SPASVO

制定测试策略

目录

- 测试风险与测试策略
- 常见要素

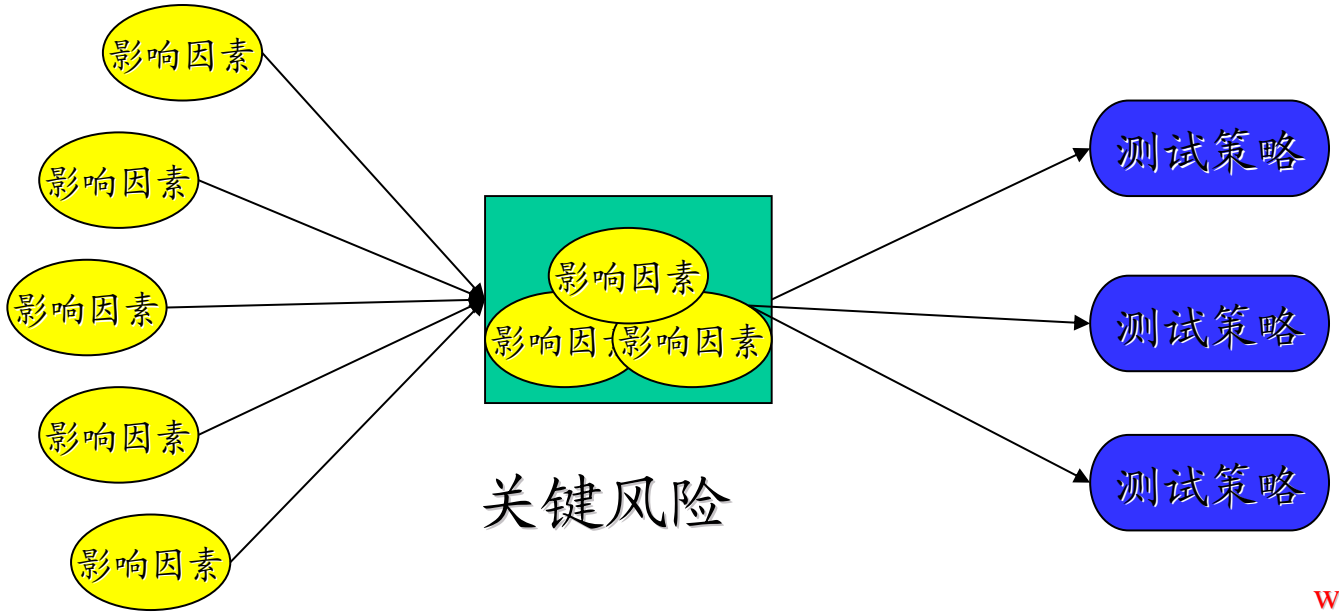
什么是测试策略？

- 策略：
  - 在一定的政治路线指导下，根据具体条件而规定的斗争原则、方式和方法
- 软件测试策略
  - 在一定的软件测试标准、测试规范的指导下，依据测试项目的特定环境约束而规定的软件测试的原则、方式、方法的集合

- 测试策略为测试提供全局分析，并确定或参考：
  - 项目计划、风险和需求；
  - 相关的规则、政策或指示；
  - 所需过程、标准与模板；
  - 支持准则；
  - 利益相关者及其测试目标；
  - 测试资源与评估；
  - 测试层次与阶段；
  - 测试环境；
  - 各阶段的完成标准；
  - 所需的测试文档与检查方法。

测试策略的来源

- 影响因素与测试策略



前言

- 测试的目的就是减少计算机系统中可能存在的风险
- 策略的作用是什么？
  - 策略必须针对于这些风险并提供减少风险的过程
- 这些风险确定了测试的目标

测试策略的两个组件

- 组件1：风险因素
- 组件2：测试阶段

SPASVO

标准风险因素

风险因素	内容
正确性	确保应用系统的数据输入、处理和输出的正确和完整
文件完整性	确保进入应用系统的数据在返回时不被更改
合法性	确保在管理范围内的数据被处理。在应用系统中，对于事务的处理同时有普通和特殊的认证方式，普通人正方式管理那些执行不同种类的商业事务的权限，特殊认证方式管理特殊行为的权限
审计跟踪	证实处理过程的能力。对数据的处理可以通过保留充分的证据来证实他的正确性、完整性、及时性和合法性

SPASVO

续前表

风险因素	内容
处理的连续性	在有问题出现的情况下持续处理的能力。处理的连续性确保了由于某些问题的出现破坏系统的完整性后，必要的规程和备份信息在恢复处理时可用
服务水平	确保期望的结果在用户可接受的时间内能够实现
访问控制	确保应用系统资源在意外或有意的更改、破坏、滥用、暴露等情况下能够受到保护
符合性	确保系统设计与组织策略、原则、过程和标准相一致
可靠性	在很长一段时间内，确保系统在需求正确的情况下完成预定的功能

SPASVO

续前表

风险因素	内容
易用性	要使用系统所需的学习、操作、准备输入数据、解释系统输出数据等过程的复杂难易程度
可维护性	在定位和修改运行中的系统错误时所需的努力，既包括系统缺陷，也有对用户需求的理解错误
可移植性	要把一个程序从一种硬件或软件环境转移到另一个环境中所需的努力程度。它包括数据转换、程序更改、系统操作和文档变化等
耦合	把一个系统中的组件与其他环境下的系统联系起来所需的努力
易操作性	把系统结合到操作环境中，然后再操作应用系统所需的努力程度

测试阶段

- 需求测试阶段
- 设计测试阶段
- 单元测试阶段
- 集成测试阶段
- 系统测试阶段
- 维护测试阶段

制定测试策略-六步法

- 第一步：选择和分级风险因素
- 第二步：确定系统过程阶段
- 第三步：确定风险内容
- 第四步：将风险置入矩阵
- 第五步：确定风险的测试策略
- 第六步：确定测试方法

三个基本问题

- 为什么担心?
- 谁关心?
- 测试多少?

好的测试策略

- 与具体产品有关
  - 不管通用的测试策略有多好，与当前具体产品和技术有关的测试策略会更好
- 关注风险
  - 显示测试过程中需要关注的重要问题
- 多样化
  - 多样化的测试策略优于单调的测试策略。即包含各种不同的测试手段和方法
  - 逃脱一种方法的缺陷还可能被另一种方法捕获
- 实用
  - 测试策略必须能被执行
  - 不要提出超过测试团队能力的测试策略

第一步：选择和分级风险因素

- 参与人员：
  - 系统消费者
  - 关键用户
  - 客户
  - 测试小组
- 多数情况下，只需要选择3~7个因素
- 将所有风险因素分级：高、中、低

第二步：确定系统过程阶段

- 项目开发小组确定系统开发的各阶段
- 测试小组将开发阶段对应到测试阶段（使用风险因素/测试阶段矩阵）

SPASVO

风险因素/测试阶段矩阵

测试阶段 / 风险因素	需求测试阶段	设计测试阶段	单元测试阶段	集成测试阶段	系统测试阶段	维护测试阶段

因素

风险项

第三步：确定风险内容

- 根据风险因素，分析每个风险因素在哪个测试阶段会存在什么风险项目
- 明确风险的内容
- 将风险内容分级，高、中、低

第四步：将风险项置入矩阵

- 将风险项填入测试因素/测试阶段矩阵

测试阶段 / 风险因素	需求测试阶段	设计测试阶段	单元测试阶段	集成测试阶段	系统测试阶段	维护测试阶段
因素1		风险项				
…						风险项
			风险项			
因素n					风险项	

SPASVO

第五步：确定风险的测试策略

- 针对每一个具体的风险内容定义相应的测试策略
- 测试策略包括
  - 单元测试
    - 白盒测试
    - 黑盒测试
  - 集成测试
    - 接口集成测试
    - 业务流程集成测试
    - 自动化回归测试
  - 系统测试
    - 需求确认测试
    - 易用性测试
    - 可靠性测试
    - 性能测试
    - 可维护性测试
    - 可移植性测试
    - 等
  - 维护测试

    自动化回归测试

第六步：确定测试方法

- 针对每一个测试策略，确定相应的测试方法
- 这个方法是大致的方法
  - 例如：
    - 使用某种工具进行白盒单元测试，包括覆盖率测试、内存泄漏测试等
    - 使用某种工具进行自动化回归功能测试，包括模块覆盖率、路径覆盖率等
    - 根据系统用户情形，编制系统需求确认测试用例，进行系统需求确认测试
    - 使用某种工具进行性能、可靠性测试
    - 根据行业界面开发标准，进行易用性测试

风险的细化

- 目的
  - 策略风险是一个软件系统会遇到的高层的业务风险
  - 战术风险则是低层的风险
  - 将策略风险分解成战术风险的目的是帮助创建解决这些风险的测试计划

战术风险的分类

- 1、结构化风险
  - 应用系统及创建应用系统的方法中所涉及的风险
- 2、技术风险
  - 创建和操作应用系统的技术所涉及的风险
- 3、规模风险
  - 软件各方面的规模所涉及的风险

SPASVO

风险的细化步骤

- 第一步
  - 准备风险细化工作表
- 第二步
  - 确定被测系统的风险级值
- 第三步
  - 计算和累计风险分数
- 第四步
  - 计算平均风险分数
- 第五步
  - 比较风险级值
- 第六步
  - 获取级别为高级的风险

SPASVO

工作表1-结构化风险评估

分级：L--低级　M--中级　M--高级　--没用到的风险		级别	级别x权值=分数
1	现存系统的变化频率 每年少于2次 每年2~10次 每年多于20次	NA=0 L=1 M=2 H=3	3
2	与该项目相关的业务领域的变化程度 很少的变化 重要的但是可管理的变化 系统功能和/或资源需求的主要变化	L=1 M=2 H=4	3
3	项目完成地点 公司设备 本地非公司设备 异地	L=1 M=2 H=5	2
4	关键的项目机构 内部 承包商，长期合作方 承包商，中标方	L=1 M=2 H=6	2

SPASVO

工作表2-技术风险评估

分级：L--低级　M--中级　M--高级　--没用到的风险	级别	级别x权值 =分数
在硬件或软件出错时完成任务的能力 ●在没有系统时也能完成 ●没有完整的操作系统时也能完成，但是有较少的能力要求 ●没有完整的操作系统时无法完成	L=1 M=2 H=6	2
系统的使用频率 ●定期使用（每周或更小的频率） ●每日使用（但不是每天24小时） ●持续使用（每天24小时）	L=1 M=2 H=5	2
依赖于和外部系统传送数据来完成功能的程度 ●功能独立：操作中不需要和其他系统传送数据 ●必须从另一系统接收或传送数据 ●必须从很多系统接收或传送数据	L=0 M=2 H=3	2

SPASVO

工作表3-规模风险评估

分级：L--低级　M--中级　M--高级　--没用到的风险	级别	级别x权值 =分数
项目完成时间 ●12个月或更少 ●13~24个月 ●多于24个月	L=1 M=2 H=3	3
项目进展与进度安排 ●提前于进度表 ●符合进度表 ●比进度表晚（3个月或更少） ●比进度表晚（多于3个月）	L=1 M=2 H=3 H=4	1
在系统内部相互联系的子系统 ●1-2个 ●3-5个 ●多于5个	L=1 M=2 H=3	3
●进行系统测试的项目资源的百分比 ●多于40% ●20%~40% ●少于20%	L=1 M=2 H=3	2

SPASVO

工作表4-风险分数分析

应用系统：						
风险领域	分数		等级			注释
	总分	平均分	高	中	低	
结构						
技术						
规模						
风险总分						
高级风险						
风险领域		风险			测试考虑	
签名：				日期：		

SPASVO

经验与教训

1-使用冒烟测试检验版本

- 冒烟测试
  - 健全性测试或者接受测试
  - 是一种测试包，其目标是检查版本的基本功能。如果该版本没有通过测试，则可宣布该版本不太稳定，布置的测试
- 方法
  - 一般情况下，当某个新版本提交测试时，要有一名测试员运行冒烟测试（可能是自动化测试、手工测试或自动/手动结合测试）。其他测试员要等到该版本通过冒烟测试后才投入测试。

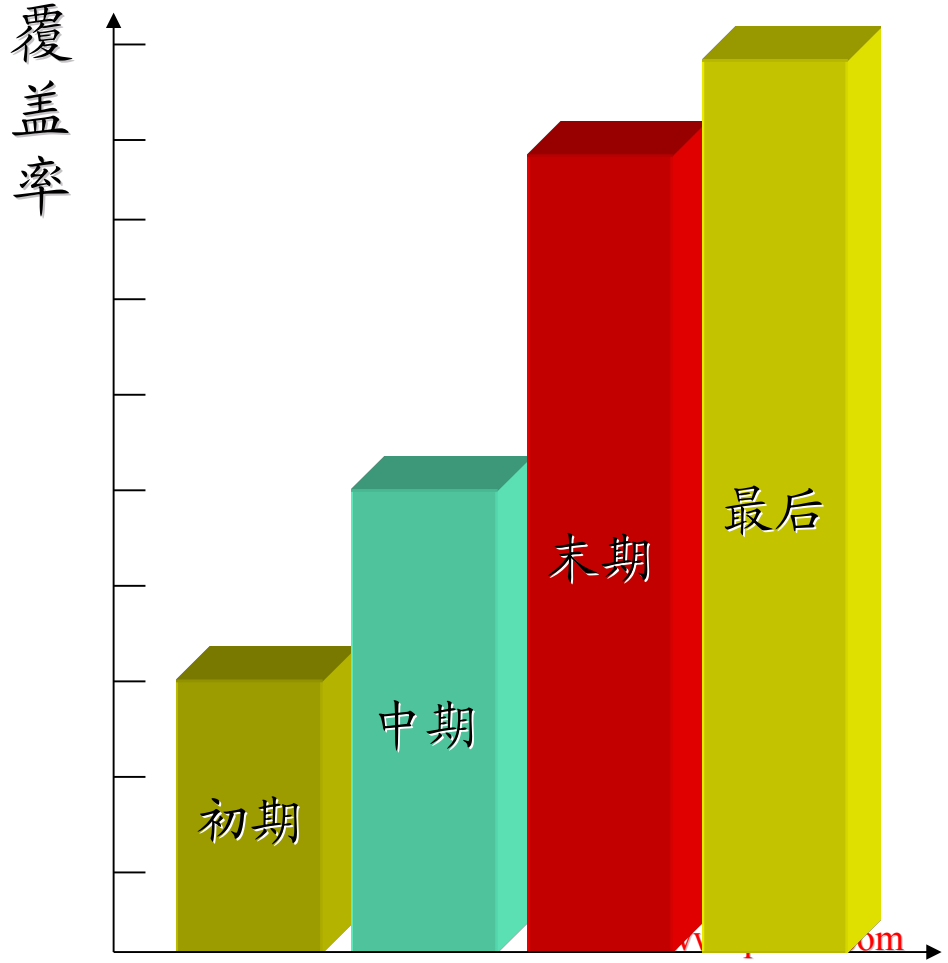
2-项目文档

- 80%的代码用于实现错误处理
- 20%的代码用于主要控制流
- 但是，在即使是完整的规格说明书中，也只会用20%的篇幅描述错误处理
- 意味着，80%的代码是程序员边编码边设计的
- 所以，不要一测试就找项目经理要开发文档等东西。你需要充分利用其他信息源
  - 用户手册；产品市场开发文献；软件变更纪录；内部备忘录

SPASVO

3-根据产品的成熟度确定测试策略

- 项目初期
  - 同情的测试
- 项目中期
  - 积极的测试
- 项目末期
  - 多样的测试
- 项目最后
  - 谨慎的测试



SPASVO

4-测试分级->简化测试策略复杂性

- 0级-冒烟测试
  - 简单测试，若测试失败，打回程序员
- 1级-能力测试
  - 保证每个函数都能执行任务。避免曲折的场景、富有挑战性的数据和功能交互
- 2级-函数测试
  - 数据覆盖、测试结果评估
  - 边界、压力和错误处理机制的测试
  - 仍避免曲折的场景和功能交互
- 3级-复合测试
  - 多组函数之间的交互和控制流
  - 复杂场景的测试
  - 性能评估、兼容性、资源紧张程度、内存泄漏、长时间可靠性

- 分级应用
  - 对早期版本首先执行0级、1级测试
  - 对中期版本执行1级、2级测试
  - 对后期版本执行3级测试

目录

- 测试风险与测试策略
- 常见要素与策略

常见要素

要素	策略	备注
测试所处于的阶段	覆盖率预期	
开发模型	根据模型、项目时间决定策略	
检查点分布	决定主要的测试项	
多系统/单系统	是否测试接口	
OLTP	联机以交易测试、流程测试为重点;	
OLAP	olap以数据测试为重点	
时间对功能的影响	营业日期、特殊营业日、累积日期问题	
GUI/数据处理	界面控制逻辑测试	
workflow	以工作流和节点为测试重点	
多运行版本环境	多系统回归、自动回归	