系统测试方法

1、功能测试(function testing)，是系统测试中最基本的测试，不管软件内部的实现逻辑，主要根据产品的需求规格说明书和测试需求列表验证产品的功能实现是否符合产品的需求规格。

功能测试是为了发现以下错误：从和用户角度来进行功能验证，以确认：（1）每个功能是否都能正常使用

（2）是否实现了产品规格说明书的要求

（3）是否能适当地接受输入数据而产生正确的输出结果

功能测试用例是功能测试工作的核心，常见的设计方法有如下几种：等价类划分法、边界值分析法、因果图、决策表、正交实验设计、基于风险的测试、错误推测法。

功能测试又可以细分为很多种：逻辑功能测试(logic function testing)、界面测试(UI testing)、易用性测试(usability testing)、安装测试(installation testing)、兼容性测试(compatibility testing)

界面测试

易用性与合理性：步骤繁琐的操作，比例不协调、摆放凌乱的窗口和控件，层次过多的子窗口和菜单
规范性：不符合 Windows 规范的控件设计，与常规 Windows 操作不符的流程与操作等
容错性：编辑控件对非法字符、超出边界值的输入处理不当或没有提示，容易造成系统重启、数据删除丢失等的操作没有提示等
帮助：无帮助信息提供，或者不提供获取帮助的快捷操作
美观与风格：界面颜色不协调、界面风格与公司相关产品风格不符、与业界通用风格不符，图片、图标等不符合公司 CI 规范。
资源：界面长时间运行操作造成系统内存耗尽、界面对系统资源独占使用等

安装测试

这里的安装是广义的，包括安装和卸载

理想情况下，一个软件的安装程序应当可以较好的与已有系统相兼容，并有相应的提示界面供用户参考，安装完毕并实现其功能。
重要性：安装是用户使用的第一步
目的：验证系统成功安装的能力，保证程序安装后能正常运行，并能够成功卸载。
要求：安装过程清晰、简单，且系统文档中有详细说明。

安装测试应考虑的方面主要有以下：

安装手册测试，安装前应先备份测试机的注册表。
自动安装和手工配置测试。
异常配置或状态情况（继电等）测试。
检查安装后目录结构和文件，以及文件属性。
所有的运行环境上测试。
卸载测试。
检测安装该程序是否对其他的应用程序造成影响。

兼容性测试(检查软件是否能够与其他软件正确协作，试点包括：)

操作系统兼容性测试
同一平台上其他相关应用软件的兼容性
硬件兼容性
数据共享兼容性测试
软件新旧数据转换

易用性测试

从软件使用的合理性和方便性等角度对软件系统进行检查，来发现软件中不方便用户使用的地方。
比如，水龙头红色和蓝色标记、超市里各种购物车等

常用的易用性测试用例

常用的功能要有快捷方式
功能相同或相近的控件放到一个区域
可能造成较长时间的等待的操作能提供取消功能
工具栏上的图标要能直观代表要完成的操作
必须提供友好的联机帮助
软件出现问题时，要在提示信息时提供相应的技术支持联系方式

2、性能测试，是软件测试的高端领域，通常我们说的高级软件测试工程师一般就是指性能测试工程师和白盒测试工程师。

性能测试是用来保证系统发布后，产品的性能满足用户要求。

性能测试在软件质量保证中起重要作用。

没有完全的标准定义，从广义上来说，压力测试、负载测试、并发测试、大数据量测试、配置测试、可靠性测试、强度测试等等均属于性能测试范畴。

在软件系统日益复杂的今天，性能已经成为软件质量最重要的衡量标准之一。例如，对于软件测试教学网站，我们至少需要测试这些性能指标：服务器响应速度、客户端上传下载文件的速度和文件大小、能同时支持的在线人数、在线教学视频的播放质量、系统运行的可靠性（稳定性）、邮箱容量、邮件收发速度……

系统的性能是一个很大的概念，覆盖面非常广泛，对一个软件系统而言，包括：执行效率、资源占用率、稳定性、安全性、兼容性、可扩展性、可靠性等等。

对软件性能的关注是多层面的：

（1）用户对软件性能的关注

① 软件对用户操作的响应时间，如用户提交一个查询操作，打开一个 web 页面的链接等等

② 业务可用度，或者系统的服务水平如何

（2）系统管理员对软件性能的关注

管理员关心的问题	软件性能描述
服务器的资源使用状况合理吗	资源利用率
应用服务器和数据库的资源使用状况合理吗	资源利用率
系统是否能够实现扩展	系统可扩展性
系统最多能支持多少用户的访问？系统最大的业务处理量是多少	系统容量
系统性能可能的瓶颈在哪里	系统可扩展性
更换哪些设备能够提高系统性能	系统可扩展性
系统能否支持7×24小时的业务访问	系统稳定性

（3）开发人员对软件性能的关注

开发人员关心的问题	问题所属层次
架构设计是否合理	系统架构
数据库设计是否存在问题	数据库设计
代码是否存在性能方面的问题	代码
系统中是否有不合理的内存使用方式	代码
系统中是否存在不合理的线程同步方式	设计与代码
系统中是否存在不合理的资源竞争	设计与代码

软件性能的几个指标

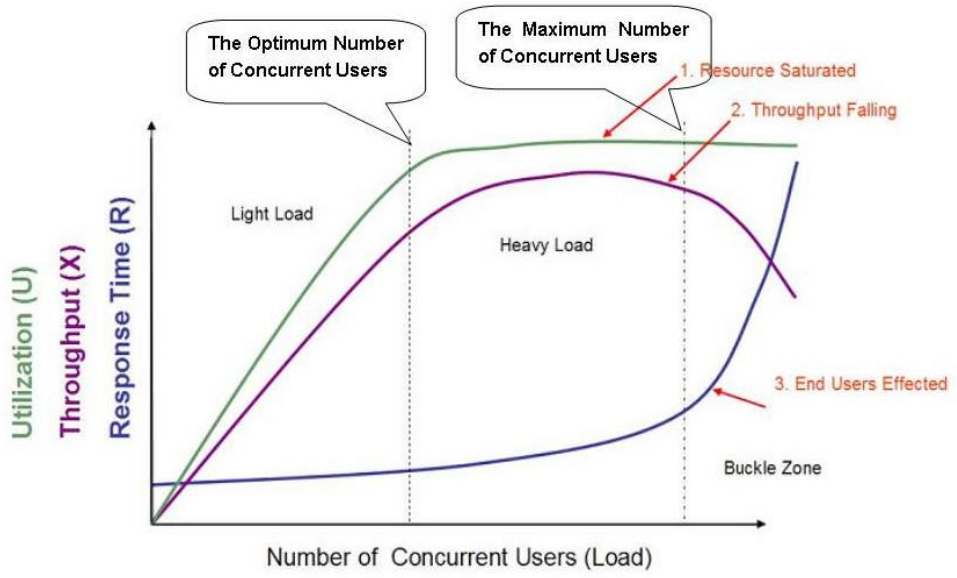
（1）响应时间：应用系统从请求发出开始到客户端接收到最后的一个字节数据所消耗的时间，是作为用户视角的软件性能的主要体现。

（2）并发用户数：系统用户数、同时在线用户人数

（3）吞吐量：单位时间内系统处理的客户请求的数量，直接体现软件系统的性能承载能力。表示方法：请求数/秒、页面数/秒、人数/天、处理的业务数/小时

（4）资源占有率：体现软件的面向用户特性

用“理发店模型”来理解软件性能的几个指标



Utilization 利用率　Throughput 吞吐率　Response time 响应时间

图中划分了三个区域：
Light Load（较轻的压力）、
Heavy Load（较重的压力）
Buckle Zone（用户无法忍受并放弃请求）

在 Light Load 和 Heavy Load 两个区域交界处的并发用户数，我们称为“最佳并发用户数（The Optimum Number of Concurrent Users）”，
而 Heavy Load 和 Buckle Zone 两个区域交界处的并发用户数则称为“最大并发用户数

进一步分析

当系统的负载等于最佳并发用户数时，系统的整体效率最高，没有资源被浪费，用户也不需要等待；

当系统负载处于最佳并发用户数和最大并发用户数之间时，系统可以继续工作，但是用户的等待时间延长，满意度开始降低，并且如果负载一直持续，将最终会导致有些用户无法忍受而放弃；

而当系统负载大于最大并发用户数时，将注定会导致某些用户无法忍受超长的响应时间而放弃

性能测试的目的：在真实环境下检测系统性能，评估系统性能以及服务等级的满足情况；分析系统瓶颈、优化系统。

性能测试一般分为四种：

（1）一般性能测试(狭义情况下的性能测试)，指让被测系统在正常的软硬件环境下运行，不向其施加任何压力的性能测试。

对于单机版的软件

我们就在其推荐配置下运行软件，检查 CPU 利用率，内存的占有率等性能指标以及软件主要事务的平均响应时间

对 C/S，B/S 结构的软件

则测试单个系统登录后，系统主要事务的响应时间和服务器的资源消耗情况，举例：测试 163 邮箱的登录模块

（2）稳定性测试(可靠性测试)，是指连续运行被测系统，检查系统运行时的稳定程度

通常用 MTBF(mean time between failure,错误发生的平均时间间隔)来衡量系统的稳定性。MTBF 越大，系统的稳定性越强。

方法比较简单，采用 24×7 的方式让系统不间断运行，具体运行时间视项目的实际情况而定。

（3）负载测试，通常指，让被测系统在其能忍受的压力的极限范围之内连续运行，来测试系统的稳定性

与稳定性测试较为类似，都是让系统连续运行，区别是，负载测试需要给被测系统施加其刚好能承受的压力。

负载测试为我们测试系统在临界状态下运行是否稳定提供一种方法。绝大多数的负载测试都是通过自动化工具完成的。被称作软件的“体能测试”

（4）压力测试，是通过逐步增加系统负载来测试系统性能的变化，并最终确定在什么负载条件下系统性能处于失效状态，以此来获得系统性能提供的最大服务级别的测试。

通常持续不断地给被测系统增加压力，直到被测系统压垮为止，来测试系统所能承受的最大压力

压力测试方法具有如下特点：

（1）压力测试是检查系统处于压力情况下的能力表现

如通过增加并发用户的数量，检测系统的服务能力和水平；通过增加文件记录数来检测数据处理的能力和水平等等。

（2） 压力测试一般通过模拟方法进行

可以对系统内存和 CPU 利用率、数据库的连接数量、数据库服务器的 CPU 利用率上进行模拟，以获得测量结果。

如将压力的基准设定为：内存使用率达到 75%以上、CPU 使用率达到 75%以上，并在此观测系统响应时间、系统有无错误产生。

（3） 压力测试一般用于测试系统的稳定性

例如：假设一个人很轻松就能背 1 袋米，背 2 袋米很吃力，最多就能背 3 袋米。

一般性能测试：就让他背 1 袋米

稳定性测试：让他背 1 袋米，然后让他去操场上跑圈，看多久累倒

负载测试：背两袋米去操场跑圈，看多久累倒

压力测试：让他背 2 袋米，3 袋米，4 袋米……不停直到累倒，发现最多背三袋。

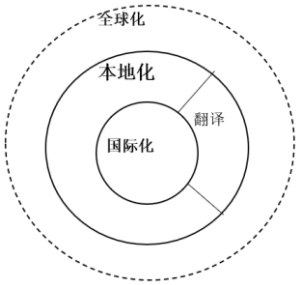
3、本地化测试基础

软件全球化(G11N)

= 软件国际化

(SW Internationalization, I18N)

+ 软件本地化(SW Localization, L10N)



软件国际化是在软件设计和文档开发过程中，使得功能和代码设计能处理多种语言和文化传统，使创建不同语言版本时，不需要重新设计源程序代码的软件工程方法。是软件本地化的前提。软件国际化的理想状态是使软件本地化过程不需要修改任何代码。

软件本地化是将一个软件产品按特定国家/地区或语言市场的需要进行加工，使之满足特定市场上的用户对语言和文化的特殊要求的软件

生产活动。软件本地化并不只是单纯地翻译用户界面、用户手册和联机帮助。软件本地化为的是克服产品本身的文化障碍，从而吸引更多的用户。

I18N 应该满足下列需求：

（1）支持 Unicode 字符集、双字节的字符（既可以处理类似于英文的单字节语言，又可以处理类似于中文，日文的双字节语言）。如 ISO 8859 所定义的字符虽然在不同的国家中广泛地使用，可是在不同国家间却经常出现不兼容的情况。

（2）分离程序代码和显示内容。如建立资源文件来处理这些内容。

（3）消除 Hard code（硬代码，指程序中包含的一些特定数据，国际化软件编程不能像一次性软件那样随意，许多东西不能“写死”），尽量使用变量处理，将数据存储在数据库或初始化文件中。

（4）使用 Header files 去定义经常被调用的代码段；

（5）改善翻译文本尺寸，具有调整的灵活性

（6）支持各个国家的键盘设置，但要统一热键。

（7）支持文字排序和大小写转换；

（8）支持各个国家的度量衡，时区，货币单位格式等的设置；

（9）拥有国际化用户界面设计。

L10N 应满足的条件

（1）翻译（软件本地化的基本工作）

（2）适应地区文化、宗教要求。（如不同地区人们对数字，颜色的不同喜好）

（3）度量衡和时区等

（4）软件用户界面（UI）（翻译后需调整，以适应本地化要求）

（5）联机文档（帮助文档和功能性的 PDF 文档）

（6）热键设置

软件本地化测试是对本地化的软件进行测试的活动。软件本地化测试目的主要是：

（1）保证本地化的软件与源语言软件具有相同的功能和性能。

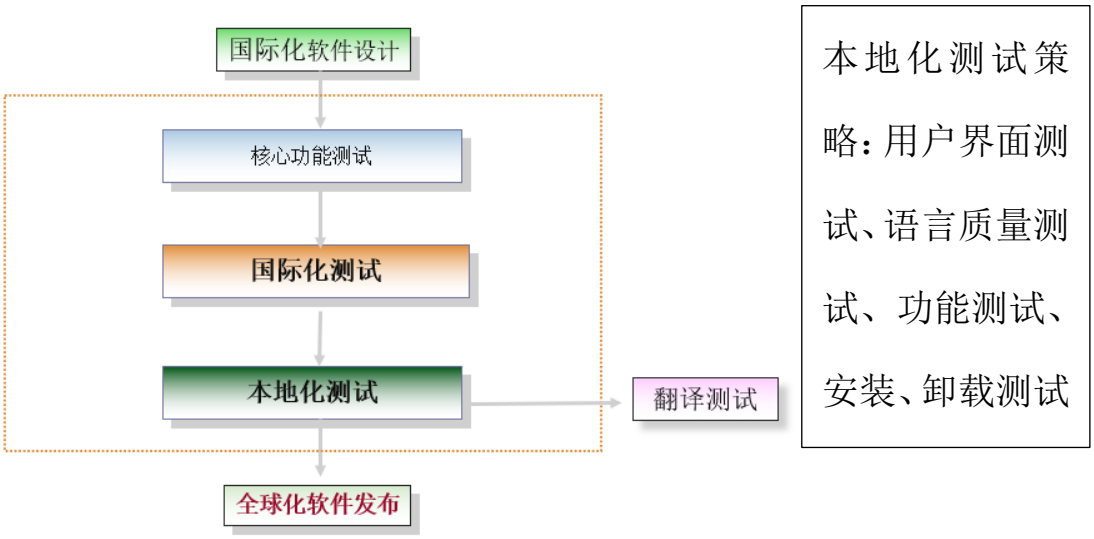
（2）保证本地化的软件在语言、文化、传统观念等方面符合当地用户的习惯。

（3）软件本地化测试是尽可能多的发现本地化软件中由于本地化而引起的软件 bug。

软件本地化测试既有一般测试的特点，又有自己的测试要求，它们二者的主要区别是：

本地化软件测试的重点是报告与本地化有关的 bug，包括翻译语言的质量、与区域有关的特征、软件界面控件布局等。**一般测试**主要测试软件的功能和性能。本地化软件测试通常需要与源软件**对比测试**，确认错误是否属于本地化错误，还是源语言功能错误。

全球化软件发布流程



本地化用户界面测试内容：

（1）主要对软件的界面文字和控件布局（大小和位置）进行测试。

（2）通常为最简单的测试类型，不需要过多的语言翻译知识和测试工具

（3）有些界面比较深入，某些 BUG 可能隐藏的很深

本地化用户界面测试清单（checklist）

（1）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

（2）控件或文字没有对齐（Misaligned）

（3）控件位置重叠（Overlapped）

（4）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

（5）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

（6）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

（7）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

（8）多余的空格（Extra space）

本地化文字字符之间存在多余的空格

本地化语言质量测试

（1）测试翻译是否准确专业，布局是否合理美观。还要注意联机帮助文件和软件用户界面的一致性。

（2）需要参照源软件，进行对比测试。

本地化语言质量测试清单(checklist)

（1）字符没有本地化（Unlocalized strings）字符不完整地本地化（Incomplete localized strings）

①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
②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

（2）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

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

①相同的文字前后翻译不一致
②相同的文字各语言之间不一致
③相同的文字软件用户界面与联机帮助文件不一致

（4）过度本地化（Over localization）

①不应该本地化的字符进行了本地化
②标点符号、版权、商标符号错误（Incorrect punctuation, Copyright）
③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

本地化功能测试内容

（1）主要测试软件经过本地化后，软件的功能是否与源软件一致，是否存在因本地化而产生的功能错误。
（2）相对于其他测试类型具有较大难度，可能需要多步组合操作才能完成。

本地化功能测试(checklist)

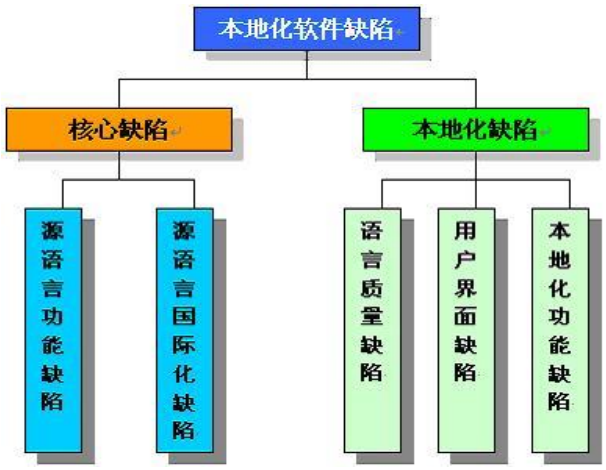
（1）功能不起作用（Not working）
菜单、对话框的按钮、超链接不起作用
（2）功能错误（Error function）
①菜单、对话框的按钮、超链接引起程序崩溃
②菜单、对话框的按钮、超链接带来与源语言软件不一致的错误结果
③超链接没有链接到本地化的网站或页面
④软件的功能不符合本地化用户的使用要求
（3）热键和快捷键错误（Error hot keys and short-cut keys）
①菜单或对话框中存在重复的热键
②本地化软件中缺少热键或快捷键
③不一致的热键或快捷键
④快捷键或快捷键无效

安装、卸载测试内容

（1）测试本地化软件是否可以正确的安装卸载在本地语言的操作系统上（包括是否支持本地语言的安装目录名）。
（2）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。

本地化测试的缺陷组成

本地化软件的错误的产生是多方面的，不能仅仅归结为软件本地化过程带来的错误。实际上，良好的国际化设计的源程序是减少软件本地化错误的根本保证。



缺陷产生的原因

核心缺陷：是由于源程序软件编码错误引起的，例如开发人员对于某个功能模块的编码错误，或者没有考虑软件的国际化和本地化能力，而将代码设定为只接受的某些语言。

本地化缺陷：由于软件本地化过程引起的，例如语言翻译质量较差、界面控件布局不当等

缺陷修正方法

本地化缺陷（只出现在本地化的版本上）
　可以由本地化工程师修改本地化软件相关资源文件解决，例如修改错误的翻译文字、调整控件的大小和位置等。
源语言功能缺陷(既出现在本地化软件，也可以在源语言软件上复现）
源语言国际化缺陷(只出现在本地化版本中)只能通过修改程序代码出现
属于源语言软件的设计错误，这类缺陷只能由软件开发人员修正。

做测试最重要的是什么？（做测试需要哪方面的能力/素质）

　概念、经验、眼光、境界、心态、沟通、突破

4、配置测试

验证系统在不同的系统配置下能否正确工作，这些配置包括：软件，硬件，网络等。

配置测试是为了保证测试的软件使用尽量多样化的硬件组合，采用不同的组件、外设、接口等查看测试的软件在不同配置下的可用性。

注意：不同的软件有不同的针对点，因此测试的内容是不一样的。比如，一个游戏软件，一个图像处理软件，一个视频播放软件等等

进行配置测试是新手软件测试员经常被分配的任务，因为它容易定义，是基本组织技巧和等价分配技术的敲门砖。

配置测试是与其他项目小组成员合作（co-operation）的任务，是管理员快速验证结果的手段。

PC 机的各种可能配置

品牌机：DELL、HP、联想等，每一家都自行设计部件或采购其他厂商的部件生产自己的 PC 机

内部配件：主板、板卡、其他内部设备

外设：可以插在主板上，从外部操纵的设备，如打印机、扫描仪、鼠标、键盘、显示器、游戏杆；

接口：通过各种接口设配器连入 PC 的，接口可以是 PC 内部，也可以是外部的，如 ISA,PCI,USB,串口，RJ-45,PS/2 等等

可选项和内存

设备驱动程序：从技术上讲驱动程序是软件，但出于测试目的，认为他们是硬件配置

配置缺陷分类

（1）软件可能包含在多种配置中都出现的缺陷；

（2）软件可能只包含在某一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揭示的缺陷。

如何分离配置权限

（1）判断缺陷是配置问题的技巧：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。

（2）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。

（3）如果在不同的配置中都出现，就可能是通用的问题

一定注意：配置问题可能跨越整个等价划分，而不仅仅是特定型号

例：一种 3D 游戏，画面丰富，多种音效，允许多个用户联机对战，还可以打印游戏细节。
市场调查：336 种显卡，210 种声卡，1500 种网卡，1200 种打印机。
测试组合=336×210×1500×1200

总计上亿种，规模庞大，如何减少工作量？
等价类划分，合并同类配置，但同时带来风险
即使把配置的可能性等价划分至最低限度，仍然需要安装许多硬件，代价高昂。
以下是解决方法：
一、只买可以或者将会经常使用的配置
二、小组中的每一个测试员都配备不同的硬件
三、与硬件生产厂商联系，寻求租借甚至赠送某些硬件
四、动员公司力量，群策群力，帮助测试

配置测试的步骤

一、确定所需的硬件类型
应用程序需要打印吗？考虑测试打印机
应用程序需要发声吗？考虑测试声卡
应用程序需要处理图形图像吗？考虑测试显卡
把软件安装盘放在面前，考虑需要哪些硬件使其工作
容易忽略的一个特性例子是联机注册使用的硬件类型—modem 和网络通信考虑在内
二、确定有那些厂商的硬件、型号和驱动程序可用
与销售和市场人员一起制订要测试的清单
找近期的专业杂志看有哪些硬件可用，哪些正在或曾经流行
研究是否有贴牌生产（OEM）的现象（需要认真考虑！典型例子：“同德五虎---七彩虹，双敏，昂达，盈通，铭瑄”）
确定要测试的设备驱动程序
操作系统自带的驱动程序
硬件附带的驱动程序
网站上的最新更新
三、确定可能的硬件特性、模式和选项
彩色打印机可以打彩色也可以打黑白；可以在不同模式下打印
显卡有不同的色彩设置和分辨率
每种设备都有选项，软件没有必要完全支持
四、将确定后的硬件配置缩减为可控制范围
把所有配置信息放在电子表格里，列出生产厂商，型号，驱动程序版本及其他可选项，软件测试人员和开发小组共同确定测试哪些配置
五、明确与硬件配置有关的软件的唯一特性
不应该也没有必要在每种配置中完全测试软件，只需要测试那些与硬件交互时互不相同（不同等级划分）的特性即可！
例：写字板之类的文字处理程序做配置测试：保存和打开特性 打印
选择唯一特性并非易事，甚至不是表面看到的那么简单。必须先进行黑盒测试（通过查看产品找出明显特性），然后与开发小组交流了解该特性与哪些配置有相关性
六、设计在每一种配置中执行的测试用例
以对文字处理软件的配置测试为例
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。

七、在每种配置中执行测试

软件测试员执行测试用例，仔细记录并向开发小组汇报，必要时还要向硬件厂商报告

注意明确配置缺陷的准确原因很难，需要和开发人员以及白盒测试人员密切配合，分离问题的原因

八、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周期，可能最初只是测试一些配置，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

5、回归测试(一种验证已变更系统完整性与正确性的测试技术)

回归测试的背景：

（1）软件总是在变化：软件不断演化—开发、维护、升级

（2）对软件的要求从未停止：软件需求不断变更

（3）软件还要适应不断变化的环境：技术更新和软/硬件升级

注：软件的每次改变都会引入潜在的风险—缺陷

对软件变化的要求：修改后的功能是否达到预期，原有功能是否被损害，解决方案：回归测试

对之前已经测试过或者经过修改的程序进行重新测试，以保证修改没有引入新的错误或者由于修改发现以前未发现的错误。

比如迅雷 5.9→迅雷 7.1
需要将 5.9 版本所执行的测试再重复一遍，而不仅仅是只测试出错的地方
如果引入新增加的功能怎么办？
补充新的测试用例，但这部分已经不属于回归测试的范围了。

测试用例的维护

测试用例的维护是一个不间断的过程，通常可以将软件开发的基线作为基准，维护的主要内容包括下述几个方面：

（1）删除过时的测试用例

需求的改变等原因可能会使一个基线测试用例不再适合被测试系统，这些测试用例就会过时。举例：数值的界限发生改变

（2）改进不受控制的测试用例

随着软件项目的进展，测试用例库中的用例会不断增加，其中会出现一些对输入或运行状态十分敏感的测试用例。这些测试不容易重复且结果难以控制，会影响回归测试的效率，需要进行改进，使其达到可重复和可控制的要求。

（3）删除冗余的测试用例

存在两个或者更多个测试用例针对一组相同的输入和输出进行测试

（4）增添新的测试用例

如果某个程序段、构件或关键的接口在现有的测试中没有被测试，那么应该开发新测试用例重新对其进行测试。并将新开发的测试用例合并到基线测试包中。

回归测试伴随着软件开发的每一个阶段，而且常常需要重复进行，因此会给测试员带来疲惫感，所以，引入自动化测试，是很不错的方法。

6、冒烟测试(smoke testing)，是指在对一个新版本进行系统大规模的测试之前，先验证一下软件的基本功能是否实现，是否具备可测性。

来源于电路板的测试

目的：确保一个程序中最重要的功能被执行，而不需要被基本的细节困扰。

方法：确定“功能点”，用功能点进行测试

测试时间：在编写代码之前，就编写冒烟测试用例。这叫做“测试驱动开发”（test-driven developing）

什么是测试驱动开发

（1）在写产品代码之前，先写它的单元测试（Unit Tests）
（2）没有单元测试的Class不允许作为产品代码

（3）单元测试例子决定了如何写产品代码
（4）不断地成功运行所有的单元测试代码
（5）不断的完善单元测试代码以及系统逻辑代码(重构)

怎么做测试驱动开发

1. 拿到需求，理解需求，先思考要实现哪些功能
2. 整理一份测试清单，根据第 1 步写出要测试的清单
3. 测试先行
    a. 写出测试代码，包括真实逻辑中要用到的类和接口
    b. 编写测试方法中的断言（这是你期望的结果）
4. 运行测试代码
5. 编写真实逻辑代码

循环第 4,5 步，直到第 4 步运行通过（绿条）为止结束

7、随机测试(random testing)，是根据测试者的经验随机的选取功能点对软件进行有针对性的测试。这种测试没有用例的指导，完全根据测试员自己的经验和相关知识来测试，又称“猴子测试”

8、系统测试的若干原则

① 应尽早地开始系统测试工作
② 充分注意测试中的缺陷密集现象
③ 严格执行测试计划，排除测试的随意性
④ 对测试过程和测试结果应进行评价，确保测试过程的有效性
⑤ 妥善保存测试计划、测试用例、故障统计和最终分析报告，为维护提供方便
⑥ 对于被测试系统要进行正常和异常两方面的测试
⑦ 在系统测试计划中，要按照资源和项目的要求清晰地定义一个完整的退出准则，这是一种权衡投入/产出比的原则，测试既不要不充分，也不要过分

9、系统测试执行和评估阶段常用度量

测试用例通过率（百分比） = 本次测试中通过的用例数/实际执行的用例数

测试用例覆盖率(百分比) = 本次测试中实际执行的用例数/计划执行的用例数

本次测试中测试通过的系统测试用例数目(条)
本次测试中测试不通过的系统测试用例数目(条)
发现的缺陷数目及缺陷等级（个数、级别）
已经解决的缺陷数目和缺陷等级（个数、级别）
遗留的缺陷数目及缺陷等级（个数、级别）
缺陷密度（分布图）
测试的工时（人时）
系统测试的需求覆盖率