目录

[**软件测试的一些常识** 2](#_Toc48593536)

[**软件测试的一些方法** 2](#_Toc48593537)

[**白盒测试** 2](#_Toc48593538)

[**黑盒测试** 2](#_Toc48593539)

[**灰盒测试** 2](#_Toc48593540)

[**单元测试** 3](#_Toc48593541)

[**集成测试** 3](#_Toc48593542)

[**系统测试** 3](#_Toc48593543)

[**验收测试** 4](#_Toc48593544)

[**性能测试** 4](#_Toc48593545)

[**一般性能测试** 4](#_Toc48593546)

[**压力测试** 4](#_Toc48593547)

[**负载测试** 4](#_Toc48593548)

[**举例** 5](#_Toc48593549)

[**接口测试** 5](#_Toc48593550)

[**接口的组成有哪些?** 5](#_Toc48593551)

[**什么是接口测试？** 5](#_Toc48593552)

[**接口测试意义** 5](#_Toc48593553)

[**接口定义是否合理** 5](#_Toc48593554)

[**接口测试意义** 5](#_Toc48593555)

[**测试流程** 6](#_Toc48593556)

[**测试用例** 6](#_Toc48593557)

[**bug六要素、生命周期、状态** 6](#_Toc48593558)

[**测试报告核心要素** 7](#_Toc48593559)

[**面试中的一些场景题** 7](#_Toc48593560)

[**登录界面** 7](#_Toc48593561)

[**排查问题的思路** 7](#_Toc48593562)

[**http请求长时间没有结果** 8](#_Toc48593563)

[**有个用户反馈上传头像失败** 8](#_Toc48593564)

[**app闪退原因** 8](#_Toc48593565)

[**偶然闪退的排查** 8](#_Toc48593566)

[**网页卡顿** 8](#_Toc48593567)

[**10%的用户反馈用不了功能，你将如何排查？** 9](#_Toc48593568)

[**登录的按钮不能点击，如何排查问题？** 9](#_Toc48593569)

[**压测的时候，QPS一直上不去** 9](#_Toc48593570)

[**APP提示无法连接网络，你会如何排查？** 9](#_Toc48593571)

[**怎么判断一个BUG到底是前端的BUG还是后端的BUG？** 10](#_Toc48593572)

**软件测试的一些常识**

软件生命周期：可行性研究和需求分析->概要设计->详细设计->编码->单元测试->继承测试->系统测试->验收测试->软件维护。

**软件测试的一些方法**

**白盒测试**

语句覆盖

判定覆盖

条件覆盖

判定条件覆盖

条件组合覆盖

修正判定条件覆盖：

对于每一个条件C,C所在判定内的所有条件，其他条件的取值完全相同；当C的取值相反，判定的计算结果相反。

e.g.ifAand(BorC)return1

return0

ABC🡪判定结果

100🡪0110🡪1101🡪1001🡪0

路径覆盖

1）画出程序的数据流图，数据流图中含有两种节点，普通节点和判定节点。

2）圈复杂度=确保所有语句至少执行一次的测试数量的上界

=流图中区域的数量=流图中边的数量-节点数量+2=流图中判定节点的数量

3）得出独立路径，即为测试用例

独立路径：至少沿一条新的边移动的路径。

**黑盒测试**

等价类划分法：有效等价类和无效等价类

边界值分析法：最小值，第二最小值，中间值，第二最大值，最大值

因果图：找到原因和结果,画出原因和结果的图,逻辑包括与或非,依据因果图写出测试用例。

判定表：n个条件，把所有可能的情况全列出来，之后将相同的结果进行合并。

正交实验设计：

确定使用的正交表，表示共有k个条件，每个条件有m种取值(水平数)，n表示正交表测试样例的数目，。例如，表示有4个3水平列和1个2水平的列，则n=4\*(3-1)+(2-1)+1=10。

确定后，去选择合适的正交表就可以了，正交表的性质如下：

(1)每列中不同数字出现的次数是相等的，如，每列中不同的数字是1，2，3，它们各出现3次；

(2)在任意两列中，将同一行的两个数字看成有序数对时，每种数对出现的次数是相等的，如L9()，有序数对共有9个：(1，1)，(1，2)，(1，3)，(2，1)，(2，2)，(2，3)，(3，1)，(3，2)，(3，3)，它们各出现一次。

**灰盒测试**

[灰盒测试](https://www.leiue.com/tags/%e7%81%b0%e7%9b%92%e6%b5%8b%e8%af%95)，是介于[白盒测试](https://www.leiue.com/tags/%e7%99%bd%e7%9b%92%e6%b5%8b%e8%af%95)与[黑盒测试](https://www.leiue.com/tags/%e9%bb%91%e7%9b%92%e6%b5%8b%e8%af%95)之间的一种[测试](https://www.leiue.com/tags/%e6%b5%8b%e8%af%95)，灰盒测试多用于集成测试阶段，不仅关注输出、输入的正确性，同时也关注程序内部的情况。灰盒测试不像白盒那样详细、完整，但又比黑盒测试更关注程序的内部逻辑，**常常是通过一些表征性的现象、事件、标志来判断内部的运行状态。**

**单元测试**

针对软件设计的最小单位程序模块进行正确性检查的测试工作，一个模块、一个函数或者一个类。，单元测试需要从程序内部结构出发设计测试用例，多个模块可以平行地独立进行单元测试。

测试阶段：编码后

测试对象：最小模块

测试人员：白盒测试工程师或开发工程师

测试依据：代码和注释+详细设计文档

测试方法：白盒测试

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

单元测试要快，应该产生可重复的一致的结果。

单元测试应该集成到自动测试的框架中，这样每个人都很容易的运行它，单元测试的错误就能及时被发现并得到修改。

单元测试必须和产品代码一起保存和维护。

比如说在商店买东西时，需要扫码付款，如果使用支付宝付款，那么正常情况下就应该用支付宝去扫描商家的支付宝收钱码进行付款而不是微信收钱码。那么要是出现异常情况的话，比如用支付宝扫了微信收钱码还会弹出支付界面吗，又或者是在无网情况下扫码会出现什么界面呢？->进行错误处理测试

**集成测试**

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。集成主要目的是检查软件单位之间的接口是否正确。

测试阶段：一般单元测试之后进行

测试对象：模块间的接口

测试人员：白盒测试工程师或开发工程师

测试依据：单元测试的模块+概要设计文档

测试方法：黑盒测试与白盒测试相结合

测试内容：模块之间数据传输、模块之间功能冲突、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。

比如说逛淘宝后需要买东西，那么在提交订单后一般会弹出确认付款界面让我们付款，我们就需要测试在点击提交订单后是否会出现确认付款界面，如果出现了这个界面，那么是否可以支付成功，这个过程就相当于在测试淘宝提交订单这个接口与付款这个接口是否能够很好地配合工作。

**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。时间大部分在系统测试执行阶段，包括回归测试和冒烟测试。

测试阶段：集成测试通过之后

测试对象：整个系统（软、硬件）

测试人员：黑盒测试工程师

测试依据：需求规格说明文档

测试方法：黑盒测试

测试内容：功能、性能、安全、兼容、界面、易用性、稳定性。

集成测试和系统测试之间的比较：

1、测试内容：集成测试是测试各个单元模块之间的接口，系统测试是测试整个系统的功能和性能；

2、测试角度：集成测试偏重于技术的角度进行测试，系统测试是偏重于业务的角度进行测试。

**验收测试**

部署软件之前的最后一个测试操作，最后一个阶段，也称为交付测试。

目的是确保软件准备就绪，按照项目合同、任务书、双方约定的验收依据文档，向软件购买者展示该软件系统满足原始需求。

测试阶段：系统测试通过之后

测试对象：整个系统（包括软硬件）。

测试人员：主要是最终用户或者需求方。

测试依据：用户需求、验收标准

测试方法：黑盒测试测试

内容：同系统测试(功能...各类文档等)

这个阶段主要是依据用户的需求和验收标准来测的，由用户进行测试，测试所有的功能是否符合他们的预期等等

**性能测试**

**性能指标：**

一般包括响应时间(RT)、每秒钟的请求数(QPS)、每秒完成的事务数(TPS)、并发用户数(进行操作的用户，不包括读网站的用户)、吞吐量（单次业务中，客户端与服务器端进行的数据交互总量。受服务器端性能和网络性能影响）、吞吐率（吞吐量/时间）、CPU利用率(CPU忙时间/总时间)、CPU负载（正在使用和等待CPU进程的平均进程数）、内存的利用率（物理内存、虚拟内存）、MTBF（错误发生的时间间隔）、响应成功的比率、响应失败的比率、响应超时的比率。。。

QPSv.s.TPS一个事务可能会对应多个请求。

**一般性能测试**

在正常环境、不加压的条件下，检验系统是否符合性能指标。

**压力测试**

压力测试：模拟与真实环境相似的测试环境，考察在这一环境下，测试系统在高负载的情况下的健壮性，找到系统所能承受的最大负荷和系统的瓶颈点。压力测试尤其关注大业务量情况下长时间运行系统性能的变化（例如是否反应变慢、是否会内存泄漏导致系统逐渐崩溃、是否能恢复）；压力测试是测试系统的限制和故障恢复能力。但这个负载不一定是系统本身造成的。比如我们经常利用脚本或工具事先吃掉服务器的一部分cpu、内存或带宽等，创造出一定的负载环境并测试被测应用系统在此环境下的事物处理能力，响应时间等等。

**负载测试**

负载测试的目标是模拟真实的环境，从比较小的负载开始，逐渐增加模拟用户的数量（增加负载），观察不同负载下应用程、序响应时间、所耗资源，直到超时或关键资源耗尽（不关注稳定性，也就是说不关注长时间运行，只是得到不同负载下相关性能指标即可）。

**举例**

以一个饭馆的例子，把饭馆的响应时间定义为从客户点餐到吃到餐的时长，在饭馆的员工数目、座位等硬件条件下，饭馆建设时，预估能够同时服务100人。

负载测试就是模拟真实环境中的客户数目不断增长，响应时间、饭菜美味程度等指标的变化，直到人数达到100人。

而压力测试是想要饭馆能够同时服务的最大客户数目M（最大负荷）以及造成只能服务M个客户的瓶颈点，比如说是座位数目太少，那如果我们想要提高饭馆这个系统的健壮性的话，可以加一些座位。压力测试比较关注的是长时间同时服务100人运行，工作人员需要承受很大的压力，那这个时候响应时间、饭菜美味程度等指标的变化。或者说有人生病请假（cpu、内存、带宽等硬件资源降低），饭馆各指标的变化。

**接口测试**

用的比较多的是web的请求服务（http/https）。

抓包。

**接口的组成有哪些?**

一个完整的接口应该包含以下内容：1.接口说明2.调用的url3.请求方法（get\post）4.请求参数、参数类型、请求参数说明5.返回参数说明

接口至少应有请求地址、请求方法、请求参数（入参和出参），部分接口有请求头header。

header是服务器以HTTP协议传HTML资料到浏览器前所送出的字串，一般存放cookie、token等信息，header和入参是有区别的。header一般存放的是一些校验信息比如cookie，它是为了校验这个请求是否有权限请求服务器。如果有权限它才能请求服务器，然后把请求地址连同入参一起发送到服务器，然后服务器会根据地址和入参来返回出参。也就是说，服务器是先接受header信息进行判断该请求是否有权限请求，判断有权限后，才会接受请求地址和入参。

**什么是接口测试？**

1)测试系统组件之间接口，检测外部系统与系统之间以及系统内部各个子系统之间的交互。2）重点是要检查数据的交换，以及系统间的相互逻辑依赖关系等。

3）接口测试就是通过测试不同情况下的入参与之相应的出参信息来判断接口是否符合或满足相应的功能性、安全性要求。

**接口测试意义**

1）系统复杂度不断上升，使用接口测试跑一下系统中的所有接口，一个时间少人工成本，一个是让测试更早的介入项目，有助于更早的发现问题。

2）而且这样可以把系统的底层逻辑尽可能的全部覆盖到，可以发现更加底层的问题。

3）和功能测试相比，周期短，有助于检查服务器的异常处理功能。因为前端的验证很容易被绕过，功能测试很难发现安全问题，而接口测试更容易验证异常情况。

**接口定义是否合理**

1）接口是否冗余；2）接口字段是否冗余；3）接口是否返回了调用方期望得到的信息；

4）接口定义是否可满足所有调用需求、是否方便。

**接口测试意义**

1. 输入，按照参数类型进行用例设计
   1. 数值型

1）传入非特定类型，程序异常退出2）超长字符未进行处理，导致存储、显示异常

3）其他用户可见设置的敏感字

* 1. 字符串
  2. 数组或链表

1)0个item时程序异常退出2）重复的item处理时未去重导致结果异常

* 1. 类

1. 接口处理，按照逻辑进行用例设计
   1. 数值、分数、等级限制e.g.积分大于50才可参与
   2. 状态限制：登录状态等
   3. 权限限制：管理员等
2. 输出，按照输出结果进行用例设计

正确的结果可能只有一个，但是错误异常返回的结果有很多。

* 1. 前端处理错误不足，导致前端异常
  2. 错误提示处理不当，导致用户看到晦涩的错误码
  3. 错误提示不准确，导致用户不知道哪里出了问题，如何解决
  4. 正常情况下，接口有返回，不返回是由于接口的超时处理不当:
     1. 未进行超时处理，导致整个流程阻塞
     2. 超时后又收到接口返回，导致逻辑出现错乱

废弃的接口指因为需求变更或其他原因暂时不用，但可能代码并未及时删除。如果利用技术手段调用这些接口，可能产生风险。因此新版本需要考虑兼容旧版本的同时，还应做好相关废弃接口的检查。

常见风险就是，约束条件判断不足，导致用户可通过特殊手段获取利益。

**测试流程**

1. 需求分析阶段
   1. 需求分析：分局需求文档、产品原型、口述或参考市面上已经成熟的同类型的产品
   2. 学习业务流程，提取功能点，并编写需求分析说明书
2. 测试设计阶段
   1. 测试计划 时间和人员以及资源的安排分配
   2. 测试方案 1)针对每个测试内容如何开展测试2)采用什么测试计划3)什么测试工具
   3. 测试策略 1)哪些测试内容先测\后测2)开始测试和结束测试的标准是什么
   4. 测试用例
3. 测试执行阶段
   1. 预期结果和实际结果做对比，如果一样 则通过，如果不一样则有问题
   2. 有问题就提交BUG
   3. 回归测试 在版本2上去检查在版本1上发现的问题有没有被解决
4. 测试总结阶段🡪编写测试报告

**测试用例**

包括：编号名称重要级优先级前置条件测试数据测试步骤预期结果实际结果

**bug六要素、生命周期、状态**

**包括**：编号名称优先级(高中低)等级复现步骤(测试用例的步骤、预期结果、实际结果）附件(bug的佐证，图片、日志、视频等)

等级🡪四个：

致命的：1）影响产品的核心流程的正常使用2）导致软件挂了，闪退，崩溃3）和钱有关严重的：导致了功能无法正常使用

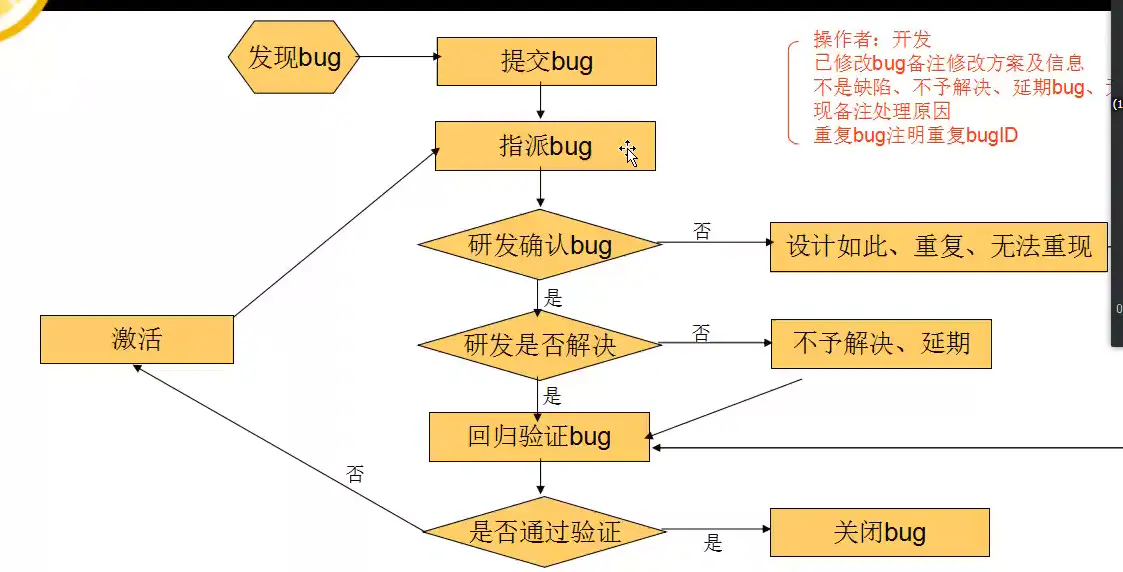
一般的：功能的某些异常场景有问题

轻微的：建议性的东西，用户体验的，UI上的问题

**生命周期**：

重复bug：1、测试人员找到对应重复bug的ID。如果确认是重复bug，直接关闭。

无法重现：1、确认开发的环境，跟操作步骤是否跟测试人员一致。

**状态：**newopen已确认已解决拒绝reopenclosed延期处理重复bug

**测试报告核心要素**

1. **测试结论**

从测试工程师的专业角度分析，是否达到发布标准，是否可发布。如果你的测试报告，结论都没有，那这份报告的意义是什么？

1. **风险**

已知风险&未知风险，抛出。

项目经理、产品经理等多部门，需要根据这份风险分析，确定最终这个版本是否发布出去。

1. **测试时间&测试人员**这是非常重要的，投入了哪些人，用了多少时间，测试起止时间。
2. **测试环境&测试设备**用到哪些测试手机，什么客户端环境，什么浏览器等等。
3. **需求大纲**当前的这个版本，到底包含了哪些大的需求点。
4. **数据分析（非常重要的一点）**

可以从多个维度分析bug，比如，测试、开发、软件模块、等级、解决时间、每个版本、bug状态、激活次数.

对被测软件的质量评估,比如一二级的BUG全部都关闭了，三级的BUG关闭80%+，四级的BUG无所谓

1. **测试总结**从测试角度，对这个版本，你觉得存在的一些问题，一些建议，等等。

**面试中的一些场景题**

**登录界面**

**排查问题的思路**

**http请求长时间没有结果**

1. 用户电脑问题，比如设置了代理，导致无法访问
2. 网络问题，网速较慢
3. 服务器未返回结果：由于默认是长连接，前端没有进行超时处理的话，会导致请求一致等待服务器响应，而服务器由于开启进程过多或受到攻击，导致响应很慢。
4. 服务器返回了结果，而前段组件出现问题，没有显示出结果

**有个用户反馈上传头像失败**

1. 图片本身：图片的尺寸、大小或格式不符合限制条件
2. 上传接口异常
   1. 上传图片的接口挂了
   2. 调用接口时，参数传递不正确
   3. 接口访问数据库时出现异常
3. 前端图片组件有bug

**app闪退原因**

1. 外部因素
   1. 手机内存不足或缓存垃圾过多
   2. 手机系统版本过低
   3. 手机中杀毒系统误判
   4. 与手机中其他软件冲突，发生资源竞争
   5. 网络不稳定
2. app自身因素
   1. app版本过低，与后端接口或新版操作系统不兼容
   2. 缺少数据包
   3. 内存泄漏
   4. 程序本身有bug：空指针异常、数组越界、堆栈溢出、逻辑错误等

**偶然闪退的排查**

1. 一般成熟的团队都会有crash的监控平台，可以从crash平台上去查看crash发生位点。
2. 手工尝试复现crash，一般偶然的闪退，都不会特别容易复现，可能需要适当施加一些比较苛刻的条件（弱网、断网、快速点击、快速划动等等）。
3. 查看crash日志，比如AndroidAPP可以用adb命令去查看：e.g.在mac下，执行Monkey或遍历测试，暴力操作手机，尝试复现bug。

**网页卡顿**

1. http请求次数太多

规范接口设计，减少http请求次数。

1. 接受数据时间过长，e.g.下载资源过大。

对HTTP传输进行压缩，可采用gzip无损压缩，压缩效果最佳。

1. JavaScript（js）脚本过大，阻塞了页面的加载

将JavaScript脚本放在标签前。script没有async和defer时，JS文件将在下载后立即执行。这种情况下，script放在顶部会阻塞页面呈现，在网速慢的情况下会导致“白屏”，直到脚本下载完毕才继续呈现页面。因此，script放在底部可以让页面尽快呈现。

1. CSS、js、图片等静态资源需要重复加载

静态资源统一放在一个静态域名上，减轻重复下载静态资源的负担。

1. cookie影响

去除没有必要的cookie，或设置合适的cookie过期时间。如果网页不需要cookie就完全禁掉。

1. 网页资源过多

使用CDN(内容分发网络)部署网络来加速，可以先通过免费的CDN供应商来分发网页资源。

**10%的用户反馈用不了功能，你将如何排查？**

1. 操作系统版本，可能是用户的操作系统过高或过低，没有做好兼容。
2. APP版本，可能是接口改动没有考虑版本控制，对低版本用户造成影响。
3. 灰度测试或AB测试，可能是功能缺陷导致对部分灰度用户产生影响。
4. 跟会员用户有关，可能是一些功能仅仅只对VIP会员开放，然而这部分功能有缺陷。
5. 跟用户分布地域有关，比如说：有些地区没有开放功能，但是也给这些用户展示了入口。

**登录的按钮不能点击，如何排查问题？**

1. 用户手机内存不够，导致页面卡死。
2. 大概率是前端问题：
   1. 前端没有响应用户点击事件，请求发不出去
   2. 网络异常，请求发不出去，前端没做提示
   3. http请求发出去了，后端接口返回异常，前端捕获异常后没有处理

**压测的时候，QPS一直上不去**

1. 被测服务器：看其资源是否被打满，导致无法连接。

给被测服务器扩容。

1. 接口：是否出现报错以及响应时间是否变慢。

接口性能优化。

1. 压测机器：是不是网络IO占满，并发数达不到。

多台压测机器并发。

1. 压测工具：是否支持并发请求。

采用多线程或协程的方式去并发请求。

**APP提示无法连接网络，你会如何排查？**

第一步：检查网络环境

1. 检查4G和Wifi是否可用，可以先看手机网络连接图标状态，有无信号，是否弱网，并且可以切换其他APP，测试网络是否可用。
2. 检查是否有网络限制，比如仅公司内网可用的APP，你在别的网络环境是无法连接的。
3. 检查是否使用了代理或代理连接是否出现异常，手机连接电脑代理之后，如果不安装证书，发起https的请求将出现异常。

第二步：检查APP的网络请求

1. 抓包，检查APP请求的域名是否正确
2. 抓包，检查后端接口是否响应超时
3. 抓包，检查后端接口是否返回异常，而APP没有做相关的异常提示。

**怎么判断一个BUG到底是前端的BUG还是后端的BUG？**

1. 样式和交互层面的Bug，大概率都是前端的Bug
2. 数据和文案相关的Bug，大概率都是后端的Bug拿

移动端来说，最简单但是又最实用的办法是对比测试，即Android和iOS对比测试，假如说Android和iOS都有问题，可能是后端Bug；假如说Android和iOS有一端有问题，则可能是Andorid或者iOS某一终端的Bug，但也不一定绝对，也有可能是后端的Bug。