第一讲widow基础

一、什么是软件（software）

计算机=硬件+软件

软件=程序+文档

提示：软件测试的对象是程序和文档（评审）

扩展：

1、软件开发的基本阶段划分

（1）需求分析阶段

由需求分析人员完成，产出物：《需求规格说明书》

（2）设计阶段

由系统架构师/分析师完成，产出物：《概要设计说明书》、《详细设计说明书》

（3）编码阶段

由开发人员完成，产出物：程序

**面试题：哪个阶段引入的bug最多？哪个其次？哪个最少？对于测试工作有什么印象？**

答：引入bug最多的是需求分析阶段 （大概占bug总数的55%左右）；

其次是设计阶段 （大概占比25%）；

最少的是编码阶段 （大概占15%）；

还有5%左右的bug是由配置错误或者兼容性问题引发的。

**由此得出结论：（1）需求和设计阶段也需要测试，不仅要测程序，文档也要测（评审）**

**（2）测试要符合“尽早测试原则”和“不断测试原则”**

**提示：测试越早介入解决bug付出的成本一般越低**

**二、什么是软件缺陷**

**缺陷—defect、bug**

**1**、缺陷定义1（5条）

说明：只要满足其中任意一条就可以定义为bug，缺陷定义与顺序无关

（1）需求要求的功能没有实现

（2）实现了需求没有要求的功能（画蛇添足）

（3）软件中出现了指明不应出现的错误

（4）需求虽未明确指明，但是应该实现的功能没有实现

说明：需求不是完美的，可能存在遗漏，测试人员如果发现需求的问题，也应提出

（5）软件不易使用、难以理解、运行缓慢等，站在用户的角度，一切不好的地方

**补充问题：软件的两个基本要素：**

1）软件的功能要能够正确实现

2）软件要有强大的异常处理能力—健壮性

2、缺陷定义二、

1）美国电气和电子工程师协会（IEEE)对缺陷的定义：

从产品内部看，缺陷是软件产品开发或维护过程中存在的错误、毛病等各种问题

从产品外部看，缺陷是系统所需要实现的某种功能的实效或者违背。

3、缺陷的同义词

错误—error

异常—exception

三、软件测试

测试—test

1、简单来说：软件测试就是从现有软件中，尽可能多的查找bug的过程

2、说明

（1）软件不是完美的，或多或少存在bug，所以测试人员的职责不是消灭bug，

而是尽可能多的查找bug

（2）软件测试强调查找bug的过程，只要完成查找bug的过程，无论是否找到bug，

都是在测试（一般发现bug的比例在10%—30%）

四、计算机的层次

1、基本层次

（1）计算机硬件（裸机）

（2）操作系统（os）

（3）应用软件（applocation）

补充：OS-operating system 操作系统

扩展问题：

Q1：裸机中有软件吗？（了解）

答：裸机中有软件，是安装在计算机主板的CMOS芯片中的“BIOS”

（basic input output system—基本输入输出系统）程序

BIOS程序的作用是“上电自检”—就是计算机通上电后，控制权首先由BIOS获取，

该程序对计算机硬件设备的连接情况和初始设置进行检查，当检查没有问题时，会启动OS，

并移交控制权；但是当检查发现一件连接出现问题时，会启动蜂鸣器发出警报音，并阻止OS

启动。  补充：如何进入BOIS？

台式机：开机后按delete（删除键）

笔记本：开机后，经常会按F2或F8键，如果不是查百度

Q2：什么是操作系统？操作系统的主要功能组成？

1）操作系统简称OS，是管理计算机硬件和软件的计算机程序

2）主要由以下5个部分组成

·进程管理

·设备管理

·存储管理

·文件管理

·作业管理（用户向操作系统提交的任何请求被看成作业）

Q3:常用的OS有哪些？（列举5种）

1. Windows（微软）

特点：简单、易用、可视化界面，pc（personal computer 个人电脑）占有率最高；但是安全和稳定性较差，导致windows系统在服务器OS领域占有率较低

1. Unix （1969年 贝尔实验室产品）

特点：安全性高、稳定性好、大量用于金融行业等安全性要求较高的软件系统中；付费系统、支持二次开发

1. Linux （自由软件）

·说明：Linux是类Unix系统

·特点：安全性、稳定性较好、免费、开源、支持二次开发、

在服务器OS领域占有率最高

提示：专业测试人员要求熟悉Linux系统，面试中经常会考Linux的常用命令

1. Mac （苹果公司 1981年）

·Mac系统是世界上第一款可视化的OS

特点：Mac系统界面设计美观、易用、对于图形、图形的处理能力优异，常应用于设计领域

5）dos系统 （微软公司）

特点：命令式、单机系统，测试人员可能会用到部分常用命令：

1. 如何打开dos

Win+r—打开运行--输入cmd--按回车打开dos界面

2）常用dos命令

Ipconfig 查看ip地址

Ping [www.baidu.com](http://www.baidu.com) 查看本机与目标的网络通信状态

Q4：常用的数据库有哪些？

说明：目前应用较为广泛的均为关系型数据库，数据库管理系统简称DBMS

1. oracles 数据库 （甲骨文公司）
2. MYSQL数据库 （被甲骨文公司收购）--低端、性价比高（便宜，有免费版本）
3. SqlServer 数据库（微软） 该数据库与其他公司软件产品的兼容性不佳，比较适用于微软的产品体系

补充：两款高端数据库

1. DB2 （IBM公司）
2. Sybase （sybase公司）

提示：所有的关系型数据库公用的数据库语言叫：标准SQL--结构化查询语言（重点考SQL）

1. 软件的分类
2. 基本软件
   * 1. 系统软件
3. 操作系统
4. 操作系统的补丁程序
5. 驱动程序
   * 1. 应用软件
        1. 开发工具软件
        2. 科学和工程计算软件
        3. 文字和数据处理软件
        4. 图形和图像处理软件
        5. 事物管理软件
        6. 游戏娱乐软件
        7. 支持服务工具
        8. 数据库管理系统
6. 按结构分类
7. 单机软件

不需要连接网络就可以应用的软件，例如：记事本

1. 分布式软件

需要连接网络才可以应用的软件，例如：QQ、微信、

注意：此网络既包括广域网也包括局域网

1. C/S 结构

C-client 客户端 s-server 服务器

1. B/S结构

B-browser 浏览器服务器 s-server 服务器

区别：C/S结构：需要安装专门的客户端程序，才能访问服务器，享受服务器提供的服务

B/S结构：不需要安装专门的客户端程序，只要有公共的浏览器，输入相应的网址，就可以享受不同服务器提供的服务

补充：B/S结构软件测试时需要进行“浏览器兼容性”测试，主流浏览器如下



微软新推出的浏览器：edge

**第二讲 缺陷报告**

1. **什么是缺陷报告**
2. 测试人员发现bug，通过缺陷报告将bug记录下来
3. 通过缺陷报告将bug告知给开发人员
4. 通过缺陷报告可以实现对bug的跟踪管理
5. 总之缺陷报告是测试人员和开发人员之间重要的沟通方式
6. 缺陷管理的常用工具

企业中通常使用缺陷管理工具来管理缺陷，常用的工具有：

Mantis（螳螂）、禅道（国产zentao）、jira（鸡爪）、testlink、QC（HP、付费）等还有的公司会自制bug管理工具

说明：不同的bug管理工具模版不完全相同，会有差别，但是毕竟都是对bug进行管理，缺陷报告的核心内容大同小异

1. 缺陷报告的核心组成

案例：两数相除，除法功能没实现

1. 缺陷编号（defect/bug ID）

提交bug的流水号能够唯一标识每个bug

1. 缺陷标题（summary）

要求：将所发现的bug，简明扼要的描述出来（可以自由发挥，意思正确即可）

1. 发现者/创建者（detected by）

测试人员发现bug，此处通常填写测试人员的真实姓名或账号 实际工作中，账号经常由bug管理工具的管理员创建

例如：测试人员账号

真实姓名：王伟

账号：wangwei\_qa

1. 日期（date）

Date-日期 data-数据

缺陷报告提交的日期

注意：发现bug应“及时”提交，bug要审核通过，认定有效bug才提交，这样尽量避免“假bug”被提交，保证bug的质量

1. 当前指派（assigned to）

常规：

测试人员➡️开发主管➡️具体负责的开发人员

其他情况：

例如：小公司

测试人员➡️具体的开发人员

例如：规模较大的公司

测试人员➡️测试主管➡️开发主管➡️具体的开发人员

1. 所属功能模块（subject）

表明在哪个功能模块中发现该bug，可以方便开发主管确认具体负责解决该bug的开发人员

1. 版本（release/version）

对于专业的测试人员来说：版本不仅仅是最终发布的版本，也包括在研发过程中曾经出现过的若干临时版本

回归测试（回测）：

1. 回归测试：就是在当前版本中，对上个版本测过的所有功能再重新测试一边
2. 回归测试的必须性：首先修改bug，可能在bug被修复的前提下，产生新的问题；其次如果有新增功能，那么新增加的功能可能会对原有功能带来影响，产生新的缺陷
3. 回归测试中存在重复执行测试的情况，如果条件允许使用自动化测试会提高测试效率
4. 状态 （status）

表明缺陷处于怎样的处理情况

常见的bug状态：

**新的--new**

**激活的--open**

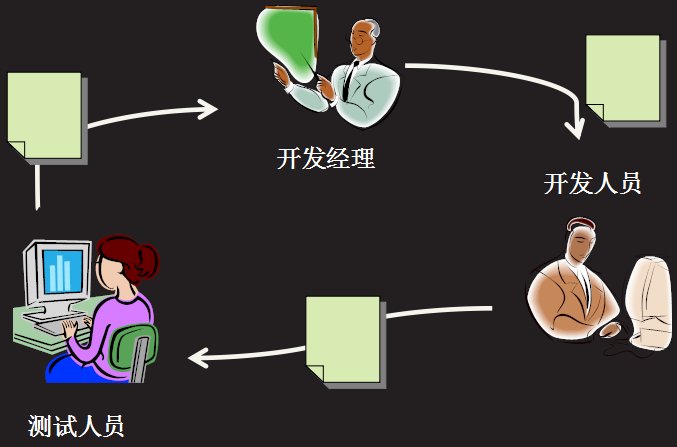
**已解决--**fixed

已关闭--closed

补充两个bug状态：

重新激活--reopen

被拒绝的--rejicted

扩展：面试题：缺陷的跟踪管理过程（步骤、流程、生命周期、bug的一生）？

答：

步骤1:测试人员发现bug，编写缺陷报告，提交给开发经理（新的--new）

步骤2:开发经理审核该bug，

情况（1）；审核通过，确认是有效bug开发经理将bug指派给相应的开发人员（激活-open）

情况（2）：审核未通过，认为是无效bug，开发经理将拒绝该bug，bug被拒绝后，测试方要进行相应的处理（下一道面试题讲）

步骤3:被指派的开发人员要解决该bug（已解决-fixed【待返测的bug】）

步骤4:测试人员对已解决的bug进行返测

情况（1）：返测通过，测试方（测试人员、测试主管）将bug关闭

情况（2）：返测未通过，将bug重新激活，指派回该名开发人员再次修改bug，直到返测通过，关闭为止

问题2:测试人员提bug，如果被开发方拒绝，要怎么办？（经验型问题）--思路

首先：确定被拒绝的原因，然后自检自查明确是否是有效bug

接下来：根据被拒的原因，找相应的人员沟通讨论（产品经理、开发人员等）

如果解决过程中遇到问题，应向直属领导汇报，由领导出面代表测试团队解决

最后：如果经过沟通处理确定是有效bug，谁拒绝的谁负责将bug激活，纳入回bug处理流程中

如果沟通讨论后确定是无效bug，那么测试方（测试人员、测试主管）负责将bug关闭

1. 严重程度（severity）

就是表明该bug有多糟糕，对程序的破坏有多大

1. 常见的严重级别：

1级--致命的--urgent

2级--严重的--high

3级--中等的--medium

4级--建议性的小问题--low

1. 说明：：
   * + 1. 严重级别不一定都分成4级，有的bug管理工具会分成5级，甚至更多级别
       2. 严重级别定义较为笼统，在实际工作中容易引发争议，所以测试团队通常会给出详细的严重级别分级说明，但要注意：不同公司、甚至同一公司，不同项目组的定义都可能会有区别，测试人员应注意参考
       3. 项目测试中，中等级别的bug最多，建议性的小问题主要集中在项目测试前期较多
2. 优先级（priority）

希望或者建议开发方在什么时候解决该bug（决策权在开发方，开发方可以修改优先级）

1. 优先级的常见级别
   1. 1级--立即解决bug--urgent
   2. 2级--下个版本解决bug--high
   3. 3级--在软件发布之前解决--medium
   4. 4级--经讨论，尽量在软件发布之前解决--low
2. 说明：
   1. 不同的bug管理工具对于优先级的划分不同，有4级的，也有超过4级的
   2. 优先级的详细说明，不同公司不同项目组会有不同的解释，测试人员应注意参考
   3. 通常情况下，优先级与严重程度是成正比的，bug越严重，优先级越高，bug越不严重，优先级越低，但是有特殊情况，例如：UI缺陷，严重程度地，但是优先级高

扩展：关于bug的严重程度和优先级的面试题？

Q1:影响优先级定义的因素有哪些？

答：（1）缺陷的严重程度，通常是越严重优先级越高

1. bug的影响范围，影响范围越大优先级越高
2. 开发人员的开发压力，开发压力越小，优先级越高
3. 解决bug的成本（时间），成本越低，优先级越高

Q2：严重程度和优先级一定是正比关系吗？

答：不一定是正比关系，例如：UI缺陷，严重级别地但优先级高

Q3：严重程度和优先级确定后，开发还能修改吗？

答：严重程度一旦确定，不能修改

优先级确定后，开发方可以修改（通常是往后改）

Q4：**在发布的软件中，是否可能存在发现但是没有解决的bug？**

**答：在发布的软件中，有可能存在发现但是没有解决的bug**

**该类bug缺陷通常严重程度不高（中等及以下），数量很少，要经过“bug讨论”，权衡解决bug的成本和不解决是否会给用户造成严重损失或带来法律诉讼，经过权衡利弊后，才能确定**

**该类bug通常会在后期通过升级版本或打补丁的方式解决**

1. 缺陷描述/重现步骤（description）

就是将发现的bug的过程（步骤+数据）记录下来，使开发人员能够根据描述重现（再现）该bug

（就是让开发人员看明白是什么bug）

具体要求：逻辑清晰、用语专业准确、易读易懂、不要出现二义性（歧义）、不要有评价性的语言，做到如实记录

常见格式：

步骤：--step

预期结果--expect result

实际结果--actual result

1. 缺陷报告-上机练习

案例：即时贴程序

要求：每个同学课上编写5条缺陷报告

练习1:添加新便签可以超过50--bug1

经验：如果功能没有实现，通常属于严重缺陷，但如果功能实现了，极值限制（最大值、最小值）出现错误，通常属于中等级别缺陷

练习2:添加的新便签都重叠排列--bug2

经验：缺陷报告中附带“证迹”（截图、视频）

提示：测试人员应养成随手截图的职业习惯

练习3:没有任何确认提示消息，就直接删除所有便签--bug3

练习4:便签编辑框可以输入超过200个字符--bug4

说明：如果企业没有对缺陷描述有格式要求，那么测试人员只要按逻辑将缺陷发现过程描述清楚即可

练习5：设置便签弹出日期时间可以超过2022年12月31日--bug5

1. 缺陷报告总结
2. 缺陷报告的作用
   * + 1. 可以记录缺陷
       2. 利用缺陷报告可以实现对bug的跟踪管理
       3. 通过缺陷报告实现对bug的分类、统计和总结
3. 如何识别缺陷？
   * + 1. 参考需求，当功能实现与需求不符时就是bug
       2. 参考缺陷的五条定义
       3. 参考测试用例的预期结果，当实际结果与预期结果不一致
       4. 测试人员与测试同事、开发人员、产品经理、用户等沟通讨论来识别bug
4. 编写缺陷报告的注意事项
5. 小的缺陷也要报告
6. 对于不可重现的缺陷（随机bug）也要报告，但要注意该bug不可重现
7. 及时报告缺陷，不做任何评价
8. 对缺陷的严重程度、优先级的划分准确、客观、不夸大bug
9. 缺陷描述清晰、准确、易读、使用最少、必须的步骤，保证缺陷可以再现
10. 在提交缺陷报告之前一定要认真审核，确保提交的缺陷是有效的，而不是因为自己的疏忽或操作不正确造成的“假缺陷”

补充：随机bug

随机bug也叫偶发bug或不可重现bug（建议少用）

随机bug：就是按照相同的测试过程，时有时无的bug

提示：如果发现随机bug也要提交bug，而且要说明该bug为随机缺陷；要尽量详细说明发现随机bug的步骤和数据，如果有机会尽量截取证迹

第三讲 测试用例

等价类划分法+边界值法

1. 软件项目的基本测试流程（重点面试题）
2. 熟悉、分析需求（准备）
3. 制定测试计划

说明：1）测试计划由测试主管（组长/经理）制定，测试人员主要阅读并按照计划实施

1. 测试计划的主要组成
   1. 测试项目简介
   2. 测试周期排期
   3. 测试参考资料及提交资料
   4. 测试资源准备、测试人员、测试环境（软件、硬件）、测试工具
   5. 测试风险及预案
   6. 缺陷严重程度和优先级的详细规则
   7. 测试策略（经验）

功能测试、性能测试、界面测试、安全性测试等

1. 面试题：你能设计测试计划吗？

可以

1. 设计测试

根据需求分析➡提测试点️➡编写测试用例➡评审测试用例

1. 执行测试，记录测试结果

测试结果：通过-pass

失败-failed -有bug

1. 记录bug，并对bug进行跟踪管理
2. 测试总结

·由真实的测试数据生成

·《测试报告》、《测试总结报告》等

说明：1）相关文档：测试计划、测试用例、缺陷报告、测试总结报告

1. 项目测试结束的一些要求：
   1. 按照需求规格说明书的要求，100%覆盖测试
   2. 所有的评审过的测试用例要100%被测试执行
   3. 缺陷的解决率要达到95%以上，并且未解决的遗留bug要有相应的说明安排
   4. 测试相关的文档要按要求提交归档
   5. 测试团队要最终给出测试质量结论（《测试报告》）
2. 测试用例
3. 什么是测试用例/案例

测试用例/案例--test case/instance

在测试执行之前由测试人员编写的，用来指导测试过程的重要文档

1. 测试用例的核心组成：

用例编号、用例标题、测试目的、预置条件、测试步骤、预期结果等

说明：不同公司用例的模版可能不同，除了以上核心组成外，还可能包括：参考资料、产品、项目、功能模块、撰写人（用例作者）等用例基本信息

1. 功能测试/黑盒测试中常用的设计测试（用例）的方法有哪些？

等价类划分法、边界值、判定表法、场景法等

补充：错误推测法（经验型）测试方法

1. 设计测试用例可以参考什么？
2. 参考需求，根据需求要求设计测试
3. 参考开发的核心技术文档

说明：有可能参考不到

1. 参考被测系统

说明：测试人员主要参考需求设计测试用例，但是可能用例设计还不太完善（大概40%-70%），所以测试人员通常会参考已开发的被测系统再补充部分测试用例，这样使用例设计更为完善

1. 与开发人员、产品经理、用户等沟通确认

补充：实际情况测试参考资料可能不齐全，测试人员应利用一切能利用的资源展开测试，尽量完成测试任务（例如：一些补充方式查找网上的资源、参考同类竞品设计测试等）

1. 测试思想
2. 穷举测试：将所有可能的数据或情况全部测试。穷举测试思想是质量最好的（100%测试），但是测试效率太低（没有那么多时间都测），所以实际测试工作中不能采用
3. 理想测试思想：就是挑选最少的测试数据，但是达到最好的测试效果（性价比高）--抽样测试

理想测试思想，测试效率高，但是毕竟没有测试所有数据，可能存在遗漏bug的风险，测试人员应在时间允许的情况下，尽量补充测试，来降低遗漏缺陷bug的风险

1. 等价类划分法
2. 应用场合（测试方法适合测什么情况）

在程序中，有数据输入的地方，适合采用等价类划分法展开测试

1. 等价类划分法的测试思想（范围）：

将大量数据划分成若干范围，再从每个范围中挑选少量数据进行测试--将无限测试变为有限测试，是一种优化的测试方法（抽样测试）

1. 两个概念：有效等价类和无效等价类

有效等价类：对程序员来说正确的、合理的输入数据集合--验证功能是否可以正确实现--正向测试

无效等价类：对于程序员来说错误的、不合理的输入数据集合--验证健壮性，异常处理能力--反向测试

1. 测试步骤（如何测）

**基本工作思路：**

**分析需求➡根据需求选择合适测试方法来提炼测试点➡记录测试点➡覆盖测试点编写测试用例➡评审用例➡执行测试➡️记录bug，跟踪管理bug**

1. 被测案例：即时贴-便签右键菜单-设置标题功能

步骤1：分析需求，初步划分等价类

需求：**设置标题：设置便签的标题，要求**

**（1）字符个数最少为1个字符，最多为40个字符**

**（2）不能包含 / ： \* ？这些特殊字符**

分析整理：标题为1-40个字符，不能为空，不能包含**/ ： \* ？这些**特殊字符

说明：以需求为依据分析，划分等价类（测试点）

A、不能为空

无效：为空

有效：不为空

B、不能包含**/ ： \* ？这些**特殊字符

无效：包含**/ ： \* ？**

**有效**：不包含**/ ： \* ？**这些字符

1. 数据类型和长度：1-40个字符

有效：1-40个（字符串长度）

无效：>40个字符，<1个字符

初步划分结果：

有效等价类：要填写1-40个字符，并且不能包含/\*？：这些特殊字符

无效等价类：为空、包含/\*？：｜标题长度>40个字符

步骤2:细化等价类

无效：包含/\*？：等特殊字符

说明：当测试某个符号无法代表其他符号的结果时，应细化等价类，分别单独测试

步骤3:填写《等价类表》--记录测试点

步骤4:编写测试用例--

每个等价类范围至少要挑选1个代表数据进行测试

注意：在工作中编写测试用例前，应明确用例的模版和用例中一些内容（例如用例辩题）的格式规范，以及提交设置等基本要求

补充说明：

1. 测试用例编写后需要评审；互评互审、开评审会
2. 案例的问题总结说明
3. 、无数数据的组合没有测试，例如：一个控件多种无效数据组合起来

分析：在每个无效数据都单独测试的前提下，没有必要重复测试没中组合，如果测试时间充分，可以适当挑选一些特殊组合重点补充测试

1. 测试方法：错误推测法（经验型）

错误推测法是常用于补充数据测试用例的方法，就是测试人员根据自己的经验或直觉，对程序中可能出错的情况，设计测试用例测试的方法

错误推测法常应用于程序员经常出错的情况，项目中之前出现过同类错误的情况，用户经常会错误操作导致出现的情况

1. 特殊数据：[空格、0、极大值、极小值、@.\_等](mailto:空格、0、极大值、极小值、@._等)
2. 特殊情况

查询结果显示0行或1行

测弱网状态

等价类划分+边界值法

1. 边界值法
   * + 1. 方法说明：边界值是程序开发中比较容易出错的数据，所以专门推出“边界值法”对易错的边界数据进行重点测试，边界值法是等价类划分法的重要补充
       2. 应用场合（用在哪？）

在程序中，有数据输入的地方适合使用边界值法，边界值法常常与等价类划分法配合使用（等价类划分负责划分范围测试，而边界值法负责针对容易出错的边界数据进行测试），从而形成一套比较完善的测试方案

1. 如何划分边界值？
2. 边界值点

有效等价类和无效等价类之间的分界点叫边界值

最小值-min

最大值-max

1. 次边界值点

边界值点左右两边相邻的点（4个）

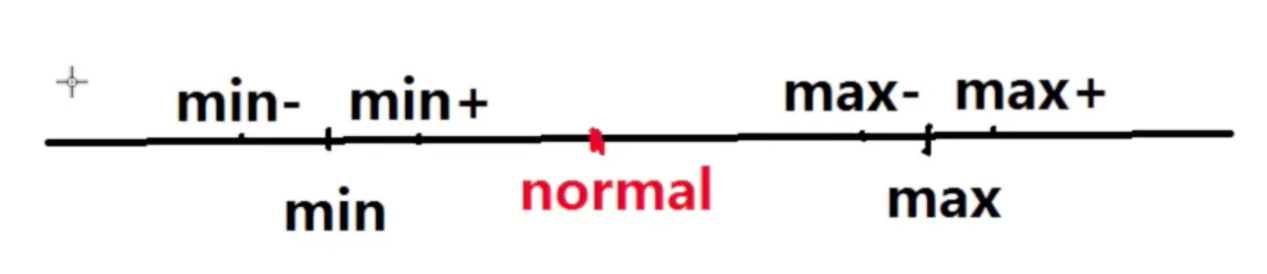
Min-：无效最小次边界

Min+：有效最小次边界

Max—：有效最大次边界

Max+：无效最大次边界

补充：normal-有效等价类的代表数据



补充：面试题

Q1：如果测试时间紧张，应优先测试哪些边界相关的点？

测试软件健壮性，就是

Q2：在需求中，是不是所有的测试数据的边界值是开始就已经确定的

不是所有数据的边界值开始就能确定，有的数据的边界值开始就可以确定，但是有的数据的边界值是在设计、研发过程中逐步确定的（需求是可变更的）

Q3：划分以下数据的边界值

1. 年龄：18-60之间的整数

Min：18

Max：60

Min+：19

Min-：17

Max-：59

Max+：61

1. 标题：1-40个字符

Min：1个

Max：40个

Min+：2个

Min-：0个=为空

Max-：39个

Max+41个

1. 等价类划分法+边界值的综合应用

案例：信息注册

（1）步骤1:分析需求，选择测试方法（等价类+边界值），提炼测试点，填入《数据分析表》

姓名：1——20个字符，不能包含数字，不能为空

有效 无效 边界值

年龄：18——60之间的整数，不能为空

有效 无效 边界值

1. 步骤2:先设计测试方案（思路），再编写测试用例
   1. 先做正向测试（正确）

说明：多个控件的有效数据组合起来测试

b）接下来做反向测试（错误）

说明：当前案例中，由于有屏蔽现象的影响，所以无效数据应单独测试

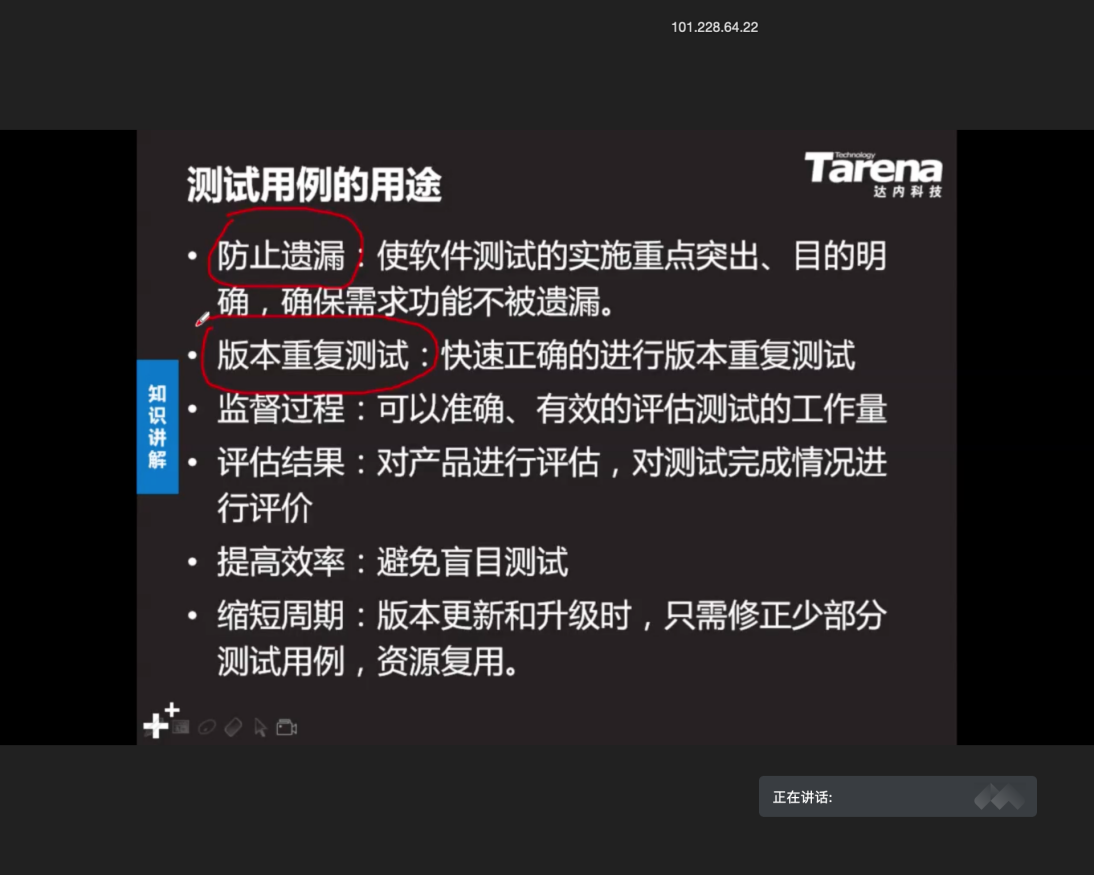
c）最后利用错误推测法，将你从经验和直觉角度出发，觉得可能出错的情况在补充设计测试

1. 测试用例总结
2. 测试用例的作用？

1）·防止遗漏：使软件测试的实施重点突出、目的明确，确保需求功能不被遗漏。

·2）版本重复测试：快速正确的进行版本重复测试

1. 监督过程：可以准确、 有效的评估测试的工作量
2. 评估结果 ：对产品进行评估，对测试完成情况进行评价
3. 提高效率：避免盲目测试
4. 缩短周期：版本更新和升级时，只需修正少部分测试用例，资源复用。



1. 关于测试用例的注意事项：
2. 编写测试用例前，应明确用例的模版、用例组成不分的填写规则、用例提交的位置（服务器）、附件的命名规范、提交位置等
3. 测试用例需要评审，评审通过方能用于测试（测试执行）
   1. 评审方式：互评互审、评审会（内部、外部）
   2. 用例评审细节

注意：用例设计应避免冗余（用例重复）

高标准：高覆盖、高效率、遗漏bug少

**关于测试中的评审**

在软件开发过程中的适当阶段对软件阶段产品进行评审，是确保软件产品最终质量的重要方法。阶段评审可以对某个开发阶段的阶段产品进行评审，也可以对某几个开发阶段的阶段产品进行综合评审。在每次阶段评审中，必须履行正式手续，填写必要的评审表格，以利于项目管理工作，利于产品验收时的质量检查工作。

流程：

1. 主持人确定评审日期，参与评审人员

2. 评审前2天，测试用例发给所有评审人员

3. 评审人员记录测试用例问题

4. 评审会议，作者讲解用例，回答别人问题

5. 会议结束，修改用例，准备再次评审

6. 再次评审通过，用例基线化

注意：如果是需求评审、设计评审等等会议，把以上步骤中测试用例改为相应的需求、设计文档即可

用例评审标注：

1. 用例设计策略

2. 用例覆盖率

3. 用例书写格式

4. 用例可读性

5. 用例可执行性

测试用例评审要点（一定要看）

1.测试用例是否覆盖了所有需求.

2.测试用例内容是否正确,是否与需求目标一致.

3.测试用例内容是否完整,是否清楚包含输入和预期输出结果.

4.测试用例是否具有指导性,是否能灵活指导测试人员通过用例发现更多缺陷,而不是限制他们的思维.

5.找出哪些需求不可测：无法准备环境、可测试性达不到等等原因；

6.对具体需求的实现结果的确认（设计人员、开发人员、测试人员的认识是否一致，如果不一致，谁说了算）；

7. 测试用例本身的描述是否清晰，是否存在二义性

8.是否考虑到测试用例的执行效率.往往测试用例中步骤不断重复执行，验证点却不同，而且测试设计的冗余性，都造成了效率的低下

参与该评审的人员

包括：开发人员、系统分析人员、项目经理和相关测试人员

1. 对于测试人员的要求
2. 基本要求

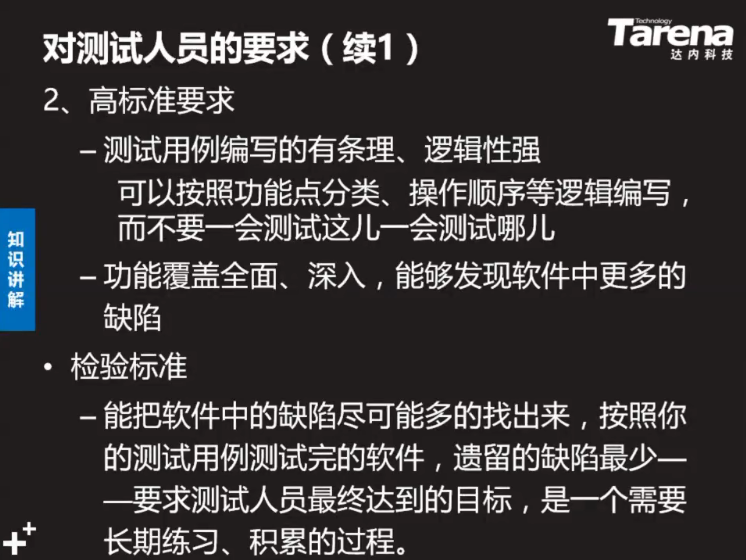
·在编写一条测试用例时，要求步骤描述清晰、准确、易读、预期结果明确

如果有特殊的设置、预置条件等，要明确写出 如果有输入数据，一般要给出输入数据取值

如果有附件，要给出附件存放位置，名称

1. 检验标准

· ·写完用例后，让别人照着自己的用例去执行测试，可以很顺畅的执行下来



第04讲 判定/决策表法

1. 应用场合

界面中有多个控件，控件之间存在组合或限制关系，不同控件组合对应不同的输出结果，为了理清输入组合对应的输出结果是否正确，可以应用判定表法测试。

注意：判定表法需要一一理清每组组合关系，所以比较适合测试组合数量少的情况，如果组合数量比较多，应选择抽样测试

1. 测试步骤

被测案例：一卡通自动充值模拟系统

步骤1:分析需求，找出输入条件（输入因素）

请投币 50元 100元

充值金额 50元 100元

步骤2:分析有哪些输出结果

1. 充值成功
2. 退卡
3. 找零
4. 给出提示充值失败

将分析结果填入《判定表》

步骤3:分析输入条件中有哪些组合和限制关系（8个组合和2个限制）

限制：请投币50元-请投币100

充值金额50元-充值金额100元

组合

投币50元+充值50元

投币50元+充值100元

投币100元+充值50

投币100元+充值100

投币50元+不选择充值金额

投币100元+不选择充值金额

没投币+充值50元

没投币+充值100元

步骤4:分析每对组合应对应怎样的输出结果，填判定表

步骤5:编写测试用例

判定表每列代表一种组合

1. 判定表练习
   * + 1. 饮料自动售货机
       2. 练习2
       3. 练习3
2. 判定表总结
   * + 1. 判定表法是适合测试多个控件组合情况的方法，常与单选按钮、复选框、选项较少的下拉列表控件配合应用
       2. 判定表的特点
3. 输入条件的顺序是无关紧要
4. 输出结果的顺序也是无关紧要的
5. 先测哪个组合后测哪个组合无关紧要
6. 每个组合之间是相互独立的

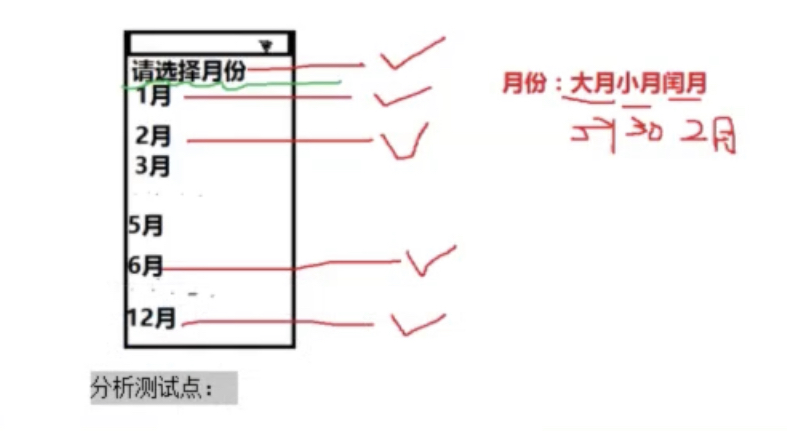
第05讲 场景法（重点）

1. 补充知识
2. 关于下拉列表或列表框控件的测试
   1. 当下拉列表或列表框中选项较多时，不应测试所有选项，一般应至少选择3项进行测试，分别为：第1项（最小值）、中间某项（有效等价类的代表数据-normal）、最后1项（最大值）

所以：下拉列表和列表框控件分析应用了等价类划分+边界值法

1. 选择测试下拉列表选项时可能会超过3项，要具体情况具体分析

例1:提月份测试点，如图所示：



分析测试点：

边界值：min-1月、max-12月

等价类：大月、小月、闰月

如果有不选月份的情况需单独测试

1. 中文字符占用空间问题（了解）

一个中文字符 占用/=（2）字节

1. 场景法的应用场合
2. 场景法是非常常用的测试方法，常用于测试软件的业务流程或业务逻辑
3. 场景法是基于软件业务的测试方法
4. 要求测试人员应熟悉所测软件的相关业务
5. 测试人员在设计测试时要以软件业务为依据来设计测试过程
6. 场景法要求测试人员将自己想象成最终用户，模拟用户使用软件测各种情景，主要模拟两类场景：
   * + - 1. 业务正确实现的场景

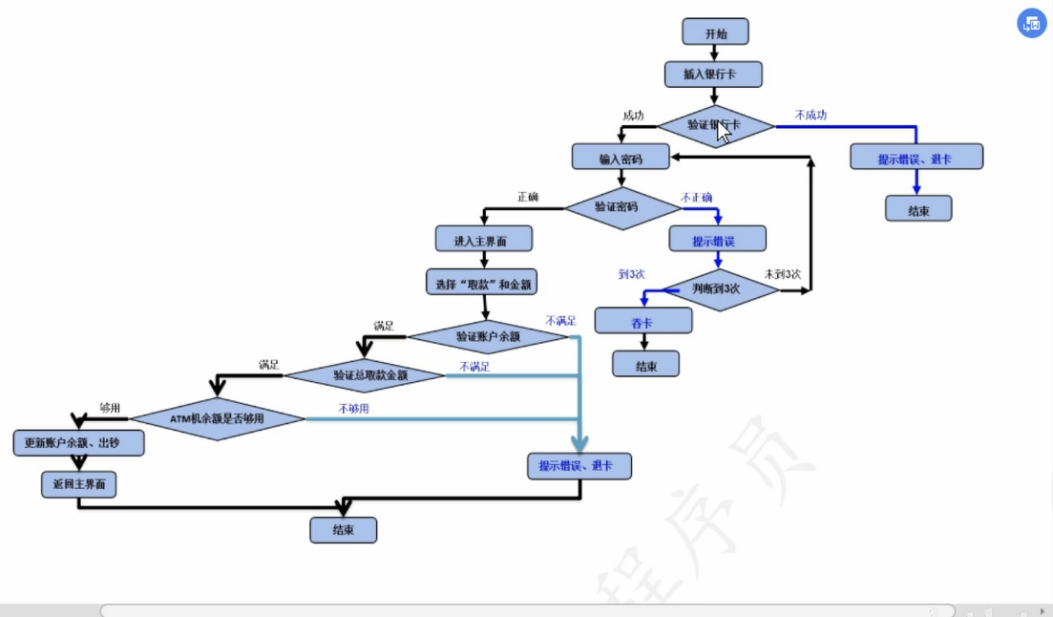
--验证功能是否可以正确实现--正向测试

* + - * 1. 业务实现错误的场景

--验证健壮性（异常情况的处理能力）--反向测试

1. 问题：当接到测试任务后，测试人员应使用场景法对整体业务展开测试，当确认核心业务可以实现，再用等价类划分法、边界值法对细节展开测试（总结：先整体后细节）
2. 场景法测试的两个要素
   * + 1. 要求测试人员要熟悉所测试软件的业务，最好能成为该软件业务方面的“专家”（项目面试中会有业务相关说明）
       2. 技术层面（两个概念）
3. 基本流：也叫正确流或有效流，模拟业务正确实现的流程
4. 备选流：也叫错误流或无效流，模拟业务实现错误的各种情景
5. 场景法的测试步骤

案例1:ATM取款功能



步骤1:熟悉需求，整理业务，列出基本流和备选流

1. 基本流：正确操作成功取钱的流程

️插卡验证银行卡➡输入正确密码（6位数字）➡选择“取款”功能➡输入/正确的取款金额➡出钞、更新账户余额、提示➡退卡

1. 备选流：取款失败的各种流程
2. 验证卡失败
3. 密码输入错误3次以下
4. 密码输入错误3次，要锁卡吞卡
5. 账户🈷余额不足
6. 超出当次取款上限（5000）
7. 超出当日取款上限（20000）
8. ATM机余额不足

说明：以上并不是所有的备选流，还有其他备选流场景，例如：卡内资金冻结、取款金额不是100的整数倍等场景没有列出等

步骤2:生成场景，填《场景表》

说明：熟练后步骤1、2可以合并

步骤3:编写测试用例

说明：通常1个测试点对应1条测试用例，但是也有特殊情况，有的场景可能需要多条用例测试，也有可能一条用例可以测多个场景

案例2:五子棋游戏

步骤1:分析需求（五子棋规则），整理有哪些场景

场景：黑棋（玩家）先手

玩家胜

玩家负

双方和棋

黑棋（玩家）下出禁手，应判黑棋负

步骤2:覆盖场景，设计测试用例

说明：在游戏场景测试中，很多时候无法进行详细的步骤设计，只能对所测游戏情景，进行大概的用例设计，这样就对测试执行人员的测试执行能力（游戏的技术）有很高的要求

第05讲 场景法（2）

1. 场景法案例3:美萍酒店管理系统-系统设置-房间设置-删除房间类型功能
   * + 1. 测试准备：
          1. 测试环境准备
          2. 测试数据准备
2. 设计测试（产出物《测试用例》）

步骤1:分析需求，整理业务，生成场景，填《场景表》--提测试点

需求：删除房间类型与服务生这两个因素有关

A、单独分析：房态

1. 如果房间类型中所以房间都没人住，那么房间类型可以删除
2. 如果房间类型中包含有房间有人住（哪怕是一间房有人住），那么该房间类型不能删

B、单独分析：服务生

1. 如果房间类型分配有服务生，那么该房间类型不能删
2. 如果房间类型没有分配服务生，那么该房间类型可以删除

C、综合考虑房态和服务生两个因素，分析什么场景房间类型可以删除？什么场景房间类型不可以删除？

1. 能删除房间类型的场景

没人住+没有分配服务生

1. 不能删除房间类型的场景

有人住+没有分配服务生

有人住+有分配服务生

没人住+有分配服务生

需求的补充分析：

没人住的房态：可供、停用、清理

有人住的房态：占用、长包房、预定

步骤2：覆盖场景，设计测试用例

1. 美萍酒店管理系统-系统设置-房间设置-删除房间功能

要求：每个同学独立分析需求，整理业务、填场景表，覆盖场景设计测试用例

提交物：场景+用例

时间：20-30分钟

需求：删除房间只与房间的房态有关

第06讲 软件测试基础理论

1. 软件开发阶段基本划分（复习）
2. 需求分析阶段

由需求分析人员完成

产出物：《需求规格说明书》

1. 概要设计阶段
2. 详细设计阶段

由系统架构师/分析师完成

产出物：《概要设计说明书》

《详细设计说明书》

1. 编码阶段

由开发人员完成

产出物：程序

问题：哪个阶段引入的bug最多？其次是哪个？哪个最少？对测试安排有哪些影响？

答：需求阶段bug最多，其次是设计，引入bug最少的是编码阶段；

由此得出结论：需求和设计阶段也需要测试，不仅要测程序，文档也要测（评审），测试要符合“尽早测试原则”和“不断测试原则”

1. 软件测试阶段划分

说明：以下测试阶段划分中没有涵盖需求和设计阶段的测试内容

1. 单元测试
   1. 单元测试是最小的测试单位，例如：测一个功能、一个窗口、一个方法（函数）、一个类等都可以是一个测试单元
   2. 单元测试参考是：详细设计说明
   3. 单元测试理论上应采用白盒测试方法

补充：在公司的实际情况是：白盒测试虽然测试质量好，但成本较高、效率较低，所以部分软件公司的单元测试为了降低成本会由程序员负责完成，为了保证单元测试的质量，公司通常会采用交换互测或开发测1轮（白盒），测试再测1轮（黑盒）的方式解决

d）什么是桩模块？什么是驱动模块？（理解概念）

说明：在单元测试中，测试者有时需要编写桩模块和驱动模块

·驱动模块：模拟被测模块的上一级模块（调用被测模块）

·桩模块：模拟被测模块的下一级模块（被‘被测模块’调用）

总结：驱动模块➡被测模块➡桩模块

1. 集成测试
   1. 集成测试阶段也叫组装测试阶段，在单元测试的基础上，将功能模块逐步、有序的合并在一起的测试过程
   2. 在集成测试阶段，功能是逐步合并的，不是一蹴而就，所以会出现若干的临时版本（迭代测试的过程）
   3. 集成测试阶段参考的是《概要设计说明》
   4. 集成测试阶段应用黑盒测试结合白盒测试的方式，一些核心功能会重点应用白盒测试方法（灰盒）
   5. 冒烟测试

说明：冒烟测试通常在开发团队提测新版本时，要先进行冒烟测试，验证该版本是否能够达到基本的质量标准

冒烟测试就是当开发方提测新版本后，组织较少的人员（1-3人），花费较少的时间（0.5-2天），对所提测版本的主要功能展开快速测试，如果主要功能可以实现，就冒烟通过，测试组全组展开全面测试；如果主要功能无法实现，版本不稳定就拒绝接受该版本，打回开发组

f）集成测试阶段：当测试团队接到新提版本后的基本工作思路--

首先：冒烟测试--通过冒烟测试验证是否要接受该版本，展开全面测试

接下来：

返测

回归测试

最后：如果有新功能，要对新功能展开测试

说明：有些版本可能只是修复版本，没有新功能

1. 系统测试
   1. 系统测试：是在软件功能组装完成后，对集成了硬件和软件的完整系统，进行的模拟真实环境的测试过程
   2. 系统测试阶段的测试要点：

完整系统在模拟真实环境中是否能符合需求规格说明书的所有要求

完整系统的软件、硬件之间的兼容性的测试

系统测试阶段参考需求规格说明

系统测试阶段测试的测试方法是黑盒测试

补充：确认测试

确认测试发生在系统测试之前，主要确认两个内容：

1. 确认组装后的完整系统是否可以进入到全面的系统测试阶段中
2. 确认相关文档是否准确齐全，尤其是交付用户的文档

说明：确认测试阶段时间相对较短、参与人员也相对较少，所以通常不将其与单元、集成、系统、验收等测试阶段所并列

1. 验收测试（UAT）

UAT：User Acceptance Test--用户接受度测试

1. 验收测试阶段：主要由最终用户参与对软件进行检查，测试人员可以从旁边辅助支持。
2. 验收测试阶段用户通常会使用自己的真实信息作为数据，并且按照自己的使用习惯来使用软件验收测试阶段主要由两个小阶段组成：
   1. alpha （å） 测试

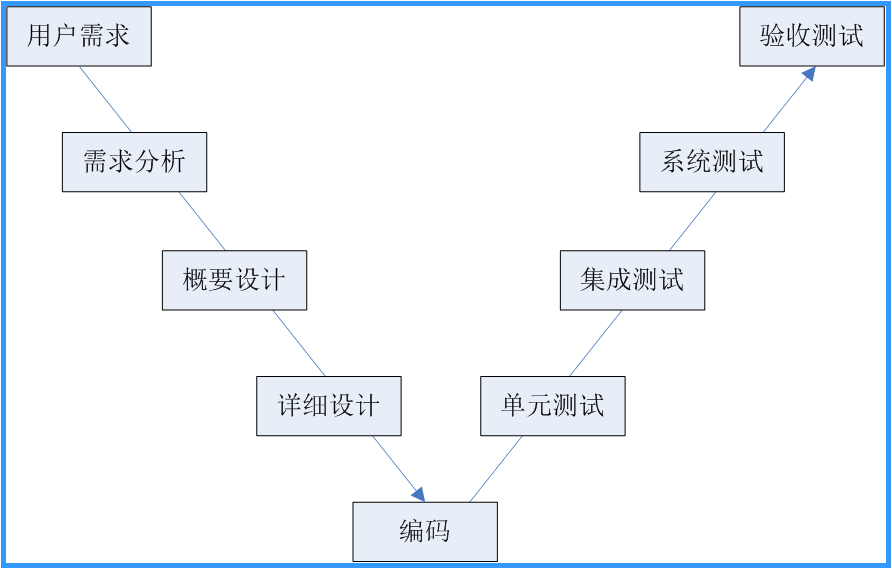
测试在软件公司指定的测试环境中进行（项目组能够对bug更有控制力，更容易获取bug相关信息）

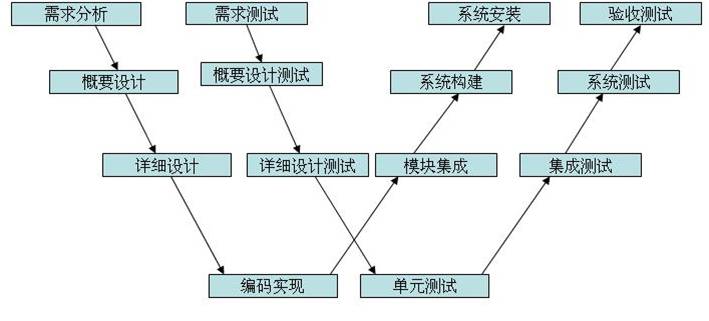
理论上alpha 测试应该由最终用户进行，测试人员辅助支持；但是实际情况最终用户可能不能参加，这样常常由测试人员代替用户进行或者用户自己请测试机构代替进行

* 1. Bate （ß ）测试

bate测试是在最终用户自己的测试环境中进行，项目团队对于bug的控制力减弱，同时获取bug信息的难度较大

例如：公共类软件（浏览器、网游、QQ等）做bate测试，就会将要测试软件免费发放给最终用户，通过收集用户使用软件的问题（bug）来获取bug信息

1. 软件测试的模型
2. 软件测试模型可以表示：开发阶段和测试阶段（级别）之间的对应关系，软件测试模型中V模型最为常用，其他还有例如：X模型、H模型、W模型等测试模型
3. V模型（常考）
4. 会画
5. 优点
   * 1. 开发阶段和测试阶段（级别）划分清晰明确
     2. 开发阶段和测试阶段（级别）之间对应关系也清晰明确
     3. 测试级别涵盖较为全面，既包括最低级别的单元测试（代码级、专业级），也包括最高级别的验收测试（界面级、用户级）
6. 缺点
   * 1. V模型缺少需求和设计阶段的测试过程，容易引起误解：测试只是开发后的收尾工作
     2. V模型不符合“尽早测试原则”和“不断测试原则”
7. W模型



1. W模型：也可以理解成是“双V”模型，第一个V是完整的开发活动，第二个V是完整的测试活动
2. W模型解决了V模型中的问题
   * 1. W模型中加入了需求和设计阶段的测试活动，使开发和测试活动之间是并行的关系，而不是测试是开发的收尾
     2. W模型更符合“尽早测试原则”和“不断测试原则”
3. 软件测试分类（概念）
4. 测试技术分类
5. 黑盒测试

也叫功能测试，就是将被测软件看成是一个黑色盒子，在不了解内部结构的前提下，对检查程序功能是否符合需求规格说明书的要求

1. 白盒测试

也叫结构测试，就是将软件看成一个透明的盒子，即清楚软件测内部逻辑结构，对所有的逻辑路径展开进行测试

补充：（a）白盒测试质量较高，但成本较高、效率低、

（b）白盒测试要求测试者要有编程基础、也有白盒测试方法，例如：条件覆盖、循环覆盖等；白盒测试也需要设计测试用例

1. 灰盒测试

黑盒测试和白盒测试的测试要素结合起来就是灰盒测试，集成测试阶段会常用到灰盒测试

1. 按测试时是否需要运行程序来进行划分：静态测试和动态测试

静态测试：不需要运行程序就可以进行测试

例如：文档测试、界面测试、静态代码测试：不是白盒测试，是对代码的规范和标准进行检查

动态测试：需要运行程序才能进行测试，例如：功能（黑盒）测试是动态测试

说明：白盒测试有时是静态测试，有时是动态测试

问题：白盒测试和静态代码测试有什么区别？

白盒测试：测试的要点是程序的逻辑实现，要求测试人员要有编程基础，会白盒测试的常用测试方法，并且可以设计测试用例；

静态代码测试：测试的要点是代码的规范和标准，不要求测试者有编程基础，也不需要编写测试用例，只需要能够参照代码检查规范，记录问题即可

1. 功能测试和性能测试

功能测试：

1. 所有的软件都要先进行功能测试，保证功能正确运行后再考虑其他测试
2. 功能测试可以手工实现也可以自动化测试
3. 常用的功能自动化测试工具有：selenium、appium等

性能测试：

1. 通常是分布式软件（C/S、B/S）需要进行性能测试
2. 性能测试不能手工实现，只能自动化实现
3. 常用的性能自动化测试工具有：jmeter、loadrunner等
4. 其他
5. 返测：测试人员对开发人员已解决的bug，进行测试，来验证该bug是否被修复
6. 回归测试：就是在当前版本中对上个版本测过的功能再重新测试一遍，；回归测试用来验证当前版本中的原有功能是否出现了新的问题；

回归测试中设计重复测试，为了提高测试回归测试效率，在满足条件的前提下可以使用自动化测试

1. 随机测试（猴子测试 monkey test）

随机测试就是随意测试，发现bug的测试方式，随机测试只能常规用例测试执行后的补充测试行为，不能作为主要测试方式使用

1. 兼容测试和兼容测试的分类

兼容性测试就是测试被测软件与硬件和其他软件之间的兼容性

兼容测试的分类 ：（3类）

1. 与硬件兼容

与整机兼容

与外设兼容

B、与软件兼容

与系统兼容（品牌、版本、位数（64位、32位））

浏览器兼容性

与其他应用软件

与数据库兼容

C、数据兼容

软件不同版本之间的数据兼容

1. 常用的黑盒/功能测试方法有哪些？选择测试方法的策略是什么？

功能测试常用的测试方法有：等价类划分法、边界值法、判定表法、场景法等

选择测试方法的策略：

1. 等价类划分法和边界值法：通常在程序中有数据输入的地方会考虑使用等价类划分法和边界值法。（这两种方法常常会配合使用，等价类划分法通过划分范围抽取代表数据，将无限测试变为有限测试；边界值法针对容易出错的边界数据进行重点测试。两种方法配合后形成一套较为完善的测试方案）
2. 判定表法：主要测试多个控件之间的组合情况，检查不同输入组合对应的输出结果是否正确，判定表法比较适合测试组合少的情况。
3. 场景法：场景法是常用的测试方法，适合测试软件的业务流程和业务逻辑

场景法是基于软件业务的测试方法

场景法要求测试人员将自己当成是最终用户，模拟用户使用软件的各种场景，主要模拟业务正确实现的场景和业务实现错误的场景