软件测试

授课教师：张剑波

授课班级：111161-2班

2019年3月

Chapter2 软件测试技术

内容提要

- 静态测试
- 动态测试
  - 经典测试方法
  - 基于质量特征的测试
  - 基于经验的测试

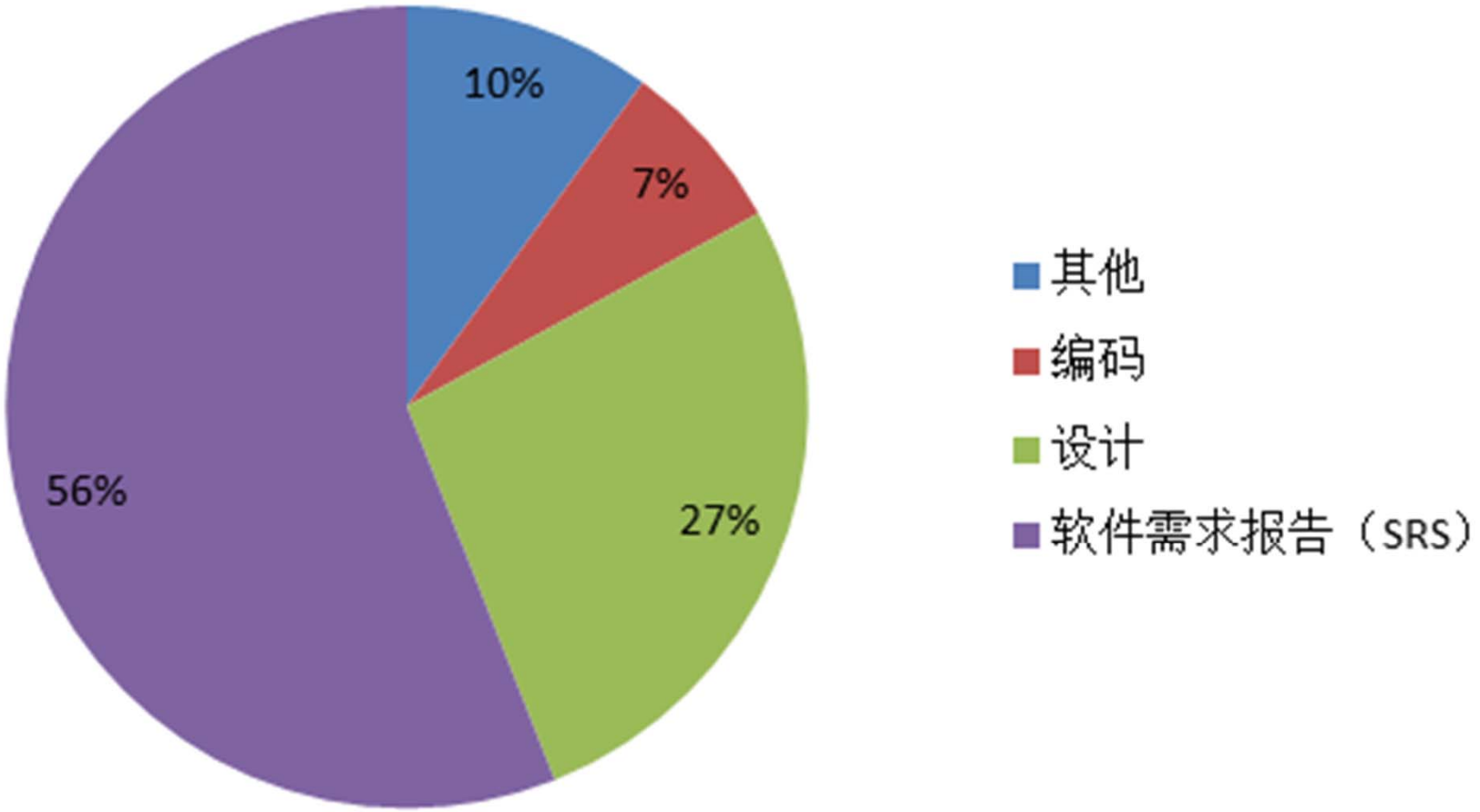
1、软件缺陷

- 缺陷的特点
  - 雪崩效应：缺陷数目会随着软件开发过程的深入而不断地增加
  - 成本放大效应：不同阶段发现和修复缺陷的成本不同，后期会急剧增加
  - 集群效应：测试过程中发现的缺陷在测试对象的不同功能模块分布，测试对象20%模块中可以发现80%的缺陷（二八原则）

软件缺陷的产生

- 缺陷的来源
  - 疏忽造成的错误(Carclessness Defect，CD)
  - 不理解造成的错误(Misapprehend Defect，MD)
  - 二义性造成的错误(Ambiguity Defect，AD)
  - 遗漏造成的错误(Skip Defect，SD)
- MD、AD、SD三类缺陷主要存在于软件开发的前期阶段，如需求分析阶段、设计阶段、编码阶段

软件缺陷的分布



2、基于缺陷分类的测试

- 当面临需求规格说明不全、模糊甚至没有需求的时候，要求测试人员确定测试什么，或者确定针对测试对象的测试思路是十分困难的。
- 基于“缺陷分类”的测试将可以较好的解决该问题
- 举例说明
  - 2名测试人员
  - 没有缺陷分类和有缺陷分类
  - 指出“网上购书系统的购物车”在哪些地方可能存在缺陷和问题？

没有缺陷分类指导的测试

- 得到的测试点：
- 1. 往购物车内添加条目失败，例如：增加某本书；
- 2. 移除购物车内的条目失败，例如：移除某本书；
- 3. 无法修改购物车条目相关的定单；
- 4. 购物车应用程序无法和一些浏览器兼容；
- 5. 购物车内选中的条目无法清楚的显示该物品的图片；
- 6. 从客户端可以修改购物车内物品的价格；
- 7. 由于安全问题，客户的信用卡信息可以明文显示；
- 8. 点击"确定"按钮，出现"PAGE NOT FOUND"错误；

缺陷分类指导的测试

- 得到的测试点：
- 1. 功能性问题
  - 往购物车内添加条目失败，例如：增加某本书；
  - 移除购物车内的条目失败，例如：移除某本书；
  - 无法修改购物车条目相关的定单；
  - 从客户端可以修改购物车内物品的价格
- 2. 计算问题
  - 购物车内增加或者删除一个物品，物品总价并不能实时更新；
  - 物品的折扣不能正确的计算；
  - 物品的快递费用，不能反映购物车内物品数目不同的时候给予的减免

缺陷分类指导的测试（续1）

- 3. 易用性问题
  - 用户不能直接从搜索的页面将物品放入购物车；
  - 用户无法直观的得知当前购物车中物品的数量；
  - 用户需要操作很多的步骤才能完成一个定单；
  - 在往购物车内添加物品、删除物品以及更新物品的时候很不方便；
  - 不能直观的得到当前物品的总的价格；
  - 无法找到"帮助"菜单
- 4. 网络问题
  - 无法连接到物品库存服务器；
  - 无法连接到客户数据库服务器；
  - 网络拥塞的时候导致服务器重启

缺陷分类指导的测试（续2）

- 5. Web服务器问题
  - WEB服务器超过最大的在线人数；
  - 登录服务器需要花费很长时间，过于漫长的连接；
  - WEB服务器连接超时；
  - WEB服务器容量不足；
  - WEB服务器资源耗尽，例如：CPU和内存等；
- 优点
  - 在相同的时间和参与人员的情况下，基于缺陷分类的指导可以得到更多的测试点
  - 通过结构化的缺陷分类作为工具支持，得到的测试点具有更好的测试范围和测试覆盖率
  - 测试人员可以更好的将精力集中在缺陷分类罗列的类型上面，因此可以得到更加全面和详细的测试点

基于缺陷分类的测试规则

- 规则1：发现的缺陷的数量说明不了软件的质量。
  - 缺陷的数量大，只能说明测试的方法很好，思路很全面，测试工作有成效。
  - 极端1：测试中发现的缺陷，绝大多数都是属于提示性错误、文字错误等，错误的等级很低，而且这些缺陷的修改几乎不会影响到执行指令的部分，而软件的基本功能或者是性能，发现很少的缺陷。
  - 极端2：如果在测试中发现的缺陷比较少，但是这些缺陷都集中在功能没有实现，性能没有达标，动不动就引起死机、系统崩溃等现象，而且，在大多数的用户在使用的过程中都会发现这样的问题。

基于缺陷分类的测试规则（续1）

- 规则2：缺陷要分类统计。
  - 在某些模块，执行的测试用例多，但是没有成比例的发现很多缺陷，所以这些模块是比较成熟的
  - 某些模块执行的测试少，却发现了更多的缺陷，这些模块修复的地方，或者发生功能变更的可能性大，所以将成为质量不稳定的关键点
  - 在以后的回归测试中，应该在质量不稳定的模块中投入更多的人手和时间，进行更全面的测试，其它模块就相应减少测试工作的投入

基于缺陷分类的测试规则（续2）

- 规则3：不要指望找出软件中所有的缺陷。
  - 该在什么时候停止软件测试，发布软件；
  - 软件中的缺陷既然是不可能全部发现的，就不要指望找出软件中全部的缺陷，当它足够少（各公司的定义是不同的）的时候，就应该停止测试了。
  - 例如：产品在发布的时候，一般都会要求，各模块致命的缺陷不得有2个，严重的缺陷不得超过5个，轻微的缺陷不能多于5个，残留的缺陷总数不得多于10个等
- 规则4：只依赖缺陷的趋势也可能有问题。
  - 软件不同阶段，各模块的质量和软件整体的质量是“不对称”的

3、缺陷模式

- 2000年以来，基于“缺陷模式”的软件测试首先在美国发展起来
- 具有如下特点：
  - 针对性强：如果说某种模式的缺陷是经常发生的，并且在被测软件中是存在的，则面向缺陷模式的测试可以检测出此类缺陷
  - 基于缺陷模式的软件测试技术往往能发现其他测试技术难以发现的故障，如内存泄漏缺陷、空指针引用缺陷
  - 工具自动化程度高以及测试效率高

缺陷模式

- 缺陷模式是和语言本身相关的，不同语言有着不同的缺陷模式。
- 以C++语言和Java语言为例
- 将软件的缺陷模式分为四个层次：
  - 故障模式
  - 安全漏洞模式
  - 缺陷模式
  - 规则模式

（1）故障模式

- 此类缺陷是故障，一旦产生，就会导致系统出错
- 分为以下几种类型：
  - 内存泄漏模式：C++
  - 资源泄漏模式：Java
  - 指针使用错误模式：空指针
  - 数组越界模式
  - 非法计算模式：如DIV(X,0)
  - 使用未初始化变量模式
  - 死循环结构模式
  - 死锁模式

（2）安全漏洞模式

- 为攻击者攻击软件提供了可能。一旦软件被攻击，系统就可能发生瘫痪，所造成的危害较大
- 从源代码角度，分为以下几种类型：
  - 缓冲区溢出模式：数据复制、格式化字符串
  - 被污染的数据模式：外部的全局变量、输入函数
  - 竞争条件模式：两种不同的I/O调用同一文件进行操作，可能发生在用户拥有不同的运行权限的程序中
  - 风险操作模式：不恰当地来使用了某些标准库函数，可能会带安全隐患；如rand伪随机值

（3）缺陷模式

- 未必会造成系统错误，但可能会隐含某些故障，或者是由于初级软件工程师不理解造成的。
- 分为以下几种类型：
  - 性能缺陷模式：包括使用低效函数/代码、使用多余函数等模式，会降低系统的性能，建议采用更高效的代码来完成同样的功能；
  - 疑问代码模式：让人费解的代码。主要指代码中容易引起歧义的、让人迷惑的编写方式。该类模式主要包括不合适的强制类型转换等

（4）规则模式

- 软件开发总要遵循一定的规则，某个团队也有一些开发规则，违反这些规则也是不允许的。
- 分为以下几种类型：
  - 代码规则
  - 复杂性规则
  - 控制流规则
  - 命名规则
  - 可移植性规则
  - 资源规则

4、探索性测试

- 探索性强调测试人员的主观能动性，抛弃繁杂的测试计划和测试用例设计过程，强调在碰到问题时及时改变测试策略。
- 探索性测试强调“测试设计”和“测试执行”的同时性，这是相对于传统软件测试过程中严格的“先设计，后执行”来说的。
- 测试人员通过测试来不断学习被测系统，同时把学习到的关于软件系统的更多信息通过综合的整理和分析，创造出更多的关于测试的主意。

Exploratory Software Testing

- 在概念上说，探索性测试是一种测试风格，而不是某一种具体的测试方法（等价类测试/边界测试等），它强调系统软件学习、设计测试用例以及测试执行同时进行，它适用于要求在短时间内以及测试需求频繁变更下寻找出重大缺陷的情况。
- 目标
- 1）理解应用程序如何工作，接口看起来怎样，实现了什么功能？
- 2）强迫软件展示其全部能力
- 3）找到缺陷。

为什么进行探索性测试

- 如果想发现应用程序业务逻辑相关的缺陷，充分发挥主观能动性的手工测试也是最理想的选择。但随着软件测试的发展，手工测试越来越倾向于精心策划。
- 现代软件项目是庞大的人员成本是有限的，如何保证在有限的时间内做最正确的事，需要手工测试在开展之前，有明确的战略和方向，但又必须预留一定的发挥空间让每个人的大脑可以充分运转起来，在测试的过程中随机应变。我们把这种测试方式称作探索性测试：它鼓励测试人员一边测试、一边计划、他们使用在测试中收集到的信息，影响自己进行测试的实际方式。

漫游测试

- 按照软件特性将软件分为相互重叠的区域
- 商业区
  - 工作得以完成的地方
  - 侧重于测试软件的重要基础特性，方法有指南法、极限法和遍历法等
- 历史区
  - 年代久远或发生过重大事件的建筑存在的地方
  - 测试遗留代码，如修复已知缺陷的代码，主要测试方法有恶邻法和博物馆法

漫游测试（续1）

- 娱乐区
  - 休闲活动的地方，也是使用频繁的地方
  - 用于测试一些辅助特性，方法有配角法、深巷法和通宵法
- 旅游区
  - 旅游者聚集的地方，为了到此一游
  - 注测试软件的各种功能，方法有收藏家法、长路径法、测一送一法等

漫游测试（续2）

- 旅馆区
  - 旅游者的休息之处
  - 测试在测试计划中较少描述的次要及辅助功能，方法有取消法和懒汉法旅游区
- 破坏区
  - 测试一些能破坏软件系统的行为，方法有反叛法和强迫症法

探索性测试方法

- 自由式探索式测试
  - 对一个应用程序的所有功能，以任意次序、使用任何如数进行随机探测，而不考虑哪些功能是否必须包括在内
- 基于场景的探索式测试
  - 对传统场景测试的补充把脚本的应用范围扩大到了更改、调查和改变用户执行路径的范畴
- 基于策略的探索式测试
  - 将自由式测试探索式与具有测试老手的经验、技能和感知融合在一起
- 基于反馈的探索式测试：上一次测试
  - 测试人员通过咨询那些覆盖指标(代码覆盖、用户界面覆盖或者其中的某一些组合)来选中新的测试用例，以使这些覆盖指标得以提高

本节小结

- 软件缺陷
- 基于缺陷分类的测试
- 缺陷模式
- 探索性测试