第九章 质量与变更管理

2018年12月4日 9:17

- **测试**不仅仅是程序的调试，也是质量管理的一部分:
  - **质量保证**是将质量构建到软件系统中的主动方法，采用主动的方式来建造有质量的软件系统
  - **质量控制**(主要是反应性的)测试软件系统质量的方法，采用（大部分被动）方式来测试软件系统的质量
- **变更管理**是整个项目管理的基础——必须记录所有的变更请求，追踪每一个变更对所开发的人工制品的影响，在变更完成之后重新进行测试
- 测试驱动的软件开发：通过明确的要求将质量构建到软件系统中，即必须在应用程序代码之前编写测试用例，并且应用程序必须通过测试才能保证质量
- Traceability强调测试和变更管理

1. 质量管理
   - **质量管理**与人员管理、风险管理以及变更管理等活动属于整个软件过程管理的一部分，包括人员管理，风险管理，变更管理等
   - **项目管理**是例外，他可以和质量管理**并行执行**，如项目进度计划、预算估算、项目进度跟踪
   - 可适应性——可理解，可维护，可扩展
   - 质量保证和控制最终目的：保证让软件没有错误
   - 软件质量：
     - 正确性
     - 可靠性
     - 健壮性
     - 性能
     - 可用性
     - **适应性：**
       - 易懂性（可理解）：说明文档，程序注释
       - 可维护性
       - 可伸缩性（可扩展）：scalability(evolvability)
     - 可复用性：软件接口定义的合理性
     - 可移植性
     - 协同工作的能力:interoperability
     - 生产力:productivity
     - 时效性:timeliness
     - 可视性

   a. 质量保证
      i. 在构建软件的时候采取一系列措施少犯错
      ii. 质量保证是制定 能够保证最终产品质量的 质量过程和质量标准

iii. 软件质量保证小组（SQA team）——不是最初的开发人员！

iv. 技术：

1）检查表：Checklists

- 在开发过程中需要仔细检查的预定义待办事项列表。
- 过程因项目而异——检查表不能永远固定
  - 此外，需要修改“基线”检查列表，以适应新的IT技术和IT开发范式中的变化。

2）评审:Reviews——手工形式的测试

- 目的是评审一个工作产品或过程

a）走查Walkthroughs

- 可以在任何开发阶段进行的一种正式的头脑风暴评审
- 一个由开发者组成的友好会议，精心策划，目标明确，议程明确，会期明确，成员明确
- 在会议前几天，与会者将收到要审查的材料
- 在会议期间，需要**找出问题**，但没有试图解决
- 已确认的问题将输入到走查问题列表中
- 开发人员使用该列表对已评审的软件产品或过程进行更正
- 后续走查可能是必要的

b）审查Inspections

- 审查也是一次友好的会议，但要在项目管理人员的密切监督下进行，目的也是找出所有故障，验证这些故障是否是真正的故障，然后记录这些故障，并安排何时，由谁完成这些故障的修正。
- 目的还在于识别缺陷，验证它们实际上是缺陷，记录它们，并计划何时以及由谁来修复它们
- 与走查不同，审查不太频繁，次数较少，可能只针对选定的关键问题，而且更正式、更严格rigorous
- 在会议期间，这些defects被识别、记录和编号
- 会议结束后，主持人立即准备在变更管理工具中记录的**缺陷日志**
- 开发人员通常被要求快速解决缺陷，并在**变更管理工具**中记录解决方案
- 主持人moderator在与项目经理协商后，将开发模块提交给组织中的**软件质量保证(SQA)**组。

3）审核/审计：Audits

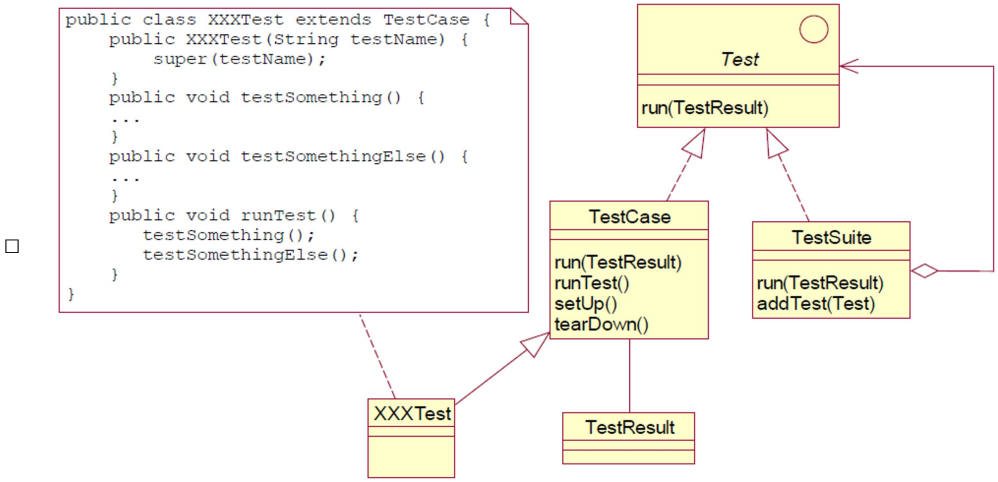
- 一种质量保证过程，以传统的会计审计为模型，在所需资源方面与之相似。一般在项目开始之前/之后审计
- 定位在整个软件过程管理
  - 它将风险管理放在自己的观点中
  - 它与IT对企业的战略重要性相关，并解决IT治理问题
  - 它在企业使命和业务目标的上下文中查看被审计项目与IT投

资的一致性

- 审计与其他QA技术的区别
  - 被审核产品或过程的生产者通常是一个团队而不是个人
  - 审核时生产者可以不在场
  - 审核需要的检查表比会议和评审要多
  - 审核可以是外部的，由组织外的审核者完成
  - 审核可持续一天到一个星期或更长的时间，但始终围绕严格定义的范围和目标

v. 测试驱动开发

- 其思想是在**开发**(设计和编程)应用**程序代码**(被测试的单元)**之前编写测试用例**和脚本以及测试程序
- **应用程序代码**是作为对测试代码的响应编写的，测试代码可用来测试应用程序代码
- 优点：
  - 允许在程序员编写应用程序代码的第一行之前澄清用户需求(和用例规范)
  - 驱动实际上是软件开发，而不仅仅是软件验证
  - 由测试框架(如JUnit)支持



b. 质量控制——主要的软件测试活动，跨越了整个系统开发生命周期

i. 发现软件中潜在的错误

ii. **质量保证是为了建造有质量的软件产品或过程，而质量控制是为了测试软件产品或过程的质量。**

iii. **质量保证主动，质量控制被动**

iv. **质量保证策略级，质量控制战术级或操作级**

- 测试概念和技术
  - 为了进行测试，必须采用**测试脚本**的形式来实现测试用例
  - 这些测试用例规定了判定一个软件产品是否满足其测试需求所需的**步骤**（测试人员必须遵守）和**验证点**（测试人员必须解决的那些问题），可将多个测试脚本的组合成较大的**测试集**。
  - 人工测试和自动化测试
    - 自动化测试由虚拟的测试人员完成，是一个**捕获/回放**工具，自动

化测试被广泛应用于回归测试

- **回归测试**是在不断的代码迭代后重新执行相关的验收测试，回收测试的目的是保证对代码的迭代补充不会产生非故意的副作用

- 各开发阶段对应的测试活动

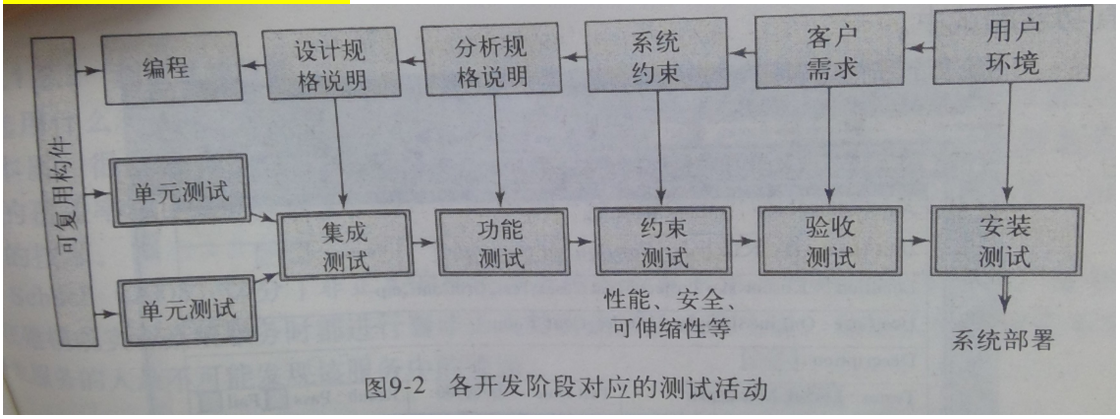


图9-2 各开发阶段对应的测试活动

- 测试系统服务
  - 静态测试（基于非执行）
    - 走查
    - 审查
  - 动态测试（基于执行）
    - **针对规格说明的测试**
      - 也称为黑盒测试，功能测试和输入/输出驱动测试
      - 有利于找出其他方法一般难以发现的故障，尤其是可以找出遗漏的功能
    - **针对代码的测试**
      - 也称为白盒测试，玻璃盒测试、逻辑驱动测试和面向路径的测试
  - 集成测试
    - Bottom-up testing
      - requires ***drivers*** (routines that call a particular component and pass a test case to it)
    - Top-down testing
      - requires ***stubs*** (routines to simulate the activity of the missing component; they answer the calling sequence and pass back output data that lets the testing process continue)
    - Big-bang integration
    - Sandwich integration
- 测试系统约束（大部分是**动态测试**）
  - 用户界面测试
  - 数据库测试
  - 权限测试

        - 程序的用户界面应该能够动态地配置自己，以便与当前用户的授权级别相对应(通过用户id和密码验证)
        - 主要是服务器的访问权限，数据库访问权限
        - 可以有单个用户授权或组授权
        - 大多数DBMS还支持角色级别
        - 需要在应用程序数据库旁边设置一个**授权数据库**，以便存储和操作客户机和服务器权限
    - 性能测试
        - 峰值负载
    - 压力测试
        - 和性能测试一起做
    - 容错测试
        - 与DBMS恢复过程密切相关
    - 配置测试
    - 安装测试
        - 扩展配置测试
        - 验证系统在每个安装的平台上运行正常

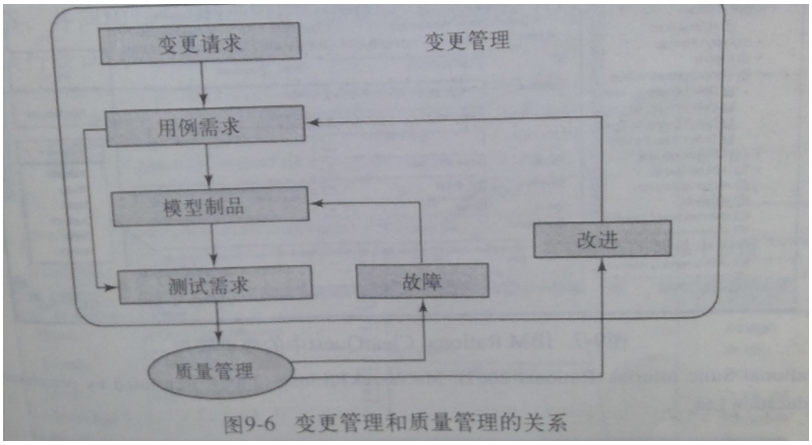
2. 变更管理
    - 

图9-6 变更管理和质量管理的关系
    - 质量控制可以找出需要修正的故障defect
    - 采用成本-效益分析技术来确定项目追踪的范围和深度
    - 捕捉-回放工具可以自动执行测试脚本

a. 工具与管理变更请求
    i. 提出一次变更需要以正式的变更请求提出

b. 可追踪性Traceability
    - 是测试和变更管理的基础，其目的是捕获、链接和追踪每一个重要的开发制品，包括需求。
    - 最终目的是保证震哥哥为念计中各种文档和模型的正确性和一致性，如从需求文档到技术文档，再到用户文档