**\*\*\*技技术有限公司**

**软件测试管理规定**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 文件编号： | | 生效日期： | | 受控编号： | |
| 密级： | | 版次：第 版 | | 修改状态 | |
| 总页数 |  | 正文 |  | 附件 |  |
| 编制或修订人： | | 审核： | | 批准： | |

（版权所有，翻版必究）

**目录**

[第一章 引言 4](#_Toc417030186)

[第一条 测试概述 4](#_Toc417030187)

[第二条 测试目标 4](#_Toc417030188)

[第三条 适用范围 5](#_Toc417030189)

[第二章 测试职责 5](#_Toc417030190)

[第三章 需求分析 6](#_Toc417030191)

[第四章 测试策略 7](#_Toc417030192)

[第四章 测试计划 8](#_Toc417030193)

[第五章 测试用例 8](#_Toc417030194)

[第一条 测试用例设计方法 8](#_Toc417030195)

[第二条 测试用例操作步骤 11](#_Toc417030196)

[第三条 测试用例选择准则 11](#_Toc417030197)

[第四条 测试软/硬件环境 12](#_Toc417030198)

[第五条 测试数据准备 12](#_Toc417030199)

[第六条 测试执行过程绩效考核 12](#_Toc417030200)

[第六章 测试执行 12](#_Toc417030201)

[第一条 项目测试周期 12](#_Toc417030202)

[第二条 项目测试启动 12](#_Toc417030203)

[第三条 项目测试阶段 13](#_Toc417030204)

[第四条 项目测试结束 13](#_Toc417030205)

[第五条 测试执行过程绩效考核 13](#_Toc417030206)

[第七章 测试变更 14](#_Toc417030207)

[第八章 缺陷管理 14](#_Toc417030208)

[第一节 缺陷基本属性 14](#_Toc417030209)

[第二节 缺陷管理流程 15](#_Toc417030210)

[第三节 缺陷分类 16](#_Toc417030211)

[第四节 缺陷定义 18](#_Toc417030212)

[第五节 缺陷完成度 19](#_Toc417030213)

[第六节 处理机制 20](#_Toc417030214)

[第九章 测试结果分析 20](#_Toc417030215)

[第一节 测试完成的标准 20](#_Toc417030216)

[第二节 允许保留的缺陷 21](#_Toc417030217)

[第十章 测试输出文档 21](#_Toc417030218)

# 第一章 引言

## 第一条 测试概述

无论怎样强调软件测试的重要性和它对软件可靠性的影响都不过分。在开发大型软件系统的漫长过程中，面对着极其错综复杂的问题，人的主观认识不可能完全符合客观现实，与工程密切相关的各类人员之间的通信和配合也不可能完美无缺，因此，在软件生命周期的每个阶段都不可避免地会产生差错。我们力求在每个阶段结束之前通过严格的技术审查，尽可能早地发现并纠正差错；

经验表明审查并不能发现所有差错，此外在编码过程中还不可避免地会引入新的错误。如果在软件投入生产性运行之前，没有发现并纠正软件中的大部分差错，则这些差错迟早会在生产过程中暴露出来，那时不仅改正这些错误的代价更高，而且往往会造成很恶劣的后果。测试的目的就是在软件投入生产性运行之前，尽可能多地发现软件中的错误。

目前软件测试仍然是保证软件质量的关键步骤，它是对软件规格说明、设计和编码的最后复审。软件测试在软件生命周期中横跨两个阶段。通常在编写出每个模块之后就对它做必要的测试(称为单元测试)，模块的编写者和测试者是同一个人，编码和单元测试属于软件生命周期的同一个阶段。在这个阶段结束之后，对软件系统还应该进行各种综合测试，这是软件生命周期中的另一个独立的阶段，通常由专门的测试人员承担这项工作。

大量统计资料表明，软件测试的工作量往往占软件开发总工作量的40％以上，在极端情况，测试那种关系人的生命安全的软件所花费的成本，可能相当于软件工程其他开发步骤总成本的三倍到五倍。因此，必须高度重视软件测试工作，绝不要以为写出程序之后软件开发工作就接近完成了，实际上，大约还有同样多的开发工作量需要完成。仅就测试而言，它的目标是发现软件中的错误，但是，发现错误并不是我们的最终日的。软件工程的根本目标是开发出高质量的完全符合用户需要的软件。

## 第二条 测试目标

下面这些规则也可以看作是测试的目标或定义：

(1)测试是为了发现程序中的错误而执行程序的过程；

(2)好的测试方案是极可能发现迄今为止尚未发现的错误的测试方案；

(3)成功的测试是发现了至今为止尚未发现的错误的测试。

从上述规则可以看出，测试的正确定义是“为了发现程序中的错误而执行程序的过程”。这和某些人通常想象的“测试是为了表明程序是正确的”，“成功的测试是没有发现错误的测试”等等是完全相反的。正确认识测试的目标是十分重要的，测试目标决定了测试方案的设计。如果为了表明程序是正确的而进行测试，就会设计一些不易暴露错误的测试方案；相反，如果测试是为了发现程序中的错误，就会力求设计出最能暴露错误的测试方案。

由于测试的目标是暴露程序中的错误，从心理学角度看，由程序的编写者自己进行测试是不恰当的。因此，在综合测试阶段通常由其他人员组成测试小组来完成测试工作。此外，应该认识到测试决不能证明程序是正确的。即使经过了最严格的测试之后，仍然可能还有没被发现的错误潜藏在程序中。测试只能查找出程序中的错误，不能证明程序中没有错误。

## 第三条 适用范围

本规范是对项目软件测试的一份指导性文件，对软件测试过程中所涉及到的测试理论、测试类型、测试方法、测试标准、测试流程以及软件产品开发单位所承担的职责进行总体规范，以有效保证软件产品的质量。

# 第二章 测试职责

测试职责是指在项目开发过程中跟测试工作有关的角色进行任务分配的，主要包含的角色以及工作职责如下：

* **测试组长**：由测试经理或项目经理指定项目组成员其他人员担任，测试组长负责：
* 分析需求并进行细化可用于执行测试的需求
* 制定测试计划
* 参与、跟踪测试过程
* 对测试活动和结果进行分析，撰写测试分析报告
* **测试人员**：由项目组成员担任，负责：
* 根据测试计划编写测试用例
* 搭建测试环境，准备测试脚本
* 执行测试，记录测试结果和缺陷
* 执行回归测试
* **开发人员：**由项目组成员担任，负责：
* 单元测试
* 功能开发完毕之后，提交测试之前的确认测试

# 第三章 需求分析

首先了解前期的需求调研报告、客户提出的业务需求功能点，以及本公司对需求的理解及说明，其次参加需求评审、设计评审。通过对文档分析，分解各功能模块，各功能点，为测试用例设计提供数据依据。

反复检查并理解各种信息，和用户交流，理解他们的要求。可以按照以下步骤执行：

1）确定软件提供的主要商业任务

2）对每个商业任务，确定完成该任务所要进行的交易。

3）确定从数据库信息引出的计算结果。

4）对于对时间有要求的交易，确定所要的时间和条件。这些条件包括数据库大小、机器配置、交易量、以及网络拥挤情况。

5）确定会产生重大意外的压力测试，包括：内存、硬盘空间、高的交易率

6）确定应用需要处理的数据量。

7）确定需要的软件和硬件配置。通常情况下，不可能对所有可能的配置都测试到，因此要选择最有可能产生问题的情况进行测试，包括：最低性能的硬件、几个有兼容性问题的软件并存、客户端机器通过最慢的LAN/WANF连接访问服务器。

8）确定其他与应用软件没有直接关系的商业交易。包括：

管理功能，如启动和推出程序

配置功能，如设置打印机

操作员的爱好，如字体、颜色

应用功能，如访问email或者显示时间和日期。

9）确定安装过程，包括定置从哪安装、定制安装、升级安装。

10）确定没有隐含在功能测试中的户界面要求。大多界面都在功能测试时被测试到。还有写没有测到，如：操作与显示的一致性，如使用快捷键等；界面遵从合理标准，如按钮大小，标签等。

# 第四章 测试策略

测试策略用于说明某项工作的测试方法与目标。系统测试策略主要针对系统测试需求确定测试类型及实施的测试方法与技术。测试策略一般包括下列内容：

**要实施的测试类型与目标**

确定系统测试策略首先要清楚地所实施系统测试的类型和测试目标。系统测试类型一般包括：

* + - * 1. 功能测试
        2. 性能测试
        3. 负载测试
        4. 强度测试
        5. 安全性测试
        6. 配置测试
        7. 故障恢复测试
        8. 文档测试
        9. 用户界面测试

其中，功能测试，配置测试，安装测试在一般情况下是必需的，其它类型的测试可根据需求进行裁剪。

1. 采用的技术：系统测试主要采用黑盒测试技术来设计测试用例来确定软件是否满足需求规格说明中的要求。
2. 用于测试评估结果和测试是否完成的标准
3. 对测试策略所述的测试工作存在影响的特殊事项

# 第四章 测试计划

根据测试的种类，测试计划分为功能测试和性能测试计划。测试计划旨在说明各测试阶段任务、人员分配、时间安排、测试要点、工作规范等。测试计划在策略和方法方面说明如何计划、组织和管理测试项目。测试计划包含足够的信息使测试人员明白项目需要做什么是如何运作的。测试计划不包括测试用例的细节和系统功能的详细信息。测试计划应附有测试功能点矩阵、测试性能点矩阵。

测试计划应在项目组内进行评审。参与测试计划评审的人员包括：项目经理、测试组长、开发组长、测试组员。

# 第五章 测试用例

测试用例是为实施测试而向被测试系统提供的输入数据、操作或各种环境设置以及期望结果的一个特定的集合。解决要测什么、怎么测和如何衡量的问题。

从测试结构上面划分分为黑盒测试、和百盒测试2种，他们各自有不同的测试方式，目前本公司只考虑黑盒测试，以下设计方法以黑盒方法为例

## 第一条 测试用例设计方法

黑盒测试用例设计方法有等价类测试、边界值分析、基于因果图的测试、基于猜错的测试、基于场景的测试、基于随机的测试。其中常用的设计方法有等价类测试、边界值分析、因果图三种方法，以下分别介绍这几种方法：

* **等价类划分**

等价类划分是一种典型的黑盒测试方法。等价类是指某个输入域的集合。它表示对揭露程序中的错误来说，集合中的每个输入条件是等效的。因此我们只要在一个集合中选取一个测试数据即可。等价类划分的办法是把程序的输入域划分成若干等价类，然后从每个部分中选取少数代表性数据当作测试用例。这样就可使用少数测试用例检验程序在一大类情况下的反映。

在考虑等价类时，应该注意区别以下两种不同的情况：

有效等价类：有效等价类指的是对程序的规范是有意义的、合理的输入数据所构成的集合。在具体问题中，有效等价类可以是一个，也可以是多个。

无效等价类：无效等价类指对程序的规范是不合理的或无意义的输入数据所构成的集合。对于具体的问题，无效等价类至少应有一个，也可能有多个。

确定等价类有以下几条原则：

如果输入条件规定了取值范围或值的个数，则可确定一个有效等价类和两个无效等价类。例如，程序的规范中提到的输入条包括“……项数可以从1到999……”，则可取有效等价类为“l考项数＜999”，无效等价类为“项数＜l，，及“项数＞999”。

输入条件规定了输入值的集合，或是规定了“必须如何”的条件，则可确定一个有效等价类和一个无效等价类。如某程序涉及标识符，其输入条件规定“标识符应以字母开头……”则“以字母开头者”作为有效等价类，“以非字母开头”作为无效等价类。

如果我们确知，已划分的等价类中各元素在程序中的处理方式是不同的，则应将此等价类进一步划分成更小等价类。

|  |  |  |
| --- | --- | --- |
| 输入条件 | 有效等价类 | 无效等价类 |
| 。。。。。。  。。。。。。 | 。。。。。。  。。。。。。 | 。。。。。。  。。。。。。 |

根据已列出的等价类表，按以下步骤确定测试用例：

为每个等价类规定一个唯一的编号；

设计一个测试用例，使其尽可能多地覆盖尚未覆盖的有效等价类。重复这一步，最后使得所有有效等价类均被测试用例所覆盖；

设计一个新的测试用例，使其只覆盖一个无效等价类。重复这一步，使所有无效等价类均被覆盖。这里强调每次只覆盖一个无效等价类。这是因为一个测试用例中如果含有多个缺陷，有可能在测试中只发现其中的一个，另一些被忽视。等价类划分法能够全面、系统地考虑黑盒测试的测试用例设计问题，但是没有注意选用一些“高效的”、“有针对性的”测试用例。后面介绍的边值分析法可以弥补这一缺点。

* **边值分析法**

边值分析法是列出单元功能、输入、状态及控制的合法边界值和非法边界值，设计测试用例，包含全部边界值的方法。典型地包括IF语句中的判别值，定义域、值域边界，空或畸形输入，末受控状态等。边值分析法不是一类找一个例子的方法，而是以边界情况的处理作为主要目标专门设计测试用例的方法。另外，边值分析不仅考查输入的边值，也要考虑输出的边值。这是从人们的经验得出的一种有效方法。人们发现许多软件错误只是在下标、数据结构和标量值的边界值及其上、下出现，运行这个区域的测试用例发现错误的概率很高。

用边值分析法设计测试用例时，有以下几条原则：

如果输入条件规定了取值范围，或是规定了值的个数，则应以该范围的边界内及刚刚超出范围的边界外的值，或是分别对最大、最小及稍小于最小、稍大于最大个数作为测试用例。如有规范“某文件可包含l至255”个记录……“，则测试用例可选1和255及0和256等。

针对规范的每个输出条件使用原则〔a〕。

如果程序规范中提到的输入或输出域是个有序的集合(如顺序文件、表格等)就应注意选取有序集的第一个和最后一个元素作为测试用例。

分析规范，尽可能找出可能的边界条件。一个典型的边值分析例子是三角形分类程序。选取a，b，c构成三角形三边，“任意两边之和大于第三边”为边界条件。边值分析相等价类划分侧重不同，对等价类划分是一个补充。如上述三角形问题，选取a＝3，b＝4，c＝5，a＝2，b＝4，c＝7则覆盖有效和无效等价类。如果能在等价类划分中注入边值分析的思想。在每个等价类中不只选取一个覆盖用例，而是进而选取该等价类的边界值等价类划分法将更有效，最后可以用边值分析法再补充一些测试用例。

* **因果图**

等价类划分法并没有考虑到输入情况的各种组合。这样虽然各个输入条件单独可能出错的情况已经看到了，但多个输入情况组合起来可能出错的情况却被忽略。采用因果图方法能帮助我们按一定步骤选择一组高效的测试用例，同时，还能为我们指出程序规范的描述中存在什么问题。

利用因果图导出测试用例需要经过以下几个步骤：

分析程序规范的描述中哪些是原因，哪些是结果。原因常常是输入条件或是输入条件的等价类。结果是输出条件。

分析程序规范的描述中语义的内容，并将其表示成连接各个原因与各个结果的“因果图”。

由于语法或环境的限制，有些原因和结果的组合情况是不可能出现的。为表明这些特定的情况，在因果图上使用持殊的符号标明约束条件。把因果图转换成判定表。把判定表的每一列写成一个测试用例。

* **猜错法**

猜错法在很大程度上是凭经验进行的，是凭人们对过去所作的测试工作结果的分析，对所揭示的缺陷的规律性作直觉的推测来发现缺陷的。

一个采用两分法的检索程序，典型地可以列出下面几种测试情况：

被检索的表只有一项或为空表；

表的项数恰好是2的幂次；

表的项数比2的幂次多1等。

猜错法充分发挥人的经验，在一个测试小组中集思广益，方便实用，特别在软件测试基础较差的情况下，很好地组织测试小组 (也可以有外来人员)进行错误猜测，是有效的测试方法。

* **随机数法**

即测试用例的参数是随机数。它可以自动生成，因此自动化程度高。使用大量随机测试用例测试通过的程序会提高用户对程序的信心。但其关键在于随机数的规律是否符合使用实际。

## 第二条 测试用例操作步骤

1. 在设计编写测试用例时，首先要从测试用例库中选择相应功能的测试用例，在原有测试用例的基础上依据系统需求文档对测试用例的进行修改、更新，评审通过后将使用该测试用例测试被测系统。
2. 在测试项目结束后，统计分析所使用过的测试用例，进行分类放到相应的测试用例库中。为以后测试用例的设计编写提供数据基础。

## 第三条 测试用例选择准则

测试用例的代表性：能够代表各种合理和不合理的、合法的和非法的、边界和越界的,以及极限的输入数据、操作和环境设置等；

测试结果的可判定性：即测试执行结果的正确性是可判定的或可评估的；

测试结果的可再现性：即对同样的测试用例,系统的执行结果应当是相同的。

## 第四条 测试软/硬件环境

根据需求文档提供的内容，和开发部沟通确定测试项目所需的软硬件环境，完成对测试项目所需软硬件资源的准备工作，使软硬件资源得到满足。

完成对软硬件资源的配置后，要进行对测试项目的软硬件环境进行评审，确认对软硬件资源配置的有效性。

## 第五条 测试数据准备

完成对测试项目基本数据的准备操作，包括数据库连接、用户信息、用户角色权限、单位组织等信息和测试相关的测试数据。

## 第六条 测试执行过程绩效考核

为促进测试人员积极主动做好测试执行工作，对测试人员进行测试执行过程进行考核。

|  |  |  |
| --- | --- | --- |
| 序号 | 测试准备内容 | 考核评分标准 |
| 1 | 测试组长工作安排 | 待定 |
| 2 | 测试组长风险评估 | 待定 |
| 3 | 测试人员设计用例 | 待定 |
| 4 | 测试人员执行用例 | 待定 |
| 5 | 开发组长配合度 | 待定 |
| 6 | 开发人员回归次数 | 待定 |
| 7 | 开发人员处理问题情况 | 待定 |

以上统计数据由项目经理提供给部长。

# 第六章 测试执行

## 第一条 项目测试周期

测试项目的测试周期可分为：单元测试、接收测试、集成测试、系统测试、回归测试、性能测试等。

## 第二条 项目测试启动

软件项目测试活动的正式启动，是在确认软件可测试性后展开的。开发人员需要对产品进行单元测试，单元测试效果通过接收测试验证。

## 第三条 项目测试阶段

测试人员依据测试计划和测试用例进行测试活动。

测试一般分为两个阶段：

1、集成测试、系统测试阶段：该阶段测试人员每天提交缺陷，并跟踪缺陷，验证缺陷，直到提交的缺陷被关闭或被保留。开发人员周期性提交修改过缺陷的新版本，测试人员在新版本上验证缺陷。

2、回归测试阶段：在集成测试、系统测试阶段完成后，产品将进入回归测试阶段。测试人员对修改后的产品进行重新功能验证，确保修改的正确性，验证在修改缺陷的同时没有引入新的问题。回归缺陷是指开发人员标示已修改的缺陷，经测试后发现仍未修改正确，或引入其他缺陷，或在前一个版本中未发现的缺陷，在后一个版本中出现。

如产品进行性能测试，则需要在性能测试后，进行一轮回归测试，确保功能的正确性。

## 第四条 项目测试结束

项目测试结束时应达到测试质量目标所规定的标准。通过评审后结束该项目测试。

## 第五条 测试执行过程绩效考核

为促进开发人员积极主动做质量工作，对开发人员进行考核。

|  |  |  |
| --- | --- | --- |
| **序号** | **开发人员考核内容** | **考核评分标准** |
| 1 | 开发人员提交的首个产品未通过单元测试标准 | 待定 |
| 2 | 开发人员无故将【严重】、【非常严重】级别无争议的缺陷延期3天修改。 | 待定 |
| 3 | 开发人员未能正确修改缺陷，导致状态为【已修改】的缺陷被【重新打开】，每天超过1个。 | 待定 |
| 4 | 开发人员千行缺陷代码率在项目组中排名第一者 | 待定 |
| 5 | 一个项目中【延迟修改】或【已知问题】的缺陷数超过总缺陷数的10% | 待定 |

以上统计数据由测试人员在项目交付后提供给部长。

# 第七章 测试变更

当需求变更，功能变化，测试人员根据变更情况，评估测试变更所需时间，提出变更风险。如变更情况被项目组通过，测试组长要修改相应的测试计划，测试人员要从新设计测试用例。

# 第八章 缺陷管理

## 第一节 缺陷基本属性

缺陷是指在软件开发过程中的针对软件产品和开发过程中的问题，这些问题已经影响或可能会影响软件产品的质量。缺陷应该具备以下属性，也就是往缺陷管理库或者缺陷列表中提交的缺陷应该具备以下属性：

|  |  |
| --- | --- |
| **属性名称** | **描述** |
| 缺陷标识 | 标记某个缺陷的一组符号，每个缺陷必须有一个唯一的标识 |
| 缺陷类型 | 根据缺陷的自然属性划分的缺陷种类 |
| 缺陷验证程度 | 因缺陷引起的故障对软件产品的影响程度 |
| 缺陷所处的模块或子系统 | 缺陷分步的模块或子系统 |
| 缺陷出现几率 | 指发现错误的几率 |
| 缺陷的重现步骤 | 详细的缺陷重现步骤 |
| 附件 | 与缺陷相关的附件（截图、附件、用例等） |
| 备注 | 对缺陷的其他描述 |

## 第二节 缺陷管理流程



* **提交缺陷**

测试人员将缺陷填写到管理工具中，选择指派人为开发组长或相应的开发人员。

* **分配缺陷**

开发人员分别对自己收到的缺陷进行评审。评审后如果对提交的缺陷有疑问，可以与提交人协商。对未能达成一致的缺陷由项目经理组织项目组成员评审。评审人员可以是项目组人员。

如果缺陷初次分配的开发人员无法修改该缺陷，初次分配的开发人员可以将缺陷再次分配给其他开发人员。但为避免缺陷被多次分配，项目经理应跟踪3天以上未修改的缺陷。

* **修改缺陷**

开发人员对已确认的缺陷进行修改，填写修改记录，修改缺陷状态为“已修改”或其他状态。

* **关闭缺陷**

测试人员对已修改的缺陷进行验证。如果已修改完成，测试人员将缺陷状态设置为关闭。如果没有修改或引起回归问题，将修改缺陷状态为“重新开启”或新增缺陷，由开发工程师继续修改。

* **保留缺陷**

对于有争议的缺陷进行，将有项目经理最终决定是否修改。如果缺陷是由于技术原因、版本原因不能修改，则保留该缺陷。

## 第三节 缺陷分类

* **根据缺陷的定义，将缺陷分为如下列**
* 文档缺陷：是指对文档的静态检查过程中发现的缺陷。检查活动包括同行评审、产品审计等。评审的缺陷要根据被评审对象的类型来确定，被评审的对象包括最终出产物和中间过程产出物，比如需求文档、设计文档、计划、报告、用例等

|  |  |
| --- | --- |
| **缺陷分类** | **描述** |
| 描述不完整 | 文档内容缺失，或文档应该包括的范围没有涵盖 |
| 不一致 | 一致性问题有两类：  一是与源头说明书不一致，比如需求和客户业务需求不一致、设计与需求不一致等  二是上下文或者与前提不一致 |
| 描述错误 | 文档描述是错误的，不可实现或导致错误的输出或结果 |
| 功能问题 | 该缺陷将会导致用户功能的错误、不满足、不可用 |
| 不清楚或有歧义 | 内容的描述不清楚、不能准确表达、或表达的意思有歧义 |
| 逻辑错误 | 内容组织逻辑不清楚、逻辑错误 |
| 接口问题 | 与最终用户接口问题、与外部系统的接口问题、内部子系统或模块的接口问题 |
| 输入输出问题 | 输入输出不完整、不正确、不可测试或验证 |
| 不细化 | 内容还需要进一步细化 |
| 性能问题 | 文档的设计或实现方式存在性能问题 |
| 安全性问题 | 文档的设计或实现方式存在安全性问题 |

* 代码缺陷：是指对代码进行同行评审、审计或代码走查过程中发现的缺陷

|  |  |
| --- | --- |
| **缺陷分类** | **描述** |
| 常量变量定义问题 |  |
| 不满足设计或需求 |  |
| 编写代码不符合规范 |  |
| 条件判断处理 |  |
| 循环处理错误 |  |
| 异常处理 |  |
| 算法逻辑问题 |  |
| 注释问题 |  |
| 代码冗余 |  |
| 性能问题 |  |

* 测试缺陷：是指由测试活动发现的测试对象（被测对象一般是指可运行的代码、系统，不包括静态测试发现的问题）的缺陷，测试活动包括单元测试、集成测试、系统测试、性能测试等
* 过程缺陷：有称为不符合项问题，是指通过过程审计、过程分析、管理评审、质量评估、质量审核等活动发现的关于过程的缺陷和问题。过程缺陷的发现者一般是测试人员、项目经理等

|  |  |
| --- | --- |
| 缺陷类型 | 描述 |
| 功能错误 | 影响了重要的特性、用户界面、产品接口或全局数据结构，并且设计文档需要争取的变更。如逻辑、循环、递归、功能等缺陷 |
| 结构错误 | Web应用程序结构化页面无法显示，或者显示错误 |
| 脚本错误 | Web应用程序当中出现脚本错误，包括客户端对数据进行校验和运算的各种情况下产生的错误 |
| 页面链接错误 | Web应用程序页面出现空链接、错误链接、死链接 |
| 页面文字错误 | Web应用程序页面出现的中外文拼写、使用、以及不同语种页面的编码错误 |
| 页面图形错误 | Web应用程序页面出现图片内容使用不当，或者无法显示 |
| ALT错误 | Web应用程序页面当中超文本标识语言、文本标签解释错误 |
| 排版错误 | Web应用程序页面排版不符合要求或者不符合使用习惯 |
| 业务逻辑不合理 | 应用程序的实现流程和规定业务流程不一致，或者实现流程无法正确完成。包括流程数据的部分并行、争用、同步等操作，引起的流程断裂、死锁、以及其他异常情况 |
| 业务逻辑不方便 | 应用程序实现流程在实际情况下虽然可以完成，但是存在不必要的反复、等待、冗余等影响使用效率的情况 |
| 其他错误 | 其他未分类错误 |
| 建议 | 系统改进建议 |

## 第四节 缺陷定义

* **缺陷等级定义**

缺陷的严重程度对以上所述的缺陷类型都是适合的，缺陷的严重程度反映的是对缺陷的发现对象可能造成的影响或后果来定义的。

|  |  |  |  |
| --- | --- | --- | --- |
| 缺陷等级 | 缺陷性质 | 系统中对应的错误分类 | 描述 |
| 一级 | 致命错误 | 系统崩溃  系统死锁 | 导致对被描述的主要对象的理解错误、不可行、不可运转、对业务和整个系统造成重大损失或损害；对使用、维护或保管人员有危险或不安全，以及对产品的基本功能有致命影响的缺陷 |
| 二级 | 严重缺陷 | 严重错误 | 对被描述的部分对象的理解或实现错误，部分的模块或系统不可行或不能运转或部分模块和系统缺失，对整个系统有重大影响或可能造成部分的损失或损害；严重影响使用安全 |
| 三级 | 一般缺陷 | 次要错误  布局不合理  文字错误 | 系统中部分单元模块或单个功能描述和实现有错误、有偏差、不一致或有缺失，不影响模块的正常运行，或有影响，但可以有替代的办法或避免办法 |
| 四级 | 微小缺陷 | 微不足道 | 基本不影响系统的运行和功能的实现。但是与标准、规范和定义不一致 |
| 五级 | 建议缺陷 | 新特性 | 不在定义、标准、范围的定义和约束之内，但是从提出者来看是需要完善的建议 |

* **缺陷优先级定义**

|  |  |
| --- | --- |
| **缺陷优先级** | **描述** |
| 特急 | 需要立刻进行修改 |
| 加急 | 一天到两天之内必须修改 |
| 高 | 介于中和加急之间 |
| 中 | 缺陷需要正常排队等待修复或列入软件发布清单 |
| 低 | 留到组后解决，如果项目的进度跟紧张可以在产品发布以前不解决 |

* **缺陷状态定义**

|  |  |
| --- | --- |
| **缺陷状态** | **描述** |
| 初始状态（New） | 测试或开发人员提交一个新的缺陷，等待开发人员或项目经理分配修改负责人 |
| 打回（FeedBack） | 要求缺陷的报告者再次对缺陷进行说明 |
| 已分配（Assigned） | 是指已经分配给属主，等待修改。 |
| 已解决（Resolved） | 缺陷被属主修改，等待测试人员验证 |
| 关闭（Closed） | 测试人员验证缺陷已经修复 |
| 重新打开（Reopen） | 测试人员验证，缺陷没有修改正确 |
| 遗留（Later） | 经项目经理和技术经理验证此缺陷在本版本中不用修改 |

## 第五节 缺陷完成度

|  |  |
| --- | --- |
| **缺陷完成度** | **描述** |
| 打开（Open） | 缺陷没有被解决 |
| 已解决（Fixed） | 缺陷已经修改 |
| 遗留（Suspended） | 此缺陷步骤本阶段解决 |
| 重新打开（Reopen） | 重新打开某个缺陷 |
| 不做修改（Won’t fix） | 不对这个缺陷进行修改 |
| 重复  （Duplicate） | 与某个缺陷重复 |
| 需求如此 | 经理和开发人员经过需求和设计的核实后决定不需要修改 |
| 不可重现 | 被指派的开发人员想要再现缺陷进行修改个时候，发现缺陷始终不能再现 |

## 第六节 处理机制

* **退回机制**

若在测试过程中发生如下情况，将系统退回到申请部门：

* 经过测试后，发现与需求说明规格说明书中定义的功能项存在较大的差异
* 单一模块，测试过程中发现缺陷输了较多或者无法继续进行系统其它功能模块的测试，继续测试无意义
* 测试过程中，频繁死机或系统崩溃
* 主业务流程出现断点
* **异常情况处理机制**

非正常情况下，需要进行特别处理的情形，此情况需要主管领导签字确认：

* 上线时间紧急的情况下，未经测试部充分测试就需要部署到用户现场
* 作为总包时，子商进度明显延迟，尚未进行验收测试就需要上线
* **报告机制**

若出现以下情况，需要及时向部门领导和项目经理汇报的情况：

* 测试后期出现重大逻辑错误，修改测试影响上线时间
* 测试过程中用户需求出现重大变更
* 测试负责人定期汇报测试情况

# 第九章 测试结果分析

## 第一节 测试完成的标准

被测试出的、在软件错误级别分类中定义的：

* 一级缺陷，致命错误，100%得到修改并且复测通过
* 二级缺陷，严重错误，100%得到修改并且复测通过
* 三级缺陷，较大错误，100%得到修改并且复测通过
* 四级缺陷，一般错误，95%得到修改并且复测通过
* 五级缺陷，轻微错误，95%得到修改并且复测通过

## 第二节 允许保留的缺陷

测试超过了预定时间表，由项目经理决定是否停止测试

测试结论及评价标准

|  |  |
| --- | --- |
| **测试结论** | **评价标准** |
| 拒绝发布 | 遗留了一级、二级缺陷 |
| 测试通过版本 | 不能遗留以一、二类缺陷  三类 一般缺陷95%得到修改并且通过复测  四类轻微缺陷85%得到修改并且通过复测 |
| 推荐使用版本 | 不能遗留以一、二类缺陷  三类 一般缺陷95%得到修改并且通过复测  四类轻微缺陷90%得到修改并且通过复测 |
| 可以证实发布版本 | 不能遗留以一、二类缺陷  三类 一般缺陷97%得到修改并且通过复测  四类轻微缺陷90%得到修改并且通过复测 |

测试结果分析是对测试结果的一个综合评估，主要描述有测试中各个等级的缺陷数量，缺陷分布情况，缺陷修改情况、回归测试提交缺陷数量，性能测试指标情况。

测试报告由测试组长编写并提交给项目经理。测试报告需要经项目组评审通过。

# 第十章 测试输出文档

1. 软件系统测试计划(方案)
2. 系统测试用例
3. 系统测试过程（缺陷跟踪与管理）
4. 测试脚本（可选）用于回归测试、性能测试
5. 系统测试报告
6. 性能测试报告