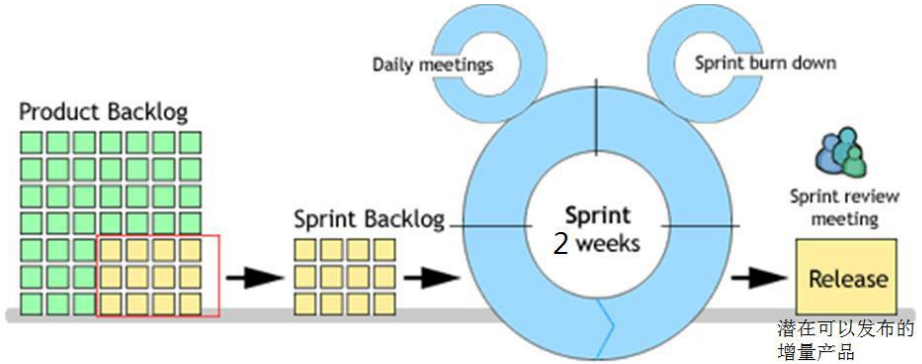
Scrum 项目管理

2016. 5. 11

什么是Scrum

- ✓ Scrum是一个敏捷(**Agile**) **开发框架**，是一个**增量**的、**迭代**的开发过程。
- ✓ 在这个框架中，整个开发周期包括若干个小的**迭代周期**，每个小的的迭代周期称为一个**Sprint**，每个Sprint的建议长度**2到4周**。
- ✓ 在Scrum中，使用**产品Backlog** 来管理产品或项目的需求。
- ✓ 产品backlog是一个按照商业价值排序的需求列表，列表条目(**PBI**)的体现形式通常为**用户故事** (**User-Story**) 。
- ✓ Scrum的开发团队总是先开发对客户具有**较高价值**的需求。在每个Sprint中，Scrum开发团队从产品Backlog中**挑选**最有价值的需求进行开发。
- ✓ Sprint中挑选的需求经过Sprint计划会议上的分析、讨论和估算得到一个Sprint的**任务列表**(**Task**)，我们称它为**Sprint backlog** (**SBI**) 。
- ✓ 在每个Sprint结束时，Scrum团队将交付**潜在可交付**的产品**增量**。



- 计划、主动争抢
  - 在球场上：在比赛每段的开始，双方都要摆开阵势，并计划本段的进攻/防守路线和策略，教练和队长都可以参与计划。
    开球时球员拼命争抢。
  - 在软件开发公司：在每个迭代的开始，团队领导者都应该做好本Sprint的计划，尤其是需求条目的优先级排序、选择本Sprint的工作、设定必须完成的内容等。
    Sprint开始时，团队成员主动领取任务。

- 目标、自组织、灵活应变!
  - 在在球场上：当哨声响起，尽管队员们努力按照既定计划推进，然而场上瞬息万变，队员不可能实时按照教练或队长的指令亦步亦趋地行事，而是靠平时训练中形成的素养见机行事，达成目标。
  - 在软件开发公司：在每个Sprint开始后，团队领导不可能也不需要事必亲恭地者介入每件事情，而是应该由具体执行的人选择如何去做。
    团队领导要做好的是协调资源、解决困难、提供指导，以达成目标。

敏捷 (Agile) 的价值观

Values (Agile Manifesto)

价值观（敏捷宣言）

- **Individuals and interactions** over processes and tools
  - **个体与互动** 高于 流程与工具
- **Working software** over comprehensive documentation
  - **工作的软件** 高于 详尽的文档
- **Customer collaboration** over contract negotiation
  - **客户合作** 高于 合同谈判
- **Responding to change** over following a plan
  - **响应变化** 高于 遵循计划

Two different ways we try to resolve problems:

Defined vs. Empirical Process

预定义 vs 试验性过程

Command and Control		Learn as we go
命令和控制		边前进边学习
Plan in details		Change happens
详细计划	**VS**	变化会发生
Enforce the plan		Embrace change
强制按计划		拥抱变化
“Control” change		Inspect and Adapt
“控制”变化		检视和调整

“When the process is too complicated for the defined approach, the empirical approach is the appropriate choice.”

--- Process Dynamics, Modeling, and Control Ogunnaike & Ray, Oxford University Press, 1992

敏捷原则（Agile Principles）

1. **尽早**地、**持续**地交付有价值的软件**来使客户满意**是最高优先级的工作。
2. 即使到了开发的后期，也**欢迎改变需求**。敏捷过程适应变化来为客户创造竞争优势。
3. 以几周到几月的间隔**频繁交付可工作的软件**，交付间隔越短越好。
4. 在整个项目开发期间业务人员和开发人员一起工作。
5. 激励团队成员建设项目。提供所需的环境与**支持并信任他们**能够完成工作。
6. 在团队内部以及团队之间最有效最高效的传递信息的方式是**面对面的沟通**。
7. **可工作的软件**是首要的进度**度量**。
8. 敏捷过程提倡持续性的开发。发起人、开发者和用户应保持长期的、恒定的工作速率。
9. 持续**追求技术卓越**和优良设计能提高敏捷性。
10. 敏捷的根本在于简单 - 使不用做的工作最大化的艺术。
11. 最好的架构、需求和设计出自于**自组织的团队**。
12. 每隔一定的时间，**团队反思**如何更有效地工作，然后相应地作出调整。

Scrum的理论基础

Scrum以经验性过程控制理论（经验主义）做为理论基础的过程。经验主义主张知识源于经验, 以及基于已知的东西做决定。Scrum 采用迭代、增量的方法来优化可预见性并控制风险。
Scrum 的三大支柱支撑起每个经验性过程控制的实现：透明性、检验、和适应。

第一：透明性（Transparency）

透明度是指，在软件开发过程的各个环节保持高度的可见性，影响交付成果的各个方面对于参与交付的所有人、管理生产结果的人保持透明。管理生产成果的人不仅要能够看到过程的这些方面，而且必须理解他们看到的内容。也就是说，当某个人在检验一个过程，并确信某一个任务已经完成时，这个**“完成”**必须等同于他们对完成的定义。

第二：检验（Inspection）

开发过程中的各方面必须做到**足够**频繁地检验，确保能够**及时**发现过程中的重大偏差。在确定检验频率时，需要考虑到检验会引起所有过程发生变化。当规定的检验频率超出了过程检验所能容许的程度，那么就会出现问题。幸运的是，软件开发并不会出现这种情况。另一个因素就是检验工作成果人员的技能水平和积极性。

第三：适应（Adaptation）

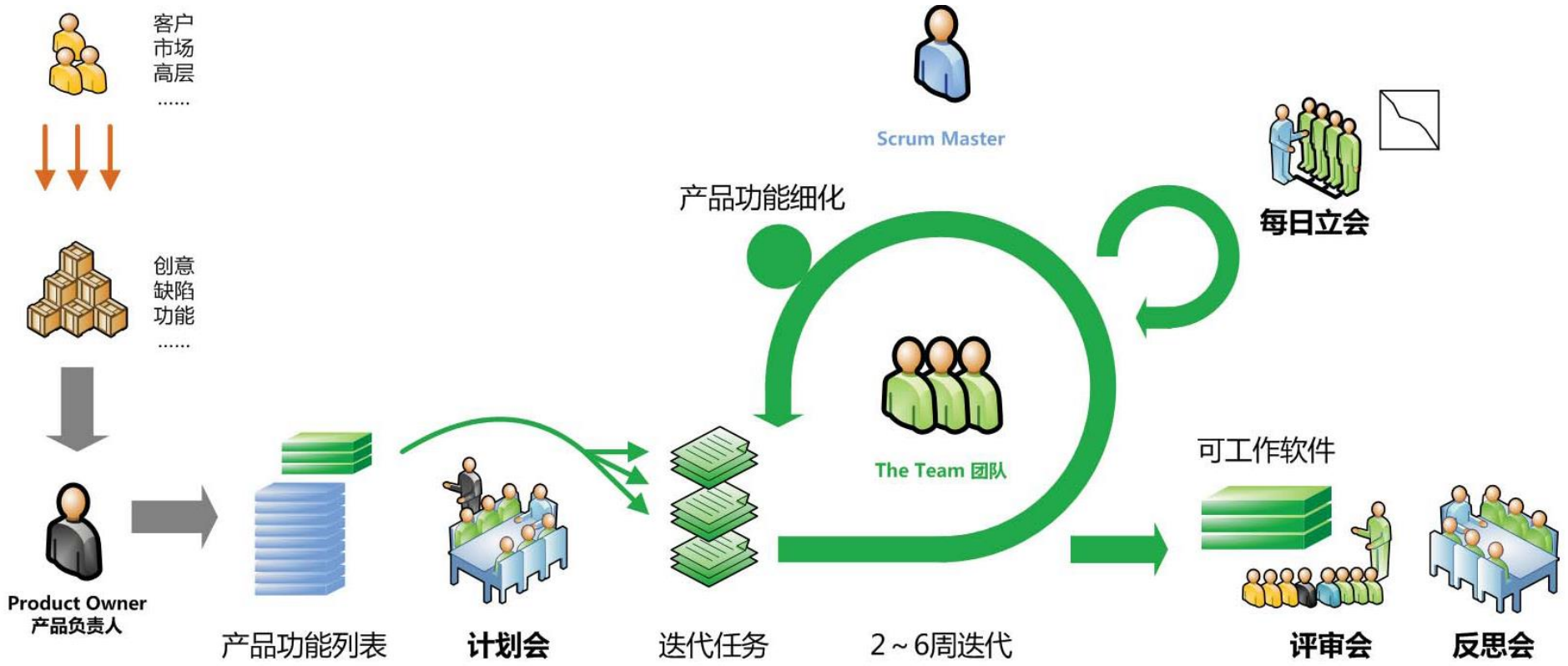
如果检验人员检验的时候发现过程中的一个或多个方面不满足验收标准，并且最终产品是不合格的，那么便需要对“**过程**”或是“**工件**”进行调整。调整工作必须尽快实施，以减少进一步的偏差。

Scrum中通过三个活动进行检验和适应：

1. **每日站立会**检验Sprint目标的进展，做出调整，从而优化次日的工作价值；
2. **Sprint评审和计划会议**检验发布目标的进展，做出调整，从而优化下一个Sprint的工作价值；
3. **Sprint回顾会议**是用来回顾已经完成的Sprint，并且确定做出什么样的改善可以使接下来的Sprint更加高效、更加令人满意，并且工作更快乐。

基本的Scrum迭代流程

- Scrum的迭代被称为**Sprint——冲刺**
- **冲刺**的特点：
  - 目标明确、路径清晰、没有干扰、纠正偏差、全力以赴



Scrum的三个角色

Scrum 团队是自组织、跨职能的完整团队，拥有完成工作所需要的全部技能。
Scrum 团队模式的目的是最大限度地优化适应性、创造性和生产力。
Scrum 团队中包括三个角色，他们分别是产品负责人（PO）、开发团队（Team）和 Scrum Master（SM）。

产品负责人（PO）

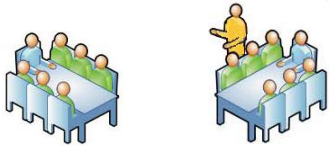
- 产品负责人是“一个人”，被授予产品决策权
- 建立产品愿景，驱动产品走向成功，确保开发团队所执行工作的价值
- 管理产品功能列表(Backlog)的唯一责任人
  1. 清晰地表达产品Backlog条目（PBI），对产品Backlog条目（PBI）进行排序
  2. 确保开发团队对产品Backlog条目（PBI）达到一定程度的理解
  3. 确保产品Backlog对所有人可见、透明、清晰，并且显示 Scrum 团队的下一步工作
- 任何人都不得要求开发团队按照另一套需求开展工作，开发团队也不允许听从任何其他人的指令。
- 决定版本发布的日期和内容
- 面向干系人代表团队，面向团队代表干系人

开发团队（Team）

- 管理Sprint Backlog，跟踪Sprint Backlog进度（包括质量），负责在每个 Sprint 的结尾交付潜在可发布的“完成”产品增量。
- 他们是自组织的，没有人（即使是 Scrum Master 都不可以）告诉开发团队如何把产品待办事项列表（task）变成潜在可发布的功能。
- 开发团队是跨职能的，团队作为一个整体拥有创造产品增量所需要的全部技能。
- Scrum 不认可开发团队成员的头衔，无论承担哪种工作他们都是开发者，此规则无一例外。
- 开发团队中的每个成员可以有特长和专注领域
- 持续进行自我改进，Sprint之间才能换人
- 开发团队最佳规模是大到足以保持敏捷性，小到足以完成重要工作（7 ± 2）。

Scrum Master（SM）

- 负责确保 Scrum 被理解并实施，确保 Scrum 团队遵循 Scrum 的理论、实践和规则。
- Scrum Master 服务于产品负责人
  - ✓ 在经验主义环境中理解长期的产品规划
  - ✓ 找到有效管理产品Backlog的技巧
  - ✓ 教导PO创建清晰简明的User-Story
  - ✓ 理解并实践敏捷，按需推动Scrum活动
- Scrum Master 服务于开发团队
  - ✓ 指导团队自组织和跨职能，培养团队
  - ✓ 移除团队Sprint中的障碍，辅助团队、保护团队
  - ✓ 在 Scrum 还未完全被采纳和理解的组织环境下指导开发团队，按需推动Scrum活动
- Scrum Master 服务于组织：
  - ✓ 领导并指导组织采用 Scrum，在组织范围内策划 Scrum 的实施
  - ✓ 负责Scrum价值观、原则和规则被采纳和彰显
  - ✓ 帮助员工及干系人理解并实施 Scrum
  - ✓ 发起能提升Scrum 团队生产力的变革



Scrum的三个“资产”

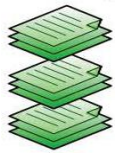
- 产品待开发项 **Product Backlog**是从客户价值角度理解的产品功能列表。
- 冲刺待开发项 **Sprint Backlog**是从开发技术角度理解的Sprint开发任务。
- 可工作软件 Working Software 是Sprint的交付增量（**Increment**）。

产品Backlog

- 功能、缺陷、增强等都可以是 **PBI** 。
- 一份动态的列表，一般以条目化的方式描述
- 客户、用户、团队等干系人必须能够理解
- 整体上从客户价值优先级排序，关注于“什么”带给用户最大的价值
- 高优先级的条目应有较详尽的描述，低优先级的条目可只有一个名称
- 常常以 **User-Story** 的方式呈现给团队，描述怎样使用而非怎样制造
- **User-Story** 即包含验收（评审）标准
- 包含估算信息，工作量一般为0.5～10个典型人天

Sprint Backlog

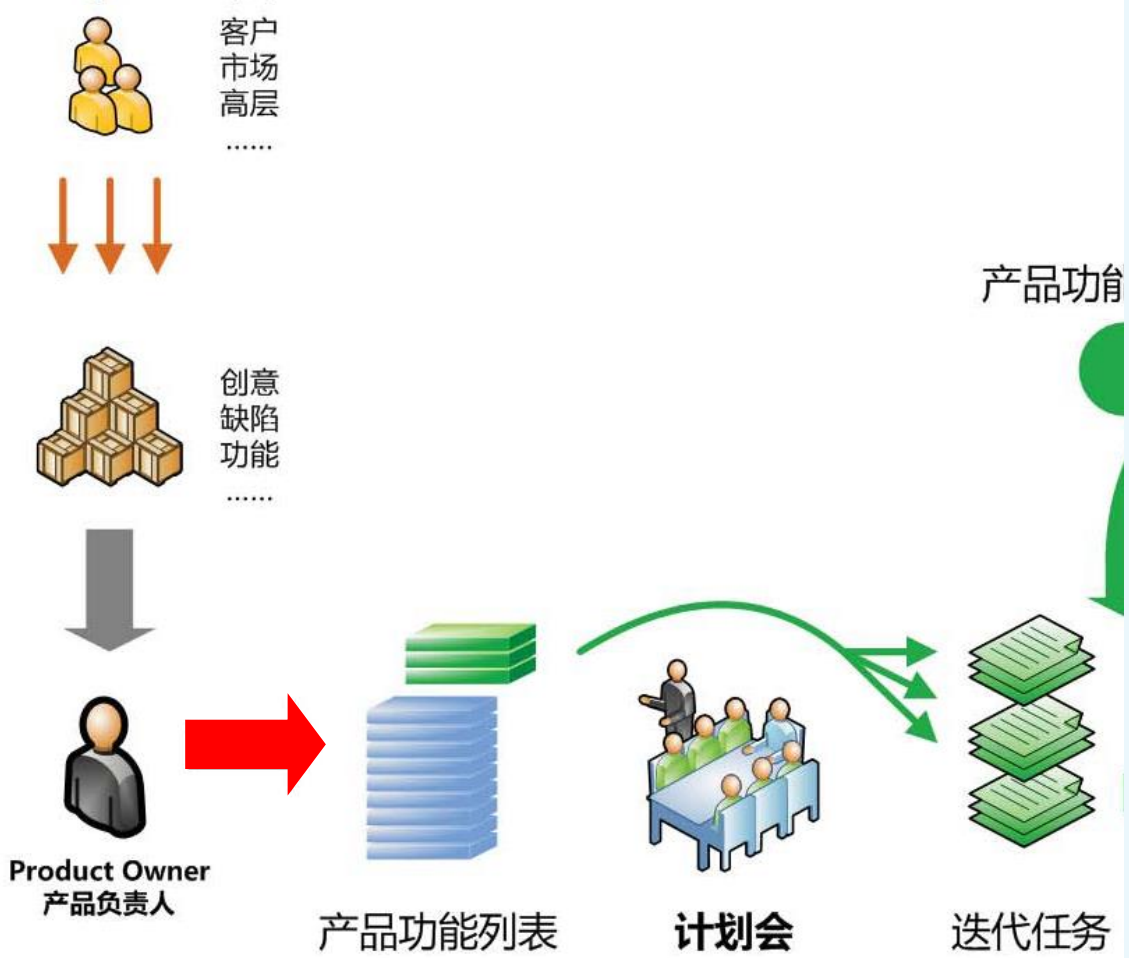
- 产品 **Backlog** 的延伸和子集
  - 为实现 Sprint 目标所要完成的工作集合
  - 将大块的工作分解为更小的单元，同时进行更细化的估算
  - 涵盖“恰到好处”的设计，开始关注于“怎么做”的问题：如何在一个 Sprint 内完成工作以交付价值
- 被开发团队拥有
  - 团队从 Product Backlog 中选取他们可以承诺完成的项目并创建 **Sprint Backlog**
  - Scrum Master 协作完成，不是由 Scrum Master 负责
  - 团队每天跟踪 Sprint 中剩余的工作
  - Sprint内的工作任务有可能动态涌现，任何团队成员可以添加，删除或变更 Sprint Backlog事项(SBI)
- 常常以 **task** 的形式出现在 **KanBan** 等可视化任务管理工具上



Increment

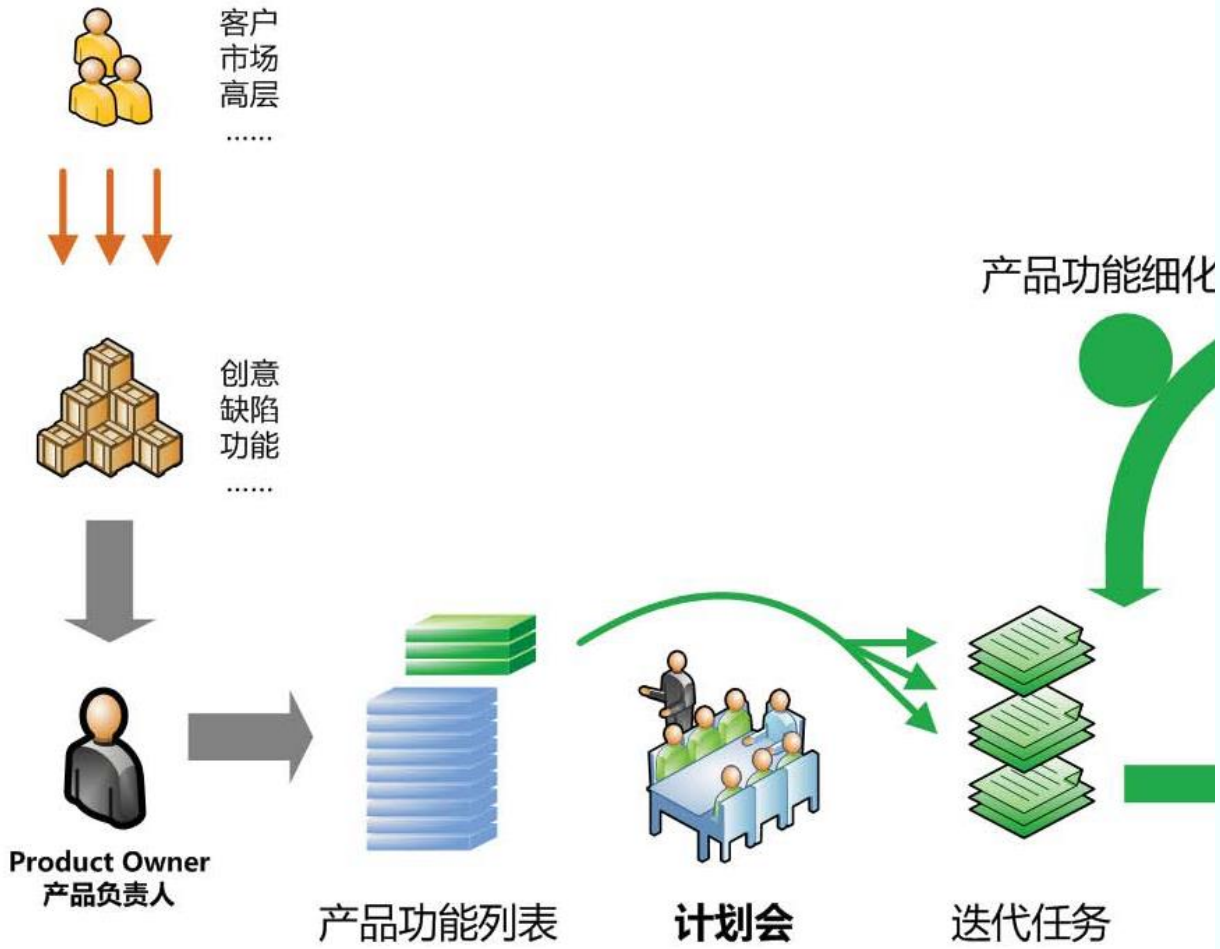
- 潜在可交付，并符合 **“完成”** 的定义
- **“可交付”** 在不同场景下差异很大，应视不同情况提前设定和选定交付标准。比如是否需要测试，是否需要性能优化，是否需要操作手册等等。
- 必须是可用的产品，不管PO是否决定对外发布
- 在正式产品中可能包括使用文档，甚至是纸质的。在新产品开发的初期，则可能只需要交付勉强可看到效果的产品。
- 产品负责人、用户代表等负责评审可工作软件。
- 若一个产品功能只是“差不多完成了”，会被视为不可交付。

Scrum的5个“活动”之一：梳理 Product Backlog



- 产品功能列表（Product Backlog）是一组条目化需求。
- 产品功能列表梳理的一个最大好处是为即将到来的几个 Sprint 做准备。
  （愿景……产品路线图……）
- 产品功能列表必须从客户价值角度描述，并按优先级排序。
- 产品负责人创建和维护产品功能列表。
- 产品开发决定了,有可能需要其它的技能和输入。
  因此，产品功能列表梳理最好是所有团队成员都参与的活动，
  至少团队的技术 Leader 应该参与（如果有），
  而不单单是产品负责人。
- 如何编写产品功能列表
  - 需求必须进行条目化管理，才能进行分配、开发、跟踪，才能描述什么做完了什么没做完。
  - 典型的描述方法，就是极限编程中提到的用户故事（User Story）。
  - “客户价值角度”就是描述用户如何使用，而不是描述技术层面如何实现。比如“实现手写输入”“实现游戏排行榜”，而不是“编写数据库底层”。
  - 用户故事的语法“作为一个……，可以……，以（以便）……”很好地保证了这一点。

Scrum的5个“活动”之二：Sprint Planning 会议



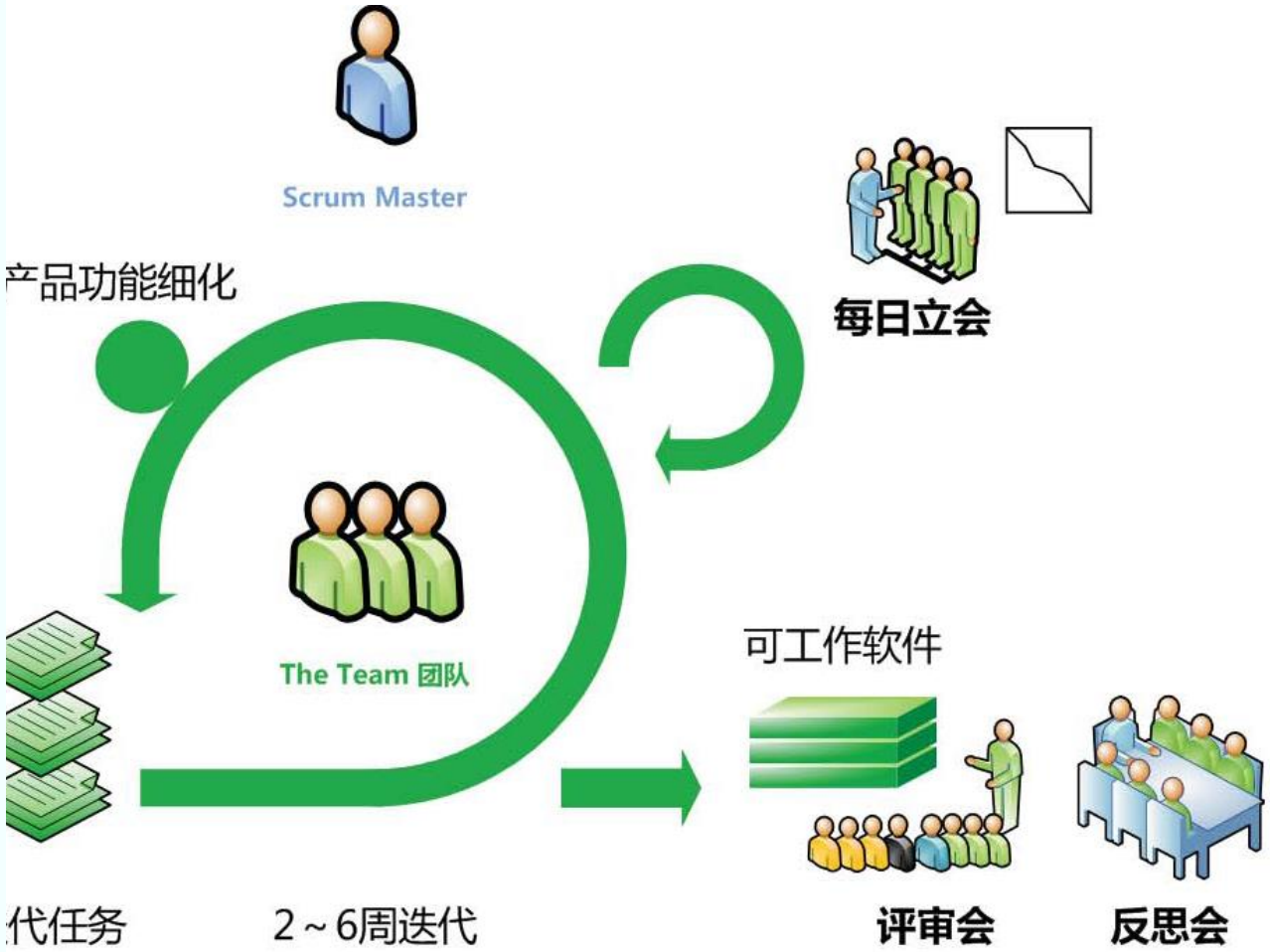
1. Sprint计划会在每个Sprint第一天召开，目的是选择和估算本次Sprint的工作项
2. 产品负责人逐条讲解最重要的产品功能
3. 开发团队共同估算故事所需工作量，直到本Sprint工作量达到饱和
4. 一般一个团队只有70%的工作可以进行正式工作，因此每个人月安排15人天左右即为饱和，新团队则可能低至10人天
5. 产品负责人参与讨论并回答与需求相关的问题，但不干扰估算结果

Scrum的5个“活动”之三：Sprint 站立会

- 参数：
  - 每日，同一时间，同一地点，站立，15分钟
- 参与者:
  - Development Team,[ScrumMaster],[Product Owner]
  - 其他人可以受邀来旁听
- 三句话：
  - 昨天我**完成**了什么?
  - 今天我要**完成**什么?
  - 有什么**障碍**影响我的进度吗?

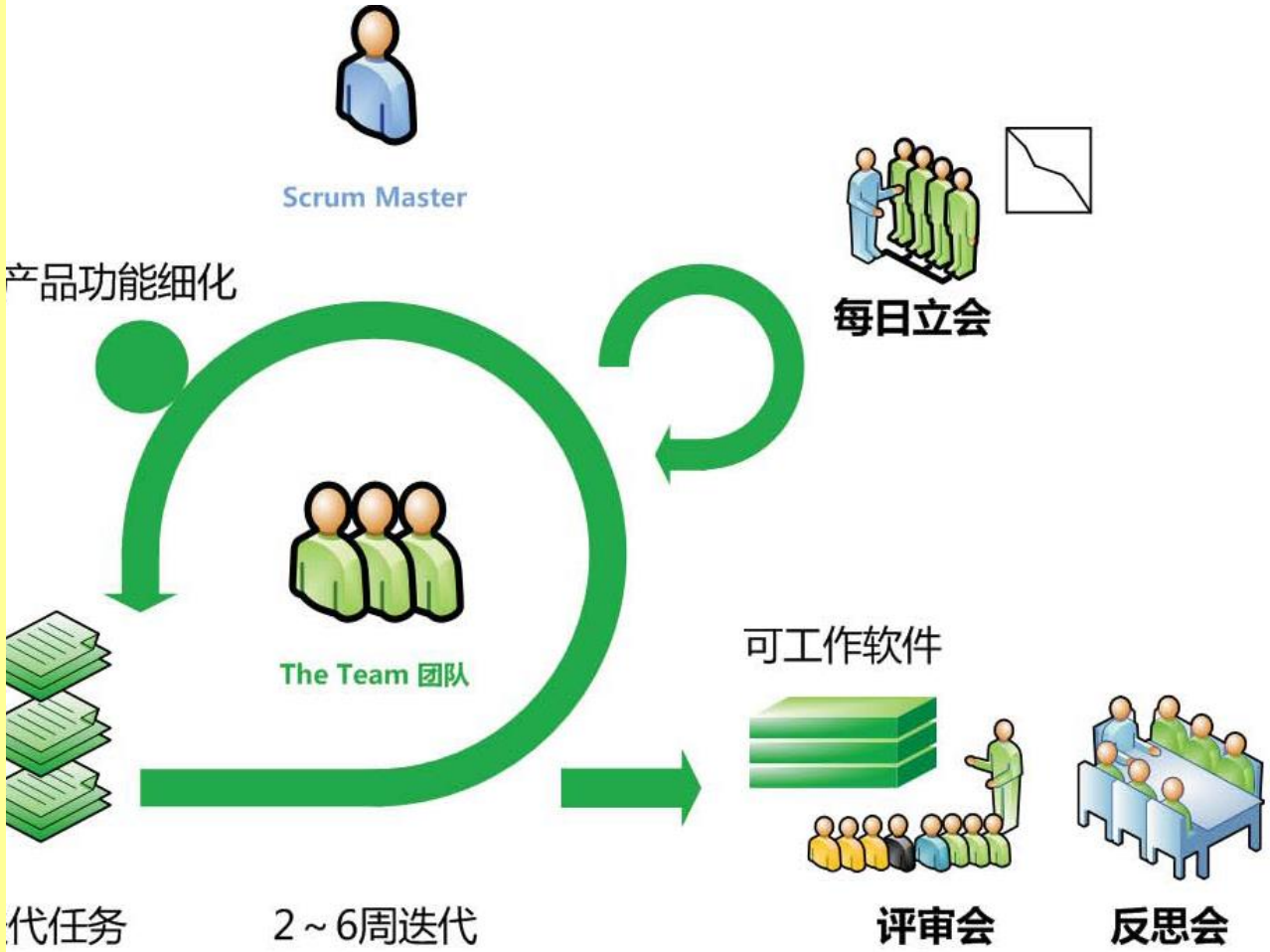
- 为达致 Sprint 目标，检视进展，发现问题和调整计划
- 不讨论和解决具体问题
- 只有团队成员，Scrum Master 和 Product Owner 可以说话

- 不是向ScrumMaster汇报状态，而是向所有组员的广播，属于自管理的一部分
  - 很多团队会在每日Scrum例会结束前一起看一下：团队到目前为止，离实现当前Sprint的目标，整体上情况如何?



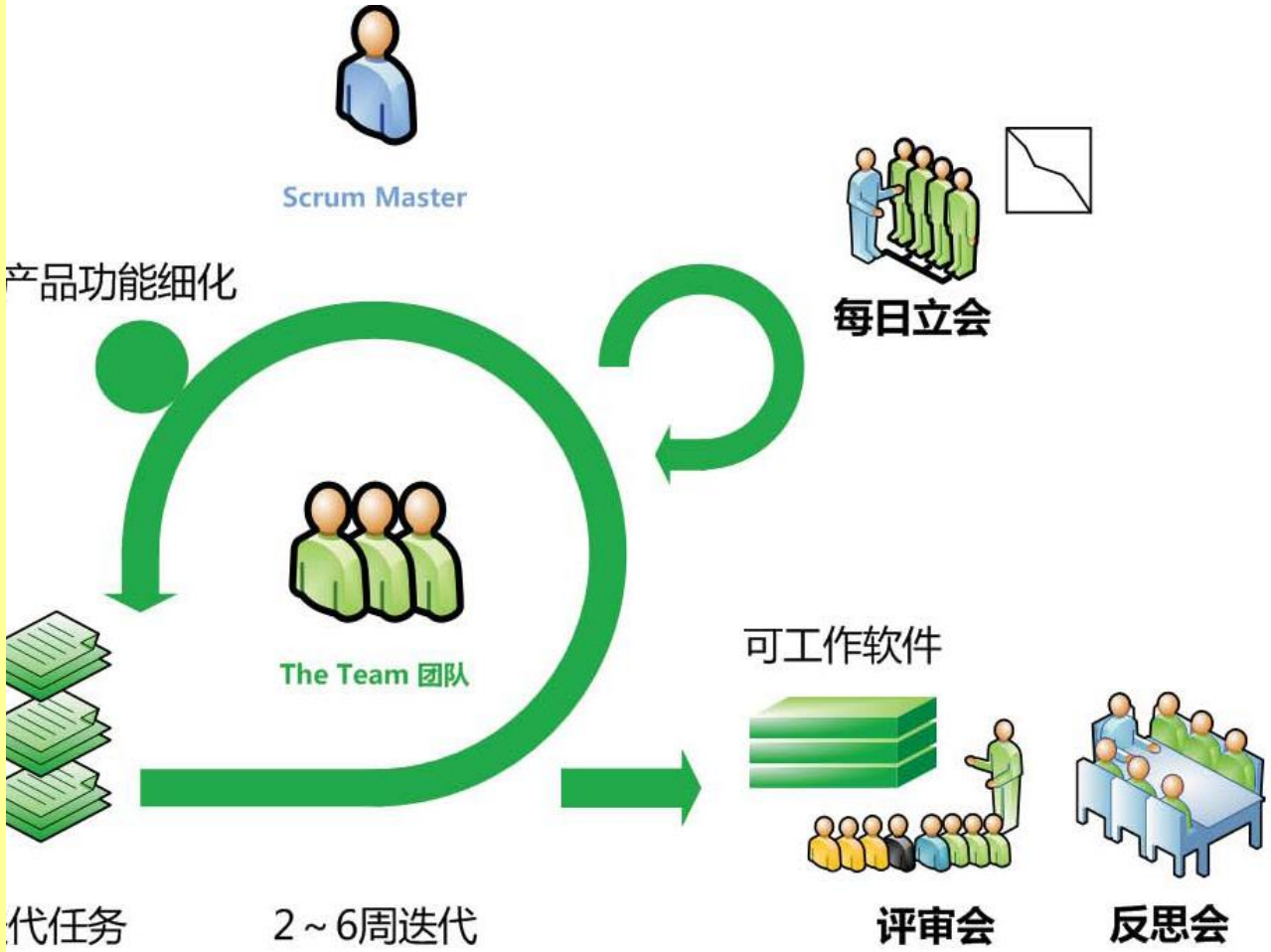
Scrum的5个“活动”之四：Sprint 评审

- 敏捷开发采用时间盒（Time Boxing）的方法，即限定时间而不限定范围。所以Sprint不会延期，因为在Sprint终点将放弃未完成的故事。
- 全Scrum团队参与，邀请所有干系人及感兴趣的人士
- 团队展示Sprint的成果
  - 通常以Demo新功能（及其依赖的架构）的形式，获取反馈和讨论要做的调整
  - 1个月的Sprint，评审展示最长4小时
  - 不要用幻灯片
- 产品负责人接受“完成”的工作及退回“未完成”的工作
  - 检视和调整产品和产品 Backlog
  - 评审的标准是整个故事是否已经达到交付标准，而不是从其中分解出来的任务完成了多少，因此若一个故事“差一点就完成了”也算没有完成。
  - 一般在Sprint计划会上设定每个故事的完成标准，如是否需要测试，是否需要考虑性能，是否需要说明文档等等。这些标准一般由项目组提前列好，每个故事只需要选中是否需要即可。
  - 评审会上发现的问题或改进将被累积到产品待开发项，也不会马上或在下一个Sprint中开发，而是由优先级排序决定何时开发。
- 常常发生很多故事都已经开始开发，但都差一点完成的现象。
  - 应按Sprint内的优先级逐条开发和交付故事。一般总是优先开发MoSCoW方法中必须完成和应该完成的故事，再考虑可以完成的故事。
  - 尽管有正式的评审会，但很多团队习惯在单个故事完成时，就让产品负责人进行单个故事评审，以确保交付时不会有“惊喜”。

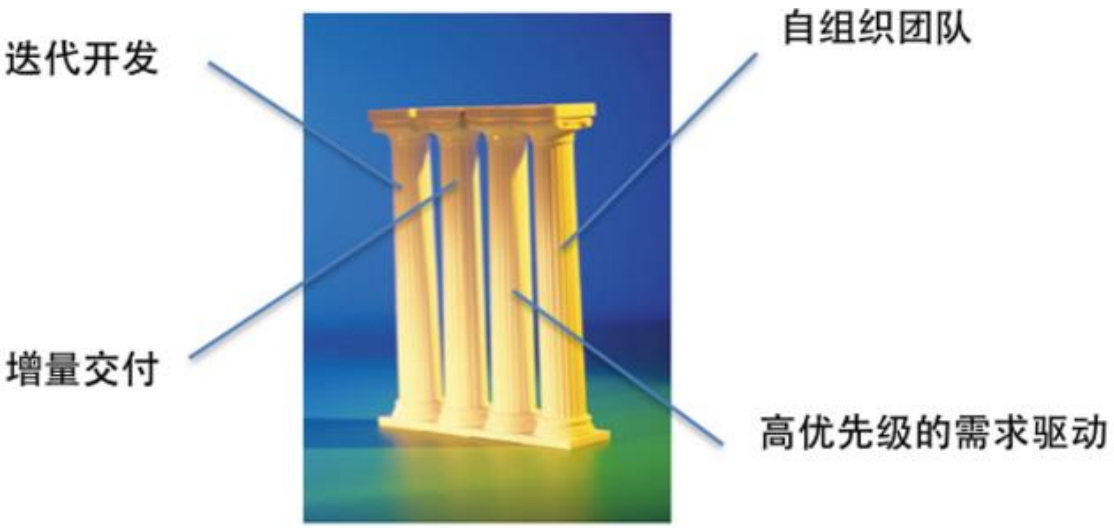


Scrum的5个“活动”之五：Sprint 回顾

- 在每个 Sprint 后召开简短的回顾会（反思会）
- 对我们如何工作进行检视和调整
  - 总结哪些事情做的好，哪些事情做的不好。
- 反思会上讨论三个问题：我们上个Sprint有哪些事情做的好希望继续，哪些事情做的不好希望改进，有何改进计划
  - 团队成员发言，列举最应该坚持的TOP1~3、最应该改进的TOP1~3
  - 团队成员就上述列举清单投票，选择最应该调整的TOP3
  - 投票结果作为下个Sprint的改进计划，其余项目留作观察
- 时间盒：
  - 1个月的Sprint最长3小时，通常30 - 60分钟
- 全Scrum团队参与
  - 欢迎其他感兴趣的受邀人士



Scrum的4大支柱



- 迭代开发
  - 在Scrum的开发模式下，我们将开发周期分成多个1-4周的迭代，每个迭代都交付一些增量的可工作的功能。迭代的长度是固定的，如果我们选择了1周的迭代，那么保持它的长度不要发生变化，在整个产品开发周期内每个迭代都是1周的长度。这里需要强调的是在每个迭代必须产出可工作的增量功能，而不是第一个迭代做需求、第二个迭代做设计、第三个迭代做代码。
- 增量交付
  - 增量是一个 Sprint 及以前所有 Sprint 中完成的所有产品代办事项列表条目的总和。 在 Sprint 的结尾,新的增量必须“完成”,这意味着它必须可用并且达到了 Scrum 团队 “完成”的定义的标准。无论产品负责人是否决定真正发布它,增量必须可用。增量是从用户的角度来描述的，它意味着从用户的角度可工作。
- 自组织团队
  - Scrum团队是一个自组织的团队，传统的命令与控制式的团队只有执行任务的权利，而自组织团队有权进行设计、计划和执行任务，自组织团队还需要自己监督和管理他们的工程过程和进度，自组织团队自己决定团队内如何开展工作，决定谁来做什么，即分工协作的方式。
- 高优先级的需求驱动
  - 在Scrum中，我们使用Product Backlog来管理需求，Product Backlog是一个需求的清单，Product Backlog中的需求是渐进明细的，Backlog当中的条目必须按照商业价值的高低排序。Scrum团队在开发需求的时候，从Backlog最上层的高优先级的需求开始开发。在Scrum中，只要有足够1-2个Sprint开发的细化了的高优先级的需求，我们就可以启动Sprint了，而不必等到所有的需求都细化之后。我们可以在开发期间通过Backlog的梳理来逐步的细化需求。

Scrum的价值观

Scrum Values　价值观

- Courage　勇气 – 有勇气做出承诺，履行承诺，接受别人的尊重
- Openness　开放 – Scrum 把项目中的一切开放给每个人看
- Focus　专注 – 把你的心思和能力都用到你承诺的工作上去
- Commitment　承诺 – 愿意对目标做出承诺
- Respect　尊重 – 每个人都有他独特的背景和经验

关于用户故事

- 用户故事是从用户的角度来描述用户渴望得到的功能。一个好的用户故事包括**三个要素**：
  - 1. 角色：谁要使用这个功能。
  - 2. 活动：需要完成什么样的功能。
  - 3. 商业价值：为什么需要这个功能，这个功能带来什么样的价值。
- 用户故事通常按照如下的格式来表达：
  - 英文： As a , I want to , so that .
  - 中文： 作为一个<角色>, 我想要<活动>, 以便于<商业价值>
- 举例：
  - 作为一个“网站管理员”，我想要“统计每天有多少人访问了我的网站”，以便于“我的赞助商了解我的网站会给他们带来什么收益。”
- Ron Jeffries的3个C：关于用户故事，Ron Jeffries用3个C来描述它：
  - 卡片（Card） – 用户故事一般写在小的记事卡片上。卡片上可能会写上故事的简短描述，工作量估算等。
  - 交谈（Conversation）- 用户故事背后的细节来源于和客户或者产品负责人的交流沟通。
  - 确认（Confirmation）- 通过验收测试确认用户故事被正确完成。

- 用户故事的六个特性- INVEST： INVEST = Independent, Negotiable, Valuable, Estimable, Small, Testable
  一个好的用户故事应该遵循**INVEST**原则。
  - 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。用户故事之间的依赖使得制定计划，确定优先级，工作量估算都变得很困难。通常我们可以通过组合用户故事和分解用户故事来减少依赖性。
  - 可协商性（Negotiable）： 一个用户故事卡片上只是对用户故事的一个简短的描述，不包括太多的细节。具体的细节在沟通阶段产出。一个用户故事卡带有了太多的细节，实际上限制了和用户的沟通。
  - 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）。一个让用户故事有价值的好方法是让客户来写下它们。
  - 可以估算性（Estimable）： 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。让故事难以估计的问题来自：对于领域知识的缺乏（这种情况下需要更多的沟通），或者故事太大了（这时需要把故事切分成小些的）。
  - 短小（Small）： 一个好的故事在工作量上要尽量短小，最好不要超过10个典型人/天的工作量，至少要确保的是在一个Sprint中能够完成。用户故事越大，在安排计划，工作量估算等方面的风险就会越大。
  - 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。如果一个用户故事不能够测试，那么你就无法知道它什么时候可以完成。
    一个不可测试的用户故事例子：软件应该是易于使用的。

关于估算

共同估算：

在估算前不应该指定谁将开发这个故事，而是以接收故事的小组为单位集体估算，每个人提出自己的看法，大家综合。这样才能以集体智慧完成任务。

在估算全程，产品负责人不能离开，因为很多估算差异问题来自于“做什么，做到什么地步”，产品负责人需要予以解答，而不是让团队按自己的理解去做。

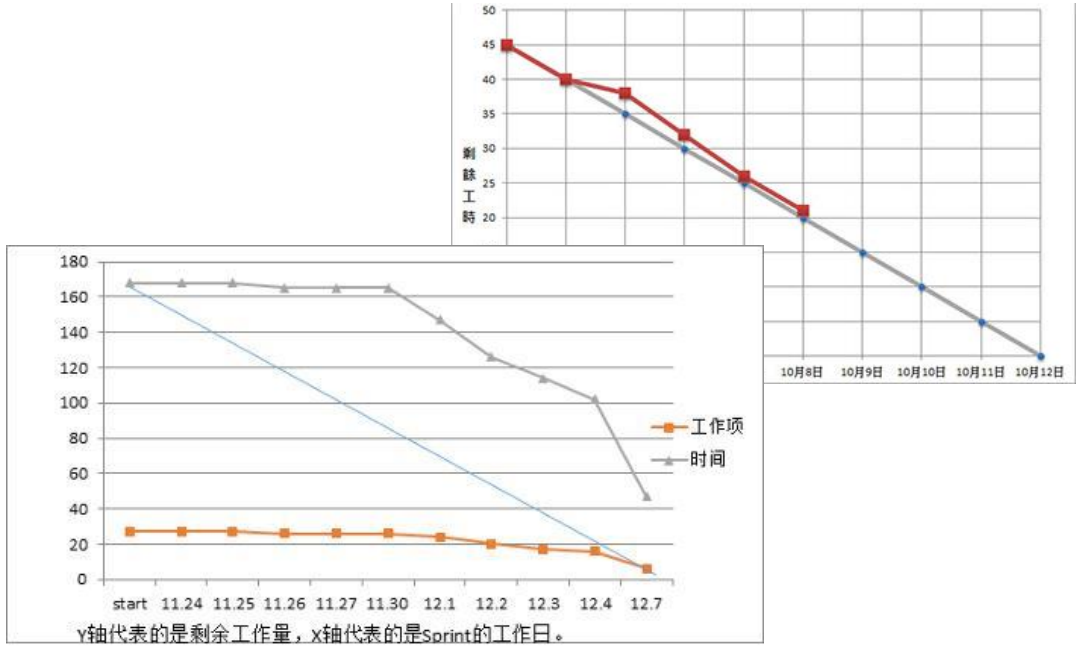
共同估算的目的不是一个数字上的统一，而是用集体智慧和知识对“做什么，怎么做”达成共识。

共同估算是共同跟进的基础，若不能共同估算，则后面的“每日立会”几乎不可能正常进行，因为大家只会关心自己曾经一起分析、思考、提问、设计乃至争论过的任务进展的怎样了，是不是和自己想象的一样。

共同估算的最佳方法是“扑克牌估算”，这看似很像一个小游戏，但却是Delphi估算的一个快速方法，同时实现了匿名性与高效性。

关于“燃尽图”

- 燃尽图Burdown Chart也叫燃烧图，是最常见的敏捷度量。
- 全称应该是“总剩余时间的燃尽图”，意指本次迭代中所有故事（或拆分的任务）的剩余度量的总和，随日期的变化而逐日递减的图。
- 每日更新，通常发生在每日站会之后



左图中的燃尽图是属于比较完美的燃尽图。

实际上每个团队完成迭代的过程差别很大，常见的情况包括：

- **先鼓起后落下**
  - 原因是计划会以常常漏掉一些事情，所以开工后不但不燃尽，还发现了很多新的任务。
- **先完美燃烧，然后突然停止燃烧**
  - 一种很常见的情况，如果任务划分太粗，比如长达10天，很容易“做了1天，剩9天；做了1天，剩8天；……到剩2～3天的时候，哎呀，好像搞不定了”。
- **缓慢燃烧，至时间盒耗尽时仍剩下一堆没完成的任务**
  - 之前提到过敏捷开发的MoSCoW方法，有些故事是次要的“可以不做的”，所以这种燃烧图也很常见；但是常常有团队没有使用MoSCoW方法，只是被动地发现有些故事没有完成。

附注

Scrum相关工具

- iceScrum
- 禅道
- Tuleap
- Jira + Agile
- ……

Scrum参考文档

- Scrum 指南
- Scrum精髓_敏捷转型指南
- 火星人敏捷开发手册
- 硝烟中的Scrum和XP