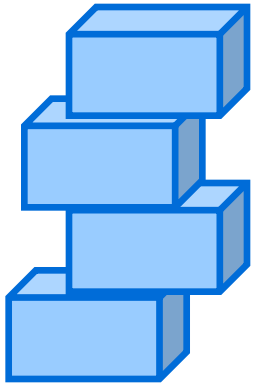
Measures

- 必须要遵守而不需要直接实现的故事
- 如：
  - 系统必须支持最大50个并发用户的峰值
  - 设计的软件要便于今后实现国际化
  - 系统的无故障运行时间要求达到99.9999%
- 约束故事不需要估算
- 约束故事单列显示

Product Backlog

Measures

- 项目中所有期望的任务的列表
  - 通常是以下的组合
    - Story-based任务 (“let user search and replace”)
    - Task-based任务 (“improve exception handling”)
  - 包括功能性与非功能性需求
  - *“As a <role> I want <ability> so that <benefit>”*
- 由product owner排出优先级
- 高优先级的比较详细
- 任何人都可以提出建议
- 派生自商业计划或vision描述
  - 有时和客户一起创建
- 持续变化



Product Backlog样例

Measures

Backlog Items	Estimate
允许游客下订房间的订单	3
作为游客，我可以取消订单	5
作为游客，我可以修改订房的日期	3
作为酒店员工，我可以查看订单的报表	8
作为酒店经理，我可以查看每月的订单情况	8

练习：制定产品Backlog

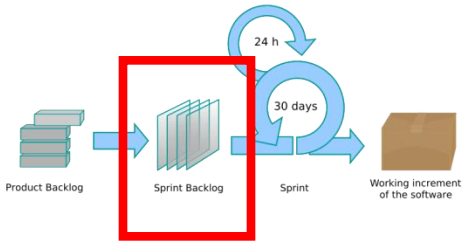
- 时间：20分钟
- 分组：不超过7个人一组
- 任务：完成家庭财务软件的产品backlog
  - *“As a <role> I want <ability> so that <benefit>”*
    - -确定每项的优先级
    - -估计每个User story的故事点与工作量
    - -不能少于10个User Story

文档-Sprint backlog

Measures

The sprint backlog

- 细化高优先级的product backlog
- 任务分解（hours/points）
- 任务分解不超过16小时 / 5-8 点
- 团队成员选择任务
- 在sprint 策划会议上达成一致
- 在Sprint期间可以增加任务不可以修改或增加需求
- 如何保证任务拆分的完备性？



- Develop level selection component (5 points)
- Develop figure assets (5 points)
- Develop game model (8 points)
- Figure entity (3 pts)
- Pile data structure (5 pts)
- Develop game plane (3 pts)
- Render pile (5 pts)
- Control falling figure (13 pts)
- ……

用户故事拆分为任务

Measures

- 用户故事：
  - 用户可以根据不同的字段搜索酒店
- 任务：
  - 编写基本搜索界面
  - 编写高级搜索界面
  - 编写搜索结果的界面
  - 为支持基本搜索查询数据库编写调试SQL语句
  - 为支持高级搜索查询数据库编写调试SQL语句
  - 在帮助系统和用户指南里写下新功能的文档

任务的工作量估算

Measures

任务	责任人	工作量（人天）
编写基本搜索界面	张山	6
编写高级搜索界面	张山	8
编写搜索结果的界面	李斯	6
为支持基本搜索查询数据库编写调试SQL语句	张山	4
为支持高级搜索查询数据库编写调试SQL语句	张山	8
在帮助系统和用户指南里写下新功能的文档	王武	2

在sprint期间

Measures

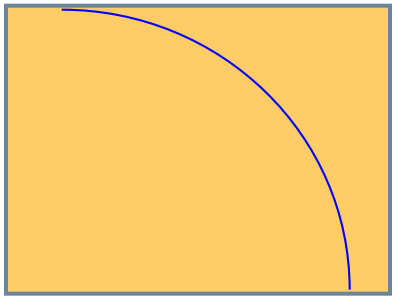
- 变化
  - 为了满足sprint目标，一旦发现需要新增任务，则团队成员可以增加新的任务
  - 可以删除不必要的任务
  - 但是Sprint Backlog只能由团队修改
- 如果有了新信息则变更估算

文档-burn down charts

Measures

Burn Down charts

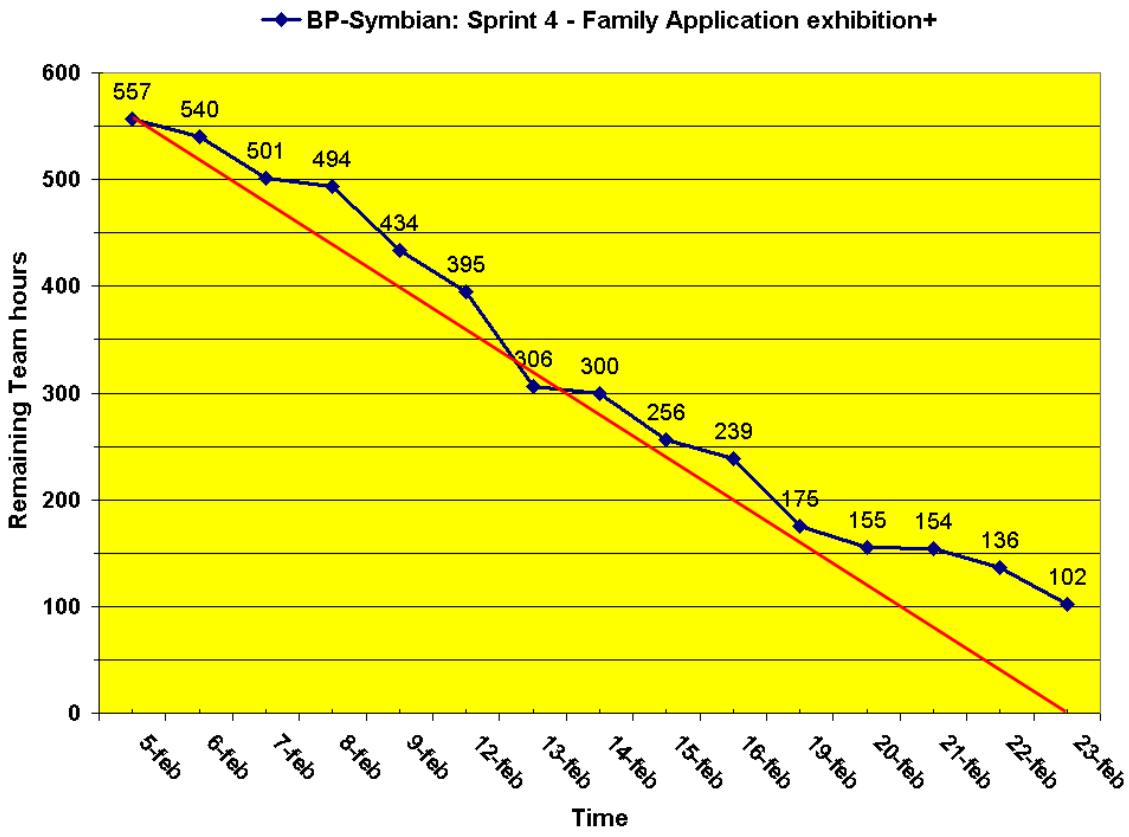
- 团队跟踪sprint进展
- 风险管理工具
- 如果工期提前，我们可以再Sprint Backlog底部增加故事
- 如果工期落后，我们可以再Sprint Backlog底部删除故事
- 图形化展示
- 对每个人要明显可见！



Burn down Charts案例

Measures

- 每次Daily Scrum后修改
- 在sprint期间是主要的沟通工具
- 可以采用excel或在墙纸上维护



何谓“任务已完成”？

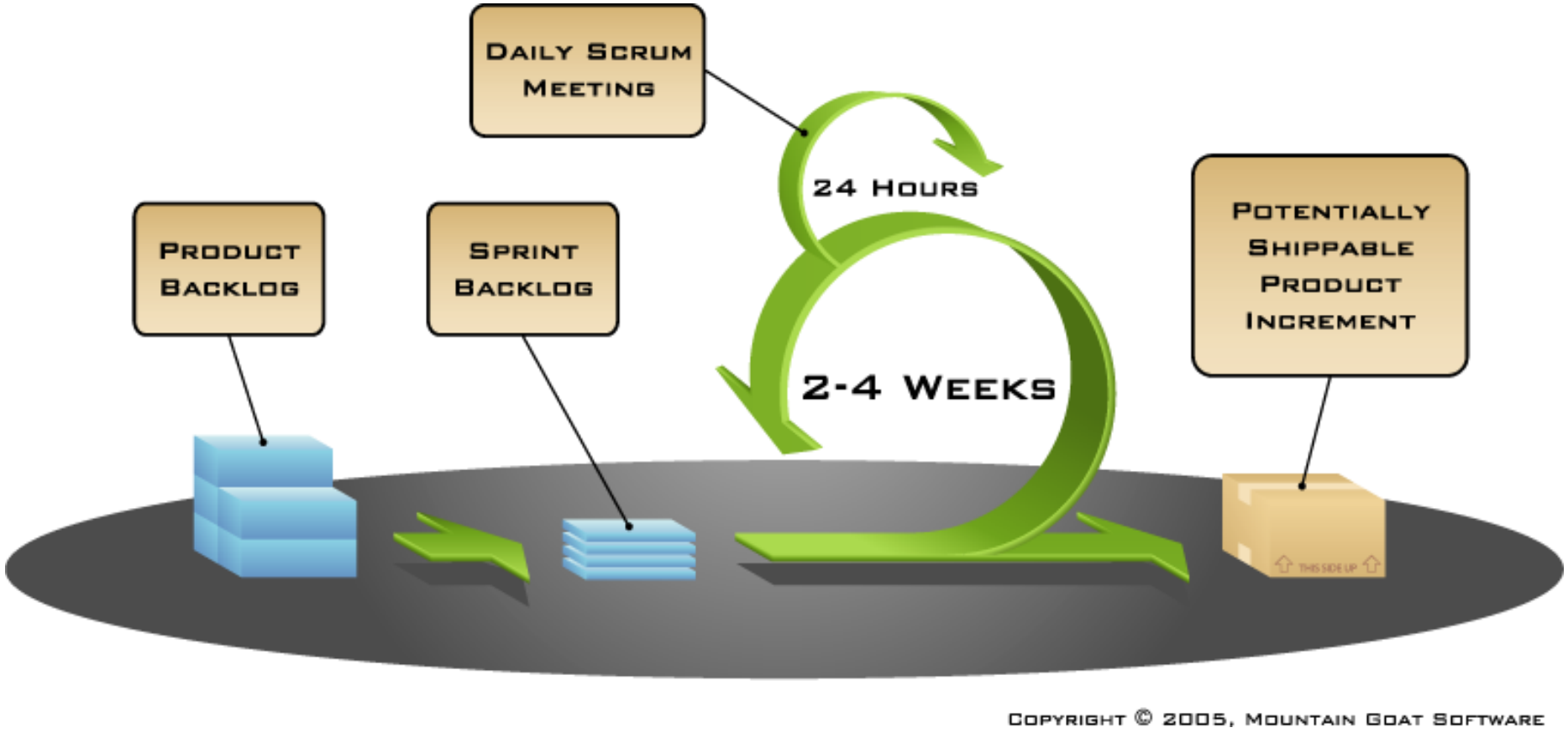
Measures

- 对于“已完成”应该明确定义，不同的团队定义可能不同
- “已完成”的任务是“潜在可发布”
- 至少应该是：
  - 构造完成
  - 测试完成
  - 被Product Owner认可

Measures

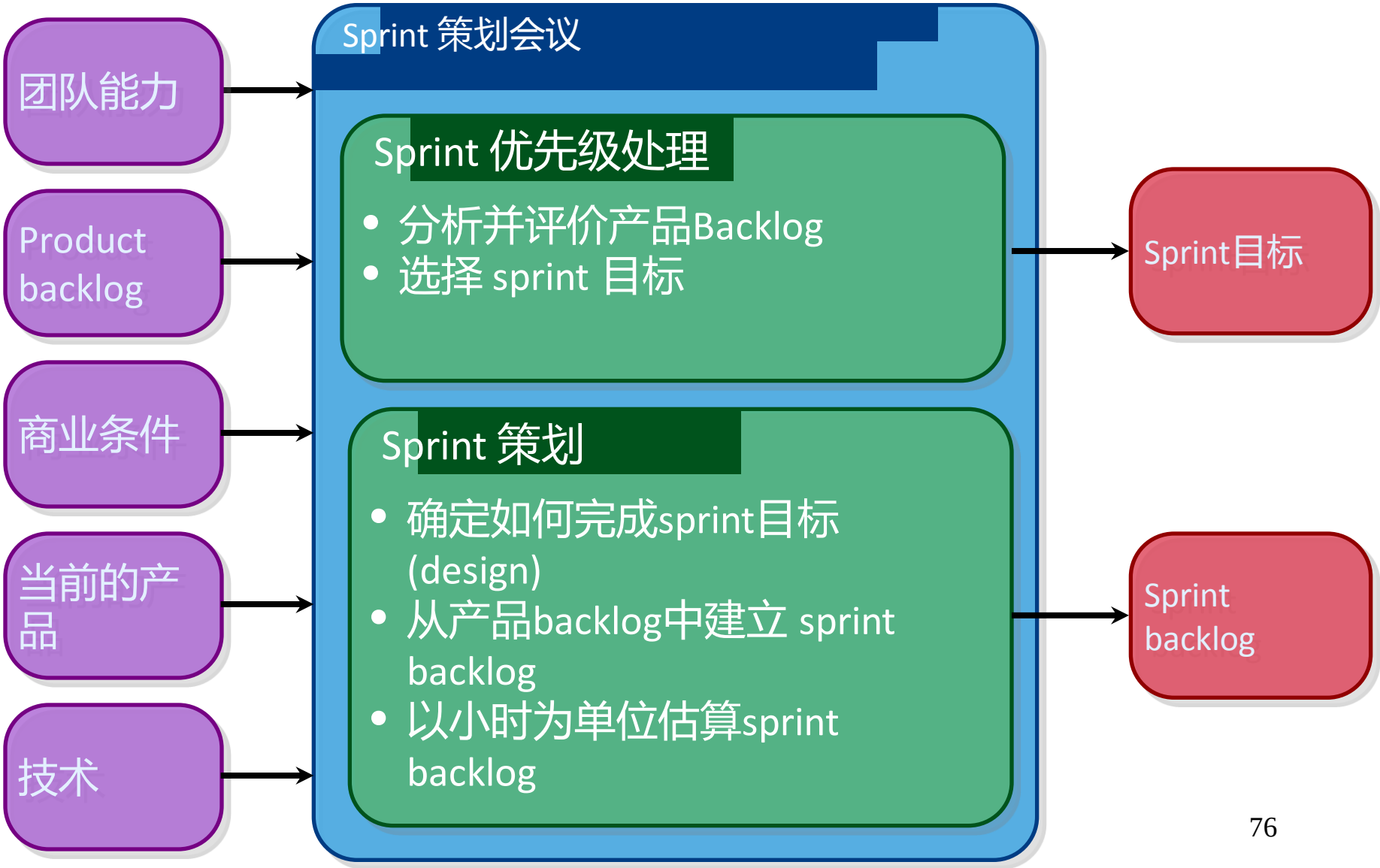
Scrum过程

Scrum雪人模型



75

Sprint策划会议


团队能力
Product backlog
商业条件
当前的产品
技术
Sprint 策划会议
Sprint 优先级处理
• 分析并评价产品Backlog
• 选择 sprint 目标
Sprint 策划
• 确定如何完成sprint目标 (design)
• 从产品backlog中建立 sprint backlog
• 以小时为单位估算sprint backlog
Sprint目标
Sprint backlog

Sprint策划会议

Measures

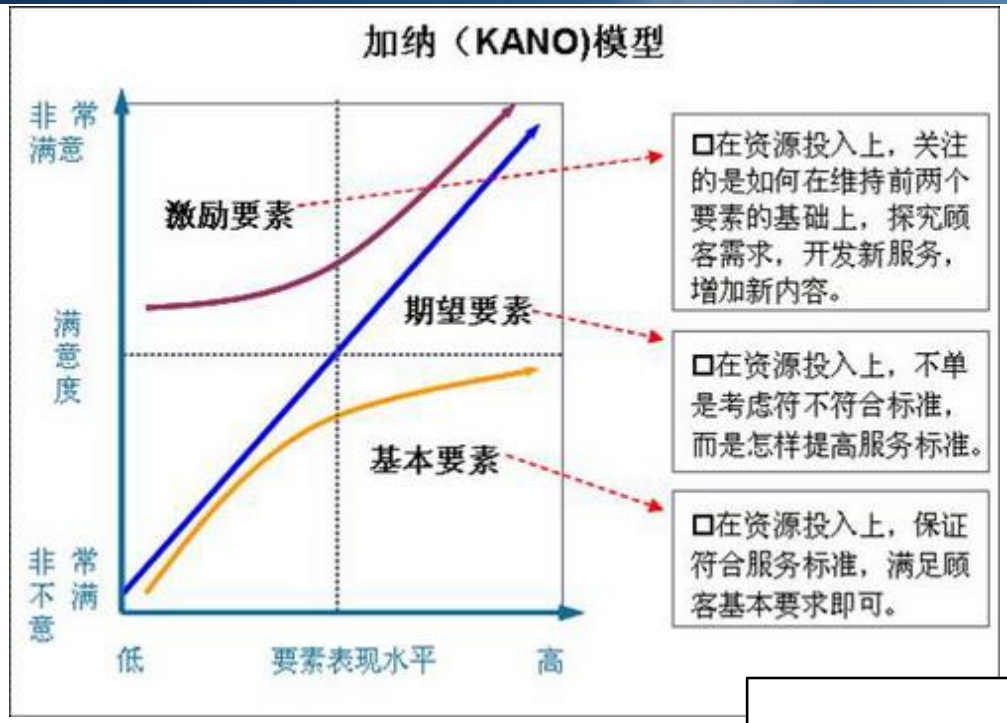
- 现时8小时，分成2部分，各4个小时
- 第一部分,挑选产品的Backlog
- 第二部分,准备Sprint Backlog
- 与会人员
  - Scrum Master, Product owner和团队
  - 可以邀请其他人补充业务或技术领域的信息与建议，提供完信息后离开。
  - 鸡类人员不能发表意见
- 产品负责人要提前在会议前准备好产品Backlog。如果缺少产品负责人或产品Backlog，并代理product owner
- 团队可提出建议，但是由product owner制定产品Backlog
- 团队负责从Product owner 制定的Backlog中挑选出期望在当前sprint内完成的工作

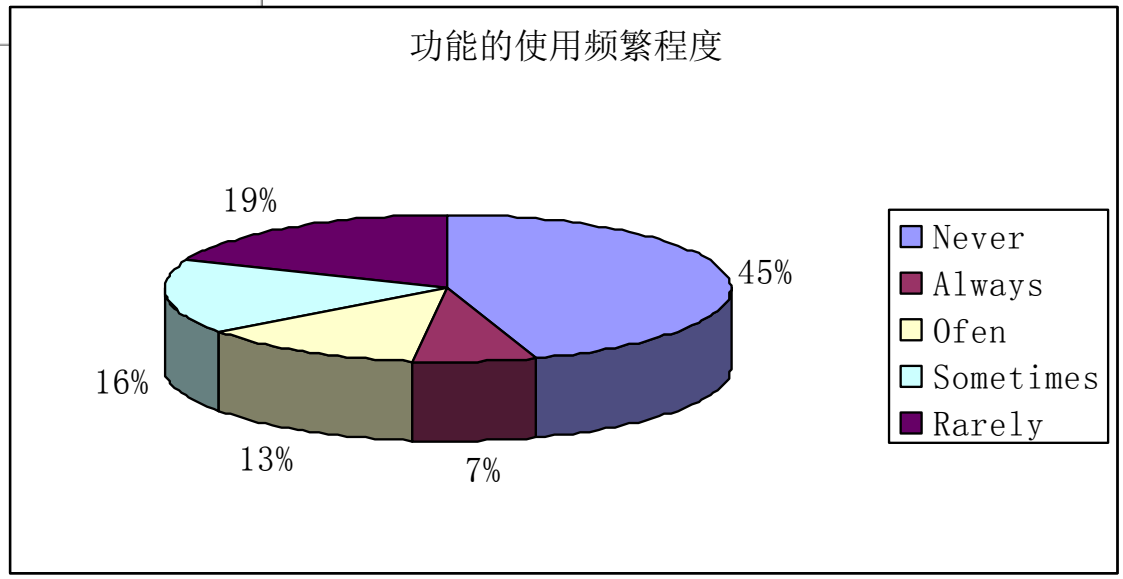
Sprint策划会议：第一部分

Measures

- 目标是挑选将承诺转化为潜在可交付产品功能增量的产品backlog条目
- 确定sprint目标
- 如果没有划定优先级，则划定优先级
- 如果没有估算工作量，则估算产品backlog中的条目
- 分析产品Backlog，沟通产品Backlog
- 成员一致认可sprint目标和产品backlog

划分需求的优先级


加纳（KANO)模型
非常满意
满意度
非常不满意
激励要素
期望要素
基本要素
低
要素表现水平
高
□在资源投入上，关注的是如何在维持前两个要素的基础上，探究顾客需求，开发新服务，增加新内容。
□在资源投入上，不单是考虑符不符合标准，而是怎样提高服务标准。
□在资源投入上，保证符合服务标准，满足顾客基本要求即可。


功能的使用频繁程度
19%
45%
16%
13%
7%
Never
Always
Ofen
Sometimes
Rarely

划分用户故事的优先级

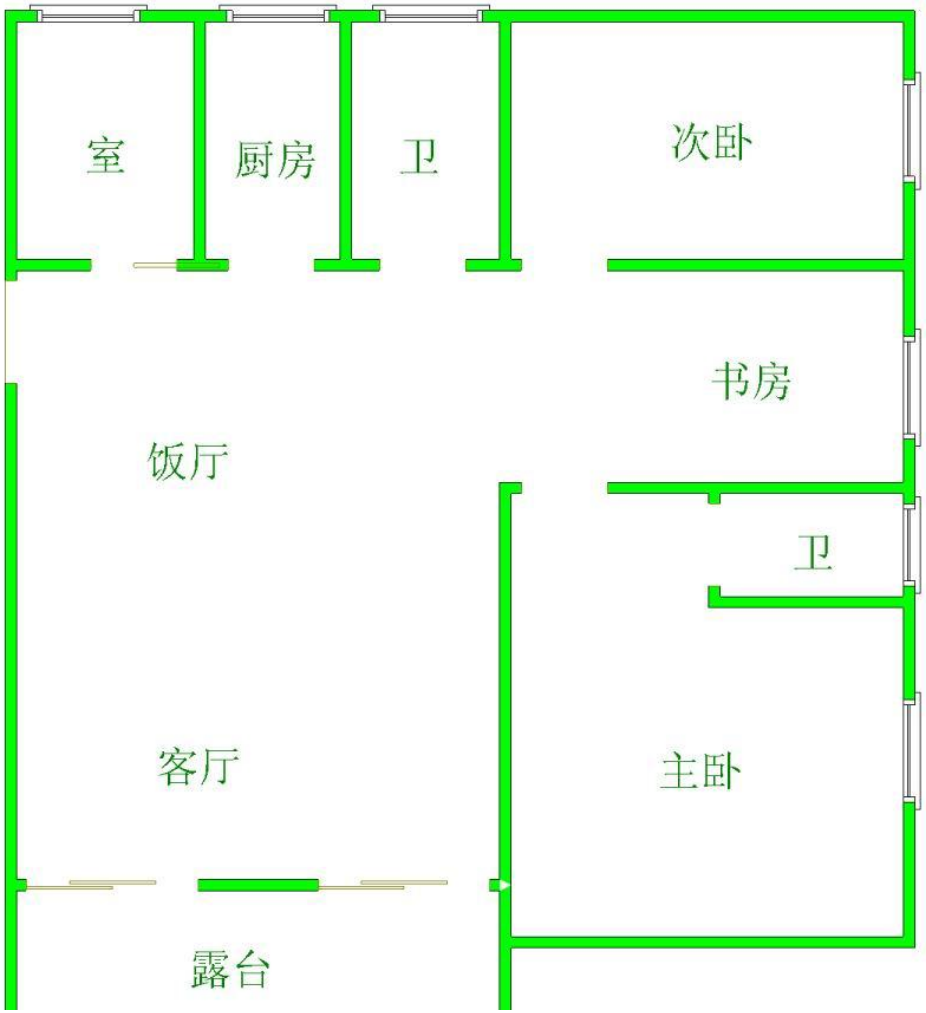
Measures

- MoSCoW规则
  - 必须有（Must have），系统的基本功能
  - 应该有（Should have），重要但是短期内有替代解决方法的功能，如果项目没有时间约束，通常认为应该有的功能是强制性的。
  - 可以有（Could have），如果没有时间就可以在发布中不予考虑的功能
  - 这次不会有（Won’t have this time），客户期望拥有但同时承认需要在后续发布中实现的功能

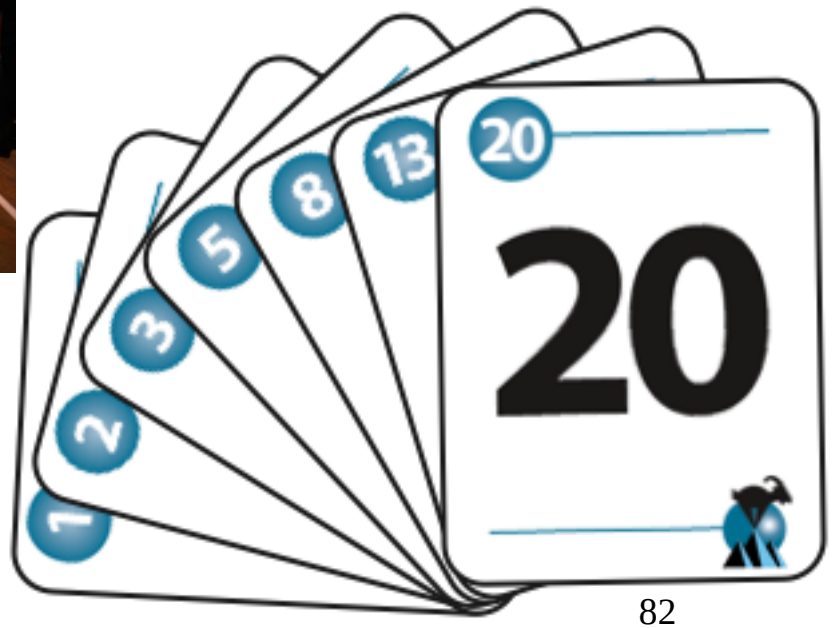
策划扑克法

Measures

- 粉刷这幢房屋需要多久？



策划扑克法



故事点

Measures

- 故事点只是一个计量单位的名称而已，你也可以给他命名为其他名字。
- 故事点其实不仅仅是对规模的度量，也包括了对需求复杂度等其他因素的度量。
- 故事点并非业界统一的一个度量单位，不象度量长度的单位：米，大家都知道1米有多长，你说的1米和他说的1米是等长的。
- 故事点仅对本项目具有近似相等的规模，不同的项目所定义的故事点很可能是不等的。

参与人员

- 包括了所有开发人员：
  - 程序员
  - 测试人员
  - 数据库工程师
  - 分析师
  - 用户交互设计人员等等，
- 在敏捷项目中一般不超过10人。
- 产品负责人参与策划扑克法但是并不作为估算专家。

步骤

Measures

① 每位参与估算的开发人员发放一副估算扑克，扑克上边的数字标为斐波那契序列：1，2，3，5，8，13，20，40。

② 选择一个比较小的用户故事，确定其故事点，将该故事作为基准故事。

③ 选择一个用户故事。

④ 主持人朗读描述，主持人通常是产品负责人或分析师，当然也可以是其他任何人，产品负责人回答估算者提出的任何问题，大家讨论用户故事。

⑤ 每个估算者对该用户故事与基准故事进行比较，选择一个代表其估算故事点的牌，在主持人号令出牌前每个人的牌面不能被其他人看到，然后大家同时出牌，每个人都可以看到其他人打出的牌。

⑥ 主持人判断估算结果是否比较接近，如果接近则接受估算结果，转向（3）选择下一个故事，直至所有的用户故事都估算完毕，否则转向（7）。

⑦ 如果结果差异比较大，请估算值最高及最小的估算者进行解释，大家讨论，时间限定为不超过2分钟。如果大家同意，也可以对该用户故事进行更细的拆分。

⑧ 转向（5），一般很少有超过3轮才收敛的现象。

估算工作量

Measures

- 在估算完故事点后，可以凭经验估算一个故事点的开发工作量，从而得到所有的用户故事的工作量。
- 也可以进行试验，试着开发一个用户故事，度量花费的工作量，得到开发效率，即在本项目中一个故事点需要花费多少工时，再去估算所有故事的工作量。

练习：策划扑克法

- 30分钟
- 5-10人一组
- 针对家庭财务软件的用户故事采用策划扑克法进行规模估计与工作量估计。
- 假设：
  - 该系统为运行在手机中的系统
  - 采用C++语言开发
- 输出：
  - （1）每个用户故事的故事点
  - （2）每个用户故事的工作量估计
  - （3）该方法的优缺点

Sprint策划会议：第二部分

Measures

- 在第一部分会议结束后立即召开第二部分会议
- Product owner必须出席，回答团队对产品backlog可能的疑问
- 团队成员将选定的产品backlog细分为任务：
  - 编码
  - 测试
  - 代码评审
  - 会议
  - 学习新技术
  - 编写文档
  - ……
- 制定出sprint backlog，包括了任务、任务估计、任务分工。
- 任务的估计工时不超过1天，否则细分之
- 任务清单不一定完整，但是必须反应团队全体成员的共同承诺。
- 该过程中团队不受外界指导，其他人只能观察或回答团队寻求进一步信息的问题
- 在过程中团队可为sprint backlog添加其他任务

Sprint Backlog案例

Measures

Sprint Goal:用户可以根据不同的字段搜索酒店

任务	责任人	工作量（故事点）
编写基本搜索界面	张山	6
编写高级搜索界面	张山	8
编写搜索结果的界面	李斯	6
为支持基本搜索查询数据库编写调试SQL语句	张山	4
为支持高级搜索查询数据库编写调试SQL语句	张山	8
在帮助系统和用户指南里写下新功能的文档	王武	2

估算

Measures

- 只能有团队成员估算
- 策划扑克
- 由于没有经验第1次次比较困难
- 作为策划活动的一部分
- 经常做就会越做越好
- 在短时间内估算故事
- 采用Fibonacci序列进行相对性估算
- 基于估算结果可能修改 Product Owner排定的优先级

练习：制定Sprint的Backlog

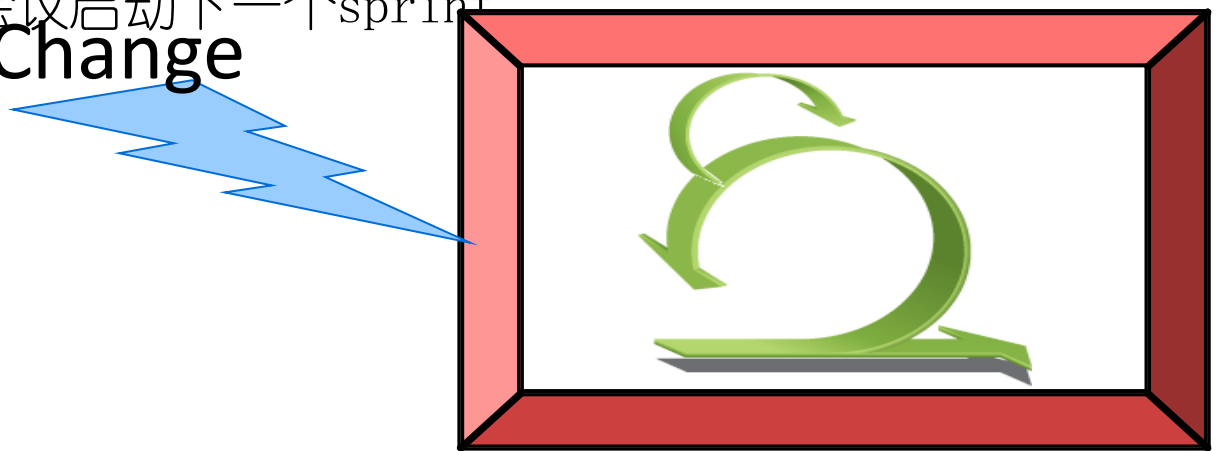
Measures

- 时间：40分钟
- 分组：不超过7个人一组
- 任务：完成家庭财务软件的第1次Sprint的backlog
  - 选择在本次Sprint中拟完成的User Story
  - 对每个User story 进行任务细分
  - 各位成员认领任务

Sprint

Measures

- 固定30天的日历时间，开发可交付的产品
- 短的sprint=短周期反馈=更频繁的交付=更频繁的客户反馈=在错误方向上花费的时间更少=学习和改进的速度更快
- 在Sprint内包括了设计、编码、测试、以及编写文档等工作
- 团队是受保护的
  - 在sprint期间其他人不可向团队下达通知、指令、评论和方向指示，团队完全进行自我管理
- 一旦Sprint开始了，只能由 Scrum团队增加或删除 Sprint backlog中的任务
- 除非Sprint的目标不再有意义，否则不会中止Sprint
- 若sprint发生异常，scrum master可非正常中止sprint，召开新的策划会议启动下一个sprint



Sprint

Measures

- 团队若认定无法在sprint内兑现产品backlog，可与产品负责人协商从当前的sprint中删除部分条目。如删除的条目过多，导致sprint失去价值和意义，scrum master应按前述方法非正常中止sprint
- 若团队认定可在sprint内超额处理sprint计划会议选定的产品backlog，可与产品负责人协商为sprint添加额外的条目
- 团队成员在sprint中有两项行政职责
  - -参加daily scrum 会议
  - -在配置库中及时更新sprint backlog
    - 将新设想的任务加入sprint backlog
    - 更新各项任务的预估剩余时间

每日 Scrum 会议

Measures

- 限时15分钟
- 每日早晨，同一地点
- 站立会议
- 所有成员必须参加
- 成员必须准时，否则惩罚
- 每个人给全体成员汇报工作进展
- 每个人都必须回答且仅回答三个问题
- 围绕三个问题讨论
- 同步进展而不是解决问题
- 只有一个声音

每日 Scrum 会议

Measures

- 鸡和猪都参加，只有猪发言
  - 帮助避免其他不必要的会议
- 鸡类人员站在团队外围，以免干扰会议
- 如果出席会议的鸡类人员过多，scrum master有权限制出席人数，确保会议有序集中
- 鸡类人员不得在会后请团队成员深入解释或向团队提供建议和指示
- 不遵守上述规定的猪类或鸡类人员将被禁止参加会议（鸡类）或被开除出团队（猪类）
- 可能在会议后再举行解决问题的讨论会，只有相关人参加

每个人都要回答3个问题

Measures

1. 昨天你做了什么？
2. 今天你将做什么？
3. 有什么障碍吗？

这些不是汇报给SCRUM master，大家是对等的

关于 Scrum meetings的问题

Measures

- 为什么是每天?
  - “How does a project get to be a year late?”
    - “One day at a time.”
      - Fred Brooks, *The Mythical Man-Month.*
- Scrum meetings 是否可以用邮件状态报告替代?
  - No
    - 整个团队每天都能看到完整的现状
    - 给每个人创建对等的压力对你将要兑现的承诺

1小时内作出决定

Measures

- 需要scrum master出面处理的障碍最好马上做出答复，或者是在一个小时内作出答复
- 错误的决定要好于不做出决定

练习：Scrum Daily meeting

Measures

- 时间：15分钟
- 角色：
  - -Product owner
  - -Scrum Master
  - -Team member
- 任务：
  - -每个人更新自己每个任务的完成情况
  - -Scrum Master更新燃烧图
  - -每个成员向本组其他成员通报情况

Sprint评审

Measures

- 限时4小时
- 团队向产品owner及其他利益相关者演示本次sprint”已完成”的功能
- 不展示未完成的功能，不展示技术文档
- 演示结束后调查利益相关者的意见，记录他们的思想、期望的变更及优先级
- 基于反馈调整Backlog
- 确定团队的能力
- 在每个sprint结束时的非正式会议
- 参与的人员
  - 客户
  - 产品owner
  - 管理者
  - 团队
- 组织可以再sprint结束后确定是否还要继续本产品的开发
- 不需要用ppt！

sprint演示的作用

Measures

- 团队的成果得到认可，他们会感觉很好
- 其他人可以了解你的团队在做些什么
- 演示可以吸引利益相关者的注意，并得到重要的反馈
- 演示是一种社会活动，其他团队可以在这里互相交流
- 演示迫使团队真正完成一些工作，进行发布，而不是完成99%，不是貌似完成
- 如果本次演示很令人尴尬，下次才能做的更好！

Sprint review的注意事项

Measures

- 确保清晰阐述了sprint目标，如果有人对产品一无所知，就多花几分钟来描述
- 不要花太多时间准备，不需要准备ppt
- 关注与业务需求的满足，而不是技术细节。重点在于展示“我们做了什么”，而不是“我们怎么做的”
- 可以的话，让观众试用一下产品
- 不要不重要的特性上花费太多的时间，要关注最重要的特性
- 如果是一些不可演示的需求，那就要想办法展示一些测试报告等，总之，要让大家确信你真的实现了可以满足需求的软件

SCRUM方法中的时间箱管理

Measures

- 每次迭代必须在固定的时间内完成，比如2周或1个月等，本次迭代必须交付一个质量得到充分检验的、可以运行的软件版本，如果有些需求不能在本次迭代内完成，则推迟到下一个迭代中完成。
- 项目的策划会议必须在4个小时内完成，某次迭代的策划会议必须在4个小时内完成。
- 每天15分钟的站立会议。
- 在每日站立会议上发现的问题要在1天内解决。
- 1个小时内做出决策，需要项目的负责人或教练对于管理的问题在1小时内做出决策，不能拖延决策。
- 在每日站立会议上提出的障碍最好在下一个会议前被排除
- 每次迭代结束后的评审会议必须在2个小时内结束。在迭代的评审会议上主要是demo本次迭代完成的产品或产品构件，获得客户及相关人员的反馈。
- 每次迭代结束后的总结会议必须在2个小时内结束，通常是30分钟内结束。主要是总结本次迭代的经验教训，下次迭代能够做的更好。

时间箱管理的好处

Measures

- 克服帕金森定律
  - 人们总是用完所有可以利用的时间
    - 帕金森经过多年调查研究，发现一个人做一件事所耗费的时间差别如此之大：他可以在10分钟内看完一份报纸，也可以看半天；一个忙人20分钟可以寄出一叠明信片，但一个无所事事的老太太为了给远方的外甥女寄张明信片，可以足足花一整天：找明信片一个钟头，寻眼镜一个钟头，查地址半个钟头，写问候的话一个钟头零一刻钟……特别是在工作中，工作会自动地膨胀，占满一个人所有可用的时间，如果时间充裕，他就会放慢工作节奏或是增添其他项目以便用掉所有的时间。
- 人们往往记住失误的日期而不是失误的特征
- 降低问题的复杂度
- 尽早促成难度大的决策和权衡

Sprint回顾

Measures

- 限定在3个小时内,通常15-30分钟
- 每次sprint之后
- 参与的人员Team, Scrum Master, 以及 Product Owner（可选）
- 三个问题：
  - 回顾历史：
    - 成功之处有哪些？需要坚持做
    - 有哪些错误的做法？需要停止的做法
  - 展望未来
    - 有哪些改进之处？需要采取新措施
- 对于识别的措施，定义责任人，有可能需要公司改进

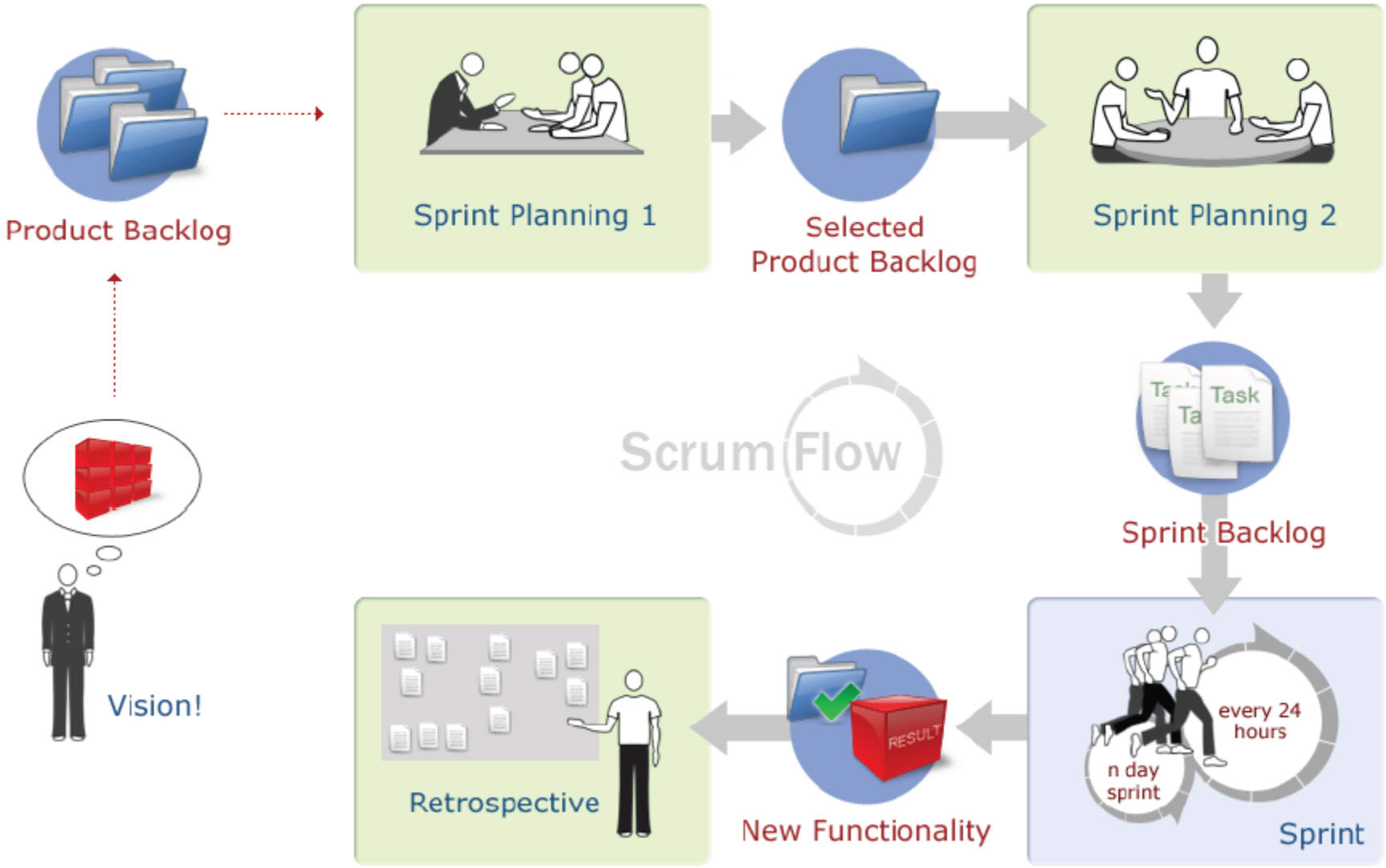
SCRUM方法中的4种会议小结

Measures

	时机	活动名字	主持人	参与的人员	主要任务	输出
①	迭代开始之前	sprint planning	Scrum master	项目组的全体成员	需求讲解； 估算； 开发顺序安排； 任务分工；	规模与工作量估算结果； sprint backlog；
②	每天	daily meeting			跟踪进展；	burn down chart
③	迭代结束	Sprint review		项目组的全体成员、客户或者用户、其他利益相关者	演示软件，评审完成的功能	修改后的product backlog； 可执行的软件
④	迭代结束	Sprint retrospective		项目组的全体成员	总结本次迭代的经验教训	经验教训总结

Scrum总结

Measures



Measures

Scrum 模拟

Scrum 模拟

Measures

- 不超过7人一组
- 选一个人作为PO，一个作为SM，其他人作为组员
- 做3天的sprint
- 在20分钟内完成user story的定义
- 在20分钟内完成一个策划会议
  - -对优先级排在前5位的需求进行估算
  - -将每个需求拆分为2-3个任务
- 7分钟内完成第1天的任务，开2分钟的站立会议
- 7分钟内完成第2天的任务，开2分钟的站立会议
- 7分钟内完成第3天的任务，开2分钟的站立会议
- 在10分钟内完成sprint review与sprint retrospective

User story的定义

- 分配角色:
  - -Product owner
  - -Scrum Master
  - -Team member
- 为一套三室一厅的房子进行装修设计，请在20分钟内完成用户故事的定义并划定优先级。

Sprint 策划会议

Measures

- 请在20分钟内完成第一次sprint的策划
  - 对用户故事进行估算
  - 迭代的周期为3天
  - 挑选出第1次迭代要完成的用户故事
  - 对用户故事进行任务拆分
  - 团队成员挑选任务

3天的开发及站立会议

Measures

- 每天开发
  - 团队成员开发
  - Product owner验收
  - 不超过7分钟
- 站立会议
  - 每位成员报告进展
  - 更新燃烧图
  - 不超过2分钟

Sprint 评审与回顾

Measures

- 展示完成的设计
- 总结经验教训
- 不超过10分钟

Scrum不是万能钥匙

Measures

No magic solution

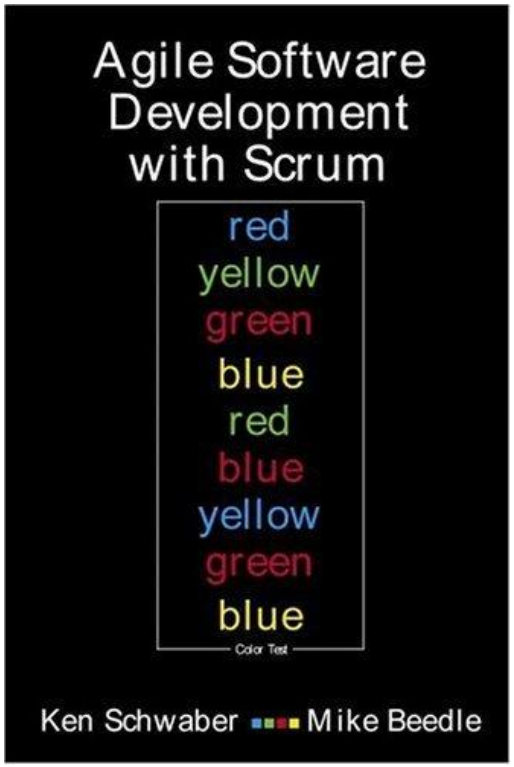
Agile will not solve your people issues, but make then more visible

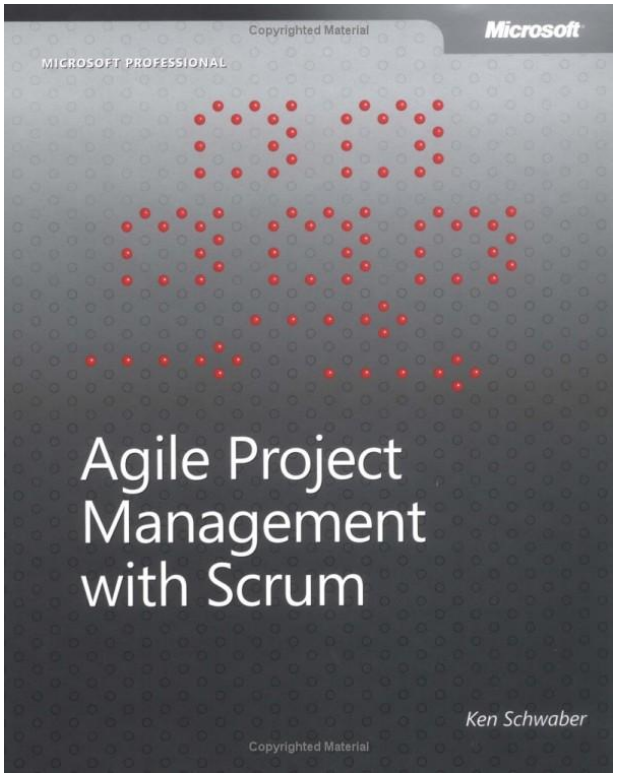
Invest in your people and let them grow

If anything, “The team will do it!” ™

资源指南 – 书籍

- Agile Software Development with Scrum by Ken Schwaber and Mike Beedle
- Agile Project Management with Scrum by Ken Schwaber





资源指南 – 网站

Measures

- Agile Alliance http://www.agilealliance.org
- Scrum web site http://www.controlchaos.com
- Scrum meets RUP
  http://www-128.ibm.com/developerworks/rational/library/feb05/krebs

Measures

极限编程简介

内容

- XP是什么
- XP的过程
- XP的12条实践
- XP的常见错误

极限编程（ eXtreme Programming ）

Measures

- 极限编程（XP）是一种全新而快捷的软件开发方法。XP诞生于1996年。
- XP是以开发符合客户需要的软件为目标而产生的一种方法论。
- XP是一种以实践为基础的软件工程过程和思想。
- XP认为代码质量的重要程度超出人们一般所认为的程度。

谁在用XP

- Fortune 500 公司中成功应用XP的公司包括Ford, Daimler-Chrysler, First Union National Bank, IBM , HP等等。
- 2-10人的小规模开发队伍（小规模开发队伍　小规模项目）。
- 越来越多的公司开始使用敏捷开发过程，或者将其与RUP等开发过程结合使用。

什么时候需要XP？

Measures

- 需求不明确、变化快
- 高风险：在特定的时间内，面对一个相当难开发的系统
- 中小型团队（人数不超过10 个）
- 有责任心的、自觉自励的团队

XP体现四个价值目标

Measures

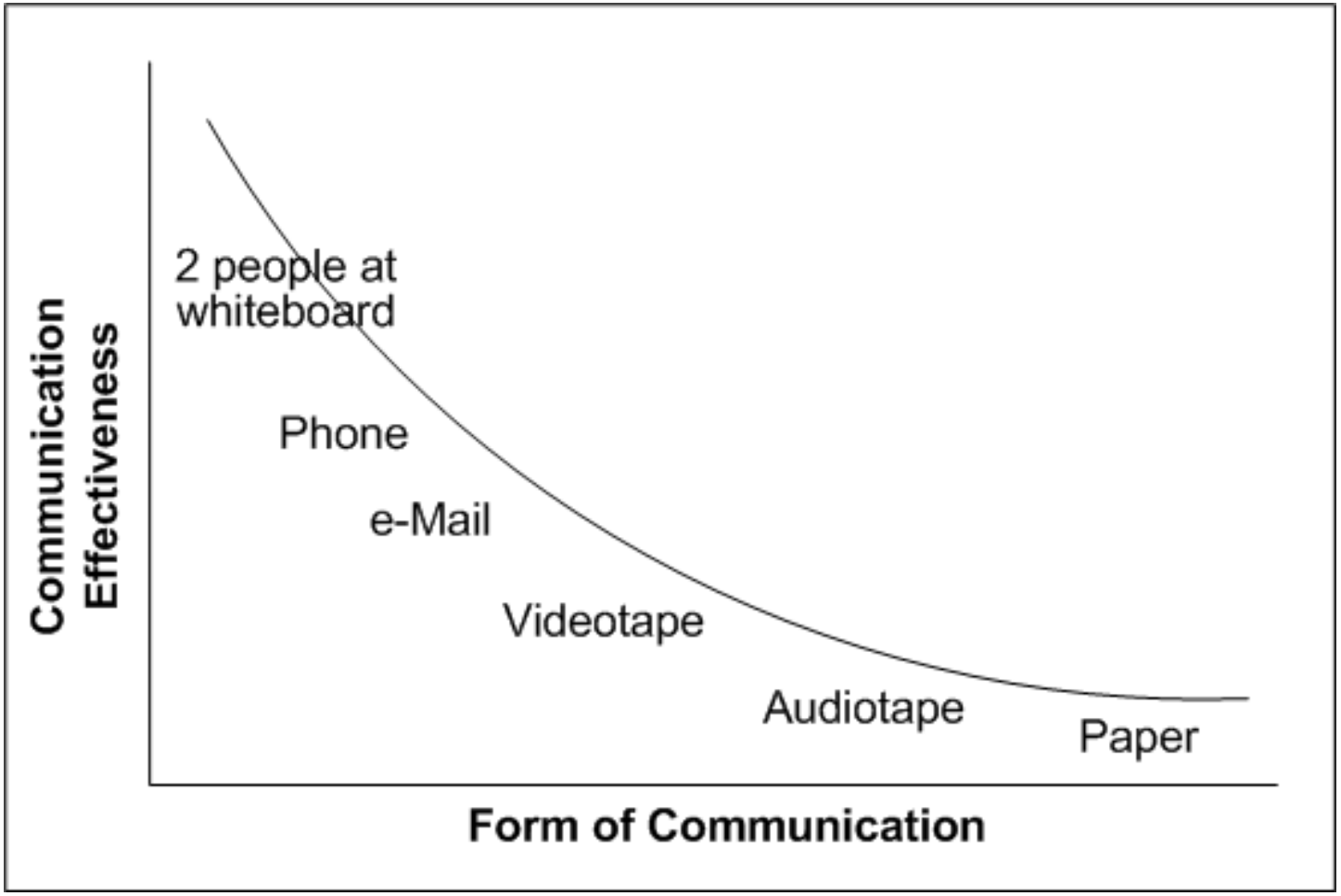
- 沟通 (communication)
- 简单 (simplicity)
- 反馈 (feedback)
- 勇气 (courage)

沟通

Measures

- 项目中出现的问题无一例外总是出自那些不愿与别人探讨重要问题的人身上：
  - 有时程序员不把重要的变化告诉别人；
  - 有时程序员不向客户问该问的问题，而导致在关键性领域的决策中出现失误；
  - 有时管理人员不向程序员问该问的问题，而导致项目进度被误报；
  - 有时程序员向管理人员报告了一个坏消息，而管理人员却迁怒与他；
  - 有时客户告诉了程序员一些重要的事情，而程序员却把他当成了耳旁风；
- 敏捷方法采用了一些实践来强制沟通，比如结对编程、策划游戏、验收测试等。
- 研究表明：对于失控软件，技术人员比管理人员更早觉察到（早72%的时间），但是并没有报告给管理者

沟通方式的有效性


Communication Effectiveness
2 people at whiteboard
Phone
e-Mail
Videotape
Audiotape
Paper
Form of Communication

简单

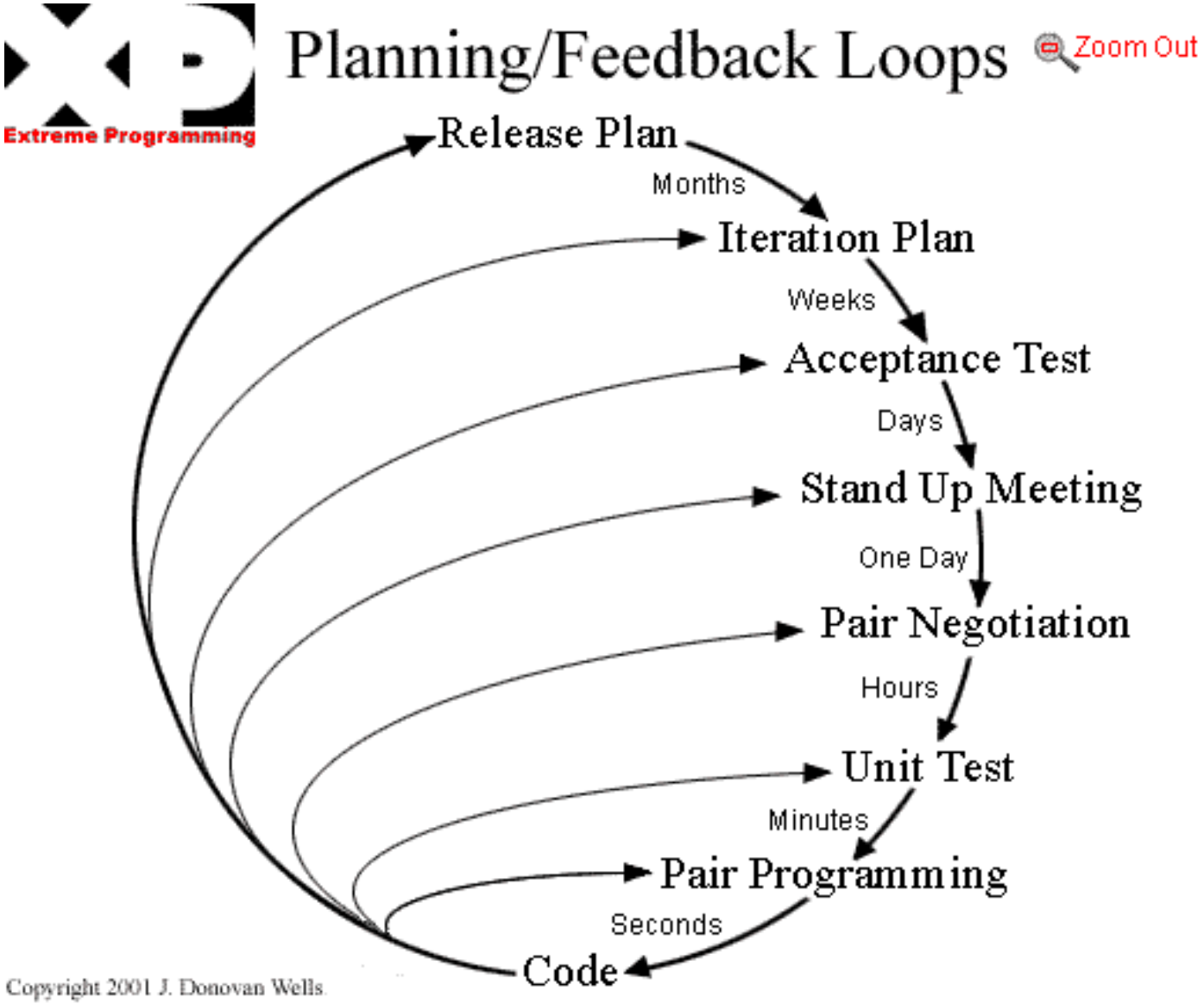
- 什么是能成功的最简单的东西？
- 今天做的简单一些，然后再明天需要时再多花些时间进行改进，要比今天做的复杂，但以后再也用不到要好
- 需求是否可以更简单一些呢？
- 设计是否可以更简单一些呢？
- 沟通的越多，就越清楚哪些该做，哪些不该做
- 系统越简单，需要的沟通就越少

反馈

Measures

- 乐观是大忌，反馈是良药
- 及时反馈
- 反馈的频度：分、小时、天、迭代周期
  - 程序员编写完程序后实时进行单元测试，以得到程序质量的反馈
  - 客户编写完新的用户故事后，程序员马上对其进行估算，以得到故事质量与投入的反馈
  - 程序员完成了用户故事后，客户进行验收测试，以反馈产品的质量
  - 每天的站立会议可以使项目组的所有人了解项目的进展
  - 每次迭代结束了，进行已完成软件的演示，以获得客户的反馈
- 尽早反馈坏消息
- 反馈越多，沟通越容易

XP 的反馈环


Extreme Programming
Planning/Feedback Loops
Zoom Out
Release Plan
Months
Iteration Plan
Weeks
Acceptance Test
Days
Stand Up Meeting
One Day
Pair Negotiation
Hours
Unit Test
Minutes
Pair Programming
Seconds
Code
Copyright 2001 J. Donovan Wells

勇气

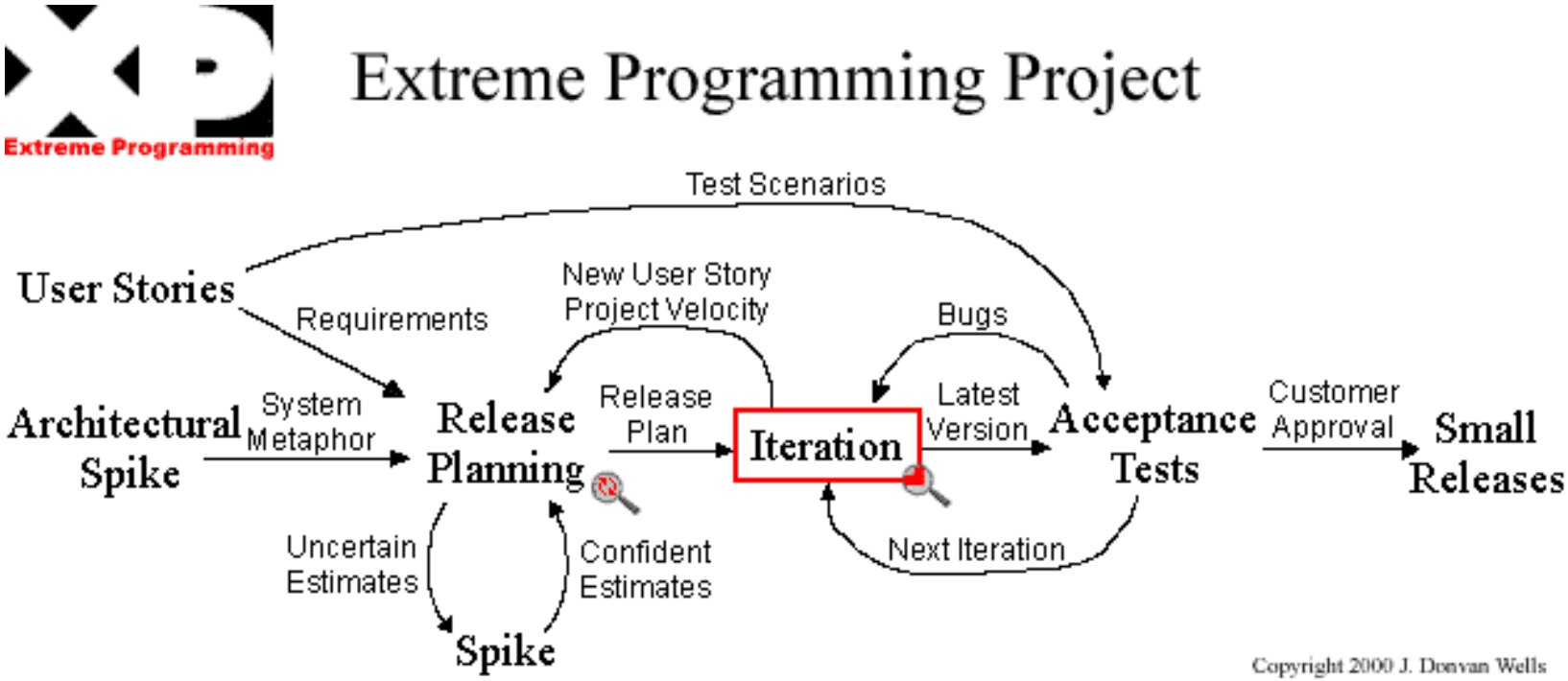
Measures

- 正视现实，而非逃避现实，正视问题，而非逃避问题
- 勇于相信别人
- 勇于采用最简单的方法解决问题
- 勇于少写文档
- 勇于让客户排定优先级，让技术人员做技术决策
- 勇于采用最简单的工具，如白板和纸
- 勇于承认每个人都犯错误
- 勇于承担任务
- 勇于修改Bugs
- 勇于重构
- 勇于应对变化
- 勇于创新，避免教条
- 为胜利而战，而不是为“不犯错”而战
- 满足了沟通、简单、反馈原则之后再具有了勇气，才是真正的有勇有谋

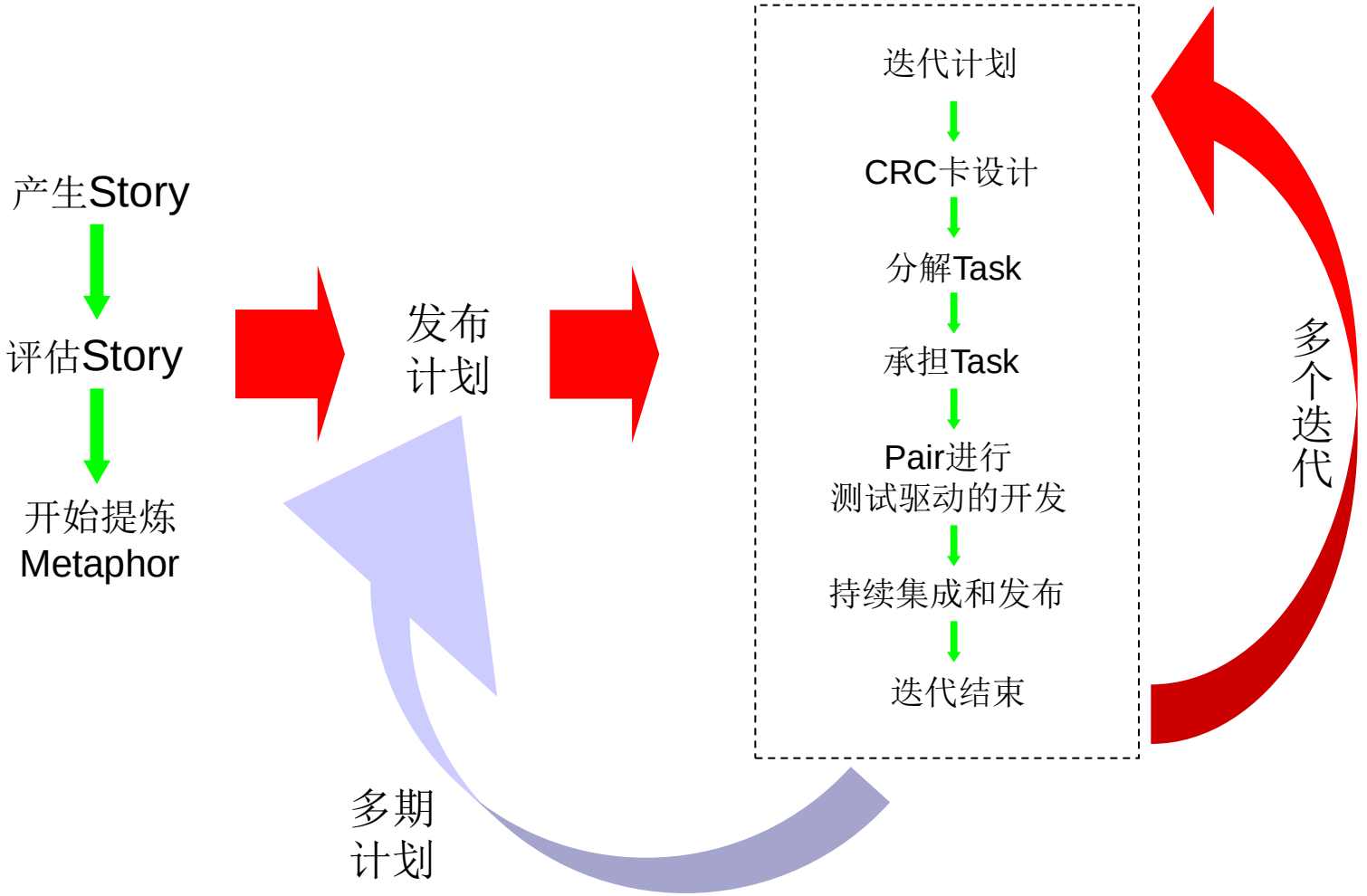
Measures

XP的开发过程

XP的过程


Extreme Programming
Extreme Programming Project
Test Scenarios
User Stories
Requirements
New User Story
Project Velocity
Bugs
Architectural
Spike
System
Metaphor
Release
Planning
Release
Plan
Iteration
Latest
Version
Acceptance
Tests
Customer
Approval
Small
Releases
Uncertain
Estimates
Confident
Estimates
Next Iteration
Spike
Copyright 2000 J. Donvan Wells

XP的开发过程


产生Story
评估Story
开始提炼
Metaphor
发布
计划
迭代计划
CRC卡设计
分解Task
承担Task
Pair进行
测试驱动的开发
持续集成和发布
迭代结束
多个迭代
多期
计划

XP过程样例

Measures

1、探索阶段

(1) 程序员和用户一起把需求分解为User Story，用户写Story卡。

(2) 用户把在Story卡上标上商业优先级(1，2，3)。

(3) 程序员看卡片，估计所需要的理想开发周，把估计的时间写在Story卡上(1，2，3) 。

(3.1) 由一个Senior估计时间。

(3.2) 如果估计不出来，做Spike。

(3.2) 如果时间大于三周，分解Story，撕掉旧Story卡，写下新的Story卡。回到(3)，重新估计。

(3.3) 由一个Junior估计时间。衡量二者的差异，取平均值。

(3.3) 如果时间大于三周，回到(3.2)。

(3.4) 如果时间小于一周，尝试和其它Story合并。合并后，撕掉原来的Story卡，写下新的Story卡。

(4) 程序员看卡片，估计开发风险。把风险级别写在Story卡上(1，2，3) 。

(4.1) 程序员将所有卡片放在一起，然后阅读。如果认为风险低，就放在第二堆，否则不移动。

(4.2) 程序员阅读第二堆卡片。如果认为风险低，就放在第三堆，否则不移动。

(4.2) 对三堆卡片分别标记。

XP过程样例

Measures

2、发布策划

(1)确定该Release中Iteration的长度(例如3周)。计算，开发总时间/迭代长度 = 迭代数。

(2)计算可完成的开发量，告诉客户。

(2.1) 根据经验，估计程序员的平均开发速度(例每个Iteration完成1个理想周)。

(2.2) 计算，平均开发速度 x 程序员数 x 迭代数 = 可完成的工作量。

(3)客户挑选Story卡，累计这些卡片上的估计开发时间，不要超过宣布的可完成工作量。挑选的原则为商业价值优先。

(4)把要完成的Story卡贴在墙上或白板上，要让所有的人都能容易地，清楚地看到。

(5)整个Team，包括客户，在一起做快速设计，并想出Metaphor。

XP过程样例

Measures

3、迭代策划

(1) Tracker宣布该Iteration的计划速度。

(1.1) 如果是第一个Iteration，则计划速度 = 程序员数 x 1。

(1.2) 否则，计划速度 = 上一个Iteration实际完成的理想开发周数目。

(2) Team挑选本次迭代要实现的Story，累计这些Story卡上的估计开发时间，不要超过计划速度。客户挑选商业价值最高的User Story，商业价值相同时挑选开发风险最高的。

(3) 客户宣读并解释User Story卡。

(4) Team用CRC卡或者其它方式设计该Story，并将其分解成Engineering Task，记录在Task卡上。

(5) Tracker宣布各程序员在该Iteration的计划速度。

(5.1) 如果是第一个Iteration，则计划速度 = 5 理想开发天。

(5.2) 否则，计划速度 = 上一个Iteration实际完成的理想开发天数目。

XP过程样例

Measures

3、Iteration Plan

(6)　每个程序员挑选一个Task，并估计所需要的理想开发天。如果大于三天就分解Task，小于1天就和其它的Task合并。

(7)　把天数和程序员的名字写在Task卡上。

(8)　每个程序员重复(6)的过程，直至所有Task的理想开发天的总和等于自己的计划速度。

(9)　留下的Task卡可以放在一个Iteration，或者程序员提前开发完后承担。

XP过程样例

Measures

4、迭代开发

(1) 客户为每个User Story定义功能测试。

(2) 每天早晨开一个简短的Standup Meeting。每个人简要讲述昨天做的工作，今天准备做的工作，和碰到哪些困难。

(3) 程序员进行Pair Programming

(3.1) 根据难度和重要性决定Senior，Intermediate和Junior的搭配。Pair每天互换。

(3.2) Pair自由决定先开发谁的Task。

(3.3) Pair开发时，自由决定谁是Driver，谁是Navigator。

(4) 设计该项Task。

(5) 为该设计的所有类和方法写Unit Test。

XP过程样例

Measures

4、Iteration Development

(6) 写程序实现该设计的所有类和方法。

(7) 运行所有的Unit Test，在100%通过后Check In。

(8) **重构**。完成后执行(7)。

(9) Build系统，只放入一个Pair完成的功能。

(10) 运行**功能测试**，通过则进入(11)，否则如下：

(10.1) 如果失败或者发现BUG，找到原因后要先写Unit Test。

(10.2) 写程序。通过所有Unit Test后Check In。

(10.3) 重新执行(9)。

XP过程样例

Measures

4、Iteration Development

(11) 进行小发布。

(12) 客户使用，给予反馈。

(12.1) 如果发现BUG，进入步骤4-10.1。

(12.2) 在此基础上产生的新需求或变化，做为一个Story或Task。

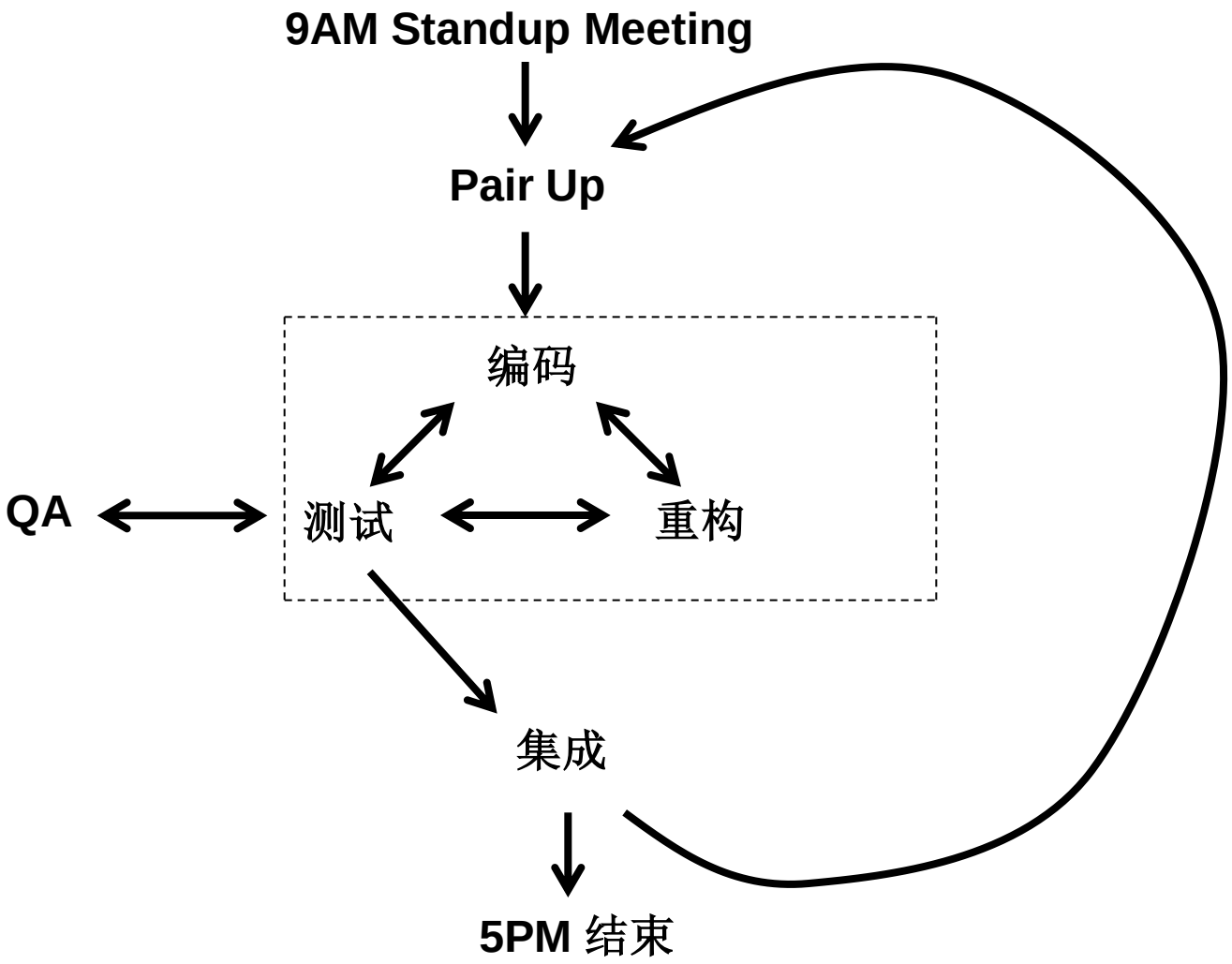
(12.2.1) 客户和Team协商何时评估该Story/Task，以及放在那个Iteration中。

(12.2.1) 评估过程参考2。

(13) Tracker跟踪统计Story和Task的完成情况。

5、开始下一个Iteration，重复4。

程序员的一天



XP的实践

XP的12个核心实践

Measures

- 现场客户(On-site customer)
- 策划游戏(Planning game)
- 小发行版(Small releases)
- 集体代码所有权(Collective ownership)
- 一周40小时 (40-hour week)
- 系统隐喻(System Metaphor)
- 简单设计(Simple design)
- 配对编程(pair programming)
- 编码标准(Coding standards)
- 测试驱动(Test-driven)
- 重构（Refactoring）
- 持续集成(Continuous integration)

On-Site Customer

Measures

客户是Team成员，在开发现场和开发人员一起工作。客户是指定义产品的特性并排列这些特性优先级的人或者团体.

传统的客户任务一般是**讲解需求，运行验收测试，接收发布的系统**。XP新增加的任务：

(1) 写User Story

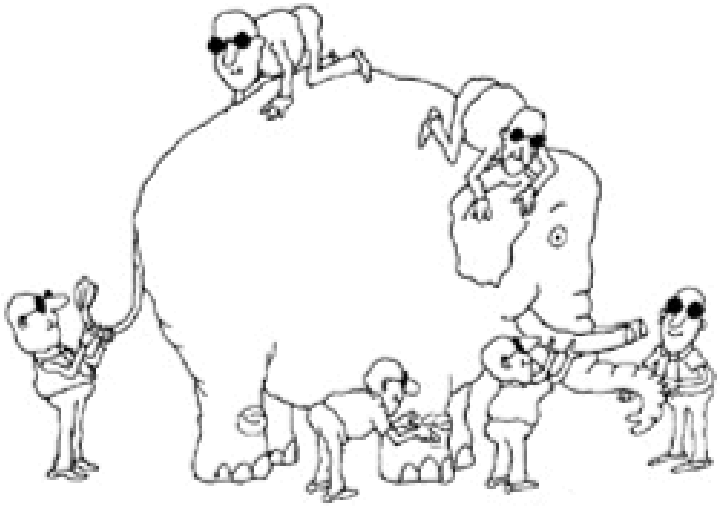
(2) 评估User Story的商业优先级

(3) 为每个User Story定义验收测试

(4) 计划开发内容

(5) 调控开发过程

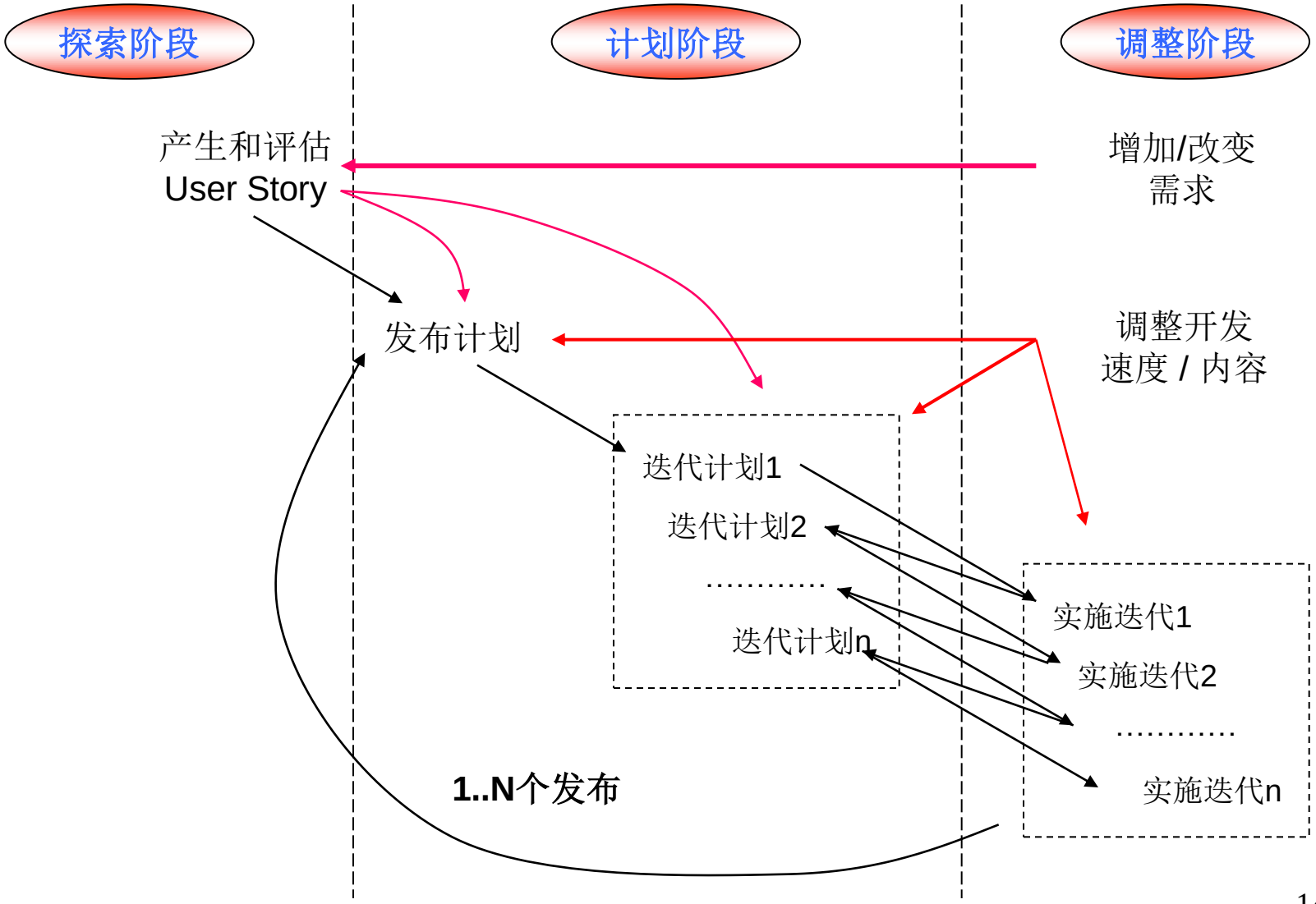
Any More?



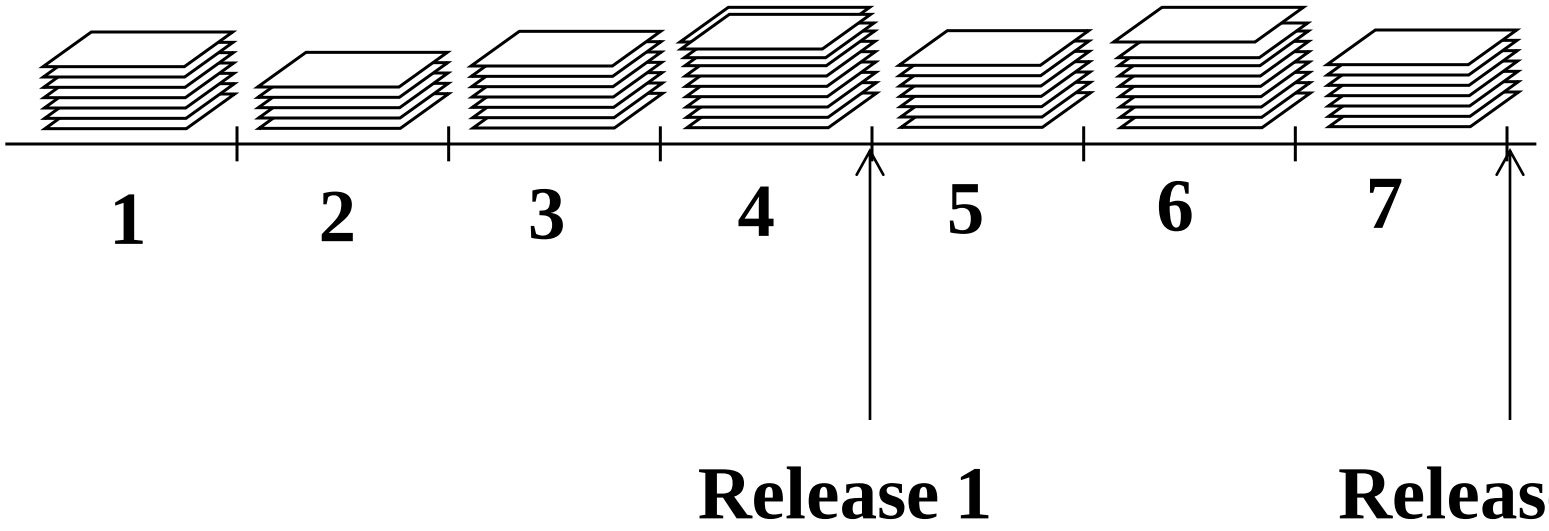
CRACK型客户

- Collaborative:易于协作
- Representative:有代表性的
- Authorized:有授权
- Committed:尽责
- Knowledgeable:在行

计划游戏


探索阶段
计划阶段
调整阶段
产生和评估
User Story
增加/改变
需求
发布计划
调整开发
速度 / 内容
迭代计划1
迭代计划2
.....
迭代计划n
实施迭代1
实施迭代2
.....
实施迭代n
1..N个发布

发布计划



频繁地小规模发布软件

Measures

- 发布过程应该尽可能地自动化、规范化。
- 不断地发布可用的系统可以告诉客户你在做正确的事情。
- 客户使用发布的系统，可以保证频繁地反馈和交流。
- 保证客户有足够的依据调控开发过程(增加、删除或改变User Story)。
- 降低开发风险。
- 随着开发的推进，发布越来越频繁。
- 所有的发布都要经过功能测试。

平稳的工作效率

Measures

平稳的工作效率指Team和个人在很长的时期内保持一定的开发效率。

- 保证了项目速度和计划过程的有效性和准确性；
- 保证了程序员可以持续地完成任务，Team可以持续地向客户交付可运行的系统(见敏捷开发宣言)；
- 加班多导致开发效率和质量下降，简洁增加了开发成本；
- 结对编程已经加大了工作强度，并且和其它XP的规则一起提高了工作效率，使少加班和维持平稳的工作效率可能而且可行。
- 提倡平稳的工作效率，体现了XP以人为本的价值观。
- 每周工作40小时,不提倡加班

System Metaphor

“The system metaphor is a story that everyone - customers, programmers, and managers - can tell about how the system works.” — Kent Beck

隐喻就是不太直白的比喻。隐喻是将整个系统联系在一起的全局视图,是系统的未来景象,隐喻通常归结为一个名字系统。系统隐喻在很大程度上替代了“体系结构”。

Metaphor使客户和程序员用共通的模型和语言进行交流 — “One Team, one language”。

例:

- Market —发布/浏览，价格洽谈，生成和履行合同;
- String, Tree, Package, Chartroom, Spider, Robot .....;
- 电影后期制作 —> 邮递 —> 电影院播放电影。

System Metaphor

Measures

- Metaphor的形成过程，是客户建立并抽象商业模型和商业概念的过程，是程序员建立并抽象设计模型和设计概念的过程。
- Metaphor可以帮助减少“知识泄露”和“支解知识”。
- Metaphor是设计过程的航标 —— 真正灵活有效的设计是针对商业原则的设计，而不是针对商业原则表现形式的设计，更不是脱离商业需求目的的学术设计。
- 随着开发的继续，Team会找到更好的Metaphor。这是知识细化、深化的结果，是“持续学习”(Continuous learning)的过程；是对商业模型和设计模型的持续重构。

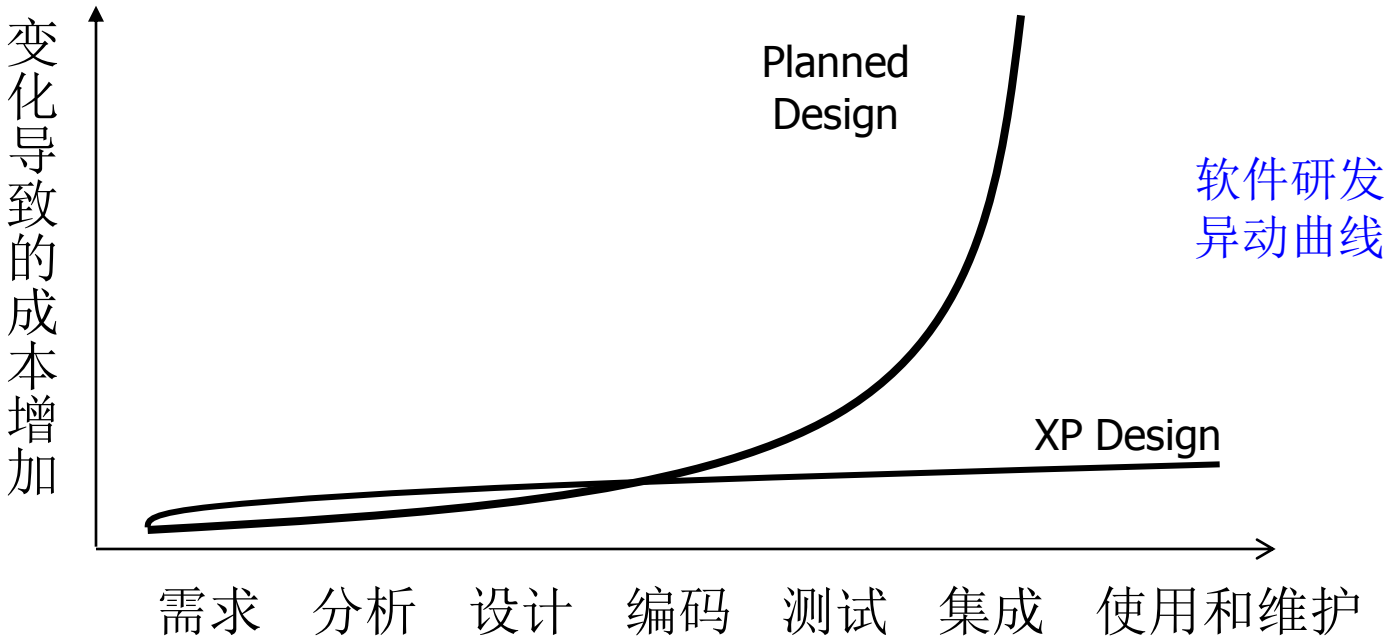
简单设计

Measures

简单设计 —— Do the simplest thing that could possibly work; You aren't going to need it(YAGNI)。

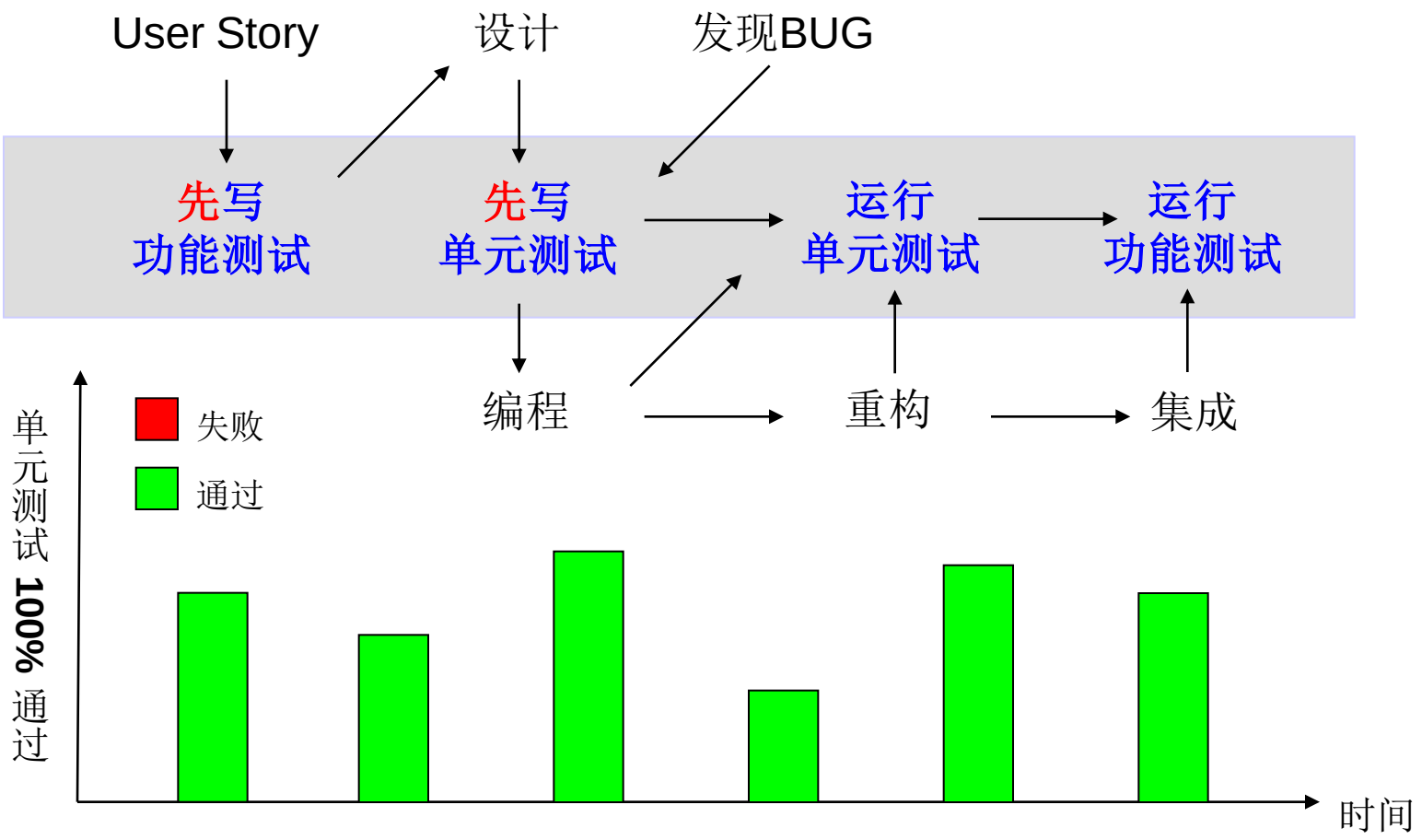
- 标准(依重要性):
  - 通过所有测试
  - 可读性高的代码
  - 避免重复 once only once ,don't repeat yourself
  - 最少数量的类或方法。
- System Metaphor给设计提供了指引，加强Team对设计的理解;
- 第一个迭代搭建了基本的系统框架。
- 以后的迭代过程，是在反馈和编程的基础上做交互式设计，减少了设计的投机性。
- 迭代过程中的CRC卡帮助Team交流设计思想，简化了设计文档。
- 重构对设计进行优化。

简单设计



- 如果没有它和众多核心实践之间的耦合，XP的演化设计就蜕化成CODE-FIX。
- XP的演化设计是在Up-front design和Refactoring之间找到新的平衡。
- 对设计文档没有要求,所以沟通很重要

测试驱动的开发


User Story
设计
发现BUG
先写
功能测试
先写
单元测试
运行
单元测试
运行
功能测试
编程
重构
集成
失败
通过
单元测试 100% 通过
时间

测试驱动的开发

Start

Write a test for new capability

Compile

Fix compile errors

Run the test And see it fail

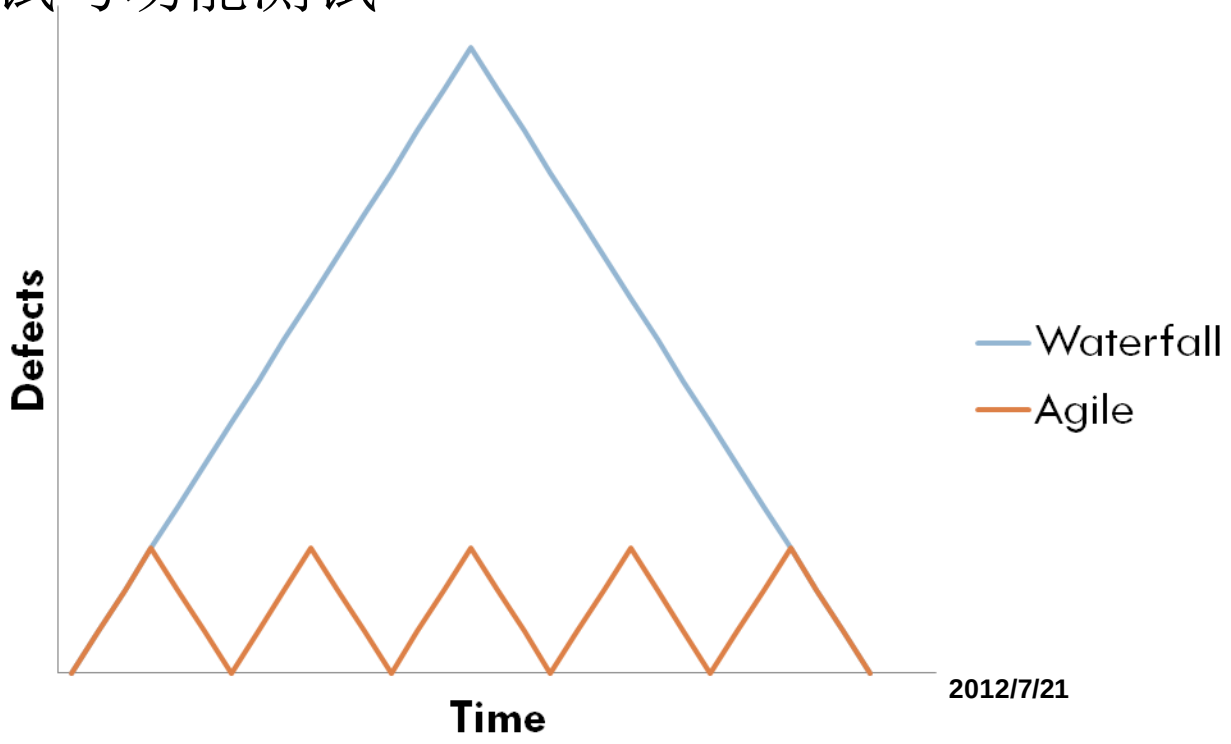
Write the code

Run the test And see it pass

Refactor as needed

测试驱动的开发

- 尽早测试 (Unit testing)
- 经常测试 (Continuous regression testing)
- 自动测试 (Testing a part of build process)
- 设计可测试的软件
- 单元测试与功能测试



什么是重构

Measures

- 重构是在不对代码的外部行为进行改动的情况下，对代码内部的结构进行优化。
- 重构的实质是对完成代码的设计进行改进。
- 重构必须和测试驱动的设计和开发伴随进行。
- 不要把重构变成不断的盲目精简代码。
- XP提倡毫不留情的重构(Refactor mercilessly)。
- 任何人可以重构任何代码，前提是重构后的代码一定要通过100%测试单元测试后才能被Check-in。
- 可以根据需要，将一个迭代的全部目标定为重构。
- 不要太在意什么是最简单的设计 —— 愿意在最后重构，比知道如何做简单的设计重要得多。

为什么要重构?

Measures

- 改进软件的设计。
  - 不断的对代码修改使代码失去了它最初的清晰结构，偏离需求或设计。
  - 重构则帮助重新组织代码，重新清晰的体现结构和进一步改进设计。
- 提高代码质量，可维护性。
  - 程序代码也是文档。而代码首先是写给人看的，然后才是给计算机看的。
- 重构帮助尽早的发现错误（Defects）
  - 重构是一个code review和反馈的过程。在另一个时段重新审视自己或别人代码，可以更容易的发现问题和加深对代码的理解。
- 重构可以提高提高开发速度
  - 好的设计和代码质量是提高开发速度的关键。在一个有缺陷的设计和混乱代码基础上的开发，即使表面上进度较快，但本质是延后了对设计缺陷的发现和对错误的修改，也就是延后了开发风险，最终要在开发的后期付出更多的时间和代价。

什么时候适合做重构?

Measures

- 在开始增加一个新的功能之前

  为了增加一个新的功能，程序员需要首先读懂现有的代码。
- 在修复一个错误的时候

  为了修复一个Bug,程序员需要读懂现有的代码。
- 在做Code Review的时候

什么时候不适合做重构?

Measures

- 代码太混乱，设计完全错误
  - 与其重构，不如重新开始。
  - 如果现有系统的20~25%需要修改，那么从头构件新产品更便宜、更容易。
- 明天是DeadLine

  永远不要做Last-Minute-Change。推迟重构，但不可以忽略，即使进入Production的代码都正确的运行。
- 重构的工作量显著的影响估算

  一个任务的估算是 3 天，如果为了重构，需要更多的时间（ 2 天或更多）。推迟重构,同步可以忽略。可以把这个重构作为一个新的任务，或者安排在重构的迭代中完成。

重构什么？

Measures

- Duplicated Code 重复的代码
- Long Method 过长方法
- Large Class 过大类
- Long Parameter List 过长参数列表
- Divergent Change 发散式变化
- Shotgun Surgery 霰弹式修改
- Feature Envy 依恋情结
- Data Clumps 数据泥团
- Primitive Obsession 基本类型偏执
- Switch Statements Switch语句
- Parallel Inheritance Hierarchies 平行继承层次
- Lazy Class 多余的类
- Speculative Generality 不确定的一般性
- Temporary Field 临时字段
- Message Chains 消息链
- Middle Man 二传手
- Inappropriate Intimacy 过度亲密
- Alternative Classes with Different Interfaces 异曲同工的类
- Incomplete Library Class 不完整的库类
- Data Class 数据类
- Refused bequest被拒绝的馈赠
- Comments 过多的注释

编程规范

Measures

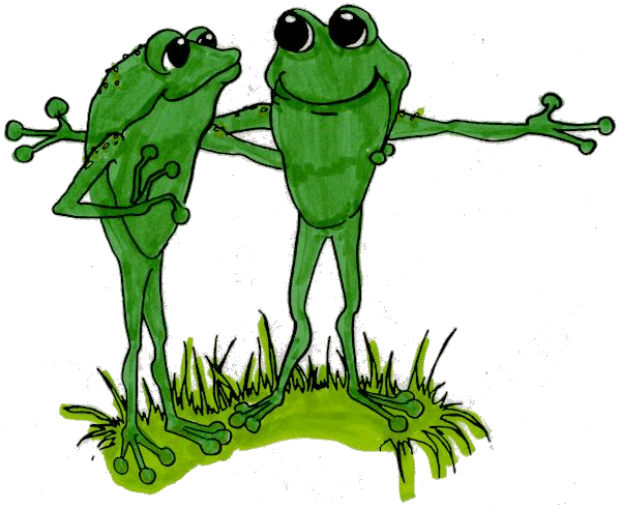
- 规定了程序的风格，包括注释如何写，变量命名的规范，代码的格式等等。
- Teamwork 的前提之一，其它众多惯例和规则(如Pair Programming， Collective Code Ownership等)的前提之一。

集体拥有代码

Measures

- “我们”的代码，而不是“我”的代码。
- 任何人可以改动任何一段代码，但改动后的代码必须通过所有相关的测试。
- 简单设计，编程规范和结队编程，使阅读和修改组内其他人的代码变得实际可行。

•程序员要负责和客户沟通需求,制定迭代计划,每个人都是系统的设计者,程序就是设计,每个人都能很好的理解别人的代码,对个人的要求很高



结对编程

- 两个程序员使用同一台电脑共同开发。XP的必须组成部分，XP中最有争议的规则之一。
- 配对是动态的，每天更换搭档。

- 老板为什么同意
- 程序员如何习惯
- 个性不同的人如何合作
- 不使用键盘的人如何进入状态

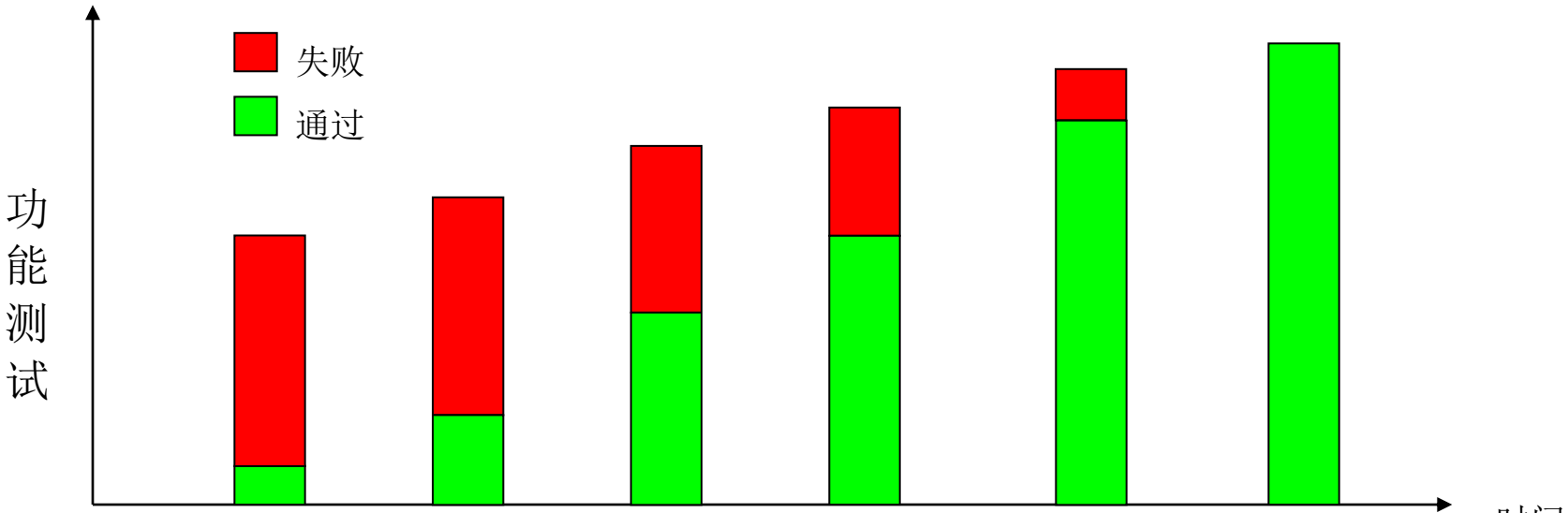
结对编程

Measures

- 结对编程可以提高代码质量
- 结对编程可以让团队的精力更加集中
- 很多抵制结对编程的人是没有尝试过结对编程的人，一旦尝试了，你会发现，你很快就会喜欢上它
- 要经常更换结对
- 结对编程可以增进团队间的知识传播
- 可以把代码评审作为结对编程的替代实践
- 鼓励大家尝试结对编程而不是强制

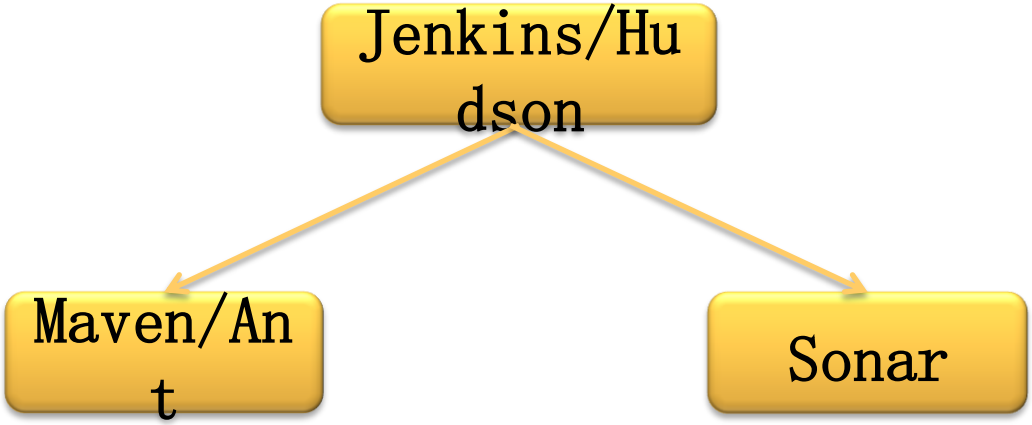
持续集成

- 测试先行是持续集成的一个重要前提。
- 持续集成指不断地把完成的功能模块整合在一起。目的在于不断获得客户反馈以及尽早发现BUG。
- 随时整合，越频繁越好；集成及测试过程的自动化程度越高越好。
- 每次只有一个结对在整合，而且必须运行功能测试。



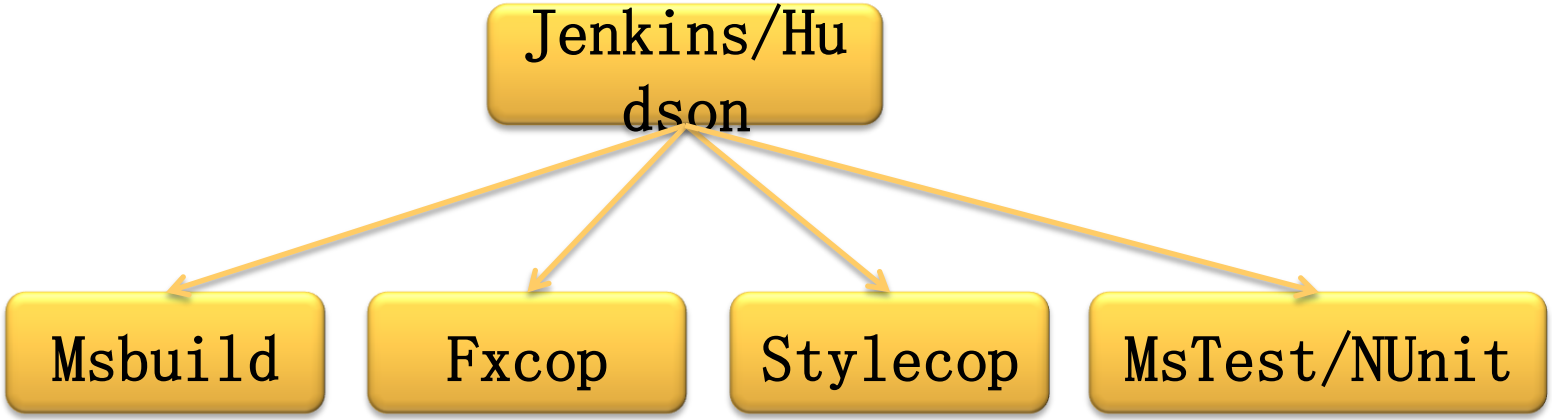
Java下的持续集成平台

Measures



Jenkins/Hudson是目前比较流行的CI引擎，特点是简单易用,扩展性强。
构建工具可以用Ant与Maven。
Sonar是一个很强大的QA工具，本身集成了Junit、覆盖率、代码规范和静态结构分析工具。

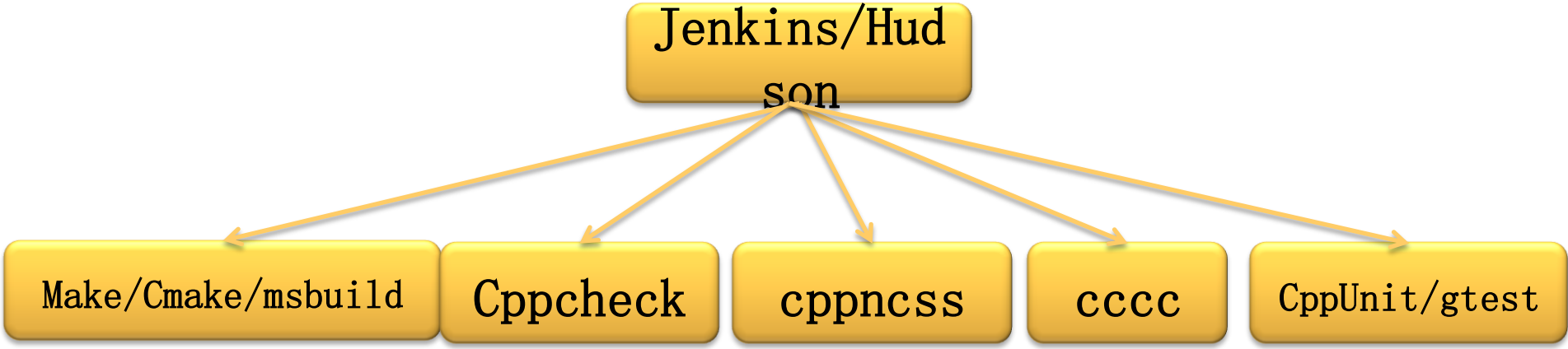
.Net下的持续集成平台



Fxcop是代码逻辑检查工具，
Stylecop是编码规范检查工具。
单元测试可以用MsTest或者Nunit.

C++下的持续集成平台

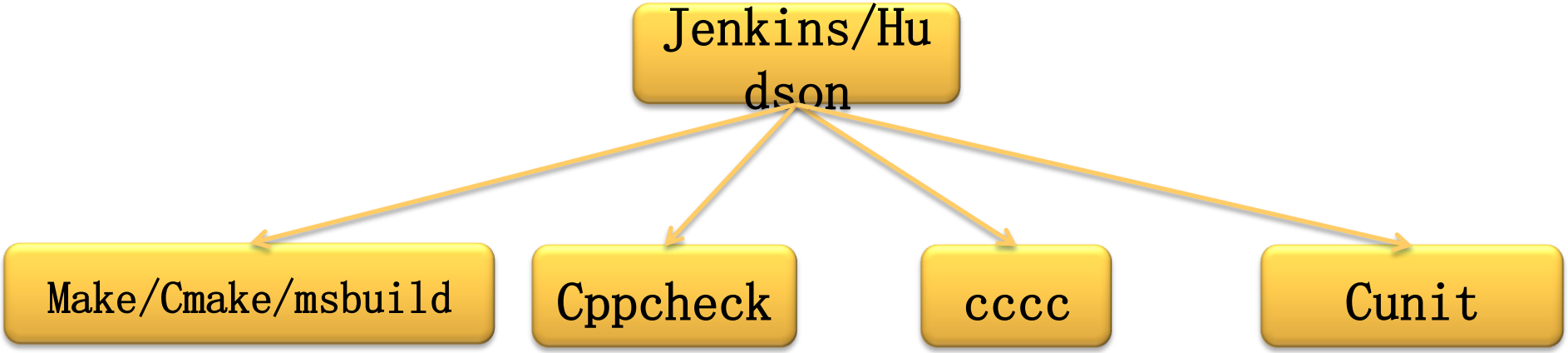
Measures



如果是Linux平台下的C++，主流的构建工具是make。
如果是windows平台的可以使用msbuild。
cppcheck是静态结构分析工具，cppncss是代码圈复杂度和代码行统计工具，
单元测试可以用CppUnit或者Google test(gtest).

C下的持续集成平台

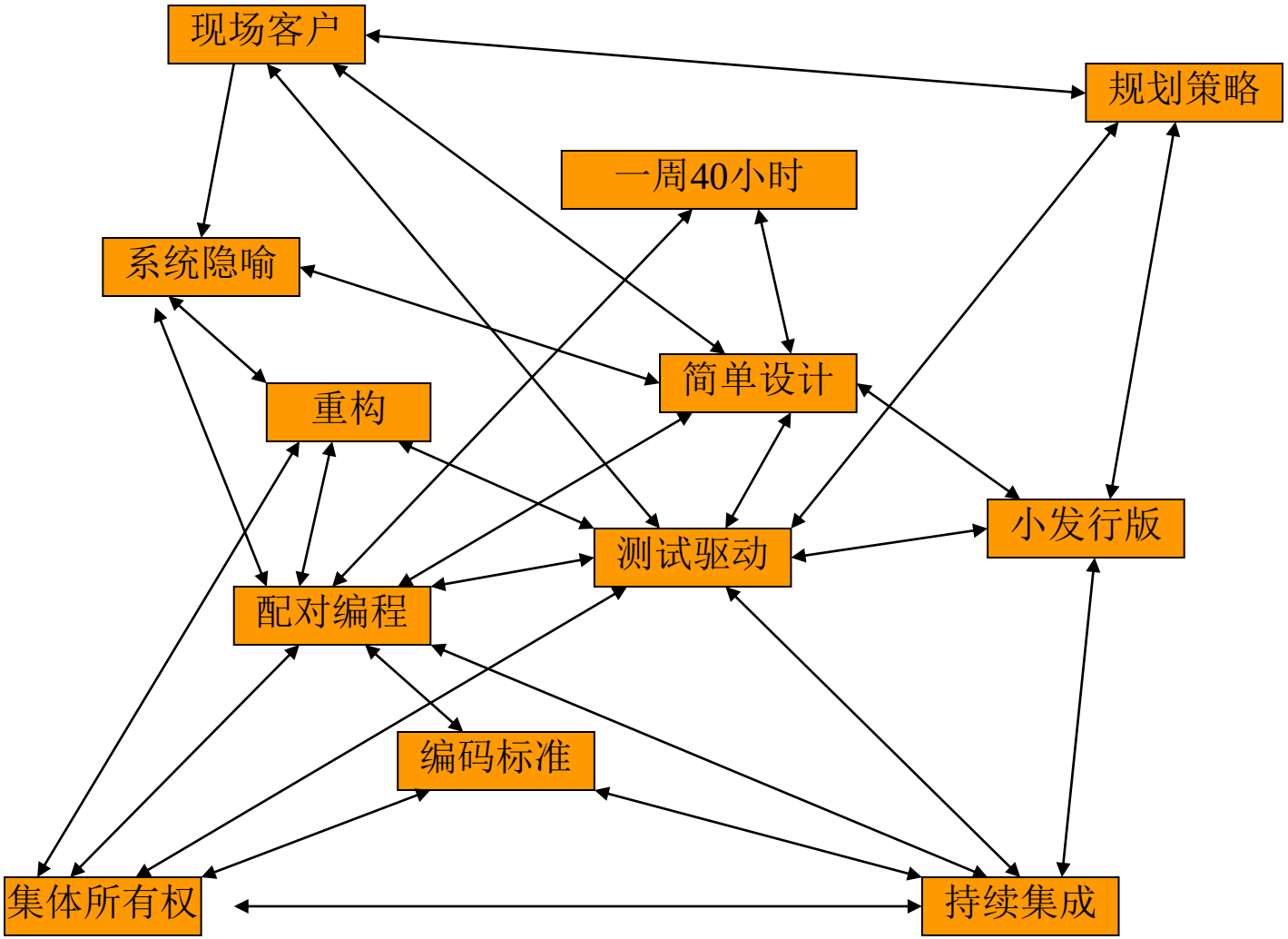
Measures



如果是Linux平台下的C，主流的构建工具是make。
如果是windows平台的可以使用msbuild。
cppcheck是静态结构分析工具，cccc是代码圈复杂度和代码行统计工具，
单元测试可以用CUnit

实践之间的互相支持

Measures



从CMM角度看XP

Measures

- XP部分满足或大部分满足了CMM 2-3 级KPA 的要求，而基本上没有涉及CMM 4-5 级的KPA
- XP 侧重于具体的过程和开发技术，而CMM 更关注组织和管理上的问题
- XP 缺少的一个重要内容是“institutionalization”

——Mark Paulk, SEI

对XP的置疑

Measures

- 技术前提：对变化成本曲线的置疑
- 技艺前提：对于有经验的人是简单的代码，对没有经验的人是复杂的
- 社会结构前提 ：程序员可以能否对自己的开发过程承担责任

不适用于XP的场合

Measures

- 不能接受XP 文化的组织
- 中大型（超过10 个人）的团队
- 重构会导致大量开销的应用
- 需要很长的编译或测试周期的系统
- 不太容易测试的应用
- 人员异地分布的物理环境

– –Beck

XP项目失败的常见错误

Measures

- 没有现场客户
- 将相互管理的实践截取其中一部分，断章取义
- XP只是迭代开发+最小文档+单元测试
- 没有首先编写单元测试
- 客户不做决策
- 没有属于客户自己的测试
- 最小限度地重构
- 和一个搭档接队时间过长，应该2天或更少的时间变换一次
- 没有集成在一起的QA
- 只有年轻的结对程序员
- 1位搭档进度太快
- 没有专门的验收测试者
- 编写验收测试的客户并不是这些测试执行的审核者
- 迭代时间太长

关于XP的参考资料

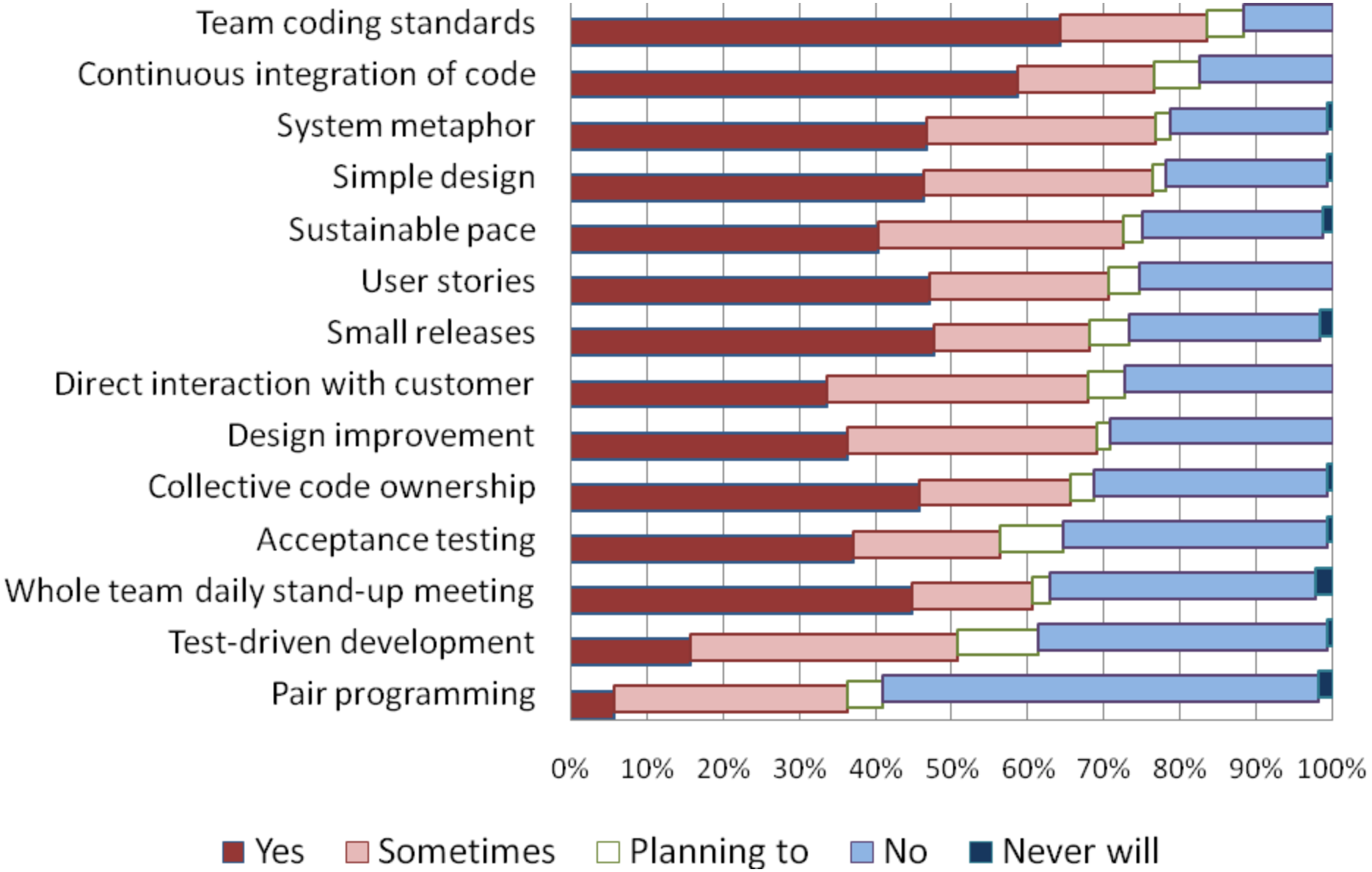
- 解析极限编程---拥抱变化
  - -Kent Beck著 唐东铭译 人民邮电出版社
- 规划极限编程
  - -Kent Beck,Martin Flower著 曹济译 人民邮电出版社
- 极限编程实践
  - -James Nerkirk,Robert C.Martin 著 王钧译 人民邮电出版社
- 极限编程研究
  - -Giancarlo Succi,Michele Marchesi 编 张辉译 人民邮电出版社

Measures

敏捷方法小结

Agile实践的调查

Measures



软件是新产品开发

Measures

预见式制造（重复生产）	新产品开发（个性定制）
在第1次完成规格说明后，就进行构建	不可能在前期创建一成不变的、详细的规格说明
在开始阶段就能估算出具有参考价值的工作量和成本	在开始阶段很难进行估算。随着经验数据的出现，计划与估算的可能性才会相应增加
有可能识别、定义、调度和安排所有的细节活动	在开始阶段，识别、定义、调度和安排所有的细节活动是不可能的。通过构建反馈周期，推动自适应步骤
一般情况下是不去适应没有预定义的变动，并且改动率也比较低。	一般情况下是主动适应没有预定义的变动，并且改动率也比较高

敏捷方法和计划驱动方法各自的擅长领域-1

Measures

特征	敏捷方法	计划驱动方法
应用		
主要目标	快速提供价值, 响应变更	可预见性;稳定性;高可靠性
规模	较小的团队和项目	较大的项目和团队
环境	难控制;高度变更	稳定;少变更
管理		
客户关系	专职的现场客户; 关注排定优先级的增量开发	根据需要和客户进行交互; 关注合同规定
计划和控制	内部的计划;定性控制	文档化的计划;定量控制
沟通	隐式的人与人之间的知识	显示的文档化的知识

敏捷方法和计划驱动方法各自的擅长领域-2

Measures

特征	敏捷方法	计划驱动的方法
技术		
需求	排定优先级的非正式的用户素材和测试用例;会经受不可遇见的变更	规范化的项目、性能、接口、质量 可预见的演化需求
开发	简单设计;短增量;认为重构代价低廉	大量的设计；较长的增量；认为重构代价昂贵
测试	测试用例可以执行并用来定义需求	文档化的测试计划和规程
人员		
客户关系	专职的、工作在一起的CRACK型客户	CRACK型客户，并不总是工作在一起
开发者	要求参与者熟悉方法,能对方法进行裁剪以适应变化	需要熟悉该方法的人,也需要可以通过培训来满足方法的要求的人员
文化	通过更多的自由度来达到舒适和权力 （靠混沌得以繁荣）	通过政策和规程框架来达到舒适和权力 靠秩序得以繁荣）

Measures

Q & A ?

Measures

谢　　谢！