2-2 敏捷过程

2019年4月13日 15:00

1 敏捷过程模型

惯例方法的问题?

- 为提高软件产品质量，不断增加过程管理控制
  - 时间↑成本↑质量?
- 庞大的重型的过程方法是否有效?
- 变化是软件的本质，软件开发必须适应变化
- 传统的惯例模型忽视了软件开发中人的弱点，敏捷开发则是重视人的作用，以人为本的软件开发方法
  - 人的弱点：不能一贯地、连续地做同一件事情

敏捷过程模型(agileprocessmodel)

- 开发过程中的“变化”是无处不在的，也是不可避免的;
- 在实际项目中，很难预测需求和系统何时以及如何发生变化;
- 对开发者来说，应将变化的意识贯穿在每一项开发活动中;

17位编程大师共同发布《敏捷软件开发宣言》:

- “人”以及“人与人的互动”胜于“过程”和”工具”
- 可运行的软件胜于面面俱到的文档
- 客户合作胜于合同谈判
- 响应变化胜于遵循计划

什么是敏捷?

- 有效地（快速、适应）响应变化
  - 参与者之间有效沟通

    - 使客户加入团队
    - 项目计划灵活调整
- 目标：快速、增量地发布软件

敏捷假设

- 提前预测哪些需求是稳定的和哪些需求会变化非常困难；同样，预测项目进行中客户优先级的变更也很困难
- 对很多软件来说，设计和构建是交错进行的。通过构建验证之前，很难估计应该设计到什么程度
- 从制定计划的角度来看，分析、设计、构建和测试并不像我们所设想的那么容易预测

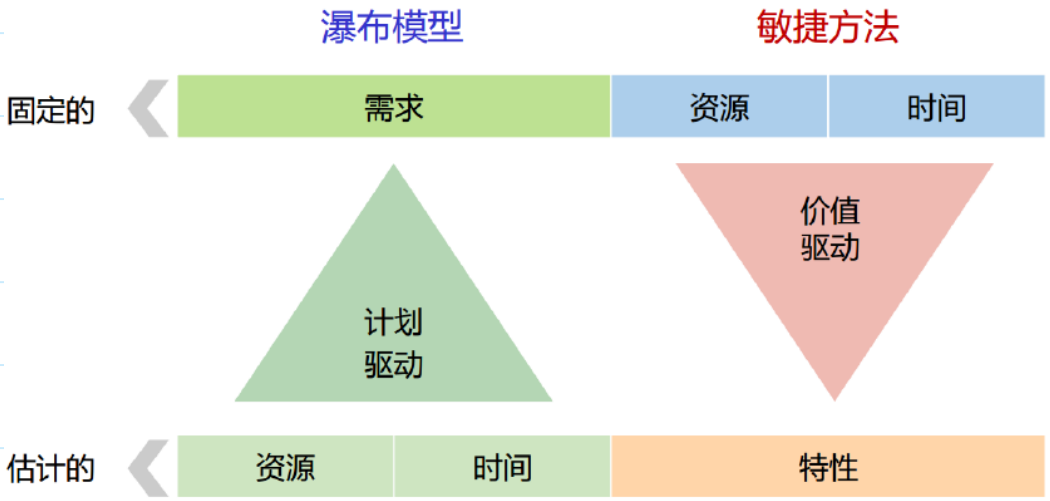
敏捷宣言遵循的原则

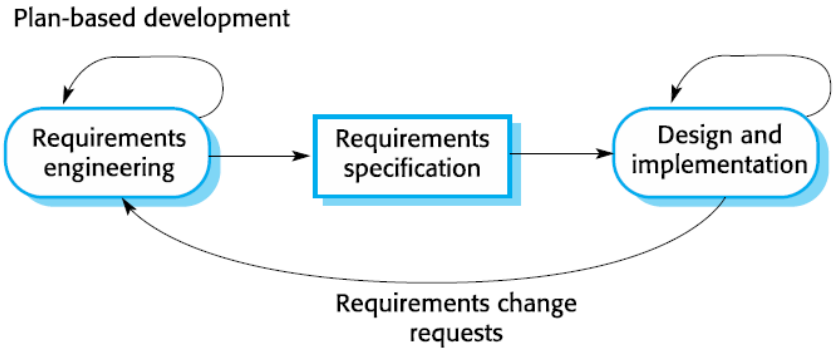
- 准则1：我们的最高目标是，通过尽早和持续地交付有价值的软件来满足客户
- 准则2：欢迎对需求提出变更——即使是在项目开发后期。要善于利用需求变更，帮助客户获得竞争优势。
- 准则3：要不断交付可用的软件，周期从几周到几个月不等，且越短越好。
- 准则4：项目过程中，业务人员与开发人员必须在一起工作。
- 准则5：要善于激励项目人员，给他们以所需要的环境和支持，并相信他们能够完成任务。
- 准则6：无论是团队内还是团队间，最有效的沟通方法是面对面的交谈。
- 准则7：可用的软件是衡量进度的主要指标。
- 准则8：敏捷过程提倡可持续的开发。项目方、开发人员和用户应该能够保持恒久稳定的进展速度。
- 准则9：对技术的精益求精以及对设计的不断完善将提升敏捷性。
- 准则10：要做到简洁，即尽最大可能减少不必要的工作，这是一门艺术。
- 准则11：最佳的架构、需求和设计出自于自组织的团队。
- 准则12：团队要定期反省如何能够做到更有效，并相应地调整团队的行为。

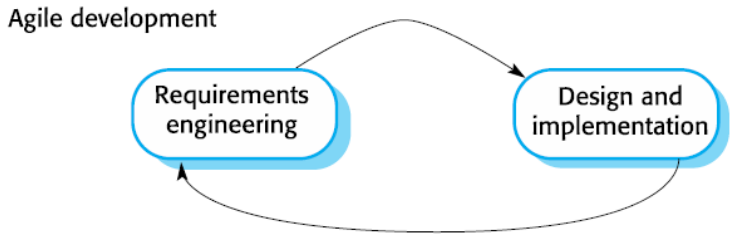
归纳：小步快跑，及时反馈

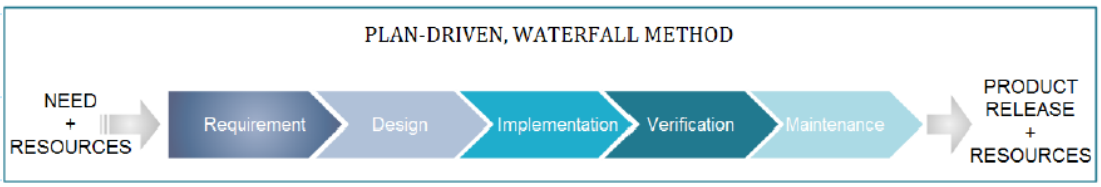
- 不强调文档，转向强调可运行的软件片段
- 开发者与客户之间频繁沟通
- 快速开发，快速反馈，快速修改，增量交付
- 连续不断的短周期迭代
- 不看重形式和工具，看重“人”和内容，保持简洁。
- 本质：以快速的增量和迭代方式进行软件开发

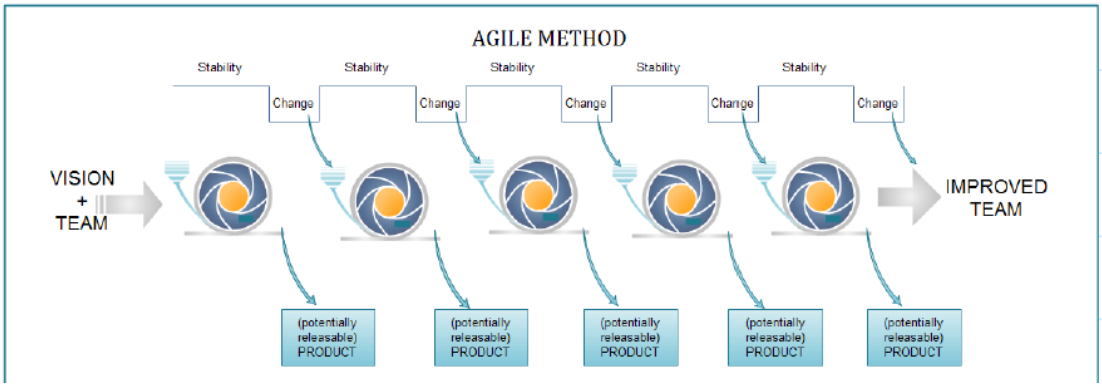
敏捷方法vs传统方法


瀑布模型
敏捷方法
固定的
需求
资源
时间
计划
驱动
价值
驱动
估计的
资源
时间
特性

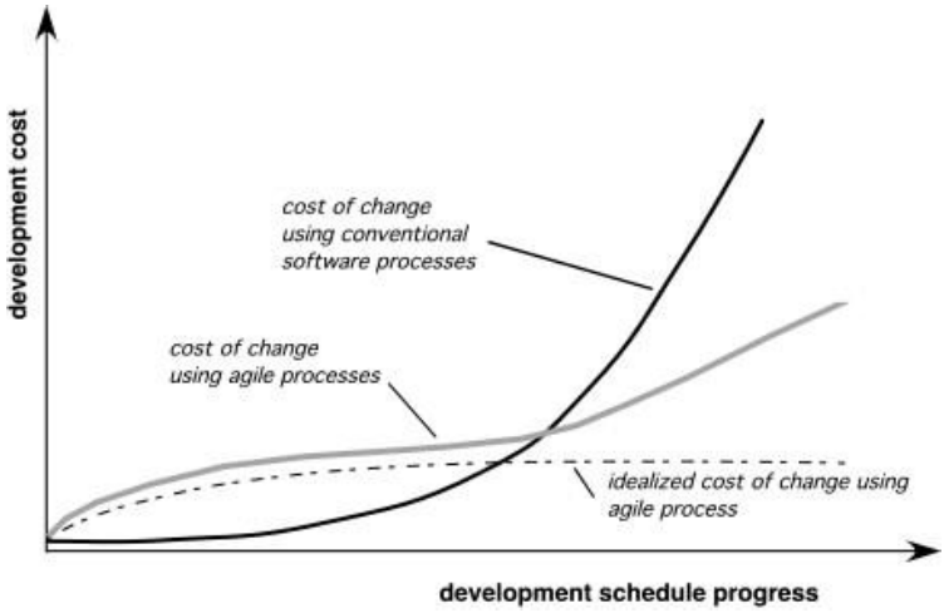

Plan-based development
Requirements engineering
Requirements specification
Design and implementation
Requirements change requests

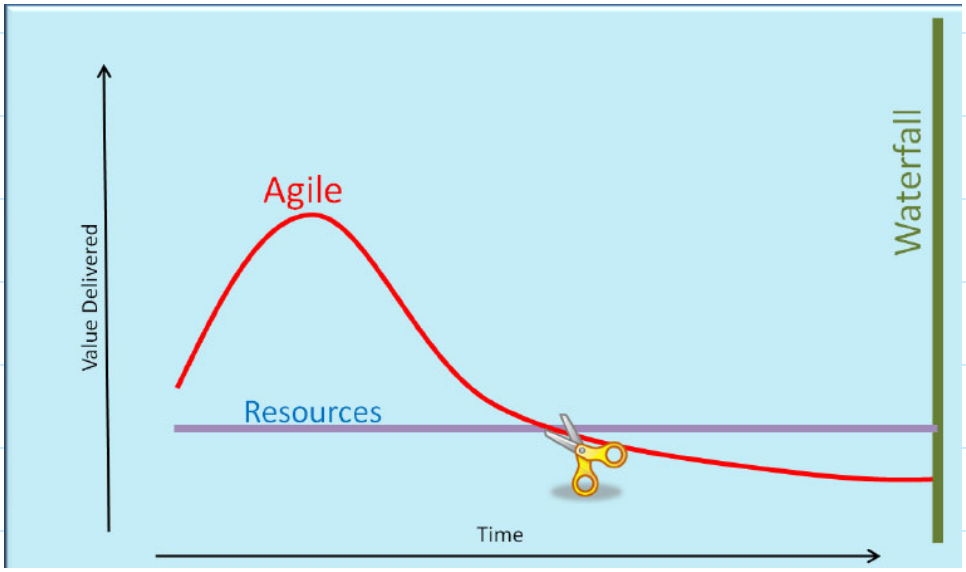

Agile development
Requirements engineering
Design and implementation


PLAN-DRIVEN, WATERFALL METHOD
NEED + RESOURCES
Requirement
Design
Implementation
Verification
Maintenance
PRODUCT RELEASE + RESOURCES


AGILE METHOD
Stability
Change
VISION + TEAM
IMPROVED TEAM
(potentially releasable) PRODUCT

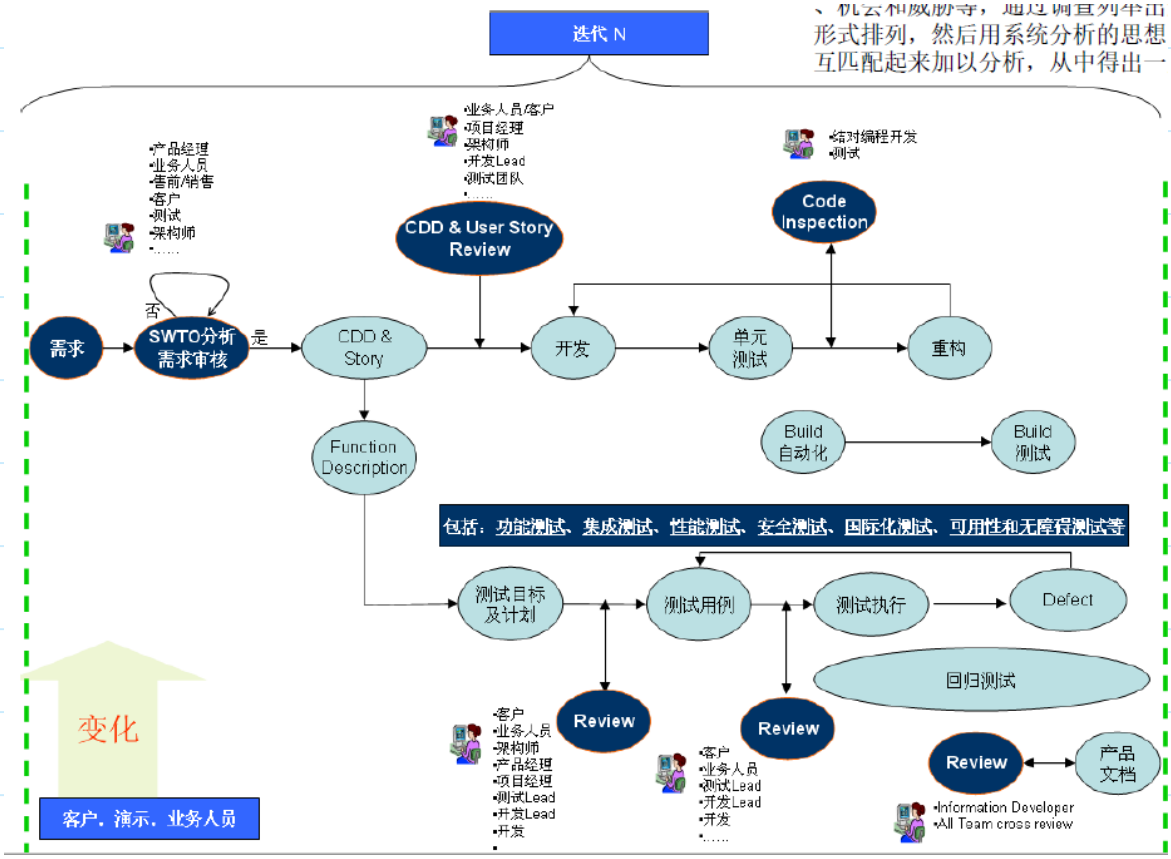
- 成本变化曲线

- 
development cost
cost of change
using conventional
software processes
cost of change
using agile processes
idealized cost of change using
agile process
development schedule progress

- 交付价值
  - 
Value Delivered
Agile
Waterfall
Resources
Time

敏捷过程：细节

形式排列，然后用系统分析的思想
互匹配起来加以分析，从中得出一



- 注：SWOT分析法又称为态势分析法：
  - 优势（Strength）、劣势（Weakness）、机会（Opportunity）、威胁（Threat）
  - 将与研究对象密切相关的各种主要内部优势、劣势、机会和威胁等，通过调查列举出来，并依照矩阵形式排列，然后用系统分析的思想，把各种因素相互匹配起来加以分析，从中得出一系列相应的结论

为何“敏捷”？

- 最初的软件(196070年代)的顾客都是大型研究机构、军方等，他们需要软件系统来搞科学计算、军方项目、登月项目等，这些系统相当庞大，对准确度要求相当高。
- 1980-90年代，软件进入了桌面软件的时代，开发的周期明显缩短，各种新的方法开始进入实用阶段。但是软件发布的媒介还是CD、DVD，做好一个发布需要较大的经济投入，不能频繁更新版本。
- 互联网时代，大部分的服务是通过网络服务器端实现，在客户端有各种方便的推送(push)渠道。
  - 由于网络的转播速度和广度，知识的获取更加容易，很多软件服务可以由一个小团队来实现。
  - 同时技术更新的速度在加快，那种一个大型团队用一个固定技术开发23年再发布的时代已经过去了。
  - 用户需求的变化也在加快，开发流程必须跟上这些快速变化的节奏

敏捷过程中最重要的因素：人

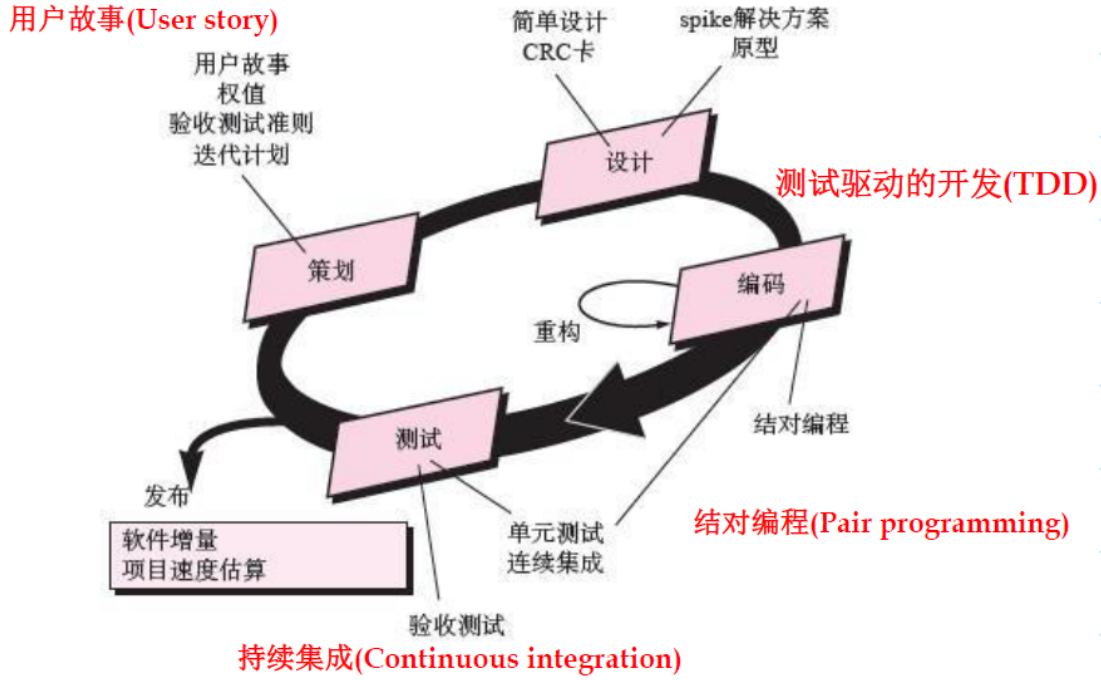
- 敏捷开发关注“人的因素”的重要性：构造可以满足人员及团队需求的过程模型
- 基本能力

- 共同目标
- 精诚合作
- 决策力
- 模糊问题解决能力
- 相互信任和尊重
- 自我组织

目前广泛使用的敏捷开发方法论

- XP(eXtremeProgramming)(KentBeck)极限编程
- Scrum(KenSchwaber)
- DSDM(DSDMconsortium)动态系统开发方法
- AdaptiveSoftwareDevelopment(JimHighSmith)自适应软件开发
- Crystal(AlistairCockburn)
- FDD(FeatureDrivenDevelopment)特征驱动开发
- PragmaticProgramming实用编程
- AUP(AgileUnifiedProcess)敏捷统一过程

2 敏捷开发方法之一：极限编程XP(eXtremeProgramming)--一种最广泛应用的敏捷开发方法

- 

XP Planning：计划阶段

- 倾听顾客陈述，形成一组“用户故事(userstories)”，描述其输出、特性、功能等。
- 按照价值或风险排序：顾客为每个用户故事指定优先级(Priority)；
- XP团队评估各用户故事，为其指定成本(Cost，开发周数)，若超过3周，则拆分；
- 将若干个用户故事指定为下一次发布的增量，确定发布日期：

    - 所有用户故事
    - 优先级高的用户故事
    - 风险高的用户故事
- 规划整体进度(projectvelocity)：以怎样的速度开展项目；
- 顾客可以在开发过程中扩展新故事、去除原有故事、改变优先级、拆分等。

XP Design：设计阶段

- 遵循KIS原则(KeepItSimple)；
- 设计模型：面向对象方法，CRC卡片(ClassResponsibilityCollaborator)；
- 遇到困难问题，创建“SpikeSolutions”(原型)；
- 对设计方案不断重构(Refactoring)
    - 遵循用户故事的外特性要求
    - 改善内部结构
    - 消除bug
    - 提高效率
    - 提高易读性

XP Coding&Testing：编码与测试阶段

- XP Coding
    - 在编码之前，根据用户故事设计单元测试用例；
    - 结对编程(Pairprogramming)：两人一起编程，实时讨论、实时评审；
    - 测试驱动的开发(TDD)：先写测试用例，再写代码；
- XP Testing
    - 自动化单元测试(Unittest)；
    - 持续集成(ContinuousIntegration)；
    - 持续进行回归测试(Regressiontest)；
    - 验收测试(Acceptancetest)。

结对编程(PairProgramming)

- 两个程序员肩并肩地、平等地、互补地进行开发工作。
    - 并排坐在一台电脑前，面对同一个显示器，使用同一个键盘，同一个鼠标一起工作。
    - 一起分析，一起设计，一起写测试用例，一起编码，一起单元测试，一起集成测试，一起写文档等。
- 特点：在高速度中完成任务，任务有较高的技术要求，任务失败的代价很高
- 驾驶员(Driver)：控制键盘输入的人。
    - 写设计文档，进行编码和单元测试等XP开发流程。

- 领航员(Navigator)：起到领航、提醒的作用。
  - 审阅驾驶员的文档、驾驶员对编码等开发流程的执行；考虑单元测试的覆盖程度；是否需要和如何重构；帮助驾驶员解决具体的技术问题。
- 驾驶员和领航员不断轮换角色，不宜连续工作超过一小时。领航员要控制时间。
- 主动参与：任何一个任务都首先是两个人的责任，也是所有人的责任；没有“我的代码”、“你的代码”或“她的代码”，只有“我们的代码”。
- 只有水平上的差距，没有级别上的差异：尽管可能大家的级别资历不同，但不管在分析、设计或编码上，双方都拥有平等的决策权利。
- 每人在各自独立设计、实现软件的过程中不免要犯这样那样的错误。在结对编程中，因为有随时的复审和交流，程序各方面的质量取决于一对程序员中各方面水平较高的那一位。这样，程序中的错误就会少得多，程序的初始质量会高很多，这样会省下很多以后修改、测试的时间。
  - 在开发层次，结对编程能提供更好的设计质量和代码质量，两人合作能有更强的解决问题的能力。
  - 对开发人员自身来说，结对工作能带来更多的信心，高质量的产出能带来更高的满足感。
  - 在心理上，当有另一个人在你身边和你紧密配合，做同样一件事情的时候，你不好意思开小差，也不好意思糊弄。
  - 在企业管理层次上，结对能更有效地交流，相互学习和传递经验，能更好地处理人员流动。因为一个人的知识已经被其他人共享。

小结：为何称为“极限”编程

极限：把某件事情做到极致

-

如果……	发挥到极限就变成……
了解顾客的需求很重要	每时每刻都有客户在身边，时时了解需求
测试/单元测试能帮助提高质量	那就先写单元测试，从测试开始写程序——TDD
代码复审可以找到错误	从一开始就处于“复审”状态——结对编程
计划没有变化快	那就别做详细的设计，做频繁的增量开发，重构和频繁地发布
其他好方法……	发挥到极限的做法……

关于XP的一些反对意见

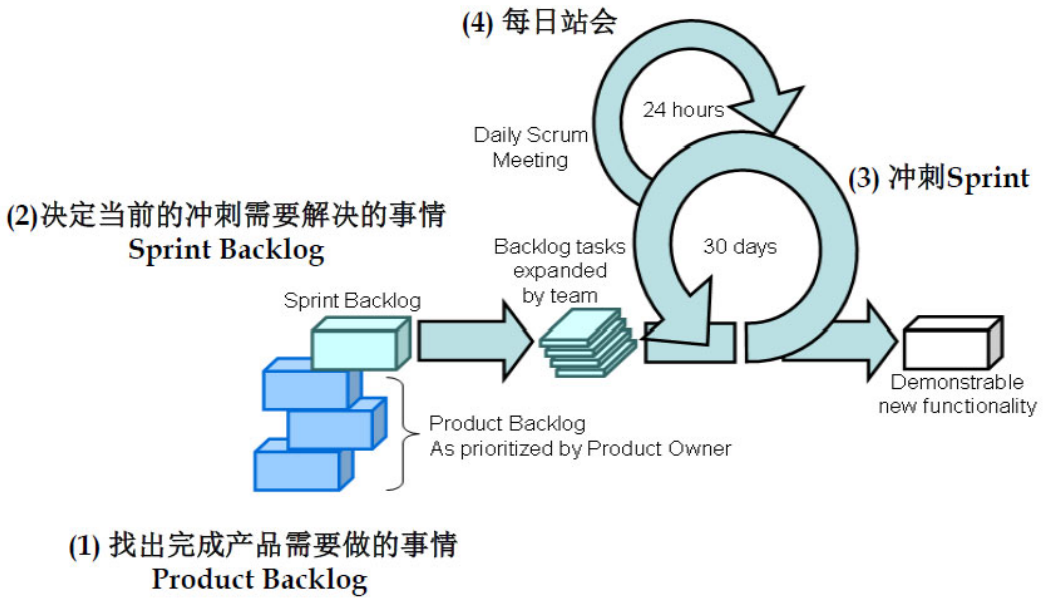
- 需求反复无常
- 需求冲突
- 需求表达非正式
- 缺少正式的设计

3 敏捷开发方法之二：Scrum

- 一个敏捷开发框架：增量/迭代的开发过程；
- 1990年代由Schwaber&Beedle提出并实践；

- 整个开发过程由若干个短的迭代周期组成，一个短的迭代周期称为一个Sprint，每个Sprint的建议长度是2到4周。
- 使用产品Backlog来管理需求，是一个按照商业价值排序的需求列表，列表条目的体现形式通常为用户故事。
- 总是先开发对客户具有较高价值的需求。
- 在Sprint中，Scrum团队从产品Backlog中挑选最高优先级的需求进行开发。挑选的需求在Sprint计划会议上经过讨论、分析和估算得到相应的任务列表(backlog)。
- 在每个迭代结束时，Scrum团队将递交潜在可交付的产品增量。

Scrum的基本过程

- 

- ProductBacklog
  - ProductBacklog可译为：
    - 积压的工作
    - 待解决的问题
    - 产品订单
  - 每项工作以“天”为单位估计

Scrum中的角色

- 猪和鸡合伙开餐馆的笑话
  - 一天，一头猪和一只鸡在路上散步。鸡对猪说：“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，那你准备给餐馆起什么名字呢？”鸡想了想说：“叫‘火腿和鸡蛋’怎么样？”“那可不行”，猪说：“我把自己全搭进去了，而你只是参与而已。”
- “猪”角色
  - 产品负责人(ProductOwner)：确定产品的功能，负责维护产品发布的内容、优先级、交付时

间、产品投资报酬率ROI；验收结果；
    - ScrumMaster：团队leader，保证开发过程按计划进行；组织每日站会、Sprint计划会议、Sprint评审会议和Sprint回顾会议；通过外在/内在协调，确保团队资源完全可被利用并且全部是高产出的；
    - Scrum团队：在每个Sprint中将产品Backlog中的条目转化成为潜在可交付的功能增量；规模在59人；具备交付产品增量所需要的各种技能；
- “鸡”角色：

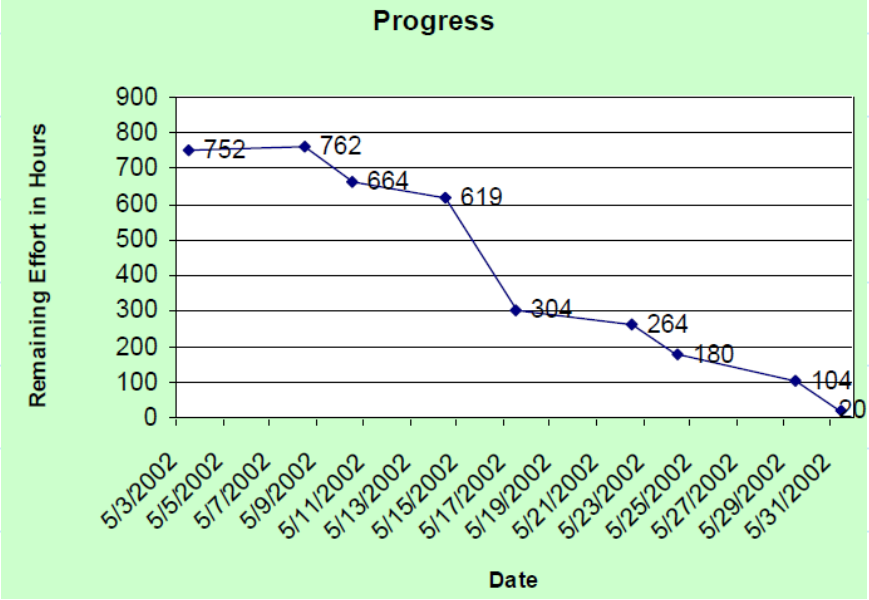
“鸡”并不是实际Scrum过程的一部分，但是必须考虑他们。敏捷方法的一个重要方面是使得用户和利益相关者参与到过程中的实践。参与每一个冲刺的评审和计划，并提供反馈对于这些人来说是非常重要的。

- 用户：软件是为了人而开发的。有人说，“假如森林里有一棵树倒下了，但没有被人听到，那么它算是发出了声音吗？”同样地，人们可以说，“假如软件没有被使用，那么它算是被开发出来了么？”
- 利益相关者（客户，提供商）：影响项目成功的人，但只直接参与冲刺评审过程；
- 经理：为产品开发团体搭建环境的人。

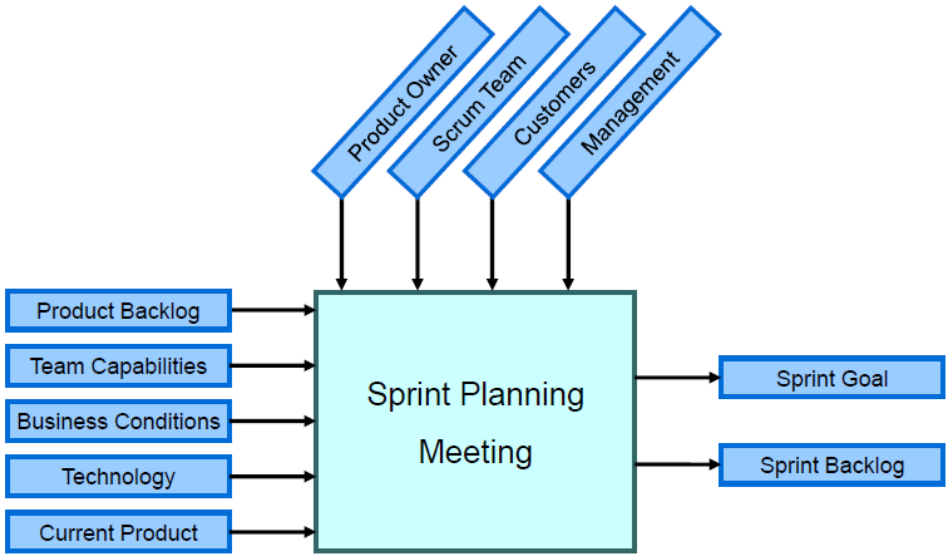
Scrum中的六项活动

- Sprint(冲刺)：代表一个2-4周的迭代；
- 发布计划会议(ReleasePlanningMeeting)->ProductBacklog
- Sprint计划会议(SprintPlanningMeeting)->SprintBacklog
- 每日站会(DailyScrumMeeting)
- Sprint评审会(SprintReviewMeeting)
- Sprint回顾会议(SprintRetrospectiveMeeting)

Sprint Burndown Chart 燃尽图

- 

Sprint计划会议

- 

每日站会

- 团队成员站着开会——强迫在短时间(15分钟)内高效率讨论问题，保持注意力集中；
- 强迫每个人向同伴报告进度，迫使大家把问题摆在明面上：
  - 我昨天做了什么(Whathaveyoudonesinceyesterday?)
  - 我今天要做什么(Whatareyouplanningtodotoday?)
  - 我碰到了哪些问题(Doyouhaveanyproblemsthatwouldpreventyoufromaccomplishingyourgoal?)
- 启动每日构建，让大家每天都能看到一个逐渐完善的版本。
- 用简明的图表展现整个项目的进度，这个图最好放在大家工作的环境中，或者每天传达给各个成员。

归纳：敏捷开发过程的最佳实践(XP&Scrum)

- 用户故事(UserStory)
- 结对编程(PairProgramming)
- 测试驱动的开发(TDD)
- 持续集成(ContinuousIntegration)
- 验收测试(AcceptanceTesting)
- 冲刺/迭代(Sprint/Iteration)
- 产品清单/冲刺清单(Productbacklog/Springbacklog)
- 燃尽图(Burndownchart)
- 每日站会(DailyStandup)

4 各种软件开发过程模型的比较

一句话归纳各类过程模型

- 瀑布模型：将全部需求以整体方式向前推进，无迭代；——基本模型
- 增量模型：将需求分成多份，串行推进，无迭代；——串行的瀑布
- RAD模型：将需求分成多份，并行推进，无迭代；——并行的瀑布
- 原型模型：迭代；——基本模型
- 螺旋模型：按瀑布阶段划分，各阶段分别迭代(原型+风险分析),——原型+瀑布
- 敏捷模型：将需求分成尽量小的碎片，以碎片为单位进行高速迭代,——增量+迭代

各类过程模型

客观因素\最适用方式	**敏捷** (Agile)	**计划驱动** (Plan-driven)	**形式化的开发方法** (Formal Method)
产品可靠性要求	不高，容忍经常出错	必须有较高可靠性	有极高的可靠性和质量要求
需求变化	经常变化	不经常变化	固定的需求，需求可以建模
团队人员数量	不多	较多	不多
人员经验	有资深程序员带队	以中层技术人员为主	资深专家
公司文化	鼓励变化，行业充满变数	崇尚秩序，按时交付	精益求精
实际的例子	**写一个微博网站;**	**开发下一版本的办公软件; 给商业用户开发软件**	**开发底层正则表达式解析模块; 科学计算; 复杂系统的核心组件**
用错方式的后果	用敏捷的方法开发登月火箭控制程序，前N 批宇航员都挂了。	用敏捷方法，商业用户未必受得了两周一次更新的频率。	敏捷方法的大部分招数都和这类用户无关，用户关心的是：把可靠性提高到 99.99%，不要让微小的错误把系统搞崩溃!

	商业系统	使命攸关的系统	性命攸关的嵌入式系统
典型应用	Internet站点 Intranet站点 游戏 管理信息系统（MIS） 企业资源计划（ERP）	嵌入式软件 游戏 Internet站点 盒装软件 软件工具	航空软件 嵌入式软件 医疗设备 操作系统 盒装软件
生命周期模型	敏捷开发、渐进原型	分阶段交付 渐进交付 螺旋型开发	分阶段交付 螺旋形开发 渐进交付
计划与管理	增量式项目计划 随需测试与QA计划 非正式的变更控制	基本的预先计划 基本的测试计划 随需QA计划 正式的变更控制	充分的预先计划 充分的测试计划 充分的QA计划 严格的变更控制
需求	非形式化的需求规格	半形式化的需求规格 随需的需求评审	形式化的需求规格 形式化的需求检查
设计	设计与编码是结合的	架构设计 非形式化的详细设计 随需的设计评审	架构设计 形式化的架构检查 形式化的详细设计 形式化的详细设计检查
构建	结对编程或独立编码 非正式的check-in手续或没有check-in手续	结对编程或独立编码 非正式的check-in手续 随需代码评审	结对编程或独立编码 正式的check-in手续 正式的代码检查
测试与QA	开发者测试自己的代码 测试先行开发 很少或没有测试（由单独的测试小组来做）	开发者测试自己的代码 测试先行开发 单独的测试小组	开发者测试自己的代码 测试先行开发 单独的测试小组 单独的QA小组
部署	非正式的部署过程	正式的部署过程	正式的部署过程

5 敏捷案例分析

- 公司有个项目组作的是一个网游物品交易平台。
- 该平台是典型的互联网项目，在开工的时候客户对功能需求还不明确，但需要快速推出抢占市场，正是最适合敏捷过程的项目。
- 在项目伊始，商务分析师和客户做了深入的谈话，了解他的商业构想，他的盈利模式，搞清楚宏观的结构，然后思考并整理获得的结果，花12天时间将客户需求大略整理为几十个用户故事。
- 这些用户故事并不完善，不足以做好整个系统。但对于开始项目的前部分工作，已经足够了。
- 敏捷方法希望快速交付可用的软件，实现软件的快速交付是通过迭代来完成。
- 在迭代开始前，由一组有经验的开发人员大致评估一下用户故事，标记出不同的难度和风险，并提出问题供商务分析师来获得更详细的信息，商务分析师会和相关涉众去讨论。
- 然后商务分析师将推荐优先级最高的一组用户故事给客户来挑选，客户可以选择这些用户故事，或者指出从他的视角看到的优先级更高的用户故事。这些将成为下一个迭代的内容。
- 项目经理和客户看到每个迭代交付的可运行的软件后或者得到用户反馈后，常常会有新的想法冒出来。有些想法是好的，有些想法就属于看到别家网站有这个功能，不假思索的提出的功能。
- 这些不同的需求都需要经过认真的分析，找出哪些是值得我们立即考虑的、哪些是不用急迫的去实现的。
- 有一次和客户谈话时，他说到希望增加拍卖功能。那么，我们为什么需要拍卖呢？客户说希望让用户拍卖物品以获得最高价格。经过考虑，我们发现网游物品的实时性和唯一性决定了系统不适合使用拍卖机制。拍卖的时效性无法满足实时交易的要求，因此，用户最终放弃了这个特性。
- 另一次，客服人员提出增加一个查询用户交易的功能，而此时我们有其他更加重要的功能需要先去

考虑，查询用户交易功能可以由技术人员临时通过数据库直接代为查询，因为项目运营初期交易不是很多，暂时还不需要专门的后台功能来支持客服的工作。所以把这个需求卡片一直贴在墙壁上，始终没有排到最高的优先级。

- 用户故事的跟踪和管理是由项目经理来进行：每个迭代跟踪卡片的进展，是否已经开始实现？是否已经完成代码开发？是否已经开始功能测试？
- 不同的卡片在迭代前都会评估为不同的大小，一般分为大中小三级。
- 每个迭代内分析好恰好足够下一个迭代开发的需求，就是商务分析师在每个迭代的主要工作内容。商务分析师的需求分析工作在上一个迭代完成，包括需求的了解、分析、评估和排列优先级。
- 在每个迭代开始的时候，由商务分析师主持召开迭代计划会议，在会议上向所有的程序员解释这个迭代要完成的用户故事，然后由程序员自由提问，直到他们能够获得足够开始实现该功能的信息。
- 在程序员完成一个用户故事后，商务分析师还要来代表客户做功能验收测试，查看是否完成了预计的功能，是否有程序员还没有想到的异常情况。如果存在问题需要退回给程序员继续完成。这在一定程度上保证了系统完成的需求不偏离客户的要求。