LOGO

测试方法介绍

目录

测试过程

测试计划	测试用例	Bug	测试报告	外网用户反馈
	单元测试	功能测试 性能测试 安全测试 兼容性测试 数据上报测试 系统测试	冒烟测试	线上测试

什么是Bug？

- **不符合产品说明书**
  - 实现了说明书未提及的功能---多做
  - 未实现说明书要求的功能—少做
  - 实现了说明书指明不该有的错误---错做
  - 未实现说明书虽未明确提及但应该实现的目标---可用性
  - 软件难以理解、不易使用、运行速度慢---易用性

Bug的分析

- **Bug的分析，尤其是外网bug分析十分重要**
  - 需求缺陷？
  - 需求变更？
  - 架构设计缺陷？
  - Bug修改引发？
  - 测试设计不全？
  - 历史bug？
  - 时间不够？

测试工作的目标

- **目标**

  尽可能早而全面的找出Bug，并确保其得以正确的处理。

- **用途**

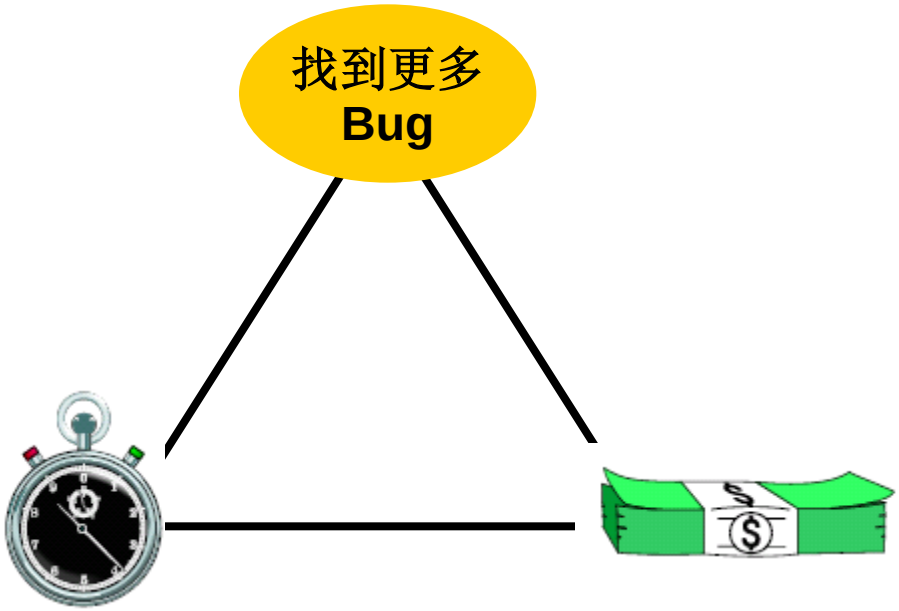
  - 找出Bug
  - **避免Bug**
  - 评估质量
  - 提高研发效率
  - ……

测试工作的原则

- 原则1 - 全程测试原则(测试活动应该贯穿整个软件生命周期)
- 原则2 - 不完全原则(穷尽测试是不可能的)
- 原则3 - 缺陷聚集原则(80-20理论)
- 原则4 - 免疫性原则 (杀虫剂悖论，测试用例需要经常的评审和修改)
- 原则5 - 不存在缺陷的谬论(测试的目的是为了证明软件有错)

测试需要明智的选择

- 测试的价值---bug 多？



问题1：如何让质量和进度达到双赢？

1）增加测试资源

2）提高质量活动的技术水平

如何提高质量活动的技术水平？

1）提高需求质量（需求评审），提高开发质量（编码规范、提测质量）

2）优化测试方法---又快又好

3）抓住重点测试---有损服务

4）自动化测试，提高效率

问题2：时间紧迫，要写测试文档么？

1. 测试用例必不可少，测试用例是保证产品质量的利器, 没有武器如何战斗？

- 测试用例的粒度和方式是可以变通的！
- 编写用例的过程是逐步完善的。

2. 测试文档的作用：

- 测试文档用来组织测试执行
- 知识、经验的积累沉淀
- 是测试人员与产品、开发交流的重要方式
- 是质量分析的工具

问题3：自动化测试 or 手工测试？

- 不要为了自动化而自动化！
- 自动化决定因素：
  - 项目的周期---长、短
  - 需求变更的频率---变化快、功能稳定
  - 脚本维护成本---高、低
  - 投入产出率

问题4：漏测只是TE的原因么？

- 漏测要具体分析，以下列举常见的top问题：
  - 产品设计的缺陷---需求评审、需求质量改进
  - 开发质量不高，bug经常反弹---编码规范、提测规范、千行代码缺陷率
  - 测试设计缺陷---测试用例的深度、广度不够、同行评审、用例缺陷率
  - 测试覆盖面小---不同测试方法的全面覆盖
  - 测试效率低下---自动化测试、测试工具、接口测试等
  - 发布策略不佳---灰度发布、外网监控、运维团队
  - 版本节奏、项目进度不合理---迭代规划、人力配比、项目管理、团队建设
  - 流程拖沓、团队配合度低---QA流程、规范、模版、报告的改进

PS：质量不是靠测试单方面保证，而是需要研发流程中各个角色一起承担，测试人员也不能仅仅关注测试质量，还要去思考整个研发过程中的问题，推动各个角色去改进。

测试用例设计方法

- 边界值分析法
- 等价类划分法
- 判定表方法
- 逻辑图方法
- 错误推测法
- 探索性测试
- 其他方法

测试用例设计原则

- 测试用例的可读性

mouseover一个印象词	1)Tips显示改为：“**、**、***这样描述我” 2)Tips给出该词的好友昵称， 3)Tips中的全部字数少于150个时，不显示省略号

打开的编辑页面筛选条件	好友状态为“在线”
	好友状态为‘Q我吧’
	好友状态为‘忙碌’
	好友状态为‘离开’
	好友状态为‘静音’
	好友状态为‘隐身’
	好友状态为‘离线’

测试用例设计原则

- 测试用例的重用性

Dialog/Button/Static/Radiobutton/Checkbox/Edit/Scrollbar/Tab/listctrl/Menu/多控件测试/UI测试/

测试项目	检查项	实际输出	备注
checkbox	勾选后的图标是否显示正确?		
	勾选后的功能是否正常?		
	取消勾选后的图标是否显示正确?		
	取消勾选后的功能是否正常?		
	勾选的设置项是否支持漫游?		

测试项目	测试步骤			预期输出	实际输出
登录页面	使用控件	button	用例参考《专项测试用例》	正常	
		edit		正常	
		checkbox		正常	

测试用例设计原则

- 测试用例的维护性

  -粒度

  -分层

  -流程图

- 设计原则：多，快，好，省

问题：测试用例的粒度如何把握？

- 需要考虑以下因素：
  - 产品的稳定性（新产品？需求变更的频率？）
  - 需求文档的精确程度和详细程度（没有需求文档如何测试？）
  - 产品形态的不同（CS\BS产品的区别）
  - 团队成员的成熟度（产品素养、开发素养、测试素养）
  - 团队的稳定性（人员流动）
  - 自动化的需要（测试效率提升）

结论：粗、细没有定律，关键是选择适合自己团队的。

边界值分析法

- 正常值：
- 边界值：At, below, above
- 次边界：特殊字符、日期的特殊值（闰年）
- 默认值、空白、空值、零和Null
- 非法数据

练习-边界值

输入设备的边界

鼠标：时间间隔，连续点击次数

键盘：时间间隔，连续点击次数

屏幕：有效区域的边界

界面元素的边界

- 帐号输入框
- 密码输入框
- 验证码输入框
- 验证码显示框
- 换一张按钮
- 用户服务协议
- 注册并同意协议按钮
- 关闭窗口按钮

逻辑边界值

- 帐号输入框: 0,3,4,5,15,16,17, ∞
- 密码输入框: 0,5,6,7,19,20,21 , ∞

软件功能性边界：

- 注册错误次数
- 验证码刷新频率

组合边界：

- 帐号、密码、验证码的组合

等价类划分

- 把软件具有相似输入、相似输出、相似操作的分在一个组

- 最常用的划分方法：

  -有效(valid) 等价类

  -无效(invalid)等价类

- 划分等价类的标准：

  -完备性，划分为互不相交的一组子集，而子集的并是整个集合

  -互斥性，子集互不相交，保证一种形式的无冗余性

练习-等价类划分

输入等价类	有效等价类	无效等价类
帐号的类型	1）小写英文 2）数字 3）英文+数字的合法组合	1）大写英文 2）非英文 3）非数字 4）合法+非合法组合 5）非合法+非合法组合
帐号的长度	4<=长度<=16	1）长度小于4； 2）长度大于16；
密码的类型	1）大写字母 2）小写字母 3）数字 4）符号 5）合法组合	1）非字母 2）非数字 3）非符号 4）合法+非合法组合 5）非合法+非合法组合
密码的长度	6<=长度<=20	1）长度小于6； 2）长度大于20；

判定表方法

- 判定表是分析和表达多逻辑条件下执行不同操作的情况的工具。
- 最常用的使用方法：

  1) 确定规则的个数

  2) 列出所有的条件桩和动作桩

  3) 填入条件项和动作项

  4) 简化.合并相似规则（相同动作）
- 判定表的优、缺点：

  -优点：它能把复杂的问题按各种可能的情况一一简明易于理解，也可避免遗漏。

  -缺点：不能表达重复执行的动作，例如循环结构。

练习-判定表

1)确定规则数

这里有2个条件，每个条件2个取值，规则数=2*2=4

2)列出所有的条件桩和动作桩

条件	帐号是否为空？
	密码是否为空
动作	报错
	不报错

3) 填入条件项和动作项

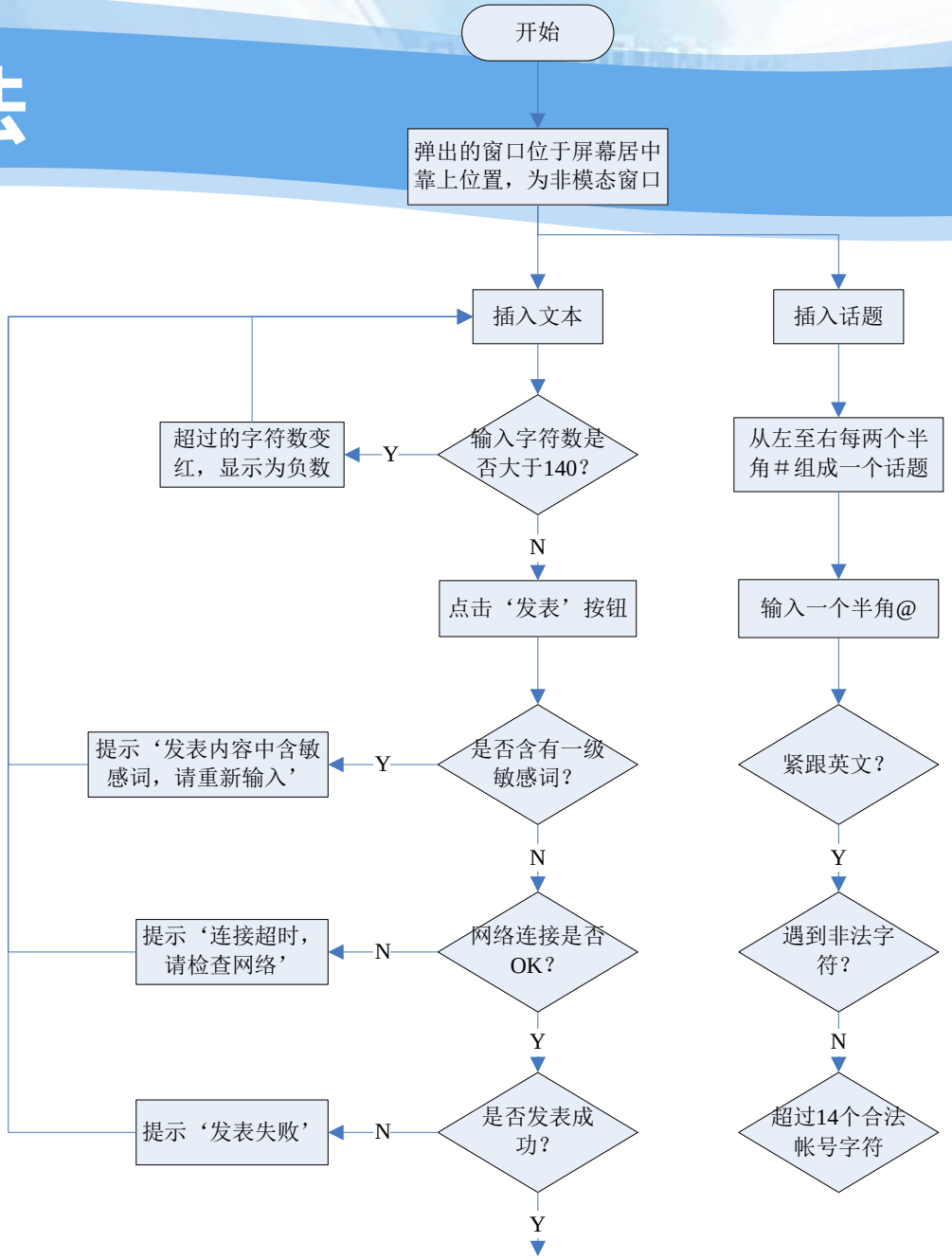
编号		1	2	3	4
条件	帐号是否为空？	Y	Y	N	N
	密码是否为空	Y	N	Y	N
动作	报错	√	√	√	
	不报错				√

4) 简化.合并相似规则

编号		1	2	3
条件	帐号是否为空？	Y	–	N
	密码是否为空	–	Y	N
动作	报错	√	√	
	不报错			√

逻辑图方法

- 作用：帮助我们理解待测系统的功能逻辑，提高用例覆盖的深度、广度，以便于设计正确的测试用例。



错误推测法

- 定义：就是利用经验和直觉去找Bug。
- 特点：列举程序所有可能的错误和容易发生错误的特殊情况，是测试的必要补充。
- 建议：
  - 像最小白的用户一样做
  - 像黑客一样思考，破坏性测试
  - 凭经验、直觉和预感，测试的第六感
  - 到已经找到缺陷的地方再找找

探索性测试

- 定义：就是利用经验和直觉去找Bug。
- 特点：

  1）同时性：同时设计测试和执行测试，碰到问题时及时改变测试策略

  2）创造性：不断学习被测系统，设计出新的或更好的测试点。

- 建议：
  - 向开发了解系统的功能---协议、server、底层、应用层
  - 历史测试经验的积累
  - 不走寻常路

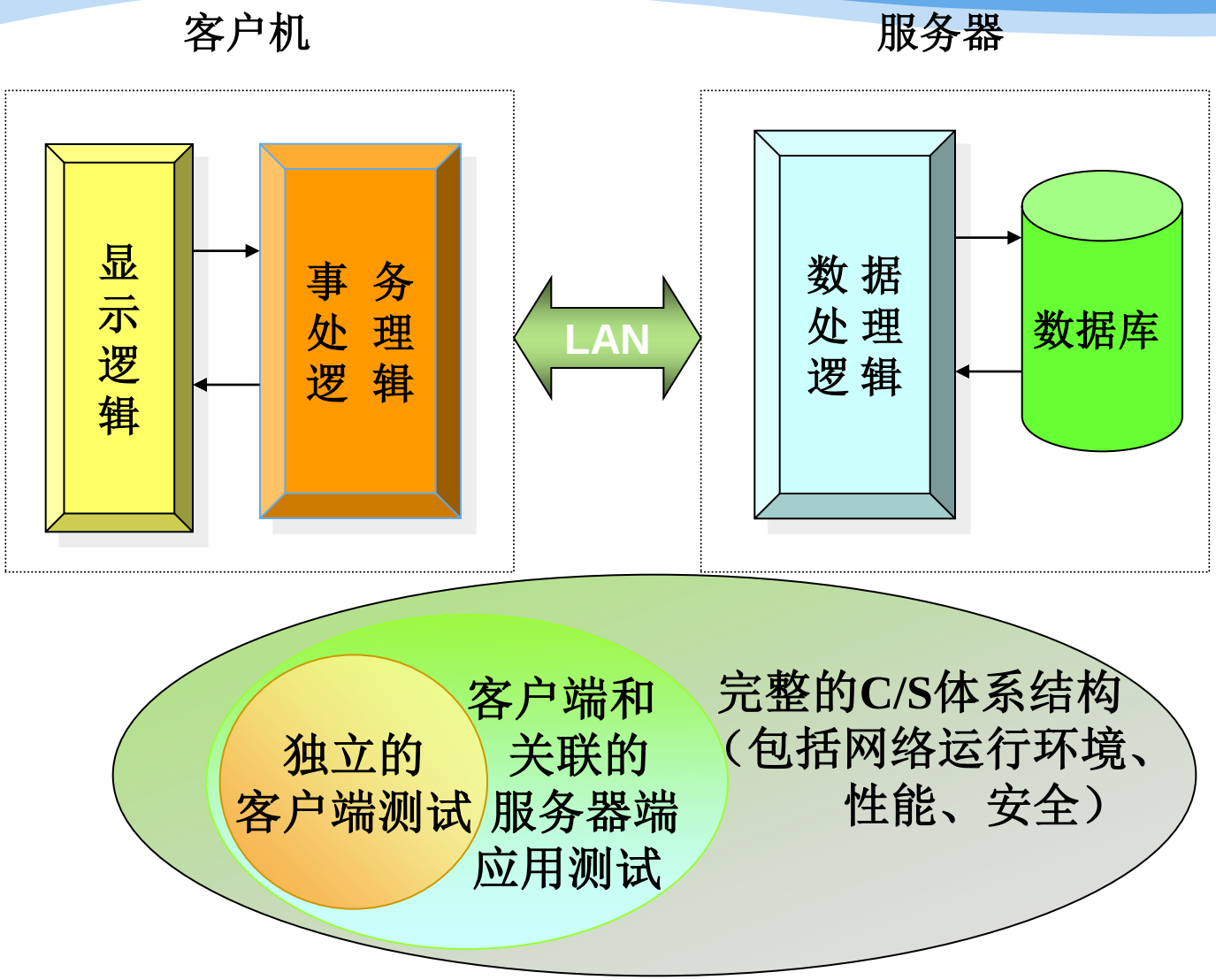
测试用例设计准则

- 在任何情况下都必须使用边界值分析方法，经验表明用这种方法设计出测试用例发现程序错误的能力最强。
- 必要时用等价类划分方法补充一些测试用例。
- 如果程序的功能说明中含有输入条件的组合情况，则一开始就可选用判定表 。
- 对照程序逻辑图，检查目前测试用例的逻辑覆盖度，再适当补充足够的测试用例。
- 用错误推测法再追加一些测试用例。
- 利用探索性测试方法，在测试时发散思维构造测试用例。

什么是C/S模式

- 客户端Client/服务器Server结构
- 一般使用大型数据库系统
- 表示层或者用户界面使用的, 放在client端
- 业务逻辑一般分布在Server端和Client端
- Client端和server端使用Lan或者Internet连接
- 多个client,一个或多个server

C/S系统的测试策略


客户机
服务器
显示逻辑
事务处理逻辑
LAN
数据处理逻辑
数据库
独立的客户端测试
客户端和关联的服务器端应用测试
完整的C/S体系结构（包括网络运行环境、性能、安全）

例子

- YY客户端登录模块要考虑到哪些测试类型?

例子

客户端测试

功能测试　界面测试　性能测试　安全性测试

配置测试　兼容性测试　安装测试　易用性测试

客户端和服务器的关联测试

接口测试　数据测试　负载测试

完整的C/S体系

网络通信测试

客户端测试

- 功能测试：验证测试对象的功能是否满足产品说明书
- 界面测试：外观、尺寸、焦点、分辨率、特效
- 性能测试：速度（响应时间）、资源（内存泄漏？GDI泄漏？句柄泄漏？）
- 安全性测试：病毒和木马，用户信息安全（帐号、密码），网页脚本安全漏洞
- 配置测试：程序所需配置的各种情况的可达到性, 笔记本用户、连接走中转等
- 兼容性测试：版本兼容性、软件兼容性、硬件兼容性
- 安装测试：全新安装、覆盖安装、在线安装
- 易用性测试：快捷键、支持滚轮、控件分层、tooltips提示

客户端和服务器的关联测试

- 接口测试：接口稳定性、容错、容灾处理机制
- 数据测试：数据正确性、数据安全性
- 负载测试：大数据量、大访问量时的系统使用情况

网络通信测试

- 电信
- 联通
- 教育网
- 移动网络
- 网络代理
- 断网重连
- 限速丢包环境

一些容易遗漏的测试点

- 预埋代码
- 安装包目录
- 多余Log
- 用户权限
- 随机bug
- 版本信息检查
- 下载测试

谢谢