



Scrum



# Scrum框架的全球企业应用现状

- 背景：全面 VUCA（易变、不确定、复杂、模糊）时代
- 行业渗透：华为、腾讯、Google、Amazon、SAP等科技/制造/金融领域头部企业
- 例子
  - 美国银行用Scrum重组信用卡系统开发，原本需要半年的审批流程压缩到3周。
  - 西门子医疗在CT机研发中，用Scrum协调德国工程师和上海工厂，通过两周一次的"冲刺评审会"，把设计到量产的周期缩短40%。
  - (几乎所有互联网企业) ...

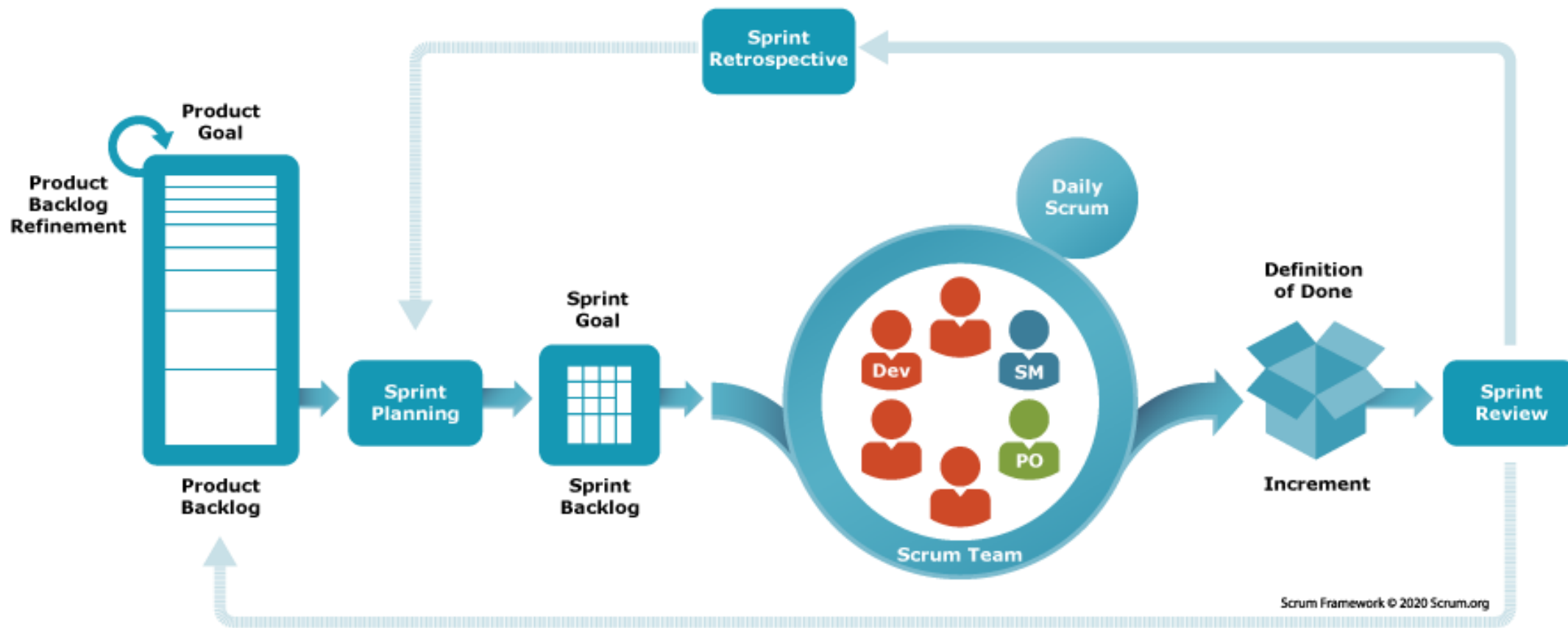
# Scrum方法论演进历程

- 概念起源：1986年竹内弘高《新型新产品开发策略》首次提出"Scrum"术语
  - 1986年，“The New New product Development Game” 竹内弘高和野中郁次郎阐述了一种新的整体性的方法，该方法能够提高商业新产品开发的速度和灵活性：他们将这种新的整体性方法与橄榄球相比较，前者各阶段相互重叠，并且由一个跨职能团队在不同的阶段完成整个过程，而团队“作为一个整体前进，把球传来传去”。
  - Scrum在英语是橄榄球运动中争球的意思。
- 框架形成：1995年Jeff Sutherland团队在Easel公司完成首个完整实践,在奥斯汀举办的OOPSLA 95上，Sutherland和Schwaber联合发表了论文首次提出了Scrum概念。

# Scrum定义

- Scrum 不是构建产品的一种过程或一项技术,而是一个框架,在这个框架里可以应用各种流程和技术。Scrum 能使产品管理和开发实践的相对功效显现出来,以便随时改进。
- “Scrum是一个基于团队进行复杂系统和产品开发的框架 ” -Scrum联盟
- Scrum 框架包括 Scrum 团队及其相关的角色、事件、工件和规则。框架中的每个模块都有一个特定的目的,对 Scrum 的成功和使用都至关重要。
- “据其定义, Scrum事实上并未谈及软件。 Scrum所涉乃是非软件项目亦可使用的工作管理和团队动力学。 ” ----- Jeff McKenna

# Scrum以整体的形式存在，才能作为其他技术、方法论和实践的容器良好运作



# Scrum Guide 2020.11 by Ken Schwaber & Jeff Sutherland

- Scrum 是一个轻量的框架，它通过提供针对复杂问题的自适应解决方案来帮助人们、团队和组织创造价值。
- 简而言之，Scrum 需要 Scrum Master 营造一个环境，从而：
  - 1. 一名 Product Owner 将解决复杂问题所需的工作整理成一份 Product Backlog。
  - 2. Scrum Team 在一个 Sprint 期间将选择的工作转化为价值的 Increment。
  - 3. Scrum Team 和利益攸关者检视结果并为下一个 Sprint 进行调整。
  - 4. 重复



# Scrum 33355 (或 3355)

- 三大支柱
  - 透明、检视、适应
- 五个价值
  - 承诺、专注、开放、尊重和勇气
- 三个角色
  - 开发人员、PO、Scrum Master
- 三个工件
  - 产品 Backlog、Sprint Backlog、增量
- 五个事件
  - Sprint
  - Sprint 计划会议
  - 每日 Scrum 会议
  - Sprint 评审会议
  - Sprint 回顾会议

# Scrum 理论

- Scrum 基于经验主义和精益思维。经验主义主张知识源自实际经验以及根据当前观察到的事物作出的判断所获得。精益思维减少浪费，专注于根本（第一性思维）。
- Scrum 将 4 个正式事件组合在一起以在一个容器型事件 Sprint 中进行检视和适应。这些事件之所以起作用，是因为它们实现了基于经验主义的 Scrum 的三个支柱：透明、检视和适应。



# 经验主义

- 经验主义（Empiricism）是哲学认识论中的一个核心流派，主张人类知识的根本来源是实际经验，而非先天的观念或纯粹的理性推理。
- 经验主义强调通过观察具体现象，再通过归纳法（从个别案例推导普遍规律）形成知识。
- 经验主义深刻影响了现代科学方法，强调证据、实验和可验证性。例如：
  - 医学通过临床试验验证药物效果，
  - 物理学通过观测数据建立理论模型。



## 三个支柱

- 透明**：浮现（emergent）的过程和工作必须对执行工作的人员和接受工作的人员都是可见的。在 Scrum 中，重要的决策是基于其 3 个正式工件的感知状态。透明度较低的工件可能导致做出降低价值并增加风险的决策。透明使检视成为可能。没有透明的检视会产生误导和浪费。
- 检视**：Scrum 工件和实现商定目标的进展必须经常地和勤勉地检视，以便发现潜在的不良的差异或问题。为了帮助检视，Scrum 以 5 个事件的形式提供了稳定的节奏。检视使适应成为可能。没有适应的检视是毫无意义的。Scrum 事件旨在激发改变。
- 调整**：如果过程的任何方面超出可接受的范围或所得的产品不可接受，就必须对当下的过程或过程处理的内容加以调整。调整工作必须尽快执行以最小化进一步的偏差。



# Scrum 价值观





## Scrum Team

- Scrum Team 由一名 Scrum Master, 一名 Product Owner 和 Developers 组成。在 Scrum Team 中, 没有子团队或层次结构。
- Scrum Team 规模足够小以保持灵活, 同时足够大以便可以在一个 Sprint 中完成重要的工作, 通常只有 10 人或更少。总的来说, 我们发现较小的团队沟通更好, 效率更高。
- 如果 Scrum Team 变得太大, 则应考虑将他们重组为多个具有凝聚力的 Scrum Team, 每个团队都专注于同一产品。

# 跨职能的Scrum Team

- Scrum Team 是跨职能的（cross-functional），这意味着团队成员具有在每个 Sprint 中创造价值而所需的全部技能。
  - 完整技能闭环：具备端到端交付价值所需全部能力
  - 这意味着它不依赖于跟其他团队的工作交接，也不用等待其他团队的工作。
  - 反例：职能团队，U I团队、业务逻辑团队、测试团队和数据团队等等。团队等待，责任扯皮，交流沟通成本剧增。
- 跨职能不是万能药：保持核心技能深度的同时，建立足够的协作带宽。  
(不是要求每个人都成为全栈工程师)

# 自管理的 Scrum Team

- 这意味着他们在团队内部决定谁做什么、何时做以及如何做。
- Hackman 权力矩阵

整体方向的设定	管理者职责			
团队及其组织环境的规划				
工作过程和进度的监控和管理	团队自己的职责			
团队任务的执行				
	管理者 领导型团队	自管理型 团队	自规划型 团队	自治理型 团队

# 微观管理及其危害

- 微观管理（Micromanagement）一般是指：在对员工的工作管理中，管理者过度关注和控制工作细节的管理风格和管理行为。
- 客观地说，关注细节并非坏事。但是一旦过于关注和控制细节，就会带来种种问题：
  - 第一，会导致团队成员失去主观能动性。由于完全沦为执行者，导致员工只能按照管理者的思路走，从而失去了自主性和创造力。
  - 第二，会导致工作中出现决策等待和低效。当实际工作场景与管理者最初设想的不一致时，员工无法继续按照管理者最初的思路工作，就只能把问题反馈给管理者，等待管理者作出决策。
  - 第三，会影响到团队的积极性和士气。由于团队成员经常处于要向管理者汇报的压力中，经常处于被管理者纠偏的状态，很容易让团队成员感到不自信和缺乏成就感。
- 例子：设计师调整像素需要邮件审批，程序员每半小时汇报代码行数，这就像让交响乐指挥去纠正小提琴手的每个运弓动作。微观管理本质上是管理精度的失控，把导航仪变成了方向盘控制器。

# “自我管理” 和 “自组织”

## ●自我管理 (Self-Management)

- 指个体或团队在明确目标下，通过自我规划、自我监督、自我调整等方式，自主完成任务的机制。例如：个人时间管理、团队自主决策项目分工。
- 核心：对“行为”的自主控制，强调个体或小单元的主动性和责任感。

## ●自组织 (Self-Organization)

- 指复杂系统（如生态系统、社会组织）在没有外部指令干预的情况下，通过内部成员或元素的互动，自发形成有序结构和功能的过程。例如：蚂蚁群体的协作、开源社区的演化;维基百科内容由用户自发编辑完善，形成知识网络。
- 核心：系统层面的“结构或秩序”自发涌现，强调整体动态适应性。

## ●均强调“去中心化”和“自主性”，反对僵化的外部控制，注重灵活性和适应性。






# Developers

- Developers 是 Scrum Team 中致力于创建每个 Sprint 可用 Increment 的任何方面的人员。
- Developers 所需的特定技能通常很广泛，并且会随着工作领域的不同而变化。但是， Developers 始终要负责：
  - 为 Sprint 创建计划，即 Sprint Backlog；
  - 通过遵循 Definition of Done 来注入质量；
  - 每天根据 Sprint Goal 调整计划； 和，
  - 作为专业人士对彼此负责。



## Product Owner

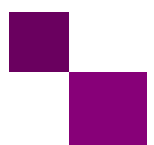
- Product Owner 负责将 Scrum Team 的工作所产生的产品价值最大化。 如何做到这一点可能在组织、Scrum Team 和个体之间存在很大差异。
- Product Owner 还负责对 Product Backlog 进行有效管理，包括：
  - 开发并明确地沟通 Product Goal ；
  - 创建并清晰地沟通 Product Backlog 条目（items）；
  - 对 Product Backlog 条目进行排序； 和，
  - 确保 Product Backlog 是透明的、可见的和可理解的。

- 
- Product Owner 可以自己做上述工作，或者也可以将职责委托他人。  
然而无论如何，Product Owner 是负最终责任的人。
  - 为保证 Product Owner 取得成功，整个组织必须尊重他们的决定。这些决定在 Product Backlog 的内容和顺序中可见，并在 Sprint Review 时透过可检视的 Increment 予以体现。
  - Product Owner 是一个人，而不是一个委员会。在 Product Backlog 中，Product Owner 可以代表许多利益攸关者的期望要求。那些想要改变 Product Backlog 的人可以尝试去说服 Product Owner 来做到这一点。



# Scrum Master

- Scrum Master 负责按照 Scrum 指南的游戏规则来建立 Scrum。他们通过帮助 Scrum Team 和组织内的每个人理解 Scrum 理论和实践来做到这一点。
- Scrum Master 对 Scrum Team 的效能负责。他们通过让 Scrum Team 在 Scrum 框架内改进其实践来做到这一点。
- Scrum Masters 是真正的领导者（服务者，仆人？），服务于 Scrum Team 和作为更大范围的组织。

- 
- Scrum Master 以多种方式服务于 Scrum Team , 包括:
    - 作为教练在自管理和跨职能方面辅导 Scrum Team 成员;
    - 帮助 Scrum Team 专注于创建符合 Definition of Done 的高价值 Increment;
    - 促使移除 Scrum Team 工作进展中的障碍; 和,
    - 确保所有 Scrum 事件都发生并且是积极的、富有成效的, 并且在时间盒 (timebox) 内完成。

- 
- Scrum Master 以多种方式服务于 Product Owner, 包括:
    - 帮助找到有效定义 Product Goal 和管理 Product Backlog 的技巧;
    - 帮助 Scrum Team 理解为何需要清晰且简明的 Product Backlog 条目;
    - 帮助建立针对复杂环境的基于经验主义的产品规划 (empirical product planning) ; 和,
    - 当需要或被要求时, 引导利益攸关者协作。



## ●Scrum Master 以多种方式服务于组织，包括：


- 带领、培训和作为教练辅导组织采纳 Scrum；
- 在组织范围内规划并建议 Scrum 的实施；
- 帮助员工和利益攸关者理解并实施针对复杂工作的经验主义方法（empirical approach）；
- 消除利益攸关者和 Scrum Teams 之间的隔阂。

# 猪与鸡的比喻



- 一天，一头猪和一只鸡在路上散步，鸡看了一下猪说：“嗨，我们合伙开一家餐馆怎么样？”
- 猪回头看了一下鸡说：“好主意，那你准备给餐馆卖什么呢？”
- 鸡想了想说：“餐馆卖火腿和鸡蛋怎么样？”
- 猪说：“不开了，我全身投入(火腿是一次性资源)，而你(鸡蛋是可再生的)只是参与而已”



- 
- 猪角色被认为是团队中的核心成员,在一个团队中产品的负责人和Scrum主管和开发团队就是“猪”角色。鸡角色不是Scrum的一部分,但必须要考虑他们,用户,客户或提供商,经理等扮演着“鸡”角色!
  - 把有兴趣关心,并无利益或价值牵扯的人,排除在项目决策团队以外!

# 用户故事

- 在20世纪90年代末，Kent在开发软件的过程中发现，其中最大的问题莫过于使用文档来精确描述我们想要的东西，即需求。
- 同样的一份文档，阅读的人不同，各自得到的信息也不一样。这种缺乏共识的情况，称为“低质量的需求”
- 停止写出完美文档的“执念”。
- 用户故事之所以得名，并不是要人们如何写出更好的 用户故事，而是如何在协作中更好地使用它。

## 需求—用户故事模板（1）

- 用户故事是产品列表的基础构件。
- 用户故事模板（不是唯一方法）：

作为<某类用户>  
我想<做某事>  
从而<创造出某些价值>

## 需求—用户故事模板 (2)

- 作为<某类用户>
  - 告诉我们想要这个功能的是谁。
- 我想<做某事>
  - 告诉我们预期功能是什么。
- 从而<创造出某些价值>
  - 告诉我们为什么用户想要这个功能。

作为邓丽君粉丝俱乐部成员，  
我想在公开售票之前通过电话预订演唱会门票，  
从而可以拿到好位置还能感觉很特别。

作为管理员，  
我想要这个网站使用情况报告，  
从而做出有根据的预测并优先完成相关功能。



## 故事模板(3)

- 用户角色(who):
- 功能(what):
- 为什么(why):

# Ron Jeffries的3C原则

- 卡片 (Card) (placeholder, 占位符)

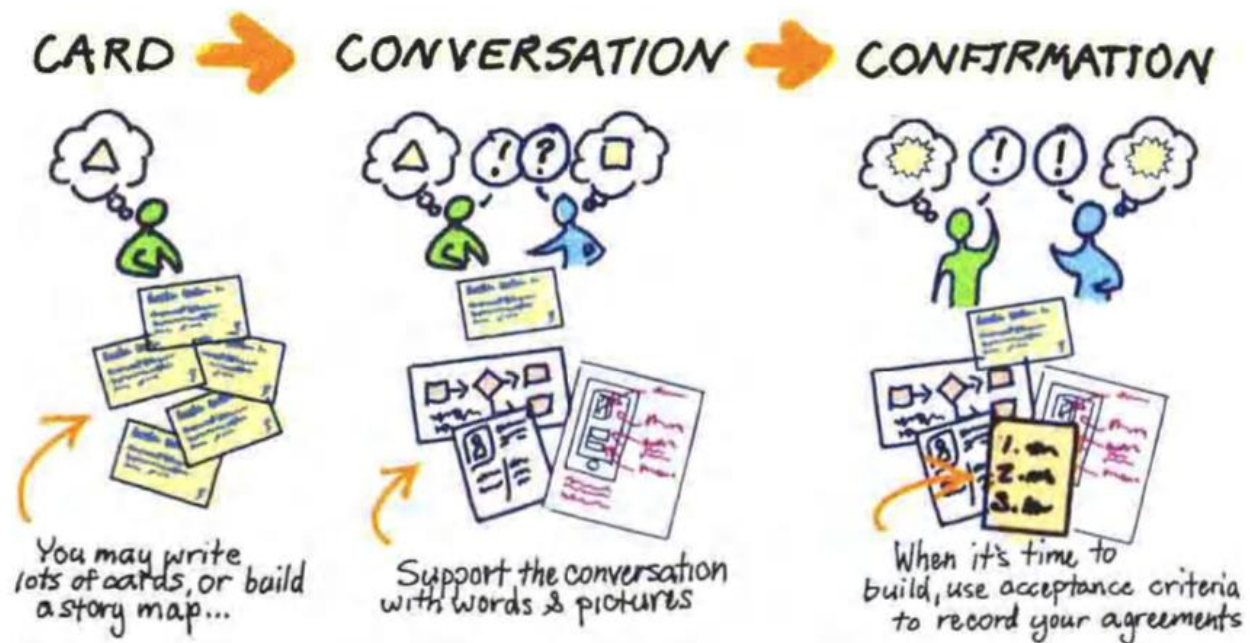
- 在一堆卡片上写下你期望的软件特性

- 交谈 (Conversation)

- 聚在一起对要开发的软件进行深入讨论

- 确认 (Confirmation)

- 对完工条件进行确认



# 用户故事是占位符

- 用户故事不是完整的需求或说明书，它们是占位符。
- 它们的信息量足以提醒团队有东西要完成，但我们刻意地不过多探讨细节.....直到必需之时。
- 需要阐述用户故事细节时，使用召集相关团队成员参与交谈的形式。交谈的目标在于，对故事内容以及所需完成的工作达成 共识。相对于依赖书面文档，使用实时对话的方式能更高效地达成此目标。它有更多的信息流动。

## 用户故事接收标准

- 交谈中，如果某一刻大家觉得对用户故事已有共识，那就该写验收标准了。找出一列测试，直到所有参与交谈的人都认同测试通过意味着故事按预期实现即可，再用简单易懂的话记录下来。
- 接收标准回答如下的问题：“我们如何得知它何时已完工？”
- 理想情况下，团队应该可以根据接收标准写出自动化测试，甚至是在功能实现之前（TDD）。位于产品列表下方的故事可能不会很快被实现，接收标准可以降低精度。



## “Definition of Done” (DoD)

- “Definition of Done” (DoD, 完成的定义) 是敏捷软件开发中的一个关键概念，用于描述一个用户故事、任务或功能何时可以被认为真正“完成”。它是团队对完成工作的标准化定义，确保开发过程中每个增量都符合质量要求并准备好交付。
  - 1.提高透明度：DoD 明确了完成的标准，减少了团队之间的沟通误解。
  - 2.保证质量：通过对代码质量、测试和文档的明确要求，确保产出的增量软件可用且可靠。
  - 3.支持验收流程：DoD 是验收用户故事或功能的依据。如果工作未达到 DoD，任务不能被标记为完成。
  - 4.防止技术债：通过明确完成标准，避免开发过程中留下未解决的问题。

# “Definition of Done” (DoD) 例子

- 假设一个团队正在开发一个电子商务应用的“用户注册功能”，其 DoD 可能如下：
- 1.开发完成：
  - 用户可以通过表单输入基本信息（用户名、密码、邮箱）。
  - 提交表单后，信息存储到数据库。
- 2.测试通过：
  - 表单的必填项验证功能正常。
  - 注册成功后，用户会收到确认邮件（已在开发环境中验证）。
  - 单元测试覆盖率达到 95%。
- 3.集成完成：
  - 功能已合并到主分支，并通过持续集成管道中的所有检查。
- 4.文档补充：
  - 用户手册中新增“如何注册”的部分。
  - 开发文档中记录了接口的详细信息和数据库结构。
- 5.验收完成：
  - 产品负责人在测试环境中成功完成了用户注册操作。

# 行为驱动开发 (Behavior-Driven Development, BDD)

- 行为驱动开发 (BDD) 是一种基于敏捷的软件开发方法论，其核心思想是通过定义软件的行为来驱动开发过程。BDD 的重点是增强团队对需求的理解，并确保开发的软件满足业务目标。
- BDD 是从测试驱动开发 (TDD) 演化而来的，强调在开发开始之前，用自然语言描述软件应如何行为。它通过让技术人员、业务人员和测试人员围绕共同的需求语言进行协作，消除了沟通中的歧义。

# BDD 的实践流程1

- 1. 编写用户故事：
- 用户故事采用以下模板描述功能需求：
  - As a [user role],
  - I want [feature/goal],
  - So that [business value].
- 作为一名电子商务网站用户,
- 我希望能够将商品添加到购物车,
- 以便稍后结算。

## BDD 的实践流程2

### ●2. 定义验收标准 (Acceptance Criteria) :

- 验收标准是用户故事“完成”的具体条件，通常以场景的形式定义：

Scenario: [场景描述]

Given [初始状态]

When [执行的操作]

Then [预期的结果]

#### 场景 1: 成功添加商品

 复制

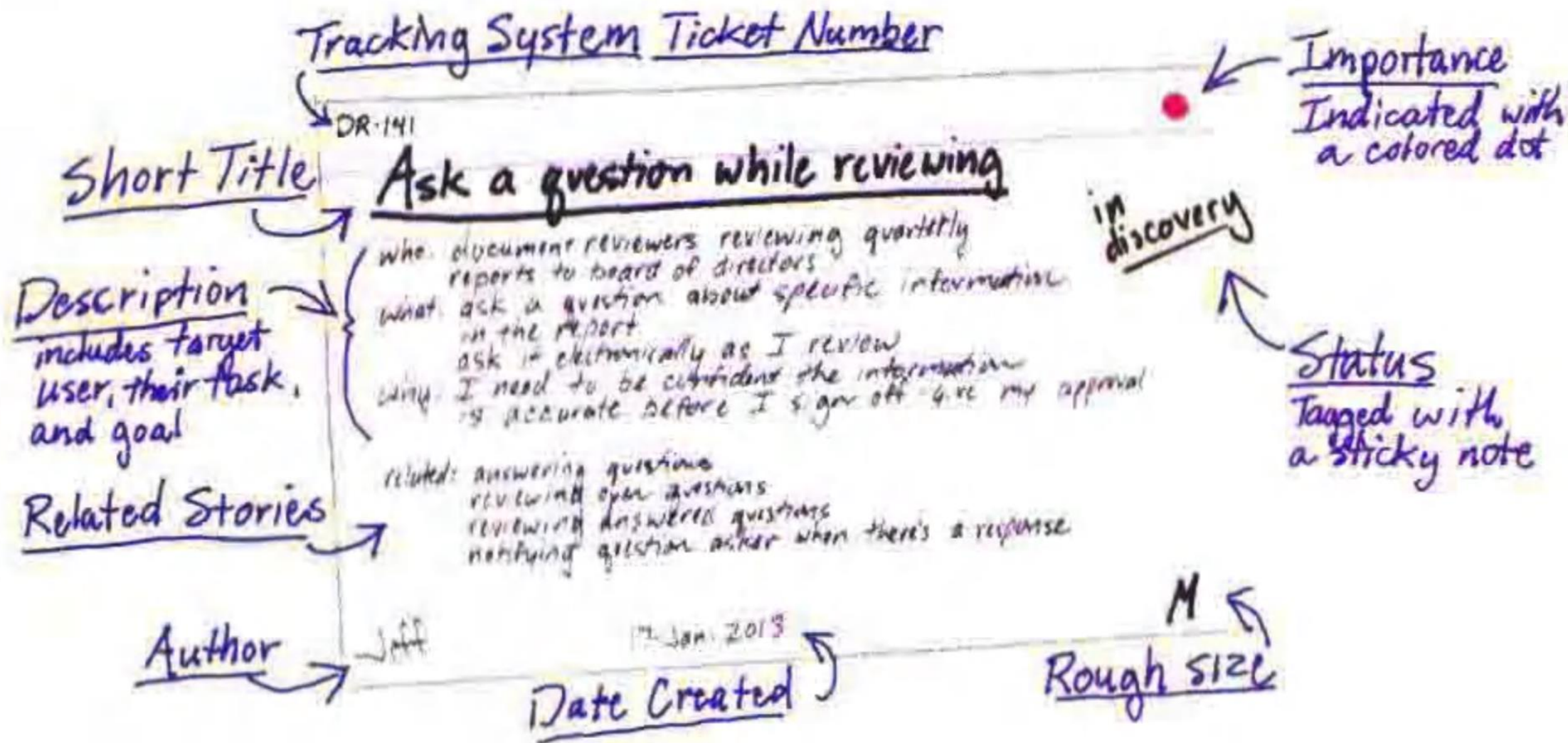
```
Scenario: User adds an item to the shopping cart
  Given the shopping cart is empty
  When the user adds a product to the cart
  Then the cart should contain 1 item
```

#### 场景 2: 商品已存在

 复制

```
Scenario: User adds an existing product to the cart
  Given the shopping cart contains 1 item
  When the user adds the same product again
  Then the cart should contain 2 items
```

# 复杂故事卡



# 用户故事 INVEST 原则(2004)

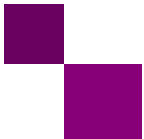
- 独立性 (Independent) — 要尽可能的让一个用户故事独立于其他的用户故事。
- 可协商性 (Negotiable) — 一个用户故事的内容要是可以协商的，用户故事不是合同。
- 有价值 (Valuable) — 每个故事必须对客户具有价值（无论是用户还是购买方）。
- 可以估算性 (Estimable) — 开发团队需要去估计一个用户故事以便确定优先级，工作量，安排计划。
- 短小 (Small) — 一个好的故事在工作量上要尽量短小，至少要确保的是在一个迭代或Sprint中能够完成。
- 可测试性 (Testable) — 一个用户故事要是可以测试的，以便于确认它是可以完成的。
- 每个用户故事都应像标准快递箱——独立封装 (Independent)、地址清晰 (Valuable)、体积适中 (Small)、运费明确 (Estimable)、可追踪 (Testable)、允许改派 (Negotiable)。这样的标准化包装，才能让需求运输既高效又可靠。



# 用户故事 SMART 原则

- S — Specific（具体）：用户故事需明确描述一个具体的功能或需求，避免模糊或笼统的描述。
- M — Measurable（可衡量）：用户故事的结果必须能用明确的指标或验收标准验证。
- A — Achievable（可实现）：用户故事需在当前迭代周期或合理的时间内完成，符合团队的技术能力和资源限制。
- R — Relevant（相关性）：用户故事需与当前项目目标或用户需求直接相关，避免无关功能。
- T — Time-bound（有时限）：用户故事需明确截止时间或优先级，避免无限延期。
- (Testable（可测试）：部分团队将最后一个 T 定义为“可测试”，强调需求可通过自动化测试验证。)





# 产品Backlog

- 是Scrum的核心，是按重要性排序的需求或故事（Story）的列表  
(客户语言描述的客户需求)

Backlog Item	优先级	用户故事	验收标准	估算工时 (Story Points)
用户注册功能	高	作为一名新用户，我希望能够通过注册表单创建账户，以便登录并使用平台服务。	- 用户可以输入用户名、邮箱、密码 - 必填项验证正确，无错误提示 - 提交后，账户信息存储在数据库 - 用户收到注册确认邮件	8
登录功能	高	作为一名用户，我希望能够通过使用用户名和密码登录账户，以便访问我的个人信息。	- 输入正确的用户名和密码后可以登录 - 输入错误时显示明确的错误消息 - 登录状态在会话中维持有效	5
商品浏览	中	作为一名用户，我希望能够通过查看商品列表，以便快速找到我需要的商品。	- 页面展示所有商品的基本信息（图片、名称、价格） - 商品列表支持分页加载	8
购物车功能	中	作为一名用户，我希望能够将商品添加到购物车，以便稍后进行结算。	- 用户可以将任意商品添加到购物车 - 购物车页面显示商品数量、总价 - 用户可以在购物车中移除商品	13
支付功能	高	作为一名用户，我希望能够通过支付页面完成订单付款，以便购买商品。	- 支持多种支付方式（如信用卡、PayPal） - 支付完成后生成订单并发送确认邮件 - 支付失败时提示用户重试	20
搜索功能	中	作为一名用户，我希望能够通过搜索栏快速找到特定的商品，以节省时间。	- 输入关键字后，展示相关商品列表 - 支持模糊搜索和关键字高亮	8
用户账户管理	低	作为一名用户，我希望能够通过修改我的个人信息（如用户名、密码、邮箱），以便保持账户信息的准确性。	- 用户可以修改个人信息并保存 - 修改后的信息实时更新在数据库	8
推荐系统	低	作为一名用户，我希望能够在商品页面看到个性化推荐，以便发现更多感兴趣的物品。	- 根据用户浏览历史推荐相关商品 - 页面加载推荐商品不影响主商品信息加载	13

## Product Backlog (动态需求池)

- Product Backlog 是一份涌现的和有序的清单，它列出了改进产品所需的内容。它是 Scrum Team 所承担工作的唯一来源。
- 能够被 Scrum Team 在一个 Sprint 中完成 (Done) 的 Product Backlog 条目被认为准备就绪，在Sprint Planning 事件中可供选择。它们通常在精化活动后获得这种透明度。
- Product Backlog 精化是将 Product Backlog 条目分解并进一步定义为更小更精确的行为。这是一项持续进行的活动，为Product Backlog 条目增添细节，例如描述、优先顺序和规模。这些属性通常随工作领域而变化。

# 什么是用户故事地图

- 用户故事地图是一门在需求拆分过程中保持全景图的技术。
- 敏捷软件开发中使用用户故事地图来发现、管理需求。
- 2008年提出。
  - J. Patton, User Story Mapping: Discover the Whole Story, Build the Right Product. Sebastopol, CA: O'Reilly Media, 2014.

# 一个例子（简单手机银行，开发）



# 故事地图解决的问题

- 1. **全局视角缺失**：传统的待办事项列表（Backlog）往往只关注单个功能，缺乏对整体产品的全局视角，导致团队难以理解产品的整体结构和用户体验流程。用户故事地图通过横向展示用户活动，纵向排列功能细节，提供了产品的全貌视图，帮助团队更好地理解用户的使用路径。
- 2. **需求优先级难以确定**：在复杂的项目中，众多需求可能让团队难以确定哪些功能应优先开发。用户故事地图通过将用户故事按照用户活动和任务进行组织，清晰地展示各功能的相对重要性，便于团队合理安排开发顺序。
- 3. **缺乏用户需求聚焦**：开发过程中，团队可能过于关注技术实现，忽视了用户的真实需求。用户故事地图强调从用户角度出发，确保开发的功能真正满足用户需求，提升用户满意度。
- 4. **难以理解功能之间的关系**：在大型项目中，不同功能之间的关系可能复杂，团队难以把握。用户故事地图通过结构化的方式展示功能之间的关联，帮助团队更好地理解和管理这些关系。
- 5. **发布计划不明确**：在敏捷开发中，确定每次发布的功能范围至关重要。用户故事地图通过清晰地展示各功能的优先级和依赖关系，帮助团队制定合理的发布计划，确保每次发布都能为用户提供有价值的功能。



# Sprint

- Sprint 是 Scrum 的核心，在这里创意 (idea) 转化为价值。
- 它们是固定时长的事件，为期一个月或更短，以保持一致性。前一个 Sprint 结束后，下一个新的 Sprint 紧接着立即开始。
- 实现 Product Goal 所需的所有工作，包括 Sprint Planning、Daily Scrum、Sprint Review 和 Sprint Retrospective，都发生在 Sprint 内。

# 一个Sprint有多长？

- 确定Sprint长度：Sprint持续多久才算合适？
  - 时间短：“敏捷”——短反馈周期=频繁交付=频繁客户反馈=错误方向  
持续时间短=学习改进速度快……
  - 时间长：更多时间作充分准备、解决问题、达成目标，不会被接二连三的会议压的不堪重负。
- 当前，Scrum 周期通常为2个星期。



## Sprint计划会议准备

- 所有重要的backlog条目都已经根据重要性被评过分,不同的重要程度对应不同的分数。分数只是用来根据重要性对backlog条目排序。
- 所有人都可以编写添加条目, 但只有Product Owner才能决定优先级。



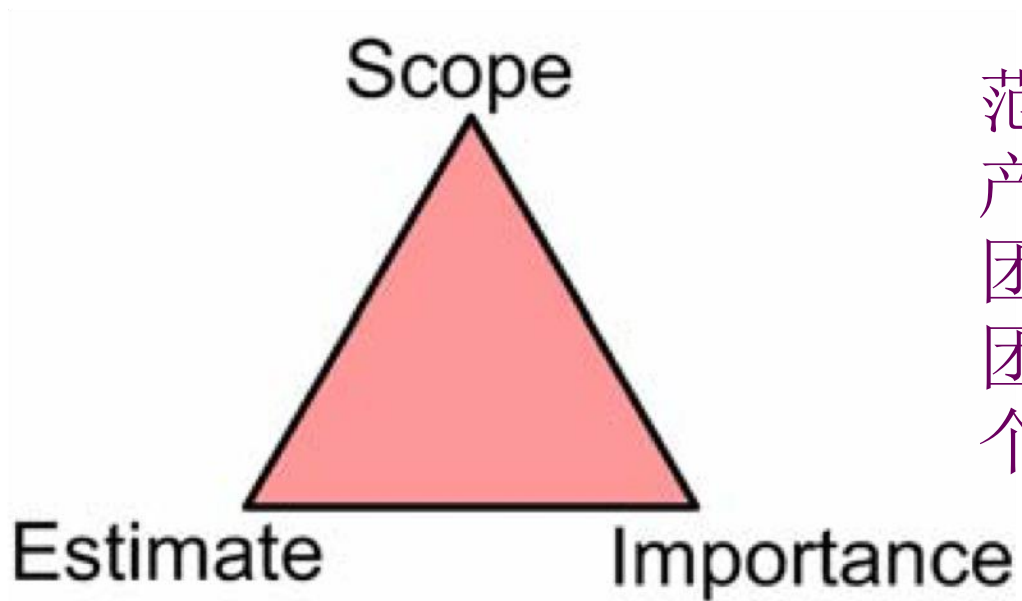
# Sprint计划会议目标

## ●以终为始

- sprint目标（尽可能简单的语言，团队成员认同）。
- 团队成员名单(以及他们的投入程度,如果不是100%的话)。
- Sprint backlog(即sprint中包括的故事列表)。
- 确定好sprint演示日期。
- 确定好时间地点,供举行每日scrum会议。

## Product Owner必须参加计划会议

- 每个故事都含有三个变量,它们两两之间都对彼此有着强烈依赖。  
(没有包含质量, 内部质量必须最好, 外部质量可以简陋一些。)



范围(scope)和重要性(importance)由产品负责人设置。估算 (estimate)由团队设置。在 **sprint** 计划会议上,经过团队和产品负责人面对面的对话,这三个变量会逐步得到调整优化。

# 计划会议议程的一个例子

- Sprint 计划会议:13:00 – 17:00 (每小时休息 10 分钟)
  - 13:00 – 13:30。产品负责人对 sprint 目标进行总体介绍,概括产品 backlog。定下演示的时间地点。
  - 13:30 – 15:00。团队估算时间,在必要的情况下拆分 backlog 条目。产品负责人在必要时修改重要性评分。理清每个条目的含义。所有重要性高的 backlog 条目都要填写“如何演示”。
  - 15:00 – 16:00。团队选择要放入 sprint 中的故事。计算生产率,用作核查工作安排的基础。
  - 16:00 – 17:00。为每日 scrum 会议(以下简称每日例会)安排固定的时间地点(如果和上次不同的话)。把故事进一步拆分成任务。

# 估算

- 估算是很困难的，因为这是在预测未来。而历史证明，我们的估算经常是错误的——误差巨大，而且在很多时候都是这样。
- 估算经常是错误的，但是估算过程仍然是有用的。估算过程是管理前方的不确定性的契机，它可以被当做风险管理的工具，可以发现误解、不一致以及需要进一步调查的地方。
- 团队共同完成估算可以让大家建立对工作一致的理解。
- 但是，根据**收益递减原理**，不应在估算上花太多的时间。可以做出一个快速但不那么准确的估计，也可以再多花一点时间做一个更准确的估计，但花上几天的时间得出精确的小时数就没用了。

## 估算单位 (story point) 1

- “猜一下附近一栋高层建筑的高度” 比较难； “那边的两个建筑，哪个更高？ 一个相对另一个高多少？” 容易一些。
- Story point：故事点，选取可识别的最小用例为2个story point.其它估算都是相对值，在所有sprint中保持该相对值一致。
- 另外的好处，可以在估算时缩小人与人的能力误差（10倍程序员）。
  - 高级工程师和入门工程师完成一个任务的绝对时间是不同的，但这个工作量却是确定的。

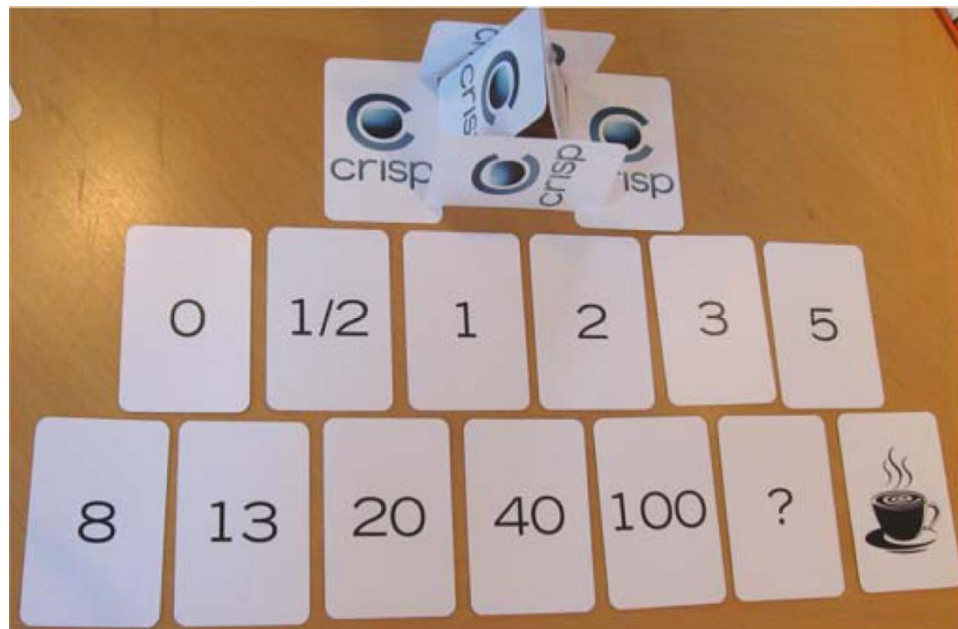
## 估算单位 (story point) 2

- 在估算速度时，估计能在一个迭代周期内能够完成的story point。
- 比如：我们一个迭代周期可以完成20个story point,则我们可以按照优先级顺序在现有product backlog中选取20个story point以内的user story放在本迭代周期内完成。

## 估算单位的另一种选择—T恤尺码

- 经常使用的序列是S、M、L。
- 如果工作项比L大，他们会把它拆分成大小为S、M和L的更小条目。一个XL的条目会占用团队太多的产能，也不便于管理。
- 必要时可以使用XS和XL。

## 估算过程—工具 “计划扑克”



- 0：这个故事很简单，几分钟搞定；？：一点概念没有，没想法；咖啡杯：太累了，歇会吧
- 斐波那契数列：0,1,1,2,3,5,8,13,21,34...
- 微信小程序：规划卡片
- 应用程序：ScrumPlanningPoker



# 估算过程

- 1.每个团队成员拿到一组卡片。
- 2.产品负责人或者一名团队成员扮演阅读者的角色，他负责阅读需要估算产品 Backlog的条目，并且询问大家是否有疑问。
- 3. 团队讨论这个条目。
- 4.当团队理解了这个条目之后，每个团队成员按照自己的想法给出估算结果，并且选择对应的扑克出牌，估算结果不能告诉其他人，出牌时数字朝下扣在桌面上。
- 5.阅读者向大家确认是否都已经确定估算结果，确认后，数“1,2,3”,大家同时展示估算结果。
- 6.团队评估不同的估算结果.我们是否想法一致？我们是否存在分歧？有没有什么是我没有考虑到的？团队共同讨论估算的差异，最终达成一致。如果差异不可接受，返回3.
- 7. 返回2，开始估算下一个条目。

## 估算有差异的原因

- 1、需求理解不同。例如：复杂登陆和简单登陆。
- 2、技术不熟悉。例如：没有嵌入式经验的工程师估算嵌入式系统开发内容。

## 扑克牌估算法价值

- 传统估算通常是一个人在思考，而使用估算扑克估算时，鼓励跨职能团队的多个团队成员参与估算，团队成员可以从不同的视角来思考和分析问题，估算的过程中考虑的更加全面、估算也更加准确。
- 在估算的过程中，团队对估算的结果进行讨论和评判，在一个高度透明的环境下，估算的结果更加真实和客观。这样也避免了很多时候过于武断，或是拍脑袋做出的决定。
- 估算的过程也是一个知识分享和学习的过程，对某一个条目不清楚的成员通过其他成员的阐述会增加对该条目涉及到的要点的认识。

## 另一种扑克估算法

- 在《Agile Adoption Patterns: A Roadmap to Organizational Success》一书中 Amr Elssamadisy 以另一个方式描述了计划扑克的过程。那就是对每一个需求:
- (1) 由一个领域专家向团队解释需求并回答问题, 以确保大家都理解。
- (2) 进入以下轮次, 只有在没能达成一致时才进入下一轮次。
  - 第一轮:大家用卡片独立投票, 不要讨论。比如像上面一样数1, 2, 3 一起投票。
  - 第二轮:让人们先思考一到两分钟, 再进入下一轮独立投票, 还是不要讨论。
  - 第三轮:大家先解释一下自己为什么给出这样的投票, 然后再次投票。
  - 如果还没有达成一致, 就暂时搁置, 先进行下一个需求的估算。
- 这个方法让你快速得到足够好的估计。如果能够达成一致, 就不会对需求做详细的讨论了。

## 卡片队列估算法1

- (1) 把每个条目的描述写在单独的卡片上，这个应该在估算前就准备好。
- (2) 把所有的卡片摞成一摞放在桌上。
- (3) 拿出第一张卡片，放在旁边。
- (4) 再拿出一张，问：“这个比第一个大还是小？”为了回答这个问题，你可能需要和其他人讨论一下这个工作项是干什么的。这需要时间，但别太长，记住这是一个快速估算。
- (5) 当你决定了它更大或更小后，把它放在原先卡片的上方或下方，形成一列。

## 卡片队列估算法2

- (6) 重复第4和第5步，直到用完所有的卡片。
- (7) 现在你可以把所有的卡片分组了——这也是该技术的第二阶段。从桌上底部(最小)的卡片开始，声明第一组的估算是“小的”。
- (8) 继续向上，决定到何时为止卡片已经变大到需要用一个新的估算值(如“中等”)了，让团队决定。
- (9) 很快，你就可以把整列卡片再过一遍，并加总(5个的估计是S, 4个M, 等等)以得到整个列的总和，这也就是需求的总工作量估计。从队列最下面的部分开始，第一部分可以被当做S。

## “金发女孩” 估算技术

- 这一技术从儿童故事《金发女孩和三只小熊》中受到启发。
- 作为对工作项大小估计的替代，你可以把工作项调整到合适的大小。
- 你不再是指定每个工作项的大小，而是去分割或组合工作项，让它们的规模大致相当，并便于后续操作。



## 减小故事规模

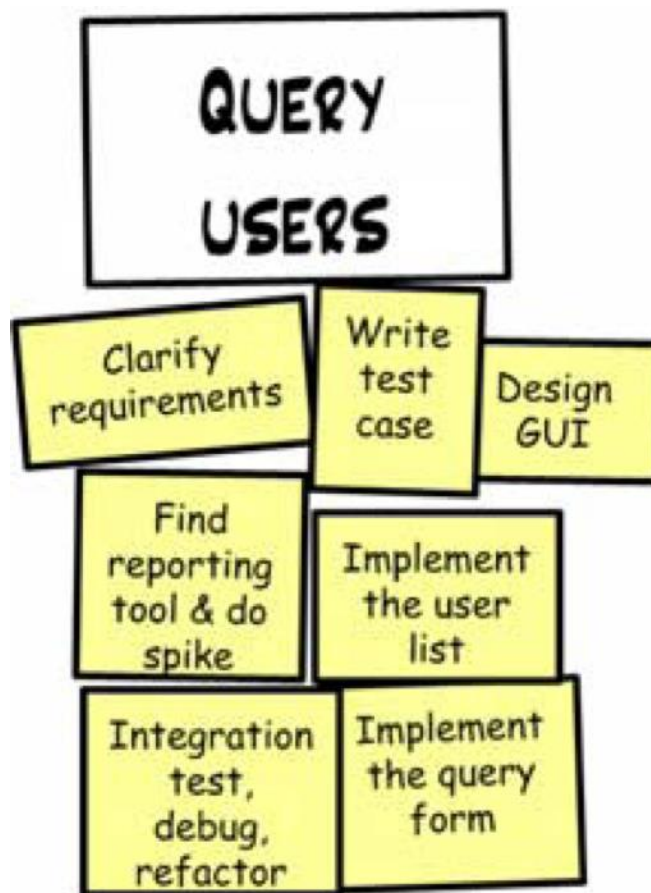
- 很多团队要求每个故事的大小在2-8个故事点，大于8要求拆分故事。





## 故事 (story) 与任务 (task)

- Story是可以交付的东西, Task是不可以交付的, Product Owner对Task不关心;



## 计划会议 (续)

- 定下每日例会的时间和地点：这必须是每一个成员都能接受的时间和地点。
- 确定技术故事：需要完成但是不属于可交付物的东西，如：
  - 安装持续构建服务器
  - 编写系统设计概览

# 计划会议产物—Sprint信息页

## **Jackass team, sprint 15**

### **Sprint goal**

- Beta-ready release!

### **Sprint backlog** (estimates in parenthesis)

- Deposit (3)
- Migration tool (8)
- Backoffice login (5)
- Backoffice user admin (5)

Estimated velocity: 21

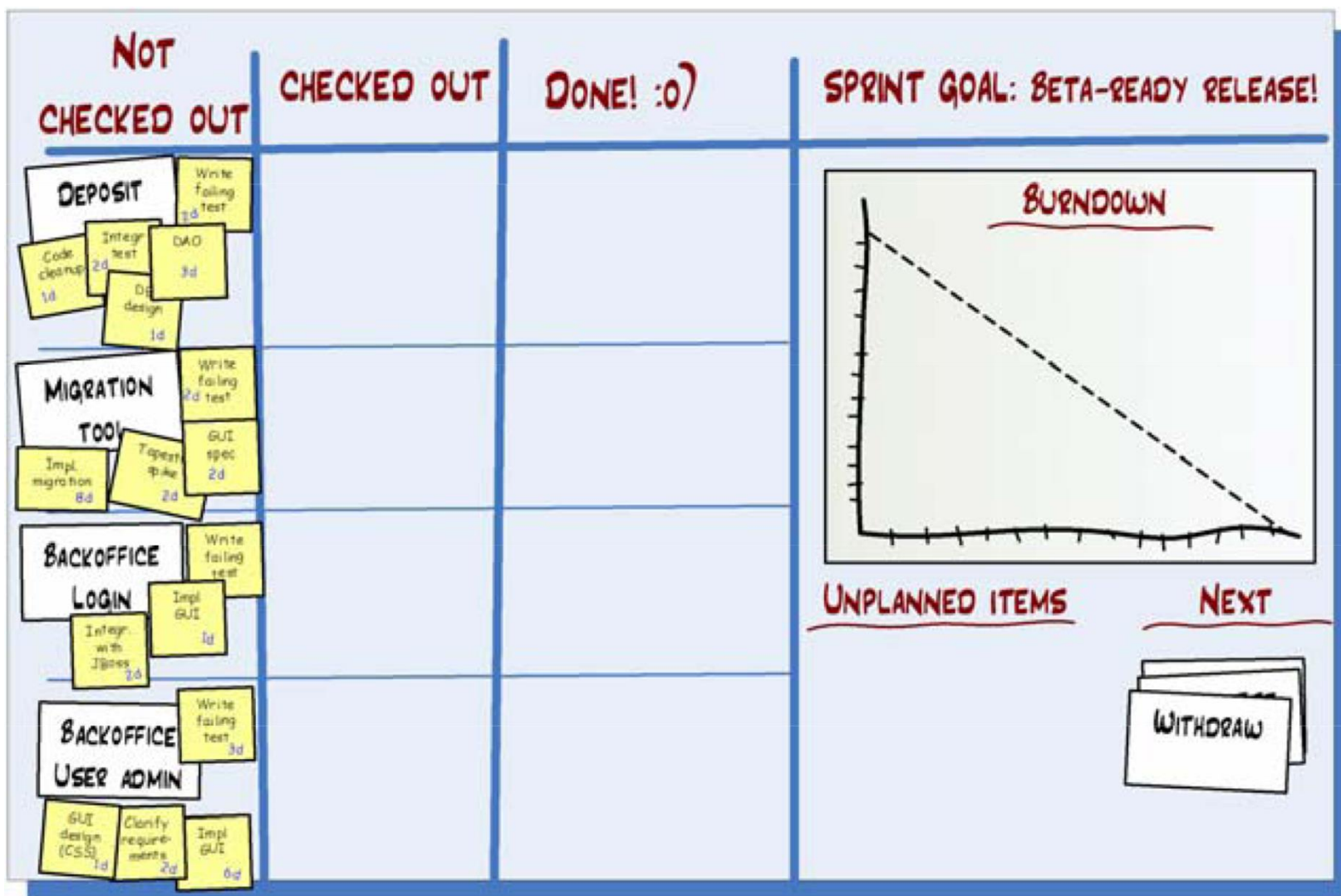
### **Schedule**

- Sprint period: 2006-11-06 to 2006-11-24
- Daily scrum: 9:30 – 9:45, in the team room
- Sprint demo: 2006-11-24, 13:00, in the cafeteria

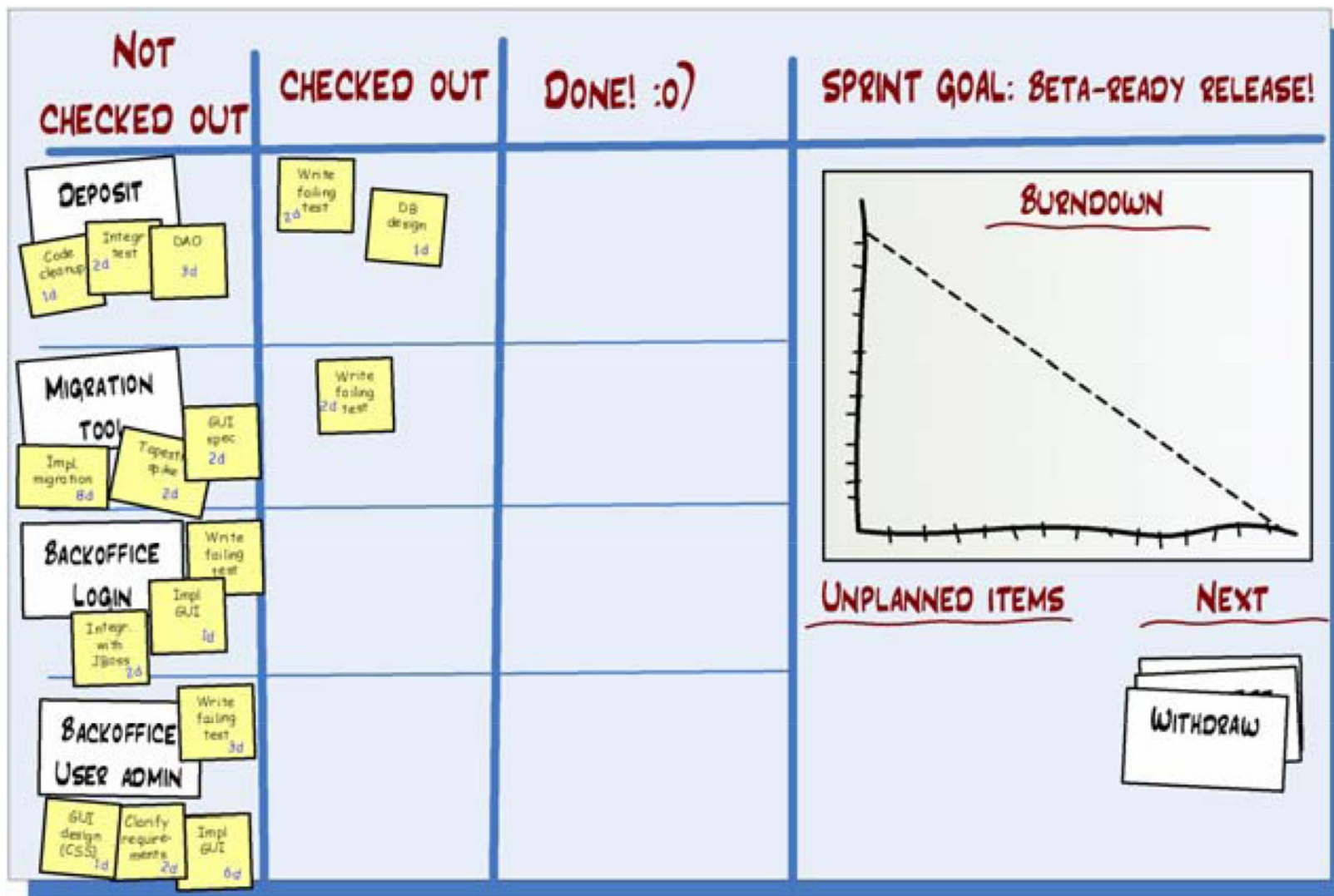
### **Team**

- Jim
- Erica (scrum master)
- Tom (75%)
- Eva
- John

# Sprint 管理—白板

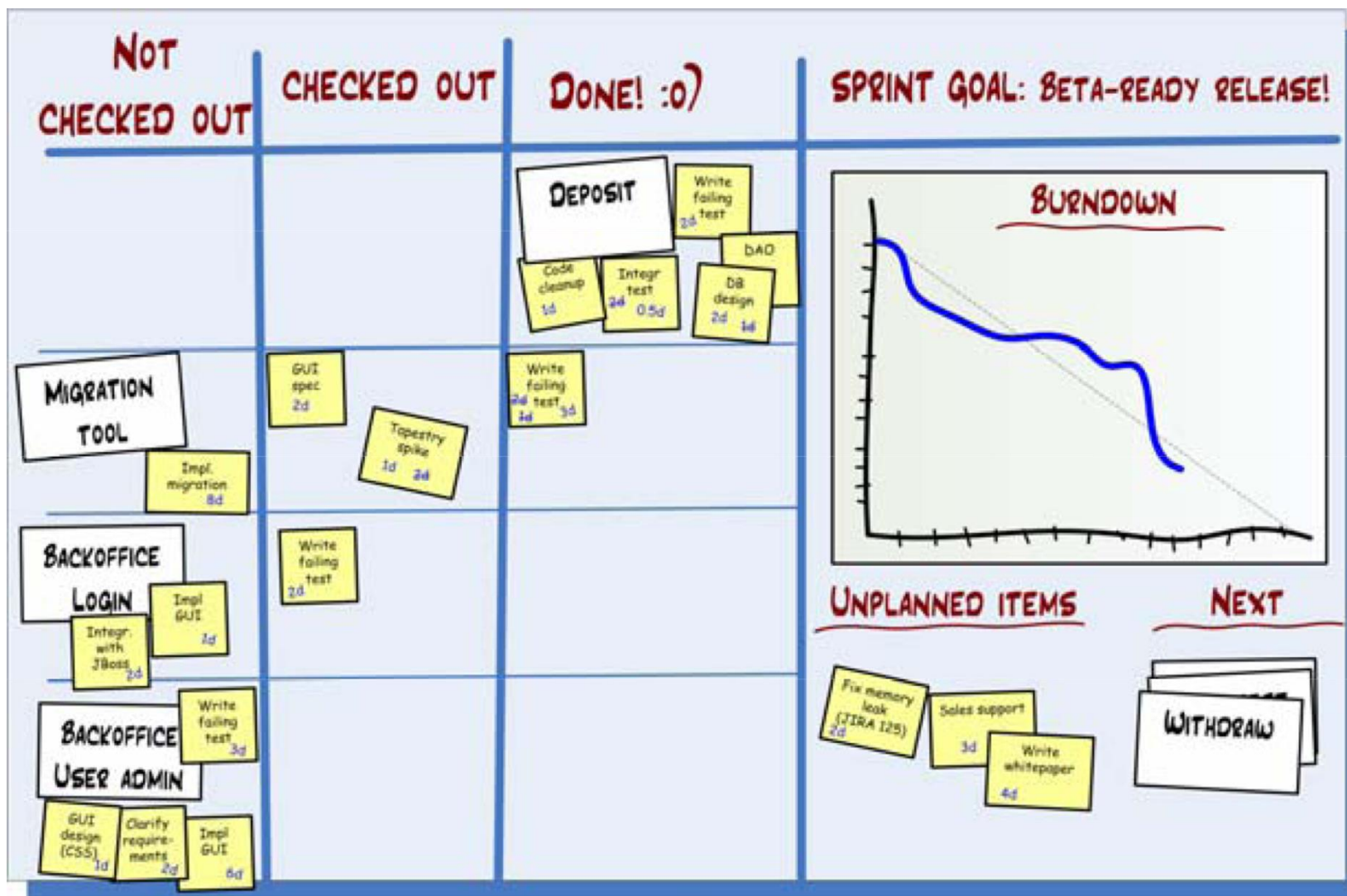


# Sprint首次工作日之后的白板

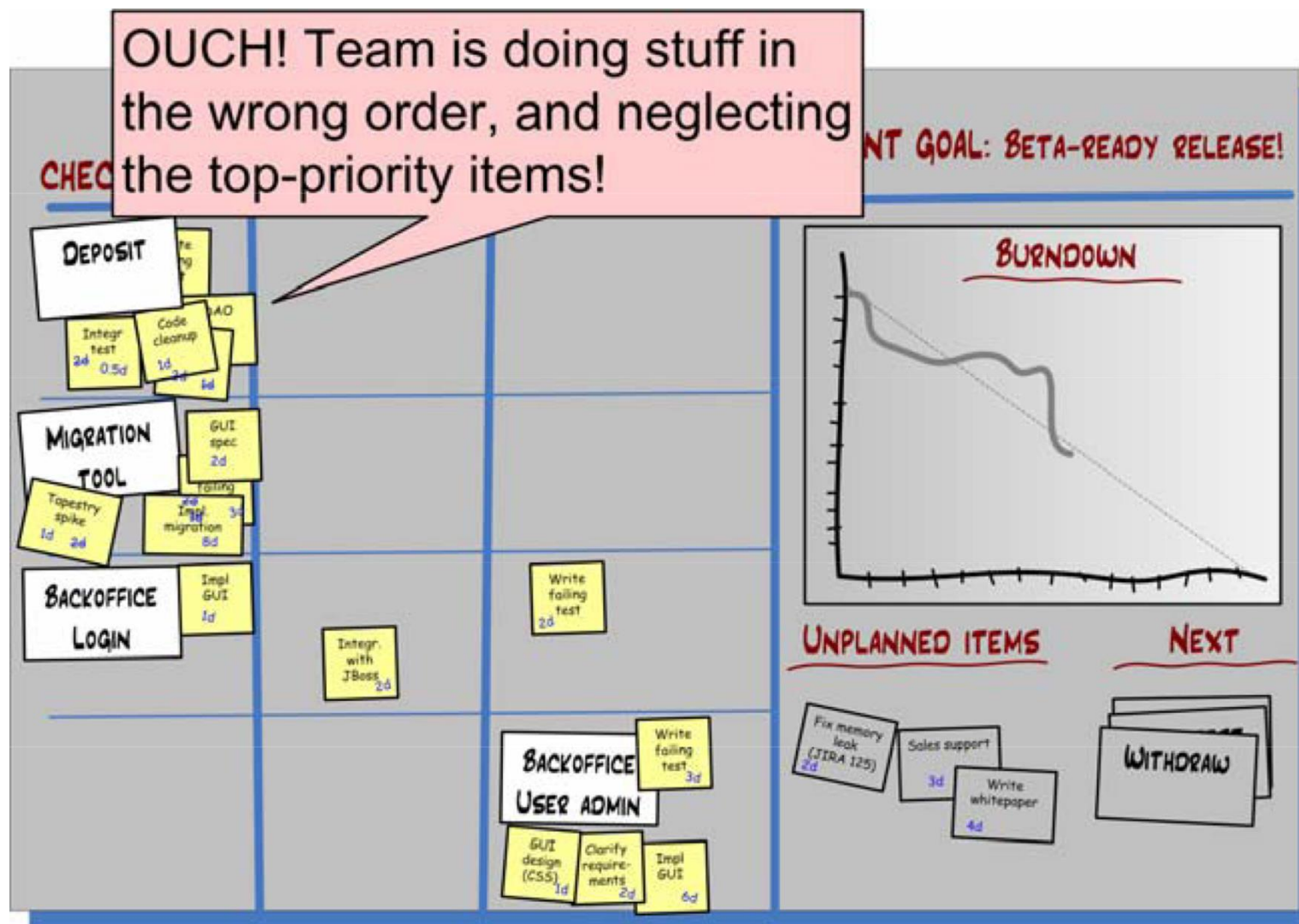




# 几天之后的白板

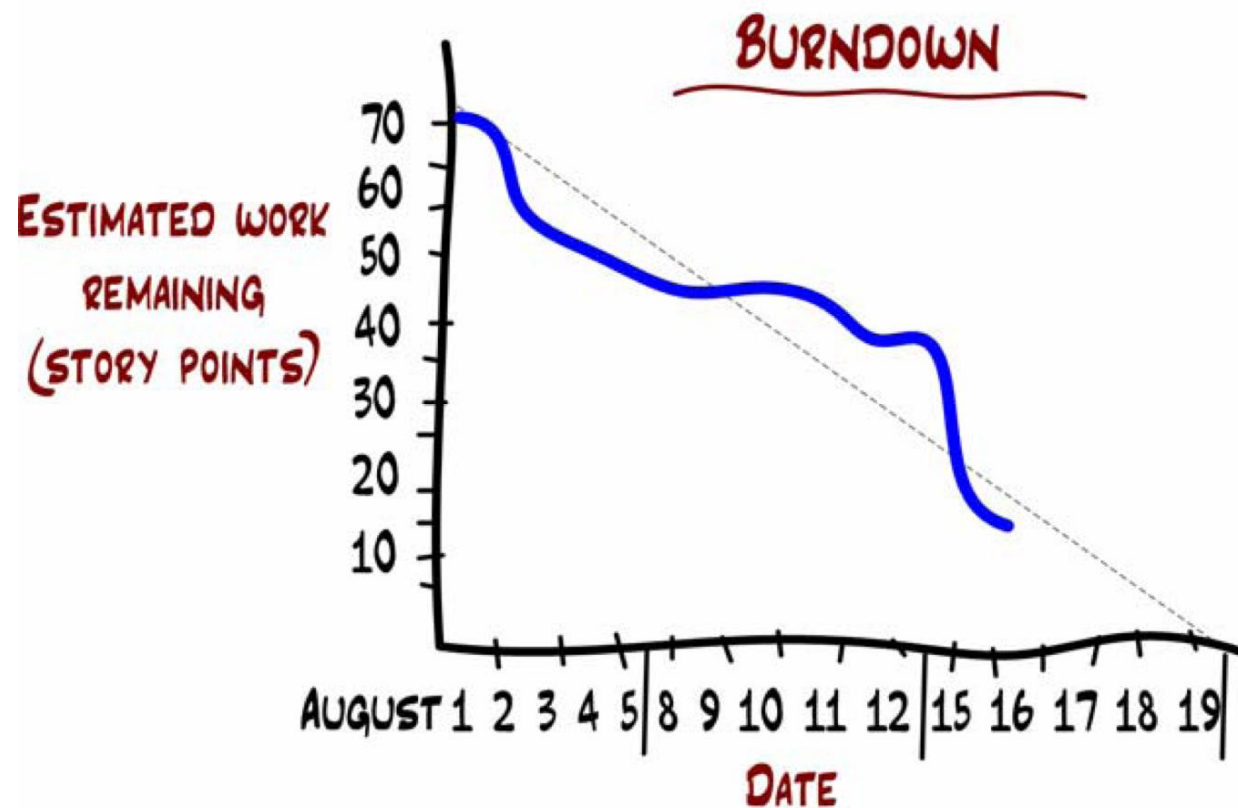


# 白板的警示



## 跟踪进度—燃尽图

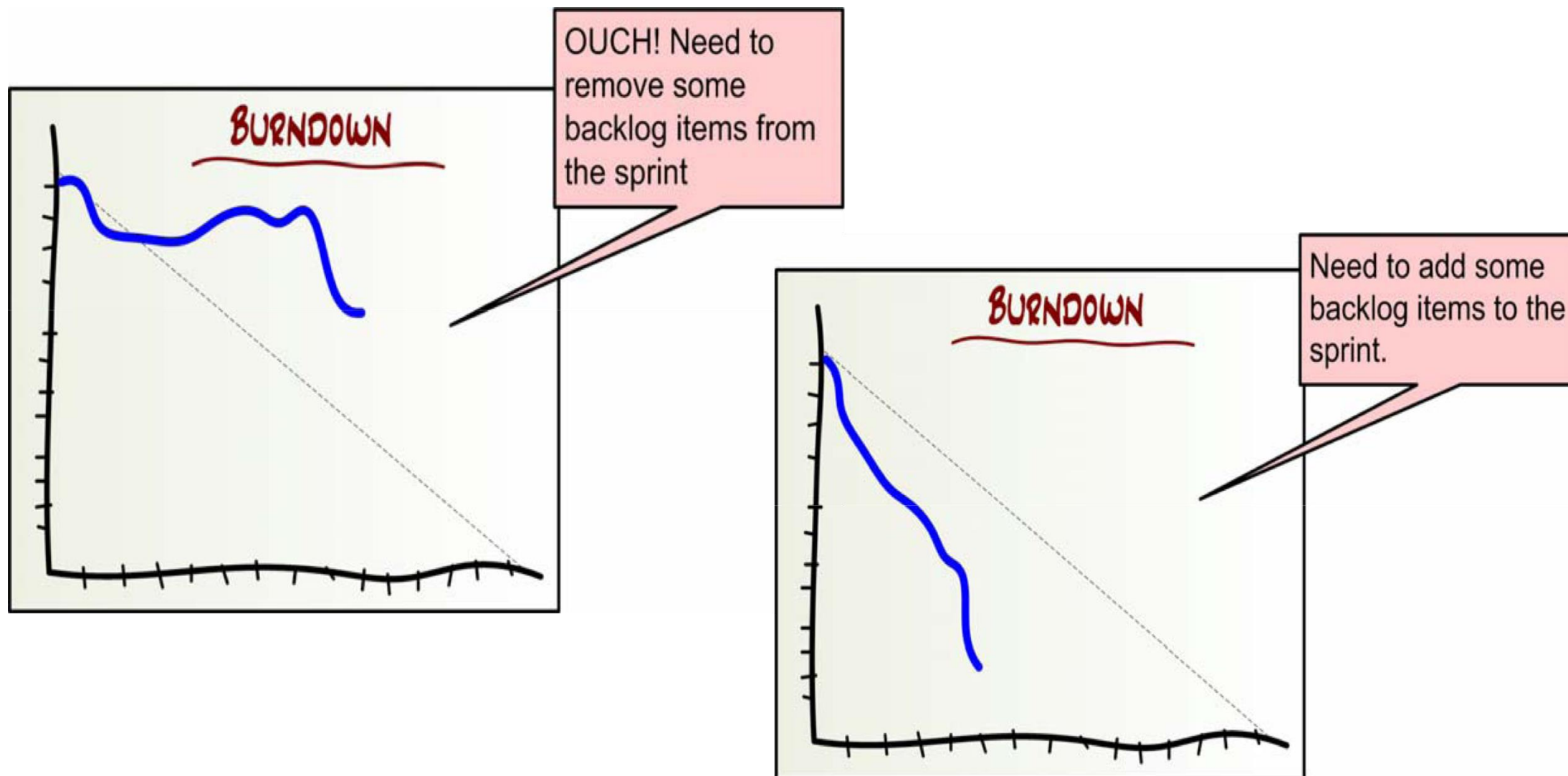
- 包含信息：
  - —该Sprint共需要完成70个故事点；
  - —目前团队还剩下15个故事点要完成
  - —估计：按照目前速度该Sprint目标可以达到



燃尽图表示生产速度，提醒团队适当修改生产速度



# 燃尽图的警示



## 每日站会—Daily Scrum

- 不超过15分钟
- 回答三个问题：
  - “昨天我做了什么。”
  - “今天准备干什么。”
  - “你遇到了什么障碍，需要其他人如何帮你。”
- 移动任务板上的即时贴到对应的地方
- 每日例会一结束就要计算剩余工作故事点并更新燃尽图
- 团队每日报到、简短、及时开始与结束、聚焦重点、有规律性

## 另一种站会

- 关注事(接力棒)— 而不是人(运动员)
  - 不关注每个人做了或没做什么，而是关注整个 workflow 或者单个工作项的流动是否有什么问题。
- 遍历看板墙
- 关注味道
- 每日站会之后，鼓励自发改善会议



# Sprint演示会议

- 为什么定义Sprint结束于演示：
  - 其他人可以了解你的团队在做什么
  - 团队得到认可，团队成员感觉很好
  - 不同的团队得到交流，讨论各自工作
  - 演示可以吸引相关人士的关注，并得到重要的反馈
  - 演示会迫使团队真正完成一些工作而不是貌似完成，这样不会污染下一个Sprint。

# Sprint演示——检查列表

- 确保明确阐述Sprint目标
- 集中精力演示可以实际工作的代码
- 演示保持快节奏
- 演示我们做了什么而不是我们怎么做的
- 不演示细碎bug的修复和微不足道的特性

## Sprint回顾会议

- Sprint回顾是仅次于Sprint计划的第二重要的事件！
- 这是做出改进的最佳时期。
- 主题：我们怎样才能在下个Sprint中做的更好，不是追究责任！

“

无论我们发现了什么，考虑到当时的已知情况、个人的技术水平和能力、可用的资源，以及现实状况，我们理解并坚信：每个人对自己的工作都以全力以赴。

”

# Sprint回顾——活动列表1

- 根据要讨论的内容范围,设定时间为 1 至 3 个小时。
  - 参与者:产品负责人,整个团队还有我自己。
  - 我们换到一个封闭的房间中,或者舒适的沙发角,或者屋顶平台等等类似的场所。只要能够在不受干扰的情况下讨论就好。
  - 我们一般不会对团队房间中进行回顾,因为这往往会分散大家的注意力。
  - 指定某人当秘书。

## Sprint回顾——活动列表2

- Scrum master 向大家展示 sprint backlog,在团队的帮助下对sprint 做总结。包括重要事件和决策等。
- 我们会轮流发言。每个人都有机会在不被人打断的情况下讲出自己的想法,他认为什么是好的,哪些可以做的更好,哪些需要在下个 sprint 中改变。
- 我们对预估生产率和实际生产率进行比较。如果差异比较大的话,我们会分析原因。
- 快结束的时候,Scrum master 对具体建议进行总结,得出下个 sprint 需要改进的地方。



## ■ Sprint回顾——使用白板

- Good: 哪些做法可以保持
- Could have been better: 那些做法需要改变
- Improvements: 具体改进想法





# Scrum局限

- 没有技术实践！
- 可以使用极限编程技术实践：测试驱动开发、简单设计、重构、持续集成等等。

# Scrum Guide 2020 核心变更与争议

## ●一、关键改进

- 团队结构简化：取消"开发团队"专属称谓，统称"开发者"
- 目标体系强化：引入产品目标、冲刺目标、承诺性待办项三层目标架构
- 仪式优化：每日站会取消固定三问，聚焦进度与障碍
- 指导原则：删减50%内容，强调适应性与精简

## ●二、主要争议

- 角色职责模糊化争议
- 仪式灵活性带来的执行差异
- 精简版指南对新手支持不足



# Scrum三大理论支柱

## ●透明

- 过程与工作成果对团队和利益相关者可见

## ●检视

- 定期审查进展，识别偏差（通过5大事件实现）

## ●适应

- 及时调整偏离目标或质量的问题

# Scrum五大价值观

- 承诺 ▶ 专注目标
- 专注 ▶ 聚焦Sprint任务
- 开放 ▶ 坦诚面对挑战
- 尊重 ▶ 认可成员能力
- 勇气 ▶ 解决棘手问题



# Scrum团队结构

## ●组成

- Product Owner：最大化产品价值，管理Product Backlog
- Scrum Master：确保框架实施，移除障碍
- Developers：跨职能开发，交付可用增量

## ●团队特性

- 10人以内，自管理、跨职能、专注单一Product Goal

# Scrum五大事件

## ●Sprint

- 固定周期 ( $\leq 1$ 个月) , 不可变更目标与质量

## ●Sprint Planning

- 确定Sprint Goal与交付计划 (时间盒: 8小时/月)

## ●Daily Scrum

- 15分钟每日站会, 调整当日计划

## ●Sprint Review

- 展示增量, 调整Backlog (时间盒: 4小时/月)

## ●Sprint Retrospective

- 改进流程与效能 (时间盒: 3小时/月)

# Scrum三大工件与承诺

工件	定义	承诺
Product Backlog	产品需求动态清单	Product Goal（产品愿景）
Sprint Backlog	Sprint任务计划（目标+选定的条目）	Sprint Goal（迭代目标）
Increment	符合完成标准的可交付成果	Definition of Done（完成标准）



# Scrum的核心优势

- 轻量灵活：仅定义必要规则，兼容多种实践
  - 持续改进：通过事件循环实现经验反馈
  - 价值驱动：以Product Goal为导向，确保交付有效性
  - 协作透明：跨角色协作，信息共享最大化
- 
- Scrum适用性
    - 超越软件领域，适用于复杂创新工作



# THANKS