一、答疑篇

1. 软件测试的概念是什么?

验证软件功能是否满足用户的需求

1）找bug

2）验证程序是正确的

2. 软件测试和研发的区别：

说到软件测试和研发一般人容易混淆软件测试和调试，先说一下测试和调试：

测试和调试的区别

- 目的不同：测试的任务是发现程序中的缺陷；而调试的任务是定位并且解决程序中的问题。
- 参与的角色不同：测试主要由测试人员和研发人员共同完成，其中黑盒测试由测试人员完成，单元/集成测试主要是由开发人员执行；调试由开发人员完成。
- 执行的阶段不同：测试贯穿整个软件开发生命周期；调试一般在开发阶段。

下面再来讲测试和研发的区别

测试和研发的区别

- 难易程度：开发的广度小，专业度高。测试广度大，专业度低。
- 繁忙程度：测试与研发相比能轻松些，但敏捷模式下差距不大，而且测试一般在产品发布前压力比较大。
- 发展前景：自动化测试、安全测试等领域发展前景和研发基本一致。
- 技能要求：测试的技能要求更广泛，包括：业务能力，设计和架构分析能力，测试手段和工具使用，用户模型分析和理解，编程能力。
- 薪水：中小型企业总体比研发低，自动化等专业领域和研发无差别。大厂研发测试基本无差别。

3. 一个优秀的测试人员所具备的素质（你为什么要选择测试）

优秀素质 + 自身经历

思维模式：

- 逆向思维：
- 发散性思维：对于一个问题喜欢去思考是否还有其他解题思路

兴趣：

- 自己是否是真的对测试感兴趣而不是只是嫌弃开发加班多

性格特征：

- 好奇心：喜欢思考这种情况既然能满足，其他情况是否依然成立，并且经常想是否存在与当前情况类似的其他情况
- 不浮躁：在大学生活中我又发现自己的闪光点，比较仔细认真，擅长处理一些细节的工作，这在我大学的社团活动中负责的工作方面也有体现；还有平时学习时，能专心学习

能力：

- 快速学习能力和自学能力：高三暑期的暑假工经历，得到带我的姐姐的夸奖；初入大学学习办公软件、平时遇到不懂的问题先去找度娘
- 沟通表达能力：与学长学姐沟通，部长放心，当上副部。大二作品数量上升。

责任感与压力：

二、概念篇

1. 什么是需求

满足用户期望或正式规定文档(合同、标准、规范)所具有的条件和权能，包含用户需求和软件需求。

解释：用户需求和软件需求的关系

> 用户需求是来源，软件需求是用户需求的细化。
>
> 软件需求是根据用户需求转化而来的。

2. 什么是 bug(软件缺陷的概念)

- 当且仅当，需求规格说明书存在且正确的时候，程序与规格说明书不匹配；
- 在没有规格说明书的时候，与用户正确的需求不匹配的时候。

3. 测试用例的概念(测试用例包括的要素)

测试用例的概念：向被测程序输入的一组集合。

集合包括：测试环境、测试数据、测试步骤、测试版本以及预期结果。

4. 软件开发的生命周期（在软件开发的五个模型上应用）

需求分析、计划、设计、编码、测试、运行维护

5. 软件开发的五个模型

瀑布模型

- 优点：强调开发阶段；强调早期计划及需求调查；强调产品测试。
- 缺点：依赖于早期进行的唯一一次需求调查，并不适应需求的变化。

螺旋模型

- 优点：强调严格的全过程风险管理；强调各开发阶段的质量；提供机会检讨项目是否有价值继续下去。
- 引入了非常严格的风险识别、风险分析和风险控制，这对风险管理的技能水平提出了很高的要求，需要更多人员、资金还有时间的投入。

增量模型

某个项目的功能要求可分为 ABCD 四个模型，可以先开发AB模块，针对AB进行测试，测试通过后，继续开发CD模块，然后对CD模块进行测试，然后完成后续的测试。

增量开发能显著降低项目风险，结合软件持续构建机制。

迭代模型

在完成某个项目的时候，先完成整个项目中各个功能的一部分，对这些功能的这些部分进行测试，然后继续完善各个功能，再次进行测试。

迭代模型相较于增量模型抗击风险能力更前进些。

敏捷模型

（1）敏捷宣言

> 个体与交互重于过程和工具
>
> 可用的软件重于完备的文档
>
> 客户协作重于合同谈判
>
> 响应变化重于遵循计划

（2）scrum角色

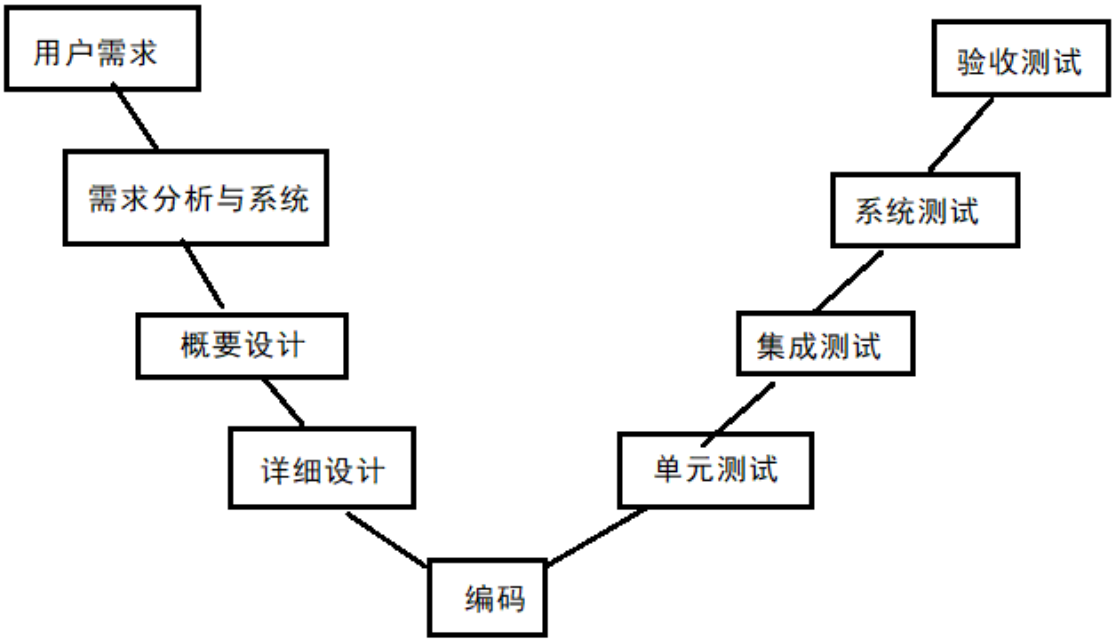
- PO(product owner)：产品经理，负责整理user story，定义其商业价值，对其进行排序，指定发布计划，和客户沟通。
- SM(scrum master)：项目经理，负责召开各种会议，协调项目，为研发团队服务。
- ST(scrum team)：研发团队，有不同技能的成员组成，通过协同完成每一次迭代的目标，交付产品。

（3）scrum 的工作流程：

- po(产品经理)负责整理 user story，形成左侧的 product backlog
- 发布计划会议：po 讲解 user story，对其进行估算和排序，发布计划会议的产出就是制定这一期迭代要完成的story列表，sprint backlog
- 迭代计划会议：项目团队对每一个 story 进行任务分解，分解的标准就是完成该 story 的所有任务，每个任务都有明确的负责人，并完成工时的初估计。
- 每日例会：每天SC召集站会，团队成员回答昨天做了什么今天计划做什么，有什么问题
- 演示会议：迭代结束后，召开演示会议，相关人员都受邀参加，团队负责向大家展示本次迭代取得的成果。期间把大家的反馈记录下来，由po整理，形成新的story。
- 回顾会议：项目团队对本次迭代进行总结，发现不足，指定改进计划，下一次迭代继续改进，以达到持续改进的效果。

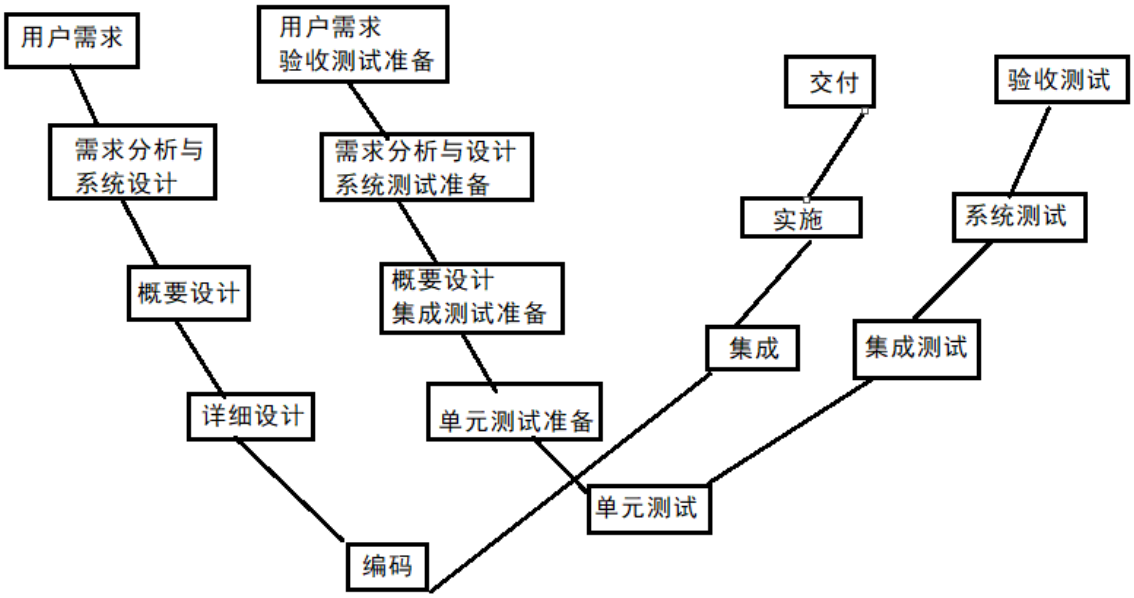
6. 软件测试的两个模型

V 模型



- V模型特点：明确的标注了测试过程中存在的不同类型的测试，并且清楚的描述了这些测试阶段和开发过程期间各阶段的对应关系。
- 局限性：把测试作为在编码之后进行的额一个阶段，未在需求阶段就进行测试，造成测试不重要的假象，需求一旦发生变化就会造成很大的浪费。

W 模型



- W 模型的特点：测试的对象不仅是程序，需求、设计等同样要测试，测试和开发是同步进行的；有利于尽早地全面地发现问题
- W 模型地局限性：需求、设计、编码等活动被视为串行地；测试和开发活动也保持着一种线性地前后关系，上一阶段完全结束，下一阶段才可正式地开始工作；无法支持敏捷开发模型；对于当前软件开发复杂多变地情况，W模型并不能解除测试管理面临着地困惑。

三、基础篇

1. 软件测试的生命周期：

需求分析 -> 测试计划 -> 测试设计、测试开发 -> 测试执行 -> 测试评估

2. 描述 bug 的五大要素

- 出现问题的版本：开发人员需要知道出现问题地版本，才能获取对应版本的代码来重现故障。并且版本的标识也有利于统计和分析每个版本的质量。
- 问题出现的环境：环境分为硬件环境和软件环境，如果是 web 项目，需要描述浏览器版本，客户机操作系统等，如果是app项目，需要描述机型、分辨率、操作系统版本等。详细的环境描述有利于故障的定位。
- 操作步骤：描述问题重现的最短步骤。
- 实际结果：描述错误的现象，crash等可以上传log，UI问题可以由截图。
- 预期结果：要让开发人员知道怎么样才是正确的，尤其要以用户的角度来描述程序的行为是怎样的。

案例：

> 故障版本：VPS20180226_01
>
> 故障类别：兼容性
>
> 故障优先级：中
>
> 故障标题：ie 下界面显示异常，界面文字有重叠
>
> 故障描述
>
> 测试环境：win7 + IE8
>
> 测试步骤：1、打开vps首页，点击“通知”连接，进入通知页面
>
> 预期结果：通知页面显示正常，一页显示 10 条通知，按时间倒序排列
>
> 实际结果：页面显示 10 条通知，通知顺序正确，但是页面文字有重叠
>
> 附件：上传截图

3. bug 的级别

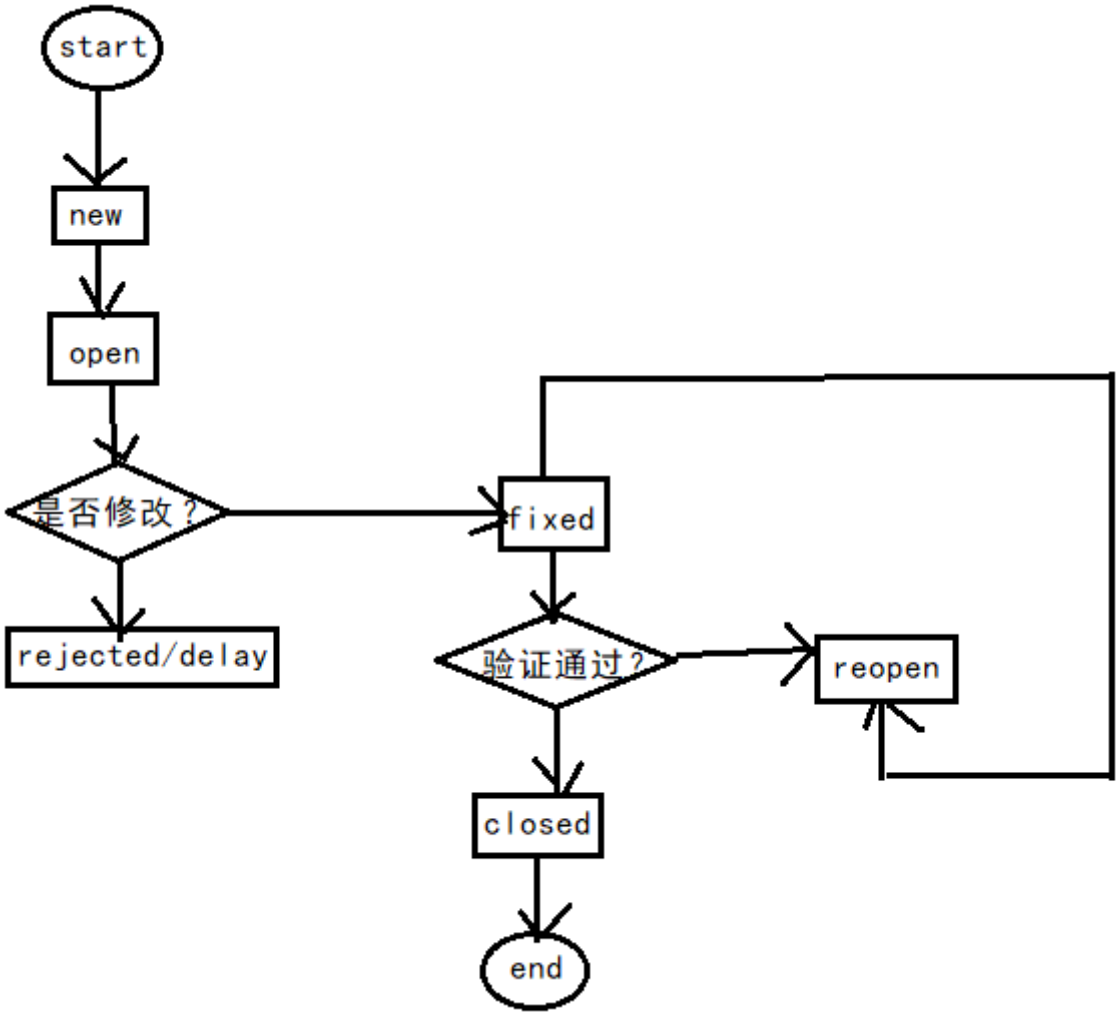
- 崩溃：阻碍开发或测试的问题；造成系统崩溃、死机、死循环，导致数据库丢失，与数据库连接错误，主要功能丧失，基本模块缺失等问题。
- 严重：系统主要功能部分丧失、数据库保存调用错误、用户数据丢失，一级功能菜单不能使用但是不影响其他功能的测试。功能设计与需求严重不符，模块无法启动或调用，程序重启、自动退出，关联程序间调用冲突，安全问题、稳定性等。
- 一般：功能没有完全实现但是不影响使用，功能菜单存在缺陷但不会影响系统稳定性。
- 建议：界面、性能缺陷、建议类问题，不影响操作功能的执行，可以优化性能的方案等。

4. 软件缺陷的生命周期

- New：新发现的 bug，未经评审是否指派给开发人员进行修改。
- Open：确认是 bug，并且认为需要进行修改，指派给相应的开发人员。
- Fixed：开发人员确认是 bug，并修改后标识为修改状态，有待测试人员的回归测试验证。
- Rejected：开发人员认为不是bug，拒绝修改。

- Delay：开发人员判定该 bug为暂时不需要或暂时不能修改，延后修改，或在下一次迭代进行修改。
- Closed：修改状态的 bug 经测试人员的回归测试验证通过，则关闭 bug。
- Reopen：如果经验证 bug 仍然存在，则需要重新打开 bug，开发人员重新修改。

bug 在测试人员和开发人员之间流转的状态



5. 因为bug 和开发人员发生争执怎么办

- 先自己检查自身，是否 bug 描述不清楚。
- 站在用户角度考虑问题，应该让开发人员了解到 bug 对用户可能造成的困扰，这样才能促使开发人员更加积极地、高质量地修改 bug。
- bug 定级有理有据
- 提高自身的技术和业务水平，不光要提出问题，最好也能直接提出 bug 的解决方案
- 开发人员还是不接受时，不要争吵，可以发起bug 第三方评审

四、用例篇

1. 基于需求的设计：

重点关注一下两个问题：

- 验证需求是否正确、完整、无二义性，并且逻辑一致。

- 要从黑盒的角度，设计出充分并且必要的测试集，以保证设计和代码都能完全符合需求。

2. 具体设计测试用例的方法（黑盒测试设计测试用例的方法）

等价类

> 设计理念：依据需求将**输入**（特殊情况下会考虑输出）划分为若干个等价类，从等价类中选出一个测试用例。如果这个测试用例通过，则认为所代表的等价类测试通过，这样就可以用比较少的测试用例达到尽量多的功能覆盖，可以解决不能用穷举测试的问题。

```
- 有效等价类：对于程序的规格说明书是合理的、有意义的输入数据构成的集合，利用有效等价类验证程序是否实现了规格说明中所规定的功能和性能。
- 无效等价类：根据需求的说明书，不满足需求的集合。
- 在测试时，有效等价类和无效等价类都要进行测试。
- 适用场景：输入、输入无穷
```

边界值

> 设计理念：边界值分析法就是对输入或输出的边界值进行测试的一种黑盒测试的方法。

- 取值规则：
  - 开区间：向内取值
  - 闭区间：向外取值
- 边界值编写测试用例的方法一般与等价类编写测试用例的方法结合起来使用。

因果图

> 设计理念：是一种简化了的逻辑图，能直观地表明程序输入条件(原因)和输出动作(结果)之间地相互关系。因果图法时借助图形来设计测试用例地一种系统方法，适用于被测试程序具有多种输入条件、程序地输出又特别依赖于输入条件的各种情况。

- 常用的关系：恒等、与、或、非
- 设计步骤：
  - 分析出所有可能的输入和可能的输出
  - 找出输入和输出之间的对应关系
  - 画出因果图
  - 把因果图转换成判定表
  - 把判定表对应到每一个测试用例

正交表设计法：

> 设计理念：正交表是研究**多因素、多水平**的一种设计方法，他是根据正交性，由试验因素的全部水平组合中挑选出部分由代表的点进行实验，通过对这部分实验结果的分析了解全面实验的情况，找出最优的水平组合。 有代表性的点

- 两个概念：
  - 因素：凡是需要考虑的变量称为因素(变量)
  - 水平：在实验范围内，因素被考察的值称为水平(变量的取值)
- 正交表的组成
  - 行数：正交表中行的个数，即实验次数，用 N 表示。
  - 因素数：需要考虑的变量的个数，用 C 表示。
  - 水平数：任何单个因素能够取得的值的最大个数，用 T 表示。
  - 正交表的表示形式：L = N(TC) = 行数(水平数 因素数)，其中 N = C * (T - 1) + 1
- 正交表的性质：

    - 每一列中各数字出现的次数都一样多
    - 任何两列所构成的各有序数对出现的次数都一样多
- 设计步骤：
    - 有哪些因素(变量)
    - 每个因素有哪几个水平(变量的取值)
    - 选择一个合适的正交表
    - 把变量的值映射到表中
    - 把每一行的各因素水平的组合作为一个测试用例
    - 加上你认为可疑且没有在表中出现的用例组合。

场景设计法

设计理念：现在的软件几乎都是用事件触发来控制流程的，事件触发时的情景便形成了场景，而同一事件不同的触发顺序和处理结果就形成了事件流。

错误猜测法

设计理念：基于经验和直觉，找出程序中你认为可能出现的错误，有针对性地设计测试用例。

进阶篇

1. 测试金字塔和业务测试分析

- UI 界面层：测试客户端/用户端
    - 功能验证测试
    - 兼容性于用户测试
- 业务逻辑层：会接触到代码
    - 客户端模拟测试
    - 内外接口测试
    - SDK接口测试
- 数据处理曾：代码测试代码、需要有一定地代码开发能力
    - 单元测试
    - Code Review

金字塔越往下(UI -> 数据处理层)，越难，专业度要求越高，代码能力要求越高。

2. 测试技术分类 ——> 按开发阶段划分

单元测试：

单元测试是对软件组成单元进行测试，其每目的是检验软件基本组成单位地正确性。

- 测试阶段：编码后或者编码前，编码后也叫做TDD(软件驱动开发，研发人员根据测试用例写代码)
- 测试对象：软件设计的最小单位：模块，也叫做单元模块
- 测试方法：白盒测试 —— 对代码进行测试
- 测试内容：模块接口测试、局部数据结构测试、路径测试、错误处理测试、边界测试

    - 模块接口测试：模块之间的接口(模块之间传递的参数)，测试模块间传递符合要求的接口，还有测试模块间传递不符合要求的接口
    - 局部数据结构测试：对数据的作用域范围进行测试
    - 路径测试：if 语句的每个路径都要进行测试(符合条件和不符合条件的路径都要进行测试)，for 循环中，假如循环了 10 次，就得进行 10 次的循环测试，还有判断该循环是否进行了 10 次，10次是否全部被进行了覆盖
    - 错误处理测试：验证代码的健壮性，判断程序运行要是出错了，程序的处理结果
    - 边界测试：假如循环了 10 次，第一次循环的结果是多少，第 10 次循环的结果是什么
- 测试人员：白盒测试员 或 开发工程师
- 测试依据：代码和注释 + 详细设计文档

集成测试：

- 集成测试也称为联合测试、组装测试，将程序模块采用适当的集成策略组装起来，对系统的接口及集成后的功能进行正确性检测的测试工作。其主要目的是检查软件单位之间的接口是否正确。
- 测试阶段：单元测试之后
- 测试对象：模块间的接口
- 测试方法：黑盒测试 + 白盒测试
- 测试内容：模块间的数据传输、模块间的功能冲突、模块组装功能正确性、全局数据结构、单模块缺陷对系统的影响
- 测试人员：白盒测试工程师 / 开发工程师
- 测试依据：单元测试后的模块 + 概要设计文档

系统测试：

- 将软件系统看成是一个系统的测试。包括对功能、性能以及软件所运行的软硬件环境进行测试。包括回归测试 和 冒烟测试。
- 测试阶段：集成测试之后
- 测试对象：整个系统
- 测试内容：功能、界面、可靠性、易用性、性能、兼容性、安全性等
- 测试人员：黑盒测试工程师
- 测试依据：需求规格说明文档
- 先 冒烟测试 —— > 系统测试 ——> 回归测试
    - 冒烟测试：概念来源于硬件，在软件中，冒烟测试是对核心主干流程进行测试。通过冒烟测试判断测试人员是否接受本次测试
    - 回归测试：回归测试是指修改了旧代码后，重新进行测试以确定修改没有引入新的错误或导致其他代码产生错误。回归测试的范围，跟测试的阶段相关(只回归 bug 或者跟 bug 相关的一些功能测试点，在最后的测试阶段不仅要回归 bug 还要回归整个系统)。 —— 往往采用自动化测试

验收测试：

- 验收测试是技术测试的最后一个阶段。目的是确保软件准备就绪，按照项目合同、任务书的验收依据文档，向加访展示该软件系统满足原始需求
- 测试阶段：系统测试通过后
- 测试对象：整个系统
- 测试人员：主要是最终用户 或 需求方
- 测试内容：同系统测试一样(功能、界面、可靠性、易用性、性能、兼容性、安全性等)
- 测试方法：黑盒测试
- 测试依据：用户需求、验收标准

3. 测试技术分类 ——> 按照实施组织划分

α 测试(阿尔法)

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时机操作环境下进行的。

β 测试(贝塔)

Beta 测试是一种验收测试。Beta 测试由软件的最终用户们在一个或多个场所进行测试。

第三方测试

介于开发和用户方间的

α 测试 和 β 测试的区别

- 测试的场所不同：Alpha 测试是指把用户请到开发方的场所来测试，β 测试是指在一个或多个用户的场所进行的测试
- α 测试的环境是受开发方控制的，用户的数量相对比较少，时间比较集中。β 测试的环境是不受开发方控制的，用户的数量比较多，且时间不集中
- Alpha 测试 先于 Beta 测试执行

4. 测试技术分类 —— 按照代码是否运行划分

静态测试：文档 + 代码

静态测试是指不运行被测程序本身，进通过分析或检查程序的语法、结构、过程、接口等来检查程序的正确性。对需求规格说明书、软件设计说明书、源程序做结构分析、流程图分析、符号执行来找错。

动态测试：代码

动态测试方法是指通过运行被测程序，检查运行结果于预期结果的差异，并分析运行效率、正确性和健壮性等性能。这种方法由三部分组成：构造测试用例、执行程序、分析程序的输出结果。

5. 测试技术分类 ——> 按照是否手工划分

手工测试

手工测试就是传统测试，由人一个一个的输入用例，然后观察结果，和机器测试相对应，属于原始但是必须的一个步骤。

自动化测试

- 自动化测试就是在预设条件下运行系统或应用程序，评估运行结果，预先条件应包括正常条件和异常条件。简单来说自动化测试就是把以人为驱动的测试行为转化为机器执行的一个过程。
- 自动化测试步骤：
    - 完成功能测试，版本基本稳定
    - 根据项目特性，选择适合项目的自动化工具，并搭建环境
    - 提取手工测试的测试用例转化为自动化测试的用例
    - 通过工具、代码实现自动化的构造输入，自动检测输出结果是否符合预期
    - 生成自动化测试报告
    - 持续改进，脚本优化

6. 测试技术分类 ——> 按照是否查看代码划分

黑盒测试

黑盒测试就是功能测试，不看软件的代码，只在对软件进行一系列操作之后，看软件是否能达到自己期望的结果。例如，测试淘宝的付款功能，点击付款看是否能够进入付款页面。

白盒测试

白盒测试就是对代码进行测试，测试代码里边的接口、数据结构、错误处理、路径覆盖、边界、业务逻辑等。接口测试是白盒测试的一种，但白盒测试不是接口测试。

白盒测试的方法：语句覆盖、条件覆盖、路径覆盖法、逻辑覆盖法、循环覆盖法、边界值、接口

灰盒测试

灰盒测试是介于白盒测试与黑盒测试之间的一种测试，灰盒测试多用于集成测试，不仅关注输出、输入的正确性，同时也关注程序内部的情况。

7. 测试技术分类 ——> 按照地域划分

本地化测试

国际化测试

软件的国际化和软件的本地化是开发面向全球不同地区用户使用的软件系统的两个过程。而本地化测试和国际化测试则是针对这类软件产品进行的测试。

8. 测试技术分类 ——> 按照测试对象划分

业务测试

测试人员把系统各个模块串接起来运行，模拟真实用户实际的工作流程，满足用户需求定义的功能来进行测试的过程。**场景法测试用例**

界面测试

- 测试用户界面的功能模块的布局是否合理
- 整体风格是否一致
- 各个控件的放置位置是否符合客户使用习惯
- 测试界面操作是否便捷性
- 界面中文字是否正确
- 命名是否同一
- 页面是否美观
- 文字图片组合是否完美等

容错性测试

- 输入不符合规格的数据，看系统如何处理：与无效等价类有关但不全是，是刻意地输入异常数据，看系统如何处理
- 灾难恢复性测试：自动启动(例如服务器挂了，自动启动备灾服务器)、人工干预(数据库内容出错，必须人工手动修改数据库内容)不让用户感知系统发生崩溃。

文档测试

开发文件共有14种文件，可分为3大类：开发文件、用户文件、管理文件

文档测试地关注点：

- 文档的术语
- 文档的正确性

- 文档的完整性
- 文档的一致性
- 文档的易用性

兼容性测试

- 兼容性主要是指软件之间能否很好的运作，会不会有影响、软件和硬件之间能否发挥很好的效率工作，会不会影响导致系统的崩溃。
- 兼容性测试的点：
    - 平台测试
    - 浏览器测试
    - 软件本身能否向前或者向后兼容(他之前的功能还能否正常使用)
    - 检测软件能否与其他相关的软件兼容
    - 数据兼容性测试(输入框输入的数据)

易用性测试

符合大部分用户的使用习惯即可

安装卸载测试

测试程序的安装、卸载

- 安装完成后是否会生成快捷方式
- 安装完成后的文件大小、个数是否完整
- 注册表

性能测试

- 资源利用的精确度量
- 响应时间
- 日志时间(如中断、报错)
- 吞吐量(TPS、处理订单数量)
- 辅助存储区(例如缓冲区、工作区的大小等)
- 处理精度

内存泄漏测试

如有内存管理、使用不合理，具体常见引起内存泄漏的有以下几点

- 分配完内存后忘记回收
- 程序写法有问题，造成没办法回收
- 某些 API 函数的使用不正确，造成内存泄漏
- 没有及时释放内存

五、工具篇