Agile—敏捷的思想与实践

世紀鼎利

李晗

Fleet 项目组

Outline

- 敏捷的基本理念介绍
- 敏捷的主要实践方法：Scrum + XP
- Scrum的部分实践方法介绍
- 相关讨论

世紀鼎利

敏捷开发现状及发展

- **敏捷的历史**
  - 简单的说，敏 捷开发是一种以人为核心、以迭代的方式循序渐进完成最终目标的软件开发方法。
  - 2001年《敏捷宣言》发布，经过近十年的发展，敏捷开发已经从最初的概念走向实践，敏捷开发方法也经过不断的迭代在人们的实践中走向普及。

- **业界的应用现状**
  - 国际：（Google、Yahoo、IBM、Microsoft、Siemens、Motorola, SAP, Cisco, GE）

    不仅在公司内部开发过程中使用，而且有了相关产品
  - 国内： （腾讯、华为、阿里巴巴）

    敏捷中国大会（AgileChina）已经举办四届（2006-2009）

世紀鼎利

软件开发的重要趋势

- 云计算
- 以**Web**为平台
- 并行计算
- 设备和用户界面的多样化
- 敏捷开发过程
- 分布式开发

-- **S.Somasegar**（微软软件开发者部门的全球高级副总裁）
2010.2.23《Key Software Development Trends》

http://blogs.msdn.com/somasegar/archive/2010/02/23/key-software-development-trends.aspx

世紀鼎利

敏捷宣言 – 四荣四耻

2001年2月11日到13日，17位软件开发领域的领军人物聚集在美国犹他州的滑雪胜地雪鸟（Snowbird）雪场。经过两天的讨论，“敏捷”（Agile）这个词为全体聚会者所接受，用以概括一套全新的软件开发价值观。这套价值观，通过一份简明扼要的《敏捷宣言》，传递给世界，宣告了敏捷开发运动的开始。

《敏捷宣言》

我们通过身体力行和帮助他人来揭示更好的软件开发方式。经由这项工作，我们形成了如下价值观：

个体与交互 重于 过程和工具
可用的软件 重于 完备的文档
客户协作 重于 合同谈判
响应变化 重于 遵循计划

在每对比对中，后者并非全无价值，但我们更看重前者。

Kent Beck Mike Beedle Arie van Bennekum Alistair Cockburn Ward Cunningham Martin Fowler
James Grenning Jim Highsmith Andrew Hunt Ron Jeffries Jon Kern Brian Marick
Robert C. Martin Steve Mellor Ken Schwaber Jeff Sutherland Dave Thomas

世紀鼎利

敏捷价值观

- 人和交互重于过程和工具
  - 人是获得成功最重要的因素
  - 一个优秀的团队成员未必就是一个一流的程序员，好的合作（沟通以及交互）能力比单纯的编程能力更为重要
  - 不要过分夸大工具的作用，团队的构建比环境的构建重要得多

- 可以工作的软件重于面面俱到的文档
  - 需要易于阅读的文档来对系统及其设计决策的依据进行描述
  - 短小并且主题突出，仅论述系统的最高层结构和概括的设计原理
  - 在向新团队成员传授知识方面，最好的两份文档是代码和团队

世紀鼎利

敏捷价值观 cont.

- 客户合作重于合同谈判
  - 不能仅仅写下一份软件描述，然后就让人在固定的时间内以固定的价格去开发。
  - 一个指明了需求、进度以及项目成本的合同存在根本上的缺陷，为开发团队和客户的协同工作方式提供指导的合同才是最好的合同。

- 随时应对变化重于遵循计划
  - 随时应对变化的能力常常决定着一个软件项目的成败
  - 较好的策略：为下一周做详细计划，为下三个月做粗略计划，再以后做极为简略的机会

世紀鼎利

敏捷实践的原则

- 1）最优先要做的是通过尽早地、持续地交付有价值的软件来使客户满意。
- 2）我们欢迎需求的变化，即使到了开发后期，敏捷过程能够驾驭变化，为客户创造竞争优势。
- 3）经常交互可以工作的软件，从几个星期到几个月，实践间隔越短越好。
- 4）在整个项目开发期间，业务人员和开发人员必须朝夕工作在一起。
- 5）微软斗志高昂的人构建项目。给他们提供所需的环境和支持，并且信任他们能够完成工作。
- 6）在团队内部，最有效率也最有效果的信息传达方式，就是面对面的交流。
- 7）可以工作的软件是进度主要的度量标准。
- 8）敏捷过程提出可持续开发。出资人、开发者和用户应该总是保持稳定的开发速度。
- 9）对作乐技术和良好设计的不断追求有助于提高敏捷性。
- 10）简单 - 尽量减少工作量的艺术是至关重要的。
- 11）最好的构架、需求和设计都源于自我组织的团队。
- 12）每隔一定时间，团队都要总结如何更有效率，然后相应地调整自己的行为。

世紀鼎利

主要的敏捷过程

- **XP**： XP（极限编程）的思想源自Kent Beck和Ward Cunningham在软件项目中的合作经历。XP注重的核心是沟通、简明、反馈和勇气。因为知道计划永远赶不上变化，XP无需开发人员在软件开始初期做出很多的文档。XP提倡测试先行，为了将以后出现bug的几率降到最低。
- **SCRUM**： SCRUM是一种迭代的增量化过程，用于产品开发或工作管理。它是一种可以集合各种开发实践的经验化过程框架。SCRUM中发布产品的重要性高于一切。
- **Crystal Methods**：Crystal Methods（水晶方法族）由Alistair Cockburn在20世纪90年代末提出。之所以是个系列，是因为他相信不同类型的项目需要不同的方法。
- **FDD**：FDD（Feature-Driven Development，特性驱动开发）由Peter Coad、Jeff de Luca 、Eric Lefebvre共同开发，是一套针对中小型软件开发项目的开发模式。此外，FDD是一个模型驱动的快速迭代开发过程，它强调的是简化、实用、易于被开发团队接受，适用于需求经常变动的项目。
- **ASD**：ASD（Adaptive Software Development，自适应软件开发）由Jim Highsmith在1999年正式提出。ASD强调开发方法的适应性（Adaptive），这一思想来源于复杂系统的混沌理论。ASD不象其他方法那样有很多具体的实践做法，它更侧重为ASD的重要性提供最根本的基础，并从更高的组织和管理层次来阐述开发方法为什么要具备适应性。
- **DSDM**：DSDM（动态系统开发方法）是众多敏捷开发方法中的一种，它倡导以业务为核心，快速而有效地进行系统开发。
- **轻量型RUP**：RUP其实是个过程的框架，它可以包容许多不同类型的过程

方法使用比例：

Scrum成为所有敏捷技术中最受欢迎的，选择Scrum方法的高达41%；其次是极限编程（XP），占到15%。另有14%的被调查者选择了混合使用Scrum和XP，13%的人使用自定义或其他类型的混合敏捷方法。Crystal和动态系统开发方法（Dynamic Systems Development Method，DSMD）都只有3%以下的使用者。

- **参考书目：**

An agile war story, Scrum and XP from the Trenches - How we do Scrum --- Henrik Kniberg
敏捷软件开发 - 原则、模式与实践（C#版） Agile Principles， Patterns， and Practicses in C# --- Rebort C. Martin Micah Martin

世紀鼎利

XP介绍

- 1）完整团队
- 2）用户故事
- 3）短交付周期
- 4）结对编程
- 5）验收测试
- 6）测试驱动开发
- 7）集团所有权
- 8）持续集成
- 9）可持续开发速度
- 10）开放的工作空间
- 11）计划游戏
- 12）简单设计
- 13）重构
- 14）隐喻Metaphor

世紀鼎利

Scrum介绍

- Scrum的基本假设：

  开发软件就像开发新产品，无法一开始就能定义软件产品最终的规程，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专案成功。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模型
  - **Product Backlog**：可以预知的所有任务， 包括功能性的和非功能性的所有任务。
  - **Sprint**：一次迭代开发的时间周期，一般为4-6周。在这段时间内，开发团队需要完成一个制定的Backlog,并且最终成果是一个增量的，可以交付的产品。
  - **Sprint Backlog**：一个Sprint周期内所需要完成的任务。
  - **Sprint Planning Meeting**: 在启动每个sprint前召开。该会议需要制定的任务是：Product Owner和团队成员将Backlog分解成小的功能模块, 决定在即将进行的Sprint里需要完成多少小功能模块，确定好这个Backlog的任务优先级。另外，该会议还需详细地讨论如何能够按照需求完成这些小功能模块。
  - **Daily Scrum Meeting**：开发团队成员召开，一般为15分钟。每个开发成员需要向Scrum Master汇报三个项目：今天完成了什么？是否遇到了障碍？即将要做什么？通过该会议，团队成员可以相互了解项目进度。
  - **Sprint Review Meeting**：在每个Sprint结束后，这个Team将这个Sprint的工作成果演示给Product Owner和其他相关的人员。
  - **Sprint Retrospective Meeting**：对刚结束的Sprint进行总结。会议的参与人员为团队开发的内部人员。

世紀鼎利

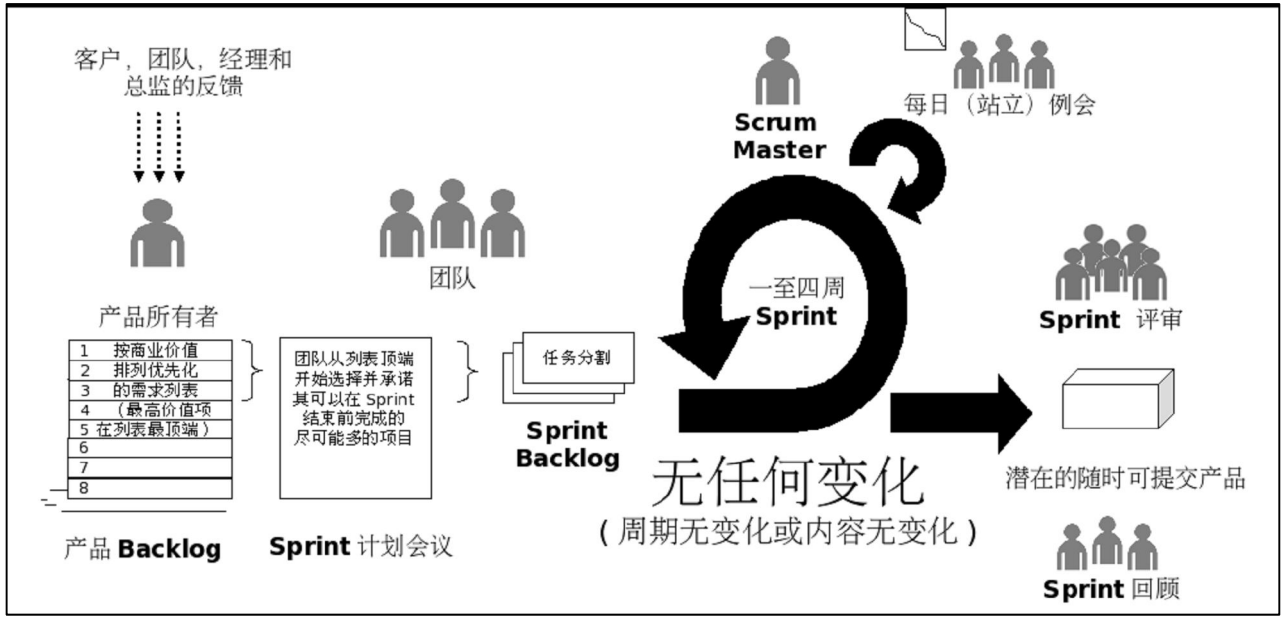
Scrum中的角色

- 产品所有者（Product Owner）- 客户，产品经理或产品市场经理
  - 确定产品的功能。
  - 决定发布的日期和发布内容。
  - 为产品的profitability of the product (ROI)负责。
  - 根据市场价值确定功能优先级。
  - 在Sprint内调整功能和调整功能优先级。
  - 接受或拒绝接受开发团队的工作成果。
- 开发团队（Scrum Team） - Pigs
  - 具有不同特长的团队成员，一般为五到十人，包括程序员，界面设计师，测试员和研究人员。
  - 确定Sprint目标和具体说明的工作成果。
  - 在项目向导范围内有权利做任何事情已确保达到Sprint的目标。
  - 高度的自我管理能力。
  - 向Product Owner演示产品功能。
- Scrum Master
  - 保证团队资源完全可被利用并且全部是高产出的。
  - 保证各个角色及职责的良好协作。
  - 解决团队开发中的障碍。
  - 做为团队和外部的接口，屏蔽外界对团队成员的干扰。
  - 保证开发过程按计划进行，组织 Daily Scrum, Sprint Review and Sprint Planning meetings。
  - 服务于整个团队，帮助团队铲除壁垒而取得成功，不指示和分配工作，只是协助流程的实施，推动团队自我组织和管理。
- “经理”的转变
  - “保姆”角色：布置任务，收取进程报告
  - “指导“角色：指导职业发展，在职辅导培训，协助铲除障碍，帮助解决问题，提供创新的建议和指导团队成员的技能发展
  - 经理们需要改进他们的管理方式方法；比如，运用苏格拉底哲学提问方式来帮助开发团队找寻问题的解决办法，而不是简单地决定解决方法并分配给开发团队。

世紀鼎利

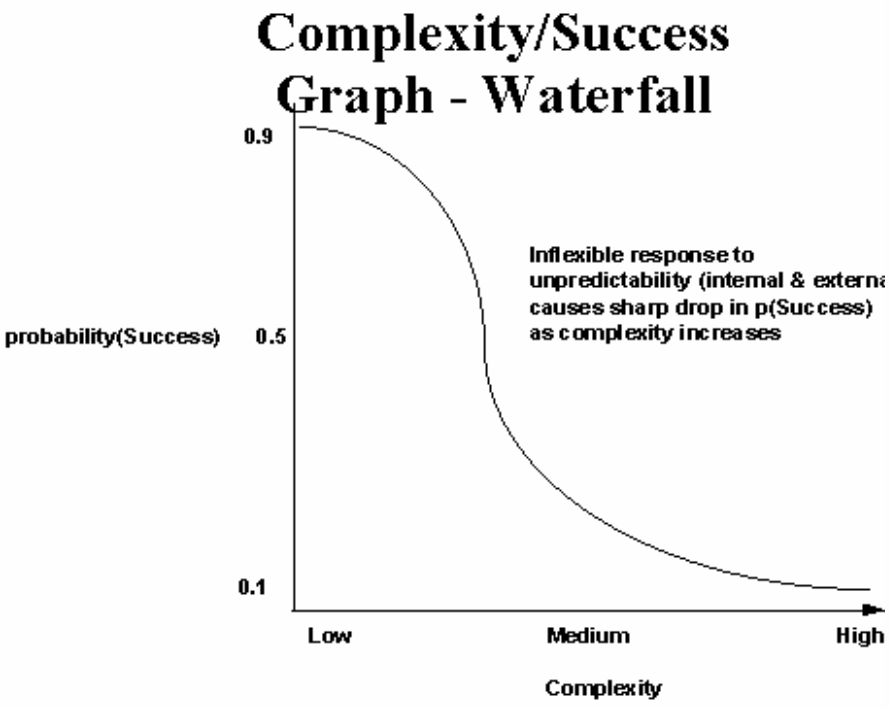
实施Scrum的过程

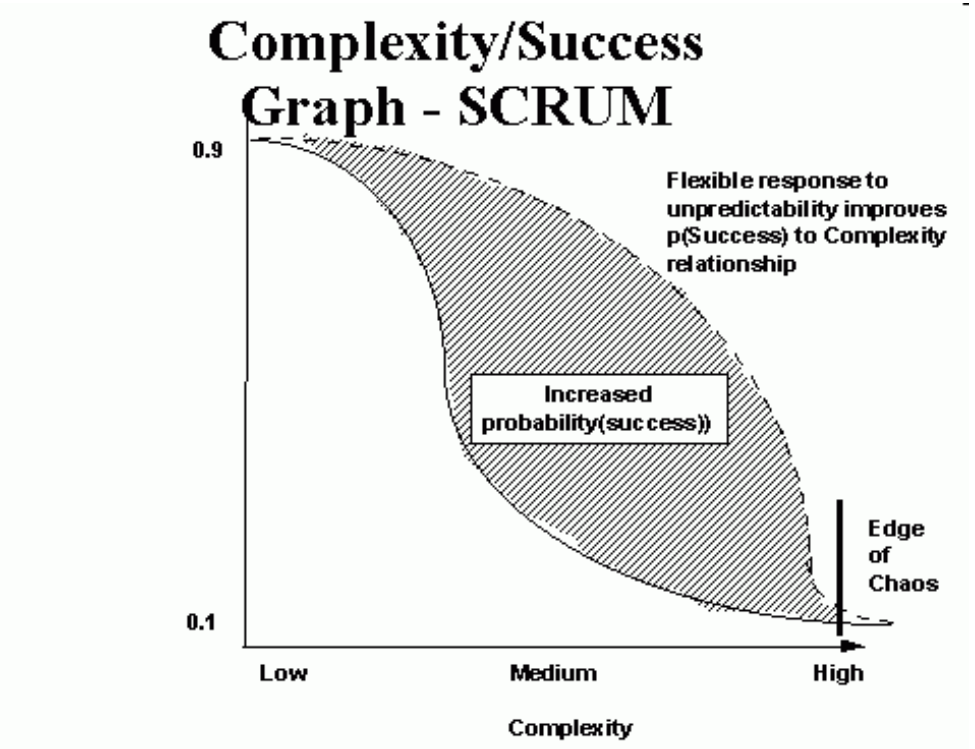
- 将整个产品的Backlog分解成Sprint Backlog,这个Sprint Backlog是按照目前的人力物力条件可以完成的。
- 召开Sprint Planning Meeting，划分/确定这个Sprint内需要完成的任务，标注任务的优先级并分配给每个成员。
- 进入Sprint开发周期，在这个周期内，每天需要召开Daily Scrum Meeting。
- 整个Sprint周期结束，召开Sprint Review Meeting，将成果演示给Product Owner.
- 团队成员最后召开Sprint Retrospective Meeting，总结问题和经验。
- 按照同样的步骤进行下一次Sprint.



世紀鼎利

Scrum模型和传统模型的对比

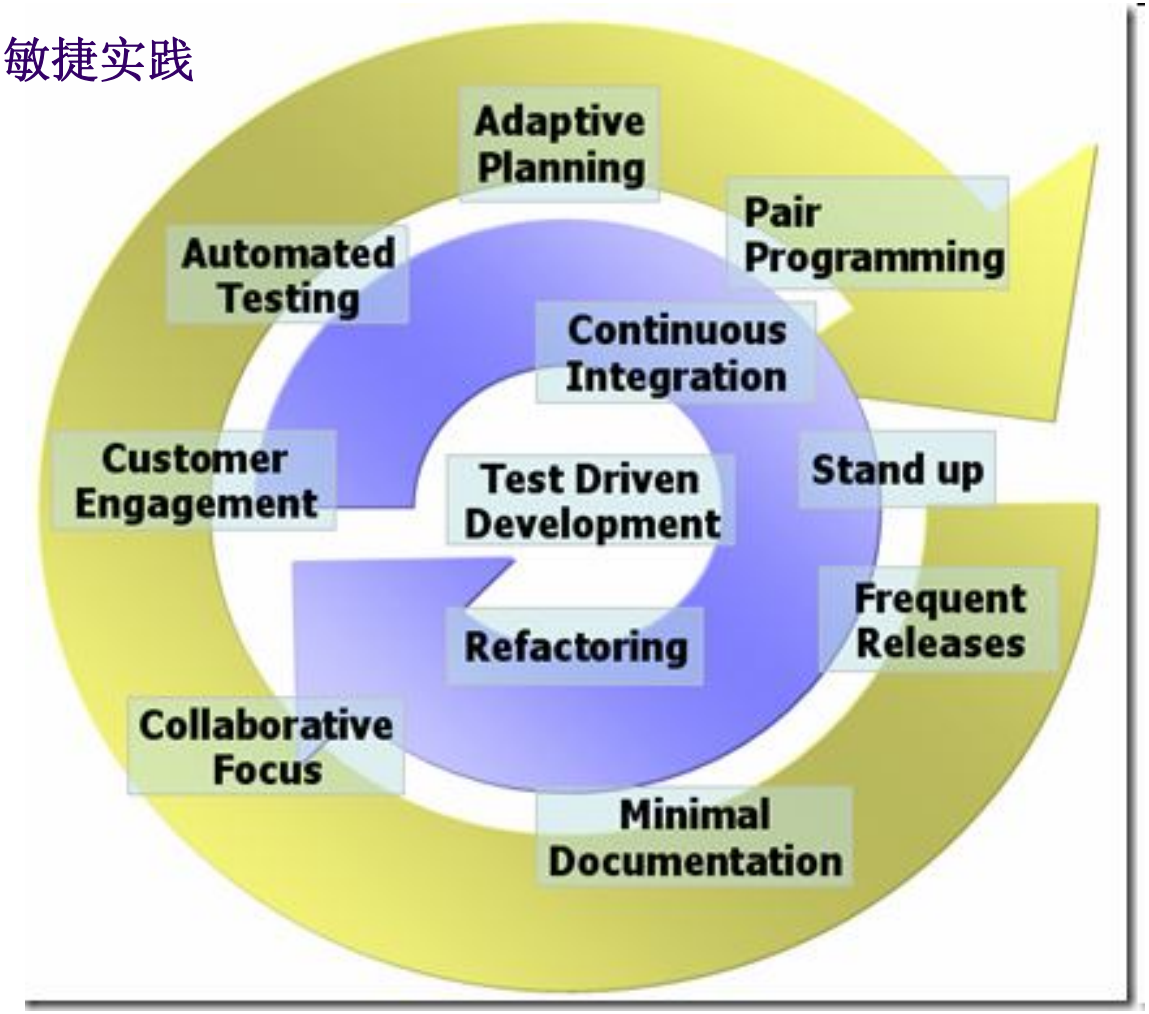




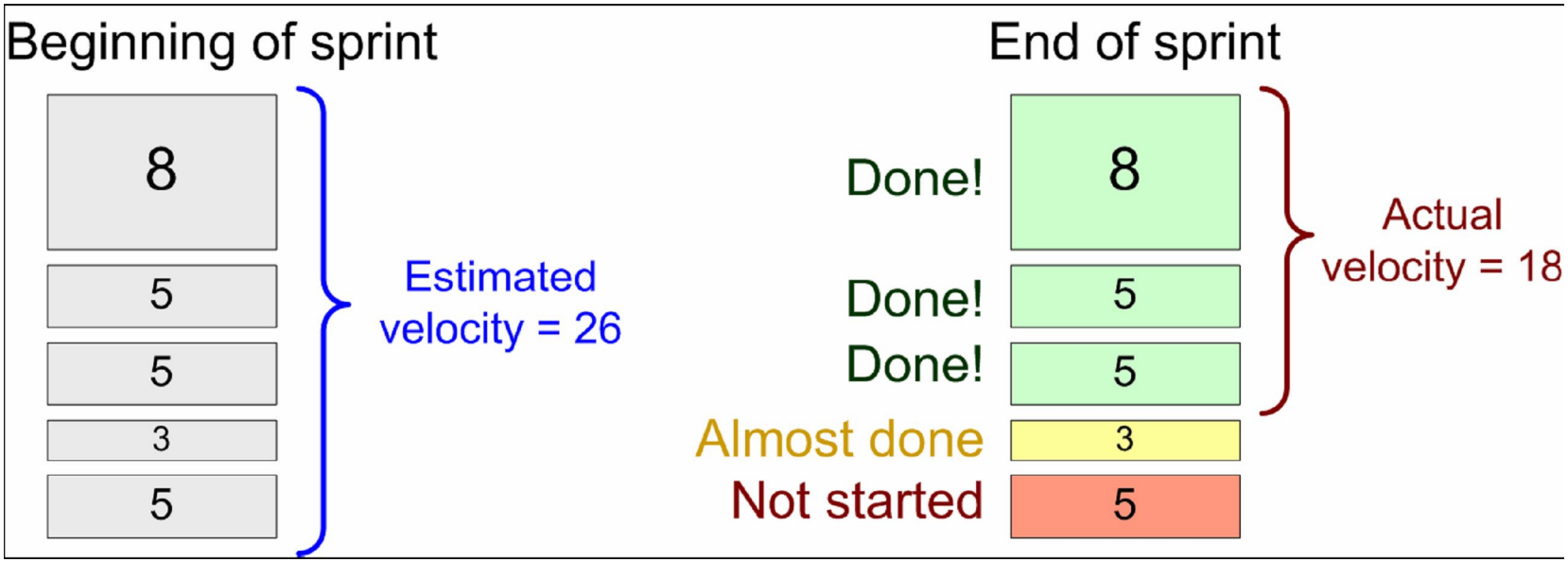
世紀鼎利

Scrum + XP 组合

不同的视角上的敏捷实践



进度估计



进度估计 – cont.

	Available days
Tom	15
Lisa	13
Sam	15
Dave	7
	50 available man-days

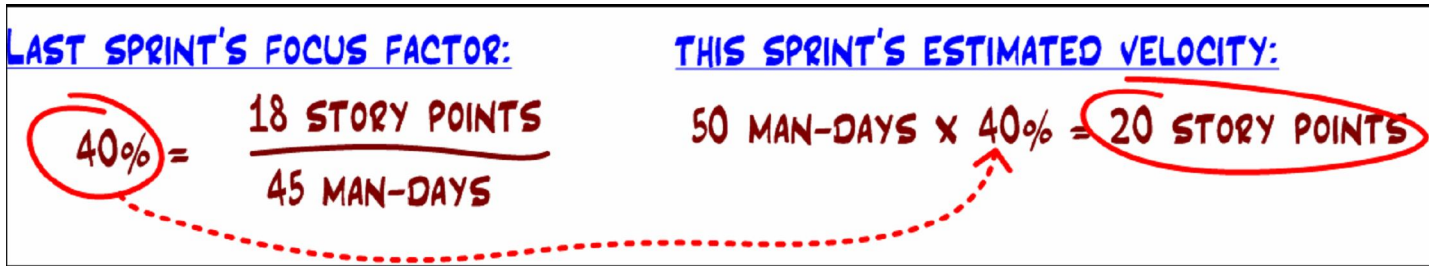
This sprint's estimated velocity:

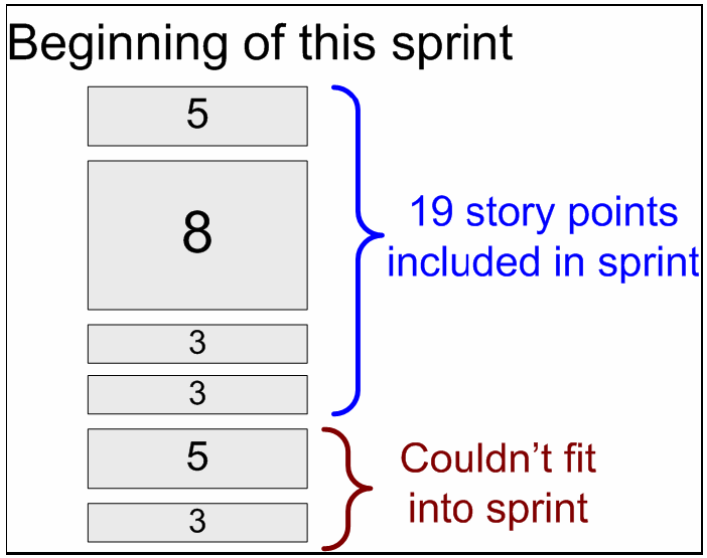
(available man-days) x (focus factor) = (estimated velocity)

Last sprint's focus factor:

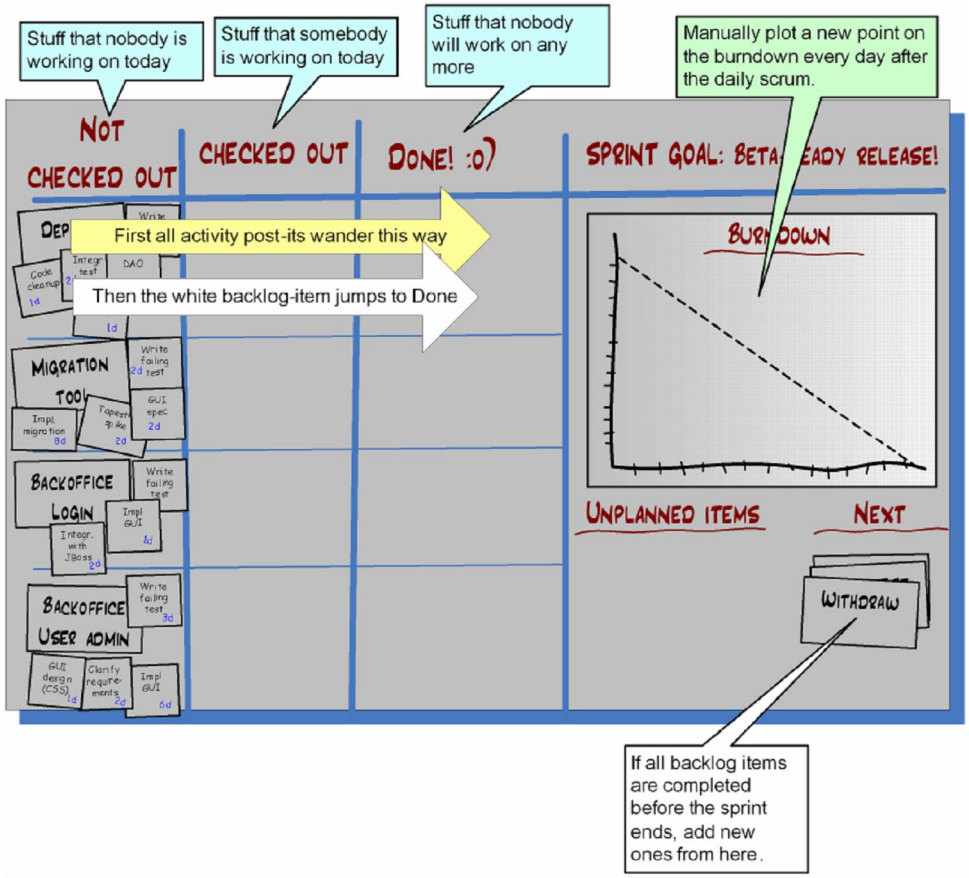
$$\text{(focus factor)} = \frac{\text{(actual velocity)}}{\text{(available man-days)}}$$

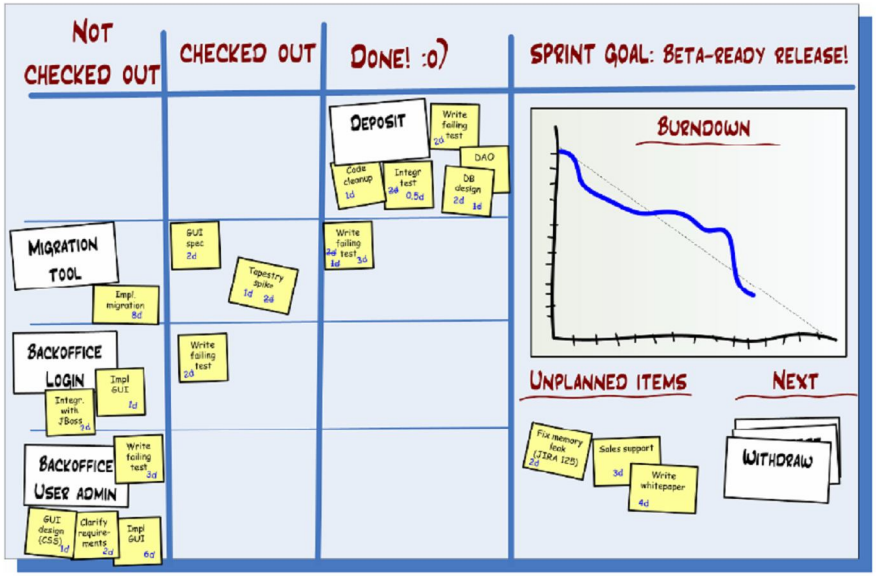
进度估计 – cont.





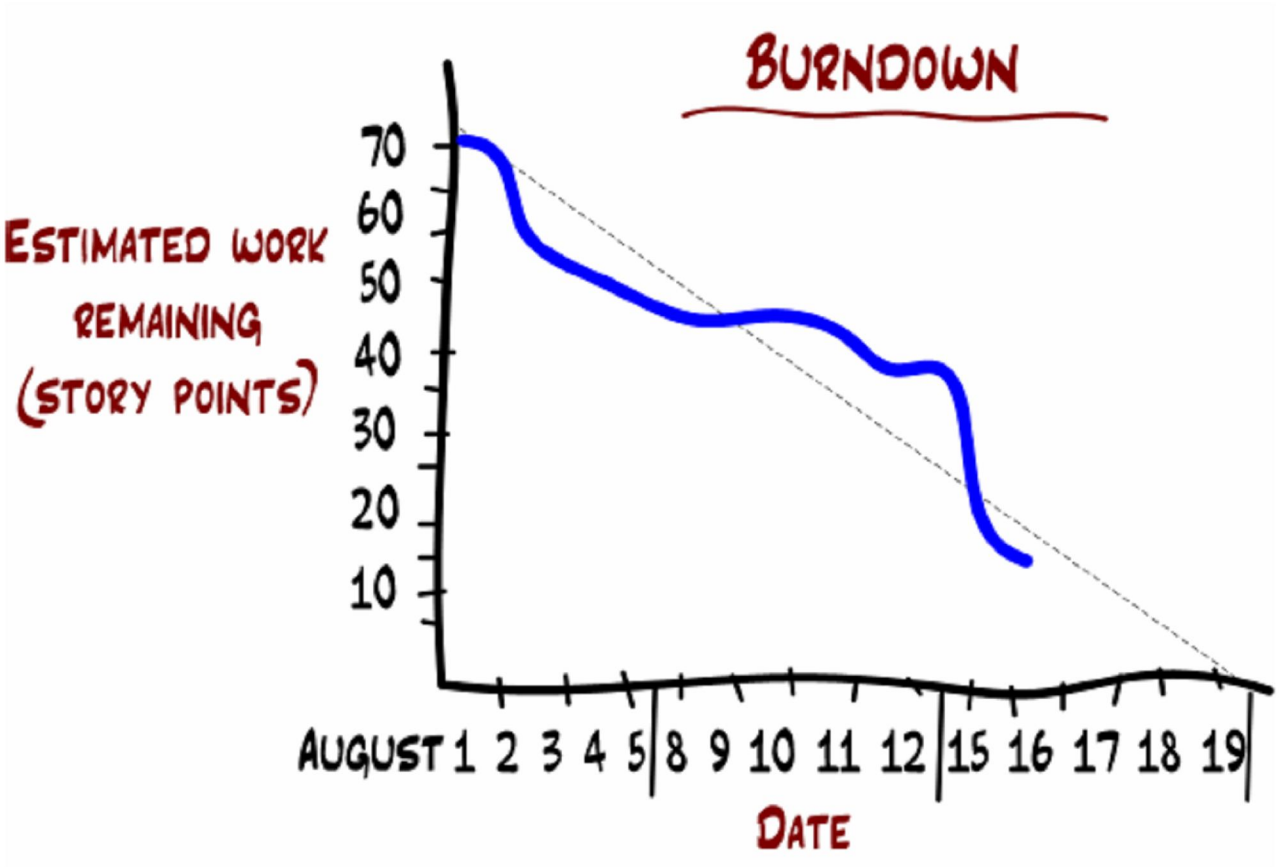
任务板（Taskboard）



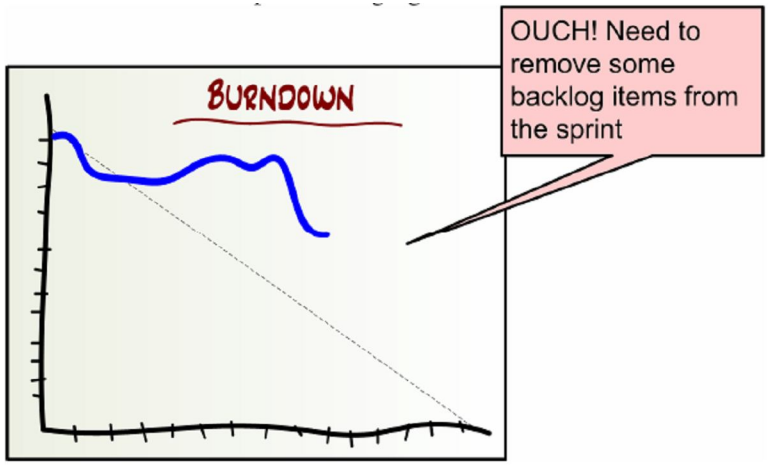


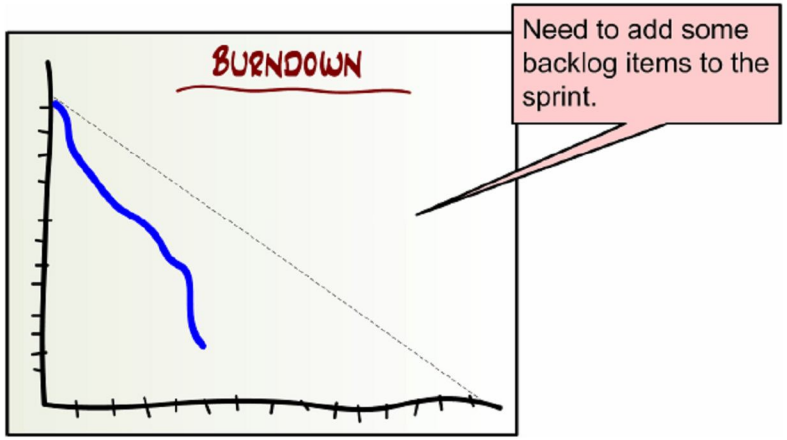
世紀鼎利

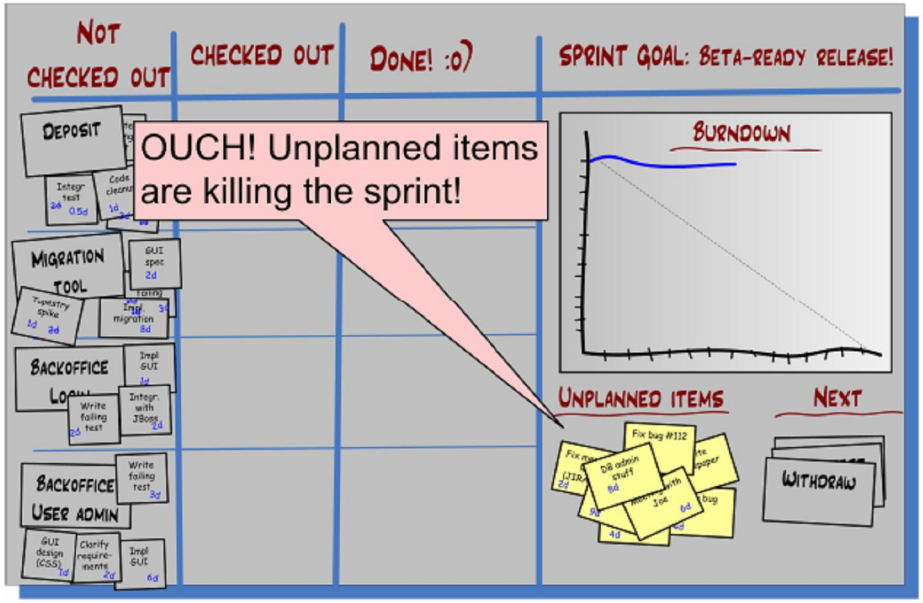
进度跟踪 – Burndown Chart



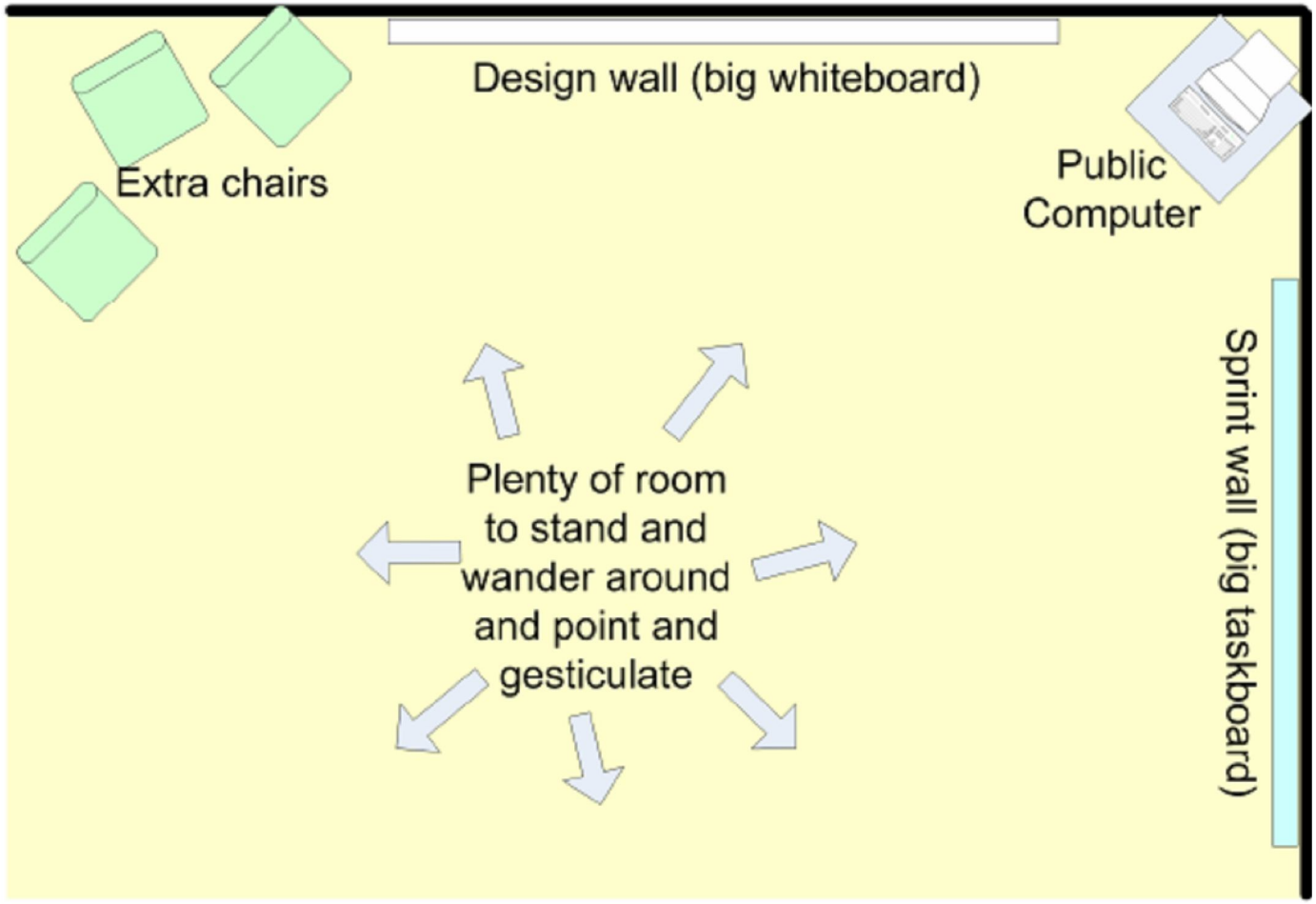
进度跟踪 – 问题发现







会议室设置



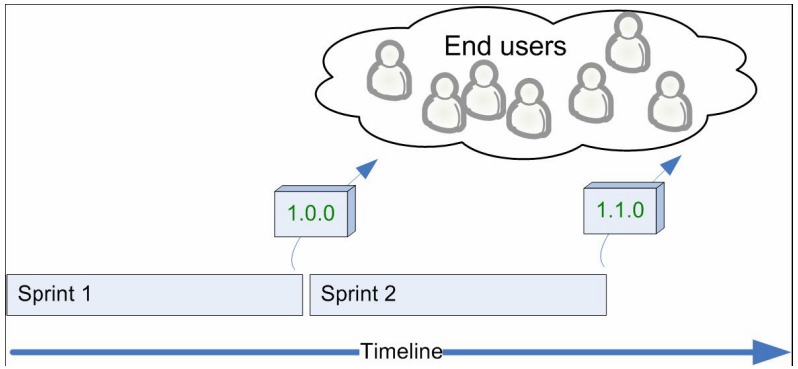
世紀鼎利

会议室设置 – cont.

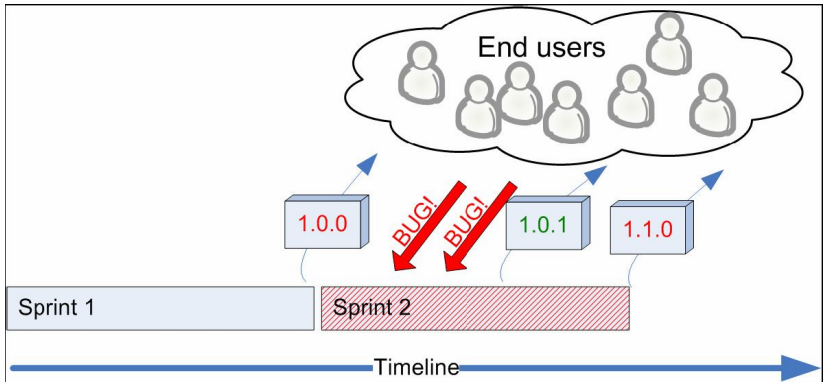
世紀鼎利

Sprint周期控制 – 问题

- 理想状况：版本持续发布



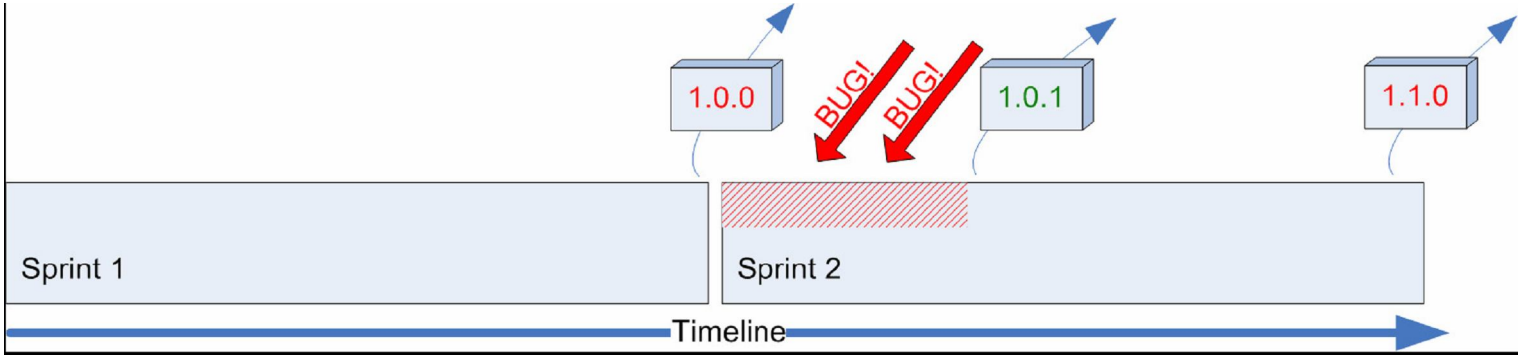
- 实际情况：发布版本出现问题



Sprint周期控制 – 解决

- 等待Bug修改完成。

- 优先修改Bug，并行进行新版本开发。



- 错误方法：优先进行新版本开发。

世紀鼎利

敏捷开发中的几大误解

- **敏捷仅仅是一个软件过程**
  - 人的作用提高到过程之上
  - 企业管理的层面，包括企业的价值观和文化
- **敏捷对人的要求很高**
  - 对于技术水平较低的开发人员，敏捷方法和传统方法对他的帮助是差不多的，因此看不到显著的效果；而随着开发人员技术水平的提高， 敏捷方法能够解开对人的束缚，鼓励创新，效果也会越来越显著。
- **敏捷没有文档，也不做设计**
  - 由文档具有的意义，产出和投入的比例，以及项目的具体情况决定
  - 持续设计，将设计工作分摊到了每天的日常工作中，不断的设计、改善（重构），使得设计一直保持灵活可靠。
- **敏捷好，其他方法不好**
  - 每一种方法都有他最善于解决的问题和最佳的发挥环境，在需求稳定、软件复杂度相对不高的时代，瀑布模型也可以工作的很好，而敏捷恰好适用于变化快风险高的 项目 — 这恰恰是现在很多项目的共性。
- **敏捷很好，因此我要制定标准，所有项目都要遵循着个标准**
  - 敏捷是动态的，而非静止不变的，因为这个世界本身就是变化的，在变化的世界使用不变的方法，是不现实的。银弹从来就没有过，在有限的将来也不会存在。

世紀鼎利

敏捷项目中的资源管理 - Roland Ceullar

- 问题：“同时有这么多项目在进行，我怎么能让人这么长时间只参加一个项目呢？”
  - 假定我们必须让人们参加不同的项目。
  - 同时进行的项目数目。
- 人不能被视为可被随意移动的单个个体。
  - 四个阶段：“组建期， 激荡期，规范期，执行期（Forming, Storming, Norming, Performing Model）”
  - 应该保证高效团队在尽量长时 间内不不被拆散。
  - 根据各个高效团队的不同能力和技术来分配项目，而不是根据项目来分配人。
- 针对组织中多项目的管理，Roland建议使用放慢速度的方式。
  - 组织应该建立项目队列，而不是让团队同时开发多个项目。
- 把关注点放在人身上。
  - 如果人们都能得到最好的利用，敏捷项目就能得到回报。所以，最好让项目以团队为中心，而不是让人们在项目间移动

世紀鼎利

给敏捷团队发奖金就像在刀尖上跳舞

- Distributing Bonus to Agile Teams is Like Playing with Dynamite（http://www.infoq.com/news/2008/03/bonus-for-agile-teams）
- 既然软件开发是团队活动，那么发奖就不应该针对个人，而是要根据团队的绩效。
- 50%根据团队绩效，50%根据个人绩效。
  - 问题：如何去检查敏捷团队中个人的绩效
- 通过个人绩效评估发奖金往往都会影响生产力
  - 会成为团队内部冲突的主要因素，让一个运作良好的团队很快分崩离析。
  - 一旦按照个人绩效来发奖金，那人们就会把个人目标凌驾于团队目标之上。
- 两种方式，第一种是每个人得到工资的X%作为奖金，另一种是整个团队得到$X平均分配。
  - 按百分比来算，只会让工资越多的人拿得奖金也越多。
  - 把奖金平均分配就是明显的吃大锅饭，这对多干活的人是不公平的，会让他们情绪低落，效率降低
- 让团队决定怎么分配奖金
  - 团队拿到了一笔很大的奖金，被告知他们自行分配。他们会想办法让大家的意见达成一致，但是这个过程会在团队内部造成巨大的难以修复的裂痕。最后他们能够做 到的就是平均分配，尽管很多人会认为它不公平。让他们自行分配导致的冲突，会让大多数人觉得还不如一开始没有这笔钱呢。

敏捷开发：程序员你不能一个人在战斗

- 单打独斗并不能成为英雄，而只会成为“堂吉诃德”式的人物。
- 东方人内敛的性格造成我们对于承担责任有种天生的抵触，在Scrum中这恰恰是不负责任的表现。谁都不为项目负责，最 后导致Scrum敏捷开发的失败。
- 沟通能力上的欠缺，使我们很愿意单打独斗，而不是以一个整体去战斗。

世紀鼎利

结束的话

- 传统软件工程思想是希望稳定需求，通过关注过程而保证结果，
- 敏捷思想，关注的是参与开发的人，以及提交物的质量，强调沟通、协作，主张拥抱变化。

- 无法执行的规定还不如没有规定。
- 在工作中，不要让流程或者制度成为推卸责任的借口。
- 代码质量应该在一切可能的情况下成为开发人员的第一目标。