**安硕苏州集中开发中心**

**[测试操作规范]**

**—— 机密文件 ——**

上海安硕信息技术股份有限公司

2015.05

文档信息

|  |  |
| --- | --- |
| **文档编号** | 01 |
| **创建日期** | 2015/05/15 |
| **作者** | 集中开发测试团队 |

版本历史

| 版本编号 | 版本日期 | 描述 | 修订人 |
| --- | --- | --- | --- |
| V1.0 | 2015/05/15 | 初步规范文档的编写 | 集中开发测试团队  胡治瑜、古艳丽、相朋辰、徐佳、褚煜琴 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录[[测试操作规范] 1](#_Toc419477030)

[1 概述 6](#_Toc419477031)

[1.1 编写目的 6](#_Toc419477032)

[1.2 读者对象 6](#_Toc419477033)

[1.3 阅读建议 6](#_Toc419477034)

[2 理论基础 7](#_Toc419477035)

[2.1 正确认识测试 7](#_Toc419477036)

[2.2 测试的原则 7](#_Toc419477037)

[2.3 测试与开发的关系 8](#_Toc419477038)

[2.4 测试分类 9](#_Toc419477039)

[2.4.1 按是否关注代码划分 9](#_Toc419477040)

[2.4.1.1 白盒测试 9](#_Toc419477041)

[2.4.1.2 黑盒测试 9](#_Toc419477042)

[2.4.1.3 灰盒测试 9](#_Toc419477043)

[2.4.2 按是否执行程序划分 9](#_Toc419477044)

[2.4.2.1 静态测试 9](#_Toc419477045)

[2.4.2.2 动态测试 9](#_Toc419477046)

[2.4.3 按软件开发的阶段划分 10](#_Toc419477047)

[2.4.3.1 单元测试 10](#_Toc419477048)

[2.4.3.2 集成测试 10](#_Toc419477049)

[2.4.3.3 确认测试 10](#_Toc419477050)

[2.4.3.4 系统测试 10](#_Toc419477051)

[2.4.3.5 回归测试 10](#_Toc419477052)

[2.4.3.6 验收测试 11](#_Toc419477053)

[2.4.4 按测试目的划分 11](#_Toc419477054)

[2.4.5 易用性测试 11](#_Toc419477055)

[2.4.6 功能测试 11](#_Toc419477056)

[2.4.7 兼容性测试 12](#_Toc419477057)

[2.4.8 性能测试 12](#_Toc419477058)

[2.4.9 安全测试 12](#_Toc419477059)

[2.5 常用测试方法 12](#_Toc419477060)

[2.5.1 等价类划分 12](#_Toc419477061)

[2.5.1.1 划分方法 13](#_Toc419477062)

[2.5.1.2 确定等价类 13](#_Toc419477063)

[2.5.1.3 生成测试用例 14](#_Toc419477064)

[2.5.2 边界值分析 14](#_Toc419477065)

[2.5.3 错误推测 14](#_Toc419477066)

[2.5.4 因果图 15](#_Toc419477067)

[2.5.5 综合策略 15](#_Toc419477068)

[2.5.6 案例说明 15](#_Toc419477069)

[3 amarsoft测试流程 16](#_Toc419477070)

[3.1 流程图 16](#_Toc419477071)

[3.2 阶段说明 17](#_Toc419477072)

[3.3 准入准出标准 17](#_Toc419477073)

[3.3.1 测试执行开始标准 17](#_Toc419477074)

[3.3.2 测试退出标准 18](#_Toc419477075)

[3.3.3 测试暂停标准 18](#_Toc419477076)

[3.3.4 测试恢复标准 18](#_Toc419477077)

[3.3.5 各阶段始终准则 18](#_Toc419477078)

[4 测试计划 21](#_Toc419477079)

[4.1 概述 21](#_Toc419477080)

[4.2 原则 21](#_Toc419477081)

[5 测试需求 22](#_Toc419477082)

[5.1 概述 22](#_Toc419477083)

[5.2 依据与收集 22](#_Toc419477084)

[5.3 测试需求分析方法 22](#_Toc419477085)

[5.4 优先级的确定 23](#_Toc419477086)

[6 测试用例设计 24](#_Toc419477087)

[7 测试执行 25](#_Toc419477088)

[7.1 测试准备 25](#_Toc419477089)

[7.2 测试执行 25](#_Toc419477090)

[7.2.1 操作准则 25](#_Toc419477091)

[7.2.2 缺陷管理 26](#_Toc419477092)

[7.2.2.1 缺陷管理流程 26](#_Toc419477093)

[7.2.2.2 缺陷登记 27](#_Toc419477094)

[7.2.2.3 缺陷分类 28](#_Toc419477095)

[7.2.2.4 缺陷等级分类 30](#_Toc419477096)

[7.2.2.5 缺陷优先级分类 31](#_Toc419477097)

[7.2.2.6 缺陷状态分类 31](#_Toc419477098)

[7.2.3 测试日志 32](#_Toc419477099)

[8 性能测试 33](#_Toc419477101)

[9 测试报告 34](#_Toc419477102)

[10 自动化测试 35](#_Toc419477103)

[11 常见问题 36](#_Toc419477104)

12 附件 37

# 概述

## 编写目的

本文档规范了测试过程中的各项内容，从软件测试基础（流程、分类及方法）、测试计划、测试需求、测试案例设计、测试执行、缺陷管理、自动化测试、性能测试、测试报告等方面着手，为展开测试工作提供了一定的指导作用。

## 读者对象

本文档预期读者为：

参与项目的业务需求分析人员；

参与项目的系统分析、设计和开发人员；

参与项目的测试人员；

参与项目的QA人员。

## 阅读建议

通过本文档的学习了解，我们应熟知以下知识：

熟悉软件测试的概念、目的、原则、分类、流程以及方法；

掌握测试计划、测试需求、测试案例的设计方法；

了解测试执行过程注意事项；

熟悉缺陷分类及严重等级，学会提交优秀的缺陷报告并掌握缺陷管理流程；

了解自动化测试、性能测试；

提高自身测试知识的积累。

# 理论基础

## 正确认识测试

软件开发中出现错误或缺陷的现象频频存在，市场对软件质量重要性的认识逐渐增强。所以，软件测试在软件项目实施过程中的重要性日益突出。但是，现实情况是，与软件编程比较，软件测试的地位和作用，还没有真正受到重视，对于很多人（甚至是软件项目组的技术人员）还存在对软件测试的认识误区，这进一步影响了软件测试活动的开展和软件测试质量的提高。

每当测试一个程序时，人们总希望为程序增加一些价值。利用测试来增加程序的价值，是指通过测试，尽可能多的找出并修改程序缺陷，从而提高程序的可靠性或质量。因此，不要只是为了证明程序能够正确运行而去测试程序。相反，应该一开始就假设程序中隐藏着错误（这种假设几乎对所有的程序都成立），然后测试程序，发现尽可能多的错误。

软件测试（software testing），描述一种用来促进鉴定软件的正确性、完整性、安全性和质量的过程。换句话说，软件测试是一种实际输出与预期输出间的审核或者比较的过程。软件测试的经典定义是：在规定的条件下对程序进行操作，以发现程序错误或缺陷，衡量软件质量，并对其是否能满足设计要求进行评估的过程。

IEEE729-1983对缺陷有一个标准的定义：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失效或违背。

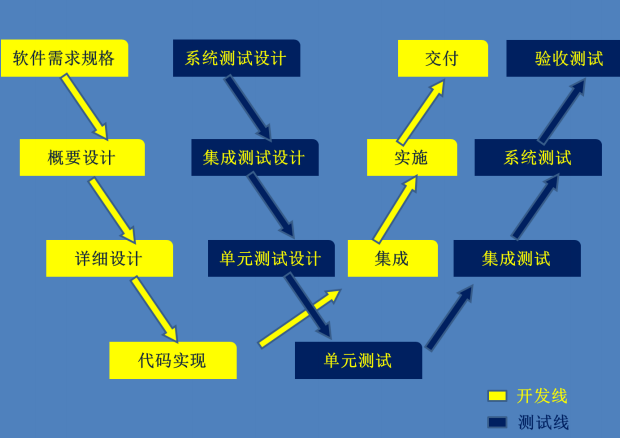
## 测试的原则

* 尽早和不断的测试；
* 所有测试标准都应建立在用户需求之上；
* 程序员应该避免测试自己的程序，可以采用同行评审及交叉测试等方式进行。
* 不仅要检验程序是否做了应该做的事，还要检查它是否做了它不应该做的事；
* 设计测试用例时应该考虑到合法的输入和不合法的输入以及各种边界条件，特殊情况下要制造极端状态和意外状态；
* 要充分注意测试中的错误集群发生现象，程序某部分存在问题的可能性与已发现问题的个数成正比。如某部分程序发现较多的错误，应该增加额外的测试；
* 避免测试的随意性，随意性测试无法保证测试质量；
* 回归测试的关联性一定要引起充分的注意，修改一个错误而引起更多的错误出现的现象并不少见；
* 计划测试工作时，不应默许假定不会发现错误；
* 穷举测试是不可能的，测试需要终止；在测试成本、收益和风险之间求得平衡；
* 测试执行之前应明确预期结果，作为实际运行结果的对比参照；
* 妥善保存一切测试过程文档。

## 测试与开发的关系

软件开发和测试的关系可以用“双 v 模型”来表示：图中黄色代表开发线、蓝色的是测试线。这样的模型相比传统的“需求-设计-编码-测试-交付”的单一“瀑布式”的模型的好处是：

* + 测试活动的尽早开展可以极大减少需求规格说明书、概要和详细设计文档中存在的问题
  + 单元测试和代码实现同步进行、强调开发人员自测并保证集成测试可执行
  + 开发在实施阶段集中处理系统测试发现的 Bug 从而保证交付产品的质量、并且能与需求和设计相互对应，方便总结 Bug



## 测试分类

### 按是否关注代码划分

#### 白盒测试

软件的白盒测试是对软件的过程性细节做细致的检查。这种方法是把测试对象看作一个打开的盒子，它允许测试人员利用程序内部的逻辑结构及有关信息，设计或选择测试用例，对程序所有逻辑路径进行测试。通过在不同点检查程序状态，确定实际状态是否与预期的状态一致。因此白盒测试又称为结构测试或逻辑驱动测试。设计出的测试用例要保证在测试中程序的每个可执行语句至少执行一次。

#### 黑盒测试

黑盒测试也称功能测试，它是通过测试来检测每个功能是否都能正常使用。在测试中，把程序看作一个不能打开的黑盒子，在完全不考虑程序内部结构和内部特性的情况下，在程序接口进行测试，它只检查程序功能是否按照需求规格说明书的规定正常使用，程序是否能适当地接收输入数据而产生正确的输出信息。黑盒测试着眼于程序外部结构，不考虑内部逻辑结构，主要针对软件界面和软件功能进行测试。

黑盒测试方法主要有等价类划分、边值分析、因—果图、错误推测法等。

#### 灰盒测试

灰盒测试，是介于白盒测试与黑盒测试之间的，可以这样理解，灰盒测试关注输出对于输入的正确性，同时也关注内部表现，但这种关注不像白盒那样详细、完整，只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，有时候输出是正确的，但内部其实已经错误了，这种情况非常多，如果每次都通过白盒测试来操作，效率会很低，因此需要采取这样的一种灰盒的方法。

### 按是否执行程序划分

#### 静态测试

静态测试是指不运行被测程序本身，仅通过分析或检查源程序的语法、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。

静态测试包括代码检查、静态结构分析、代码质量度量等。

#### 动态测试

动态测试是指通过运行被测程序，检查运行结果与预期结果的差异，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。

### 按软件开发的阶段划分

#### 单元测试

单元测试的对象是软件设计的最小单位——模块。单元测试是在软件开发过程中要进行的最低级别的测试活动，软件的独立单元将在与程序的其他部分相隔离的情况下进行测试。

单元测试的依据是详细设计描述，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。单元测试多采用白盒测试技术，系统内多个模块可以并行地进行测试。

#### 集成测试

集成测试，也叫组装测试或联合测试。在单元测试的基础上，将所有模块按照设计要求（如根据结构图）组装成为子系统或系统，进行集成测试。它最简单的形式是：把两个已经测试过的单元组合成一个组件，测试它们之间的接口。

常用测试方法有自顶向下测试、自底向上测试、核心系统测试、高频集成测试。

#### 确认测试

确认测试的目的是向未来的用户表明系统能够像预定要求那样工作。经集成测试后，已经按照设计把所有的模块组装成一个完整的软件系统，接口错误也已经基本排除了，接着就应该进一步验证软件的有效性，这就是确认测试的任务，即软件的功能和性能如同用户所期待的那样。

#### 系统测试

系统测试是将已经确认的软件、计算机硬件、外设、网络等其他元素结合在一起，进行信息系统的各种组装测试和确认测试，系统测试是针对整个产品系统进行的测试，目的是验证系统是否满足了需求规格的定义，找出与需求规格不符或与之矛盾的地方，从而提出更加完善的方案。

系统测试分类：恢复测试、安全测试、强度测试、性能测试。

#### 回归测试

回归测试是指代码修改后，重新进行测试以确认修改没有引入新的错误或导致其他代码产生错误。

回归测试我们通常称之为复测，但两者并不是一个概念。复测是以确认曾发现的缺陷已经被修正为目的而再次进行的测试，有一条被发现的缺陷，就有一条与之对应的复测。回归测试则需要执行构建(build)前执行过的所有测试用例，有时也包括新增加或改进后的用例，是为了防止修改后引起一些旧版本中本不存在的缺陷而进行的全面测试。因此，回归测试比复测工作量大很多，在一般的以手工测试为主的测试中无法进行较全面的回归测试，需要借助自动化测试技术来实现。

已修正的缺陷

修改后新增的缺陷

回归测试范围

复测范围

另外除了自动化测试技术我们还可以通过静态代码审查来确认修改后是否会引发新的缺陷。

#### 验收测试

验收测试是部署软件之前的最后一个测试操作。它是技术测试的最后一个阶段，也称为交付测试。验收测试是向未来的用户表明系统能够像预定要求那样工作。

通常可以包括：安装（升级）、启动与关机、功能测试（正例、重要算法、边界、时序、反例、错误处理）、性能测试（正常的负载、容量变化）、压力测试（临界的负载、容量变化）、配置测试、平台测试、安全性测试、恢复测试（在出现掉电、硬件故障或切换、网络故障等情况时，系统是否能够正常运行）、可靠性测试等。

* + Alpha测试：α测试是指软件开发公司组织内部人员模拟各类用户行对即将面市软件产品（称为α版本）进行测试，试图发现错误并修正。
  + Beta测试：β测试是一种验收测试。β测试是指软件开发公司组织各方面的典型用户在日常工作中实际使用β版本，并要求用户报告异常情况、提出批评意见。

### 按测试目的划分

#### 易用性测试

易用性测试是指用户使用软件时是否感觉方便，比如是否最多点击鼠标三次就可以达到用户的目的。测试内容包括（1）易理解性；（2）易学习性；（3）易操作性；（4）吸引性；（5）依从性。

#### 功能测试

功能测试就是对产品的各功能进行验证，根据功能测试用例，逐项测试，检查产品是否达到用户要求的功能。（参考黑盒测试）

#### 兼容性测试

指对所设计程序与硬件、软件之间的兼容性的测试。

兼容性测试分类主要可以分为三大类：硬件兼容性测试、软件兼容性测试、数据兼容性测试。

#### 性能测试

性能测试是通过自动化的测试工具模拟多种正常、峰值以及异常负载条件来对系统的各项性能指标进行测试。负载测试和压力测试都属于性能测试，两者可以结合进行。

一般性能测试中最常见的基本类型有基准测试、配置测试，负载测试，压力测试等，以下是各种测试类型的具体概念：

* + 基准测试——把新服务器或者未知服务器的性能和已知的参考标准进行比较
  + 配置测试——确认服务器在不同的配置下性能的可接受性。（操作条件不变）
  + 负载测试——确认服务器在不同的负载条件下性能的可接受性。（操作条件不变）
  + 压力测试——确认服务器在异常或者极限的条件时性能的可接受性，例如，减少资源或大数量的用户。
  + 竞争测试——确认服务器可以处理多个客户对同一个资源的请求竞争
  + 容量测试——是在数据库中有较大数量的数据记录情况下对系统进行的测试。
  + 渗入测试——使用固定数目的并发用户测试系统的总体健壮性。
  + 峰谷测试——不断增加一些新用户直至达到固定数目的用户的测试方法，其目标是确定从高负载（例如系统高峰时间的负载）恢复、转为几乎空闲、然后再攀升到高负载、再降低的能力。

#### 安全测试

安全测试是在IT软件产品的生命周期中，特别是产品开发基本完成到发布阶段，对产品进行检验以验证产品符合安全需求定义和产品质量标准的过程 。测试方法：

* + 模式匹配方法：将程序看作字符串
  + 状态机模型：将程序看作状态机
  + 黑盒模型：将程序看作黑盒子
  + 白盒模型：将程序看作路径的组合

## 常用测试方法

### 等价类划分

等价类划分是把所有可能输入的数据，即程序的输入域划分成若干部分（子集），然后从每一个子集中选取少数具有代表性的数据作为测试用例。等价类划分是一种重要的、常用的黑盒测试用例设计方法。

#### 划分方法

在输入条件规定了取值范围或值的个数的情况下，则可以确立一个有效等价类和两个无效等价类。如：输入值是学生成绩，范围是0~100；



* + 在输入条件规定了输入值的集合或者规定了“必须如何”的条件的情况下，可确立一个有效等价类和一个无效等价类。
  + 在输入条件是一个布尔量的情况下，可确定一个有效等价类和一个无效等价类。
  + 在规定了输入数据的一组值（假定n个），并且程序要对每一个输入值分别处理的情况下，可确立n个有效等价类和一个无效等价类。例：输入条件说明学历可为：专科、本科、硕士、博士四种之一，则分别取这四种的四个值作为四个有效等价类，另外把四种学历之外的任何学历作为无效等价类。
  + 在规定了输入数据必须遵守的规则情况下，可确立一个有效等价类（符合规则）和若干个无效等价类（从不同角度违反规则）。
  + 在确知已划分的等价类中各元素在程序处理中的方式不同的情况下，则应在将该等价类进一步的划分为更小的等价类。
  + 使用等价划分方法设计测试用例主要有两个步骤：（1）确定等价类；（2）生成测试用例；

|  |  |  |
| --- | --- | --- |
| 外部条件 | 有效等价类 | 无效等价类 |
|  |  |  |

#### 确定等价类

确定等价类是选取每一个输入条件（通常是需求规格说明中的一个句子或短语）并将其划分为两个或更多的组。可以使用等价类列举表来进行划分。注意，我们确定了两类等价类：有效等价类代表对程序的有效输入，而无效等价类代表的则是其他任何可能的输入条件（即不正确的输入值）。注意无效和未预料到的输入情况；举例如下表；

|  |  |  |
| --- | --- | --- |
| **外部条件** | **有效等价类** | **无效等价类** |
| 审批权限为：1-1000万（包含） | 1万<=金额<=1000万 | 金额<1万，金额>1000万 |
| 个贷部门只能发起个贷产品 | 产品属性适用部门有个贷部的产品 | 产品属性适用部门不是个贷部的 |
| 邮编必须是6位数字 | 6位数字的任意组合 | 不是6位数字的组合,比如：7位数字组合，组合中有字母，组合中有符号等 |

#### 生成测试用例

* + 为每个等价类设置一个不同的编号。
  + 编写新的测试用例，尽可能多地覆盖那些尚未被涵盖的有效等价类，直到所有的有效等价类都被测试用例所覆盖（包含进去）。
  + 编写新的用例，覆盖一个且仅一个尚未被覆盖的无效等价类，直到所有的无效等价类都被测试用例所覆盖。

### 边界值分析

使用边界值分析方法设计测试用例时一般与等价类划分结合起来。但它不是从一个等价类中任选一个例子作为代表，而是将测试边界情况作为重点目标，选取正好等于、刚刚大于或刚刚小于边界值的测试数据, 经验证明，考虑了边界条件的测试用例与其他没有考虑边界条件的测试用例相比，具有更高的测试回报率。

* + 如果输入条件规定了值的范围，可以选择正好等于边界值的数据作为合理的测试用例，同时还要选择刚好越过边界值的数据作为不合理的测试用例。如输入值的范围是[1，100]，可取0，1，100，101 等值作为测试数据。
  + 如果输入条件指出了输入数据的个数，则按最大个数、最小个数、比最小个数少1、比最大个数多1等情况分别设计测试用例。如，一个输入文件可包括1~255个记录，则分别设计有1个记录、255个记录，以及0个记录的输入文件的测试用例。
  + 对每个输出条件分别按照以上原则(1)或(2)确定输出值的边界情况。如，一个学生成绩理系统规定，只能查询 95~98 级大学生的各科成绩，可以设计测试用例，使得查询范围内的某一届或四届学生的学生成绩，还需设计查询 94 级、99 级学生成绩的测试用例(不合理输出等价类)。
  + 由于输出值的边界不与输入值的边界相对应，所以要检查输出值的边界不一定可能，要产生超出输出值之外的结果也不一定能做到，但必要时还需试一试。
  + 如果程序的规格说明给出的输入或输出域是个有序集合(如顺序文件、 线形表、 链表等)，则应选取集合的第一个元素和最后一个元素作为测试用例。

### 错误推测

错误推测法是指：在测试程序时，人们可以根据经验或直觉推测程序中可能存在的各种错误，从而有针对性地编写检查这些错误的测试用例的方法。

错误推测方法的基本思想：列举出程序中所有可能有的错误和容易发生错误的特殊情况，根据他们选择测试用例。例如，在单元测试时曾列出的许多在模块中常见的错误。以前产品测试中曾经发现的错误等，这些就是经验的总结。还有，输入数据和输出数据为0的情况。输入表格为空格或输入表格只有一行。这些都是容易发生错误的情况。可选择这些情况下的例子作为测试用例。

### 因果图

从用自然语言书写的程序规格说明的描述中找出因（输入条件）和果（输出或程序状态的改变），可以通过因果图转换为判定表。

因果图法是一种适合于描述对于多种输入条件组合的测试方法，根据输入条件的组合、约束关系和输出条件的因果关系，分析输入条件的各种组合情况，从而设计测试用例的方法，它适合于检查程序输入条件涉及的各种组合情况。

因果图法着重分析输入条件的各种组合，每种组合条件就是“因”，它必然有一个输出的结果，这就是“果”。

### 综合策略

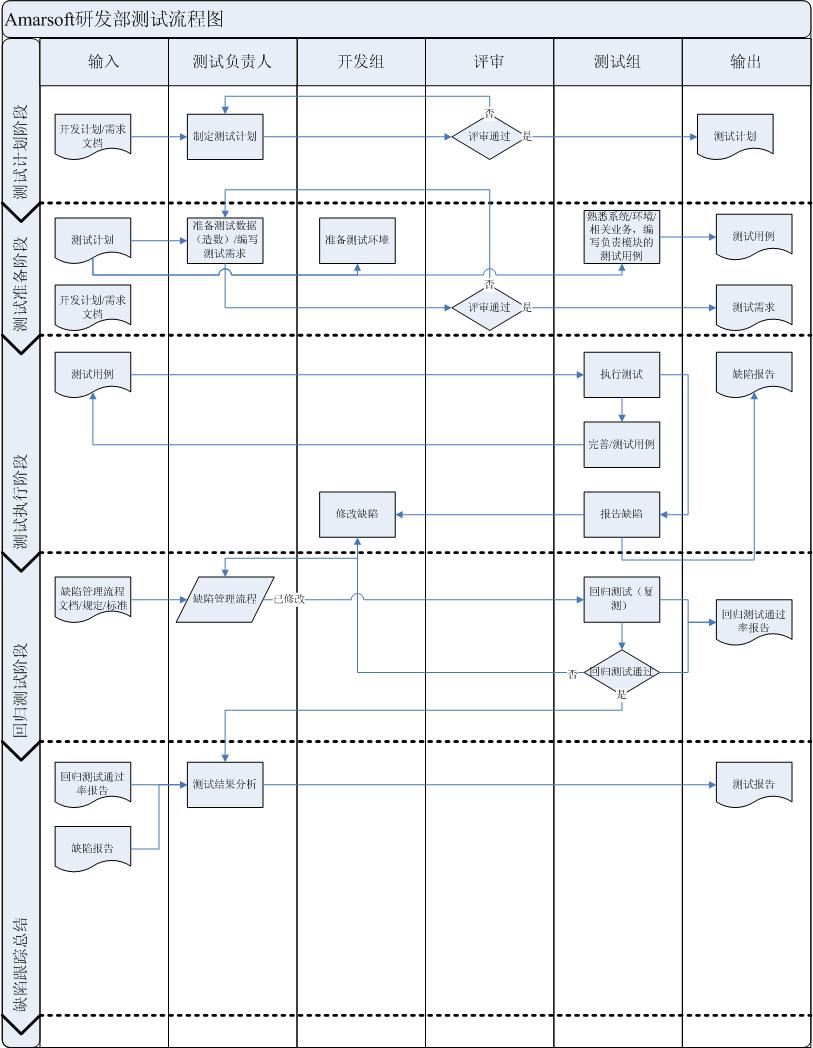
每种方法都能设计出一组有用例子，用这组例子容易发现某种类型的错误，但可能不易发现另一类型的错误。因此在实际测试中，联合使用各种测试方法，形成综合策略，通常先用黑盒法设计基本的测试用例，再用白盒法补充一些必要的测试用例。

### 案例说明



# amarsoft测试流程

## 流程图



## 阶段说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **阶 段** | **参与人员角色** | **主要内容** | **输入文档** | **输出文档** |
| 需求评审阶段 | 项目经理、需求组、开发经理、开发人员、测试经理、测试人员 | 召开会议，确认所有人员对需求的理解正确，无异议 | 需求规格说明书草拟版 | 需求规格说明书最终版本 |
| 测试计划阶段 | 项目经理、开发经理、测试经理、测试人员 | 测试经理根据需求、项目实际情况，编写相关测试计划且通过评审，评审人员一般为项目经理和开发经理，确认测试计划的正确性、合理性和可执行性 | 需求规格说明书  开发计划表 | 测试计划表（总体计划、详细计划） |
| 测试准备阶段 | 项目经理、测试经理、测试人员 | 测试经理根据测试计划，进行人员工作安排，小组分工  测试环境搭建、数据准备  测试经理安排测试人员编写测试需求、测试用例且通过评审 | 测试计划表  需求规格说明书最终版本  概要设计设计书（重要，非必须，可以由完整的需求规格说明书覆盖） | 测试需求  测试用例 |
| 测试执行阶段 | 测试经理、测试人员 | 测试执行阶段分为：单元测试、集成测试、系统测试、验收测试，每个阶段可按照实际情况进行多轮测试  测试人员向测试经理或测试组长汇报阶段性测试成果  测试经理向项目经理汇报阶段性成果  项目经理向其他更高级领导或客户汇报阶段性测试成果 | 测试需求  测试用例  需求规格说明书 | 阶段性测试报告（可分为日报、周报、月报等，也可为模块报或轮次报）  缺陷登记表 |
| 回归测试阶段 | 需求组、开发经理、开发人员、测试经理、测试人员 | 回归测试阶段主要为缺陷的流转，涉及较多人员。还有范围较广的回归测试，可设计为第二轮测试 | 缺陷登记表  测试需求  测试用例  需求规格说明书 | 阶段性测试报告  缺陷登记表 |
| 缺陷跟踪总结 | 项目经理、需求组、开发经理、开发人员、测试经理、测试人员 | 根据回归测试的阶段性报告，测试经理整合为最终的测试报告，交付项目经理、更高级领导或客户 | 缺陷登记表  阶段性测试报告 | 缺陷登记表  测试报告完整版 |

## 准入准出标准

### 测试执行开始标准

* 测试计划评审通过
* 测试用例已编写完成，并已通过评审；
* 存在已提交的可测试的系统；
* 测试环境已搭建完毕。

### 测试退出标准

* 软件系统经过单元、集成、系统测试，分别达到单元、集成、系统测试停止标准；
* 测试用例全部通过；
* 存在的问题已得到合理的处理。（致命错误与严重错误应达到100%（如对项目进度、测试范围等影响较大，需汇报至项目经理决定），一般错误与低级错误修复率应达到95%以上（需保证不影响正常功能使用），优化错误修复率应达到70%以上，总体缺陷修复率达到90%（暂定90%，后续根据产品要求可调整））

### 测试暂停标准

* 在进行软件系统测试时，发现程序存在重大bug（影响基本功能性的）或bug过多时，测试无法正常进行，可向领导申请暂停测试；
* 存在其他优先级更高任务时，可向领导申请暂停测试；
* 软件项目需暂停以进行调整时，测试应随之暂停，并备份暂停点数据；
* 软件项目在其开发生命周期内出现重大估算、进度偏差，需暂停或终止时，测试应随之暂停或终止，并备份暂停或终止点数据。

### 测试恢复标准

* 重大bug被解决或程序通过重新修正；
* 优先级更高的任务已经被完成；
* 软件项目被调整后重新启动，测试任务应随之启动。

### 各阶段始终准则

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **阶段** | **进入准则** | **暂停准则** | **恢复准则** | **退出准则** |
| 单元测试 | 测试环境准备就绪。 | 测试环境被破坏。 | 测试环境重新搭建好。 | 1. 按照单元测试计划完成了所有规定单元的测试 2. 达到了测试计划中关于单元测试所规定的覆盖率的要求 3. 被测试的单元每千行代码必须发现至少3个错误（不含五级错误） 4. 软件单元功能与设计一致 5. 在单元测试中发现的错误已经得到修改，各级缺陷修复率达到标准 |
| 本单元代码类编写完成。 | 需求理解错误，实现与需求完全不一致；  有重大缺陷，严重影响后续操作。 | 实现与需求基本一致；  缺陷修正。 |
| 单元测试计划通过评审；  测试所需文档已准备就绪。 |  |  |
| 集成测试 | 测试环境已经准备好。 | 测试环境被破坏。 | 测试环境重新搭建好。 | 1. 按照集成构件计划及增量集成策略完成了整个系统的集成测试 2. 达到了测试计划中关于集成测试所规定的覆盖率的要求 3. 被测试的集成工作版本每千行代码必须发现至少2个错误（不含五级错误） 4. 集成工作版本满足设计定义的各项功能、性能要求 5. 在集成测试中发现的错误已经得到修改，各级缺陷修复率达到标准 |
| 已经完成提交测试的模块内容；  主要功能无页面点击错误。 | 主要功能页面点击错误。 | 主要功能不会出现页面点击错误的情况。 |
| 单元测试已经完成。 |  |  |
| 测试所需的文档资料已经完整。 |  |  |
| 系统测试 | 测试环境已经准备好。 | 测试环境被破坏。 | 测试环境重新搭建好。 | 1. 阻塞测试的内容（即测试暂停的产生原因）在短时间内无法解决。 2. 按照系统测试计划完成了系统测试； 3. 达到了测试计划中关于系统测试所规定的覆盖率的要求，测试用例均已执行，用例通过率90%； 4. 功能需求都已经开发和测试完成，系统性能和功能已达到需求标准； 5. 在系统测试中发现的错误已经得到修改，各级缺陷修复率达到标准（若有遗留由项目经理和产品经理审批）、一、二级错误修复率应达到100%，三、四级错误修复率应达到95%以上，五级错误修复率应达到60%以上，总体缺陷修复率达到80%（暂定80%，后续根据产品要求可提高）。 |
| 系统基本业务流程能走通。 | 系统基本业务流程不通。 | 系统基本业务流程可以走通。 |
| 无任何功能的页面点击错误。 | 任何功能的页面点击错误。 | 页面点击错误问题解决。 |
| 测试所需的文档资料已经完整。 |  |  |
| 集成测试工作已经完成。 |  |  |
| 内部确认测试/UAT | 测试环境已经准备好。 | 测试环境被破坏。 | 测试环境重新搭建好。 | 测试内容已经全部完成。 |
| 系统正常功能已正确实现。 | 系统业务流程不通。 | 系统业务流程能走通。 | PM根据测试报告，认为系统可以满足客户的要求。 |
| 业务流程能走通。 | 正常功能未正确实现。 | 正常功能实现。 | PM要求修改的缺陷已经全部修复。 |
|  | 用户很容易重现的严重缺陷产生。 | 需要解决的缺陷解决。 | 到了时间，系统必须发布。 |
| 验收测试 | 测试环境已经准备好。 | 测试环境被破坏。 | 测试环境重新搭建好。 | 1. 软件需求分析说明书中定义的所有功能已全部实现，性能指标全部达到要求。 2. 在验收测试中发现的错误已经得到修改，各级缺陷修复率达到标准 3. 所有测试项没有残余一级、二级、三级和四级错误。 4. 需求分析文档、设计文档和编码实现一致。 5. 验收测试工件齐全（测试计划、测试用例、测试日志、测试通知单、测试分析报告，待验收的软件安装程序。） |
| 客户要求的功能都已经完成。 | 发现需要修改的缺陷。 | 修改完需要修改的缺陷。 |
| 业务流程可以走通。 |  |  |
| 性能测试 | 测试环境已经准备好； | 测试环境被破坏； | 测试环境重新搭建好； | 所有要求的测试用例和测试脚本都已执行； |
| 系统的功能正常实现； | 系统流程存在缺陷； | 解决影响性能测试的缺陷。 | 完成性能分析工作。 |
| 不存在影响系统流程的缺陷。 | 被测试功能存在缺陷； |  |  |
|  | 程序的版本更新，存在影响系统功能实现的缺陷。 |  |  |

# 测试计划

## 概述

软件测试计划是指导测试过程的纲领性文件，包含了产品概述、测试策略、测试方法、测试区域、测试配置、测试周期、测试资源、测试交流、风险分析等内容。借助软件测试计划，参与测试的项目成员，尤其是测试管理人员，可以明确测试任务和测试方法，保持测试实施过程的顺畅沟通，跟踪和控制测试进度，应对测试过程中的各种变更。

## 原则

做好软件的测试计划不是一件容易的事情，需要综合考虑各种影响测试的因素。为了做好软件测试计划，需要注意以下几个方面。

* + **明确测试的目标，增强测试计划的实用性：**测试目标必须是明确的，可以量化和度量的，而不是模棱两可的宏观描述。另外，测试目标应该相对集中，避免罗列出一系列目标，从而轻重不分或平均用力。根据对用户需求文档和设计规格文档的分析，确定被测软件的质量要求和测试需要达到的目标。
  + **坚持“5W”规则，明确内容与过程：**利用“5W”规则创建软件测试计划，可以帮助测试团队理解测试的目的（Why），明确测试的范围和内容（What），确定测试的开始和结束日期（When），指出测试的方法和工具（How），给出测试文档和软件的存放位置（Where）。在需要测试的内容里面突出关键部分，可以列出关键及风险内容、属性、场景或者测试技术。对测试过程的阶段划分、文档管理、缺陷管理、进度管理给出切实可行的方略。
  + **采用评审和更新机制，保证测试计划满足实际需求**:在创建完测试计划后，提交到由项目经理、开发经理、测试经理等组成的评审委员会审阅，根据审阅意见和建议进行修正。
  + **分别创建测试计划与需求规格、测试用例:**最好的方法是把详细的测试技术指标包含到独立创建的测试需求规格文档，把测试步骤放到独立创建的测试用例文档中。测试计划和测试需求规格、测试用例之间是战略和战术的关系，测试计划主要从宏观上规划测试活动的范围、方法和资源配置，而测试需求规格、测试用例是完成测试任务的具体战术。

测试计划模板可参考附件：AMARSOFT集中开发\_XX系统\_测试计划模板\_YYYYMMDD\_v1.0.doc。

# 测试需求

## 概述

确切地讲，所谓的测试需求就是在项目中要测试什么（参考测试计划的“5W”规则）。测试需求是测试计划的基础与重点。

详细的测试需求可以帮助了解测试规模、复杂程度与可能存在的风险。测试需求不明确，会造成测试人员无法完全把握所测软件预期实现的功能点。测试需求越详细精准，表明对所测软件的了解越深，对所要进行的任务内容就越清晰，就更有把握保证测试的质量与进度。

## 依据与收集

测试需求通常是以待测对象的软件需求为原型进行分析而转变过来的。但测试需求并不等同于软件需求，它是以测试的观点根据软件需求整理出一个checklist，作为测试该软件的主要工作内容。 测试需求主要通过以下途径来收集：

* + 与待测软件相关的各种文档资料。如软件需求规格、Use case、界面设计、项目会议或与客户沟通时有关于需求信息的会议记录、其他技术文档等。
  + 与客户或系统分析员的沟通。
  + 如果以旧系统为原型，以全新的架构方式来设计或完善软件，那么旧系统的原有功能跟特性就成为了最有效的测试需求收集途径。
  + 在整个信息收集过程中，务必确保软件的功能与特性被正确理解。因此，测试需求分析人员必须具备优秀的沟通能力与表达能力。

## 测试需求分析方法

测试需求需要考虑几个层面的因素：

* + **第一层：测试阶段：**
    - 系统测试阶段,需求分析更注重于技术层面，即软件是否实现了具备的功能。如果某一种流程或者某一角色能够执行一项功能，那么我们相信具备相同特征的业务或角色都能够执行该功能。为了避免测试执行的冗余，可不再重复测试。
    - 验收测试阶段，更注重于不同角色在同一功能上能否走通要求的业务流程。因此需要根据不同的业务需要而测试相同的功能，以确保系统上线后不会有意外发生。但是否有必要进行这种大量的重复性质的测试，不过也是见仁见智的做法，要看测试管理者对测试策略与风险的平衡能力了。
    - 大多数的测试都会在系统测试中完成，验收测试只是对于系统测试的回归。此种情况也是合理的，关键看测试周期与资源是否允许，以及各测试阶段的任务划分。
  + **第二层：待测软件的特性：**不同的软件业务背景不同，所要求的特性也不相同，测试的侧重点自然也不相同。除了需要确保要求实现的功能正确，银行/财务软件更强调数据的精确性，网站强调服务器所能承受的压力，ERP强调业务流程，驱动程序强调软硬件的兼容性。在做测试分析时需要根据软件的特性来选取测试类型，并将其列入测试需求当中。
  + **第三层：测试的焦点：**测试的焦点是指根据所测的功能点进行分析、分解，从而得出的着重于某一方面的测试，如界面、业务流、模块化、数据、输入域等。目前关于各个焦点的测试也有不少的指南，那些已经是很好的测试需求参考了，在此仅列出业务流的测试分析方法。

## 优先级的确定

优先级别的确定，利于测试工作有的放矢的展开，使测试人员清晰了解核心的功能、特性与流程有哪些，客户最为关注的是什么，由此可确定测试的工作重点在何处，更方便处理测试进度发生问题时，实现不同优先级别的功能、模块、系统等迭代递交或取舍，从而缓和测试风险。

测试需求模板请参考附件：AMARSOFT集中开发\_XX系统\_测试需求模版\_YYYYMMDD\_v1.0.xls。

# 测试用例设计

测试用例（Test Case），指对一项特定的软件产品进行测试任务的描述，体现测试方案、方法、技术和策略。内容包括测试目标、测试环境、输入数据、测试步骤、预期结果、测试脚本等，并形成文档。

测试用例的设计，可根据常用的测试方法进行设计，如基本路径测试法、覆盖法、等价类划分、边界值分析、错误推测、因果图、综合策略等，详情请参考Amarsoft\_集中开发中心\_测试方法实例。

测试用例模板请参考附件：AMARSOFT集中开发\_XX系统\_测试用例模版\_YYYYMMDD\_v1.0.xlsx。

# 测试执行

## 测试准备

测试准备工作在测试过程中展现了非常大的作用，如准备工作未考虑完善，对后续测试工作将造成严重后果。如下展示常见几类，详细可参考Amarsoft\_集中开发中心\_测试工作准备V1.0.docx。

* + 文档类：需求文档、测试计划表、测试需求、测试用例文档、缺陷登记表
  + 环境类：
    - 操作系统：Windws xp、Windws 7、Linux等
    - 浏览器：IE、Firefox等
    - 开发环境：JDK+Tomcat+Eclipse
    - 数据库：squirrel-sql
    - BUG管理工具：QC、Bugzilla等
  + 其他类：号码生成器、系统用户角色表、与核心交互的客户表（有可用账户）等

## 测试执行

### 操作准则

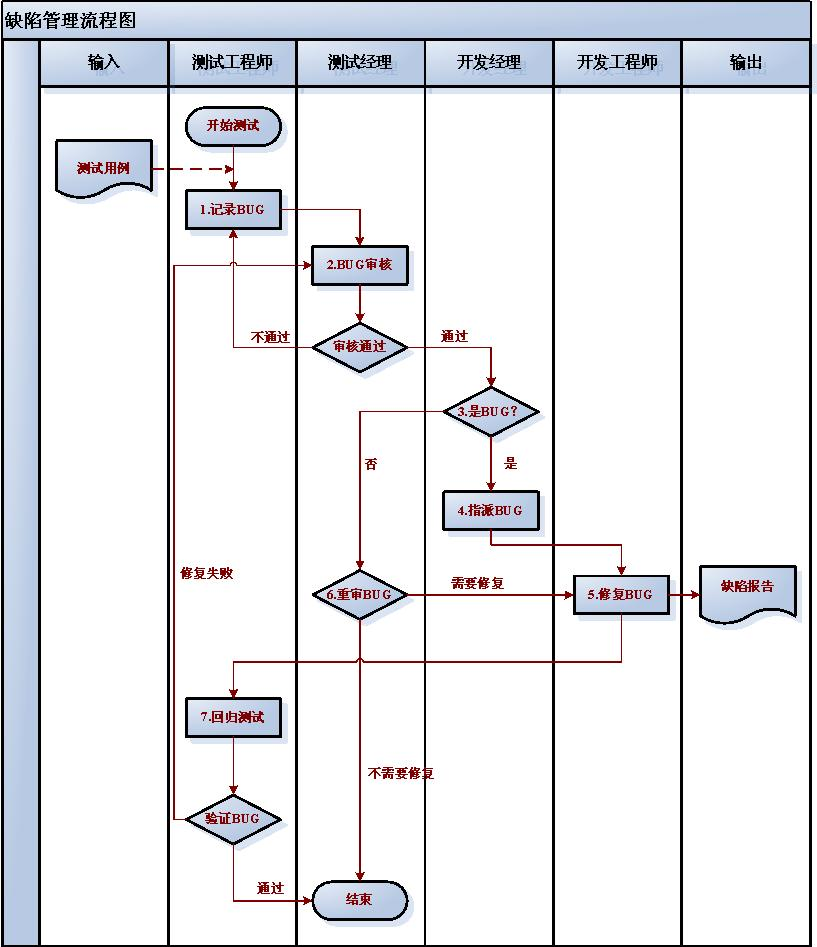
当测试用例编写完成，并通过审核后，就进入到软件测试最主要的阶段，就是执行测试用例，进行软件测试。不过在执行测试是有几个地方需要注意：

* + **检查软件测试环境是否搭建成功**:执行测试前要按照测试用例中描述的测试环境去搭建，因为测试用例中的执行都是建立在这个测试环境之上的，如果测试环境的不一致，会影响测试用例的执行，和测试目的的证明。
  + **注意测试用例中的前提条件和特殊规程说明**：因为有些测试软件是有顺序性的，那么它的测试用例就会有一些执行前提或特殊说明。如果前提条件和特殊说明没有注意，会导致测试用例的无法执行。
  + **严格保证每条用例至少执行一遍：**因为编写测试用例时，考虑了测试覆盖率的问题，每条测试用例都对应一个功能点，如果少执行一条，就会有一个功能点没有测试到，只有每个功能点我们都测试一遍，才能保证待测试软件能正确满足用户需求。
  + **要求必须要做测试记录：**测试过程中要求必须记录测试数据和操作情况，不仅限于案例结果有错误的，而是所有执行过的案例数据，便于后续统计、跟踪或者数据重用等。
  + **准确、详细记录软件系统的实际输入输出：**仔细对比实际输出和测试用例中预期输出是否一致。如果不一致，要从多个角度多测试几次，尽量详细的定位软件出错的位置和原因，并测试出因为这个错误会不会导致更严重的错误出现，最后把详细的输入和实际的输出，以及对问题的描述写到测试报告中。在一个项目组中，项目的开发时间是有限的，如果我们测试时能把问题描述的详细一些，那么开发人员就会很容易的重现这个问题，也就能更快的解决问题，节省项目时间。
  + **不要放过任何偶然现象：**我们在测试时，有时会发现某条用例执行时，软件会出错，但是当再次执行时这个错误就不再重现，要求出现此情况后，及时反馈给相关人员，及时记录跟踪此段时间日志，并及时通过多种方式进行模拟重现，如果无法重现，把相应的日志提交专家进行程序分析。

### 缺陷管理

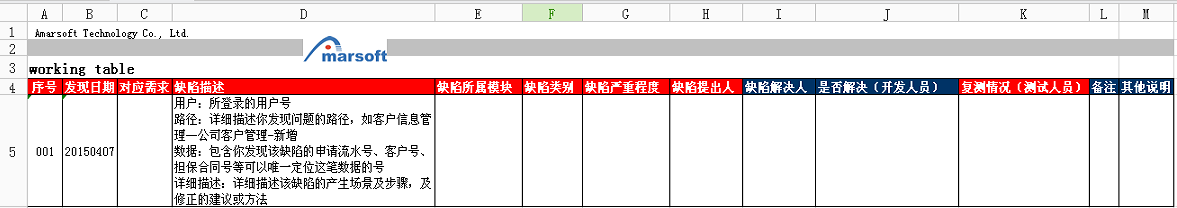
#### 缺陷管理流程

如下展示基础测试管理流程，模板请参考附件：Amarsoft\_集中开发中心\_缺陷管理流程V1.0.doc



#### 缺陷登记

缺陷登记模板请参考附件：AMARSOFT集中开发\_XX系统\_缺陷登记模板\_YYYYMMDD\_姓名.xlsx。



#### 缺陷分类

缺陷种类：根据缺陷的自然属性来划分。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **编号** | **缺陷类型** | **描述** | **子类型** | |
| **编号** | **名称** |
| 01 | 功能问题  F- Function | 影响了重要的特性、用户界面、产品接口、硬件结构接口和全局数据结构。并且设计文档需要正式的变更。如指针，循环，递归，功能等缺陷。 | 0101 | 功能错误 |
| 0102 | 功能缺失 |
| 0103 | 功能超越 |
| 0104 | 设计二义性 |
| 0105 | 算法错误 |
| 02 | 接口问题  I-Interface | 与其他组件、模块或设备驱动程序、调用参数、控制块或参数列表相互影响的缺陷。 | 0201 | 模块间接口 |
| 0202 | 模块内接口 |
| 0203 | 公共数据使用 |
| 03 | 逻辑问题  L-Logic | 需要进行逻辑分析，进行代码修改，如循环条件等 | 0301 | 分支不正确 |
| 0302 | 重复的逻辑 |
| 0303 | 忽略极端条件 |
| 0304 | 不必要的功能 |
| 0305 | 误解 |
| 0306 | 条件设置错误 |
| 0307 | 循环不正确 |
| 0308 | 错误的变量检查 |
| 0309 | 计算顺序错误 |
| 0310 | 逻辑顺序错误 |
| 04 | 计算问题  C-Computation | 等式、符号、操作符或操作书错误，精度不够、不适当的数据验证等缺陷。 | 0401 | 等式错误 |
| 0402 | 缺少运算符 |
| 0403 | 错误的操作数 |
| 0404 | 括号用法不正确 |
| 0405 | 精度不够 |
| 0406 | 舍入错误 |
| 0407 | 符号错误 |
| 05 | 数据问题  A-Assignment | 需要修改少量代码，如初始化或控制块。如声明、重复命名，范围、限定等缺陷。 | 0501 | 初始化错误 |
| 0502 | 存取错误 |
| 0503 | 引用错误的变量 |
| 0504 | 数组引用越界 |
| 0505 | 不一致的子程序参数 |
| 0506 | 数据单位不正确 |
| 0507 | 数据维数不正确 |
| 0508 | 变量类型不正确 |
| 0509 | 数据范围不正确 |
| 0510 | 操作符数据错误 |
| 0511 | 变量定位错误 |
| 0512 | 数据覆盖 |
| 0513 | 外部数据错误 |
| 0514 | 输出数据错误 |
| 0515 | 输入数据错误 |
| 0516 | 数据检验错误 |
| 06 | 用户界面问题  U-User Interface | 人机交互特性：屏幕格式，确认用户输入，功能有效性，页面排版等方面的缺陷。 | 0601 | 界面风格不统一 |
| 0602 | 屏幕上的信息不可用 |
| 0603 | 屏幕上的错误信息 |
| 0604 | 界面功能布局和操作不合常规 |
| 07 | 文档问题  D-Documentation | 影响发布和维护，包括注释等缺陷。 | 0701 | 描述含糊 |
| 0702 | 项描述不完整 |
| 0703 | 项描述不正确 |
| 0704 | 项缺少或多余 |
| 0705 | 项不能验证 |
| 0706 | 项不能完成 |
| 0707 | 不符合标准 |
| 0708 | 与需求不一致 |
| 0709 | 文字排版错误 |
| 0710 | 文档信息错误 |
| 0711 | 注释缺陷 |
| 08 | 性能问题  P-Performance | 不满足系统可测量的属性值，如：执行时间，事务处理速率等缺陷。 |  | |
| 09 | 配置问题  B-Build/package/merge | 由于配置库、变更管理或版本控制引起的错误。 | 0901 | 配置管理问题 |
| 0902 | 编译打包缺陷 |
| 0903 | 变更缺陷 |
| 0904 | 纠错缺陷 |
| 10 | 标准问题  N-Norms | 不符合各种标准的要求， 如编码标准、设计符号等缺陷。 | 1001 | 不符合编码标准 |
| 1002 | 不符合软件标准 |
| 1003 | 不符合行业标准 |
| 11 | 环境问题  E-Environments | 由于设计、编译和运行环境引发的问题。 | 1101 | 设计、编译环境 |
| 1102 | 运行环境 |
| 12 | 兼容问题 | 软件之间不能正确地交互和共享信息 | 1201 | 操作平台不兼容 |
| 1202 | 浏览器不兼容 |
| 1203 | 分辨率不兼容 |
| 13 | 其他问题  O-Others | 以上问题所不包含的其他问题。 | | |

#### 缺陷等级分类

缺陷等级一般分为五个等级，致命、严重、一般、轻微、优化。详细如下：

|  |  |  |
| --- | --- | --- |
| **严重级别** | **对应缺陷严重等级** | **描 述** |
| 1-严重（**Critical）** | 严重缺陷 | 不能执行正常工作功能或实现重要功能，包括：  1）可能有灾难性的后果，如造成系统崩溃，造成事故等；  2）数据库错误，如数据丢失等。 |
| 2-重要**（Major）** | 较大缺陷 | 产生错误的结果，导致系统不稳定，运行时好时坏，严重地影响系统要求或基本功能实现的问题。如：  1）造成数据库不稳定的错误；  2）在说明中的需求未在最终系统中实现；  3）程序无法运行，系统意外退出；  4）业务流程不正确； |
| 3-中等**（Normal）** | 一般缺陷 | 不正确的，但不会影响系统稳定性的：  1） 过程调用或其它脚本错误；  2） 系统刷新错误；  3） 产生错误结果，如计算结果错误，数据不一致等；  4） 功能的实现有问题，如在系统实现的界面上，一些可接受输入的控件点击后无作用，对数据库的操作不能正确实现；  5） 编码时数据类型、长度定义错误；  6） 虽然正确性、功能不受影响，但系统性能和响应时间受到影响；  7） 对于输入数据没有进行必要的类型校验 |
| 4-次要（M**inor）** | 轻微缺陷 | 不正确的，但仅是使系统使用起来不太方便的错误，重点指系统的UI问题：  1）系统的提示语不明确，不简明；  2）滚动条无效；  3）可编辑区和不可编辑区不明显；  4）光标跳转设置不好，鼠标（光标）定位错误；  5）上下翻页，首尾页定位错误；  6）界面不一致，或界面不正确；  7）日期或时间初始值错误（起止日期、时间没有限定）；  8）出现错别字，标点符号错误，拼写错误，以及不正确的大小写等； |
| 5-有待改进（**Enhancement）** | 其他缺陷 | 系统中值得改良的问题：  1） 容易给用户误解和歧义的提示；  2） 界面需要改进的，某个控件没有对齐等。  3） 对有疑虑的部分，提出修改建议 |

#### 缺陷优先级分类

缺陷优先级：指缺陷必须被修复的紧急程度。“优先级”的衡量抓住了在严重性中没有考虑的重要程度因素。

|  |  |
| --- | --- |
| **缺陷优先级** | **描 述** |
| 1-立即解决（Resolve immediately） | 导致测试无法继续进行，必须立刻进行修复；对用户产生很大影响，必须优先解决。 |
| 2-高度关注（Highly Focus） | 对此缺陷给以高度重视，应优先进行修复。 |
| 3-正常排队（Normal Queue） | 缺陷需要正常排队等待修复或列入软件发布清单。 |
| 4-低优先级（Not Urgent） | 缺陷可以在方便时被纠正。 |

#### 缺陷状态分类

缺陷状态：指缺陷通过一个跟踪修复过程的进展情况。

|  |  |  |
| --- | --- | --- |
| **序号** | **缺陷状态** | **描 述** |
| 1 | 提交(Submitted) | 测试人员提交新的错误入库。 |
| 2 | 激活或打开(Active or Open) | 问题还没有解决，存在源代码中，确认“提交的缺陷”，等待处理。 |
| 3 | 拒绝（Rejected） | 拒绝“提交的缺陷”：不需要修复(Won’t fix)或不是缺陷(Invalid)或缺陷已经被其他的软件测试人员发现(Duplicate)。 |
| 4 | 已修正或修复(Fixed or Resolved) | 已被开发人员检查、修复过的缺陷，通过单元测试，认为已解决但还没有被测试人员验证 。 |
| 5 | 验证(Verify) | 缺陷验证通过 |
| 6 | 关闭或非激活(Closed or Inactive) | 测试人员验证后，确认缺陷不存在之后的状态。 |
| 7 | 重新打开(Reopen) | 测试人员验证后，还依然存在的缺陷，等待开发人员进一步修复。 |
| 8 | 推迟(Deferred) | 这个软件缺陷在下一个版本中解决。 |
| 9 | 保留(On hold) | 由于技术原因或第三者软件的缺陷，开发人员暂不能修复的缺陷。 |
| 10 | 不能重现(Cannot duplicate) | 开发不能复现这个软件缺陷，需要测试人员检查缺陷重现的步骤。 |

### 测试日志

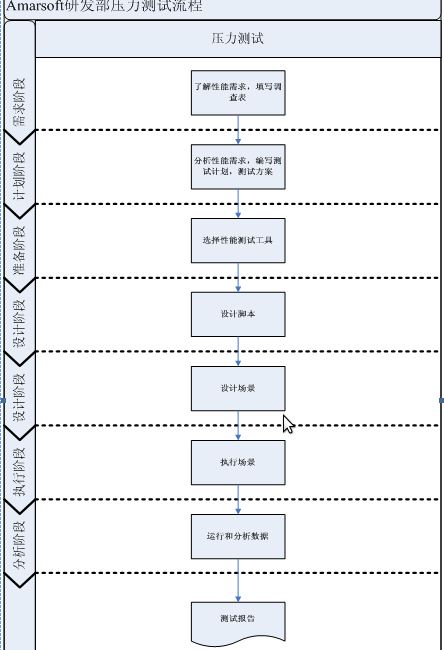
编写测试日志，除了记录测试人员的工作情况、保证测试工作的可追溯性之外，也包含了个人的一些不理解或有问题的地方。

测试日志模板可参考附件:AMARSOFT集中开发\_XX系统\_日志模板\_YYYYMMDD\_姓名.xls。

# 性能测试

通过性能测试，确定在各种工作负载下系统的性能，目标是测试当负载逐渐增加时，系统各项性能指标的变化情况。压力测试是通过确定一个系统的瓶颈或者不能接受的性能点，来获得系统能提供的最大服务级别的测试。

性能测试基本流程：



# 测试报告

测试报告作为测试团队的具体工作总结，它详细描述了测试过程和结果，通过得出的测试数据客观地进行分析，评价软件版本的质量，并为纠正软件的存在的质量问题提供依据，同时为软件验收和交付打下基础。

测试报告内容需包含：测试范围、测试版本、测试环境、测试时间、测试地点、测试策略、测试结果及分析、缺陷情况、遗留问题、后续改进方案等。

测试报告模板请参考附件: AMARSOFT集中开发\_XX系统\_测试报告模板\_YYYYMMDD\_v1.0.doc。

# 自动化测试

软件测试工具是通过一些工具能够使软件的一些简单问题直观的显示在读者的面前，这样能使测试人员更好的找出软件错误的所在。软件测试工具分为自动化软件测试工具和测试管理工具。软件测试工具存在的价值是为了提高测试效率，用软件来代替一些人工输入。测试管理工具是为了复用测试用例，提高软件测试的价值。一个好的软件测试工具和测试管理工具结合起来使用将会使软件测试效率大大的提高。

软件测试工具：

QuickTest Professional（QTP）：用于创建功能和回归测试。

LoadRunner（LR） ：预测系统行为和性能的负载测试工具。

测试管理工具：

Quality Center（QC）：需求管理，测试计划，测试执行以及错误跟踪等。

Bugzilla：管理软件开发中缺陷的提交（new），修复（resolve），关闭（close）等整个生命周期。

# 常见问题

1. 缺陷问题描述不清楚，开发理解错误。
2. 缺陷截图过小，以致开发定位问题缓慢。
3. 当开发驳回你的缺陷时，该怎么处理？

# 附件