敏捷开发

1 概述

传统——瀑布模型：以预见性为原则，以文档驱动，以过程控制为核心

软件开发之道：

- 今天应更侧重于：弹性的开发管理方式，通过初始计划开始工作，项目的资源管理和控制
- 应关注交付的价值

敏捷开发：一种基于更紧密的团队协作、能够有效应对快速变化需求、快速交付高质量软件的迭代和增量的新型软件开发方法。更关注协作、更关注质量、更关注可工作的产品、更关注全才化的专才、基于实践而非理论。

适应而非预测，以人为导向而非过程导向。

敏捷开发方法是一组轻量级开发方法的总称，包含很多具体的开发过程和方法，最有影响的两个方法是极限编程（XP）和 Scrum 开发方法。

敏捷宣言：

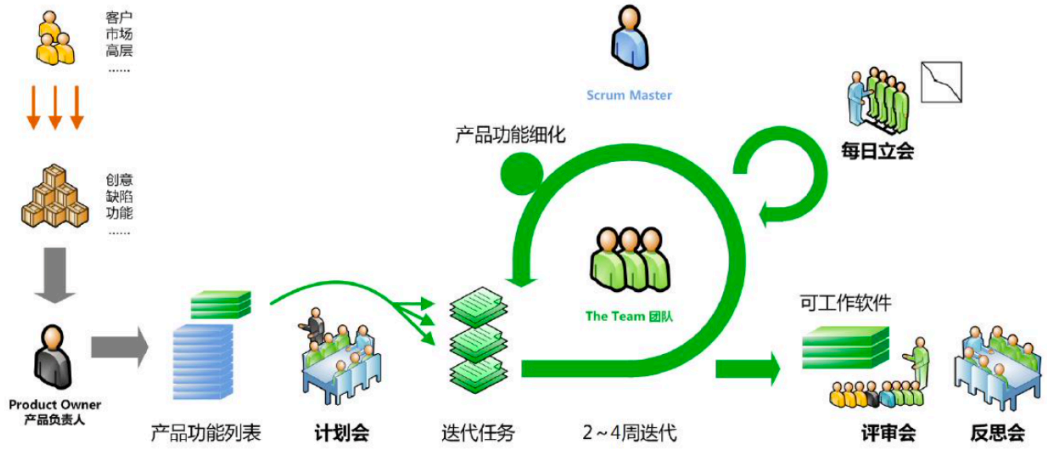
- **个体和交互** 胜过 **过程和工具**
- **可以工作的软件** 胜过 **面面俱到的文档**
- **客户合作** 胜过 **合同谈判**
- **响应变化** 胜过 **遵循计划**
- ……

敏捷核心理念：

- 聚焦客户价值
- 激发团队潜能
- 不断调整以适应变化

2 Scrum 框架

兼具计划性和灵活性，将整个开发过程划分为若干次更小的迭代，每个迭代周期称为一个冲刺（sprint）。



Scrum 迭代开发:

每一个小迭代是一个小的瀑布模型，包括需求分析、设计、实现和测试等活动，结束时都要生成一个稳定和被验证过的软件版本。

关键要点:

- 每一次迭代都建立在稳定的质量基础上，并做为下一轮迭代的基线，整个系 统的功能随着迭代稳定地增长和不断完善。
- 每次迭代要邀请用户代表验收，提供需求是否满足的反馈。
- 在一次迭代中，一旦团队作出承诺，就不允许变更交付件和交付日期；如果发生重大变化，产品负责人可以中止当次迭代。
- 在迭代中可能会出现“分解”和“澄清”，但是不允许添加新工作或者对现有的工作进行“实质变更”。
- 对于“分解”和“澄清”，如果存在争议，那么将其认定为变更，放到产品订单中下一次迭代再考虑。

Scrum 团队角色:

- 产品负责人：定义开发目标以及需要实现的特性和优先级。
- Scrum 主管（master）：保证团队高效而不受打扰地工作，优化工作条件和过程。
- 团队成员：自组织地完成项目开发，使用一切可行手段保证进度和质量。

Scrum 团队组织：民主式结构。小组成员完全平等，名义上的组长与其他成员没有任何区别；大家享有充分的民主，项目工作由全体讨论协商决定，并根据每个人的能力和经验进行适当分配。

- 优点：同等的项目参与权激发大家的创造力，有利于攻克技术难关
- 缺点：缺乏明确的权威领导，很难解决意见分歧

Scrum 制品:

- 产品订单：从客户价值角度理解的产品功能列表

  功能、缺陷、增强等都可以是产品订单项

  整体上从客户价值进行优先级排序

- 迭代订单：从开发技术角度理解的迭代开发任务

  简单环境：可直接把产品订单项分配到迭代中

  复杂环境：可把一个产品订单项分为 Web/后台……软件/硬件……程序/美工……等开发任务

- 可工作软件：可交付的软件产品

  “可交付”应视不同情况前提设定和选定交付标准

  正式产品可能包括使用文档，在新产品开发初期可能只需要交付勉强看到效果的产品

Scrum 活动：

- 迭代规划会议

  在每次迭代（或冲刺）开始时召开，选择和估算本次迭代的工作项

  整个会议分为两个部分：

  - 第一部分以需求分析为主，选择和排序本次迭代需要实现的订单条目
  - 第二部分以设计为主，确定系统设计方案和工作内容
- 每日站立会议

  团队在会议中做计划，协调其每日活动，还可以报告和讨论遇到的障碍。

  任务板帮助团队聚焦于每日活动上，应在这个时候更新任务板和燃尽图。
- 迭代评审会议

  Scrum 团队在会议中向最终用户展示工作成果，团队成员希望得到反馈，并以之创建或变更 Backlog 条目。
- 迭代总结会议

  每一次迭代完成后，都会举行一次迭代总结会议，会上所有团队成员都要反思这个迭代。举行迭代总结会议是为了进行持续过程改进，会议的时间限制在 4 小时。

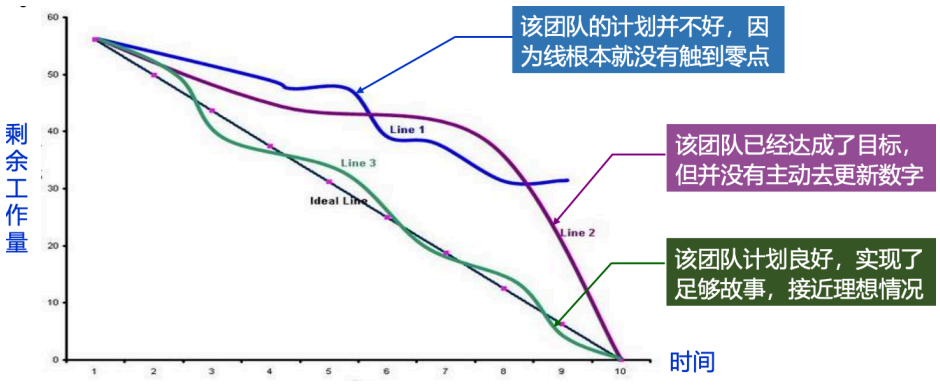
Scrum 规划：

两级项目规划：

- 发布规划：对整个产品发布过程的展望，其结果是产生产品订单
- 迭代规划：只是对一次迭代的展望，其结果是确定包含一次迭代中具体工作任务的迭代订单

可视化管理：

- 任务白板
- 燃尽图：以图形化方式展现剩余工作量与时间的关系



软件实施、维护与演化

1 软件交付工作

1.1 实施

将软件系统部署到客户方的计算机系统上，协助客户准备基础数据，使软件系统顺利上线运行。

- 保证软件复合需求，质量过关

  全面做好测试工作（集成测试、功能测试、性能测试）
- 制定实施计划

  要发布的代码版本、数据库创建方式、基础数据准备方式
- 准备好程序代码和相关文档

  用户手册以及其他系统文档（如需求说明书、设计文档等）

1.2 培训

- 选择合适的培训人员

  经验丰富、了解业务和系统
- 准备好培训内容

  不要临时抱佛脚
- 制定培训计划

  与客户沟通协调，安排时间

1.3 验收

客户对系统进行验收测试，包括范围核实和质量核实。

2 软件演化法则

Lehman 法则	
持续变化	在用的程序持续地经历变化，或逐渐变得不可用
递增复杂性	程序的不断修改将导致结构恶化，增加了复杂性
程序演化法则	程序演化服从统计上的确定趋势和恒定性
组织稳定守恒	编程项目总体活动统计上是不变的
熟悉程度守恒	后续发行对于整个系统功能不会产生很大改变

软件演化策略：

- 软件维护：为了修改软件的缺陷或者增加新功能而对软件进行修改；软件的修改通常发生在局部，一般不会改变整个结构。
- 软件再工程：为了避免软件本身退化而对软件的一部分进行重新设计、编码和测试，以便提高软件的可维护性和可靠性等。

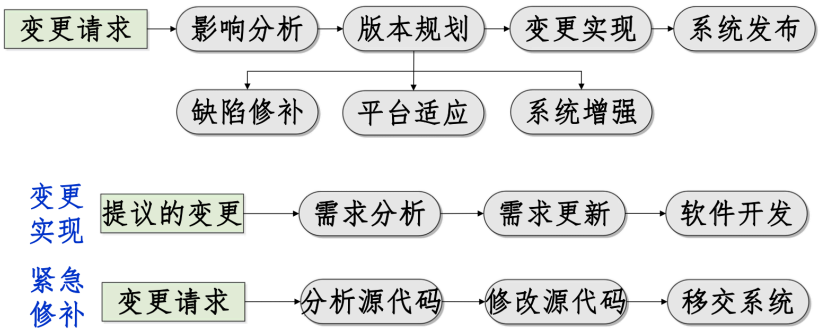
3 软件维护

类型：

- 改正性维护：修改软件缺陷或不足
- 适应性维护：修改软件使其适应不同操作环境，主要包括硬件变化、操作系统变化或者其他支持软件变化等
- 完善性维护：增加或修改系统功能，使其适应业务的变化

软件维护成本是很昂贵的。

软件维护过程：



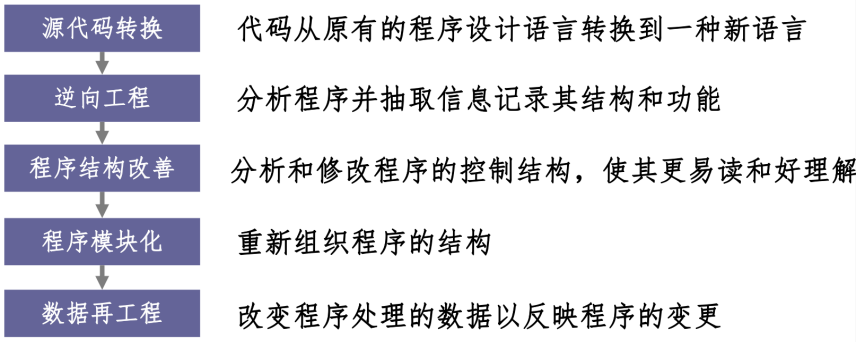
4 软件再工程

- 重新构造或编写现有系统的一部分或全部，但不改变其功能
- 在大型系统中某些部分需要频繁维护时，可应用软件再工程
- 再工程的目的是努力使系统更易于维护，系统需要被再构造和再文档化

优势：

- 减少风险：重新开发一个在用的系统具有很高的风险，可能会有开发问题、人员问题和规格说明问题
- 降低成本：再工程的成本比重新开发软件的成本要小得多

再工程过程：


源代码转换
代码从原有的程序设计语言转换到一种新语言
逆向工程
分析程序并抽取信息记录其结构和功能
程序结构改善
分析和修改程序的控制结构，使其更易读和好理解
程序模块化
重新组织程序的结构
数据再工程
改变程序处理的数据以反映程序的变更