面向考试的测试复习。。。。。。

**啊，为啥要考测试呢？知识分子何苦难为知识分子？**

**啊，为啥要用英文PPT呢？中国人何苦难为中国人。**

只整理了理论的东西。覆盖和测试路径覆盖那块，同志们还是好好看看相关PPT啥的吧，俺英语实在不成，就不折腾了。。这东西，还是有些技术性的，还是要自己学的会。。。。

专项测试部分，是比较大段的理论文字，大家看看了解下，自行删减吧，考试的时候有的扯就好了。恩恩。

考试相关部分十分感谢 **文烁** 总结整理。。

恩，有点多，看着晕的话，可以打开Word，最上面，**视图选项卡下，中间的文档结构图**的选项。

# 测试概论

## 软件失效的机理是什么？什么是错误、缺陷、故障和失效？

软件失效机理可以描述为：软件错误→软件缺陷→软件故障→软件失效

* 软件错误（**software error**）：是指在软件生存期内的不希望或不可接受的人为错误，其结果是导致软件缺陷的产生。
* 软件缺陷（**software defect**）：是存在于软件（文档、数据、程序）之中的那些不希望或不可接受的偏差，其结果是软件运行于某一特定条件时导致出现软件故障，此时称软件缺陷被激活。当软件意指程序时，软件缺陷（**defect**）与软件**/**程序污点（**bug**）同义。软件缺陷是存在于软件内部的、静态的一种形式。
* 软件故障**（software fault）**：是指软件运行过程中出现的一种不希望或不可接受的内部状态。出现软件故障时若无适当措施（容错）加以及时处理便产生软件失效，软件故障是一种动态行为
* 软件失效（**software failure**）：是指软件运行时产生的一种不希望或不可接受的外部行为结果。

## GB/T12504-90中软件质量的定义是什么？

* 软件产品中能满足给定需求的各种特性的总和
* 这些特性称为质量特性，它包括功能度、可靠性、易使用性、时间经济性、资源经济性、可维护性和可移植性等

## IEEE1983中软件测试的定义是什么？目的是。

定义：

使用人工或自动手段来运行或测定某个系统的过程；其目的在于检验它是否满足规定的需求或是弄清预期结果与实际结果之间的差别

目的 ：

有不同的观点，一种观点认为软件测试应该把软件验证纳入工作范围内，另一种观点认为发现bug是软件测试的终极目标。验证扩展是指广义的测试，找到bug是狭义的测试。但是验证和确认是必不可少的，所以大多数认为是广义的软件测试，但是随着软件规模的扩大，验证活动也变得困难，比如可靠性测试就不好验证；为了排除心理障碍，也可以以找到bug作为终极目标，实用性较强

## 软件测试与软件质量保证之间的关系是什么？

* Program testing can be used to show the presence of bugs, but never their absence。
* 软件质量不是在软件产品中被测试出来的，而是在软件开发和生产过程中形成的。
* 软件测试不是万能的，它不能取代其它软件质量保证手段。

## . Validation & Verification（确认和验证）

 确认：是在开发过程之中或结束时评价系统或组件，以确定其是否满足所描述需求的过程。

 验证：是评价系统或组件，以确定给定价段的产品是够满足该阶段开始时确定的条件的过程。

## 软件测试由哪几个阶段组成？

* 单元测试是对软件设计的最小单位：模块的测试，目的是检验每个软件单元能否正确地实现其功能满足其性能和接口要求。
* 单元测试验证程序与详细设计说明的一致性。
* 集成测试将经过单元测试的模块逐步进行组装和测试，组装测试亦称集成测试。
* 组装测试验证程序和概要设计说明的一致性。
* 系统测试指软件与该软件所属的系统对接并测试其接口的过程，目的是在真实工作环境下检验软件是否能与系统正确连接，并满足软件研制的系统目标。
* 验收测试是检验所开发的软件是否按软件需求规格说明中确定的软件功能、性能、约束及限制等技术要求进行工作。

|  |  |
| --- | --- |
| Software Requirement | Define test objectives (criteria)  Project test plan |
| System Design | Design system tests  Design acceptance tests  Design usability test, if appropriate |
| InterMediate Design | Specify system tests  Integration and unit test plans  Acquire test support tools  Determine class integration order |
| Detailed Design | Create tests or test specifications |
| Implementation | Create tests  Run tests when units are ready |
| Integration | Run integration tests |
| System deployment | Apply system test  Apply acceptance tests  Apply usability tests |
| Operation and maintenance | Capture user problems  Perform regression testing |

**步骤：**

1. 设计测试输入；
2. 产生测试用例；
3. 运行测试用例脚本；
4. 分析测试结果；
5. 报告测试结果；

## 软件测试主要包括哪些测试技术？

### 静态测试技术

定义：不执行程序代码而寻找程序代码中可能存在的缺陷或评估程序代码的过程

1. 静态测试只要求提供产品的源代码，不要求提供二进制代码或可执行程序。

2. 静态测试不在计算机上执行程序，而是由人阅读代码，以确定：

a) 代码是否能够满足功能需求；

b) 代码是否与项目生存周期初期开发的设计一致；

c) 是否遗漏功能代码；

d) 代码是否恰当地处理错误；

* 主要由人工进行。
  + 代码审查。
  + 代码走查。
  + 桌面检查。
* 主要由软件工具自动进行的静态分析。

### 动态测试技术

定义：通过在抽样测试数据上运行程序来检验程序的动态行为和运行结果以发现缺陷

### 黑盒测试（功能测试、基于规约的测试）

* + 测试人员无须了解程序的内部结构，直接根据程序输入和输出之间的关系或程序的需求规范确定测试数据，推断测试结果的正确性
  + 边界值分析
  + 等价类划分
  + 基于决策表的测试
  + 因果图
  + 正交实验设计
  + 状态测试

### 白盒测试（结构测试、基于程序的测试）

* + 测试人员根据程序的内部结构特性和与程序路径相关的数据特性设计测试数据。
  + 控制流测试
    - 语句覆盖
    - 分支覆盖
    - 条件覆盖
    - 判定-条件覆盖
    - 路径覆盖

## 良好的软件测试充分性准则应具有那些基本性质？

* 空集不充分性：空测试数据集对任何软件都是不充分的
* 有限性：对任何软件都存在有限的充分测试数据集合
* 单调性：如果一个软件在一个测试数据集上的测试是充分的，那么再增加一些测试数据也应该是充分的
* 非复合性：即使一个测试数据集对于软件所有的成分是充分的，也不能保证它对于整个软件是充分的
* 非分解性：即使一个测试数据集对于整个软件是充分的，也不能保证它对于软件所有的成分是充分的
* 非外延性：即使两个程序语义上是等价的，也不能保证对其中一个程序充分的测试数据集对另一个程序也是充分的
* 一般多重修改性：即使两个程序语法结构相同，也不能保证对其中一个程序充分的测试数据集对另一个程序也是充分的
* 复杂性：软件越复杂，需要的测试数据就越多
* 回报递减律：测试得越多，进一步测试所得到的充分性增长就越少

## 测试覆盖准则具有什么作用？

### 测试覆盖准则与测试充分性

* 通常用测试覆盖准则度量测试充分性
* 针对程序内部结构的测试覆盖准则，主要包括控制流测试覆盖准则和数据流测试覆盖准则
* 控制流测试覆盖准则

􀂄 包括语句覆盖、分支覆盖、条件覆盖、判定-条件覆盖、路径覆盖

* 数据流测试覆盖准则

􀂄 包括定义覆盖、引用覆盖、定义-引用覆盖等准则

### 测试覆盖准则的作用

* 定量地规定软件测试需求
* 指导测试数据的选择
* 度量测试数据集揭示软件特定特征的能力
* 对测试结果和软件可靠性评估具有重要影响

# 等价类测试

## 等价类

* + 弱一般等价类测试

在测试组合中，每个等价类至少包含一次。

* + 强一般等价类测试

包含所有等价类的测试组合。

* + 弱健壮等价类测试

仅考虑了对变量维度的无效的输入变量。

* + 强健壮等价类测试

考虑所有不同等价划分的无效输入变量。即测试所有种类的无效输入集。

## 等价类测试方法讨论

1、按照规格说明中的“输入条件”（或者输出条件）划分等价类

* + 确定等价类的原则
    - 有效等价类
    - 无效等价类

2、设计一个新的测试用例，使其尽可能多的覆盖尚未覆盖的有效等价类；重复这一步骤，直到所有的有效等价类都被覆盖为止

3、设计一个新的测试用例，使其仅覆盖一个无效等价类；重复这一步骤，直到所有的无效等价类都被覆盖为止

## 输入域建模步骤：

i. 确定可测试的函数；

ii. 找到所有的输入参数；

1. 是非常需要创造性的一步；

2. 更多的特性意味着要做更多的测试；

3. 这一步其实就是简单的，机械的将参数转化为特性；

4. 需要考虑的特性包括：

a) 前置条件 & 后置条件

b) 变量间的联系；

c) 特殊变量间的联系；（如null，0等）

iii. 输入域建模；（即根据输入参数的特定进行等价类划分）

1. 包含有效值，无效值，特殊值；

2. 对一些块的划分；

3. 探索域的边界；

4. 包含哪些正常使用的值；

5. 尽量平衡每个块中的特性；

6. 检查等价类间的完整性和不相交性。

iv. 根据测试的标准选择输入变量的组合。（此时用到各种CC的定义）

1. 当划分和参数定义下来以后，下一步就是选择测试值；

2. 使用相关的测试标准，选择有效测试集；

3. 最好的标准当然就是选择所有的组合，但是。。。

4. 测试的数量即是在每个特性中划分数量的连乘。

v. 在每个划分中选择合适的值。

1. 应用边界值分析，On/Off - point，谓词，路径条件符号执行等相关知识。

d) 输入域建模分类

i. 基于接口的方法：（Interface-Based Approach）

1. 适用于简单的应用，并可部分自动化。

ii. 基于功能的方法：（Functionality-Based Approach）

1. 要做更多的设计工作。并有可能导致更多的测试。

# 路径测试和覆盖！！

## 各种覆盖标准 ！！

* NC——Node
* EC——Edge
  + EPC——Edge-Pair
  + CPC——Complete Path
  + SPC——Specified Path
  + PPC——Prime Path

**Simple Path :** A path from node ni to nj is simple if no node appears more than once, except possibly the first and last nodes are the same

* + - No internal loops
    - Includes all other subpaths
    - A loop is a simple path

**Prime Path :** A simple path that does not appear as a proper subpath of any other simple path——尽量长的path

* + SRTC——Simple Round Trip
  + CRTC——Complete Round Trip

**Round-Trip Path :** *A prime path that starts and ends at the same node*

* BPT——基本路径测试（圈）详见下面
* DFT——数据流测试
  + ADC——All-defs
  + AUC——All-uses
  + ADUPC——All-du-paths
* DC­——Decision——PC——Predicate
* CC——Condition——CC——Clause
* C/DC——CC ∩ DC——CoC
* MC/DC——ACC ∩ DC——ACC——Active——ICC
  + GACC——General——无所谓？
  + RACC——Restricted （受限的）次要条件不可变
  + GACC——Correlated（关联的）次要条件可变

## 基本路基覆盖（BPC）

* 1、绘制程序的控制流图
* 2、计算McCabe圈复杂度（设为n）
* 3、确定基本路径集的确定，即构造n条独立路径
  + (1) 任意构造一条从（唯一）入口结点到（唯一）出口结点的路径，将该路径加入基本路径集
  + (2) 修改基本路径集中路径，至少经过一条以前未走过的边，将新路径加入基本路径集
  + 重复第(2)步，直到基本路径集中包含n条路径
* 4 设计测试用例，使基本路径集中的路径能走通

# Software Performance Test

## Load testing

Load testing is the simplest form of performance testing

## Stress testing（压力）

Stress testing is normally used to understand the upper limits of capacity within the system

## Endurance testing (soak testing)（持久性）

Endurance testing is usually done to determine if the system can sustain the continuous expected load

## Spike testing（最大值）

Spike testing is done by suddenly increasing the number of, or load generated by, users by a very large amount and observing the behavior of the system

## Configuration testing（配置）

Rather than testing for performance from the perspective of load ，tests are created to determine the effects of configuration changes to the system's components on the system's performance and behavior.

A common example would be experimenting with different methods of load-balancing.

## Isolation testing （隔离）

Isolation testing is not unique to performance testing but a term used to describe repeating a test execution that resulted in a system problem.

Often used to isolate and confirm the fault domain.

# 配置测试

## 配置测试综述

配置测试的目的是保证被测试的软件在尽可能多的硬件平台上运行。

Configuration testing is the process of checking the operation of the software you are testing with all these various types of hardware

如果开始准备进行软件的配置测试，就要考虑哪些配置与程序的关系最密切。

理想情况是所有生产厂家都严格遵照一套标准来设计硬件，那么使用这些硬件的软件就会毫无疑问地正常运行。

但遗憾的是，标准并没有被严格遵守。有时，标准是相当松散的——称为规范。

## 分离配置缺陷

判断缺陷是配置问题而不仅仅是普通缺陷最可靠的方法是，在另外一台有完全不同配置的计算机上一步步执行导致问题的相同操作，如果缺陷没有产生，就极有可能是特定的配置问题，在独特的硬件配置下才会暴露出来。

谁来修复配置缺陷？

首先，要找出问题所在。这通常是动态白盒测试员和程序员调试的工作。一个配置问题产生的原因不少，全都要求有人在不同的配置中运行软件时仔细检查代码，以找出缺陷：

（1）软件可能包含在多种配置中都会出现的缺陷。

（2）软件可能只包含在某一个特殊配置中出现的缺陷。

（3）硬件设备或者其设备驱动程序可能包含仅由软件揭示的缺陷。

（4）硬件设备或者其设备驱动程序可能包含一个借助许多其它软件才能看出来的缺陷——尽管它可能对测试的软件特别明显。

前两种情况，显然要由项目小组负责修复缺陷。

后两种情况，责任不那么清晰。如果该硬件设备属于流行产品，被各界广泛使用，那么，开发小组需要针对缺陷对软件做修改，即使软件的运行是正确的。

归根结底，无论问题出在哪里，解决问题都是开发小组的责任。

## 计算工作量

配置测试工作量可能非常巨大。  
假设我们有一种新的3D游戏，画面丰富，具有多种音效，允许多个用户联机对战，还可以打印游戏细节以便进行策划。  
此时，至少我们需要考虑各种图形卡、声卡、网卡和打印机进行配置测试。如果决定进行完整、全面的配置测试，检查所有可能的制造者和幸好组合，就会面临巨大的工作量。  
市场上大致有336种显卡，210种声卡，1500种网卡，1200种打印机。则测试组合的数目就是336×210×1500×1200，总计上亿种，规模之大难以想象。  
减少麻烦的答案是等价划分。需要找出一个方法把巨大无比的配置可能性减少到尽可能控制的范围。由于没有完全测试，因此存在一定的风险，但这正是软件测试的特点。

## 执行任务

确定测试哪些设备和如何测试的决定过程是相当直观的等价划分工作。什么重要，怎样才会成功，是决定的内容。

1、确定所需的硬件类型

联机注册：在选择用哪些硬件来测试时容易忽略的一个特性例子是联机注册。  
如果软件有联机注册功能，就需要把调制解调器和网络通信考虑在配置测试之中。  
2、确定有哪些厂商的硬件、型号和驱动程序可用  
确定要测试的设备驱动程序，一般选择操作系统附带的驱动程序、硬件附带的驱动程序或者硬件或操作系统公司网站上提供的最新的驱动程序。  
3、确定可能的硬件特性，模式和选项  
4、将确定后的硬件配置缩减为可控制的范围  
假设没有时间和计划测试所有配置，就需要把成千上万种可能的配置缩减到可以接受的范围——即要测试的范围。  
一种方法是把所有配置信息放在电子表格中，列出生产厂商、型号、驱动程序版本和可选项。软件测试员和开发小组可以审查这张表，确定要测试哪些配置。  
注意：用于把众多配置等价划分为较小范围的决定过程最终取决于软件测试员和开发小组。这没有一个定式，每一个软件工程都不相同，都有不同的选择标准。一定要保证项目小组中的每一个人（特别是项目经理），搞清楚什么配置要测试（什么不测试），选择它们引起的变化有哪些。  
5、明确与硬件配置有关的软件唯一特性  
不应该也没有必要在每一种配置中完全测试软件。只需测试哪些与硬件交互时互不相同的特性即可。  
选择唯一特性进行尝试并非那么容易，首先应该进行黑盒测试，通过查看产品找出明显的特性，然后与小组成员交流，了解其内部的白盒情况。最后会惊奇的发现这些特性与配置有一些紧密的关联。  
6、设计在每一种配置中执行的测试用例  
（1）从清单中选择并建立下一个测试配置；  
（2）启动软件；  
（3）打开文件configtest.doc；  
（4）确认显示出来的文件正确无误；  
（5）打印文档；  
（6）确认没有错误提示信息，而且打印的文档符合标准；  
（7）将任何不符之处作为软件缺陷记录下来。  
实际上，这些步骤还有更多内容，包括具体要做什么、找什么的细节和说明。目标是建立任何人都可以执行的步骤。  
7、在每种配置中执行测试  
执行测试用例，仔细记录并向开发小组报告结果，必要时还要向硬件生产厂商报告。  
明确配置问题的准确原因通常很困难，而且非常耗时，软件测试员需要和程序员紧密合作。  
如果软件缺陷是硬件的原因，就利用生产厂商的网站向其报告问题。  
8、反复测试直到小组对结果满意为止  
配置测试一般不会贯穿整个项目期间。最初可能会尝试一些配置，接着整个测试通过，然后在越来越小的范围内确认缺陷的修复。最后达到没有未解决的缺陷或缺陷限于不常见或不可能的配置上。

## 获得硬件

购买每一样硬件则费用很高昂。  
（1）只买可以或者将会经常使用的配置；  
（2）与硬件厂商联系，看能否租借甚至赠送某些硬件；  
（3）向全公司的人询问其家里是否有硬件。

## 明确硬件标准

了解硬件说明书的一些细节，有助于做出更多清晰的等价划分决定。

## 对其它硬件进行配置测试

根据从设备使用者、项目经理或者销售人员那里获得的信息来建立硬件的等价划分。开发测试用例，收集所选硬件，执行测试。

## 总结

进行配置测试是软件测试新手经常被指派的工作，因为它容易定义，是基本组织技能和等价划分技术的入门；是与其它项目小组成员合作的任务；是经理快速验证结果的手段。缺点是有可能很繁杂。

# 兼容性测试

本章主要讲述与上一章类似的交互测试领域——检查软件是否能够与其它软件正确协作。  
随着用户对来自各个厂商的各种类型程序之间共享数据能力和充分利用空间同时执行多个程序能力的要求，测试程序之间能否协作变得越练越重要了。  
软件兼容性测试工作的目标是保证软件按照用户期望的方式进行交互。

## 兼容性测试综述

软件兼容性测试（Software Compatibility Testing）是指检查软件之间是否能够正确地进行交互和共享信息。  
对新软件进行软件兼容性测试，需要解决：  
（1）软件设计要求与何种其它平台和应用软件保持兼容？如果要测试的软件是一个平台，那么设计要求什么应用程序在其上运行？  
（2）应该遵守何种定义软件之间交互当地标准或者规范？  
（3）软件使用何种数据与其它平台和软件交互和共享信息？  
这些问题的答案是基本的静态测试——既有黑盒又有白盒，包括整体分析产品说明书和所有支持说明书。

## 常用术语

1、术语  
向后兼容（Backward Compatible）：是指可以使用软件的以前版本；

Load old data

Import old data

Exchange new data

向前兼容（Forward Compatible）：是指可以使用软件的未来版本。

注意：并非所有软件或者文件都要求向前兼容或者向后兼容。这是软件设计者需要决定的产品特性，而软件测试员应该为检查软件向前或向后兼容性所需的测试提供相应的输入。  
2、测试多个版本的影响  
测试平台和软件应用程序多个版本相互之间能否正常工作可能是一个艰巨的任务，假定对一个流行操作系统的新版本进行兼容性测试。  
注意：在开始兼容性测试任务之前，需要对所有可能的软件组合等价划分，使其成为验证软件之间正确交互的最小有效集合。  
由于不可能在一个操作系统上全部测试数千个软件程序，因此需要决定测试哪些是最重要的。决定要选择的原则是：  
（1）流行程度；  
（2）年头：应选择近三年以内的程序和版本；  
（3）类型：把软件分为绘图、文字输入、财务、数据库、通信等类型；  
（4）生产厂商：另一个原则是根据制作软件的公司来选择软件。  
以上是新操作系统平台的兼容性测试的。

## 标准和规范

1、研究可能适用于软件或者平台的现有标准和规范  
（1）高级标准：是产品普遍遵守的规则；  
（2）低级标准：是本质细节；  
两者都很重要，都需要测试以保证兼容。  
2、高级标准和规范  
如：Microsoft Windows认证徽标  
要求：软件必须通过由独立测试实验室执行的兼容性测试，其目的是确保软件在操作系统上能够稳定可靠地运行。  
认证徽标对软件有以下几点要求：  
（1）支持三键以上的鼠标；  
（2）支持在C:和D:以外的磁盘上安装；  
（3）支持超过DOS8.3格式文件名长度的文件名；  
（4）不读写或者以其它形式使用旧系统文件win.ini、system.ini、autoexec.bat和config.sys。  
3、低级标准和规范  
通信协议、编程语言语法以及程序用于共享信息的任何形式都必须符合公开的标准和规范。  
低级兼容性标准可以视为软件说明书的扩充部分。

## 数据共享兼容性

在应用程序之间共享数据实际上是增强软件的功能。写得好的程序支持并遵守公开标准；允许用户与其它软件轻松传输数据，这样的程序可称为兼容性极好的产品。  
（1）文件保存和文件读取是人人共知的数据共享非法。  
（2）文件导出和文件导入是许多程序与自身以前版本、其它程序保持兼容的方式。  
为了测试文件的导入特性，需要以各种兼容文件格式创建测试文档——可能要利用实现该格式的原程序来创建。  
（3）剪切、复制和粘贴是程序之间无需借助磁盘传输数据的最常见的数据共享方式。  
（4）DDE，COM（Component Object Model）和OLE是windows中在两个程序之间传输数据的方式。  
DDE表示动态数据交换；OLE表示对象链接和嵌入。  
DDE和OLE数据可以实时地在两个程序之间流动。

## 总结

（1）对兼容软件的所有可能选择进行等价划分，使其成为可以控制的范围；  
项目经理要认可测试清单，并接受由于为完全测试而引起的风险。  
（2）研究适用于测试软件的高级/低级标准和规范；  
把它们当作产品说明书的补充内容。  
（3）测试软件程序之间不同的数据流动方式。  
其中的数据交换就是程序之间保持兼容的因素。

# 多语言测试

如果你是一个有竞争力的软件测试员，并且熟练掌握除英语之外的一门外语，你就有了很有价值的技能。

## 使文字和图片有意义

软件的国际化  
除了语言，还需要考虑地域（region或locale）——用户的国家和地理位置。  
使软件适应特定地域特征，照顾到语言、方言、地区习俗和文化的过程称为本地化（localization）或国际化（internationalization）。测试此类软件称为本地化测试。

## 翻译问题

管翻译只是整个本地化工作的一部分，但是从测试角度看这是重要的一环，最明显的问题是如何测试用其它语言做的产品。  
软件测试员或者测试小组至少要对所测试的语言基本熟悉，能够驾驭软件，看懂软件显示的文字，输入必要的命令执行测试。  
注意：软件测试小组一定要有人对测试的语言比较熟悉。  
对多种语言的情况，可以委托本地化测试公司进行测试。  
（1）文本扩展（text expansion）  
实践证明，当英语被翻译为其它语言，用来表达同一事务时往往需要加一些字符。  
一个好的大拇指规则是每个单词长度预计增加100％。  
因为这些扩展现象，故必须仔细测试可能受到变长了的文本影响的软件部分，要找出没有正确换行、截断的和连字符位置不对的文本；还要找到虽然文本有足够的扩展空间，但这是通过把其它的文本挤出去来实现的情况。

变长了的文本还可能导致主程序失败，甚至系统崩溃。  
（2）ASCII、DBCS和Unicode  
ASCII字符集只能表示256种不同的字符——远不足以表示所有语言的全部字符。  
当开始为不同语言开发软件时，就需要找到克服该限制的解决方案。  
常用的一个方法是代码页（code page）技术，实质是ASCII表的替换，每一种语言用一个不同的代码页。  
这个方法虽然笨，但对于少于256个字符的语言还是可行的。但对于中文、日文则不行。这时，使用DBCS（双字节字符集）的系统提供对超过256个字符的语言的支持。用两个字节代替一个字节来表示最多可容纳65536个字符。  
代码页和DBCS在许多情况下足够了，但是会遇到一些问题，最重要的是兼容性问题。  
解决这个麻烦的方法是使用Unicode标准。  
Unicode为每一个字符提供唯一编号，无论何种平台，无论何种程序，无论何种语言。  
（3）热键和快捷键  
在软件的本地化版本中，需要测试所有热键和快捷键工作是否正常，而且使用起来不困难。  
（4）扩展字符  
本地化软件，甚至非本地化软件中存在的一个常见问题是扩展字符（extended characters）。  
扩展字符指的是普通英文字母A~Z和a~z之外的字符。  
测试扩展字符的方法是找出软件所有接受字符输入和输出之处。在每一处尝试使用扩展字符，看能否与常规字符一样处理。对话框、文本域都是合适的对象。  
技巧：测试扩展字符是否被正确处理的最简单的方法是，把它们加入测试的标准字符所在的等价划分之中。  
（5）字符计算  
与扩展字符有关的问题是软件在对其进行计算时如何接受解释。  
要弄清楚测试的语言采用什么样的排序规则，并开发测试用例专门检查排列次序的正确性。  
扩展字符计算打破的另一个领域是大小写转换。  
（6）从左向右和从右向左读  
翻译中有一个大难题是某些语言（例如希伯莱文和阿拉伯文）从右向左读，而不是从左向右读。  
幸好大多数主要操作系统提供了处理这些语言的内部支持。如果没有这一点，完成任务几乎是不可能的。即便这样，翻译这样的文本也不是容易的事。  
（7）图形中的文字  
另一个翻译问题是处理图形中的文字。  
它的影响是当软件本地化时，每一个图标都要改变，以反映新的语言。  
（8）让文本与代码脱离  
所有文本字符串、错误提示信息和其它可以翻译的内容都应该存放在与源代码独立的文件中。  
大多数本地化人员不是程序员，也没有必要是。让其修改资源文件（resource file）的简单文本文件，该文件包含软件可以显示的全部信息。  
当软件运行时，通过查找该文件来引用信息，不管信息的内容是什么，都按照原文显示。  
这就是说，对于白盒测试员来说，检查代码，确保没有任何嵌入的字符串未出现在外部文本文件中很重要。  
这个问题的另一个变化形式是当代码动态生成文本信息时。

## 本地化问题

翻译问题只是全部问题的一半。  
翻译文字和允许字符串包含不同字符和长度都不难，难的是修改软件使其适应国外市场。  
经过准确翻译和仔细测试的软件是精确和可靠的，但是如果程序员不考虑本地化的问题，程序就可能不够准确和高质量。  
（1）内容  
这里的内容是指产品中除了代码之外的所有东西。  
以下清单给出了本地化问题要仔细审查的各类内容：范例文档，图标，图片，声音，视频，帮助文件，有边界争端的地图，市场宣传材料，包装，Web链接。  
（2）数据格式  
不同的地区在诸如货币、时间和度量衡上使用不同的睡觉单位格式。  
在本地化时需要修改程序代码。  
如果测试本地化软件，就需要对当地使用的度量单位非常熟悉，为了正确测试软件，需要从原版软件创建的测试数据中建立不同的等价划分。

## 配置和兼容性问题

（1）国外平台配置  
键盘也许是语言依赖性最大的硬件——键盘布局。  
从根本上讲，软件可能会用到的任何外设都要在平台配置和兼容性测试的等价划分中去考虑。  
（2）数据兼容性

## 测试量的大小

如果软件从一开始就考虑到了本章所述的问题，那么本地化版本中包含更多软件缺陷和增大测试量的风险就很小。  
注意：本地化测试量的要求是一个有风险的选择。  
另一个问题关系到整个软件产品中什么需要改变。如果本地化工作只限于修改诸如文本和图形等内容——不是代码，——测试工作可能只是对改动进行合法性检查。如改动基本代码，则需考虑测试代码，并且检查功能和内容。

# 可用性测试

易用性（Useability）是交互的适应性、功能性和有效性的集中体现。  
人体工程学（ergonomics）是一门将日常使用的东西设计为易于使用和实用性强的学科。  
人体工程学的主要目标是达到易用性。

## 用户界面测试

用于与软件交互的方式称为用户界面或UI。

## 优秀UI的构成

软件测试员要负责测试软件的易用性，包括其用户界面。  
记住，软件测试员不需要去设计UI，只需要把自己当作用户，然后去找出UI中的问题。  
优秀UI具备的七个要素：  
（1）符合标准和规范  
最重要的用户界面要素是软件符合现行的标准和规范——或者有真正站得住脚的不符合的理由。  
注意：如果测试在特定平台上运行的软件，就需要把该平台的标准和规范作为产品说明书的补充内容。像对待产品说明书一样，根据它建立测试用例。  
这些标准和规范由软件易用性专家开发。它们是经由大量正规测试、使用、尝试和错误而设计出的方便用户的规则。  
也并非要完全遵守准则，有时开发小组可能想对标准和规范有所提高。  
平台也可能没有标准，也许测试的软件就是平台本身。  
在这种情况下，设计小组可能成为软件易用性标准的创立者。  
（2）直观  
用户界面是否洁净、不唐突、不拥挤？  
UI的组织和布局合理吗？  
有多余功能吗？  
帮助系统有效吗？  
（3）一致  
如果软件或者平台有一个标准，就要遵守它。如果没有，就要注意软件的特性，确保相似的操作以相似的方式进行。  
快捷键和菜单选项  
术语和命名  
听众  
诸如OK和Cancel按钮的位置。  
（4）灵活  
多种视图的选择：  
状态跳转  
状态终止和跳过  
数据输入和输出  
（5）舒适  
软件使用起来应该舒适，不能给用户工作制造障碍和困难。  
恰当；  
错误处理；  
性能。  
（6）正确  
要测试正确性，就是测试UI是否做了该做的事。  
注意：市场定位偏差、语言和拼写、不良媒体、WYSIWYG（所见即所得）。  
（7）实用  
是否实用事优秀用户界面的最后一个要素。

## 为有残疾障碍的人员测试：辅助选项测试

辅助选项测试（accessibility testing）也就是为有残疾障碍的人测试。  
残疾有许多种：视力损伤、听力损伤、运动损伤、认知和语言障碍。  
（1）法律要求：  
开发残疾人可以使用的用户界面的软件有一些法律规定。在美国，有3条法律：  
美国公民残疾人条例（ADA）声明  
居民条例第508款  
通信条例第255款  
（2）软件中的辅助特性  
软件可以有两种方式提供辅助。  
最容易的方式是利用平台或者操作系统内置的支持。  
如果测试的软件不在这些平台上运行，或者本身就是平台，就需要定义、编制和测试自己的辅助选项。  
注意：如果正在测试产品的易用性，一定要专门为辅助选项建立测试用例。  
如windows系统，提供了：粘滞键，筛选键，切换键，声音卫士，声音显示，高对比度，鼠标键，串行键。

## 总结

总之，不要让易用性测试的模糊性和主观性阻碍测试工作。易用性测试的模糊和主观是固然的，即使设计用户界面的专家也会承认有的地方是这样的。

# 测试文档

软件产品由大量工作和为数不少的非软件部分组成，非软件部分主要是文档。  
软件文档最常见的是拷贝到软件安装盘的readme文件。  
软件测试员通常不限于仅测试软件，而要负责组成整个软件产品的各种部分。保证文档的正确性也在职责范围之内。

## 软件文档的类型

如果软件文档仅有readme文件，测试很简单。要保证该文档包含应有的所有材料，全部内容从技术角度讲准确无误，还要进行拼写检查和磁盘病毒扫描，这就是文档测试的内容。  
现在，软件文档要占到整个产品的一大部分。  
以下是可以归类于文档的软件组成部分。  
（1）包装文字和图形  
包括盒子、纸箱和包装纸。  
文档困难包含软件的屏幕截图、功能列表、系统要求和版权信息。  
（2）市场宣传材料、广告以及其它插页  
（3）授权/注册登记表  
这是客户注册软件时填写并寄回的卡片，也可以作为软件的一部分，显示在屏幕上让用户阅读、认可，并完成联机注册。  
（4）EULA  
代表最终用户许可协议。可能在软件安装过程中弹出显示在屏幕上。  
（5）标签和不干胶条  
可能出现在媒体、包装盒或者打印材料上。  
它们还包括序列号不干胶条盒封EULA信封的标签。  
（6）安装和设置指导  
有时该信息直接打印在磁盘上。对于复杂软件，可以是完整的手册。  
（7）用户手册。  
主要是联机手册。  
（8）联机帮助  
联机帮助一般可以和用户手册互换使用，有时甚至取代用户手册。  
联机帮助索引和搜索功能，用户查找所需信息更加容易。  
（9）指南、向导和CBT（计算机基础训练）  
这些工具将编程代码和书写文档融合在一起，一般是内容和类似宏的高级编程的混合体，通常捆绑在联机帮助系统中。  
（10）样例、示例和模板  
编译器可能用一小段代码来演示如何使用编程语言的某些方面。  
（11）错误提示信息

## 文档测试的重要性

软件用户把这些独立的非软件部分当做整个软件的一部分。  
好的软件文档以几种方式确保产品的整体质量：  
（1）提高易用性  
（2）提高可靠性  
可靠性是指软件稳定和坚固的程度。  
（3）降低支持费用  
好的文档可以通过恰当的解释和引导用户解决困难来预防这种情况。  
注意：作为软件测试员对待文档要像对待代码一样给予同等关注和投入。它们对用户是一样的。如果没有要求测试文档，一定要把此作为问题提出来并在整个测试计划中包括这部分。

## 审查文档时要找什么

测试文档有两个等级。  
如果是非代码，测试就是静态过程，可以视之为技术编辑或技术校对。  
如果是文档和代码紧密结合在一起，就要进行动态测试。这种情况属于真正的软件测试。  
注意：无论文档是不是代码，像用户那样对待它都是非常有效的测试方法。  
仔细阅读，按照每个步骤操作，检查每个图形，尝试每个示例。  
如果有简单代码，测试代码是否按照描述的方式进行。  
最后，如果文档是软件驱动的，就要像软件其余部分一样进行测试。  
检查索引表是否完整，搜索结果是否正确，超级链接和热点是否跳转到正确的页面。利用等价划分技术确定尝试哪些测试用例。

## 文档测试的实质

（1）文档常常得不到足够的重视；  
如果负责测试软件中的一个领域，一定要为伴随代码的文档测试做出预算，像对待软件一样给予关注。  
（2）编写文档的人可能对软件做什么不甚了解；  
文档作者不必是软件功能方面的专家。最重要的是，指出发现的代码中难以使用或者难以理解之处，让它们在文档中更好地解释。  
（3）印刷文档制作要花不少时间，可能是几周，甚至几个月。  
由于这个时间差，软件产品的文档需要在软件完成之前完稿——锁定。

## 总结

从用户的角度看，软件文档和软件都是同样的产品。联机帮助索引遗漏一个重要条目，安装指导中存在错误步骤，或者出现显眼的拼写错误，都属于与其它软件失败一样的软件缺陷。  
如果正确地测试文档，就可以在用户使用之前发现这些缺陷。

# 安全测试

黑客、病毒、蠕虫、间谍软件、后面程序、木马、拒绝服务攻击都是计算机的安全问题。

## 驾驶攻击

随着在城域网中普及无线高保真（WiFi）网络，黑客们可以驾驶车子，带着笔记本，在城市的街道上兜圈子，就可以搜索到未受保护的无线网络，这种技术就是“驾驶攻击”。

## 了解动机

作为软件测试员很重要的一点是要了解为什么有人要攻击你的软件。  
了解动机能帮助软件测试员考虑到测试的软件中有哪些安全方面的漏洞。  
安全产品：是指产品在系统的所有者或管理员的控制下，保护用户信息的保密性、完整性、可获得性，以及处理资源的完整性和可获得性。  
安全漏洞：是指使产品不可行的缺陷——即使是正确地使用产品时——来防止攻击者窃取系统的用户权限、调节操作、破坏数据，或建立未授权的信任。  
黑客：精通计算机编程和使用的人，电脑玩家。使用编程技能来获得对计算机网络或文件的非法访问的人。  
黑客想获得系统访问权限的5个动机是：  
（1）挑战/成名  
（2）好奇  
（3）使用/借用  
（4）恶意破坏  
（5）偷窃

## 威胁模式分析（threat modeling）

威胁模式分析目的是由评审小组查找产品特性设置方面可能会引起安全漏洞的地方。  
根据这些信息，小组可以选择对产品做修改，花更多的努力设计特定的功能，或者集中精力测试潜在的故障点。  
最终，这样理解会使产品更加安全。  
注意：除非产品开发小组的每个人——包括项目经理、程序员、测试员、技术文档写作员、市场人员、产品支持——都理解和认同可能存在的安全威胁，否则小组不可能开发出安全的产品来。  
执行威胁模式分析并非软件测试员的责任。  
这个责任应该落到项目经理的任务清单上，并非项目小组每个成员都要参与。  
一个复杂的系统要求全面的威胁模型分析来确认安全漏洞。  
（1）构建威胁模型分析小组  
对于小组来说，重要的一点是了解他们的最初目标不是解决安全问题，而是确定安全问题。  
在后期可以举行一些小规模的特定团队参加的会议，以隔离安全威胁，设计解决方案。  
（2）确认价值  
考虑系统所有的东西对于一个入侵者来说价值有多大。  
（3）创建一个体系结构总体图  
要确认计划用在软件中的以及如何实现互联的技术。  
小的威胁模型分析小组会创建一个体系结构图表示出主要的技术模块和它们之间如何通信。  
（4）分解应用程序  
这是一个格式化的过程，用来确认数据所在位置以及如何通过系统。  
（5）确认威胁  
一旦完全理解了所有的部分（价值、体系结构、数据），威胁模型分析小组可以转向确认威胁。  
每一个部分都应该考虑成为威胁目标，并且应假设它们会受到攻击。  
（6）记录威胁  
每个威胁都必须用文档记录，并且应进行跟踪以确保其被解决。  
文档是一种简单方式，用于描述威胁、目标、攻击可能采用的方式、系统用于防御攻击有哪些反制手段。  
（7）威胁等级评定  
理解并非所有的威胁生来就平等这一点很重要。

## 恐怖公式（DREAD Formula）

1. 潜在的危害  
   （2）可反复性  
   （3）可利用性  
   （4）受影响的用户  
   （5）可发现性

## 软件安全是一项功能吗？软件漏洞是一个缺陷吗？

软件安全可以简单地看做是软件产品或系统的另外一项功能。  
软件测试员可能会负责测试软件的整个安全性，或者可能仅仅负责测试被分配测试的功能是安全的。  
软件测试员不需要拿到一份清楚明白地定义软件安全性是如何实现的产品说明书。  
软件测试员也不能假设威胁模型分析是完全和准确的。  
技巧：测试安全缺陷是失效性测试行为，也常常覆盖产品中没有被完全理解和说明的部分。

## 了解缓冲区溢出

任何软件产品中都有一个安全问题——缓冲区溢出。  
数据引用错误——既使用没有被正确申明和初始化的变量、常数、数组、字符串或记录引起的缺陷。  
缓冲区溢出就是这种缺陷。  
由于字符串的不正确处理引起的缓冲区溢出是目前为止最为常见的一种代码编写错误，其结果是导致安全漏洞。

## 使用安全的字符串函数

通用C和C++函数中容易引起缓冲区溢出的编码错误。这些函数自身并不差，但是要安全的使用它们，需要在程序员这边进行更为深入的错误检测。如果忽略了这种错误检测，代码就会有安全漏洞。  
衡量一下这种疏忽带来的风险，最好还是开发或改进一组新的函数，即用强壮、完全测试通过的、文档齐全的新函数集替代这些容易引起问题的函数集。  
这些新的函数，叫做安全字符串函数（Safe String Functions），在windowsXP的SP1版本、最新的windows DDK和平台SDK中已经具有。常用的操作系统、编译器，处理器也具有其它很多实现了安全字符串的商用的或免费的库。  
使用新函数的好处（部分）：  
（1）每个函数接收目标缓冲的长度作为输入。这样函数就能确保在写入时不会超过缓冲区的长度。  
（2）函数空字符中止所有的输出字符串，即使操作截断了预计的后果。  
（3）所有函数返回一个NTSTATUS值，该值只有一个可能成功的代码。调用函数能轻易地确定函数的执行是否成功。  
（4）每个都提供版本。一个支持单字节的ASCII字符，另一个支持双字节的Unicode字符。

## 计算机取证

注意：用户变更时未被删除的保留数据叫做潜在数据。潜在数据时潜在的安全漏洞，需要在小组采用的任何威胁模型分析中进行讨论。也许这些数据不会被看成是产品的问题，也许会被看成是一个大问题。  
潜在数据的更复杂的例子是由计算机安全专家用来发现可以用做犯罪调查的证据。

## 总结

没有计算机系统是安全的。为帮助设计一个安全的系统，在产品设计的最开始就必须注意安全的问题。  
测试员仅能测试软件的安全性，软件安全必须先计划、评审、设计，然后才是测试。

# 网站测试

网站测试囊括许多领域，包括配置测试、兼容性测试、易用性测试、文档测试、安全性测试，并且假如网站是面向全球范围的浏览者，还包括本地化测试。当然，黑盒、白盒、静态、动态测试都是要用上的。  
要关注：  
（1）网页的哪些基本部分需要测试  
（2）在网页测试中要运用哪些基本的白盒测试技术和黑盒测试技术  
（3）如何运用配置测试和兼容性测试  
（4）为什么易用性测试是网页的主要问题  
（5）如何使用工具协助网站测试

## 网页基础

简单来说，网页就是由文字、图片、声音、视频和超级链接组成的文档。  
在这些程序中，网站用户可以通过单击具有超级链接的文字和图片在网页间浏览，搜索单词或者短语，查看找到的信息。  
网页的特性：  
（1）不同大小、字体和颜色的文字；  
（2）图片和照片；  
（3）超级链接文字和图片；  
（4）不断滚动的广告；  
（5）下拉式文本选择框；  
（6）用户输入数据的区域。  
使网站更加复杂的特性如下：  
（1）自定义的布局，允许用户更改信息出现在屏幕上的位置；  
（2）自定义的内容，允许用户选择想看的新闻和信息；  
（3）动态下拉式选择框；  
（4）动态变化的文字；  
（5）取决于屏幕分辨率的动态布局和可选信息；  
（6）与不同网络浏览器、浏览器版本，以及硬件和软件平台的兼容性；  
（7）大量加强网页易用性的隐藏格式、标记和嵌入信息。  
安全电子商务网站更为复杂。

## 黑盒测试

最容易的测试起步是把网页或者整个网站当做是一个黑盒子。  
注意：在测试网站时，首先应该建立状态表，把每个网页当做不同的状态，超级链接当做状态之间的连接线。完整的状态图有利于对整个任何更好地进行审视。  
（1）文本  
网页文本应该当做文档对待，并依据文档测试的方法进行测试。  
检查核实读者的水平、术语、内容以及题目素材、准确度——特别是可能过期的信息，经常不断地检查拼写。  
注意：不要依赖拼写检查工具来做，尤其是用在网页文本内容的检查上。  
拼写检查工具可能只检查常规文本，但不检查包含在图片、滚动块、表单等中的文字。用拼写检查工具执行完所谓的完全拼写检查之后，检查者可能认为检查很彻底，但事实上网页中仍然会有拼写错误。  
如果有电子邮件地址、电话号码或者邮政编码等联系信息，要检查是否正确。  
保证版权声明正确、日期无误。  
测试每个网页是否都有正确的标题，标题文本的正确性。  
常常被忽视的一种文本是文字标签（ALT text），用以替代文字（ALTernate text）。当用户把鼠标光标移动到网页中的图片上时，可以看到弹出对图片语义信息的说明。  
注意：并非所有的浏览器都支持显示文字标签。  
通过大幅缩放浏览器窗口来检查文字布局问题。  
（2）超级链接  
链接可以和文字或者图片拴在一起。每一个链接都要检查，确保它的正常跳转。  
超级链接一定要明显，文字链接一般有下划线。  
如果链接打开电子邮件信息，就填写内容并发送，要确保能够得到回应。  
查找孤页，它是网站的一部分，但是不能通过超级链接访问，因为网页作者忘记把它挂接上。需要向网站设计人员索取网页清单，与自制的状态图进行比较。  
（3）图片  
图片中可能出现的许多软件缺陷在易用性测试时被掩盖下来，但是利用简单的黑盒测试方法可以检查一些明显的地方。图片的正确载入和显示的问题。  
如果网页中文本和图片交织在一起，要保证文字正确地环绕在图片周围。  
载入网页时的性能如何？  
（4）表单  
表单是指网页中用于输入和选择信息的文本框、列表框和其它域。  
域的大小是否正确？是否接受正确数据，拒绝错误数据？  
（5）对象和其它各种简单的功能  
网站可能包含诸如单击计数器、滚动文本选择框、变换的广告和站内搜索等特性。  
在计划网站测试时，要仔细验明每个网页上的所有特性，把每一个特性按照常规程序的功能对待，并利用所学的标准测试技术分别进行测试。

## 灰盒测试（gray box testing）

灰盒测试时白盒测试和黑盒测试的结合。灰盒测试把黑盒测试和白盒测试的界限打乱了，仍然把软件当做黑盒来测试，但是通过简单查看软件内部工作机制作为补充。  
网页特点使其非常适合进行灰盒测试。因为大多数网页是由HTML创建，在IE上很轻易地就能看到源代码。

## 白盒测试

网页的静态内容一般由HTML直接创建。同时，网页还包括可自定义和动态改变的内容。  
创建这些附加的动态特性需要用可以执行和支持判断分支结构的程序代码来补充。  
创建此类特性的流行Web编程语言：DHTML，Java，JavaScript，ActiveX，VBScript，Perl，CGI，ASP和XML。  
运用白盒测试不需要测试员一定成为这些语言的专家，而只要熟悉到能够阅读和理解这些语言，并根据代码的内容来确定测试用例即可。  
测试一些特性：  
（1）动态内容  
动态内容是根据当前条件发生变化的文字和图片——例如：日期时间、用户喜好或者特定用户操作。  
在HTML中嵌入JavaScript之类的简单脚本语言可以对这些内容编程，这称为客户端（client-side）编程。——在查看代码时可以使用灰盒测试。  
为了提高执行效率，大多数动态内容编程在网站服务器上进行，这称为服务端（server-side）编程，需要具有Web服务器的访问权限才能查看源代码。  
（2）数据库驱动的网页  
许多显示分类目录或者货物清单的电子商务网页是数据库驱动的。  
（3）用编程方法创建的网页  
许多网页，特别是包含动态内容的网页是用编程方法创建的。  
（4）服务器性能和加载  
流行的网站每天可能要接受数百万次点击。每一次点击都要从网站的服务器下载数据到浏览者的计算机上。  
如果测试一系统的性能和负载能力，就必须找到一种方法来模拟数百万个连接和下载。  
（5）安全性  
金融、医疗和其它包含个人数据的网站风险特别大，需要密切了解服务器技术来测试其安全性。

## 配置和兼容性测试

配置测试是在各种硬件和软件平台类型以及其不同的设置情况下检查软件运行的过程。  
兼容性测试是检查软件和其它软件一起运行的过程。  
要测试一个网站，需要考虑可能会影响网站运行和外观的硬件和软件配置：  
（1）硬件平台；  
（2）浏览器软件和版本；  
（3）浏览器插件；  
（4）浏览器选项；  
（5）视频分辨率和色深；  
（6）文字大小；  
（7）调制解调器速率。

## 易用性测试

易用性测试是一个难以定义的过程。遵守并测试一些基本规则有助于使网站更加有用。  
（1）盲目使用不成熟的新技术  
网站不应该靠吹嘘采用最新Web技术来吸引用户。  
（2）滚动文字、滚动块和不停运行的动画  
不要让网页上有不停移动的元素。  
（3)滚动显示的长页面  
减少滚动，所有重要的内容和导航选项应该位于页面顶端。  
（4）非标准的链接颜色  
指向用户未曾看过的页面的超级链接应该是蓝色；指向已经看过的页面的链接应该是紫色或者红色。  
（5）过期信息  
维护是加强网站内容的经济之道。  
（6）下载时间过长  
传统的人为因素规范指出：0.1秒是用户感觉系统反应迅速的极限；1秒是用户思路不间断的极限；10秒是用户完全丧失兴趣的最长响应时间。  
（7）缺少导航支持  
需要以在结构和位置上给人强烈感觉的形式进行支持；站点设计应该从很好地了解信息空间的结构开始，并把结构明确地传达给用户。  
（8）孤页  
所有网页一定要包含本身所属网站的明确指示，因为用户可能不经过主页而直接访问网页。  
（9）复杂的网站地址（URL）  
URL应该包含反映网站内容的本质的便于人们阅读的名称。  
（10）使用框架  
框架是允许在一个网页中显示其它网页的HTML技术。  
如果测试网站，就要充分利用测试员的权限报告易用性方面的软件缺陷。回顾基本用户界面设计技术，了解良好易用性的组成要素。

## 自动化测试

要对复杂的大型网站进行彻底的测试，就需要用自动化的测试工具了。

# 考试内容 ！！

## 题型与内容提要

### 单项选择8题24%

（专项测试应关注什么东西，比如兼容性、易用性、安全等专项测试活动）

### 简答3题20%

（第一题是软件测试的整体概念，第二题是测试技术的细节问题，还有一题发散思维，是描述一项测试活动应该从哪些方面进行，读清楚题干的要求）

### 应用题2题56%

（给你一个被测对象，要做一个XX测试，写出测试用例和执行结果，被测对象可能有很明显的错误）第一题是黑盒白盒测试技术的测试用例，比如要做一个等价类、路径、覆盖等测试，比如测试一个用户密码的等价类划分的情况；路径给一段短程序，做一个路径覆盖。

专项测试中关键部分是[兼容性、配置、安全性]测试的bug

## 测试的基本概念！！

* 测试的概念（原则）：

常识性的测试内容，比如所有的软件都有bug，bug是测不完的，不能被完全的修正，找到bug越多的模块隐含的bug越多，不能进行全路径覆盖、

* 测试的目的：

有不同的观点，一种观点认为软件测试应该把软件验证纳入工作范围内，另一种观点认为发现bug是软件测试的终极目标。验证扩展是指广义的测试，找到bug是狭义的测试。但是验证和确认是必不可少的，所以大多数认为是广义的软件测试，但是随着软件规模的扩大，验证活动也变得困难，比如可靠性测试就不好验证；为了排除心理障碍，也可以以找到bug作为终极目标，实用性较强

* 测试的步骤：

分阶段分层次（单元、集成、系统、验收等等步骤），但是因为标准存在，所以步骤可能不同

* 测试和软件的开发模型（开发生命周期的测试活动嵌入，测试活动需要被嵌入到生命周期中），比较晚的介入方式（测试晚于开发）、或者极限编程的测试驱动开发（先写测试用例）、或者专注于软件测试的软件开发模型（某一个开发活动与某个测试相关，会影响测试的标准和目标，所以要早期进行测试的计划和设计）

## 测试的基本技术！！

黑盒、白盒和灰盒：结构和功能的测试方法的掌握，基本技术的复习需要比较深入

静态和动态，有没有使用计算机相关的工具还是纯手工完成。静态测试是有流程上的要求，有检查表check list（说明要检查哪些项，或者像测试用例来列一些用是/否回答的问题）；动态测试利用计算机测试工具来完成

单元、集成、系统、验收：了解概念

## 基本路径测试（还有等价类、边界值、覆盖等的测试）

1. 绘制控制流图（图应当能够自解释，最简单的只有两种图示符号，一个是圈表示语句，圈中有编号表示语句编号，一个是箭头表述程序流向）
2. 基本路径需要计算圈复杂度（有多种计算方式，但是结果相同）：（1）图论方式，圈复杂度=v-n+2（边数-节点数+2），+2是因为程序流图是单入口单出口的有向图。（2）数在平面上的封闭区域（内外皆算）。（3）数控制节点（只有控制节点才会形成圈），如果控制节点是两个分支的话，则为控制节点+1；如果有三分支，则一个控制节点+2
3. 基本路径的个数是圈复杂度
4. 构造测试用例（同样适用于专项测试）

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 用例 | 测试结果 | Bug说明 |
|  |  |  |  |

## 专项测试重点概要

配置、兼容性、安全性、可靠性，其所测试的bug类型以及表现形式，以及相关的方向所要考虑的问题——**注意看文档**

* 配置测试（软硬件配置，向前兼容、向后兼容的问题）

目的是保证被测试的软件在**尽可能多的硬件平台上运**行。

* 兼容性测试（浏览器）

 目的是保证软件按照**用户期望的方式进行交互**。

* 本地化和国际化测试（语言特征的测试）

使软件适应特定地域特征，照顾到语言，方言，地区习俗和文化的过程称为本地化或国际化。

* 易用性测试（界面、菜单栏、界面的整体结构、空间、帮助系统、安装）

是交互的适应性、功能性和有效性的集中体现。

* 文档测试（文档是否齐全，文字是否一致性）

i. 软件产品由大量工作和为数不少的非软件部分组成，非软件部分主要是文档。

ii. 软件测试员通常不限于仅测试软件，而要负责组成整个软件产品的各种部分。保证文档的正确性也在职责范围之内。

* 安全性测试（用户输入身份验证的安全性，加密方式的安全性）

没有计算机系统是安全的。为帮助设计一个安全的系统，在产品设计最开始就必须注意安全问题。

* web网站测试（前面的东西应用到一个特定的web网站上面），测试用例要通过是/否来筛选答案

# 兼容性测试表格

## 浏览器兼容性测试记录样本表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Applets | JavaScript | ActiveX | VBScript |
| Internet Explorer 7.x |  |  |  |  |
| Internet Explorer 8.x |  |  |  |  |
| Internet Explorer 9.x |  |  |  |  |
| Netscape Navigator 6.x |  |  |  |  |
| Netscape Navigator 7.x |  |  |  |  |

## 浏览器和操作系统兼容性测试记录样本表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | IE7 | IE8 | IE9 | … |
| Windows XP Pro |  |  |  |  |
| Windows 2000 Ser |  |  |  |  |
| Windows 7 |  |  |  |  |
| … |  |  |  |  |

# 易用性测试表格

## 菜单界面测试用例样本表

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 测试用例及说明 | 测试结果 | 缺陷原因 |
| 1 | 菜单功能是否正确执行 |  |  |
| 2 | 下拉菜单是否根据菜单选项的含义进行分组 |  |  |
| 3 | 菜单是否有快捷命令方式 |  |  |
| 4 | 文本字体、大小和格式是否正确 |  |  |
| 5 | 菜单功能是否随当前的窗口操作加亮或者变灰 |  |  |
| 6 | 菜单功能的名字是否具有自解释性 |  |  |
| 7 | 菜单项是否有帮助 |  |  |
| 8 | 右键快捷菜单是否采用与菜单相同的准则 |  |  |
| 9 | 是否可以通过鼠标访问所有的菜单功能 |  |  |
| 10 | 是否恰当的列出了所有菜单功能和下拉式子功能 |  |  |
| 11 | 下拉式操作是否正常工作 |  |  |
| 12 | 是否根据系统功能进行合理分类，将选项进行分组 |  |  |
| 13 | 菜单深度是否控制在三层以内 |  |  |
| 14 | 菜单标题是否简明、有意义 |  |  |
| 15 | 是否根据使用频率排列 |  |  |
| 16 | 是否根据逻辑顺序排列 |  |  |
| 17 | 是否根据使用顺序排列 |  |  |
| 18 | 各级菜单显示格式和操作方式是否一致 |  |  |

## 帮助系统测试用例样本表

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 测试用例及说明 | 测试结果 | 缺陷原因 |
| 1 | 系统是否提供F1及时帮助功能 |  |  |
| 2 | 在界面上调用帮助时是否能及时定位到与操作对应的帮助 |  |  |
| 3 | 对功能采用及时帮助是否能准确定位到帮助系统的位置 |  |  |
| 4 | 利用帮助索引是否能定位到帮助主题和内容 |  |  |
| 5 | 是否具有打印功能 |  |  |
| 6 | 帮助目录是否划分并有层次 |  |  |
| 7 | 帮助内容描述准确，详细到可以解决问题 |  |  |
| 8 | 在系统不同位置激活帮助内容与当前操作是否想关联 |  |  |
| 9 | 是否有微帮助：由状态栏或者控件上提供有帮助文本 |  |  |

# 安全性测试表格

## 常见的输入验证漏洞表

|  |  |
| --- | --- |
| 漏洞 | 影响 |
| 超文本标记语言（HTML）输出流中有未经验证的输入 | 应用程序易受XSS攻击 |
| 用于生成SQL查询的输入未经验证 | 应用程序易受SQL注入攻击 |
| 依赖客户端的验证 | 客户端验证容易被忽略 |
| 使用输入文件名、URL或者用户名制定安全策略 | 应用程序易出现规范化错误并导致安全漏洞 |
| 对恶意输入仅采用应用程序筛选器 | 这基本上不可能达到目的，因为可能的恶意攻击范围过大。应该对输入进行约束、拒绝和净化 |

## 常见的身份验证漏洞表

|  |  |
| --- | --- |
| 漏洞 | 影响 |
| 弱密码 | 增加破解密码和词典攻击的风险 |
| 配置文件中使用明文凭据 | 可访问服务器的内部人员都能直接攻击该凭据 |
| 通过网络传递明文凭据 | 增加通过网络监控来盗取身份验证凭据的风险 |
| 账户的特权过强 | 增加与进程、账户泄露相关的风险 |
| 长会话 | 增加与会话劫持相关的风险 |
| 混用个性化数据的身份验证数据 | 个性化数据适用于永久的cookie，但身份验证的cookie不应是永久的 |

## 常见的授权漏洞表

|  |  |
| --- | --- |
| 漏洞 | 影响 |
| 依赖单个网关守卫 | 如果网关守卫被忽略或配置不当，用户能不经授权进行访问 |
| 不能依据应用程序身份锁定系统资源 | 攻击者可强制应用程序访问受限的系统资源 |
| 不能将数据库访问限定于特定存储过程 | 攻击者可利用SQL注入攻击来检索、操作或毁坏数据 |
| 特权隔离不充分 | 没有责任或能力对每个用户进行授权 |

## 常见的配置管理漏洞表

|  |  |
| --- | --- |
| 漏洞 | 影响 |
| 不安全的管理界面 | 未经授权的用户可重新配置应用程序，并访问敏感数据 |
| 不安全的配置存储 | 未经授权的用户可访问配置存储并获取机密信息 |
| 明文配置数据 | 可登录服务器的所有用户都能查看敏感的配置数据 |
| 管理员太多 | 使得管理员的审核和评价变得困难 |
| 进程账户和服务账户特权过高 | 导致特权提升攻击 |

## 常见的敏感数据处理漏洞表

|  |  |
| --- | --- |
| 漏洞 | 影响 |
| 不必要的情况下存储机密信息 | 大大增加安全风险 |
| 在代码中存储机密信息 | 如果代码位于服务器，攻击者可能下载它 |
| 明文形式存储机密信息 | 可登录服务器的所有用户都能看到机密信息 |
| 通过网络传递机密信息 | 增加通过网络监控盗取并篡改机密信息的风险 |

## 常见的会话管理漏洞表

|  |  |
| --- | --- |
| 漏洞 | 影响 |
| 通过未加密的通道传递会话标识符 | 攻击者可捕获会话标识符来窃取身份 |
| 会话生存期延长 | 增加会话劫持和重播攻击的风险 |
| 会话存储不安全 | 攻击者可访问用户的私有会话数据 |
| 查询字符串中的会话标识符 | 可通过在客户端修改会话标识符来窃取身份 |

## 常见的加密漏洞表

|  |  |
| --- | --- |
| 漏洞 | 影响 |
| 使用自定义加密 | 安全性低于经过检测的加密 |
| 使用过旧的加密算法或密钥过小 | 使用较新的加密算法，或者较大的密钥 |
| 不能确保加密密钥的安全性 | 加密密钥的安全性和加密数据同样重要 |
| 在延长的时间段使用同一密钥 | 如果长时间使用静态密钥很容易被发现 |

## 常见的参数操作漏洞表

|  |  |
| --- | --- |
| 漏洞 | 影响 |
| 无法验证所有的输入参数 | 易受到拒绝服务和代码注入的攻击，包括SQL注入和XSS |
| 未经加密的cookie中有敏感数据 | cookie数据可在客户端更改，或者通过网络传递获取并更改 |
| 查询字符串和表单字段中存在敏感数据 | 可在客户端更改 |
| 信任HTTP头信息 | 可在客户端更改 |
| 视图状态未保护 | 可在客户端更改 |

## 常见的异常管理漏洞表

|  |  |
| --- | --- |
| 漏洞 | 影响 |
| 无法使用结构化的异常处理 | 易受到拒绝服务的攻击，并容易出现逻辑错误 |
| 向客户端公开太多信息 | 攻击者可使用这些信息来规划和调整攻击 |

## 常见的审核和日志管理漏洞表

|  |  |
| --- | --- |
| 漏洞 | 影响 |
| 无法审核失败的登录 | 入侵企图得不到检测 |
| 无法确保审核文件的安全 | 攻击者可以掩饰攻击 |
| 无法跨应用程序进行审核 | 增加的抵赖的风险 |