My 测试

1. 题型
2. 单选(24%) 8道题

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

1. 简答(20%) 3道题

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

第三题发散思维，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

1. 应用(56%) 2道题

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

1. 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。

**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）：
3. 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

1. 基本路径的个数是圈复杂度
2. 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 |
|  |  |  |  |

1. 第二题:专项测试中关键部分是[兼容性、配置、安全性]测试的bug

专项测试寻找特定的bug：

1. 配置测试（软硬件配置，向前兼容、向后兼容的问题）
2. 兼容性测试（浏览器）
3. 本地化和国际化测试（语言特征的测试）
4. 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）
5. 文档测试（文档是否齐全，文字是否一致性）
6. 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）
7. 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

注意看文档

1. 知识要点
2. 测试的相关概念(PPT01)
3. 测试的概念(原则):
4. 使用人工或自动手段来运行或测定某个系统的过程(IEEE1983)
5. 在软件结构层次范围内或多种环境下评价软件功能点和行为的过程；
6. 根据规格说明书和用户需求验证&有效性确认软件产品；
7. 发现软件bugs。

(同学笔记)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

1. 测试的目的:

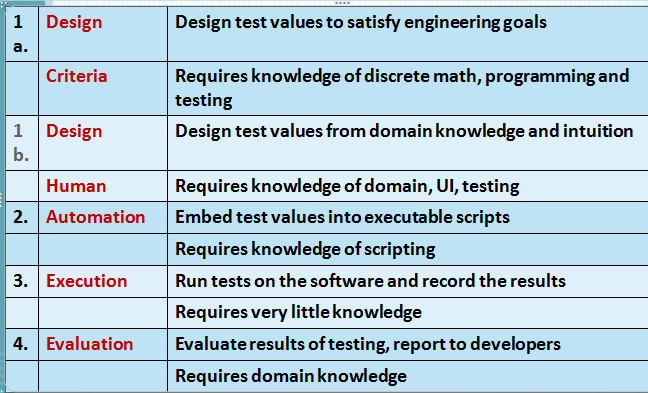
检验系统是否满足规定的需求或是弄清预期结果与实际结果之间的差别(IEEE1983)

(同学笔记)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

1. 几种测试活动



1. 测试的步骤(测试的几个阶段):

(同学笔记)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

1. 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

1. 集成测试

将经过单元测试的模块逐步进行集成和测试

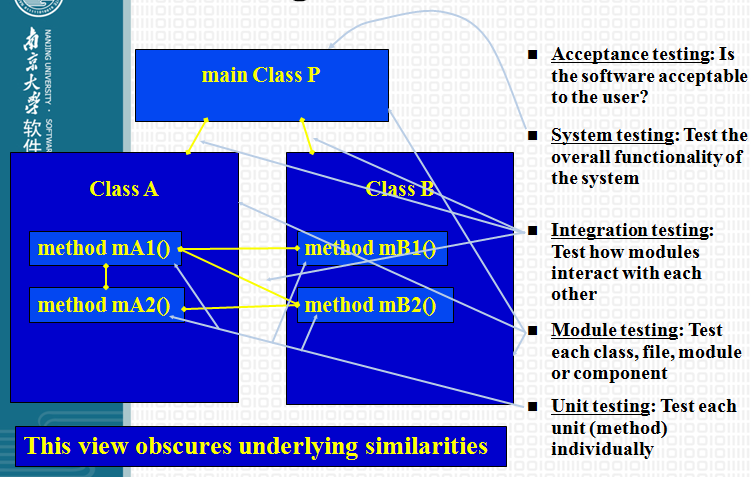
集成测试验证程序和**概要设计说明**的一致性

1. 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

1. 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



1. 一些术语
2. **Validation** & **Verification**

Validation : The process of evaluating software at **the end of** software development to ensure compliance with intended usage

Verification : The process of determining whether the products of **a given phase** of the software development process fulfill the requirements established during **the previous phase**

1. **Testing**(测试) & **Quality Assurance**(质量保证) (QA)

**测试**的目的是找bug,尽可能早得找到bug并保证它们被修复

**质量保证**的目的是建立改善软件开发过程的标准和方法来预防bug的产生

1. **Static** & **Dynamic** Testing 静态测试和动态测试

**静态测试**:**不用执行待测程序**

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

是对流程上的要求,有检查表check list

由人工进行:代码审查,代码走查,桌面检查

**动态测试**:**执行待测程序,有输入**

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

一般利用计算机测试工具来完成

核心内容:生成测试用例,运行程序,验证程序的运行结果

辅助工作:文档编制,数据管理,操作规程,工具应用

1. Software **Faults**, **Errors** & **Failures**

Fault:软件内的静态缺陷

Error:软件内部的不正确的状态,是一些fault的表现形式

Failure:外部的,产生与需求或其他描述中期待的行为不符的行为

1. **Testing** & **Debugging**

Testing: 找到能引起软件出错的输入

Debugging:The process of finding a fault given a failure

1. **Fault** & **Failure** Model

Failure能被观察到的三个条件:

1. Reachability(可达)
2. Infection(传染,即这种fault要能引起软件产生不正确的状态)
3. Propagation(传播,即软件产生的不正确的状态要能引起一些不正确的输出,不正确的外部行为)
4. **Test Case Values**& **Expected Results**

Test Case Values:满足某个测试需求的值

Expected Results:对某个测试用例执行结果的期待值

1. **Observability** &**Controllability**

观察性:程序的行为被观察到的容易程度

控制性:向程序提供需要的输入的容易程度

影响观察性和控制性的**输入**:

Prefix Values & Postfix Values(分为两类: Verification Values, Exit Commands)

1. **White-box** & **Black-box** Testing

***from what level of abstraction to we derive tests***

黑盒测试: 功能测试,基于外部的软件描述,如规格说明,需求,设计

白盒测试: 结构测试,基于软件源码,包括分支,条件等

基于模型的测试: 基于软件模型,如UML图

1. **Top-Down** & **Bottom-Up** Testing

Top-Down:先测试主过程,然后测试被它调用的过程,从根到叶子

Bottom-Up:与上面相反,先测叶子,再往根进行

1. **Stress Testing**

路径测试: 输入局限于特定输入域的测试

1. Testing Levels Based on Test Process Maturity基于测试过程的成熟度的测试级别
2. Level 0: Testing is the same as debugging
3. Level 1: Purpose is to show correctness
4. Level 2: Purpose is to show failures
5. Level 3: The purpose of testing is not to prove anything specific, but to **reduce**

**the risk** of using the software

1. Level 4: Testing is a mental discipline that helps all IT professionals develop

higher quality software

1. 测试和软件开发模型

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）

极限编程的测试驱动开发（先写测试用例）

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）

1. 如何改善测试?
2. 需要更好的测试工具
3. 测试者需要吸取更有效的测试经验和方法
4. 测试者需要更多专业知识
5. 需要更多专门从事测试的人员

吸取更有效方法的障碍,对研究者以及教育学家的要求,以及对未来软件测试的展望(看STQ01)

1. 测试的基本技术(PPT02—PPT05)
2. Domain Testing也称为等价类测试
3. 划分必须满足两点要求:

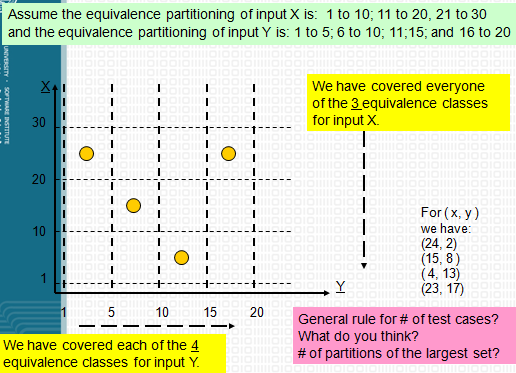
划分出来的各个块之间要无重叠

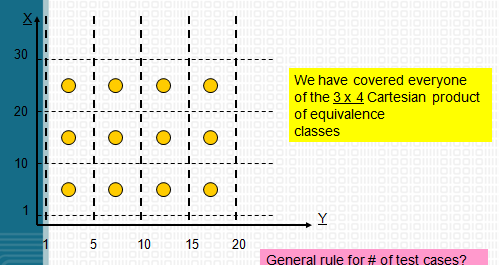
所有块和在一起要覆盖整个输入域

1. 等价类的三个性质:自反性,对称性,传递性
2. 弱等价类测试 & 强等价类测试

输入均为**有效输入**,区别于健壮性等价类测试

图中X,Y为两个输入,上面的为弱,下面的为强

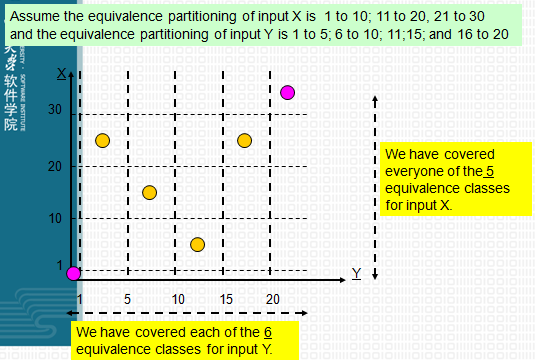


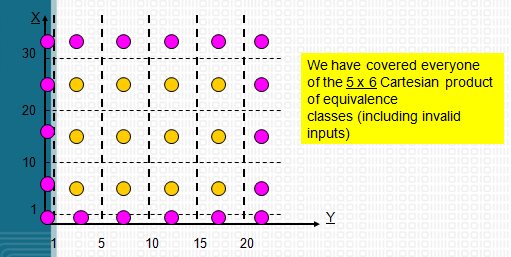


1. Weak Robust(弱健壮性?)等价类测试 & 强健壮性等价类测试

弱健壮性等价类测试与弱等价类测试类似,只是多了对**无效输入**的测试

上面的图为弱健壮性等价类测试,下图为强健壮性等价类测试





1. 输入域建模的流程
2. Identify testable functions
3. Find all the parameters
4. Model the input domain
5. Apply a test criterion to choose combinations of values
6. Refine combinations of blocks into test inputs
7. 输入域建模的两种方法
8. 面向接口(针对方法的每一个参数)
9. 面向功能(考虑程序的逻辑行为)

举例:一个输入为3个Int的方法,这三个参数代表三角形的三条边长

面向接口关注这3个int与0的关系来划分3个参数的等价类

面向功能关注这个方法的功能,按照三个参数能构成的三角形的类型来划分等价类

1. 对边界值的测试

一种选取等价类划分的扩展策略:

1. 在等价类有效值范围内任选一个
2. 在上下边界处分别选择刚刚好在边界上的值
3. 分别选择刚刚超出上下边界值的值
4. **On Point** & **Off Point**

On Point:在边界上(边界封闭)或在边界附近,满足条件的要求

Off Point:在边界附近,不满足条件的要求

举例:条件为X<=10

On Point 为10

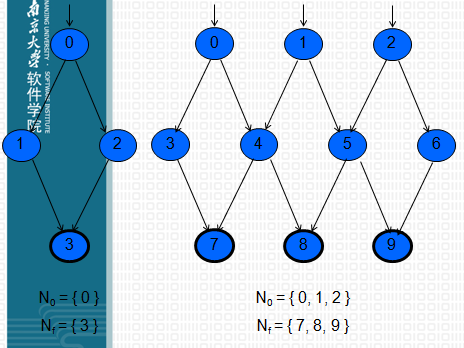
Off Point 为11

对选择这两者的策略不太理解,对PPT上的几张图不理解

1. 还有一些别的,都不太理解……
2. Graph Coverage
3. 图的定义

有起始结点,有终止结点,且两者都不能为空,而可以为多个,有边把点连起来(不知道是否可以允许有孤立的点?)

举例:

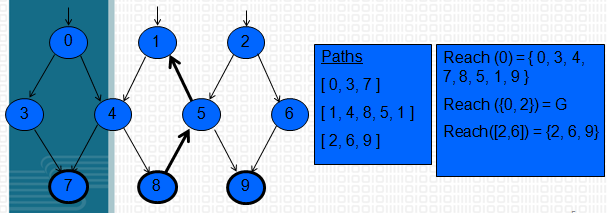


1. 几个概念

Path length(边的数目) Sunpath

Reach(n) 能从n到达的子图

举例:



1. Test Path & SESE

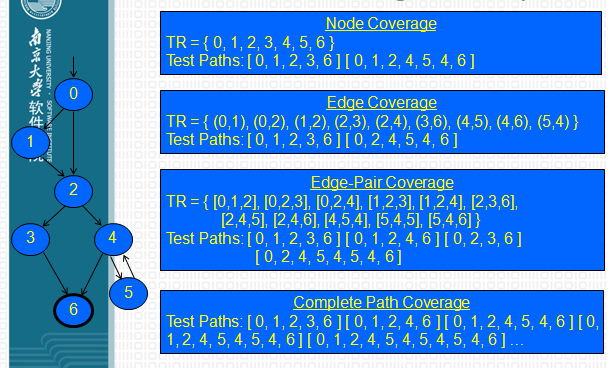
Test Path:开始于起始节点,结束与终止节点的路径,代表了测试用例的执行

SESE graph: 起始节点和结束节点均只有一个

1. Visiting & Touring
2. 结点覆盖 & 边覆盖 & 边对覆盖(EPC) & 完全路径覆盖(CPC) &

指定路径覆盖(SPC)

举例说明:



如何处理图里的循环? **基本路径**

1. Simple Paths(简单路径) & Prime Paths(基本路径)

Simple Path: 路径中的结点只出现一次(除了开始结点和结束节点相同的情况)

Prime Path: A simple path that does not appear as a proper subpath of any other simple path一条不是其他任何一条简单路径的子路径的简单路径

1. 基本路径覆盖 Prime Path Coverage (PPC)

即测试需求TR包含图中的每一条基本路径

1. Round-Trip Path

开始结点和结束节点相同的基本路径

1. Simple Round Trip Coverage (SRTC) & Complete Round Trip Coverage (CRTC)

不太理解

1. Source Code Coverage
2. control flow graph (CFG) & Data flow coverage(DFG)
3. CFG的绘制:

圈代表语句,箭头代表程序流向

几种程序结构的CFG绘制要留意:

If , if-Return , while and for Loops , do Loop, break and continue ,

case (switch)

注意手动绘制PPT上的例子!!!

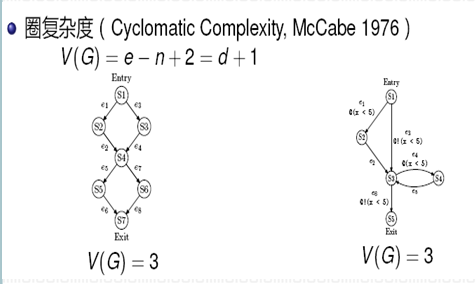
1. 基路径测试(**Basis Path Testing**)

基路径测试定义:

**基路径测试的步骤:**

1. 画CFG
2. 计算圈复杂度

计算方法:e为边的数量,n为结点数,d为圈的数量



若CFG中所有路径都可行,则存在V(G)条线性独立的完整路径(基本路径),且其他路径都可由这些路径线性表示

若CFG中存在不可行的路径,那么判断程序是否存在V(G)条基本路径是不可判定的

1. 从路径中选出一个测试集合

如何选定测试集合呢???

1. 构造测试用例
2. Data Flow Coverage for Source
3. Def & use
4. DU pair & DU path
5. All-defs coverage (ADC) & All-uses coverage (AUC) &

All-du-paths coverage (ADUPC)

1. 对实例的理解比较困难……
2. Logical Coverage

* DC ≥ SC
* CC not ≥ SC
* DC not ≥ CC， CC not ≥ DC

3种方法:

1. Condition & Decision
2. 对条件和判定概念的理解
3. 判定覆盖DC & 条件覆盖CC & 条件/判定覆盖CDC &

修改的条件/判定覆盖MC/DC

1. Clause & Predicate
2. A ***predicate*** is an expression that evaluates to a boolean value

A ***clause*** is a predicate with no logical operators,如与,或,异或等

1. Predicate Coverage (PC) & Clause Coverage (CC) &

Combinatorial Coverage (CoC)

1. CoC代价太高,一个改善方案是”making clauses active”

即让每个clause独立得影响predicate的结果

1. ACC: Active Clause Coverage &

General Active Clause Coverage (GACC)(Cj在Ci为真,为假时不一定完全一样)

& Restricted Active Clause Coverage (RACC) (Cj在Ci为真,为假时要完全一样)

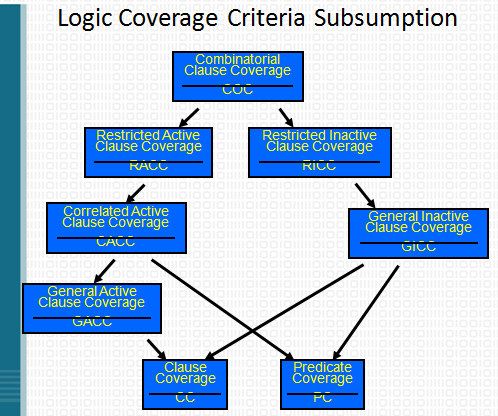
& Correlated Active Clause Coverage (CACC)(Cj的取值要使得 *p(ci = true) != p(ci = false)*满足)

1. ACC & ICC(Inactive Clause Coverage)

ACC确保major clause影响predicate的结果

ICC与其相反,确保major clause不影响predicate的结果

1. GICC & RICC
2. Logic Coverage Criteria Subsumption

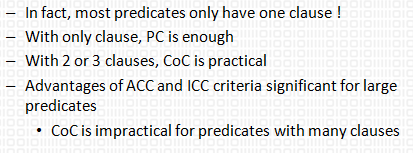


1. Making Clauses Determine a Predicate

*pc = pc=true* ⊕ *pc=false*

*pc* describes exactly the values needed for *c* to determine *p*

1. 小结



1. Fault-based Testing
2. 常见错误类型:

操作符错误:

Operator Reference Fault (*ORF)*

Expression Negation Fault (*ENF)*

Variable Negation Fault (*V NF)*

Associative Shift Fault (*ASF)*

操作数错误:

Missing Variable Fault (*MVF*)

Variable Reference Fault (*VRF)*

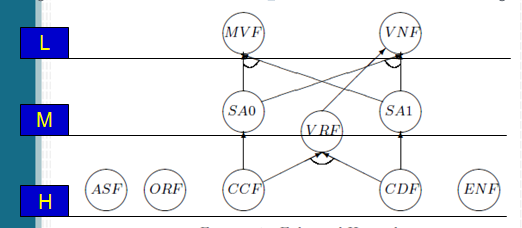
Clause Conjunction Fault (*CCF)*

Clause Disjunction Fault (*CDF)*

Stuck-At-0 Fault (*SA0*)

Stuck-At-1 Fault (*SA1*)

1. Fault Class Hierarchy



1. 专项测试
2. 配置测试
3. 性能测试的类型
4. Load testing 负载测试

用来观察系统在一定负载下的表现，这个负载一般在系统可承受范围内，比如测试一定数量用户同时在线时系统的表现

1. Stess testing 压力测试

用来了解系统能承受的负载的最大值，来测试系统的鲁棒性

1. Endurance testing (soak testing) 持久性能测试

给系统加以持久性的一定负载，观察系统的表现

1. Spike testing 峰值测试

突然加大负载，观察系统的表现

1. Configuration testing 配置测试

改变系统的配置组件，分别观察系统的表现

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

1. Isolation testing 隔离测试

重复做能发现系统某一个问题的一个测试，常用来隔离并确定这个错误域

1. 配置测试
2. 要了解的一些配置测试知识

**通常使用动态的白盒测试**

**不要假设硬件标准，协议等都是正确的**

**首先要分辨出那些硬件部件是真正影响待测软件的表现的**

**由于硬件众多，因此测试时要做等价类划分，然后选择最流行的硬件，当下主流的硬件，使用了软件中独特特征的硬件来测试**

1. 配置测试的目的是要识别并分离出配置bug：

区别常规Bug 和配置bug：在不同配置下进行相同的测试步骤，若该bug均出现，很有可能知识常规bug；若只在特定配置下出现，则很有可能是配置bug

BUG类型：

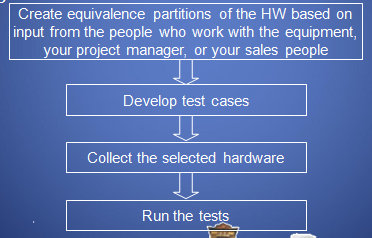
在众多配置条件下均出现的Bug

在某一特定配置条件下出现的Bug

硬件设备或它的驱动中有的只被你的软件揭露出的Bug

硬件设备或它的驱动中有的被很多软件揭露出的Bug

1. 配置测试的流程：
2. 决定需要的硬件类型（哪些硬件是跟待测软件相关的，不能盲目得测试所有计算机硬件）
3. 决定哪些硬件分支，模型，设备驱动是可用的（哪些是当下流行的等等）
4. 决定要使用硬件的哪些特征，模式，选项是可用的（硬件往往有很多使用模式及选项等）
5. 适当削减使得已决定好要用来测试的硬件是一个可控的集合（由于硬件数量还是过于庞大）
6. 找出待测软件与硬件配置有关的独特特征（比如独特的音效，画面等）
7. 设计测试用例
8. 执行测试用例
9. 反复执行测试直到结果符合要求（配置测试往往只能一次执行软件的一小部分，反复执行，并不断修复发现的小bug，直到不能发现bug，说明这部分测试通过了）



How can you tell if a bug you find is a general problem or a specific configuration problem

在不同配置环境下重新运行揭示该bug的步骤，如果在这些实验中问题没有再次出现，那么很可能是配置的问题。如果它们在不同的配置环境下出现了，那么很可能是general 问题。

* + Is it acceptable to release a software product that has configuration bugs?

可以这样做。你们不可能修复所有问题。因为在所有测试中，那些过程都是基于风险的。你和你的团队要决定什么事可以解决什么是不能解决的。对于一些只有在少量配置环境下才能显现出来的隐晦问题不处理它们也是一种明智的选择。

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。

1. 兼容性测试
2. 兼容性测试的定义

Checking that your software interacts with and shares information correctly with other software

检查待测软件是否能和其他软件进行正确的交互和信息分享

1. 兼容性测试开始之前需要明确的内容：

待测软件需要和哪些平台和软件兼容

判断是否兼容的标准是什么

待测软件要和其他平台，软件交互，分享哪些类型的数据

1. 平台和应用的版本

向后/向前兼容性

测试多个版本的影响（待测软件与多种软件的兼容性测试）

1. Standards and Guidelines

High-Level Standards and Guidelines：对软件的外观，使用感觉，支持的特征等的兼容性要求

Low-Level Standards and Guidelines：对类似文件格式，网络通信协议等兼容性的要求？

1. 数据分享兼容性

要求允许用户能很容易得将数据从一个平台转移到另一个平台，并保持数据的原有性质

如：文件的保存和加载；

文件的导出和导入；

剪切，复制，粘贴等

* + If you're assigned to test compatibility of your product's data file formats, how would you approach the task?

调查你的程序是否要遵循现有的文件标准。如果是的话，测试它是否能符合这些标准。等价划分可能读写你程序文件的程序，设计你程序可能存储或加载的文件样本。测试这些文件在你的程序和其他程序之间的交互能力。

1. 外国语言测试
2. 基本内容
3. Translation 翻译

仅仅针对语言上的翻译

1. Internationalization 国际化  **I18N**

设计一个软件产品使得其能适应于不同语言、地区而不需要工程化的改变的过程

一个国际化的软件产品具有如下特征：

1. 有了当地语言的支持，软件能在全世界运行
2. 对各种语言的支持并不是硬编码在程序里的，而是存储在软件源码之外并能动态获取到的
3. 在不需要重新编译的情况下就能支持新语言
4. 展现给用户的数据符合当地的文化，比如日期的格式
5. 能方便、快速得被本地化
6. Localization 本地化 **L10N**

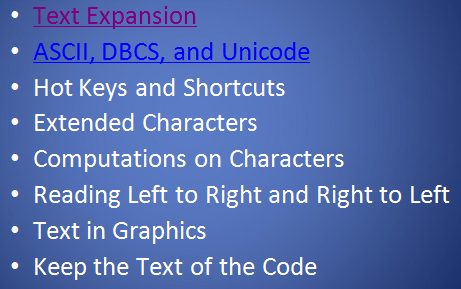
是将已经实现国际化的软件进行翻译等，使其能适应于不同的语言和文化。本地化需要实现界面、语音、图片的改变，文本的翻译

有哪些内容是本地化时需要考虑的？

1. menu菜单，dialog box对话框，Hint提示,alt test文字说明，error message错误信息，static text静态文字
2. 与当地时区对应的时间，如期，数字，度量标准等
3. 快捷键，热键，位图，图标等
4. 安装向导
5. 消息模板，在线帮助，示例等
6. 文档
7. 软件包的外观
8. Globalization 全球化 **G11N**

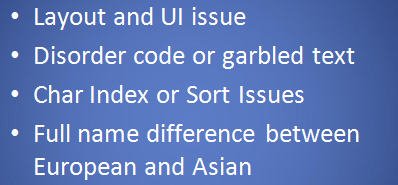
**包含I18N & L10N**

1. Translation issue



DBCS：设置好页面上文字允许输入的的最大长度，防止出现数据库访问错误

1. I18N issue



1. L10N issue

两点需要注意的：

1. **内容**的本地化
2. **数据格式**的本地化
3. Configuration and Compatibility Issue
4. Localized OS
5. East Asian System locale

编码不是单字节的，是Non-Unicode

1. European System locate

OEM & ANSI

1. 外国语言需要测试的内容

Internationalization Testing **I18N**

Localization Testing **L10N**

Linguistic/Translation Testing

测试所有语言特性是否被本地化了

Focuses on all the language elements of an application. Done (ideally) in running localized application. Uses test scripts ideally

Cosmetic/UI Testing

测试软件外观是否本地化

Focuses on all the visual elements of the UI. Done (ideally) in running localized application

Functionality Testing

测试本地化后软件功能和以前是不是一样正确

Focuses on whether the application still works after localization. A well internationalized product will likely not fail functionality testing. Not a standard task, but done at vendor request with test scripts.

More on Functionality Testing

Delivery Testing

（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。

* + Identify several areas where extended characters can cause

字母、单词的排序顺序，大小写的互换，和总的显示及打印问题

* + Name a few types of data formats that could vary from one localized program to another

度量衡，时间，汇率

1. 易用性测试（usability testing）看ppt吧
2. 什么是易用性？

容易发现，容易学习，容易使用，有效性

1. 怎样能有好的易用性？

易见 visibility 映射 mapping 反馈 feedback

1. UI测试

易用性测试的主要内容就是UI测试

怎样可以有好的UI？

1. 遵循相关标准、规范
2. 直观的 Intuitive
3. 正确的
4. 一致的
5. 灵活的

软件状态间能灵活跳跃，某一状态可以终止、被跳过

数据的输入输出有多种方式：直接输出，粘贴，加载文件，拖拽等

1. 令人感觉舒适的
2. 有用的
3. 简单的
4. Testing for the disabled: - Accessibility Testing

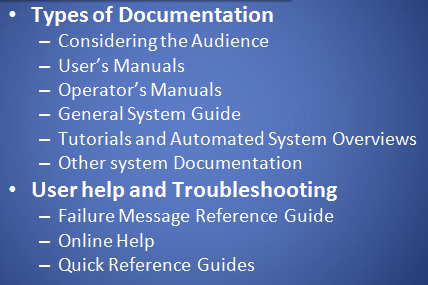
测试软件是否为为残疾人提供了相应服务

针对不同的残疾类别，比如：视觉障碍，听觉障碍，行动障碍（有人无法使用鼠标等），认知障碍

1. 文档测试
2. 文档种类

两大类读者：系统文档 & 用户文档

文档分类：





1. 文档测试的重要性

提高软件易用性 提高软件实用性 It improves support costs？？？

1. 文档审查时要注意的3点
2. What Makes Good documentation?一份好文档的特点

文档应使用简洁的语言

内容应涵盖软件的使用方法，错误提醒，常见问题，示例等

1. Documentation testing Criteria 文档测试标准

正确性，完整性，易懂性

1. A Documentation testing Checklist 文档测试检查表

包括常规检查点（内容，主题，术语，目标读者等），以及正确性的检查（示例，语法等）

1. 文档测试的现状
2. 文档开发受到的关注，以及供给的资金，资源比较少
3. 文档开发者通常都不是软件方面的专家
4. 印制文档需要时间？？？
   * It improves **usability**
   * It improves reliability
   * It improves support costs
5. 安全性测试
6. “黑客”的动机
7. 为了挑战，获得名望
8. 好奇心
9. 为了使用或利用对方电脑
10. 为了摧残：3-D：defacing损害外观

destruction 破坏

denial of service（DoS）拒绝服务

1. 窃取信息、数据等
2. 威胁建模
3. 组建威胁建模团队
4. 明确要保护的“资产”
5. 建立一个全局架构
6. 分解应用
7. 明确威胁，给它们排定优先级，并记录
8. 测试需要的5项原则（或者应翻译为应测试的5项基本内容……）
9. 认证：识别-有效（如登录）
10. 一致性：防止破坏、欺骗？？？
11. 隐私：防止窃听
12. 不能拒绝：负责任（不能拒绝服务吗？？？）
13. 可用性：RAID,云（这是神马？？？）

Authentication: Identity - Validity

Integrity: protection from tampering/spoofing

Privacy: protection from eavesdropping

Non-Repudiation: accountability

Availability: RAID,clusters,cold standbys

1. 保证安全性使用的一些方法

认证 LDAP 密码学知识（对称加密、非对称加密） 解密

1. 服务器安全（web服务器 app服务器 数据库服务器）

防止出现拒绝服务 防止缓冲溢出 日志文档要互相分离

Patches？？？

Compilers/Interpreters - don’t keep in cgi-bin ？？？

1. 客户端安全（浏览器，应用，组件）

可能出现安全隐患的地方，需要关注：

浏览器设置 宏指令 OLE 木马 BIOS里的软驱引导

1. Cookies
2. 对应与OSI模型的一些安全保护协议

会话层：SSL TLS PCT

应用层：S-HTTP

网络层：IPSec

1. 网络方面

防火墙

Router based (Packet filtering) at IP level

Proxy based (gateways)

1. DMZ

把局域网内的一台电脑映射到外网，很容易受到攻击，所以一定要装防火墙

1. VPN
2. 病毒 & 蠕虫

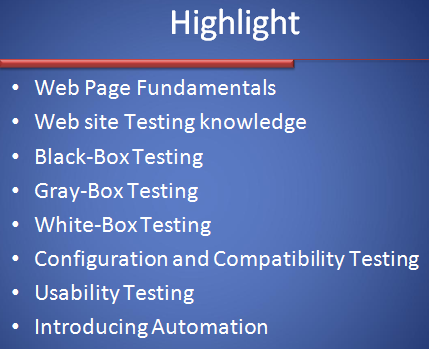
木马／蠕虫都属于病毒的一种。

有这样一个比喻：

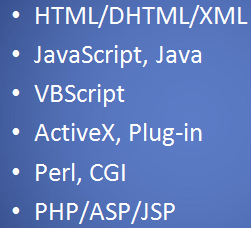
木马象隐形的战斗机，能长时间进行象手术刀一样的精确攻击。

蠕虫如轰炸机，重在即时性的攻击（机器感染蠕虫病毒，能在短时间内造成系统瘫痪）

1. 这部分实在扛不住了……
2. 网站测试



1. 网站测试要关注的东西、
2. 网页上的内容（文字，文字大小，字体，颜色，图像等）
3. 功能性（链接，按钮，侧边栏，逻辑等）
4. 易用性
5. 安全性
6. 性能
7. 网页中的技术



1. 黑盒测试

关注点：

1. 文字，文字说明
2. 超链接
3. 图像
4. 表单
5. 其他简单的功能性内容
6. 灰盒测试

黑盒和白盒的结合体，是测试中的一把利器，把整个软件当成一个黑盒，但关注到某个特殊点时，考虑细节，用白盒的思维方式，思考这个地方软件是怎么工作的

1. 白盒测试

前提条件：

对待测网站有一定了解

具备编程知识

关注点：

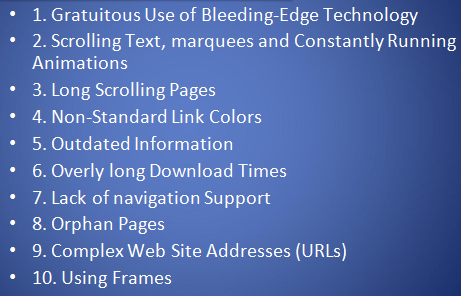
1. 动态内容
2. 数据库驱动下的网页
3. 程序创造的网页（应该指程序动态影响下的网页）
4. 服务器的性能和负载
5. 安全性
6. 配置测试和兼容性测试
7. 在不同的OS和浏览器下的表现
8. 确保使用的技术与多个平台有好的兼容性
9. 向前/向后兼容性
10. 网络测试

测试待测产品是可以在网络上运行的

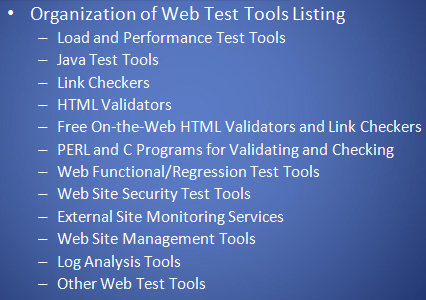
确保待测产品是能在多个网络OS（什么叫网络OS？？？）环境下运行的

需要一定的压力测试

1. 易用性测试
2. 网站设计中常见的10大错误



1. 自动化介绍



Rational Site Check：用于管理互联网和企业互联网网页站点的工具

自动化测试的过程（参考QTP）：GUI记录，回放

相关工具：Microsoft: Visual Test Suite

Rational: Visual Test and TeamTest

Mercury Interactive: XRunner, WinRunner

Compuware Corporation: QARun

WinRunner

SilkTest

Robot

1. 负载测试

也有相关工具，比如WebLoad 3.0.1 WAS (Web Application Stress)

1. 开源测试工具JMeter
2. Uptime Monitoring Tool

每15分钟向服务器请求一个页面，要是网站不可访问，会发邮件给你，然后继续请求，等网站可以访问时再发邮件通知你。

1. 自动化测试工具

