IBM Rational 软件创新论坛

Let's build a smarter planet.

开发有道
创新"智"造

Innovate2010

敏捷开发项目管理

孙昕

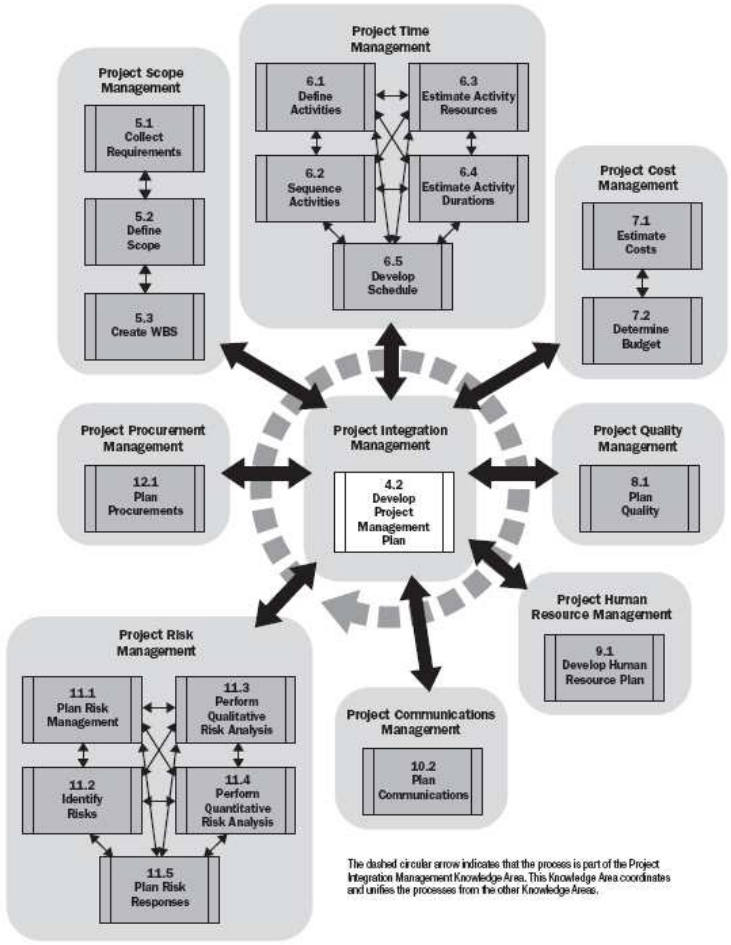
IBM Rational资深技术顾问

议题

- 软件项目管理的演进
- 敏捷软件项目中的一些最佳实践
- RTC敏捷项目管理的最佳载体

传统项目管理 - PMI

- 5大过程组
- 9大知识领域
- 44个管理过程
- PMI的关心和焦点:
  - 计划驱动
  - 紧密控制
  - 面向业务进行管理



软件项目管理的实践

我们需要一个完整、详尽的计划。

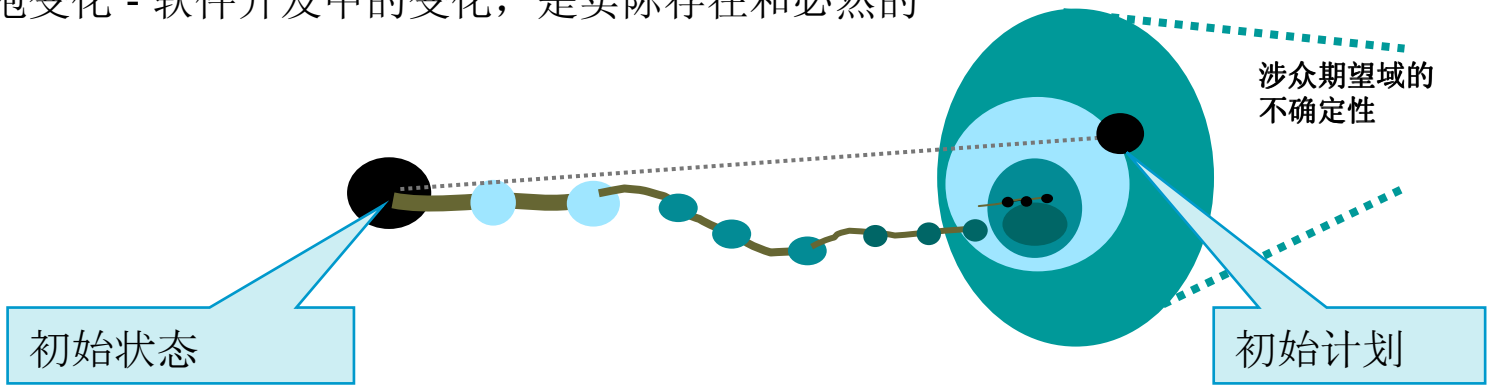
软件开发中能实现么?

我们需要设计考虑所有的风险和预留的内容。

软件项目能否预先考虑到所有的风险?

软件项目中难以预知所有的内容和风险，这不是传统行业！！！

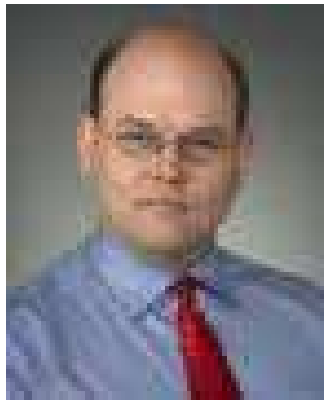
- 软件开发工作，是一个逐步认知和明晰的活动
- 拥抱变化 - 软件开发中的变化，是实际存在和必然的

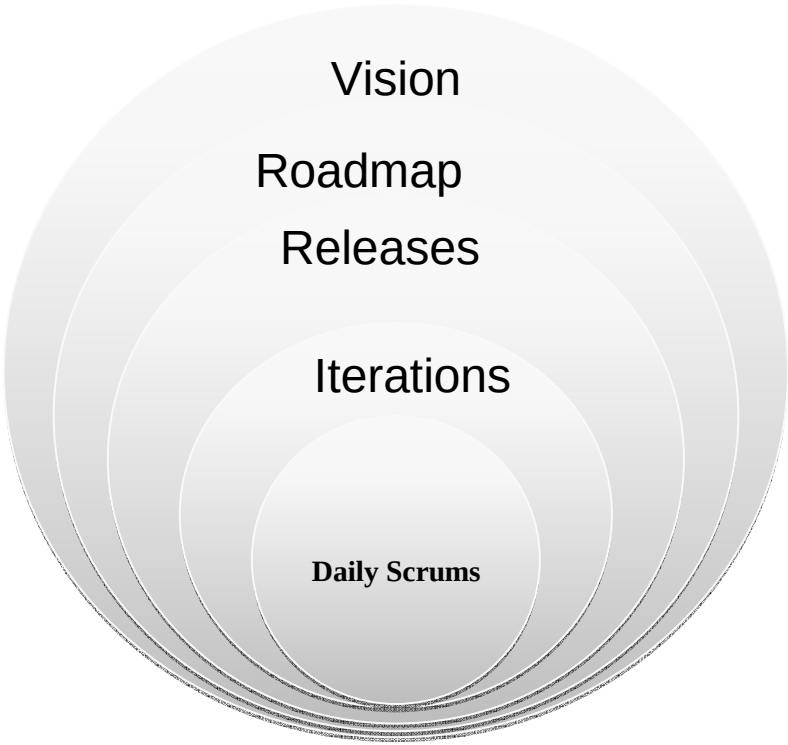


敏捷项目管理

Scott, 我们今天的软件项目管理更应侧重这些内容:

- 弹性的项目管理方式
- 通过初始计划开展工作
- 项目的资源管理和控制



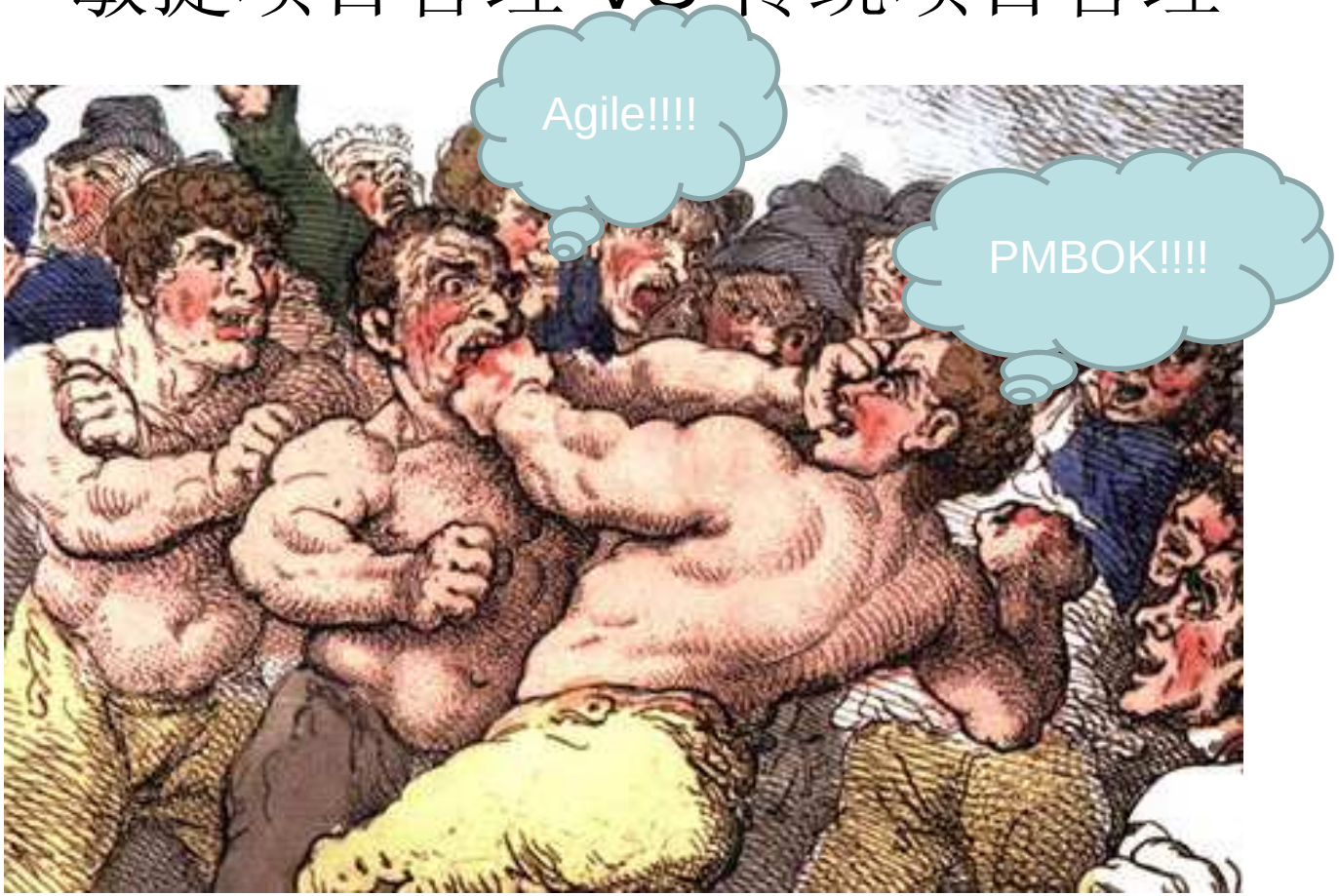


敏捷项目管理和传统项目管理比较

- 传统项目管理关注与整个系统的构建，从整体角度考虑
  - 项目计划的建立、项目计划的执行
  - 项目的风险根据项目计划考虑
- 敏捷项目管理更关注与交付的价值
  - 高质量的交付物是最重要的
  - 系统不是一次构建而成，而是迭代演进的
  - 基于完整的场景构建计划、并按优先级执行

您是想获取一些更有价值的交付产品呢，还是
只想完成进度表!!

敏捷项目管理 VS 传统项目管理



鱼和熊掌可以兼得

议题

- 软件项目管理的演进
- 敏捷软件项目中的一些最佳实践
- RTC敏捷项目管理的最佳载体

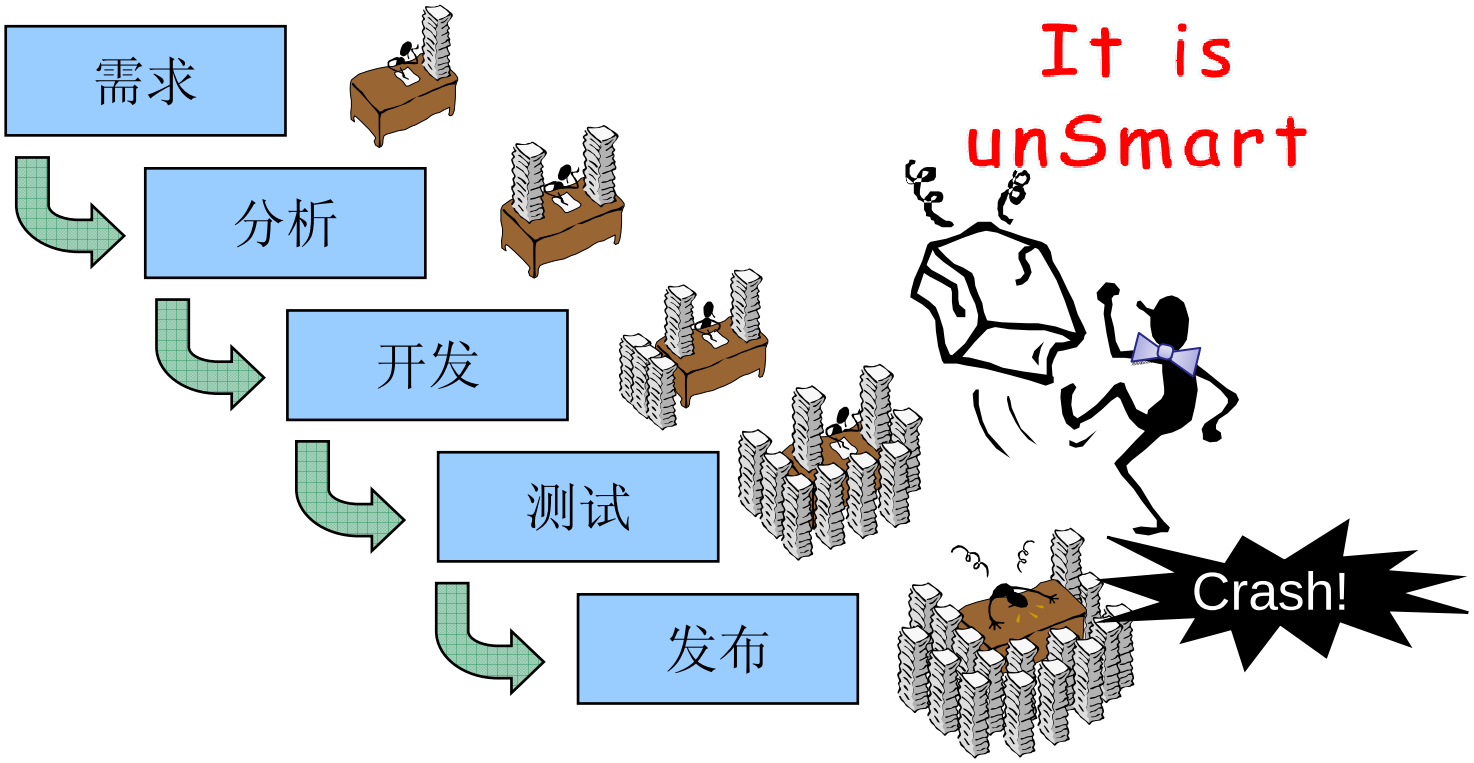
敏捷项目管理方法：Scrum

- 敏捷核心
- 迭代开发
- 2级项目规划
- 整体团队
- 持续集成
- 测试驱动开发

敏捷项目的核心-迭代

敏捷核心
- 迭代开发
- 2级项目规划
- 整体团队
- 持续集成
- 测试驱动开发

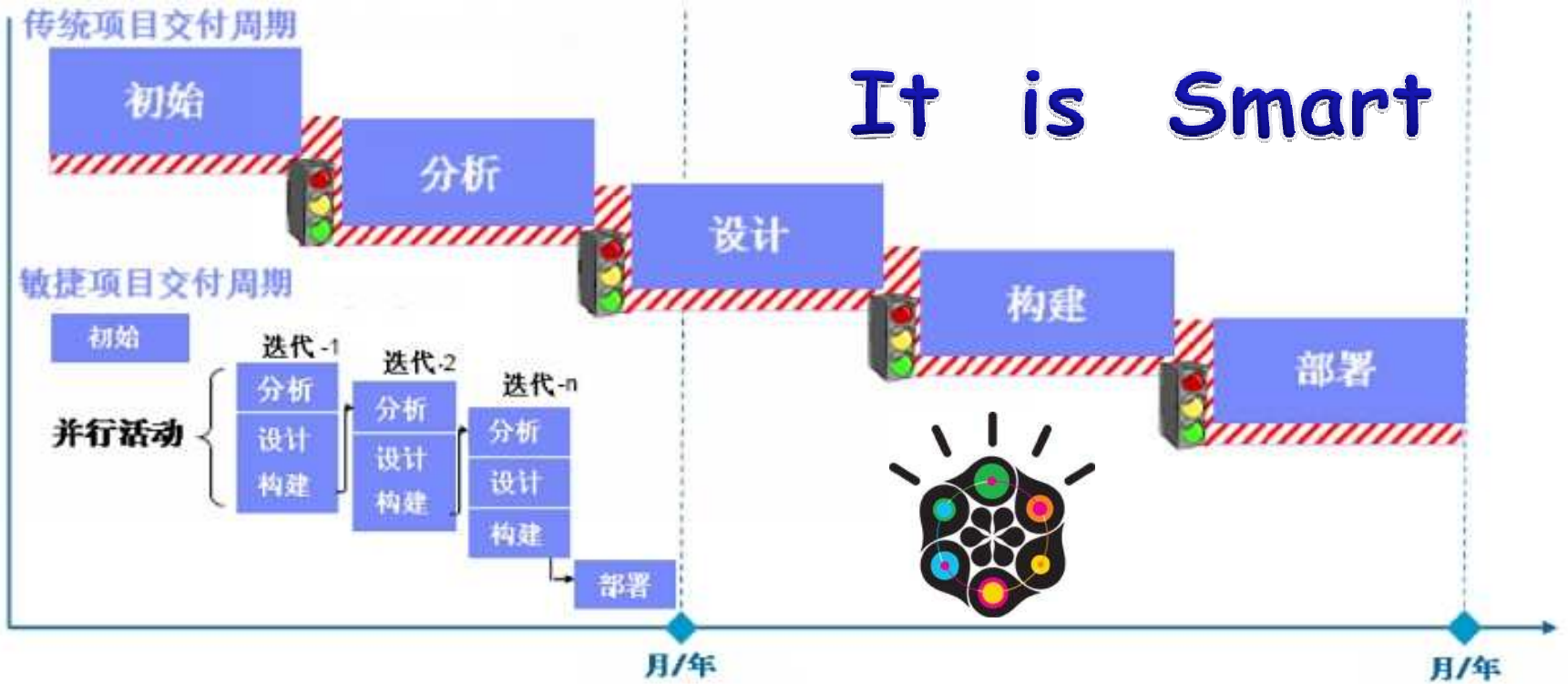
- 大部分公司还在使用瀑布模型



迭代式开发:本质改变 – 你多久运行你的项目一次?

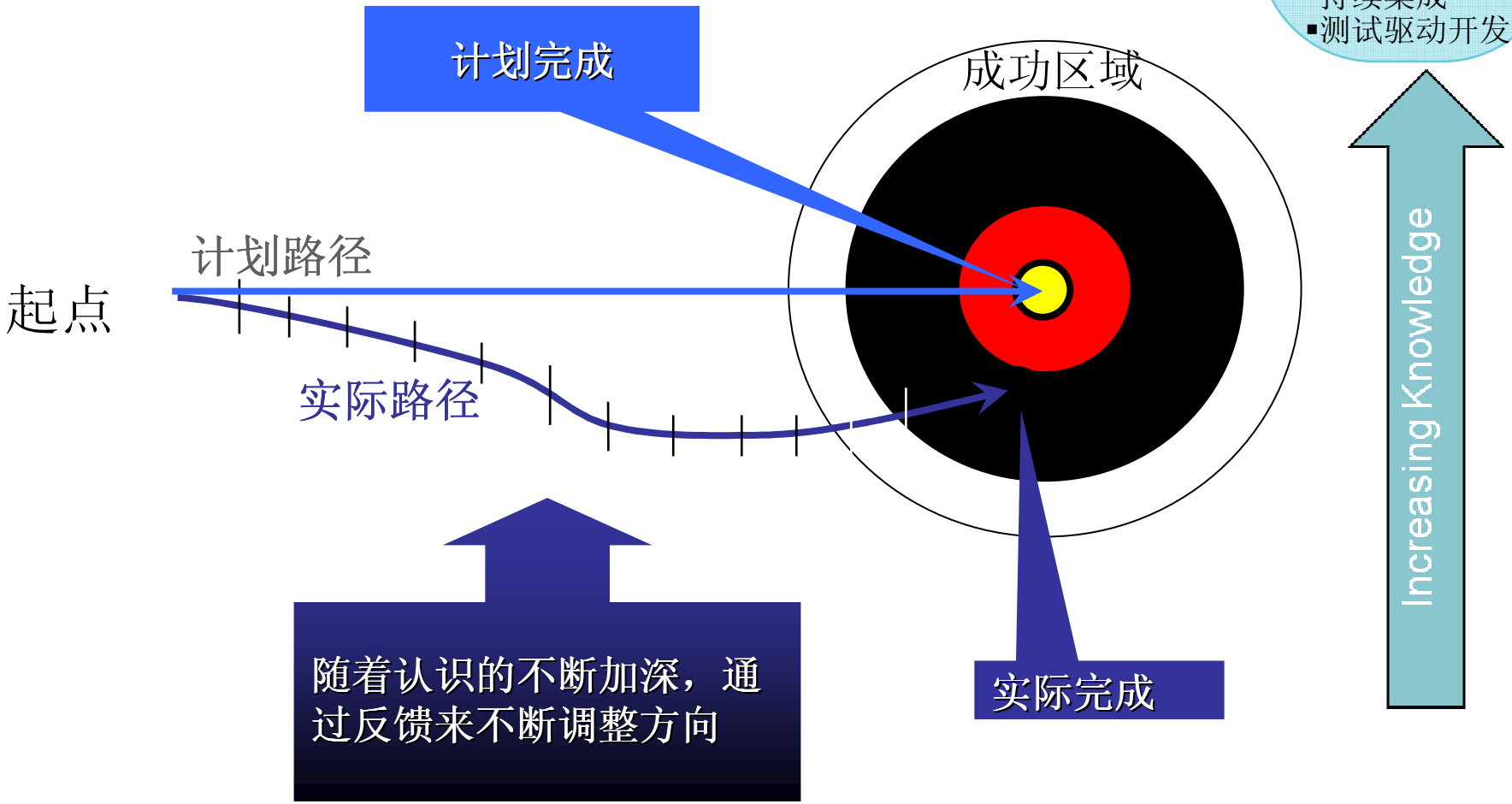
重复=自信=速度

- 变长的、复杂的项目周期为短的、简单的迭代周期
  - 团队在快速迭代的重复过程中，快速得到反馈/获取经验
  - 团队每执行一次迭代，信心都得到提升
  - 信心提升，效率和速度也相应提高了



迭代和两级规划：通过反馈来不断调整方向

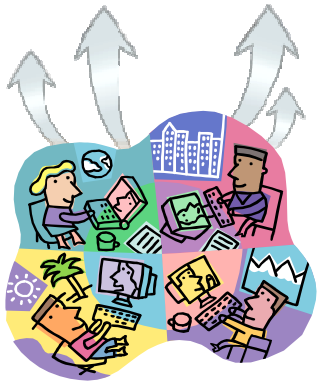
敏捷核心
- 迭代开发
- **2级项目规划**
- 整体团队
- 持续集成
- 测试驱动开发



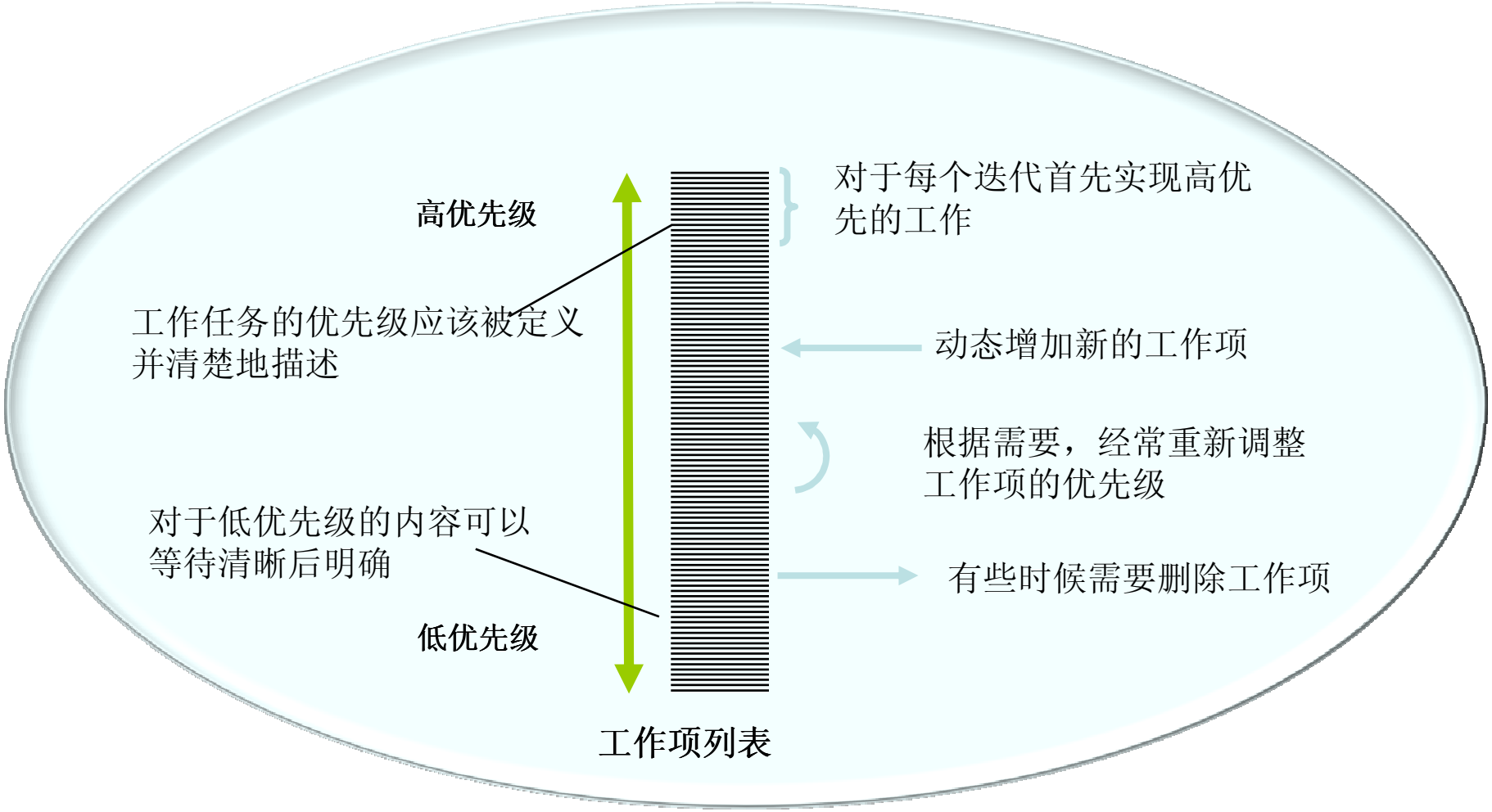
符合需求开发的启发式过程要求

为什么要进行两级项目规划

- 项目的特点：逐步完善 – 决定了计划的渐进明细
- 不断的获取干系人反馈，修订范围
- 符合启发式的需求开发过程要求
- 计划和变化的有效平衡
- 复杂的事情简单化



根据真正的需要修改工作安排



敏捷核心
- 迭代开发
- 2级项目规划
- **整体团队**
- 持续集成
- 测试驱动开发

核心敏捷最佳实践：“整体团队”

- ✓Scrum团队一般有6～8个人
- ✓拥有多种技能的、跨职能协作的团队
- ✓关注向干系人交付价值
- ✓整体团队：自指导、自组织、可持续的速度

文化变革： 人本管理从麦格雷戈的X理论向Y理论的转变

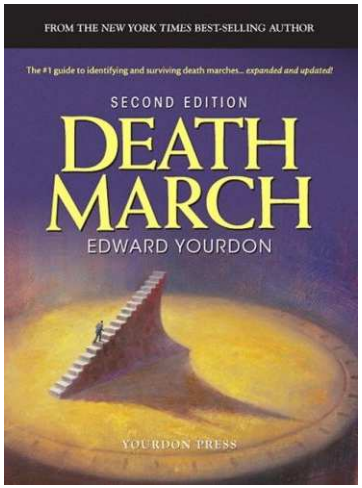
敏捷最佳实践：“整体团队”的自指导团队

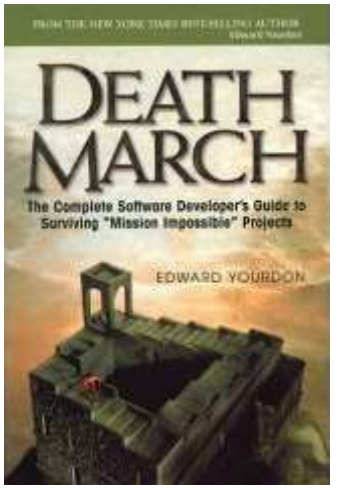
共享远景
和目标

团队承诺

- 拥有**共同目标、分担责任，彼此承诺**致力于目标实现
- **团队**拥有足够的**授权和资源**，解决问题，找到自己的成功之路
  - **做自己喜欢的事情**
  - 有效的沟通和信息的透明
  - 适当的团队建设

敏捷最佳实践：“整体团队”的可持续速度





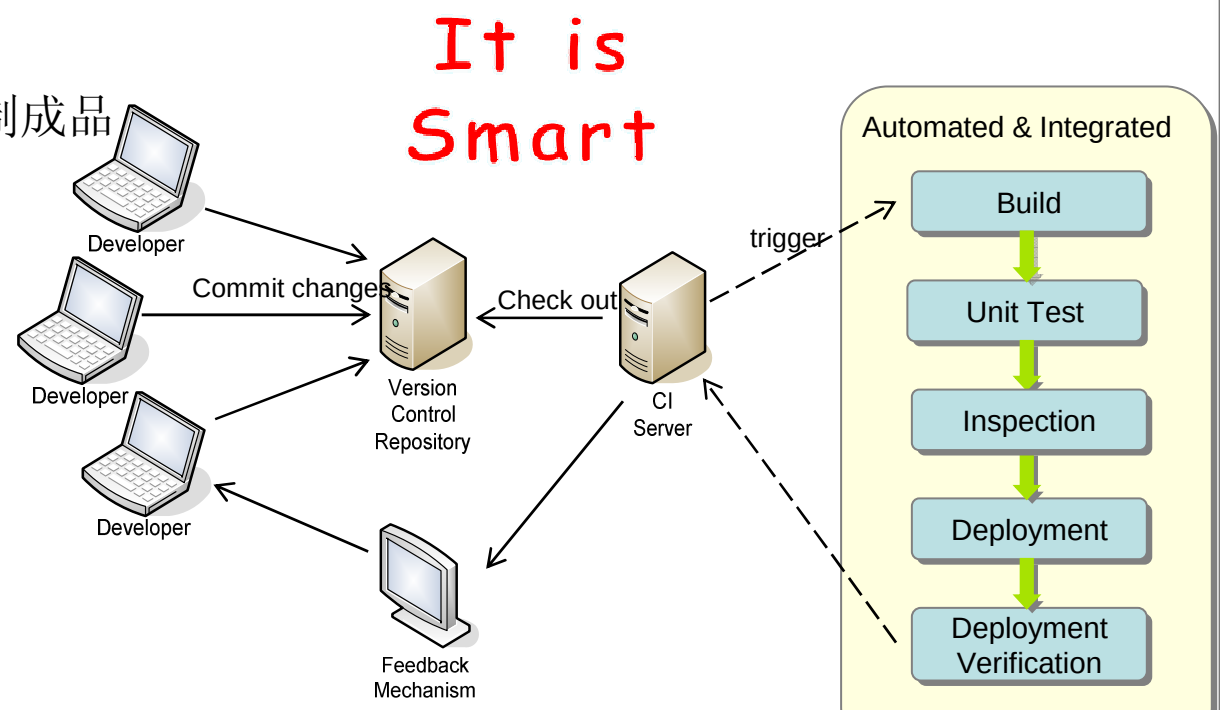
敏捷过程推行可持续的开发– 保持团队在一个可持续的生产力水平上工作

文化变革： 关注可持续发展

敏捷核心
- 迭代开发
- 2级项目规划
- 整体团队
- **持续集成**
- 测试驱动开发

有效的持续集成

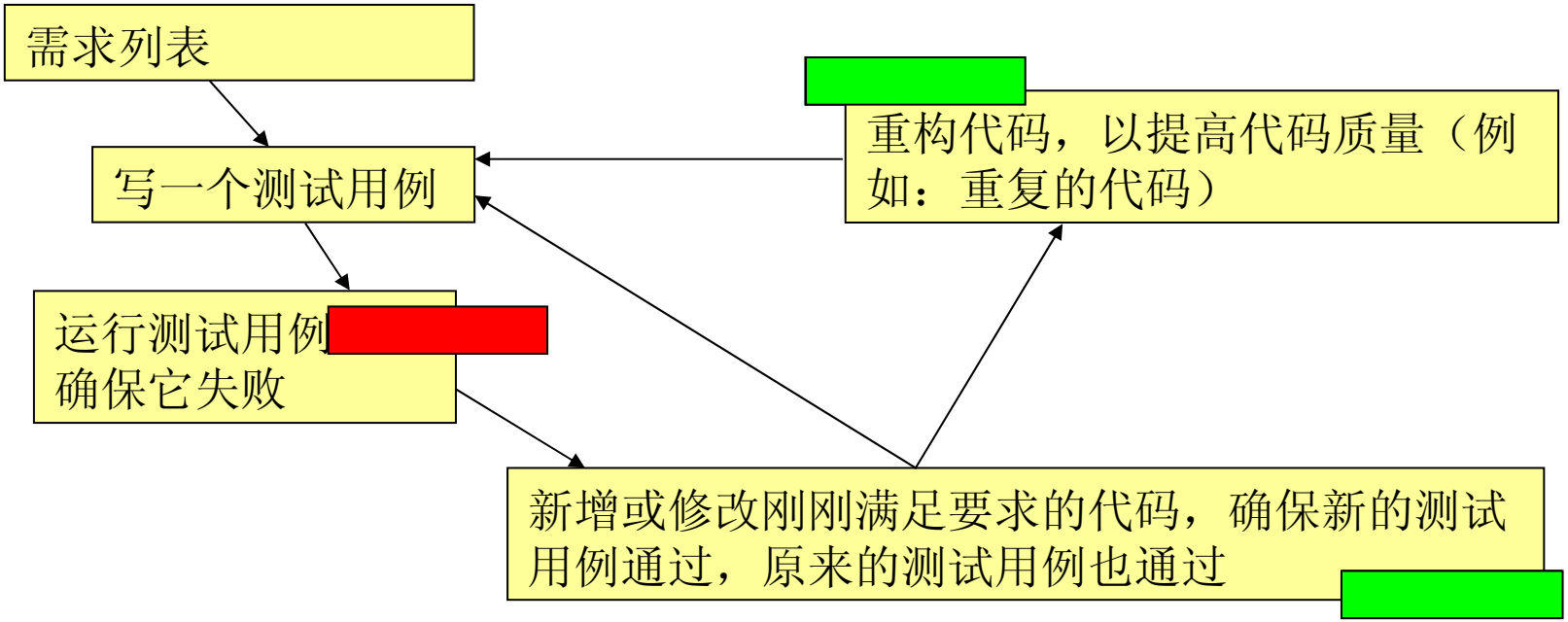
- 持续集成（CI）是一种实践，能够让团队在持续地构建的基础上，不断收到反馈并进行改进，不必等到开发周期后期才寻找和修复缺陷。应包含：
  - 自动化的运行测试
  - 自动产生可部署的二进制成品
  - 自动化的部署
  - 自动的版本标识
  - 自动的回归测试
  - 自动化的生成度量报告

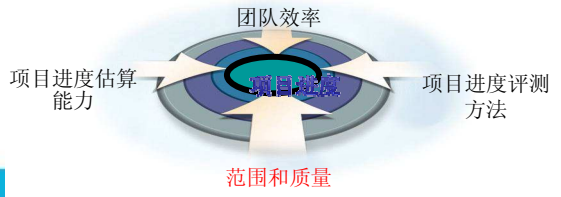


敏捷最佳实践：测试驱动开发(TDD)

敏捷核心
- 迭代开发
- 2级项目规划
- 整体团队
- 持续集成
- **测试驱动开发**

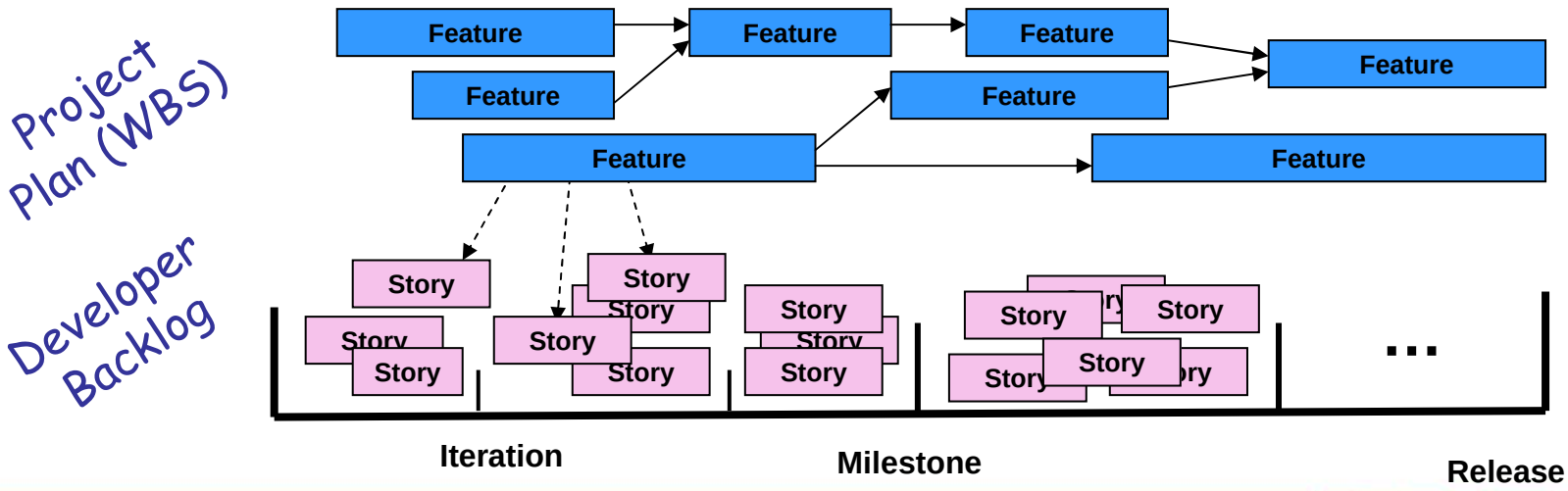
一种编程实践：所有的代码编写都是为了响应一个失败的测试





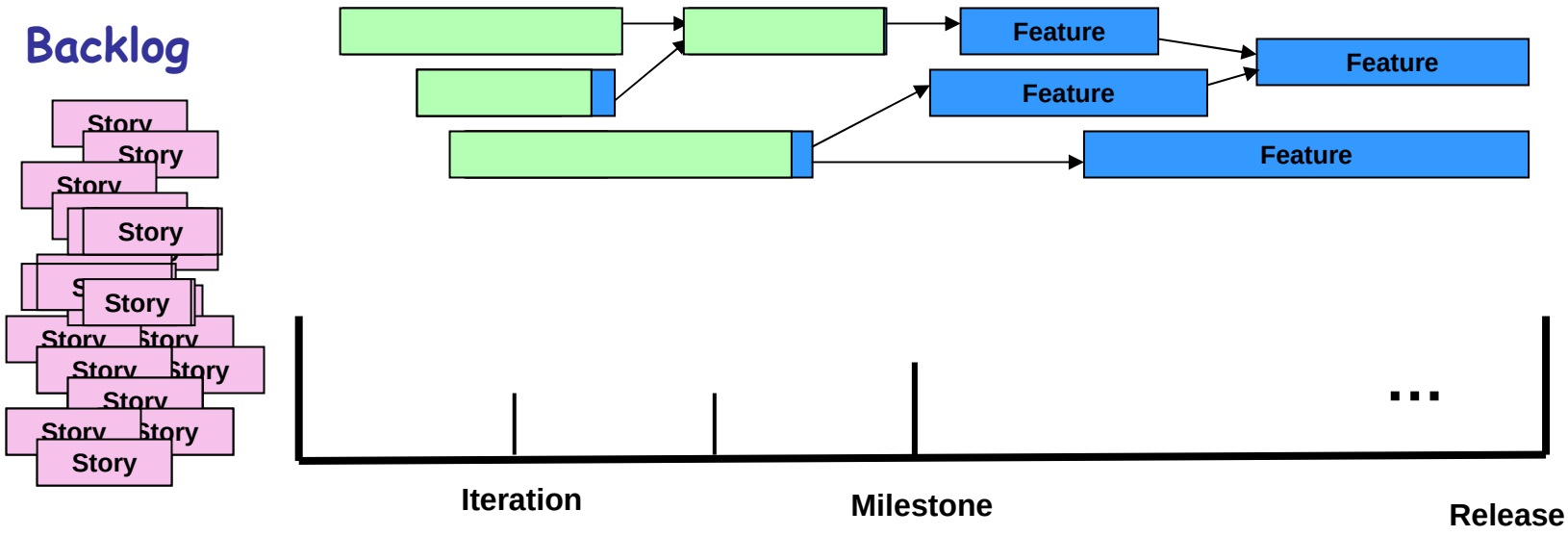
将敏捷项目管理和计划结合起来

- 基于一个完整的Story描绘产品，产品计划基于Story组成
- 通过Story进行优先级排列，而不是features
  - 可能部分Story没有实现，没有问题，准备继续！
- Story是最小的迭代管理元素，一个迭代起码包括一个Story
- 里程碑也是基于Story组成的



Features和Stories

- 业务分析分析人员，仍可以基于Feature分析业务特性，但是他需要将其和Story连接起来
- 里程碑控制也要以Story控制
- 开发团队通过Story，开展开发工作
- Stories本身也是不断变化的，可以增加、删除、修改、以及从新排列优先级

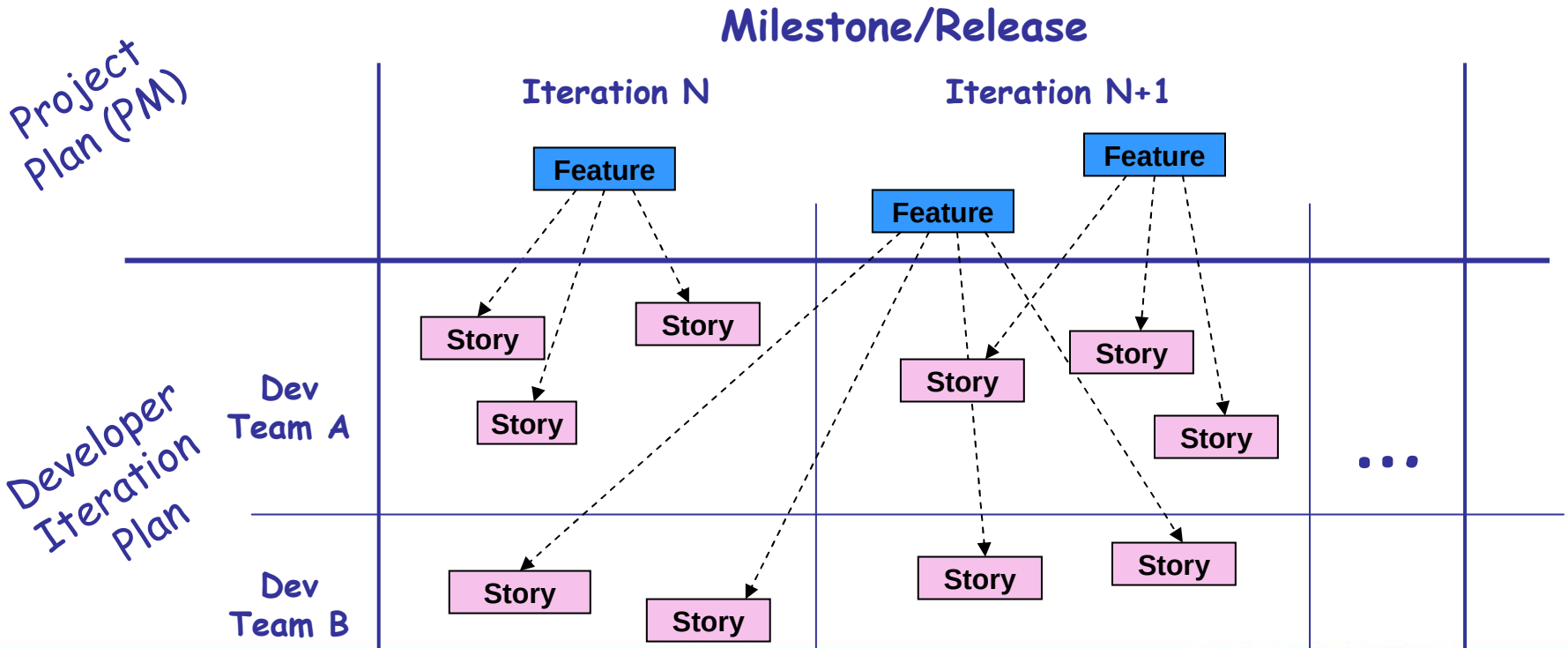


使用Backlog进行任务计划的优势

- Backlog基于Story管理，具有完整的业务价值，
  - Story可以支撑Feature
- 确保用户了解完整的Story而不只是Feature
- 小力度的Story能够提供更准确的任务情况
- Stories更能保证系统交付的可测试性

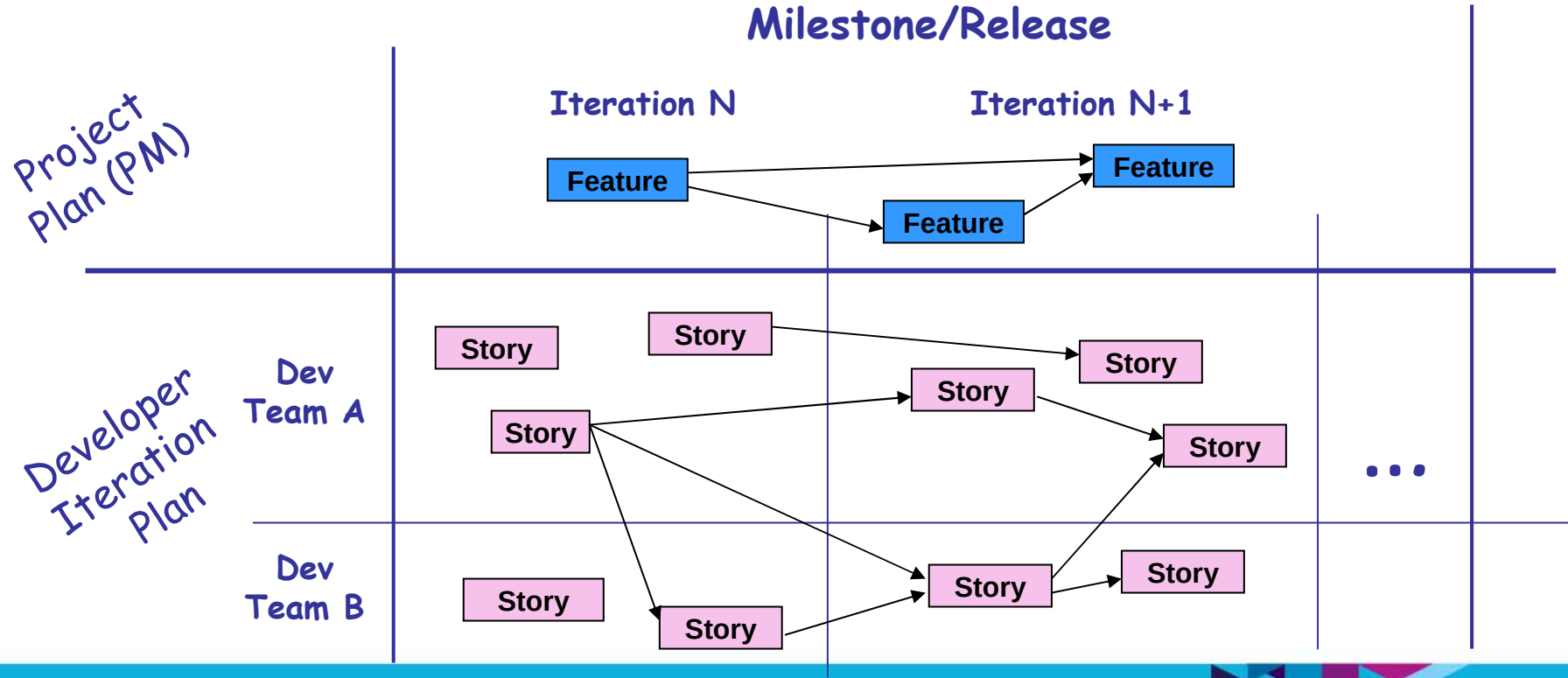
开发团队组织

- 理想中，Feature和整个Story的开发最好有一个团队完成
- 实践中，Feature需要和Story在统一个迭代中完成
  - 未完成的Story需要制定到下个迭代中



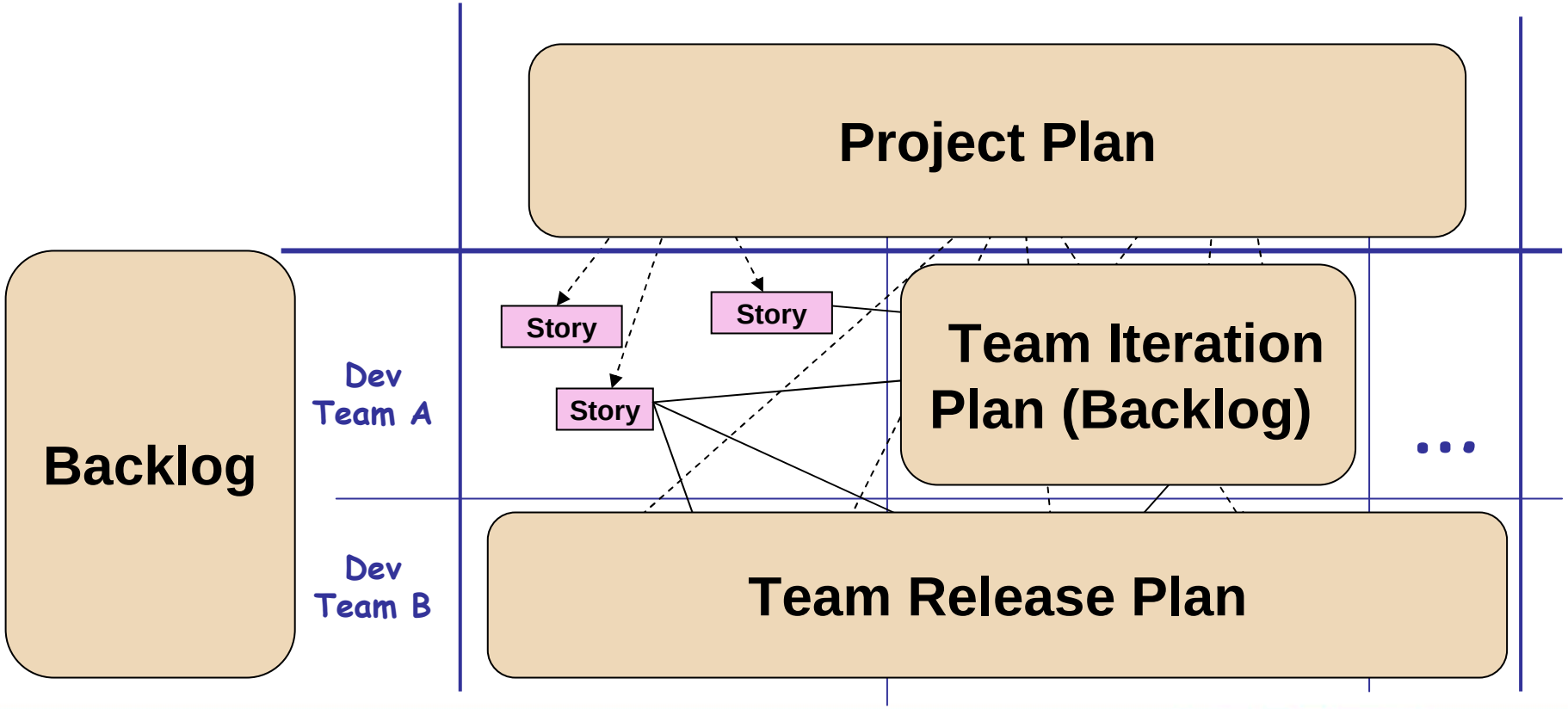
具体的工作任务分配

- Stories与Feature间有紧密的联系，也体现在依赖上
- 优先级和进度都基于Story
  - 根据Story的完成情况，调整Feature的工作计划

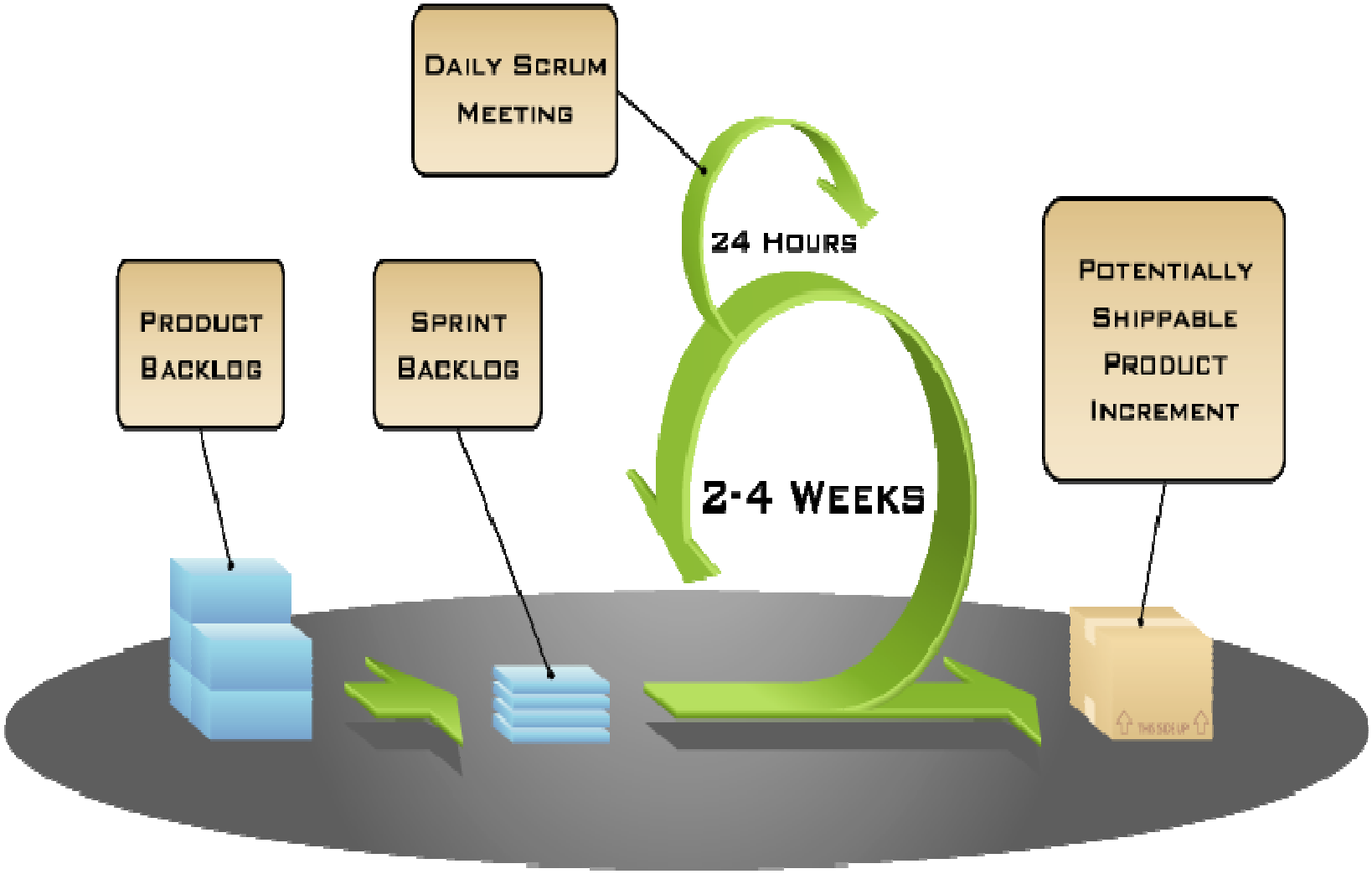


几个管理元素的关系

- 将多个计划有效的集成起来，互相反馈
- 有效管理计划的层次关系，并结合开发的实际



回归一下 Scrum

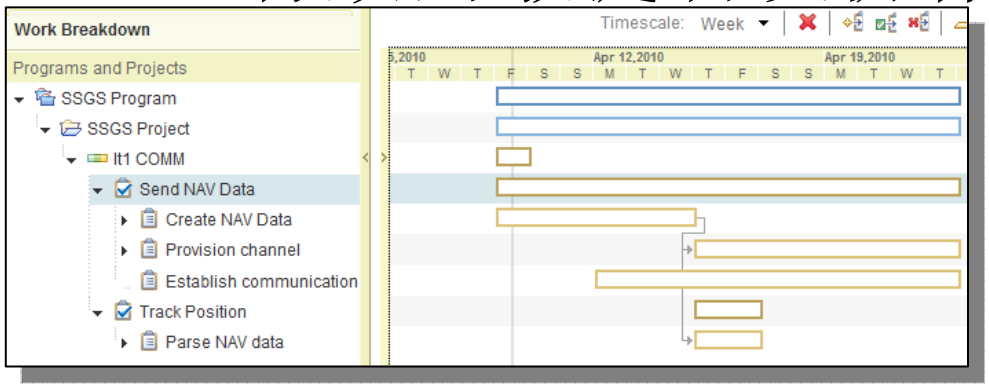


议题

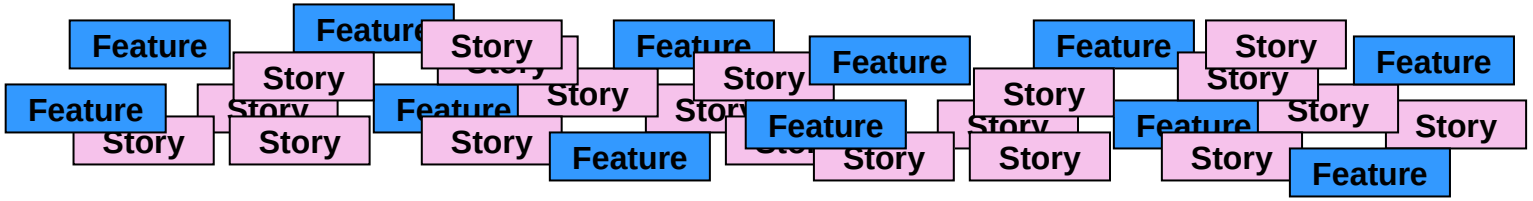
- 软件项目管理的演进
- 敏捷软件项目中的一些最佳实践
- RTC敏捷项目管理的最佳载体

RTC+RPC 有效的敏捷研发协作

Planning in Project Conductor



Common work

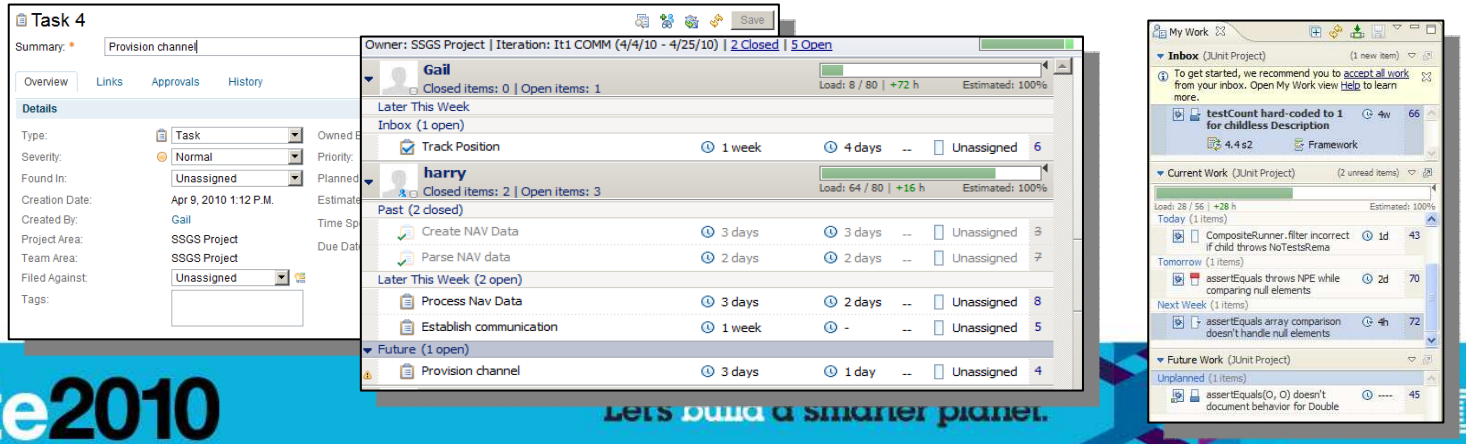


Execution in Team Concert

Estimates, time spent, priority, discussion, etc.

Project- and developer-level plans

Individual developer assignments




Thank
You