**1、按照测试方法分类**

* 静态和动态测试
  + 静态测试

静态测试是不运行被测试本身而寻找程序代码中可能存在的错误或评估程序代码的过程。静态测试通过分析或者监察源程序的语法、结构、过程、接口等来检查程序的正确性，找出问题

* + 动态测试

动态是运行被测试程序，输入相应的测试数据，检查运行结果与预期结果的差异，判定执行结果是否符合要求，从而检验程序的正确性、可靠性和有效性，并分析系统运行效率和健壮性等性能。

* 黑盒测试、灰盒测试、白盒测试
  + 黑盒测试

黑盒测试（Black-box Testing）：又称为功能测试、数据驱动测试或者基于规格说明书的测试。注重于测试软件的功能需求。

* + 灰盒测试

灰盒测试是介于黑盒测试与白盒测试之间的测试方法。在执行白盒测试的时候考虑黑盒测试的方法。

* + 白盒测试

白盒测试（Write-box Testing）：又称结构测试、玻璃盒测试、逻辑驱动测试或基于程序本身的测试

* 冒烟测试
  + 冒烟测试（Smoke testing）的对象是每一个新编译的需要正式测试的软件版本，目的是确认软件基本功能正常，可以进行后续的正式测试工作。

一、冒烟测试的准备工作

1、主流程和主功能的确认

* + - 正式测试的前提是主功能、主流程通过。
    - 测试人员对自己项目的整体把握程度要强，在前期了解清楚需求后，把最重要的流程和功能列举出来，在冒烟测试前和开发人员一一确认，这对于冒烟测试是非常重要的一环。最好能够将功能点和流程在冒烟测试时要的预期结果和开发人员说明清楚。

2、预计冒烟测试的最短和最大时间

* + - 根据列出来的功能点和开发人员以往提交测试人员代码质量的可信度，评估下冒烟测试在不同环境下可能花费的最大时间和最小时间，然后列到测试计划中。

3、冒烟测试数据的准备

* + - 在前期对主要功能对应表的结构都了解地很透彻，需要准备的数据及时准备好。真正冒烟测试开始后，就不会因为准备数据或者了解表存储结构而浪费时间。

二、冒烟测试的执行工作

* + 测试工程师严格按照前期的约定去校验主流程，全部校验和开发人员报告情况

三、冒烟测试的总结工作

1、时间：冒烟测试是否按时完成？有延误的话，要分析这段时间是不是会对后面正式测试的时间有影响。如果影响比较大可以给开发提建议，看后期有什么补救的方法可以既保证了质量又保证了按时上线，比如提高开发人员修复BUG的效率，测试时间顺延等。

2、问题：分析冒烟测试中发现的问题，和开发人员强调这个影响主流程的问题在冒烟修复验证通过后，不能在正式测试中再次出现，否则加大测试人员重复验证的工作量，影响测试进度。

对于一个小项目，也许冒烟测试只是花费2，3个小时就结束了，但是冒烟测试是麻雀虽小，五脏六腑全有。从前期确认主要功能，到最后的总结报告，我觉得每个流程都不能马虎，只有都准备好了，才能真正意义达到“冒烟测试”意义：仅用一袋烟功夫完成测试。

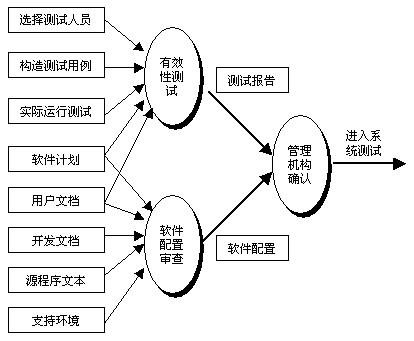
* 回归测试
  + 在发生修改之后重新测试先前的测试以保证修改的正确性。理论上，对软件的任何新版本，都需要进行回归测试，验证以前发现和修复的错误是否在新软件版本上再现。
* 回归测试的风险
* 一个很小的改动可能引起大问题
* 回归测试因为时间等原因不能讲系统全部重新测试
* 基于风险的测试方法
  + 评估系统不同部分蕴含的风险，测试最高风险的部分
  + 该方法系统的某些部分缺乏充分的测试，甚至完全不测，但是风险是最低的。
  + 分析可能性和影响
    - 可能出错的机会
    - 出错后造成的影响程度
* 功能测试和性能测试
  + 功能测试(Functional testing)，根据产品特征、操作描述和用户方案，测试一个产品的特性和可操作行为以确定它们满足设计需求
  + 性能测试（Performance testing），评价一个产品或组件与性能需求是否符合的测试。包括负载测试、强度测试、数据库容量测试、基准测试等类型
* 压力测试和负载测试
  + 压力测试：获取系统正确运行的极限，检查系统在瞬间峰值负荷下正确执行的能力。
  + 负载测试：用于检查系统在使用大量数据的时候正确工作的能力，即检验系统的能力最高能达到什么程度。

* 易用性测试
  + 主要从使用的合理性和方便性等角度对软件系统进行检查，发现人为因素或使用上的问题。在保证足够详细的程度下，用户界面要便于使用，对输入的响应时间和响应方式合理，输出有意义、正确，出错信息能够引导用户去解决问题，文档全面、确切等。易用性测试多数情况下没有一个量化的指标，主观性较强。
* 安装测试
  + 安装测试（Installing testing），确保该软件在正常情况和异常情况的不同条件下 能够正确安装
* 界面测试
  + 指测试用户界面的风格是否满足客户要求，文字是否正确，页面是否美观，文字，图片组合是否完美，操作是否友好等等
* 配置测试
  + 配置测试(Configuration Testing)主要检查计算机系统内各个设备或各种资源之间的相互连接和功能分配中的错误
* 文档测试
  + 主要检查文档的正确性、完备性和可理解性
* 兼容性测试
  + 测试软件是否和系统的其它与之交互的元素之间兼容
* 安全性测试
  + 检查系统对非法浸入的防范能力，检查系统中已经存在的系统安全性、保密性措施是否发挥作用，有无漏洞
* 恢复测试
  + 主要检查系统的容错能力
* 可移植性测试
  + 测试软件是否可以被成功移植到指定的硬件或软件平台上。
* 引导测试
  + 软件开发中，验证系统在真实硬件和客户基础上处理典型操作的能力
* 随机测试
  + 猴子测试
  + 没有书面测试用例、记录期望结果、检查列表、脚本或指令的测试。主要是根据测试者的经验对软件进行功能和性能抽查
  + 理论上，每一个被测软件版本都需要执行随机测试，尤其对于最后的将要发布的版本更要重视随机测试。随机测试最好由具有丰富测试经验的熟悉被测软件的测试人员进行测试。对于被测试的软件越熟悉，执行随机测试越容易
* 健全测试
  + 软件主要功能成分的简单测试以保证它是否能进行基本的测试
  + 典型地是指一个初始化的测试工作，以决定一个新的软件版本测试是否足以执行下一步大的测试努力。例如，如果一个新版软件每5分钟与系统冲突，使系统陷于泥潭，说明该软件不够“健全”，目前不具备进一步测试的条件。
  + 强度测试
  + 检验系统的能力最高实际限度。进行强度测试时，让系统的运行处于资源的异常数量、异常频率和异常批量的条件下。例如，如果正常的中断平均频率为每秒一到二次，强度测试设计为每秒10次中断。又如某系统正常运行 可支持10个终端并行工作，强度测试则检验15个终端并行工作
* 手工和自动测试
  + 手工测试是采用手工的方式执行测试；
  + 自动化测试（Automated Testing），使用自动化测试工具来进行测试，这类测试一般不需要人干预，通常在GUI、性能等测试中用得较多。
* 国际化测试、本地化能力测试、本地化测试
  + 国际化测试（International testing），国际化测试的目的是测试软件的国际化支持能力，发现软件的国际化的潜在问题，保证软件在世界不同区域中都能正常运行。
  + 本地化能力测试 (Localizability testing) ，本地化能力是指不需要重新设计或修改代码，将程序的用户界面翻译成任何目标语言的能力。为了降低本地化能力测试的成本，提高测试效率，本地化能力侧是通常在软件的伪本地化版本上进行
  + 本地化测试（Localization testing），本地化测试的对象是软件的本地化版本。本地化测试的环境是在本地化的操作系统上安装本地化的软件。从测试方法上可以分为基本功能测试，安装/卸载测试，当地区域的软硬件兼容性测试。测试的内容主要包括软件本地化后的界面布局和软件翻译的语言质量，包含软件、文档和联机帮助等部分
* 通过测试(正面测试)
  + 通过测试主要用于验证系统和它陈述的需求一致，确认软件至少能做什么，一般通过分析规格需求说明书来设计测试用例。
* 失败测试(反面测试)
  + 为了破坏软件而设计和执行的测试案例称为失败测试或迫使出错测试。
  + 失败测试用于证明“一个系统不会做不需要它做的事情”，同时也保证了系统的容错性。

* 错误猜测
  + 错误猜测是基于经验和直觉推测程序中所有可能存在的各种错误，从而有针对性地设计测试用例的方法。
* 需要测试人员具备的技术：
  + 掌握被测系统的知识，如设计方法或实现技术
  + 有关的早期测试阶段的结果的知识
  + 测试类似或相关系统的经验
  + 典型错误的知识
  + 通用的测试经验规则

**2、按照测试阶段分类**

* 单元测试
  + 单元测试（Unit testing）：最小模块的测试，可以是一个函数或子程序，一般由开发者在系统开发过程中进行执行。单元测试针对每一个程序模块进行正确性检验，检查各个程序模块是否正确地实现了规定的功能。单元测试是测试的第一步，起依据是详细设计，单元测试应对模块内所有重要的控制路径设计测试用例，以便发现模块内部的错误。
  + 多个模块可以平行地独立进行单元测试。
* 集成测试
  + 集成测试（Integration testing），被测试系统的所有组件都集成在一起，找出被测试系统组件之间关系和接口中的错误。该测试一般在单元测试之后进行
  + 集成测试也叫组装测试或联合测试、部件测试，在单位测试的基础上，将所有模块按照设计要求组装成为子系统或系统，进行集成测试。一些模块虽然能够单独地工作，但并不能保证连接起来也能正常的工作。程序在某些局部反映不出来的问题，在全局上很可能暴露出来，影响功能的实现。
  + 集成测试简单的表现形式就是每日构建（Daily Build）, 保证Build构建成功，也就是保证软件的组件或单元能组装成一个系统。每日构建是一个很好的实践
* 集成测试的原则
  + 所有公共接口都要被测试到
  + 关键模块必须进行充分的测试
  + 集成测试应当按一定的层次进行
  + 集成测试的策略选择应当综合考虑质量、成本、进度的关系
  + 集成测试应当尽早开始，并以总体设计为基础
  + 在模块与接口的划分上，测试人员应当和开发人员进行充分的沟通
  + 当接口发生修改时，涉及的相关接口必须进行再测试
  + 测试执行结果应当如实记录
* 集成测试
  + 非增式集成
  + 增式集成
    - 自顶向下增式测试
    - 自底向上增式测试（最常使用的方法 ）
* 确认测试
  + 着重考虑软件是否满足合同规定的所有功能和性能，文档资料是否完整、准确人机界面和其他方面（例如，可移植性、兼容性、错误恢复能力和可维护性等）是否令用户满意。
  + 保证软件配置的所有成分齐全，质量都符合要求。应该遵守用户手册和操作手册中的规定步骤。



* 系统测试
  + 是将通过确认测试的软件，作为整个基于计算机系统的一个元素，与计算机硬件、外设、某些支持软件、数据和人员等其他系统元素结合在一起，在世纪运行环境下，对计算机系统进行全面的功能覆盖。
  + 目的在于与系统需求比较，发现问题
* 验收测试 （ UAT）
  + Alpha testing (α测试,非正式验收 ),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受控测试，Alpha测试不能由程序员或测试员完成。
  + Beta testing(β测试),测试是软件的多个用户在一个或多个用户的实际使用环境下进行的测试。开发者通常不在测试现场，Beta测试不能由程序员或测试员完成。
  + 验收测试分类
    - alpha测试，beta测试
    - 正式验收测试，alpha测试（非正式验收 ），beta测试

按照阶段：

**3、按照测试实施组织划分**

* 开发方测试

 通常也叫“验证测试”或“α测试”。开发方通过检测和提供客观证据，证实软件的实现是否满足规定的需求。验证测试是在软件开发环境下，由开发者检测与证实软件的实现是否满足软件设计院说明或软件需求说明的要求。主要是指在软件开发完成以后，开发方对要提交的软件进行全面的自我检查与验证，可以和软件的“系统测试”一并进行。

* 用户测试

 在用户的应用环境下，用户通过运行和使用软件，检测与核实软件实现是否符合自己预期的要求。通常情况用户测试不是指用户的“验收测试”。而是指用户的使用性测试，由用户找出软件的应用过程中发现的软件缺陷与问题，并对使用质量进行评价。  
 β测试通常是被看用一种“用户测试”。β测试主要是把软件产品有计划地免费分发到目标市场，让用户大量使用，并评价、检查软件。通过用户各种方式的大量使用，来发现软件存在的问题与错误，把信息反馈给开发者修改。β测试中厂商获取的信息，可以有助于软件产品的成功发布。

* 第三方测试
  + 介于软件开发方和用户方之间的测试组织的测试。第三方测试也称为独立测试。软件质量工程强调开展独立和确认活动。软件第三方测试也就是由在技术、管理和财务上与开发方和用户方相对独立的组织进行的软件测试。一般情况下是在模拟用户真实应用环境下，进行软件测试。

**4、按照软件质量特性分类**

* 根据软件质量模型的软件特性来划分，软件测试可以划分为：功能测试、性能测试、可靠性测试、易用性测试、可移植性测试、可维护性测试以及近几年来因为网络软件安全问题而引起人们重视的软件安全测试。

GB16260标准 GB25000 GB15532