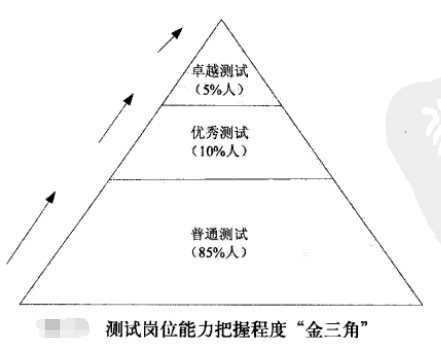
# 1、测试的三种境界

测试入门、优秀测试人员、卓越测试人员



## 1.1、测试入门

从两方面来概括测试所需要的能力：测试技术和测试素质

技术方面：测试设计能力、文档编写

测试素质：热爱测试、诚实、耐心和毅力、细心、善于沟通、学习能力、怀疑精神

## 1.2、优秀测试

如果一个人用优秀来评价时，说明该员工一定在某一方面做出了突出的成绩，至少是非他做不可。

包括以下方面：精通业务、精通测试技术、创造性、富有探索精神、分析定位问题

## 1.3、卓越测试

从字面意思理解就是比优秀更好。

包括以下几个方面：把测试当做事业、测试指挥官、分享与传递、专业技术的带头人、引领未来。

***思考：软件测试的目的是什么？①保证软件的质量，提高软件的质量；②验证程序的运行负荷需求定义，是正确的；③发现程序中的Bug***

# 2、找Bug的三重境界

第一重：围绕着Bug转，通过发现Bug提高产品质量；第二重：站在Bug之上；第三重：预防Bug。

## 2.1、第一重：围绕着Bug转

王国维《人间词话》：“古今之成大事业、大学问者，必经过三种之境界：

* + **昨夜西风凋碧树。独上高楼，望尽天涯路**。此第一境也。做学问成大事业者，首先要有执着的追求，登高望远，瞰察路径，明确目标与方向，了解事物的概貌；
  + **衣带渐宽终不悔，为伊消得人憔悴**。此第二境也。比喻成大事业、大学问者，不是轻而易举、随便可得的，必须坚定不移，经过一番辛勤劳动，废寝忘食，孜孜以求，直至人瘦带宽也无怨无悔；
  + **众里寻他千百度，蓦然回首，那人却在，灯火阑珊处**。此第三境也。” 要达到第三境界，必须有专注的精神，反复追寻、研究，下足功夫，自然会豁然贯通，有所发现，有所发明，就能够从必然王国进入自由王国。

围绕着Bug第一重：分成3个阶段：

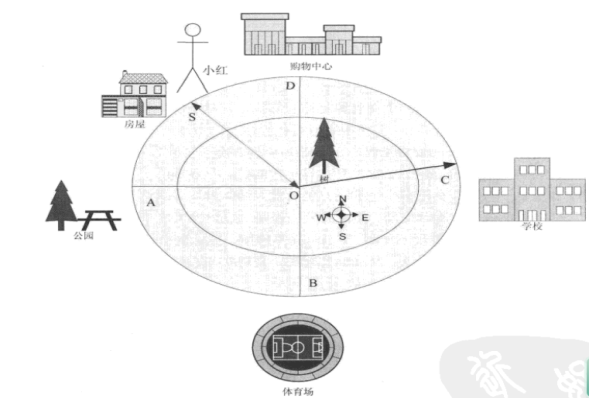
* + 发现Bug(软件测试的目的在于发现程序中的错误)
  + 定位Bug(最短路径重现Bug、重现偶发Bug)
  + 关闭Bug(测试不仅仅是发现程序中的Bug，更重要的是跟踪解决这些Bug，因为只有Bug被解决了，质量才能得到提高，测试的价值才得以体现出来；于开发交流修复缺陷时，若想赢的人心，首先要让他相信你是最真诚的朋友)。

## 2.2、测试的两种思维

包括逆向思维和发散思维。

### 2.2.1、逆向思维(美特斯邦威)

路人甲每天从家S出发到学校C上课，通常情况下是从S--->O--->C。但是她某一天的路线是S--->O--->C--->O--->C或者S--->O--->A--->D--->C。当然实际生活过程中走此路线的人并不会多，但是在软件中一些隐藏的比较深的Bug就是需要这种迂回曲折的路劲后才能暴露出来。



### 2.2.2、发散性思维

发散性思维是一种探求多种答案，最终使问题获得圆满解决的思维方法。由于其视野开阔、思维活跃，可以产生大量独特的新思想。发散性思维在两个阶段得到充分体现：一个是设计阶段，另一个是测试执行阶段。

## 2.3、第二重境界---站在Bug之上

站在全局的角度看待整个项目。项目的三要素时间、成本和质量。我们需要在三者之间寻找一个平衡点，来保证在规定的时间内，以较低的成本获得较高的质量。也就是前面提到的测试如何配合项目成功做正确的事，以及正确的做事。

对于测试来说做正确的事就是站在用户的角度进行常用功能点的模块的重点测试，避免非常用功能点的过度测试，浪费成本、资源。而正确的做事就是采用合理、全面的测试方法验证软件是否满足用户需求。(2-8原则)通过该原则，可以使软件测试在项目的成本与时间做到效益最大化。

首先，分析我们的系统哪些功能对用户来说是核心且重要的功能，然后安排合适的测试工程师负责这些模块，设计测试用例并进行重点评审，测试执行过程重点跟踪。每一次版本迭代，即使此区域的代码没有调整，仍然需要对此进行回归测试。

2-8原则的具体内容：80%的错误是由20%的模块引起的；80%的测试成本是由20%的模块引起的；80%的测试时间都是花在20%的的模块中。

当跳出"测试的目的在于发现Bug"这个圈子时，会发现山外有山，软件只不过是产品的一个小的部分，它的运行还建立在其他系统或者硬件资源上，要想获得项目的成功，整个体系的各环节需紧密合作，相互支持。例如在整机测试过程中会发现，有时候找到的Bug不是软件的问题，而是硬件的问题。

## 2.4、第三重境界---挑战零缺陷

孔子有言：人无远虑，必有近忧。如果用到软件测试的活动中理解，如果我们不从源头解决问题，那Bug是找不完的。后期所有的手段都是治标不治本的方法，你修复完了这个，另外一个地方会有新的问题产生。这便是本节的要介绍的：挑战零缺陷，预防Bug。

缺陷的防与堵。对于后续发现的Bug，在项目后期要进行漏侧分析，目的是通过分析找出漏侧的根本原因，分析测试中哪个环节工作存在缺失，以拿出可规避的措施来。

"零"缺陷文化。Bug发现的越早修复的成本越低，发现的越晚修复的成本更高，甚至成千上万倍。零缺陷不是意味着没有缺陷，就连赫赫有名的微软，每周还是不是的发布几个补丁来进行修补漏洞。所谓的零缺陷，是我们的测试活动要以这个为最终目标，Bug为0。分解到每个人身上就是，每个人负责的模块要做到无缺陷。

零缺陷不是意味着测试人员都失业了，而是意味着测试的难度加大，寻常的技术已经难以发现Bug，而要集中精力去发现更深层次的Bug。

# 3、测试景观设计

测试用例设计是测试设计在技术上的核心所在。在测试领域除了技术路线上的设计还存在着围绕测试工作的测试流程和管理上的设计，使它们构成了一个整体，是事、人、技术流程的统一体。

* 事：一件要完成的任务，可能是某个项目或某个模块的软件测试任务，它是测试工作的驱动源；
* 人：有了任务之后，需要人的参与；
* 技术：如同软件设计一样，软件测试也是一项技术性很强的工作，如何能更好、更快地完成测试任务，与测试策略的制定、测试方法的选取、测试方案和用例的设计密不可分；
* 流程：人是活动的个体，人与人之间也是千差万别的，任何一个公司或组织都是一个团队在工作，人作为个体如果没有流程的制约，组成的团队将会变成一盘散沙，离有效、快捷协作相差甚远。
* 管理：上面这些因素的考虑，都离不开"管理"二字。管理分技术上的管理和非技术上的管理(人与事)，但是这些都是围绕着事来运行的。

## 3.1、解读测试设计

在《软件测试的艺术》中对测试的描述：软件测试是一项极度富有想象力和高智商的、有挑战性的工作。当版本稳定下来时，通过执行以往的测试用例时，将很难发现Bug，此时再往下挖掘如同大海捞针。而往往由于软件本身的特殊性，以及在测试过程中要考虑时间和成本代价，不允许我们做到100%的穷举测试，这也注定我们不可能把所有的Bug都找出来，。在这阶段，一个人的Bug敏感度尤显得重要，而对Bug的敏感度依赖于过硬的技术积累与极富有挑战性的创造性思维。而这种技术的积累与创造性的思维相结合迸发出的一种思想，往往能带来一种高效地改进现有测试方法或流程的新设计。

测试流程管理的内容：

* + 用例管理工具；
  + 用例设计规范；
  + Bug管理工具；
  + Bug管理规范；
  + 测试方案用例评审机制。

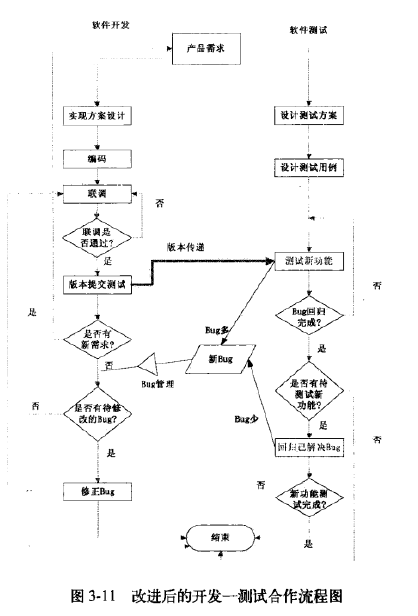
## 3.2、测试组织模式设计

包括三大类：以开发为核心的组织模式、以项目经理为核心的组织模式、独立的测试组织模式。

* 以开发为核心的组织模式，测试负责人负责向开发负责人汇报工作；
* 以项目经理为核心的组织模式：测试负责人负责向项目经理汇报工作，真实反映项目的质量情况；
* 独立的测试组织模式：测试组织具有真正意义上的独立，具有权威的地位，能够客观的反应产品的质量情况。

## 3.3 测试流程设计

从广义上讲，测试流程是所有关于测试事务的抽象统一体，流程有静态性，同时也有动态发展的特点。从静态来看，主要指已有的测试流程规范，包括测试工作指南、测试计划、测试方案、测试用例设计模板等，这些属于测试流程体系中的文档输出指导范畴。还有属于测试管理流程范畴的内容，包括测试新员工入职指引、项目测试手册、测试配置管理手册、测试只是经验库、培训库等。还有如功能测试方法库、系统测试方法集、经典Bug分析集，灰盒测试、白盒测试指南等一系列技术性的流程、规范或指南。这些流程体系、知识库的积累，并不是一朝而就，是一个测试团队成长的重要标志。也正因为如此，**流程体系有动态发展的一面，随着公司内外环境的变化，需要根据不断变化的需求做合适的调整。**



负责版本发布的开发人员，在版本正式提交给测试前，必须在真实的用户场景下运行软件，确认实现的新功能或更改的正确性，并作确认记录，此记录随同软件传递给测试。

测试人员制定版本接入环节：时间不超过2小时，如果出现严重或致命Bug，影响测试工作的开展，通知测试leader，停止当前的测试。

## 3.4、测试应用设计

对测试对象进行深度分析，更深层次的挖掘测试对象，不仅仅停留在表面去观察测试对象的反应。

了解用户的操作习惯及系统的工作原理，才能设计出完善的测试用例。

# 4、测试架构设计

测试架构设计同其他方向的架构设计一样，需要纵横全局思考，**不仅考虑从测试技术的应用、研究，还需要考虑测试人员的管理，测试流程的设计**等等。

测试架构设计包括两个方面：测试管理架构和测试技术架构。

测试管理架构设计的精髓：让每个测试人员看到希望。---职业发展路线(管理路线、技术路线、其他路线)

## 4.1、技术路线

* 初级测试工程师

技能要求：执行用例、记录测试发现的Bug、跟踪Bug声明周期状态、回归Bug，参与项目测试方案与测试用例设计等的评审。

对象：一般为刚踏入测试行业的新手。

* 测试工程师

技能要求：设计模块级测试方案、测试用例、测试代码为工作重点，执行测试用例为辅，组织模块级测试涉及评审，参与模块级设计方案评审，能独立完成规范测试流程各节点的工作。

对象：工作1~3年的测试人员。

* 高级测试工程师

技能要求：设计项目的总体测试方案，制定测试计划，对项目的某类或某特性进行测试，如自动化测试、性能测试、安全测试等，对有一定技术难度或深度的测试有独挡一方面的能力，培养测试新人成为合格的测试工程师

对象：工作3~5年的测试人员

* 资深测试工程师

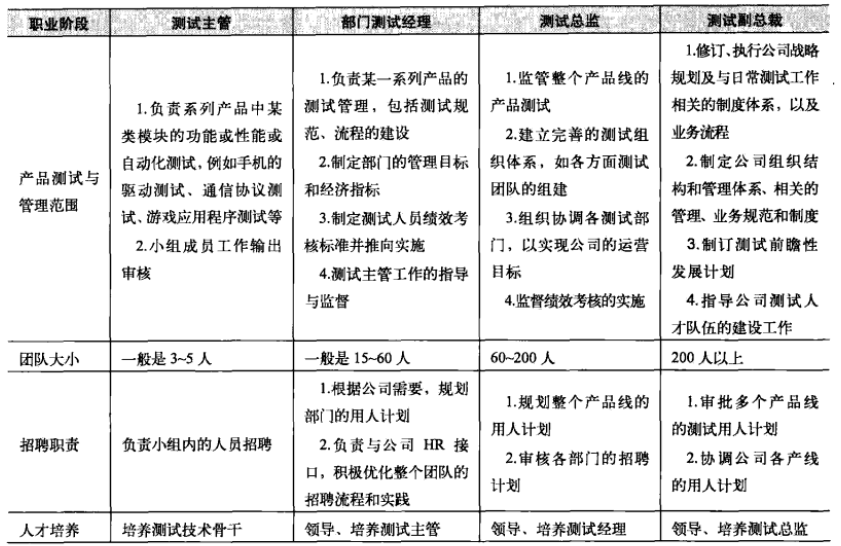
技能要求：制定某类产品测试总体策略、测试流程，制定相关测试规范、指南，负责某类产品测试平台建设；指导重点测试方案的设计，对测试设计有一定的创新能力，高级测试工程师的导师；

对象：一般为6~10年的测试人员

* 测试架构师

技术要求：负责产品线测试策略、测试方法、测试规范的制定，规划、设计和开发测试平台，为了不断降低公司成本而进行新测试技术的研究、实践和推广，技术线上测试人才梯队的结构设计。

## 4.2、管理路线



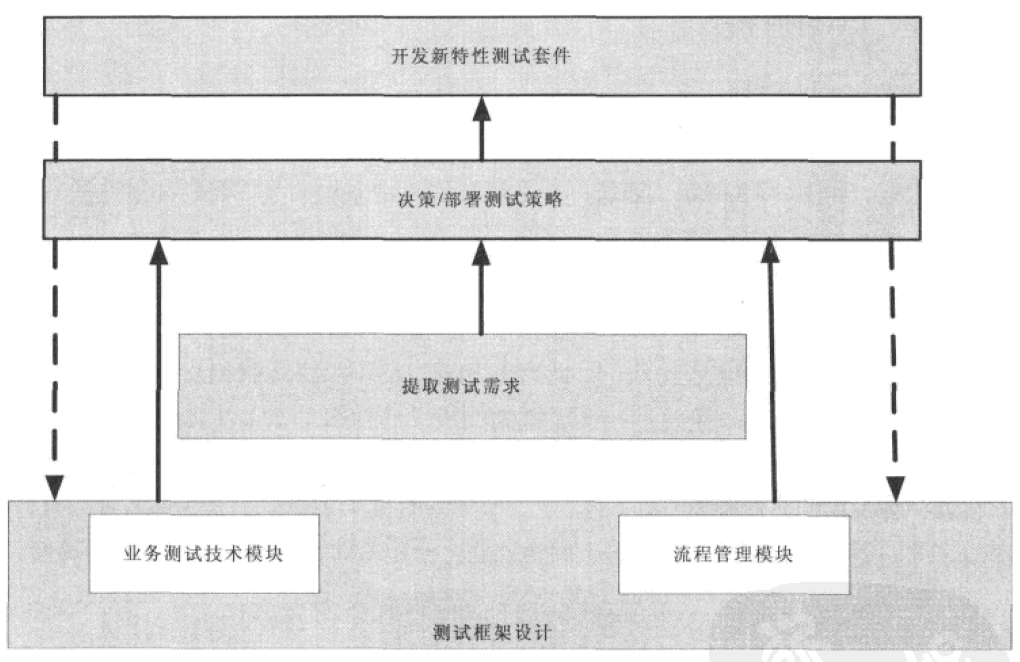
## 4.3、业务测试架构设计

业务测试框架设计：包括业务测试技术和流程管理两部分，基本框架的设计离不开业务需求与公司流程体系。其表现形式可以是一种测试方法、一块代码程序、一系列的流程规范等；

提取测试需求：包括与测试工作相关的业务需求和非业务需求；

决策/部署测试：为测试需求服务的一系列解决方案；

开发测试套件：具体解决测试需求的措施集，如测试用例集、脚本程序、测试工具等。



## 4.4、测试建设之基石---测试框架的设计

测试框架设计包括业务测试技术模块和流程规范管理模块。业务测试技术框架包括功能测试框架、性能测试框架、自动化测试框架；流程规范管理模块包括用例编写规范、自动化脚本编写规范、测试用例模板、测试方案模板、测试报告模板等。

## 4.5、搭建测试框架的方法

测试框架设计的主要目的是项目的测试都能以框架平台为基础，并在框架的控制内进行实施，使得在不断的项目测试历练中积累的经验能够沉淀、传承、复用，从而提高测试的整体效率，降低项目研发与维护成本。

* 业务功能测试框架：可重用的测试思路，框架中可列出各测试点及其测试方法；
* 测试数据：特别是一些影响性能的数据，包括测试数据生成程序，集中一起管理，并写好使用说明，以便各项目测试过程中需要随时取用；
* 自动化测试脚本、接口函数等测试套件：需无条件纳入配置库进行版本管理，对项目通用的接口函数考虑封装为独立的中间库，把此库作为自动化测试框架的元素之一进行维护；
* 回顾Bug库中历年发生的Bug：对这些Bug进行分类，分析这些Bug的发生原因，拿出日后如何控制的可行方法，并对原有测试框架做改进或补充；
* 测试文档设计模板：可结合公司内部开发流程要求，重点考虑测试文档本身应有的内容，模板中给出例子为宜；
* 制定测试设计评审机制、测试用例设计规范、Bug录入规范、测试代码设计规范，并在项目的执行过程中不断完善。

***测试套件包括：测试方案、测试用例、测试报告、测试总结、测试数据、测试代码等一些列为某特定测试对象而服务额测试输出工件总和。***

# 5、测试需求分析与测试策略制定

## 5.1、测试从需求开始

测试需求主要来源于用户的业务需求，那么该如何了解产品的业务，为测试任务迈开重要的一步呢？首先是识别测试需求，接着分析测试需求，最后确定并提取测试需求：

识别测试需求

分析测试需求

提取测试需求

提取完测试需求后接下来要做的就是针对每一个对象要如何进行测试，拿出具体的方法和措施，便是测试策略的制定。

***测试项------>测试点------>检查点***

## 5.2、多管齐下溯需求

在没有规范的开发流程支持或执行有偏离的情况下，测试人员的主观能动性发挥着重要作用。在产品研发的过程中需求会一直变化，而这些变化常可能是体现在专家评审的会议纪要、邮件或者QQ中，在这种情况下，我们需要把关注范围扩大，以多管齐下的方式关注需求的入口。从各种可能的渠道及时获知需求信息，作为工作的来源，同时反推需求，要求把零散的需求文档或纳入需求库，正式的给出测试的依据。

## 5.3、考虑可测试性需求

①快速理解需求的捷径---需求宣讲

需求宣讲的组织：内审后，通知项目组相关人员事先预审，给出问题反馈截止时间；介绍需求的背景：让开发人员及测试人员能够清楚知道用户的用途，实现后的软件是用来做什么的，能满足用户哪些需求，能给公司创造什么样的价值，使开发人员及测试人员的工作目标很明确，处处从用户立场考虑；需求宣讲内容：将所有内容都讲解一遍，可按照重要性分批次讲解，确保开发及测试人员都正确理解需求；辅助答疑式宣讲：在宣讲时，针对项目组成员提出的问题进行一一回复。

在设计测试用例的过程中会发现不少需求定义的Bug，这时候也要将这些Bug更新到缺陷库中，跟踪解决此类问题("好记性不如烂笔头")。

②需求定义也会错

不同的人对同一句话会有不同的理解，这就有了显式需求和隐式需求。显式需求指有明确定义的一系列约束软件实现的要求，可以是给定具体值的数据字典，或有序的业务流程图。而隐式需求并不是需求设计人员特意隐藏，更多的是由理解人员对某方面专业知识，对产品的业务了解程度有限导致的。

例如：某软件的需求中有这样一句定义：对用户产生的数据，需每天定时备份(默认为每天晚上12:00，用户可设置)，并按需生成书面报告，报告中的数据不存在法规风险。此处的"法规风险"具体指什么呢？所以需要测试人员保持一个敏感的质疑的心。

③从设计需求中提取测试需求

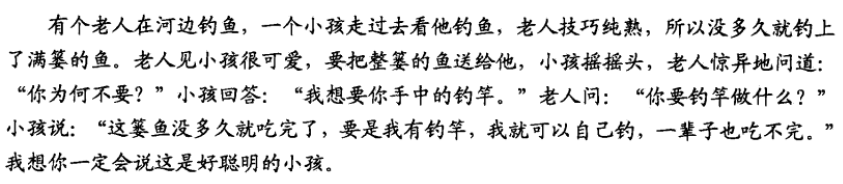
软件需求是软件测试需求的主要来源，但不是全部来源，软件概要设计、详细设计等都是测试需求分析的对象，如某些开发语言对环境有特殊的依赖，这时候就需要进行额外的分析。

## 5.4、确定顶层方向性测试类别

提取测试架构设计模型中的测试对象，需要确定顶层方向性的测试对象，也可理解为测试业务涉及的测试类别，以使测试人员能针对不同的测试类别考虑对应的测试策略。不同的测试类别，需要不同的测试方法，会涉及不同的测试资源，这在项目开始之初，需要在测试计划中规划好。测试按照重要等级依次为：功能测试、性能测试、接口测试、安全性测试、安装升级卸载测试、兼容性测试、帮助文档测试。

# 6、聚焦测试方案设计

钓鱼的故事---知其然，不知其所以然



## 6.1、测试方案设计注意点

* 测试方案只有结论，没有分析过程。只说明某个测试对象(如某业务功能点)要测试的方法。为什么要这样测试，而不是那样测试？
* 方案的设计要能以理服人，测试经验固然重要，但没有以分析为背景的经验结果的直接表达，常会引导测试人员只看到对象的表面，而缺失深度分析。
* 测试方案的设计仅限于对软件需求的理解，是不够的，需要结合软件的概要设计、详细设计进行实现原理上的剖析，补充测试点与测试方法。
* 测试方案的分析结果，就是测试点与测试方法，可以通过建立索引表的方式，把测试点与测试方法转换为用例设计的思路。

## 6.2、测试模板示例

第一章 概述

1.1背景介绍

描述该测试方案设计的背景

1.2 目的

编写本测试方案的目的，指出预期的读者范围

1.3 用途

方案设计对象在软件系统中的位置、用户使用场景等

1.4 定义

专业词语、缩写词等

1.5 参考资料

列出此方案编写的参考资料

第二章 测试范围

在顶层测试需求(通常项目测试负责人提供)的基础上，确认本测试方案的服务对象有哪些，如测试需求提出"模块A的测试"，在此要明确包括其功能、性能、可靠性、与其他模块的接口等具体的测试方向。

第三章 测试分析

这是方案设计的重点，根据测试对象在软件系统中所处的位置采取合适的分析方法，提取出需验证的测试点，以细化测试对象。测试分析的对象不同，要求立足的角度不同，分析的范围也不同。如同软件开发中有总体设计方案、模块设计方案一样，测试也可分系统级的总体测试方案和模块级的模块测试方案，这个可以根据具体情况来操作。

第四章 测试方法

针对测试分析中提取出的测试点，提出具体的测试思路及方法，例如，哪些测试点采用手工功能测试，哪些采用自动化测试；哪些采用白盒测试，哪些采用回合测试等；是否需要特别开发的测试数据、测试工具等。

第五章 测试用例设计

本处指出测试方案与测试用例的关系，例如，以测试方案中的测试点、测试思路为依据，设计测试用例，保证每一个测试需求都有测试用例与之对应。

具体的用例设计采用的工具，以及结合用例模板如何管理等也可在此处说明（此处只说明用例设计的总体思路与方案的关系）

## 6.3、三层架构模式分析法

三层架构模式分析法的原理

该方法把整个软件系统划分为3个层次，包括功能应用层、模块接口层、系统接口层；找出测试对象在整个系统中所处的位置，分析其本身业务功能，以及其他模块之间，与系统接口之间的关系，然后由外向内，逐层深入分析。

分析的示例



## 6.4、多叉树节点分析方法

多叉树节点分析方法分析测试对象的总体思路：测试人员站在用户的角度，以用户使用场景为主线，结合设计分析分析目标测试对象与哪些因素有关，包括模块内部和模块外部。用户场景分为基本场景和扩展场景，基本场景是指用户常用的功能操作流程，扩展场景则是指某些比较特殊或不常用的功能操作流程。用户的主流程可以看成是一棵树的树干，而与其相关联的流程则是树枝。

# 7、测试用例设计

设计用例的常用方法有：等价类、边界值、反常规操作法、倒推法、综合策略。

测试用例设计规范：

* 用例编号命名规范化：用例的ID在整个产品的用例库是唯一的，如：TC-01-001，TC表示测试用例，NO表示模块编号，001为具体的用例编号；
* 定义用例的必填元素：包括用例编号、用例标题、操作步骤描述、预期输出，其中用例标题不超过20字；
* 可操作性强：操作步骤描述清楚，无歧义，不能出现“或”“如果”“多个”“等等”等不确定的词语，避免不同的执行人员测试出不同的结果；
* 预期输出唯一、明确：规定用例的预期输出不能出现模棱两可的词汇，如“可能”“如果”“大约”等不确定词语。

# 8、测试执行流程的设计

## 8.1、需求测试

主要是通过分析检查系统的需求分析定义，尽早地修正和验证需求，使用各种黑盒测试设计方法来设计最小数目的测试用例，同时满足最大的功能覆盖，关注质量问题产生的源头。

提出评审的开始时间和结束时间点。有明确的计划，方便评审人员把评审纳入工作的一部分来做；

评审内容再突出重点的基础上求全。参与评审的内容进行模块划分，可由各负责人来负责，做到重点模块全员参与。

记录需求问题，采用统一的方式，对需求问题进行记录。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 需求文档名称 |  | 版本号 | |  | | | |
| 需求位置 | 问题描述 | 提出人 | 提出时间 | 解决方案 | 处理人 | 处理时间 | 状态 |
| 电话本模块P10 | 内置记忆卡最多可以保存多少条电话本记录 | 张三 | 1965.01.11 |  |  |  | 提交 |
| 电话本模块P18 | 电话号码除了数字外，是否允许字符 | 李四 | 1966.02.02 | 补充需求：允许支持以下几个字符"+"-"P"W" | 王五 | 1966.02.03 | 已解决 |
| ...... | ...... |  |  |  |  |  |  |

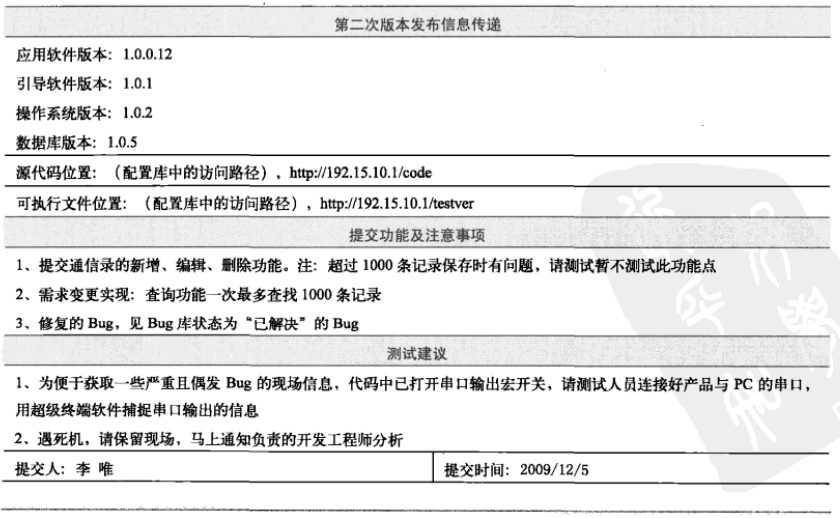
需求评审宣讲。需求问题反馈后，在较短时间内由需求负责人组织需求宣讲会，并就内审提出的问题一一答复处理措施。

此后在测试方案与测试用例设计的过程中，由于测试方案的设计需从需求中提取测试对象，需对每一点需求进行深入理解与分析，用例设计的预期输出需要需求中有明确的输入与输出定义。因此测试方案与用例的设计过程，常会遇到一些需求不明确或需求没考虑到的问题，此类这样的问题必须记录下来，作为Bug反馈个需求人员进行需求整改。

需求测试检查点：可读性、二义性、一致性、统一性(风格统一)、是否存在需求过度(需求不需要涉及开发设计方面的内容)。

## 8.2、内部版本发布测试

软件研发完成转测试时，除了软件本身外，另外重要的内容就是关于版本的变更信息，如注意事项、对测试的建议、新增的功能点、改动及修复的功能点等。如果测试不清楚这些内容，很多情况下会造成无效的测试、遗漏测试，从而不能掌握测试的重点。这时候可以通过建立“测试传递表”机制来保证测试与开发的信息一致性。如下图所示：



在拿到新版本后测试人员有时候会遇到测试一两个小时发现遇到一些重大的Bug导致后续的操作无法进行，这时候开发抓紧时间修改；然后测试了一两小时后有发现另外一个Bug导致测试无法进行，这样一次次的测试---修改，会造成测试的重复工作。这时候建立“冒烟测试机制”，可以预防此类现象。冒烟测试作为一种测试策略，是对基本功能的确认，非深入测试。这时候可以通过自动化来完成。具体测试工作可以由测试人员提供测试用例，开发负责人指定人员在测试环境进行测试，或者由测试人员执行。

## 8.3、回归测试

一个软件系统在其开发及维护阶段，都将面临不断地变更，这些变更可以概括为3个方面：Bug修复、新增加功能、代码优化及代码移植。软件一旦有变更，必须对其进行验证。

* 确定回归测试内容

***新增需求、缺陷修复、代码优化及移植***

* 基于用例的测试回归方法

*影响分析回归：指把变更影响到的功能点对应的测试用例挑选出来，重新回归这些用例。站在黑盒测试的角度，找出更改点所处的业务位置，结合系统中与它关联的业务，画出业务功能的影响关系图。然后根据关系图进行增删相应的测试用例的选取，最终完成回归测试。*

* 基于Bug的回归测试方法

*根据之前的Bug进行回归测试，很可能会遇到当前Bug修复了，但是又引入了新的Bug，这叫回归不充分。*

*可通过以下的方式来预防该问题：记录Bug是如何被修改的；开发人员确认Bug被更改；记录Bug是如何被测试的。*

## 8.4、交叉测试

敏捷开发中有结对编程方法来保证开发质量，而在测试中同样可以采用此种方法来提高测试的质量。每个人的思维都有思维误区，因此测试人员A通过的模块再交给测试人员B去测试，往往能发现新的Bug，这就是所谓的交叉测试。交叉测试效果显著，是解决测试过程中出现迷茫时的一剂良方，但是该方法如何用？什么时候用？是一个需要谨慎考虑的事情。

英国杰出的戏剧作家萧伯纳说：“倘若你有一个苹果，我也有一个苹果，而我们彼此交换这些苹果，那么我和你还是一个人一个苹果；倘若你有一种思维，我也有一种思维，而我们彼此交换这些思维，那么，我们每人将有两种思想。”

**交叉测试模式：保守式交叉测试、渐进式交叉测试、零交叉。**

* 保守交叉模式：该模式的开始时期为软件版本已经基本稳定，模块测试负责人自身认为已很难再发现Bug时，交出模块进行交叉测试，缺点为一些深入的Bug依旧会存在；
* 渐进式交叉测试：将整个模块分成多个阶段，A测试模块a后，B再测试模块a+其他模块，依次进行；
* 零交叉：并不是所有的测试都需要进行交叉测试，如：压力测试、自动化测试等输入为固定，执行过程也为固定，完全由程序进行控制的测试，就没有必要进行交叉测试。

# 9、测试输出管理设计

## 9.1、Bug管理

### 9.1.1、Bug生命周期示意图



### 9.1.2、Bug级别定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1级致命错误 | 2级致命错误 | 3级致命错误 | 4级致命错误 | 5级致命错误 |
| 法律法规 | 不符合行业标准的法律、法规定义、可能导致人身伤害、财产损失等 |  |  |  |  |
| 功能性能 | 程序引起的死机、软件崩溃、死循环 | 数据丢失或破坏、业务流程错误、程序接口错误等主要功能或性能受到影响 | 次要功能或性能丧失或受到影响，如界面刷新慢、打印报表格式错误等 | 界面个别字符串显示不全，但不影响整体意思的理解 | 需求没有明确定义的建议处理意见 |
| 用户体验 |  | 严重影响用户使用，用户容易察觉到、可能引起用户严重不满，导致用户投诉 | 影响用户使用，但不会引起用户严重不满，导致投诉，如UI风格不一致、提示不友好 | 不影响用户使用，用户很难察觉或无法发现的缺陷 | 符合用户常规使用习惯的建议 |
| 生产效率 |  | 严重影响生产效率和直通率 |  |  |  |
| 品牌形象 |  | 明显影响公司品牌形象和公司声誉 | 可能对公司品牌形象和公司声誉有直接影响 |  |  |
| 使用频率 | 用户高频率使用的核心功能 | 用户常用功能 | 用户不常用，但给用户使用的功能 | 不影响用户使用，但存在缺陷 |  |

### 9.1.3、Bug录入约束(简称三要素)

* **对于测试人员而言：**
* 标题：简洁描述该Bug是什么，发生了什么样的现象；
* Bug的详细描述：对Bug发生的环境、条件、现象进行详细的描述，是标题的详细扩展；
* 产生Bug的步骤：按步骤把产生Bug的路径写出来，各步骤中关键描述操作软件的动作。最终目的是让开发人员按照此步骤能复现该Bug。更进一步讲，测试人员在提交Bug的时候需要再思考，上面的步骤是否是重现该Bug的最短路径，是否是误操作导致。
* **而对于开发人员而言：**
* 问题分析：给出故障原因；
* 修复方案：给出解决的方案描述，适当时候可直接给出代码修改片段或涉及的文件名、代码行等；
* 受影响分析：对更改涉及的模块或功能做影响说明；
* 测试建议：根据更改点，建议一些方法或路径来进行回归测试。
* **对回归测试而言：**
* 验证的版本：回归测试的当前版本是什么版本
* 回归策略：写清楚采用了哪些方法回归此Bug，覆盖了哪些路径，用了哪些数据等；
* 验证结果：描述回归测试结果，是通过还是不通过，通过就关闭Bug，不通过则需要重新打开。

### 9.1.4、Bug库的应用

* 为新人快速进入测试角色提供一种捷径；
* 测试经验与技术沉淀的宝库；
* 测试人员相互学习的平台；
* 测试人员与开发人员之间交流的桥梁；
* 通过Bug分析建立Bug预防体系。

### 9.1.5、处理不可重现的Bug

理论上来说，不可重现的Bug是不存在的。那些所谓的不可重现Bug发生与条件有着密切的联系，有些条件非常苛刻或严密，不能重现是因为我们还不清楚其触发的条件。不能重现的Bug根据其发生的概率标注"经常发生"、"偶尔发生"，同时可采取如下操作：

* 保留现场：对于严重的Bug一定要保留现场，请开发人员过来一起察看分析；
* 追溯测试场景：仔细回忆之前的操作和测试数据等，尽可能找到重现该Bug的路径；
* 着手分析：要求软件中能输出运行日志信息的功能，记录用户的操作记录和数据，可为Bug分析提供重要判断；
* 专事专办：不要以为偶然就不会发生，在测试环境会发生在用户环境也有可能会发生，只不过是时间和概率的问题，而一旦发生导致的损失可能不仅仅是一次重启，有可能是一个客户、合同等。根据Bug的影响程度，考虑是否由专人处理这类问题，以快速定位解决。

## 9.2、用例管理

### 9.2.1、用例结构与元素的设计

用例结构一般包含3层，包括：模块---功能点---测试用例标题。

用例元素其关键要点为：测试标题、预设条件、操作步骤描述、预期输出。次要的属性为：优先级、用例类型等。

## 9.3、测试文档模板设计

### 9.3.1、测试流程图



### 9.3.2、测试计划模板设计

测试计划应包括如下内容：

* 事：要做哪些事？包括测试对象和测试范围；
* 工作量评估：评估完成项目测试任务需要的人力、时间。评估工作量时，需初步确定方向性的测试策略，如开展哪些方面的测试手段，包括单元测试、集成测试、系统测试、性能测试、自动化测试、安全测试等；
* 人：要完成测试任务，需要哪些方面的人，需要多少；
* 测试里程碑定义：测试分几个阶段？各里程碑阶段做什么？输出是什么；
* 设备资源：测试过程需要哪些设备、工具来支持；
* 计划维护措施：定义什么情况下需要变更计划。

### 9.3.3、测试方案模板设计

测试方案模板的设计，犹如开发对软件的概要设计。首先要明白的是测试方案是为了解决什么问题。测试计划主要是测试管理上的问题，而测试方案设计的核心是为了达到计划中的目标，解决如何做的问题，是测试分析、测试策略的制定、具体方法确定等一系列测试技术上的综合体现。

测试方案设计可分为3个步骤：第一步是测试对象分析；第二部是测试点的提取；第三部是对各个测试点思路的表达。

## 9.4、测试知识库设计

管理知识库系统的工具有sharepoint、Mediawiki。

可以借鉴的测试知识库的层次架构部署图：



# 10、控制测试过程的实用方法

## 10.1、三级评审机制

第一：自审。自己对自己编写的文档进行审核，自审中发现了哪些问题，以及如何修改的都需要记录下来；(设立有效的检查单)

第二：内审，部门内部审核。

第三：外审，与产品、研发部门一起审核。因为参与评审参与的人较多，站的角度也有不同，尝尝能提出测试设计方面方向性的缺失问题。如某需求的验证没有提到，特别是一些隐含需求；核心用户场景的测试策略考虑不够等。

**测试方案检查单：**

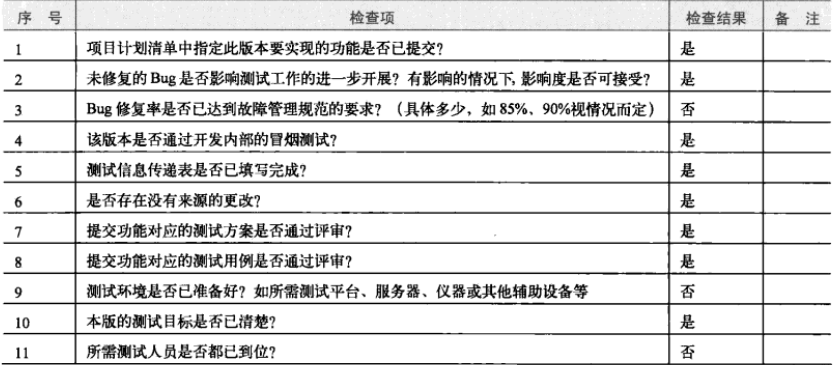
|  |  |
| --- | --- |
| 序号 | 检查项 |
| 1 | 测试对象的分析是否考虑了模块间的相互影响 |
| 2 | 该测试对象是否存在性能上的要求 |
| 3 | 设计需求是否有对应的测试方案 |
| 4 | 对某一测试点的验证方法，除了正向的思路，是否也考虑了逆向的思路 |
| 5 | 每一个测试点的验证思路是否都可以直接导出测试用例 |
| 6 | 用户场景的测试点是否足够 |
| 7 | 业务流程的测试分析是否用了流程图来表达 |
| 8 | 方案中的专业术语是否有专门部分进行了解释 |
| 9 | 方案中的英文缩略词是否有全词的说明 |
| 10 | 测试方案的组织模板结构是否符合公司的模板要求 |

**测试用例检查单：**

|  |  |
| --- | --- |
| 序号 | 检查项 |
| 1 | 是否每一个需求都有对应的测试用例来验证 |
| 2 | 测试用例的设计思路是否全面 |
| 3 | 每条用例的预期结果是否唯一 |
| 4 | 每条用例是否都可操作 |
| 5 | 用例的测试条件是否清楚 |
| 6 | 每一个测试点是否都有逆向用例 |
| 7 | 每一个测试点是否都有异常的用例 |
| 8 | 测试用例是否包含了已知的边界值，如特殊字符、最大值、最小值等 |
| 9 | 对流程业务，是否有对应的流程用例 |
| 10 | 是否考虑了性能测试用例 |
| 11 | 用例的编写是否符合规范要求 |
| 12 | 是否考虑了安装/卸载测试用例 |
| 13 | 是否考虑了升级兼容性测试用例 |

## 10.2、版本控制

**版本接收检查单**



**测试停止检查单**

