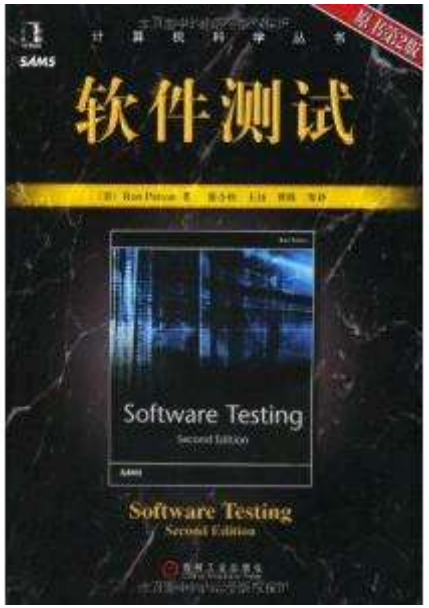
测试技术 – 常用测试

测试技术 – 常用测试

- 几种常见系统测试
- 用户文档测试
- 配置测试
- 本地化测试
- 易用性测试
- 基于应用服务器的测试
  - Client/Server 测试
  - 网站测试
  - 对Web进行压力测试
- α 测试和 β 测试
- 实时系统测试
- 其他测试
- 调试



ISBN：9787111185260

测试文档

- 用户文档测试

用户文档测试

是检查用户文档(如产品说明书、用户手册等)的正确性、清晰性和精确性，检验文档是否和实际应用存在着差别。

用户文档测试的内容

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；拼写、语法正确；不包含病毒。

从用户的角度看，软件文档和软件都是同样的产品

如，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，或出现显眼的拼写错误，都是属于与软件失效一样的软件缺陷。

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。

3

测试文档

- 用户文档测试的重要性

软件用户把这些独立的文档当做整个软件的一部分。好的文档以几种方式确保产品的整体质量：

（1）提高易用性

好的文档，增加了用户对产品的易用性。

（2）提高可靠性

可靠性是指软件稳定和坚固的程度。

如用户阅读文档，然后使用软件，使软件更加可靠。

（3）降低支持费用

用户有麻烦或遇到意外，就会请公司帮助，增加了公司的费用。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，降低支持费用。

测试文档

- 对文档高级审查

  测试文档第一步不是去找缺陷，而是在一个高度上审查。

  审查文档是为了找出根本性大问题，疏忽或遗漏之处。很好地了解产品以及影响其设计的外部因素。

  1.站在客户角度思考：

  设身处地的为客户着想，测试的时候把自己当成客户。

  2.研究现有的标准和规范

  把标准和规范视为文档的一部分。

  检验是否套用正确的标准，没有遗漏，是否与标准和规范相抵触。

  如：公司惯用语和约定、行业要求、政府标准、GUI，安全标准等。

测试文档

- 对文档高级审查（续）

  3.审查和测试同类软件文档

  同类软件有助于制订测试条件和测试方法，还可能暴露没想到的潜在问题。

  在审查同类软件时注意的问题包括：

  规模

  复杂性

  测试性

  质量和可靠性

  安全性

  这些，为仔细审查文档积累大量经验。

测试文档

- 文档的低层次测试技术

1、文档属性检查

传统教科书上，优秀文档应当具有的8个属性：

1）完整

是否有遗漏和丢失？完全吗？单独使用时是否包含所有内容？

2）准确

既定解决方案正确吗？目标定义明确吗？有没有错误？

3）精确、不含糊、清晰

描述是否一清二楚？是否有单独的解释？容易看懂和理解吗？

4）一致

产品功能是否自相矛盾，或与其它功能有无冲突？

测试文档

1、文档检查

5）贴切

描述功能的陈述是否必要？有没有多余信息？功能是否符合原来的客户要求？

6）合理

在规定的预算和进度下，以现有人力、工具和资源能否实现？

7）代码无关

产品说明书是否坚持定义产品，而不是定义其软件设计、架构和代码？

8）可测试性

功能是否测试？给测试员提供的建立验证操作的信息是否足够？

如文档不具备这些属性，那就发现了缺陷。

测试文档

2、文档术语检查清单

在审查文档时，还有一个“问题用语”检查清单。问题用语通常表明功能没有仔细考虑。可视为缺陷。

1）总是、每一种、所有、没有、从不

看到此类绝对肯定或否定的描述，需要确认是这样的。

软件测试员要考虑违反这些情况的用例。

2）当然、因此、明显、显然、必然

这些话意图说服你接受假定情况，不要上当。

3）某些、有时、常常、通常、经常、大多、几乎

这些话太过模糊，“有时”发生作用的功能无法测试。

4）良好、迅速、廉价、高效、小、稳定

这些是无法量化的术语，无法测试。必须进一步准确定义其含义。

测试产品说明书

2、文档术语检查清单（续）

5）等等、诸如此类、例如、依此类推

以这样的词结束的功能清单无法测试。

功能清单要绝对或者解释明确，以免让人对功能清单内容产生迷惑。

6）处理、进行、拒绝、跳过、排除

这些用语可能会隐藏大量需要说明的功能。

7）如果...那么...：

找出“如果...那么...”而缺少配套的“否则”结构的陈述。想想没有如果发生会怎样。

3、文档检查点清单

检查点	检查内容
总体	
读者	文档的读者水平，不要太易，更不应太难
术语	术语适合读者吗？前后一致吗？如用缩写，他们是标准的还是需要定义？所有缩写应被索引。
内容和题材	标题是否恰当，或遗漏？或不应有？材料是否合适？
正确性	
事实检查	所有的信息是准确的并采用正确的技术；查找作者由于过期产品说明书和销售人员夸大事实而导致的错误；检查表格内容，索引和参考文献；检查网站的网址、支持电话是否正确
操作步骤检查	仔细阅读文本，严格遵照指南。不要假设什么！防止缺失步骤。用户不知会有什么缺失。检查在文档中显示的那些结果是否正确。
图片和屏幕截图	检查图片的精准性。图像正确吗？屏幕截图是老版本的？图片标题正确？
示例	像用户那样加载并使用每个示例，如是代码，键入或复制并运行它。没什么比示例不正常还尴尬了！
拼写和语法	不出现错别字，不要出现有二义性的说法。特别要注意的是屏幕截图或绘制图形中的文字。有可能没有工具，只能人工完成。

测试文档

- 几个具体文档测试要领

1、产品说明书

对于产品说明书，依据GB/T25000.51，**检查原则**：

- 产品说明宜是充分可理解的、完整的并且易于浏览。
- 产品说明应避免内部不一致，每个术语在任何地方都应有相同的意义。
- 产品说明中包含的说明应是可测试的并且是可验证的。
- 产品说明应与源代码保持一致。

产品说明的**检查内容**

产品说明应包含潜在需方所需的必要信息，以便评价该软件对其需要的适用性。包含：1)标识和标示 2)功能性陈述 3)可靠性陈述 4)易用性陈述 5)效率陈述 6)可维护性陈述 7)可移植性陈述 8) 使用质量陈述

测试文档

2、用户文档集

对于用户文档集，依据GB/T25000.51，应当具有的6个属性：

- 完备性
- 正确性
- 一致性
- 易理解性
- 易学性
- 可操作性

测试文档

- 完备性（11条）
  - 用户文档集应包含使用该软件必需的信息；
  - 用户文档集应说明在产品说明中陈述的所有功能以及最终用户能调用的所有功能；
  - 用户文档集应说明可靠性特征及其操作。
  - 用户文档集应列出所处置的和引起应用系统失效或终止的差错和失效，特别是那些导致数据丢失的应用系统终止的结束条件。
  - 用户文档集应给出必要数据的备份和恢复指南。
  - 对于所有关键的软件功能（即失效后会对安全产生影响或会造成重大财产损失或社会损失的软件），用户文档集应提供完备的细则信息和参考信息。

测试文档

- 完备性（续）
  - 用户文档集应陈述在产品说明中给出的所有限制。
  - 用户文档集应陈述安装所要求的最小和最大磁盘空间。
  - 对用户要完成的应用管理职能，用户文档集应包括所有必要的信息。
  - 在用户所完成的应用管理职能的信息中，应包括让用户能验证是否成功完成应用管理职能的信息。
  - 如果用户文档集分若干部分提供，在该集合中至少有一处应标识出所有的部分。

测试文档

- 正确性
  - 用户文档集中的所有信息都应是正确的

    所有信息的正确性都宜追溯到权威来源。
  - 用户文档集不应有歧义的信息
- 一致性
  - 用户文档集中的各文档不应自相矛盾、互相矛盾以及与产品说明矛盾
- 易理解性
  - 用户文档集应采用该软件特定读者可理解的术语和文体，使其容易被 COTS 软件产品主要针对的最终用户群理解
  - 应通过经编排的文档清单为理解用户文档集提供便利

测试文档

- 易学性
  - 用户文档集应为用户学会如何使用该软件提供必要的信息
- 可操作性
  - 如果用户文档集不以印刷的形式提供，则文档集应指明是否可以被打印，如果可以打印，那么指出如何获得打印件 。
  - 卡片和快速参考指南以外的用户文档集，应给出目次（或主题词列表）和索引。
  - 对于不常用的术语和首字母缩略语，用户文档集应加以定义。

测试文档

3、需求说明书

对需求说明书，应该对功能的正确性、完整性和清晰性，以及其它需求给予评价。评审的主要内容是：

- 系统定义的目标是否与用户的要求一致；
- 系统需求分析阶段提供的文档资料是否齐全；
- 文档中的所有描述是否完整、清晰、准确反映用户要求；
- 与所有其它系统成分的重要接口是否都已经描述；
- 被开发项目的数据流与数据结构是否足够，确定；
- 所有图表是否清楚，在不补充说明时能否理解；
- 主要功能是否已包括在规定的软件范围之内，是否都已充分说明；
- 软件的行为和它必须处理的信息、必须完成的功能是否一致；
- 设计的约束条件或限制条件是否符合实际；
- 是否考虑了开发的技术风险；

（continue）

测试文档

需求说明书评审内容：(续)

- 是否考虑过软件需求的其它方案；
- 是否考虑过将来可能会提出的软件需求；
- 是否详细制定了检验标准，它们能否对系统定义是否成功进行确认；
- 有没有遗漏，重复或不一致的地方；
- 用户是否审查了初步的用户手册或原型；
- 软件开发计划中的估算是否受到了影响.

为保证软件需求定义的质量，评审应以专门指定的人员负责，并按规程严格进行。　　评审结束应有评审负责人的结论意见及签字。

除分析员之外，用户／需求者，开发部门的管理者，软件设计、实现、测试的人员都应当参加评审工作。一般，评审的结果都包括了一些修改意见，待修改完成后再经评审通过，才可进入设计阶段。

测试文档

- 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，理解有偏差。最重要的是，文档应指出软件中难以使用或难以理解处，对它们更好地解释。

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。

测试技术 – 常用测试

- 文档测试
- 配置测试
- 本地化测试
- 易用性测试
- 基于应用服务器的测试
  - Client/Server 测试
  - 网站测试
  - 对Web进行压力测试
- α 测试和 β 测试
- 实时系统测试
- 面向对象的软件测试
- 其他测试
- 调试

配置测试

- 配置测试（硬件兼容性测试）综述
  - 概念
  - 为什么要配置测试
  - 测试内容
  - 分离配置缺陷
  - 计算工作量
- 执行任务
- 其他
  - 获得硬件建议
  - 明确硬件标准

配置测试

- 配置测试（硬件兼容性测试）综述
  - 概念

    侧重于确保测试对象在不同的硬件和/或软件配置上按预期运行的测试。（RUP）

    检测被测软件是否能在多种硬件组合配置环境下正常运行。功能和性能是否都满足设计要求。

    常用配置：个人计算机；部件（CPU、主板等）；外设；接口（ISA、PCI、USB等）；可选项和内存；设备驱动程序等。
  - 为什么要配置测试

    因为不是所有生产硬件的商家都严格遵照标准来设计硬件。如，为了市场，可能会对硬件附加功能、提高性能或仓促推出，结果是软件使用某种硬件配置无法正常工作。

    另外，还有些没有标准，只有规范,所以需要测试。

配置测试

- 测试内容

进行配置测试通常习惯性地采用表格统计方式来进行数据统计，最后得出结论。

- 主机兼容性测试
  - 软件能否运行在不同的主机上
  - 软件的最低配置和推荐配置（CPU、内存、硬盘）

如，不同主机配置测试操作表

测试名称	Window7 配置测试	测试编号		备注
用例编号		测试人员	XXX	
准备环境	各个主机平台			
主机名称		操作步骤	预期结果	实际结果
苹果		长时间运行操作	兼容	
戴尔		长时间运行操作	兼容	
……		……	……	
测试结论			测试时间	

配置测试

- 测试内容
  - 不同组件（板卡、配件）的配置测试

    不同组件指的是同一主机平台上的不同硬件配置。

    如，不同组件的配置测试

测试名称	Window7 配置测试	测试编号		备注
用例编号		测试人员	XXX	
准备环境	各个主机平台			
CPU	主板	操作步骤	预期情况	实际情况
Intel	华硕	长时间运行操作	兼容	
Intel	七彩虹	长时间运行操作	兼容	
AMD	华硕	长时间运行操作	兼容	
AMD	技嘉	长时间运行操作	兼容	
……	……	……	……	……
结论		测试时间		

配置测试

- 测试内容
  - 外设的兼容性测试

    实际上就是对驱动程序的测试。
    - 检查外设是否提供驱动，如有，简单测试，确认即可；
    - 检查OS是否提供外设的驱动(性能较低)且自动安装，手动安装高性能驱动；
    - 如驱动是随应用程序提供，应仔细、严格测试。

    (续)

配置测试

- 测试内容
  - 外设的兼容性测试(续)

测试用的外设选择主流、通用的产品

一个外设上测试通过/未通过后，换一个不同品牌的相同外设继续测试。若测试未通过，先考虑硬件，检查硬件没问题再考虑软件

如，不同外设的配置测试：

测试名称	Office Word 2000	测试编号		备注
用例编号	MSOF2897	测试人员		
准备环境				
检查部件	操纵步骤	预期结果	实际结果	
鼠标	左、右键交互单击操作	正常		
键盘	输入操作	正常		
打印机	打印操作	正常		
……				
测试结论		测试时间		

配置测试

- 测试内容
  - 不同接口的配置测试

接口兼容性测试指操作系统在主机上运行时对各个接口的兼容性测试。如对USB端口的兼容性。一般的软件系统是建立在操作系统的平台上对端口进行使用的。

  - 可选项和内存的配置测试

可选项配置测试是增加或减少可选择硬件组件的兼容性测试。

测试名称	Window7	测试编号		备注
用例编号		测试人员	XXX	
准备环境				
测试部件	操作步骤	预期结果	实际结果	
内存条	用 512M 内存运行系统	正常		
内存条	用 1G 内存运行系统	正常		
内存条	用 256M 内存运行系统	正常		
……	……	……	……	
结论		时间		

配置测试

- 分离配置缺陷
  - 判断缺陷是配置问题还是普通缺陷（软件缺陷）的方法：

    在另一台配置完全不同的机器上执行相同的操作。如果缺陷没产生，那就很可能是配置问题了，如果缺陷在多种配置中产生，应该是普通的缺陷。

配置测试

- 分离配置缺陷（续）
  - 判断缺陷是开发程序的问题还是硬件的问题，要找出问题所在：

（1）软件在多种配置中都会出现缺陷。

（2）软件只在某一个特殊配置中出现缺陷。

（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到的缺陷-尽管它可能对测试的软件特别明显。

前两种情况，由开发小组负责修复缺陷。

后两种情况，责任不太清晰。

但是即使是硬件的问题，都是开发小组的责任，因为客户不管缺陷是怎么产生的，他们只要求在自己的系统配置中能正常运行。----软件开发的柔性。

配置测试

- 计算工作量

配置测试工作量可能非常大，测试员不可能把会出现的配置都测试。

如，Windows中有二十多类硬件，每类硬件有多个生产商，每个生产商有多个型号...等。它们的组合总计超亿，规模之大，不可想象！

再如：一种3D游戏，画面丰富，多种音效，允许多个用户联机对战，还可以打印游戏画面。

市场调查：336种显卡，210种声卡，1500种网卡，1200种打印机。

其测试组合=336×210×1500×1200

减少麻烦的答案是等价类划分：需要找出一个方法把巨大的配置可能性减少到尽可能控制的范围。

实际测试工作中，一般是根据软件的属性和质量要求来选择其中的某些来测试，并不一定要全部考虑到，由于没有完全测试，因此存在一定的风险。

配置测试

- 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类划分工作。什么重要，怎样才会成功，是决定的内容。计划配置测试时采用的一般过程如下：

1.确定所需的硬件类型

如，程序有打印功能，那么需测打印机；

程序要发声，需测声卡；…… 等。

2.确定哪些硬件，型号和驱动程序可用

一般选近几年主流的硬件型号和驱动（可询问销售、市场人员，或借助类似《PC杂志》）；然后进行等价类划分：这些设备大同小异？是贴牌生产？等等。

3.确定可能的硬件特性，模式和选项

每一种设备都有选项，软件可能不全支持。

如有的游戏，要求显示器颜色和分辨率，如果配置低于该要求，游戏不运行。

配置测试

- 执行任务（续）

4.将确定后的硬件配置缩减为可控制的范围

如，将需测配置放在一电子表格中，列出硬件类型、生产商、型号、驱动版本、流行程度、备注等。测试人员审查表格，确定要测哪些配置。

5.明确使用硬件配置的软件唯一特性

只需测试那些与硬件交互时有关的特性。

如测试Word打印之类，不必在每种配置中测试打开、保存特性，它们与打印无关。

6.设计在每一种配置中执行的测试用例

7.在每种配置中执行测试

8.反复测试直到小组对结果满意为止

配置测试

- 获得硬件

  即使把要配置的硬件可能性用等价类划分到最低限度，仍然需要N多的硬件，没那么多钱怎么办？

  （1）只买可以或者将会经常使用的配置。

  （2）与硬件生产商联系，看能否租借甚至白送。

  （3）问公司内部人有什么硬件，是否允许进行测试。

  （4）与专业的 配置和兼容性测试实验室 进行外协测试

- 明确硬件标准

  尽可能多的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。

  Microsoft公司发布了一套软件和硬件接受Windows徽标的标准

  http://www.microsoft.com/whdc/system/platform/default.mspx

  http://www.microsoft.com/whdc/winlogo/default.mspx

配置测试

- 对其他硬件进行配置测试

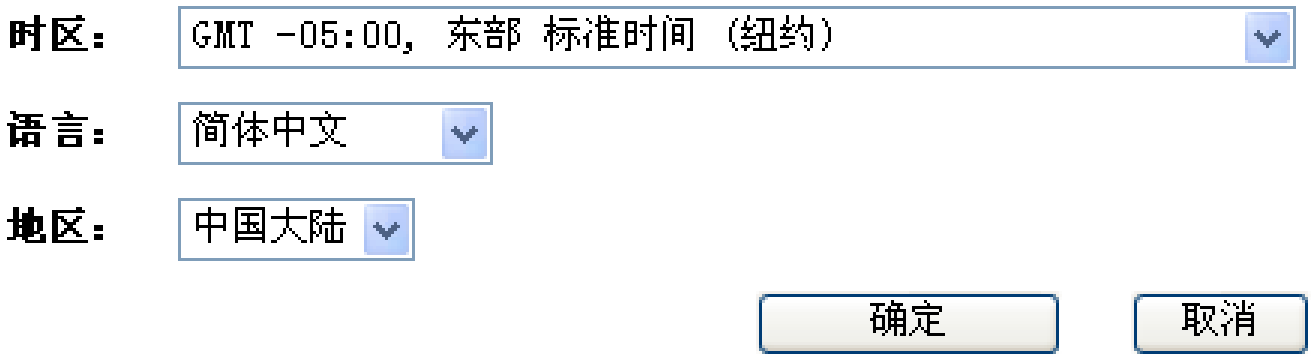
  根据从设备使用者，项目经理或者销售人员那里获得的信息建立硬件的等价区间，写测试用例，收集所选硬件，执行测试。

测试技术 – 常用测试

- 文档测试
- 配置测试
- 本地化测试
- 易用性测试
- 基于应用服务器的测试
  - Client/Server 测试
  - 网站测试
  - 对Web进行压力测试
- α 测试和 β 测试
- 实时系统测试
- 其他测试
- 调试

本地化测试

- 软件本地化概念
- 软件本地化测试
  - 本地化测试的主要方式
  - 软件本地化测试模型
  - 本地化测试的重点
  - 本地化软件缺陷类型
  - 测试量的大小



本地化测试

- 软件全球化、国际化、本地化概念

  软件全球化是指软件提供多国语言版本。包含国际化和本地化

  - 国际化（internationalization）

    是指在设计软件时，将软件与特定语言及地区脱钩的过程。当软件被移植到不同的语言地区时，软件本身不用做内部工程上的改变或修正。

    SW Internationalization常简写成I18N,中间的数字是前后两个字母间的字母个数。

  - 本地化（localization）

    是指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。

    L10N依据“localization” 使用与I18N同样的命名规则。

  G11N = I18N + L10N （globalization全球化）

I18N

- I18N设计要遵循的通用准则：
  - 在项目初期融入国际化思想，并使国际化贯穿于项目的整个生命周期。
  - 采用单一源文件进行多语言版本的本地化，不针对不同的语言编写多套代码。
  - 需要本地化的文字、数据与软件源代码分离，存储在单独的资源文件中。不使用基于源语言的数字常量、屏幕位置、文件和路径名。（消除Hard code）
  - 支持Unicode字符集、双字节的字符；
  - 使用Header files 去定义经常被调用的代码段；

（续）

I18N

- I18N设计要遵循的通用准则：（续）
  - 善翻译文本尺寸，具有调整的灵活性
  - 支持各个国家的键盘设置；
  - 软件能正确支持某区域的文字排序和大小写转换。
  - 支持各个国家的度量衡，时区，货币单位格式等的设置；
  - 国际化用户界面设计（自我定义）。
  - 如果软件中采用第三方开发的软件或组件，需要检查和确认是否满足国际化的要求。

L10N

- I18N设计要遵循的通用准则：
  - 翻译
  - 地区文化、宗教
  - 度量衡和时区等
  - 软件用户界面（UI）
  - 联机文档（帮助文档和功能性的PDF文档）
  - 热键设置

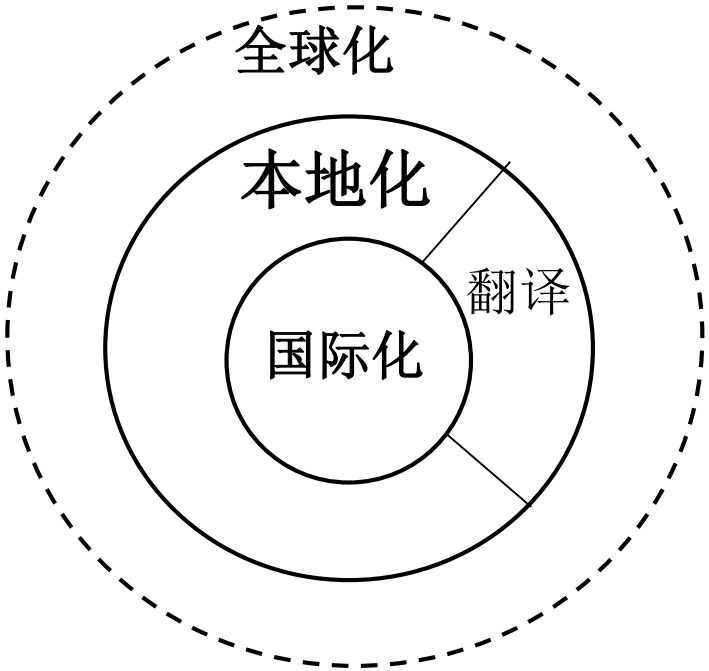
本地化测试

- 软件本地化工程

对一项产品来说，国际化只需做一次，但本地化则要针对不同的区域各做一次。两者互补，只有两者结合，才能让一个系统适用於各地。

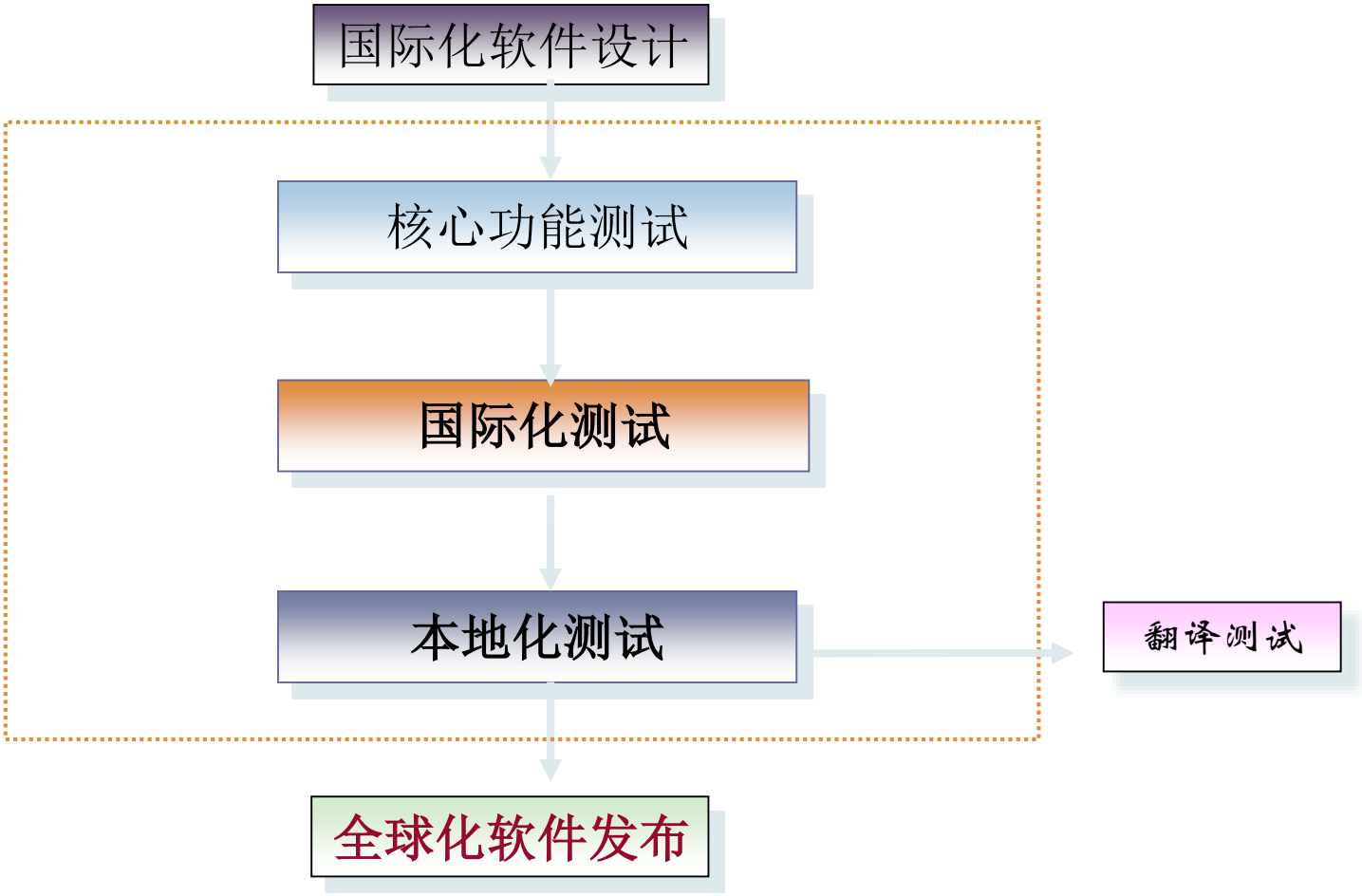
“国际化”是“本地化”的基础和前提，“本地化是“国际化”向特定语言环境的转换，是对“国际化”的补充和完善，它还包括为实现对某种特定语言良好的支持而进行的有针对性的翻译调整以及对软件进行的打补丁工作等。

它融合了软件工程、翻译技术和桌面出版等技术。

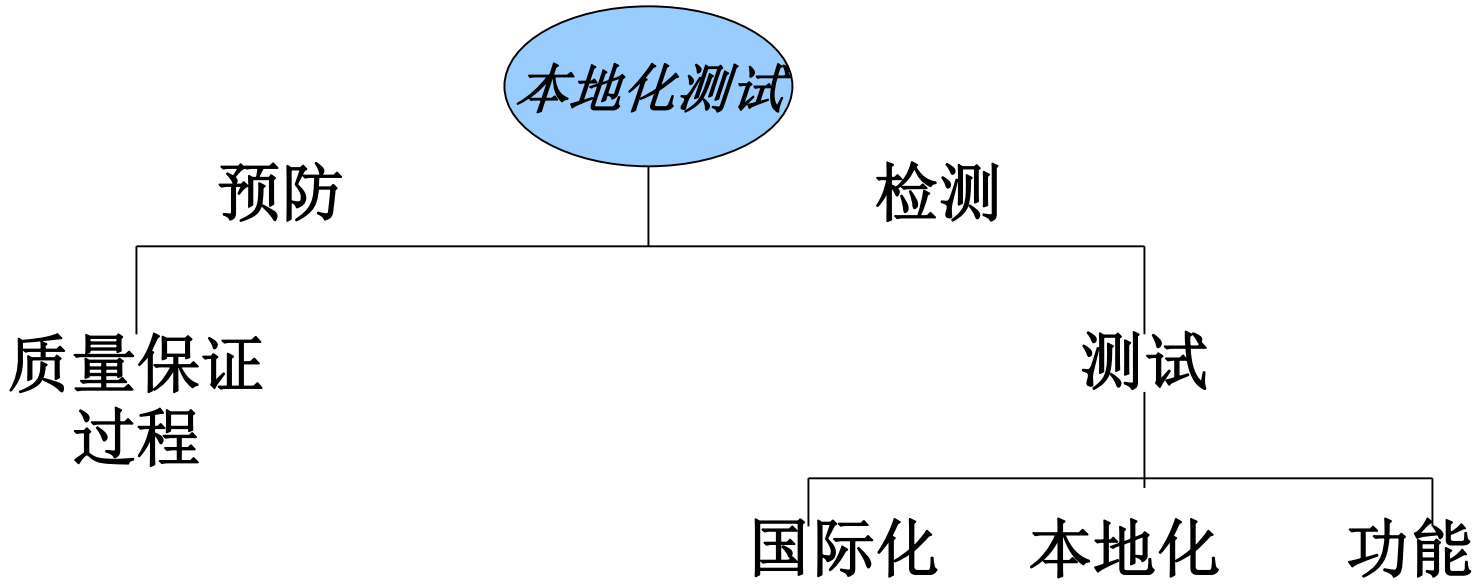


测试此类软件称为本地化测试。

本地化测试



本地化测试



本地化测试

- 本地化测试的主要方式
  - 1. 安装/卸载性能测试
    - 是否可以正确地安装/卸载在本地语言的操作系统上；
    - 安装/卸载前后安装文件、快捷方式、程序图标和注册表等的变化是否与源语言程序一致。
  - 2. 软件功能测试
    - 功能是否与源语言软件功能相同；
    - 是否支持当地语言的输入和输出，如对双字节支持和正确显示；
    - 对当地日期，时间，货币符号等的支持性能；
    - 是否支持当地语言的文件名和目录名。

本地化测试

- 本地化测试的主要方式
  - 3. 软件界面测试
    - 窗口中的按钮，菜单等的布局是否合理，美观；
    - 软件运行后的界面元素，包括菜单、块捷键、对话框、屏幕提示、按钮、列表框的布局和本地化字体和字号是否正确；
    - 界面文字的翻译是否与术语表一致，是否存在没有翻译的元素。
  - 4、帮助文件功能和翻译质量
    - 本地化帮助文件的功能是否与源语言软件一致；
    - 本地化帮助文件的布局是否合理，美观；
    - 本地化帮助文件的文字翻译是否准确、专业，是否存在没有翻译的段落。

本地化测试

- 软件本地化测试模型

1、本地化集成测试模型

指测试人员完成包含本地化功能测试、用户界面测试和语言质量的全部三项内容。

- 优点：测试效率高，质量稳定。由于测试人员完成全部三项测试内容，避免了与语言人员的交流过程，可以在较短的时间内完成测试任务，而且测试方法和基准相同，保证了测试质量的稳定性和一致性。
- 缺点：要求测试人员不仅掌握良好的测试技能，熟悉被测产品的功能特性，还要精通本地化语言和当地的文化习俗，是三种测试模型中对测试人员的要求最高的类型。招聘这样的测试人员可能比较困难，并且会提高测试的成本。
- 适用场合：较适用于被测试本地化语言是测试人员母语的情形，或者对测试效率和测试质量要求较高的本地化测试。

本地化测试

- 软件本地化测试模型

  2、本地化“一加一”测试模型

  是指安排一个（或一组）测试人员和一个（或一组）语言人员坐在一起，共同执行功能测试、用户界面测试和语言质量测试。

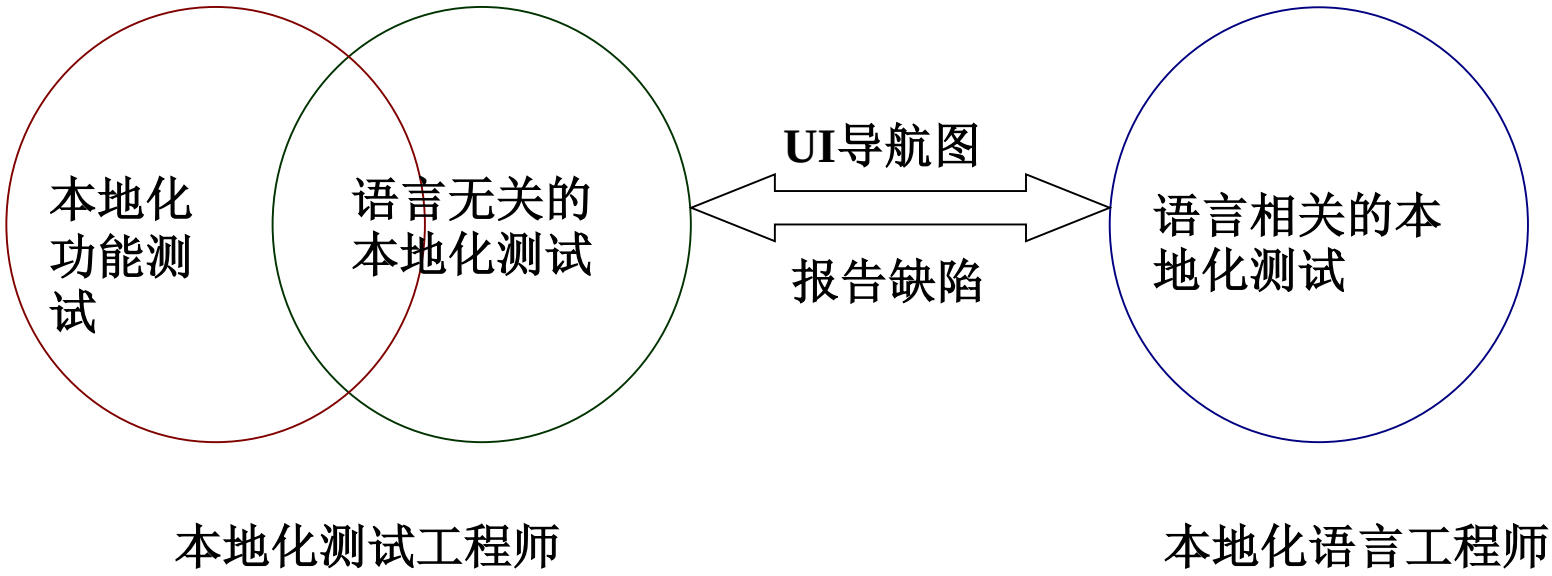
  - 优点：是可以发挥测试人员的技术优势，也可以充分利用语言人员的语言优势，二者可以相互补充和配合，保证测试的质量的稳定性和一致性。另外，测试的综合成本比集成测试要低一些。
  - 缺点：测试效率不高，由于需要相互交流，一个人在执行测试时，另一个只能等待，时间的利用率不高，造成测试花费较长时间。另外，这种方式测试的覆盖性可能不高，有些本地化内容也可能没有进行测试，还需要增添额外的随机测试进行补充。
  - 适用场合：在测试人员测试非母语本地化产品时采用，对于那些测试时间比较充裕，要求不太高的测试成本，能较容易找到本地化语言为母语的语言人员的情形。

本地化测试

- 软件本地化测试模型

  3、本地化分布测试模型

  本地化分布测试模型（见图）是指按照一定的时间顺序，将测试内容进行详细分工，安排不同技能的人员，分别执行本地化功能、用户界面和语言质量的测试。



本地化测试

例如，测试人员先执行本地化功能测试和用户界面测试，测试完成后再由语言人员测试本地化语言质量，反之亦然。测试人员和语言人员可以同时并行执行各自的测试内容，前提条件是语言人员必须熟悉测试的步骤。

- 优点：提高测试效率；测试成本较低；测试覆盖性高。
- 缺点：语言测试的准备过程复杂；报告语言缺陷的过程复杂；测试的进度可能不易控制；某些语言缺陷不好准确判断。
- 适用场合：在同时测试许多种本地化语言的场合适用。

如在中国大陆同时测试十多种欧洲语言的本地化测试的项目，而且测试的成本要求尽可能低，同时测试时间较短。

本地化测试

- 本地化测试的重点
  - 技术层面的测试

    调整大小、调整默认设置、重新编译、创建新的图形、重新编排文档格式；
  - 文化层面的更改

    包装、图标、宣传、样品、政治敏感的术语，地方规章和宗教信仰
  - 测试注意以下几个方面
    - 使文字和图片有意义
    - 翻译问题
    - 本地化的问题
    - 配置和兼容性问题

本地化测试

- 使文字和图片有意义(翻译的准确)

  简单说就是不要把“What are you doing”翻译成“什么是你在做”。不过这些不是一般测试工程师能做的，通常都是外包给一些翻译公司。

  除了语言，还需要考虑地域（region或locale）：用户的国家和地理位置。

- 翻译问题
  - 文本扩展(text expansion)。

    有可能英文版的软件里面只有2个单词，但是翻译到阿拉伯或者中文的时候就需要7~8个字符，那么就有可能出现这个所谓的文本扩展。

    再如，English: “Press Ctrl + Alt + Del to restart.”。 German: "Dr *ü* cken Sie Strg+Alt+Entf, um den Computer neu zu starten."

    - 显示问题
    - 内存溢出

本地化测试

- 翻译问题（续）
  - ASCII, DBCS and Unicode

    ASCII字符集只能表示256种不同的字符。

    对于中文、日文，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。

    不同国家有不同的语言符号，OS用代码页来说明适合于一种或多种国家/地区语言使用的一个字符集。与字符集对应，代码页有单字节代码页和双字节代码页。代码页和DBCS在许多情况下足够了，但是会遇到兼容性问题。代码页是否正确？如在英文OS中显示中文文本，可能没有相应的代码页或相互间的转换，导致乱码。

    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。

本地化测试

- 翻译问题（续）
  - 热键和快捷方式(Hot Keys and Shortcuts)

    法语和英语的“搜索”：法语是R开头，英语是S开头，可能会有一些问题。

    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。
  - 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。如重音字符 ，还有特殊的符号例如♡。

    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。

本地化测试

- 翻译问题（续）
  - 字符转换(Computations on Characters)

    在ASCII码里面，大写字母+32就等于小写字母了。不过对于其他的希腊文字语言，却不是这样子的，如果程序里面只是+32来转换的话，多半会出现问题。
  - 从左到右和从右到左的阅读

    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。

    这个问题幸亏操作系统已做好，测试一下就行了。
  - 图形中的文字(Text in Graphics)

    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。

    如，最常见的3个图标，加粗(B)、斜体(I)和下划线(U)。它是英文相关单词的开头，对中文、日文毫无意义。

本地化测试

- 翻译问题（续）
  - 文字和代码分离

    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资源文件中。

    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出现在代码中。

本地化测试

- 本地化问题

  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。

  - 内容适应

    这里的内容是指产品中除了代码之外的所有东西。确保其适应。

    如中国人对龙是有崇拜的，那么一个中国本地化的软件最好不要在龙上面做什么手脚，毕竟文化有差异。

    再如first floor 在美国和英国楼层不同。

    本地化问题要仔细审查各类内容：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。

本地化测试

- 本地化问题（续）
  - 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数据单位格式。

    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。

本地化问题 - 数据格式

Region	Date			Time		Number			Currency
	Short mm = month, dd = day, yyyy = year	Example: "Feb. 5, 2001"	Long www= weekday, dd = day, yyyy = year	am/pm or 24 hr.	Example	Group	Dec-imal: . or ,	Example	Units
U.S.	m/d/yy	2/5/01	WWW, MMM d, yyyy	am/pm	2:30 pm	,	.	1,234.56	US Dollars
Canada	dd/mm/yy	05/02/01	WWW, MMM d, yyyy	am/pm	2:30 pm	,	.	1,234.56	Canadian Dollars
French Canada	dd/mm/yy	05/02/01	www d mmm yyyy	24 hr	14:30	space	,	1 234,56	Canadian Dollars
China	yy-m-d	01-2-5	yyyy年M月d日	24 hr	14:30	,	.	1,234.56	
France	dd/mm/yy	05/02/01	www d mmm yyyy	24 hr	14:30	space	,	1 234,56	French Francs
Germany	dd.mm.yy	05.02.01	WWW, d. mmm yyyy	24 hr	14:30	.	,	1.234.56	Deutschmarks
Hong Kong	dd/mm/yy	05/02/01	WWW, d MMM yyyy	am/pm	2:30 pm	,	.	1,234.56	HK Dollars
Japan	yyyy/mm/dd	2001/05/02	yyyy年M月d日	24 hr	14:30	,	.	1.234.56	Yen
Korea	yyyy/mm/dd	2001/05/02	yyyy 년 M 월 d 일 dddd	24 hr	14:30	,	.	1,234.56	Won
Spain	dd/mm/yy	05/02/01	www, d mmm yyyy	24 hr	14:30	.	,	1.234,56	Peseta
Sweden	yyyy-mm-dd	2001-02-05	www d mmm yyyy	24 hr	14:30	space	,	1 234.56	Krona
Taiwan	yyyy/m/d	2001/5/2	yyyy年M月d日	am/pm	2:30 pm	,	.	1,234.56	
Great Britain	dd/mm/yy	05/02/01	WWW, d MMM yyyy	24 hr	14:30	,	.	1,234.56	British Pounds

本地化测试

- 配置和兼容性问题
  - 数据兼容性

    比如文档在英文环境中编辑，在德文环境中能否访问？
  - 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。

    如，XP提供了106种语言，66种不同的键盘布局。键盘提供的字符和布局支持这些语言吗？每一种键盘都有自己语言的专用键？

    据说，在日文操作系统里没有“\”反斜杠的，他们都是用日圆符号“ ”来代替。

    软件可能用到的任何外设，如打印机的字符等，都要在平台配置和兼容性测试的等价划分中去考虑。

本地化测试

USA and China

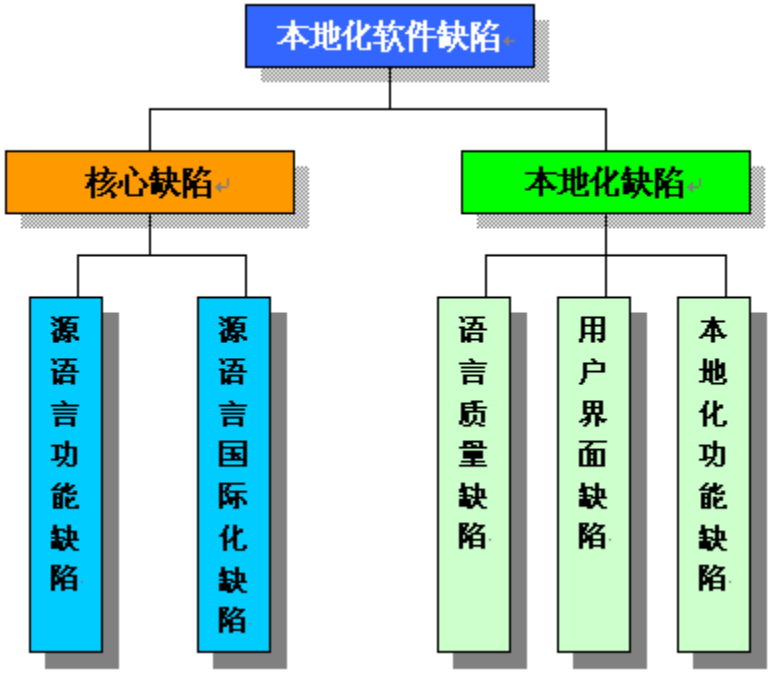
German

Japan

Vietnam（越南）

本地化测试

本地化软件缺陷类型



本地化测试

缺陷表现特征

1 用户界面缺陷

控件的文字被截断（Truncation）

对话框中的文本框、按钮、列表框、状态栏中的本地化文字只显示一部分

控件或文字没有对齐（Misaligned）

对话框中的同类控件或本地化文字没有对齐

控件位置重叠（Overlapped）

对话框中的控件彼此重叠

多余的文字（Extra strings）

软件程序的窗口或对话框中的出现多余的文字

丢失的文字（Missed strings）

软件程序的窗口或对话框中的文字部分或全部丢失

本地化测试

缺陷表现特征

1 用户界面缺陷（续）

不一致的控件布局（Inconsistent layout）

本地化软件的控件布局与源语言软件不一致

文字的字体、字号错误（Incorrect font name and font size）

控件的文字显示不美观，不符合本地化语言的正确字体和字号

多余的空格（Extra space）

本地化文字字符之间存在多余的空格

本地化测试

缺陷表现特征

2 语言质量缺陷

字符没有本地化（Unlocalized strings）

对话框或软件程序窗口中的应该本地化的文字没有本地化

字符不完整地本地化（Incomplete localized strings）

对话框或软件程序窗口中的应该本地化的文字只有一部分本地化

错误的本地化字符（Error localization）

源语言文字被错误地本地化，或者对政治敏感的文字错误地进行了本地化

不一致的本地化字符（Inconsistent localized string）

相同的文字前后翻译不一致

相同的文字各语言之间不一致

相同的文字软件用户界面与联机帮助文件不一致

本地化测试

缺陷表现特征

2 语言质量缺陷（续）

过度本地化（Over localization）

不应该本地化的字符进行了本地化

标点符号、版权、商标符号错误（Incorrect punctuation, Copyright）

标点符号、版权和商标的本地化不符合本地化语言的使用习惯

3 本地化功能缺陷

本地化功能缺陷是本地化软件中的某些功能不起作用，或者功能错误，与源语言功能不一致。

功能不起作用（Not working）

菜单、对话框的按钮、超链接不起作用

本地化测试

缺陷表现特征

3 本地化功能缺陷（续）

功能错误（Error function）

- 菜单、对话框的按钮、超链接引起程序崩溃
- 菜单、对话框的按钮、超链接带来与源语言软件不一致的错误结果
- 超链接没有链接到本地化的网站或页面
- 软件的功能不符合本地化用户的使用要求

热键和快捷键错误（Error hot keys and short-cut keys）

- 菜单或对话框中存在重复的热键
- 本地化软件中缺少热键或快捷键
- 不一致的热键或快捷键
- 快捷键或快捷键无效

本地化测试

缺陷表现特征

4 源语言功能缺陷

源语言功能缺陷是在源语言软件和全部本地化软件上都可以复现的错误。

功能不起作用（Not working）

- 菜单不起作用
- 对话框的按钮不起作用
- 超链接不起作用
- 控件焦点跳转顺序（Tab键）不正确

文字内容错误（Incorrect strings）

- 软件的名称或者版本编号错误
- 英文拼写错误、语法错误
- 英文用词不恰当等

本地化测试

缺陷表现特征

5 源语言国际化缺陷

源语言国际化缺陷是在源语言软件设计过程中对软件的本地化能力的处理不足引起的，它只出现在本地化的软件中。

区域设置错误（Error regional setting）

本地化日期格式错误

本地化时间格式错误

本地化数字格式（小数点、千位分隔符）错误

本地化货币单位或格式错误

本地化度量单位错误

本地化纸张大小错误

本地化电话号码和邮政编码错误

本地化测试

缺陷表现特征

5 源语言国际化缺陷（续）

双字节字符错误（Error DBCS）

不支持双字节字符的输入

双字节字符显示乱码

不能保存含有双字节字符内容的文件

不能打印双字节字符

本地化测试

- 测试量的大小

这个问题需要用风险的分析来决定。如某语种版本的市场很大，那么就不要马虎。

测试量大小，具体地说，考虑二个问题：

1. 项目从一开始就计划本地化了吗？

   如果软件从一开始就考虑到了本地化，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。

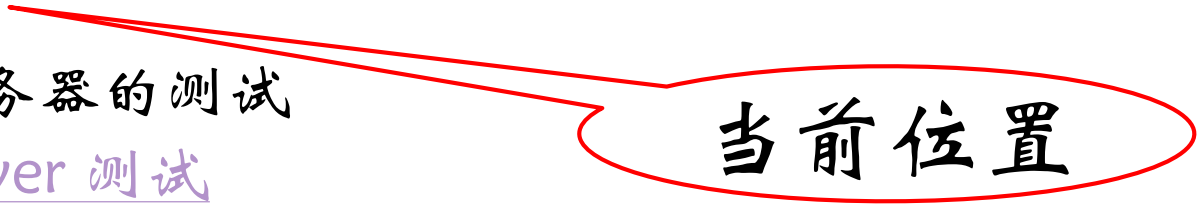
2. 本地化版本中更改程序代码了吗？

   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。

测试技术 – 常用测试

- 文档测试
- 配置测试
- 本地化测试
- 易用性测试
- 基于应用服务器的测试
  - Client/Server 测试
  - 网站测试
  - 对Web进行压力测试
- α 测试和 β 测试
- 实时系统测试
- 其他测试
- 调试



易用性测试

- 术语
- 优秀用户界面的构成
- 为有残疾障碍的人员测试：辅助选项测试
- GB/T 15532 中的易用性测试
- ISO 25051 中的易用性
- GUI测试

易用性测试

- 术语
  - 易用性

    是交互的适应性、功能性和有效性的集中体现。

    GB/T16260-2003(ISO 9126-2001)《软件工程 产品质量》质量模型中，提出易用性包含：易理解性、易学习性和易操作性；即易用性是指在指定条件下使用时，软件产品被理解、学习、使用和吸引用户的能力。
  - 易用性测试

    软件易用性方面的测试。不但包括程序测试，同时还包括对用户手册系统文档的测试。
  - 用户界面

    用于与软件交互的方式称为用户界面或UI。

易用性测试

- 优秀UI的构成

  优秀UI具备的七个要素：

  1. 符合标准和规范
  2. 直观
  3. 一致
  4. 灵活
  5. 舒适
  6. 正确
  7. 实用

  这些没有具体量化的指标，主观性较强。

- 下面具体描述：

易用性测试

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。

如，软件要在Mac或者Windows等平台上运行，应符合平台的标准和规范—它们描述了软件应该具有什么样的外观和感觉，何时使用复选框，单选按钮，何时使用提示信息、警告信息或是出错信息等。

“标准和规范”— 由软件易用性专家开发。经过大量实践而设计出的方便用户的规则。

注：如果测试在某平台上运行的软件，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进行测试。

也并非要完全遵守准则，有时开发方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。

易用性测试

1）符合标准和规范（续）

Microsoft 标准：在《Microsoft Windows User Experience》（Microsoft Press）书中描述。

这是一本为运行在Microsoft Windows家族的操作系统上的应用程序创建设计良好的、外观和功能一致的用户界面而编写的正式指导。

本书包括一些用户界面设计的基本原则和设计方法，并且详细说明了如何将诸如对象和属性等这样的以数据为中心的概念应用到界面设计中。本书还包括一些鼠标、键盘及其他输入设备交互操作方面的详细信息，以及如何使用由操作系统提供的通用界面元素的详细信息，它还包括有关支持国际用户和残疾用户的信息。

易用性测试

2）直观

- 用户界面是否洁净、不唐突、不拥挤？

UI不应该为用户使用造成障碍。所需功能或者期待的响应应该明显、并出现在预期出现的地方。

- UI的组织和布局合理吗？

是否允许用户轻松地从一个功能转到另一个功能？下一步做什么明显吗？任何时刻都可以决定放弃、退回、退出吗？输入得到确认了吗？菜单或者窗口是否太深了？

- 有多余功能吗？

是否有太多特性把工作复杂化了？是否感到信息太庞杂？

- 帮助系统有效吗？

如果其他所有努力失败，帮助系统真能帮忙吗？

易用性测试

3）一致

用户希望对一个程序的操作方式能够带到另一个程序中。

如软件或者平台有一个标准，就要遵守。如没有，就要注意软件的特性，确保相似的操作以相似的方式进行。

测试时，注意以下几方面：

- 快捷键和菜单选项。

  如，F1总是用于得到帮助信息。

- 术语和命名。

  软件使用同样的术语和命名方式吗？如Find是否一直叫Find，而不是叫Search

- 诸如OK和Cancel按钮的位置。

易用性测试

4）灵活

用户喜欢选择 — 不要太多，但应允许他们选择想要做的和怎样做。测试时，注意以下几方面：

- 多种视图的选择。
- 状态跳转和终止。 如网站的flash封面。
- 多种形式的数据输入和输出。

  如，Microsoft Excel可用键盘输入、粘贴、或从其他文件导入等，允许以14种标准和20种自定义图形查看数据 — 组合数量很大。

易用性测试

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。判断参考：

- 恰当。

  软件外观和感觉应该与所作的工作和使用者相符。不要太夸张也不要太朴素。

- 错误处理。

  程序应该在执行关键操作之前提出警告，并且允许用户恢复由于错误操作而丢失的数据。如Undo、Redo。

- 性能。

  有些操作不要太快，如提示信息。Windows的状态条是一个好例子。

易用性测试

6）正确

要测试正确性，就是测试UI是否做了该做的事。

特别注意处：

- UI与产品说明书不符。
- 市场定位偏差：是否与宣传材料不符？
- 语言和拼写。
- 不良媒体。UI中包含的所有图标、图像、声音和视频
- WYSIWYG（所见即所得）。保证UI显示的就是实际得到的。

易用性测试

7）实用

不是指软件本身是否实用，而仅仅指的是具体特性是否实用。

大型软件的开发或周期较长经过几次反复的软件开发中容易产生一些没有实用性的功能。

审查UI是否对软件具有实际的价值。它们有助于用户执行软件设计的功能吗?如果认为它们没必要，就要研究一下找出他们存在于软件的原因。多余的特性，不论是在什么软件中（重要与否）对用户都是不利的，同时还意味着需要更多的测试工作。

易用性测试

- 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

易用性测试

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。

GB/T 15532 中的易用性测试

根据 GB/T 16260.1，易用性包含易理解性、易学性、易操作性、吸引性以及易用性的依从性。

1 易理解性方面

从易理解性方面考虑，可测试：

a) 系统的各项功能，确认它们是否容易被识别和被理解；

b) 要求具有演示能力的功能，确认演示是否容易被访问、演示是否充分和有效；

c) 界面的输入和输出，确认输入和输出的格式和含义是否容易被理解。

2 易学性方面

从易学性方面考虑，可测试系统的在线帮助，确认在线帮助是否容易定位，是否有效；还可对照用户手册或操作手册执行系统，测试用户文档的有效性。

GB/T 15532 中的易用性测试

3 易操作性方面

从易操作性方面考虑，可测试：

a) 输入数据，确认系统是否对输入数据进行有效性检查；

b) 要求具有中断执行的功能，确认它们能否在动作完成之前被取消；

c) 要求具有还原能力（数据库的事务回滚能力）的功能，确认它们能否在动作完成之后被撤消；

d)包含参数设置的功能，确认参数是否易于选择、是否有缺省值；

e) 要求具有解释的消息，确认它们是否明确；

f) 要求具有界面提示能力的界面元素，确认它们是否有效；

g) 要求具有容错能力的功能和操作，确认系统能否提示出错的风险、能否容易纠正错误的输入、能否从差错中恢复；

（continue）

GB/T 15532 中的易用性测试

3 易操作性方面

h) 要求具有定制能力的功能和操作，确认定制能力的有效性；

i) 要求具有运行状态监控能力的功能，确认它们的有效性。

注：以正确操作、误操作模式、非常规操作模式和快速操作为框架设计测试用例，误操作模式有错误的数据类型作参数、错误的输入数据序列、错误的操作序列等。如有用户手册或操作手册，可对照手册逐条进行测试。

4 吸引性方面

从吸引性方面考虑，可测试系统的人机交互界面能否定制。

ISO 25051 中产品说明书的易用性陈述

- ISO/IEC 25051中产品说明书的易用性陈述
  - 产品说明应根据 ISO/IEC 9126.1 包含有关易用性陈述，要考虑易理解性、易学性、易操作性、吸引性以及易用性的依从性，并以书面形式写出能够证实可验证的依从性证据。
    - 易理解性：与用户为认识逻辑概念及其应用范围所花的努力有关的软件属性
    - 易学性：与用户为学习软件应用所花的努力有关的软件属性
    - 易操作性：与用户为操作和运行控制所花努力有关的软件属性
  - 产品说明应规定用户接口的类型。

    例如，这些接口可以是：命令行；菜单；视窗；web浏览器；功能键；帮助功能。

ISO 25051 中产品说明书的易用性陈述

- 中产品说明书的易用性陈述
  - 产品说明应规定使用和操作该软件所要求的专门知识

    例如，这些专门知识可以是：所使用的数据库调用和协议的知识；技术领域的知识；操作系统的知识；经专门培训可获得的知识；产品说明中已写明的语言之外的其他语言的知识。
  - 当该软件能由用户作适应性修改时，则应标识用于修改的工具或规程及其使用条件。

    例如，使用的条件可以是：参数的变更；计算算法的变更；接口定制；功能键指派。
  - 当预防版权侵犯的技术保护妨碍易用性时，则应陈述这种保护。

    例如，这些防护可以是：程序设置的使用截止日期；拷贝付费的交互式提醒。
  - 产品说明应包括可访问性的规定标示，特别是对有残疾的用户和存在语言差异的用户

ISO 25051 中软件的易用性质量要求

- ISO 25051 中软件的易用性质量要求（9条）
  1. 有关软件执行的各种问题、消息和结果都应是易理解的。

     如，借助以下的手段可以达到易理解性：恰当地选择术语；图形表示；提供背景信息；由帮助功能解释。
  2. 软件出错消息应指明如何改正差错或要报告差错向谁联系
  3. 软件应以最终用户易于理解的形式提供信息，即以可见易读的文本或图形输出，或以易听的音频输出。
  4. 出自软件的消息应设计成使最终用户易于理解的形式。

     例如，这些消息可以是：确认；软件发出的查询；警告；出错消息。
  5. 屏幕输入格式、报表和其他输出对用户来说应是清晰且易理解的。

ISO 25051 中软件的易用性质量要求

- ISO 25051 中软件的易用性质量要求（9条）
  5. 对具有严重后果的功能的执行应是可逆的，或者软件应给出这种后果的明显警告，并且在这种命令执行前要求确认。
  6. 借助用户接口、帮助功能或用户文档集提供的手段，最终用户应能够学习如何使用某一功能。
  7. 当遇有执行某一功能其响应时间超出通常预期限度会引起冲突时，最终用户应被告知。
  8. 每一元素（数据媒体、文件等）均应带有产品标识，如果有两种以上的元素，则应附上标识号或标识文字。

易用性测试 GUI 测试

- GUI 测试

  图形用户界面（GUI）对软件测试提出了有趣的挑战，因为 GUI 开发环境有可复用的构件，开发用户界面更加省时而且更加精确。同时，GUI 的复杂性也增加了，从而加大了设计和执行测试用例的难度。因为现在 GUI 设计和实现有了越来越多的类似，所以也就产生了一系列的测试标准。下列问题可以作为常见 GUI 测试的指南：

93

易用性测试 GUI 测试

- 窗口：
  - 窗口是否基于相关的输入和菜单命令适当地打开？
  - 窗口能否改变大小、移动和滚动？
  - 窗口中的数据内容能否用鼠标、功能键、方向键和键盘访问？
  - 当被覆盖并重新调用后，窗口能否正确再生？
  - 需要时能否使用所有窗口相关的功能？
  - 所有窗口相关的功能是可操作的吗？
  - 是否有相关的下拉式菜单、工具条、滚动条、对话框、按钮、图标和其他控制可为窗口使用，并适当地显示？
  - 显示多个窗口时，窗口的名称是否被适当地表示？
  - 活动窗口是否被适当地加亮？
  - 如果使用多任务，是否所有的窗口被实时更新？
  - 多次或不正确按鼠标是否会导致无法预料的副作用？
  - 窗口的声音和颜色提示和窗口的操作顺序是否符合需求？
  - 窗口是否正确地被关闭？

94

易用性测试 GUI 测试

- 下拉式菜单和鼠标操作：
  - 菜单条是否显示在合适的语境中？
  - 应用程序的菜单条是否显示系统相关的特性（如时钟显示）？
  - 下拉式操作能正确工作吗？
  - 菜单、调色板和工具条是否工作正确？
  - 是否适当地列出了所有的菜单功能和下拉式子功能？
  - 是否可以通过鼠标访问所有的菜单功能？
  - 文本字体、大小和格式是否正确？
  - 是否能够用其他的文本命令激活每个菜单功能？
  - 菜单功能是否随当前的窗口操作加亮或变灰？
  - 菜单功能是否正确执行？
  - 菜单功能的名字是否具有自解释性？
  - 菜单项是否有帮助，是否语境相关？

95

易用性测试 GUI 测试

- 下拉式菜单和鼠标操作：（续）
  - 在整个交互式语境中，是否可以识别鼠标操作？
  - 如果要求多次点击鼠标，是否能够在语境中正确识别？
  - 光标、处理指示器和识别指针是否随操作恰当地改变？
- 数据项：
  - 字母数字数据项是否能够正确回显，并输入到系统中？
  - 图形模式的数据项（如滚动条）是否正常工作？
  - 是否能够识别非法数据？
  - 数据输入消息是否可理解？

96

测试技术 – 常用测试

- 文档测试
- 配置测试
- 外国语言测试
- 易用性测试
- 基于应用服务器的测试
  - Client/Server 测试
  - 网站测试
  - 对Web进行压力测试
- 实时系统测试
- α 测试和 β 测试
- 其他测试
- 调试

应用服务器的类型和特征

- 1.1 应用服务器分类
  - Web 服务器
  - 数据库服务器
  - 实时通讯服务器
  - 邮件服务器
  - 群件服务器
  - 文件/打印服务器
  - 构件的服务器

应用服务器的类型和特征

- C/S模式优点及缺陷

  服务器端，一般采用高性能的工作站或专业服务器，并根据需要采用大型的数据库系统，而客户端则需要安装专用的客户端软件。

  - 优点

    应用服务器运行数据负荷较轻。数据的储存管理功能较为透明。

    由于客户端与服务器的直连，没有中间环节，因此响应速度快；

    操纵界面漂亮、形式多样，可以充分满足客户自身的个性化要求；

    C/S结构的治理信息系统具有较强的事务处理能力，能实现复杂的业务流程。

  - 缺点

    需要专门的客户端安装程序，分布功能弱，针对点多面广且不具备网络条件的用户群体，不能够实现快速部署安装和配置。

    兼容性差，对于不同的开发工具，具有较大的局限性。若采用不同工具，需要重新改写程序。

    开发本钱较高，需要具有一定专业水准的技术职员才能完成。

应用服务器的类型和特征

- B/S模式优点及缺陷
  - 优点

    具有分布性特点，可以随时随地进行查询、浏览等业务处理。

    业务扩展简单方便，通过增加网页即可增加服务器功能。

    维护简单方便，只需要改变网页，即可实现所有用户的同步更新。

    发布简单，共享性强。
  - 缺点

    个性化特点明显降低，无法实现具有个性化的功能要求。

    操纵是以鼠标为最基本的操纵方式，无法满足快速操纵的要求。

    页面动态刷新，响应速度明显降低。

    无法实现分页显示，给数据库访问造成较大的压力。

    功能弱化，难以实现传统模式下的特殊功能要求。

应用服务器的类型和特征

- 三层（多层）结构及应用：表示层,业务层,应用层各个层次的定义和分工
  - 第一层是表示层，主要由类似于图形用户界面的部分组成；中间层为业务层，由应用逻辑和业务逻辑构成；而第三层为数据层，包括了应用程序中所需要的数据。

- 多层结构的划分如下：

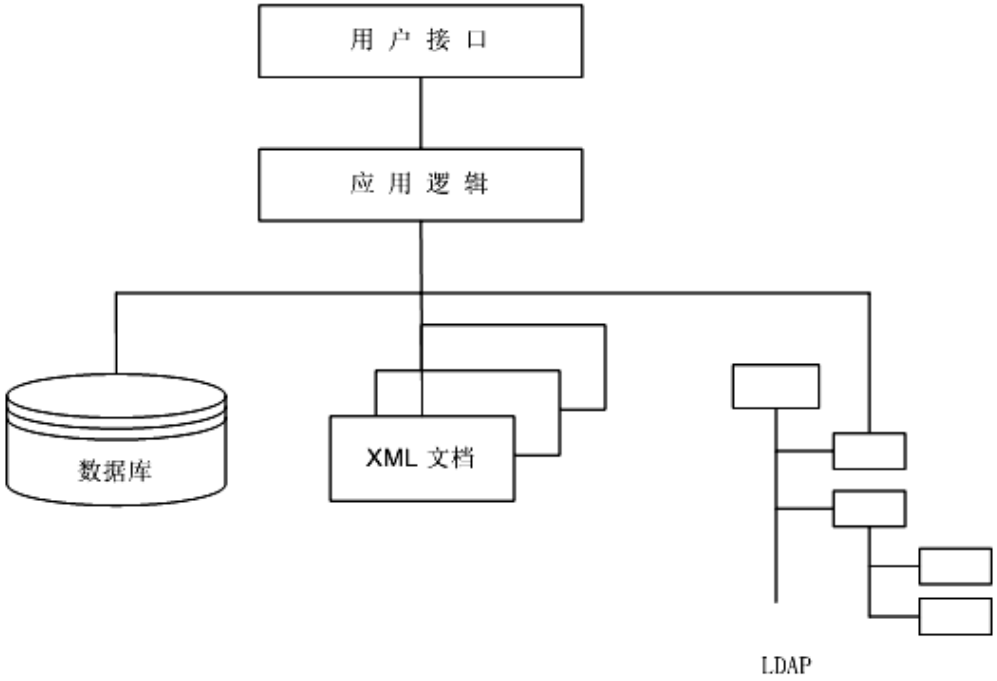
  (1)用户接口层：

  (2)表示逻辑层

  (3)业务逻辑层

  (4)基础框架服务层

  (5)数据层



应用服务器的类型和特征

- 不同类型的应用服务器不同的测试重点和角度,如:
  - 在邮件服务器中,容量,吞吐能力,防病毒攻击等是测试重点.
  - 在实时通信系统中,测试需要考察服务器功能的实现,数据的及时性,完整性,系统稳定性,系统数据备份的冗余设计,防网络攻击及故障恢复处理系统的设计

  - Client/Server 测试
  - 数据库服务器测试
  - 网站测试
  - 对Web进行压力测试

Client/Server 测试

- Client/Server测试目标
  - 检查系统是否达到公布的功能说明
  - 检查是否满足性能要求
  - 检查是否能否处理要求的负载
  - 检查在要求的各种软硬件平台上是否有错
- Client/Server测试的内容

通常，客户/服务器软件的测试发生在三个不同的层次：

1. 个体的客户端应用以“分离的”模式被测试——不考虑服务器和底层网络的运行；
2. 客户端软件和关联的服务器端应用被一起测试，但网络运行不被明显的考虑；
3. 完整的C/S体系结构，包括网络运行和性能，被测试。

因此，常用下面方法测试

103

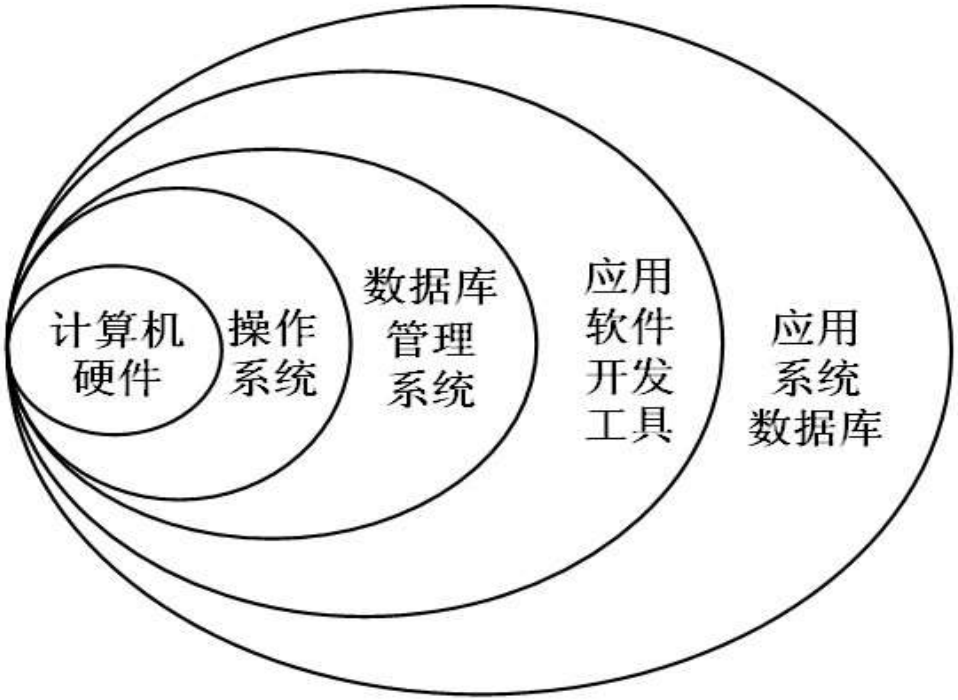
Client/Server 测试

- C/S 应用中常用到的测试方法：
  - 应用功能测试——客户端应用被独立地执行，以揭示在其运行中的错误。
  - 服务器测试——测试服务器的协调和数据管理功能，也考虑服务器性能（整体反映时间和数据吞吐量）。
  - 数据库测试——测试服务器存储的数据的精确性和完整性，检查客户端应用提交的事务，以保证数据被正确地存储、更新和检索。
  - 事务测试——创建一系列的测试以保证每类事务被按照需求处理。测试着重于处理的正确性，也关注性能问题。
  - 网络通信测试——这些测试验证网络节点间的通信正常地发生，并且消息传递、事务和相关的网络交通无错的发生。

104

基于数据库服务器的测试

- 数据库服务器的组成
- DBMS
- 常用的数据库服务器平台



数据库服务器测试

- 数据库服务器性能测试

  1. 大数据量测试
  2. 大容量数据测试

  这方面的性能测试至关重要，不合理的表结构以及程序中不合理的代码将使数据库的性能降低，甚至崩溃，因此，通过性能测试优化数据库。

- 性能测试过程和策略
  - 理解测试需求:
    - 是性能规划测试还是基准测试?
    - 是单个用户大数据量测试还是多个用户并发测试?
    - 什么地方是可能的性能瓶颈?
    - 可能是哪个方面的性能测试? (应用程序or系统参数or Schema对象)

数据库服务器测试

- 性能测试过程和策略(续)
  - 选择测试工具
    - 页面级的辅助测试工具: JMeter, Webload etc
    - 第三方监控程序: 如DB Flash, 或者有些通过log文件, 来监控动态的访问路径, 高占用的进程,会话和SQL语句
    - 自定义或者常用的命令, 来动态监控和获取执行SQL, PL-SQL所需要的时间, 占用的CPU和内存资源.
    - 数据库系统本身的性能工具包, 如在Oracle中使用的Explain Plan, AutoTrace, PKPROF以及Statspack.
  - 设计测试场景以及测试脚本

    数据量设计或加载； 测试环境建立和测试过程分析； 测试脚本设计
  - 收集数据和分析结果
    - 性能瓶颈参数
    - 优化和对比

数据库服务器测试

- 数据库并发控制测试

  数据库的并发控制能力是指在处理多个用户在同一时间内对相同数据同时进行访问的能力。

  - 并发主要考虑的几个方面：
    - 数据丢失
    - 不可重复数据
    - 读脏数据
    - 数据库的锁
  - 并发测试的设计过程
    - 并发流程分析
    - 并发控制测试设计

网站测试

- **网站的特性**
  - 网站的概念
  - 网页基础
- **2、网页测试**
  - 各种测试

网站测试

- 网站(website) (全国科学技术名词审定委员会)

  因特网上一块固定的面向全世界发布消息的地方，由域名(也就是网站地址)和网站空间构成，通常包括主页和其他具有超链接文件的页面。

- 特点

  1.网络集约性 (驻留在网络上，服务于客户群)

  2.内容驱动性 (网站内容由服务群体的要求所定)

  3.持续演化性 (更新采取持续演化的模式)

  4.即时性 (开发时间往往只有几周或几天)

  5.安全性 (必须有一定的安全性保障)

  6.美观性 (美观性和技术在同样程度上影响该应用的成功)

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链接的文字和图片在网页间浏览，搜索，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。

网站测试

- 网页基础（续）

  使网站更加复杂的特性如下：

  - 自定义的布局，允许用户更改信息出现在屏幕上的位置；
  - 自定义的内容，允许用户选择想看的新闻和信息；
  - 取决于屏幕分辨率的动态布局和可选信息；
  - 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；
  - 大量加强网页易用性的隐藏格式、标记和嵌入信息。
  - 数据库驱动的网页，许多网页是数据库驱动的。HTML提供网页布局，而图片、文字等则从服务器数据库中来。
  - 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。

网站测试

- 网站测试

  1）检查和验证是否按照设计的要求运行

  2）评价 系统在不同用户的浏览器端的显示是否合适。

  3）从最终用户的角度进行安全性和可用性测试，

  4）从功能、性能、可用性、客户端兼容性、安全性等方面进行评价。

网站测试

- 总体架构设计的测试
  1. 客户端
  2. Web架构
  3. 服务器的配置和分布
- 客户端设计的测试
  1. 功能设置测试
  2. 信息组织结构设计的测试
  3. 页面设计测试
- 服务器端设计的测试
  1. 容量规划的测试
  2. 安全系统的测试
  3. 数据库设计的测试

网站测试

- 对网站测试所采用的测试方法与策略

  黑盒测试、白盒测试、静态测试和动态测试都有可能用到，还会包括面向对象测试技术的运用。

- 黑盒测试（主要针对Web元素功能测试）

  把网页或者整个网站当做是一个黑盒子。不管它如何工作。

  - 建立状态表

  在测试网站时，首先应建立状态表，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网站更好地进行审视。

  - 测试

  绝大多数网页都是由文字、图片、链接以及少量表单组成，接下来，测试这几个方面。

网站测试

1）文本

- 网页文本应该当做文档进行测试

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。

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。

把鼠标光标移动到网页中的图片上时，看看弹出对图片语义信息的说明是否正确。

通过大幅缩放浏览器窗口来检查文字布局问题。

- 不要依赖拼写检查工具来做。

因为拼写检查不检查包含在图片、滚动块、表单等中的文字。

网站测试

2）超级链接

链接可以和文字或图片拴在一起。超链接一定要明显，文字链接一般有下划线。

每一个链接都能正常跳转。

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（网站的一部分，但是不能通过超级链接访问，只有知道正确的URL地址才能访问），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。

网站测试

3）图片（包括动画、边框、颜色、字体、背景、按钮等）

图片有明确的用途：不要胡乱地堆在一起，浪费传输时间。

验证所有页面字体的风格是否一致。

颜色的搭配：背景色与字体色和前景色相搭配。

图片的大小和质量：一般采用JPG 或GIF 压缩。

所有的图片是否被正确载入和显示？

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？

网站测试

4）表单

表单是指网页中用于输入和选择信息的表格，包括文本框、列表框和其它域等。表单有一些标准操作，如确认、保存、提交等。

表格、域的大小是否正确？

验证服务器是否接受、正确保存数据，拒绝错误数据？

可选域是否真正可选；提交操作的完整性等。

网站测试

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，并分别进行测试。

Cookies测试：Cookies是否起作用；是否按预定的时间进行保存；刷新对Cookies有何影响等。

网站测试

- 灰盒测试

  灰盒测试是白盒测试和黑盒测试的结合。灰盒测试仍然把软件当做黑盒来测，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。

网站测试

- 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，VB.Net,C#和XML等。

  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。

网站测试

- 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）连接速率速率。

网站测试

- 易用性测试

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。

(下页续)

网站测试

- 易用性测试(续)

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。

(continue)

网站测试

- 易用性测试(续)

  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。

网站测试

- 安全性测试

  目录测试：目录下应该有 index.html 或 main.html 页面，这样就不会显示该目录下的所有内容。

  套接字测试：确定加密是否正确；是否有相应的替代页面(适用于不支持SSL的浏览器，如IE3.0以下版)。

  登录验证：测试有效和无效的用户名和密码；

  超时：测试Web应用系统是否有超时的限制；

  日志文件：测试相关信息是否写进了日志文件、是否可追踪；

  脚本语言：是否有不安全代码；在没有经过授权时，是否能拒绝在服务器端放置和编辑脚本；

网站测试

- 内容测试

  目的是用来检验Web网站提供信息的正确性、准确性和相关性。

- 数据库测试

  数据库在Web网站中的作用：

  - 数据库为Web网站的管理、运行、查询和实现用户对数据存储的请求等提供空间。
  - 在Web应用中，最常用的数据库类型是关系型数据库，可以使用SQL对信息进行处理。

  两种主要数据库错误：

  - 数据一致性错误：主要是由于用户提交的表单信息不正确而造成。
  - 输出错误：主要是由于网络传输速度或程序设计问题等引起的。

  数据库测试就要针对这两种情况，分别进行测试。

网站测试

- 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。

  如，服务器性能及负载（压力）测试：通过应用模拟的方法实现，即通过某种程序方法（工具软件）模拟上万个链接和下载来判断服务器的响应时间、并发访问数量等性能与负载能力。

对Web进行压力测试

- Web服务的特点：

  Web服务处于分布式计算的核心位置，它们之间的交互通常很难测。

  分布式开发可能使Web服务的开发变得越来越容易隐藏错误。

  压力测试是检测这类代码错误的一种有效方法。

- Web压力测试

  压力测试是系统测试的一部分，要被设计为通过应用很大的工作负载来使软件超负荷运转，目的是要弄清被测的Web服务是不是不仅能做预期应能做的事，而且在被施加了某些高强度压力的情况下仍能继续正常运行。

  通过对软件保持高强度的压力测试（不超过性能统计数字确定的限制），经常能够发现许多其它测试无法发现的隐蔽错误。

- 压力下的错误类型
  - 内存泄露——通常要求操作重复非常多的次数后才会出现
  - 并发与同步

131

对Web进行压力测试（续）

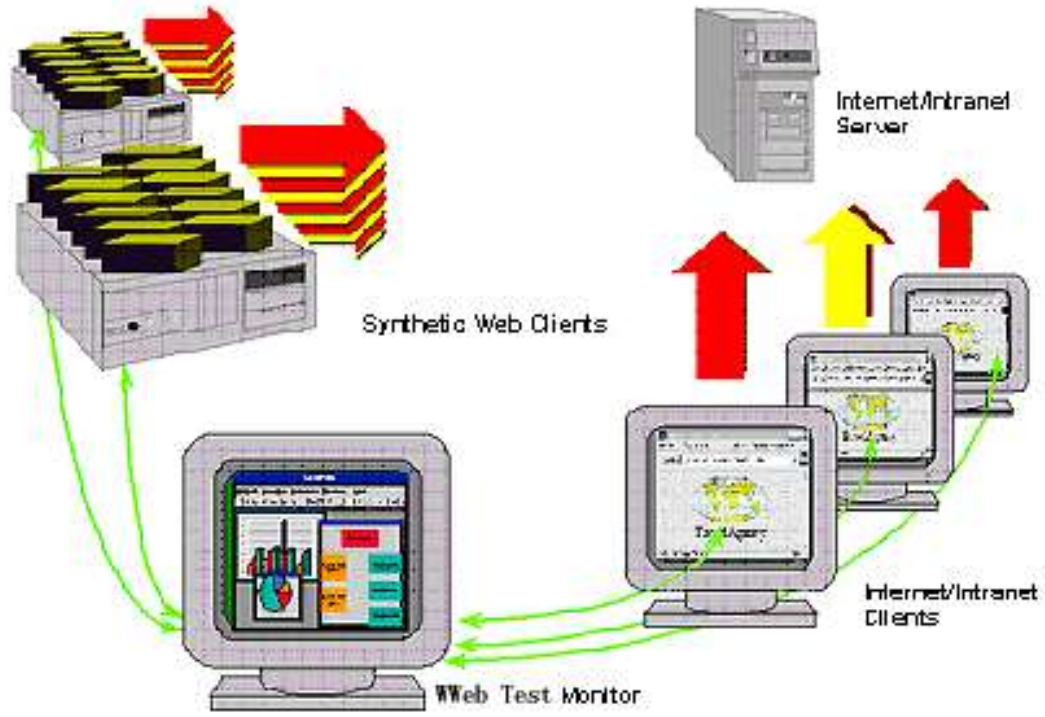


图 Web服务测试原理

132

对Web进行压力测试（续）

- 有效的压力测试的关键条件
  - 重复：就是一遍又一遍地执行某个操作或功能。这将确定一个操作能否正常执行，并且能否继续在每次执行时都表现正常。
  - 并发：就是在同一时间内执行多个操作。由并发引起的错误只能通过执行多个代码示例才能测出来，测试时要同时遍历多条代码路径。
  - 量级：要考虑到每个操作中的负载量，操作自身应尽可能给被测软件系统增加压力，即：尽量使单独的操作进行高强度的使用，增加操作的量级。
  - 随机变化：随机使用前面条件中的无数变化形式，就能够在每次测试运行时应用许多不同的代码路径。

133

使用WAS进行Web负载测试

- 负载测试是任何Web 应用开发周期中一个重要的环节。
  - 在构造一个为大量用户服务的应用之前，搞清楚产品配置能够承受多大的负载非常重要。
  - 但在实际开发过程中，若要按照实际投入运行的情况，组织成千上万的用户来进行压力测试，无论从那个方面看，都不现实。
- WAS
  - 为了有效的对Web应用程序进行负载（压力）测试，微软公司发布了简单易用、功能强大的Web应用负载测试工具WAS（Web Application Stress Tool），它能够提供一种简单的方法模拟大量用户访问目标网站，而且还能够提供Web应用程序工作时对硬件和软件的使用情况。
  - ACT (Microsoft Application Center Test)
- 使用WAS进行Web负载测试？(实验)

134

Web站点经验点滴

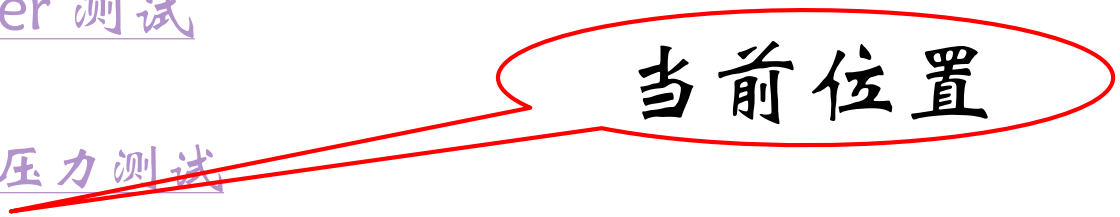
- 在执行客户端并发性能测试的过程中，需要同时监控数据库服务器、Web服务器以及网络资源等使用情况，以便对系统的性能做全面评估
- 录制脚本和手工编写脚本相结合
- 设置数据池，实现变量加载
- 业务批量执行
- 模拟用户数的递增
- 合理设置交易之间时间间隔
- 模拟IP地址变量的技术
- 超时（timeout）的设置
- 并发用户连续执行交易数的设置
- 错误处理

Web站点经验点滴

- 采用复合交易测试方案
- 尽量将执行负载测试的机器合理分布
- 并发用户数量极限点
- 加压机器的CPU使用率也有必要监控
- 设置并发点
- HTML与URL录制方式

测试技术 – 常用测试

- 文档测试
- 配置测试
- 外国语言测试
- 易用性测试
- 基于应用服务器的测试
  - Client/Server 测试
  - 网站测试
  - 对Web进行压力测试
- 实时系统测试
- α 测试和 β 测试
- 其他测试
- 调试



实时系统测试

- 实时系统测试

  很多实时系统的时间依赖性和异步性给测试带来新的困难。

  测试用例的设计不仅考虑白盒和黑盒，而且包括事件处理（如中断处理）、数据的时间序列以及处理数据任务（进程）的并发性。

  很多情况下，提供的测试数据有时使得实时系统在某状态下可以正常运行，而同样的数据在系统处于不同状态时有时又会导致错误。

  另外，实时系统的软硬件之间的密切关系也会导致测试问题，软件测试必须考虑硬件故障对软件处理的影响，这种故障很难实时仿真。

  由于实时系统的特殊性和复杂性，还没有一个完善的综合性的测试用例设计方法，但是，大致可以分为以下四个步骤：

实时系统测试

1、任务测试。

第一步，独立测试各任务。对每一个任务设计白盒和黑盒测试用例，并在测试时执行每个任务。任务测试能够发现逻辑和功能错误，但是不能发现时间和行为错误。

2、行为测试。

每种事件单独测试，以随机顺序和随机频率将事件传送给系统，检查系统有否行为错误。

3、任务间测试。

测试与时间有关的错误。任务间同步有否问题，测试通过消息队列和数据存储进行通信的任务，检查数据存储区域大小方面的问题。

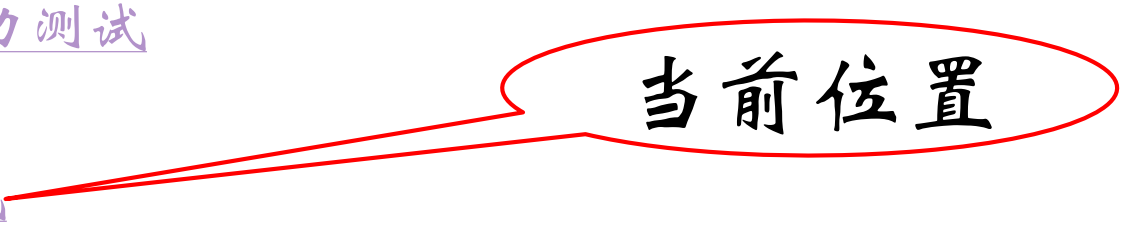
4、系统测试。

集成软件和硬件，并进行大范围的系统测试，以发现软件/硬件接口间的错误。

139

测试技术 – 常用测试

- 文档测试
- 配置测试
- 外国语言测试
- 易用性测试
- 基于应用服务器的测试
  - Client/Server 测试
  - 网站测试
  - 对Web进行压力测试
- 实时系统测试
- α 测试和 β 测试
- 其他测试
- 调试



α 测试和 β 测试

在软件交付使用之后，用户将如何实际使用程序，对于开发者来说是无法预测的。

- α 测试

  α 测试是由一个用户在开发环境下进行的测试，也可以是公司内部的用户在模拟实际操作环境下进行的测试。

- α 测试的目的

  是评价软件产品的FLURPS（即功能、局域化、可使用性、可靠性、性能和支持）。尤其注重产品的界面和特色。

- α 测试开始点

  α 测试可以从软件产品编码结束之时开始，或在模块（子系统）测试完成之后开始，也可以在确认测试过程中产品达到一定的稳定和可靠程度之后再开始。

141

α测试和β测试

- β 测试

  是由软件的多个用户在实际使用环境下进行的测试。这些用户返回有关错误信息给开发者。

  测试时，开发者通常不在测试现场。因而，β 测试是在开发者无法控制的环境下进行的软件现场应用。

  用户记录问题反馈给开发方。在 β 测试中，由用户记下遇到的所有问题，包括真实的以及主观认定的，定期向开发者报告。

- β 测试的目的

  主要衡量产品的FLURPS。着重于产品的支持性，包括文档、客户培训和支持产品生产能力。

- β 测试开始点

  只有当 α 测试达到一定的可靠程度时，才能开始 β 测试。它处在整个测试的最后阶段。同时，产品的所有手册文本也应该在此阶段完全定稿。

142

分布式系统的测试

- 分布式处理中涉及的最基本单位是线程，线程是操作系统进程内部能够独立运行的内容，它拥有自己的程序计数器和本地数据。线程是能够被调度执行的最小单位。
- 分布式系统测试主要面临的问题是并发性、网络化和分布式。
- 并发性是指多个线程同时发生。针对并发性错误的测试主要着重于两个线程的交互。在实际实施交互机制之前应对相关方法进行测试。

分布式系统的测试

- 在网络环境中各个独立的盒子连接到通信设施上，如何实现它们物理上的同步是网络计算的问题。相关的测试就是在组成一个网络系统的各个自治机器之间同步问题的测试。
- 分布式系统使用多进程来支持系统的灵活性一个对象既可以在同一台机器上分布在多个进程中，还可以分布在多个物理上的计算机上。所有这些分布式构件都要能够识别“命名服务”或“注册”对象，能够与其他构件交互。所有在配置文件中登记的机器与构件构成基础结构。 需要考虑与这些分布式构件相关的测试。

分布式系统的测试

- 分布式系统中的路径测试
  - 一条路径是一系列逻辑上连续的语句，它只有在特定的输入下才执行。路径的另一个定义是覆盖变量的定义和使用就形成一条完整的路径。
  - 在分布式系统中的路径就是设计测试用例覆盖一个同步顺序。所谓同步顺序是指同步事件按照特定次序发生的顺序，而同步事件是指一个线程产生另一个线程。
  - 测试应跟踪一个事件到另一个事件的路径。如果从一个同步事件到另一个同步事件有多条可能的控制流路径，只需覆盖其中一条路径。

分布式系统的测试

- 生存周期测试
  - 在分布式系统中，生存周期测试是指选择一系列测试用例，测试任何处于生存期中的对象。特别是在整个生存周期过程中存在多条路径，测试必须选择有代表性的路径以保证最大的覆盖范围。
  - 对于一个类来说，生存周期意味着选择一系列测试，每个测试构造类的一个实例，并通过一系列消息来使用实例，最后再撤销这个实例。
  - 一个有效的生存周期测试应能证实对象本身是否正确，还应能证实被测试项是否能够与它所在的环境正确地交互。对于一个类的实例，在它被撤销后必须检查它占用的资源是否已被释放掉。

分布式系统的测试

- 分布式模型

下面介绍用在分布式系统中的使用某些标准基础结构的测试过程。

1）基本的客户机－服务器模型

客户机－服务器模型是最简单的分布式模型。

在这种模型下，多个客户机都可访问服务器。服务器是单一进程。由于所有客户机都与同一个服务器交互，因此存在单点失败（即服务器出现问题将影响所有客户机）。测试要点：

- 在延时期间，面对同时收到的服务请求，服务器能否把正确结果发送给各个相应的客户机？
- 服务器能否处理快速增长的负载？当负载增加时，服务器的性能可能降低，因此可能选择放弃一部分负载。

分布式系统的测试

2）标准分布式模型 – CORBA

CORBA是对象管理组织OMG开发的公共对象请求代理体系结构，并将它作为分布对象系统的标准体系结构。

这种结构的核心是对象请求代理（ORB）,一个对象通过ORB与系统中的另一个对象通信。CORBA标准的特点:

- 与基础结构相联系的机器可能有不同的操作系统和不同的存储设计；
- 构成分布式系统的构件可以用不同的语言编写；
- 根据对象的分布性和网络中机器的类型，基础结构可以改变它自身的配置。

分布式系统的测试

- 测试要点：
  - 不考虑基础结构的配置，系统能够正确的工作？测试用例应能产生被测试基础结构的各种预期的配置。
  - 在标准基础结构的服务基础上，构造新的测试用例能否被重复使用？测试用例的设计应尽可能地使用基础结构。
  - 新发布的特定基础结构能否与已有的应有有效地结合起来？应有一系列的回归测试，使得新发布的基础结构能够在被集成到产品中之前得到测试。

分布式系统的测试

3）标准分布式模型 – DCOM

- DCOM是 Microsoft 开发并鼓励的一种标准的分布式构件对象模型。
- DCOM “标准” 被描述为包含特定方法的标准接口，每个标准接口都提供了一套特定的服务。单个构件可以完成几个接口的服务，或若干构件中的每一个都能完成统一接口的服务，只是方式不同。
- DCOM是低层次的技术，支持构件间最原始的联系，它不作为应用开发的部分。

分布式系统的测试

- 测试要点：
- 在对各种构件做任意配置时测试者能否正确编排唯一的标识？测试用例应能利用各种构件确保所有必要的连接能够成功。
- 每个构件能否实现必要的接口？测试用例应能利用各种构件确保所有服务是可利用的并能实现期望的功能。
- 标准接口的实现能否提供正确的行为？应针对每一种标准接口有一套测试。

分布式系统的测试

4）标准分布式模型 – RMI

- RMI是Java中的远程方法调用包，它提供一种简化的分布式环境，该环境假定不论连接的是什么样的或什么类型的机器，它们都能运行Java虚拟机。
- RMI提供一个注册对象，参与分布式系统的所有对象必须知道该注册对象监听到哪个端口的消息。
- RMI的最新版本使用ORB的Internet协议(IIOP)，使RMI对象与CORBA对象共同工作。

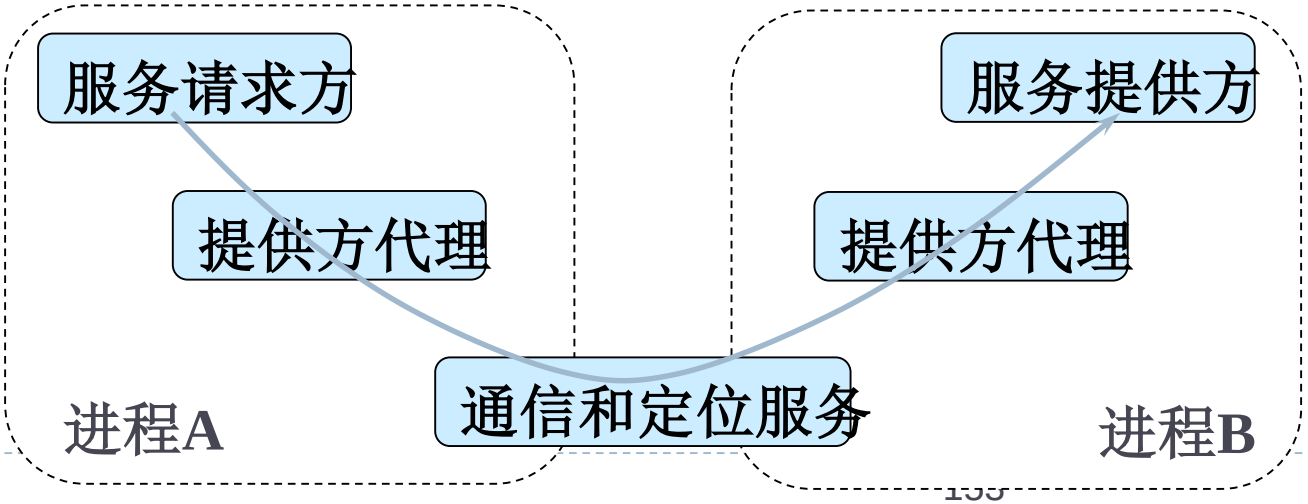
- 测试要点：

  那些种CORBA测试模式能够在以RMI为基础的系统上使用？测试用例的构造很多与CORBA的测试用例相同。

分布式系统的测试

5）一般分布式构件模型

- 分布式系统的基本体系如图，主要活动是服务请求方给服务提供方法送消息。
- 请求首先发送给请求方本地的代理对象，代理方联系通信基础结构并传送请求，通信基础结构实例化服务提供方，从对象定位器的服务中获得服务提供方的引用，并将服务请求传递给它。如果存在结果，则结果沿原路径返回。由于请求代理方可以引导请求，所以提供方不知道分布的细节。



分布式系统的测试

- 用于提供方构件的测试的要点：
  - 根据规范准备测试用例。ProviderTest类应包括相应测试驱动程序，该类继承自GenericModelTester。
  - 设计基于说明的测试用例：对提供方可提供的一系列基于分布式系统的服务设计测试用例，放在GenericModelTester类中，对基于特殊应用服务设计的测试用例则放在ProviderTest类中。
  - 设计基于实现的测试用例：满足路径或代码覆盖的范围标准。
  - 设计交互式测试用例：提供方应与请求服务且具有代表性的请求方交互。每个主要的协议都应得到测试，且至少有一个请求方参与到这个协议中来。
  - 设计基于状态的测试用例：当请求方发出请求时，提供方可能只处于一种状态。但它可能同时从多个请求方接收请求，因此应使用所有必要的协议，使提供方经历所有可能的状态。

分布式系统的测试

- 用于请求方构件的测试的要点：
  - 根据规范准备测试用例。RequesterTest类应包括相应测试驱动程序。在这个类中应提供用于执行功能性、结构性和交互性测试用例的操作。测试类在应用程序中可当作交互类使用。
  - 设计基于说明的测试用例：在API文档中构造每个方法的前置条件和后置条件。执行用于每个方法中每个后置条件中所有语句的测试用例。
  - 设计基于实现的测试用例：满足一般的范围标准。
  - 设计交互式测试用例：重点放在请求方与每个提供方之间的完整协议。为使得计时效果较为明显，应适当插入人为延时。
  - 设计基于状态的测试用例：请求方有一系列与它自己的分布相关的状态，如与基础结构相连或撤销连接等。测试应包括一系列经历所有这些状态的测试用例，还应包含与应用语义学相关的一些状态。

分布式系统的测试

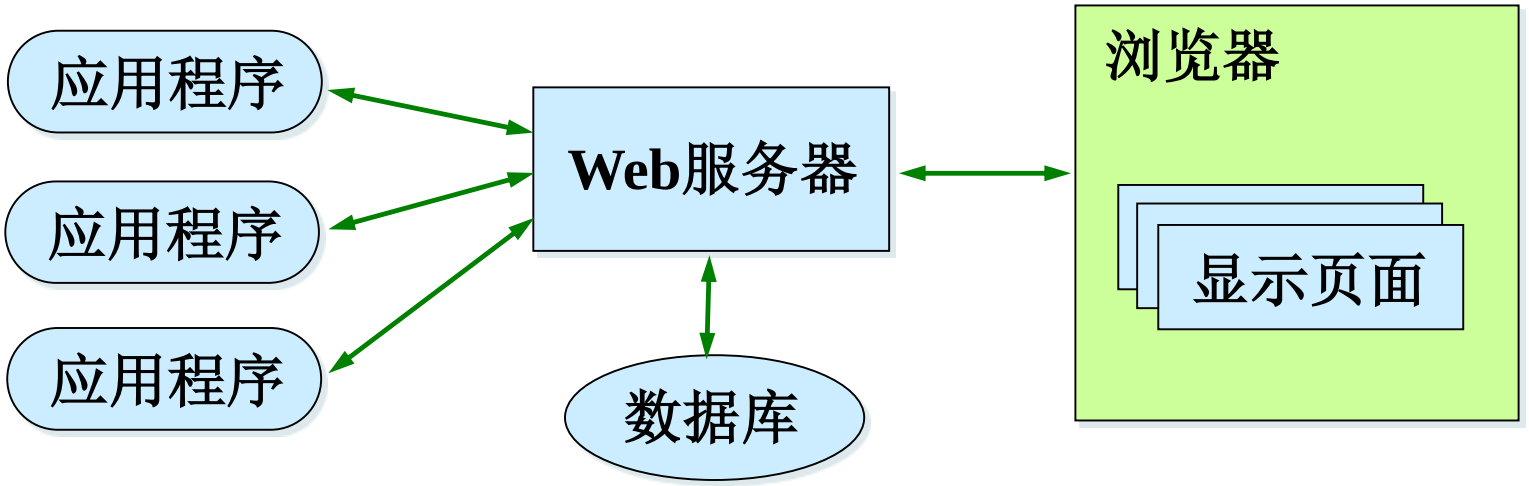
6）最大的分布式系统 – Internet测试

因特网上可以连接、增加或删除服务器。

这类测试重点在Web服务器上和因特网应用程序的生存周期上。

a）Web服务器测试

Web服务器体系结构如图所示。Web页面和浏览器、应用服务器协作产生应用。



分布式系统的测试

- 这些系统使得用户服务自动化：页面接受输入，并放入一个用户订购表中。然后产生一条顾客记录，调用能够自动产生一个账号和密码的应用程序。最后把账号信息通过Email发送给用户。
- 测试涉及两方面。

  首先，执行脚本看它所做的工作是否是其所期望做的？

  其次，检查应用程序和网页所期待的当前数据是否正确。是否是所期望的格式？

b）因特网应用程序生存周期的测试

- 跨越不同平台的一系列用户事务测试。

A/B 测试

- A/B 测试

  A/B测试其实是一种“先验”的实验体系，属于预测型结论，与“后验”的归纳性结论差别巨大。

  A/B测试的目的在于通过科学的实验设计、采样样本代表性、流量分割与小流量测试等方式来获得具有代表性的实验结论，并确信该结论在推广到全部流量可信。

  如对某电商APP某功能使用方式进行测试，假设产品有100万用户、这100万用户有各式各样的特点（性别、地域、手机品牌与型号、甚至是不是爱点按钮等行为。）。然后将这100万用户根据样本特征与相似性规则等聚类分为100组，每组取少数代表（假定1个）进行测试：

  在A/B测试的实验中，需要保证小流量的实验具备代表性，也就是说1%的流量做出来的实验结果，可以推广到100%的用户，为了保证这一点，需要保证1%的流量的样本特征与100%流量的样本特征具备相似性。

其他测试

- 安装测试
  - 安装测试的目的不是找软件错误，而是找安装错误。
  - 在安装软件系统时，会有多种选择。
    - 要分配和装入文件与程序库
    - 布置适用的硬件配置
    - 进行程序的联结。
  - 而安装测试就是要找出在这些安装过程中出现的错误。
  - 安装测试是在系统安装之后进行测试。它要检验：
    - 用户选择的一套任选方案是否相容；
    - 系统的每一部分是否都齐全；
    - 所有文件是否都已产生并确有所需要的内容；
    - 硬件的配置是否合理，等等。

159

其他测试

- 敏感性测试：

  是强度测试的一个变种。在程序有效数据界限内一个小范围内的一组数据可能引起极端的或不平稳的错误处理出现，或者导致极度的性能下降的情况发生。此测试用以发现可能引起这种不稳定性或不正常处理的某些数据组合。

- 可启动/停止测试

  系统启动及关机阶段，能否正常运行。

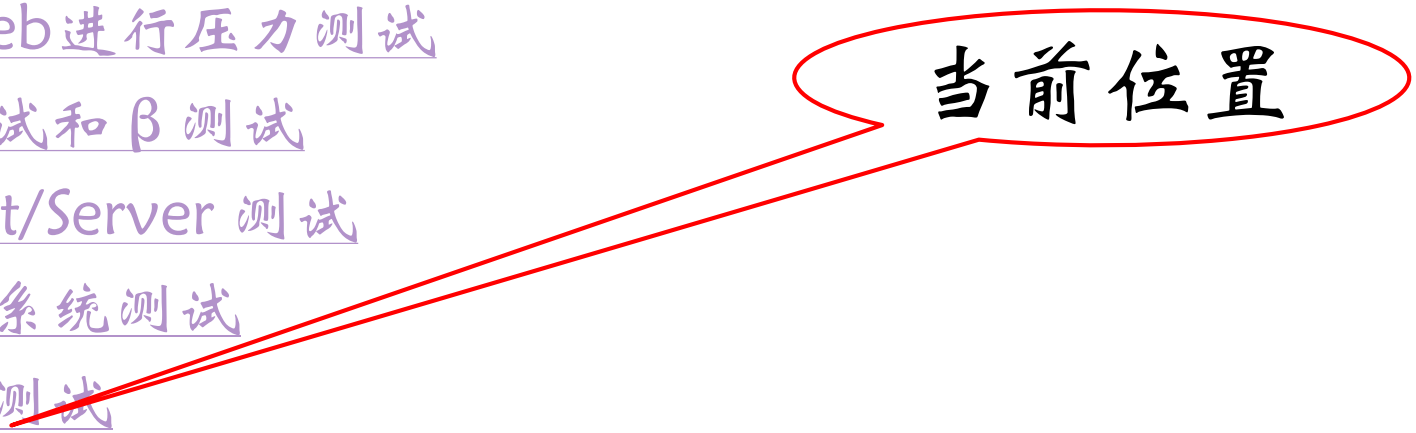
- 互连测试

  测试不同系统间的互连性

160

测试技术 – 常用测试

- 配置测试
- 外国语言测试
- 易用性测试
- 文档测试
- 网站测试
- 对Web进行压力测试
- α测试和β测试
- Client/Server 测试
- 实时系统测试
- 其他测试
- 调试



调试

- 软件调试和软件测试有完全不同的含义：

  测试：目的是显示存在的错误。是可以系统进行的计划过程，可以指导测试用例设计，定义测试策略，测试结果可以和预期的结果进行对照评估。测试是检验，发现可疑的错误征兆。

  调试：目的是发现错误或导致程序失效的错误原因，并修改程序以修正错误。是测试发现错误之后消除错误的过程。调试充分利用测试结果和测试提供的信息，全面分析，找出错误根源，修正错误。

- 成功的测试发现错误的症状引起调试的进行。
- 调试比开发难以让人理解。

162

调试

- 测试与调试的主要区别？

(1) 测试从一个侧面证明程序员的失败；调试证明程序员的正确；

(2) 测试从已知条件开始，使用预先定义的程序，且有预知的结果，不可预见的仅是程序是否通过测试；调试从不可知内部条件开始，除统计性调试外，结果是不可预见的；

(3) 测试有计划并且要进行测试设计；调试不受时间约束；

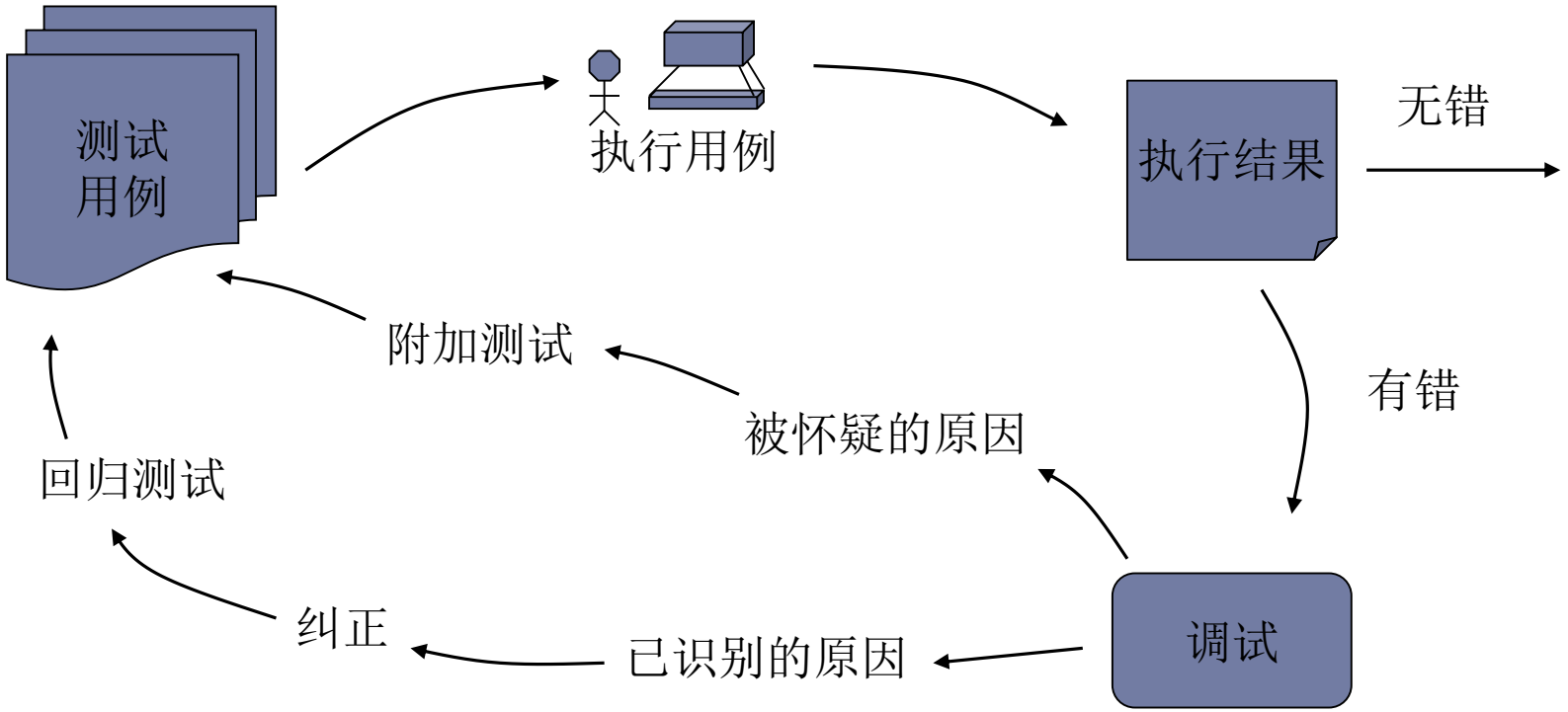
(4) 测试是发现错误、改正错误、重新测试的过程；调试是一个推理的过程；

(5) 测试执行是有规程的；调试执行要求程序员进行必要的推理；

(6) 测试由独立的测试组在不了解软件设计的条件下完成；调试由了解详细设计的程序员完成；

(7) 大多数测试的执行和设计可由工具支持；调试用的工具主要是调试器。

调试过程



调试有两种结果：

- 发现问题原因，改正
- 未发现问题原因，假设一个错误原因，设计Testcase验证假设

164

调试

- 调试的困难在于：
  - 症状和原因可能相距很远，高度耦合的程序结构加深了这种情况。
  - 症状可能在另一错误改正后消失或暂时性消失
  - 症状可能不是有软件错误引起的(如舍入误差)
  - 症状可能有人为错误引起
  - 症状可能由于时间问题而不是处理问题
  - 可能难以再现症状出现的输入条件
  - 症状可能间断出现的
  - 症状可能由分布在许多不同任务中的原因引起的

165

例：台灯不亮了

①是否整个屋子无电：总闸，外面坏了。判断邻居家也黑。

②否，台灯插到外面插座。（不亮）

或把正常电器插到此插座上。（可正常工作）

③台灯坏了

④换好灯泡，亮了（原灯泡坏了）

技术：使用断点，桌面检查，内存转储，审查，可逆执行，单步操作和跟踪，调试是对在测试与正常操作中已发现错误的反馈

168

调试的三种实现方法

- 蛮力法(Brute force)
- 回溯法(Backtracking)
- 原因排除法(Cause elimination)

每种方法都可以使用调试工具辅助完成

- 带调试功能的编译器
- 动态调试辅助工具("跟踪器")
- 自动的测试用例生成器
- 内存映像工具
- 交叉引用工具

169

调试(Debug)

- 修正错误，可能带来其它错误。Van Vleck 提出每个软件工程师排错修改前问三个问题：
  - 这个错误原因在程序其它地方也产生过吗？
  - 我要进行的修改，可能会引发的“下一个错误”是什么？
  - 为防止这个错误，我们应当做什么？

170