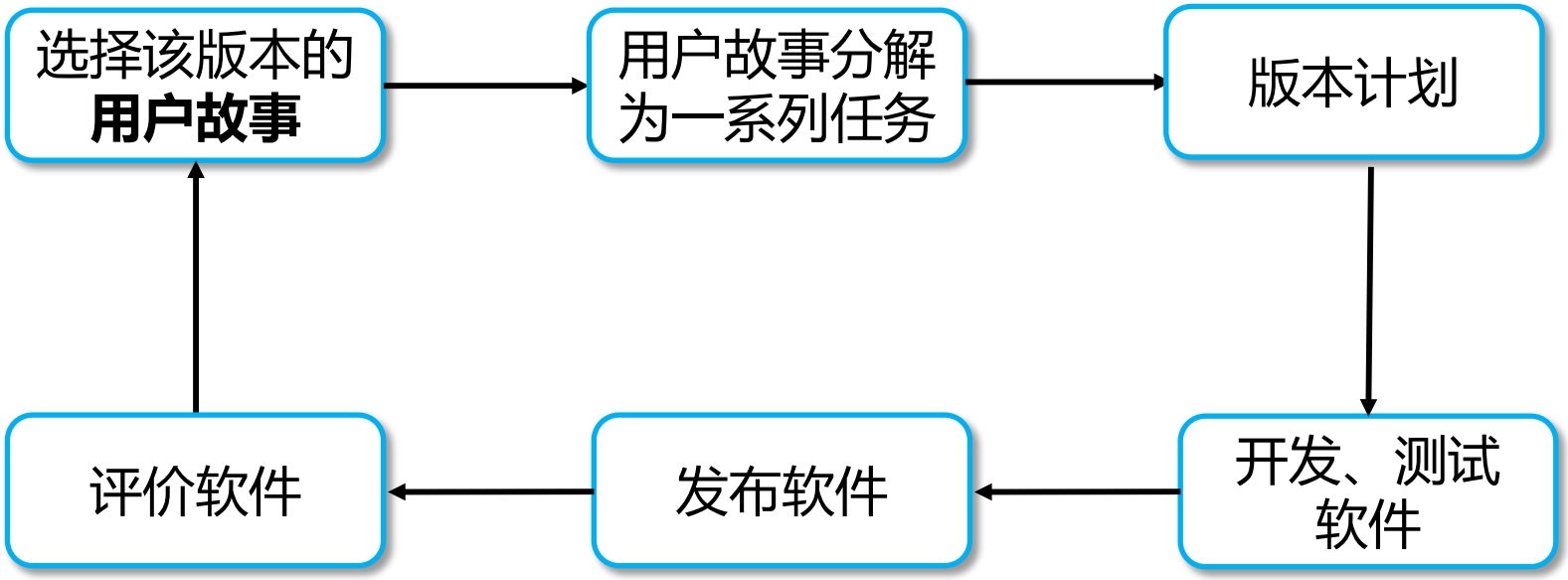
极限编程（eXtreme Programming）

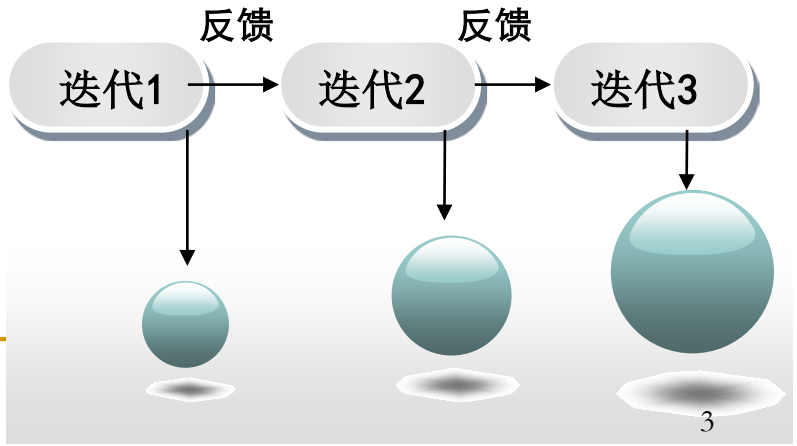
- 应用最广泛的敏捷方法
- Kent Beck
- 把好的开发实践（迭代式开发）运用到极致
  - 新的版本可能每天构建多次
  - 每两到三周完成一次迭代过程，交付出目标系统的一个可工作的版本
  - 对于每次的构建都要进行测试
  - ……

XP发布周期



XP的迭代—有节奏地小步快跑

- 划分多个小的迭代（2-3周）
- 每一次迭代都由需求分析、设计、实现和测试等多个活动组成
- 每一次迭代都可以生成一个稳定和被验证过的软件版本
- 迭代推荐采用固定的周期，迭代内工作不能完成，应当缩减交付范围而不是延长周期



用户故事（user story）

- 用户是XP团队的一部分，负责提出需求
- 用户需求表示为**场景**或**用户故事**
  - 描述将要开发的软件所需要的输出、特性以及功能
- 用户故事写在卡片上，开发团队把用户故事分解为一系列任务
- 任务可作为进度安排和工作量估算的基础

实例：“开药”故事卡

开药

Kate是一名医生，她要给她的病人开药。该病人已经在她的电脑里有看病记录了，因此，她点击开药按钮，选择“当前药物”、“新药”或者“处方集”。

如果她选择“当前药物”，则确认剂量。如果要修改剂量，她输入相应的剂量，并确认该处方。

如果她选择“新药”，输入药物名称的首字母，系统显示一系列可能的药物，她选择需要的药物，并输入剂量，再确认该处方。

如果她选择“处方集”，系统显示检索框，她检索需要的药，然后选择该药，并输入剂量，再确认该处方。

系统始终检查剂量是否在允许的范围内，如果不在，Kate需要修改剂量。

Kate确认处方后，系统要求她再次检查处方，她可以点击“OK”或者“Change”。如果选择“OK”，则该药方存入数据库。如果选择“Change”，则再次进入“开药”流程。

实例：“开药”任务卡

任务1：改变药物的剂量

任务2：处方集选择

任务3：剂量核对

剂量核对用于检查医生有没有开出过小或者过大危险剂量的药物。

使用药物ID查询处方集，并从中检索出该药物的推荐最大、最小剂量。

核对开出的剂量是否在最小和最大剂量范围之内，如果超出该范围，报相应的错误信息。如果在范围之内，则剂量核对通过。

项目开发实例

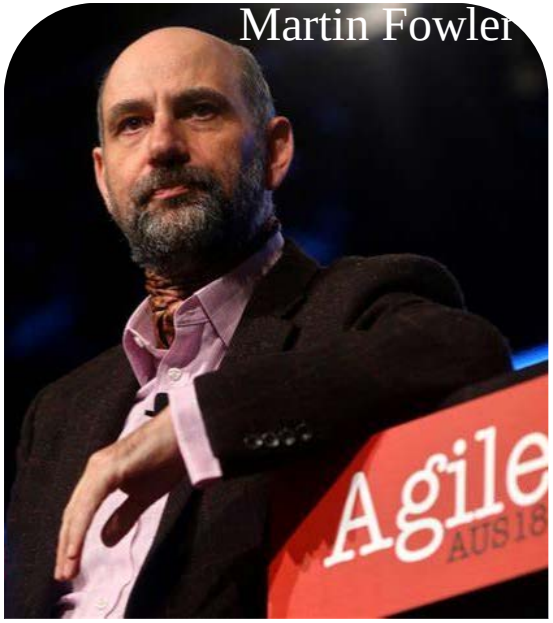
- 某个项目，系统核心是类继承体系，咨询顾问发现整个体系相当混乱
  - 在一个类的不同地方出现重复的代码
  - 好几个子类做相同的事情
- 项目经理不愿意调整
  - 进度排得很紧
  - 原来代码运行正常，调整后代码功能无增加
- 程序员花两天时间调整好继承体系
  - 删掉一半代码，功能未受影响
- 咨询顾问建议调整系统的其它核心部分
- 项目经理不愿调整，最终由于代码太复杂，无法调试，项目失败。

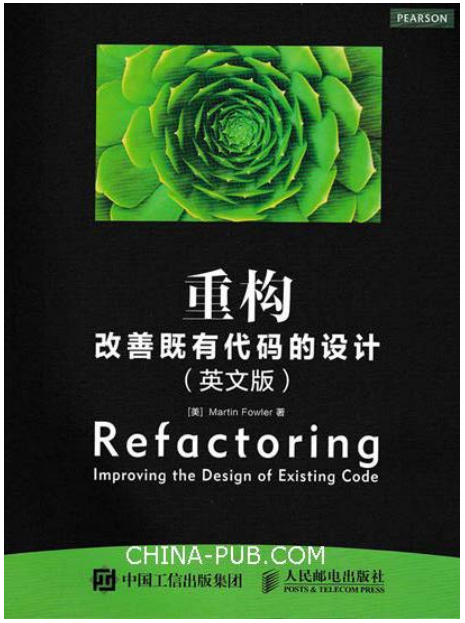
两种情形

- 情形1：代码里定义大量的类、接口、方法，类与类，类与接口之间很多是继承和实现的关系，方法的代码行数很少，一般不超过20行，代码就是方法之间的调来调去。
- 情形2：一个方法几十上百甚至两三百行都是最基本的语句构成，很少调用自己的方法。

- 前者在结构上更清晰，通过类视图就可看出设计意图，并且总的代码量不会高于后者
- 后者代码量庞大，代码冗余现象严重，结构不清晰，很难维护。

重构（Refactoring）





- 所谓重构是这样一个过程：在不改变代码外在行为的前提下，对代码做出修改，以改进程序的内部结构。

重构（Refactoring）

Kent Beck

I am not a great programmer;
I am just a good programmer
with great habits.

Refactoring helps me be much more effective at writing robust code.

代码坏味道（Bad Smell in Codes）

- 重复代码（Duplicated Code）
- 过长函数（Long Method）
- 过大的类（Large Class）
- 过长参数列（Long Parameter list）
- Switch表达式
- ……

重构方法

- 提炼函数（Extract Method）
- 函数改名（Rename Method）
- 搬移函数（Move Method）
- 内联函数（Inline Method）
- 内联临时变量（Inline Temp）
- ……