1. **软件测试方式方法**

(1) 边界测试，测试用户输入框中的数值的最大数和最小数，以及为空时的情况。

(2) 非法测试，例如在输入数字的地方输入字母。

(3) 跟踪测试，跟踪一条数据的流程,保证数据的正确性。

(4) 在开始测试时应保证数据的正确性，然后在从系统中找出各种BUG。

(5) 接口测试，程序往往在接口的地方很容易发生错误，要在此模块测试勿掉以轻心。

(6) 代码重用测试，在开发过程中有些模块功能几乎相同，程序员在重用代码时可能忘记在原有代码上修改或修改不全面，而造成的错误。

(7) 突发事件测试，[服务器](http://www.ltesting.net/html/78/category-catid-378.html)上可能发生意外情况的测试。

(8) 外界环境测试，有些系统在开发时依赖于另外一个系统,当另外一个系统发生错误时, 这个系统所受到的影响的情况。

(9) 在程序员刚修复[Bug](http://www.ltesting.net/html/98/category-catid-98.html)之后的地方,再找一找，往往程序员只修复报告出来的[缺陷](http://www.ltesting.net/ceshi/ceshijishu/qxgl/)而不去考虑别的功能在修改时可能会重新造成错误。

(10) 认真做好测试记录在做完一天的测试记录之后,第二天再根据第一天的测试记录重复测试你会发现有未修正的错误。

(11) 文字测试，如果在系统中有用词不当的地方，我想这是不应该的。

(12) 系统兼容测试，例如有些程序在IE8以上能运行正常，到IE7以下就不能运行。有些程序在WIN10下能运行，而到WINXP却不能运行。像一些很特别的用户去使用系统，你很有可能发现BUG。

(13) 用户的易用性测试，往往用户的[需求](http://www.ltesting.net/ceshi/ruanjianzhiliangbaozheng/xqgl/)是不断的变化的，而其中的一部份变化的原因，是有用户操作上不方便引起的。

软件测试是软件开发中的重中之重，没有一点可以马虎的，在[项目管理](http://www.ltesting.net/html/66/category-catid-166.html)过程，我强调的是每个过程的每一个环节都要进行测试，保证系统在每个阶段可以控制。因为软件测试中考虑的问题基本上是[项目管理](http://www.ltesting.net/ceshi/ruanjianzhiliangbaozheng/xmgl/)中考虑的问题。

我认为在项目管理中考虑的一些问题应该是在软件测试时有些体现，体现的内容是软件测试的一些侧重点，具体说，软件测试是事务性的，而项目管理是策略性，一些策略性的东西必须在一些事务性的事务上来实现。

1. 测试用例

编写测试用例的最终目标是：一个对于产品毫无所知的人员，也能够快速的熟悉用例并执行用例。

1)具有清晰名称、前提条件、操作步骤、期望结果的；

2)可被他人理解的；

3)可被他人执行的；

**1.用例名称**

1）名称要与系统节点尽量统一

2）不要包括具体操作步骤；

**2.前置条件**

1）执行用例测试步骤前需要做的所有必备条件，原则上所有用例都有前置条件；

2）不可将其他用例作为前置条件，前置条件需要语言描述；

3）完整清楚，包括入口、帐号类型、账号权限、数据准备等，具体要求如下：

3.1）入口：覆盖所有功能入口，包含URL直接访问；

3.2）账号类型和权限：覆盖全部会员类型，注意业务权限控制，比如子账号权限，disable会员权限；

3.3）数据准备：数据准备完整正确，覆盖到线上环境的所有情况；标识出业务流程处于的条件，写明数据库表字段值，如OFFER.status=TBD；对于复杂的数据准备，写清具体SQL

**3.操作步骤**

1）操作步骤描述清晰。如：在什么页面，点击什么链接或按钮；页面入口、链接、按钮名称都要写清楚；

2）操作和结果是一一对应的，但操作中不要包含结果的检查；

3）用例描述中不允许存在连词、介词，比如：而且，和，还（这种情况可以拆分为多个点）；

4）用例描述中不允许出现假设性词汇，比如：假如，或许，可能，…的时候等；

5）用例描述中不允许出现二义性语句；

**4.预期结果**

　　1）原则上每个用例必需要有预期结果，结果不能为空；

　　2）结果中只能包含结果，不能有步骤；

　　3）一个结果有多个检查点时，确保检查点完整；

　　3.1）结果含需要验证的所有结果输出，如页面检查、存储检查、消息检查等；

　　3.2）结果涉及页面，需明确页面提示结果、数据变化；

　　3.3）结果涉及存储：需明确关键值变化、数据库具体的表和关键字字段值变化；

　　3.4）结果涉及消息：需明确关键查看内容；

3.5）结果对应不同输入数据有差别时需分别对应描述清晰；

1. 面向Web应用系统的测试与传统的软件测试不同，不仅需要检查和验证是否按照需求规格说明书的要求运行，而且还要测试Web应用系统在不同浏览器上显示是否符合要求，与不同的数据库连接是否有效、更重要的是在性能、安全性、可用性等方面

功能测试、性能测试、安全性测试、配置和兼容性测试、可用性测试

**链接测试**  
 链接是Web应用系统用户界面的主要特征，它指引着Web用户在页面之间切换，以完成Web应用系统的功能。  
 测试重点：链接是否正确、链接页面是否存在、是否有孤立的页面（没有链接指向的页面）

**表单测试**  
 表单(Form)是指网页上用于输入和选择信息的文本框、列表框和其他域，实现用户和Web应用系统的交互，当用户给Web应用系统管理员提交信息时，需要使用表单操作，如用户注册、登录、信息提交、查询等  
 测试重点：表单控件的正确性、提交信息的完整性、正确性、是否有错误处理

**Cookie测试**  
Cookie通常标识用户信息，记录用户状态 。  
 使用Cookie技术，当用户使用Web应用系统时，能够在访问者的机器上创立一个叫做Cookie的文件，把部分信息（访问过的页面、登录用户名、密码等）写进去，来标识用户状态。如果该用户下次再访问这个Web应用系统，就能够读出这个文件里面的内容，正确标识用户信息  
 如果Web应用系统使用了Cookie，必须检查Cookie是否能正常工作，是否按预定的时间进行保存内容

**设计语言测试**  
 在Web应用系统开发初始，根据软件工程的要求用文档的形式确定Web应用系统使用哪个版本的HTML标准，允许使用何种脚本语言及版本，允许使用何种控件，这样可以有效的避免Web应用系统开发过程中出现设计语言问题。  
 其他测试：数据库测试、面向任务、业务逻辑的测试、探查性测试、回归测试

**速度测试**  
 对于最终的Web应用系统用户而言，最关心的性能问题是访问Web应用系统页面时，多长时间才能显示出来所需要的页面。通常情况下，响应时间不超过5秒。有些Web应用系统有超时限制，如果响应时间太慢，用户可能还没来得及浏览内容，就需要重新登录了。  
影响响应时间的原因有很多：应用程序服务器需要从数据库的大量数据中检索信息；服务器硬件影响（CPU、内存）；所访问页面文件大小；网络连接带宽

**负载测试**  
 负载测试是为了测量Web应用系统在一定负载情况下的系统性能，通常得出的结论是Web应用系统在一定的硬件条件下可以支持的并发用户数目或者单位时间数据（或事件）的吞吐量。在进行负载测试前，需要定义标准用户（活动用户）的概念，定义执行典型的系统流程，定义负载测试执行总时间，定义抓取哪些事务的平均响应时间，定义用户可以接受的平均响应时间（通常为5秒）。测试时，增加用户数量，平均响应时间就会增加，当达到用户可以接受的平均响应时间这个临界点，即是此系统可以支持的并发用户数

**压力测试**  
 对Web系统进行压力测试，类似于普通机械、电子产品进行的破坏性试验。方法是实际破坏Web应用系统，测试系统的反应。  
 压力测试是测试系统的限制和故障恢复能力，也就是测试Web应用系统会不会崩溃，在什么情况下会崩溃，崩溃以后会怎么样。在Web应用系统性能测试过程中，常常将压力测试和负载测试结合起来。在负载测试的基础上，增大负载量，直到系统崩溃。  
 实施性能测试需要注意：测试工具灵活使用、性能测试计划的制定。

**安全测试**

由于数据库安全性导致的Web应用系统安全性问题、数据库文件被下载、用户重要信息没有经过加密而存于数据库中。确认操作系统安全性 ，避免因操作系统漏洞导致Web应用程序的安全性问题。

Web应用系统多采用登录的方式，产品发布时提供默认的管理员用户名和密码。确保应用系统实际应用中可修改默认管理员帐号和密码  
1.用户名和密码设置要求（长度、大小写敏感、复杂度）  
2.允许错误登录的次数  
3.是否可以不登录而直接浏览某个页面

4.保证日志文件记录了Web应用系统的主要操作过程，并可根据日志文件追查到系统使用情况；同时还需要保证日志文件本身的安全性、完整性，防止被入侵者删除、获得  
5.当Web应用系统采用了SSL等加密技术之后，需要确认加密、解密后信息传递的正确性和完整性。

6.需要确认Web应用系统是否有超时设置，如有，则保证在超时设置时间内，如果未操作Web应用系统，当再次访问系统，需要重新登录  
7.了解安全漏洞信息，避免Web应用系统中出现的漏洞被入侵者利用；及时升级补丁程序，提高系统安全性

8.Web应用系统多采用分布式体系结构，服务器端通常包括Web服务器组件、数据库服务器组件等。服务器还可能运行在不同的操作系统上，并且这些组件、操作系统等还可以有不同的配置方法，所以针对服务器的兼容性测试往往工作量较大  
9.针对客户端浏览器的配置和兼容性测试是必不可少的，并且占据了Web应用系统客户端配置和兼容性测试的大部分时间

典型的应用服务器：  
Web服务器：  
 通过MS IIS、BEA Weblogic、IBM Websphere、Tomcat、Sun J2EE Application、Apache等中间件、插件，提供Internet/Intranet Web服务，实现与众多客户之间的数据交换和共享  
数据库服务器：  
 主要提供数据库查询、处理的平台，通过Oracle、SQLServer、Informix、DB2、Sybase、MySQL等中大型的数据库管理系统来构建  
实时通信服务器：  
 提供数据实时通信、消息传递等服务，如MSN、Yahoo message和ICQ等专用服务器

**服务器端配置和兼容性测试内容**

Web服务器

数据库服务器

防火墙

操作系统

硬件兼容性

1. 缺陷等级（等级名称与定义）

**严重缺陷（应用系统崩溃或系统资源使用严重不足）**

1. 系统停机（含软件、硬件）或非法退出，且无法通过重启恢复
2. 系统死循环
3. 数据库发生死锁或程序原因导致数据库断连
4. 系统关键性能不达标
5. 数据通讯错误或接口不通
6. 错误操作导致程序中断

**较严重缺陷 （系统因软件严重缺陷导致下列问题）**

1. 重要交易无法正常使用、功能不符合用户需求
2. 重要计算错误
3. 业务流程错误或不完整
4. 使用某交易导致业务数据紊乱或丢失
5. 业务数据保存不完整或无法保存到数据库
6. 周边接口出现故障（需考虑接口时效/数量等综合情况）
7. 服务程序频繁需要重启（每天2次或以上）
8. 批处理报错中断导致业务无法正常开展
9. 前端未合理控制并发或连续点击动作，导致后台服务无法及时响应
10. 在产品声明支持的不同平台下，出现部分重要交易无法使用或错误

**一般性缺陷（系统因软件一般缺陷导致下列问题）**

1. 部分交易使用存在问题，不影响业务继续开展，但造成使用障碍
2. 初始化未满足客户要求或初始化错误
3. 功能点能实现，但结果错误
4. 数据长度不一致
5. 无数据有效性检查或检查不合理
6. 数据来源不正确
7. 显示/打印的内容或格式错误
8. 删除操作不给提示
9. 个别交易系统反应时间超出正常合理时间范围
10. 日志记录信息不正确或应记录而未记录
11. 在产品声明支持的不同平台下，出现部分一般交易无法使用或错误。

**较小缺陷 （系统因软件操作不便方面缺陷）**

1. 系统某些查询、打印等实时性要求不高的辅助功能无法正常使用
2. 界面错误
3. 菜单布局错误或不合理
4. 焦点控制不合理或不全面
5. 光标,滚动条定位错误
6. 辅助说明描述不准确或不清楚
7. 提示窗口描述不准确或不清楚
8. 日志信息不够完整或不清晰，影响问题诊断或分析的

**其他缺陷 （系统辅助功能缺陷）**

1. 缺少产品使用、帮助文档、系统安装或配置方面需要信息
2. 联机帮助、脱机手册与实际系统不匹配
3. 系统版本说明不正确
4. 长时间操作未给用户进度提示
5. 提示说明未采用行业规范语言
6. 显示格式不规范
7. 界面不整齐
8. 软件界面、菜单位置、工具条位置、相应提示不美观，但不影响使用

**建议、优化类（由问题提出人对测试对象的改进意见）**

**若按照禅道的BUG分类定义四五六可以都归为4级BUG**