21-23

chp21 协同构建

1 协同开发实践概要

- 是其他质量保证技术的补充
- 宣传公司文化和编程知识
- 集体所有权适用于所有形式的协同构建

2 结对编程

关键点

- 制定统一的编码规范
- 主动性，不参与编程的也要参与其中
- 简单问题不需要结对编程
- 有规律的对结对编程人员和任务进行轮换
- 鼓励双方跟上对方的步伐
- 找对脾气的人在一起
- 避免新手组合
- 指定一个组长

好处

- 提高人员承压能力
- 改善代码质量
- 加快开发进度
- 其他

3 正式检查

- 预期结果

角色

- 主持人
- 作者
- 评论员
- 记录员

详查的一般步骤

- 计划

- 概述
- 准备
- 详查会议
- 详查报告
- 返工
- 跟进
- 第3个小时的会议

- 详查中的自尊心
  - 发现缺陷、而不是寻找替代方案
  - 作者不应在过程中为所谓的缺陷辩护，但是这不等于认同这个缺陷，可以会后自己思考，最终决定是否更改

- 4 其他
  - 走查（walk-througs）
  - 代码阅读
    - 2个或3个人
    - 会议前参会者看代码 1天1000行
    - 主持会议讨论发现的问题
    - 修复问题
  - 公开演示

chp22 开发者测试

- 1 在软件质量中的角色
  - 构建中测试
- 2 推荐方法
  - 先测试还是后测试
    - 有利于减少错误，只是调整开发顺序
  - 开发者测试的局限性
    - 开发者倾向于干净测试（clean tests),而不是肮脏测试（dirty tests)
    - 对覆盖率过于乐观
    - 容易忽略复杂的测试类型
      - 100%语句测试->100%分支测试

- 3 测试技巧锦囊
  - 不完整的测试
  - 结构化的基础测试
    - 思想：测试每条语句至少执行一次
  - 数据流测试
  - 等价类划分
    - 如果两个用例测试验证同一个问题，那么用一个就行了
  - 猜测错误
  - 边界值分析
  - 几类坏数据
  - 几类好数据
  - 采用容易手工检查的测试数据
- 4 典型错误
  - 哪些类包含最多的错误
    - 80%的错误存在于20%的类中||50%的错误存在于5%的类中
    - 针对容易出错的部分进行重写
  - 错误的分类
  - 不完善的构建过程引发错误所占比例
  - 你期望能发现多少错误
  - 测试本身的错误
- 5 测试支持工具
  - 为测试各个类构造脚手架（junit）
    - dummyClass（模仿类的原有功能）
    - 驱动函数
  - diff工具
  - 覆盖率监视器
  - 系统干扰器
  - 错误数据库
- 6 改善测试过程
  - 回归测试
    - 维护核心用例集

- 自动化测试

- 7 保留测试记录

chp23 调试（debugging）

1 调试概述

- 在软件中扮演的角色
- 调试效率的巨大差异
  - 软件质量越高，开发成本越低
- 让你有所收获的缺陷
- 1种效率低下的调试方法
  - 迷信编程：如果你写的程序除了出了问题， 那就是你自己的原因，和计算机无关。请对他负责

2 寻找缺陷

- 高效程序员只需花费低效程序员20分之1的时间
- 科学的调试方法
  - 1 稳定复现错误
  - 2 确定错误的来源（分析 假设 验证）
  - 3 修补缺陷
  - 4 对所修补的地方进行测试
  - 5 查找是否还有类似的错误
- 寻找缺陷的小建议
  - 增量集成
  - 新老版本替换
  - 同他人讨论问题
  - 抛开问题休息一下（潜意识解决问题）

3 修正缺陷

- 动手前要理解问题
- 理解程序本身，而不仅仅是问题
- 验证对错误的分析
- 放松一下
- 动手前保存历史代码方便回滚

- 从根本解决问题
- 修改代码前必须要有适当的理由
- 增加验证性的功能测试

4 调试中的心理因素

- 心理取向如何导致调试的盲目

5 调试工具