软件缺陷管理办法

1. **目的**

本文档定义了软件缺陷管理流程和相关规则，确保软件缺陷管理的系统性和规范性，以保证项目研发质量。

1. **适用范围**

适用于部门项目研发过程的缺陷管理，对各阶段的缺陷管理过程进行指导和规范。

1. **定义**

3.1 术语

缺陷（Defect）：存在于软件之中偏差，可被激活，以静态形式存在于软件内部。

Bug：缺陷一种表现形态，系统或程序存在的任何一种破坏正常运转能力的问题。

3.2 缺陷定义

（1）软件未达到需求规格说明书的功能；

（2）软件出现了需求规格说明书指明不会出现的错误；

（3）软件功能超出需求规格说明书的范围；

（4）软件未达到需求规格说明书未指出但应达到的目标；

（5）测试工程师认为软件难以理解、不易使用、运行速度慢，或者最终用户认为不好。

1. **缺陷生命周期**

4.1 缺陷生命周期图



4.2 缺陷状态说明

|  |  |
| --- | --- |
| **缺陷状态** | **状态说明** |
| 激活状态 | 缺陷的初始状态，或者重新被激活的状态。激活状态的缺陷可以通过编辑来修改缺陷内容，并指派给合适的工程师处理。 |
| 解决状态 | 缺陷被解决之后的状态。 激活状态的缺陷经过成功修复以后，由开发工程师操作为解决状态，系统将自动指派回创建者。 |
| 关闭状态 | 解决状态的缺陷在验证通过后关闭，缺陷状态变为关闭，生命周期结束。如果验证未修复或者新版本又发生，则重新激活，缺陷状态重新变为激活。 |

**5. 缺陷处理过程**

5.1 正常处理过程

（1）创建问题

在测试管理系统中，所有用户都可以创建新问题，包括需求问题和软件缺陷等。创建问题时，需要描述清楚，并选择正确的选项，详细请参考5.4和5.5。

（2）指派问题

创建问题时，创建者通常要指派给该项目开发负责人，再由其指派任务，或直接指派给相应模块的开发工程师。

如果指派人是错误的，或者需要他人确认或帮助，则可以重新指派给合适的工程师，写上相关备注。

（3）确认问题

通常开发工程师收到新问题后，需要分析和确认此问题是否为Bug。如果是Bug，则选择“确认状态”；如果认为非Bug，则注明原因并指派回创建者。

当创建者收到确认指派时，需要进行及时确认。如果同意为非bug，则及时关闭它；如果不同意，则需要注明理由并指派回相关工程师。

如果问题确认指派次数大于6次时，需要进入“争议处理”流程，详细请参考5.2。

（4） 解决问题

此为开发工程师的主要职责，包括Bug的复现、修改和修改验证。

开发工程师需要及时对确认状态Bug进行分析和解决，并自己验证通过，则操作为解决状态，解决方案规则请参考5.4中解决方案定义部分，在缺陷管理系统中解决方案选择相应的选项，解决后系统将自动指派回给创建者。

如果Bug无法解决或修改影响比较大，可申请进入“延期解决”流程，请参考5.2中延期处理部分。

（5） 验证问题

创建者需要及时对解决状态的Bug在对应版本上面进行验证。如果验证通过，则可关闭Bug；如果验证不通过，则激活此Bug，系统将自动指派回给解决者。

验证通过准则：相同的操作步骤，进行一定次数的验证测试都没有发生。

验证不通过准则：相同的操作步骤，全部或部分实际结果还会发生，验证不通过则激活Bug。

(6) 关闭问题

通过验证的Bug，验证者需要注明验证结果并进行关闭操作，系统将指派给Closed。

如果关闭状态的Bug在之后版本又会发生，则激活此Bug，系统将自动指派回给解决者。

5.2 特别处理过程

(1) 客户问题

客户反馈的问题可以由客户直接反馈或项目经理、市场部等了解到的客户问题，经确认后的Bug提交到测试管理系统，按照以上处理流程进行处理，由创建者或测试组进行跟踪验证关闭。

创建客户问题时，创建者需要在Bug标题开头标记为[客户问题]，测试组负责检查和更正。

(2) 争议处理

当开发和测试工程师对某问题有争议并且多次沟通无果时（暂定为6次），可以注明双方的理由，并指派给项目经理进行处理。

项目经理可以召开评审会议，或者直接与双方沟通了解，并根据项目情况给出专业意见和最终决定。开发和测试工程师根据项目经理的最终决定执行。

(3) 延期解决

当开发工程师对确认Bug进行解决时，发现或评估其解决时间紧或风险比较大等，可以说明原因或理由并指派给项目经理来确认。

项目经理可以召开评审会议，或者直接沟通了解，并根据项目情况给出最终决定。如果不同意，项目经理将此Bug指派回开发工程师，开发工程师继续分析和解决。如果同意，项目经理需要在Bug标题开头标记为[延期解决]和在处理状态选择“延期解决”，然后注明解决时间计划并指派回开发工程师，开发工程师根据解决时间计划来规划和解决此Bug。

5.3 缺陷管理工具

软件测试过程中所有缺陷要提交到公司测试管理系统进行跟踪管理。

1. 管理工具的作用
2. 确保每个被发现的缺陷都能够被跟踪与处理。
3. 收集缺陷数据并根据缺陷趋势曲线识别或报告测试状态。
4. 收集缺陷数据并在其上进行数据分析，作为测试评估的依据。

（2）缺陷驱动原则

缺陷管理系统主要通过指派状态来驱动相关开发工程师、测试工程师和项目经理尽快地处理问题，以提高研发效率，所以会特别关注缺陷指派给谁和停留时间，并反馈在定期报告。

所以，缺陷驱动原则：尽量不要让缺陷挂在你身上。

5.4. 缺陷属性定义

(1) 缺陷相关属性

|  |  |
| --- | --- |
| **缺陷属性** | **说明** |
| 缺陷ID | 缺陷ID是标记某个缺陷的一组符号。每个缺陷必须有一个唯一的ID。 |
| 缺陷类型 | 缺陷类型是根据缺陷的自然属性划分的缺陷种类。 |
| 严重程度 | 缺陷严重程度是指因缺陷引起的失效对软件产品的影响程度。 |
| 发生概率 | 缺陷发生概率指缺陷按照测试操作步骤发生的概率情况。 |
| 解决方案 | 缺陷解决方案是指缺陷被解决掉的处理方案。 |
| 缺陷描述 | 缺陷描述是对缺陷的报告，包括标题、操作步骤和结果等。 |

1. 缺陷类型说明

|  |  |
| --- | --- |
| **类型名称** | **说明** |
| 设计缺陷 | 由于软件设计或代码实现所产生的功能或流程的问题。 |
| 界面问题 | 系统页面的展示的问题。 |
| 数据问题 | 系统数据的来源，处理及处理结果的问题。 |
| 需求问题 | 软件需求测试发现的问题，也包括之后需求变更的问题。 |
| 安装部署 | 软件安装部署过程的错误。 |
| 性能问题 | 软件性能相关的缺陷。 |
| 文档问题 | 用户使用手册，软件帮助文档等出现的问题 |
| 常识问题 | 系统用户的正常使用习惯相关问题。 |
| 安全问题 | 系统漏洞安全问题。 |
| 优化建议 | 针对操作过程逻辑或界面显示的优化性建议。 |
| 其他 | 除前面分类的其他问题 |

(3) 严重程度定义

|  |  |
| --- | --- |
| **严重程度** | **定义和说明** |
| 致命 | 阻碍开发或测试工作继续进行的系统性故障，例如：* 实现的功能与产品定义或软件需求规格严重不符。
* 系统无法执行、崩溃、冻结，死循环等。
* 程序引起的死机，非法退出。
* 主要功能模块严重错误。
* 数据库链接错误，严重数据计算错误通讯错误等。
 |
| 严重 | 系统出现的严重错误，但不影响当前测试工作的错误。例如：* 模块功能错误，模块功能未实现，乱码等；
* 功能错误，如链接模块有误，基本按键使用有误等。
* 数据错误，如用户数据丢失、破坏、计算、保存有误等。
 |
| 一般 | 不影响用户使用的非严重问题* 次要功能未实现或与需求不符。
* 操作界面错误，如界面图表或字符的一般性错误，但不影响操作。
* 提示信息错误，辅助说明不清楚。
* 数据错误，数据边界、格式约束未实现或需求不一致
 |
| 建议 | 测试过程中发现不利用户操作的优化建议。* 界面字符或提示与的显示不恰当。
* 页面或操作习惯的优化性建议。
* 功能操作更好的实现方式。
 |

注：严重等级由创建者在创建缺陷时根据此定义来选择，之后都不能随意更改。

(5) 优先级的定义

|  |  |
| --- | --- |
| **优先级** | **定义和说明** |
| 立刻 | 阻碍测试工作无法进行，需要开发工程师马上解决问题。* 所有的致命问题都需要立刻解决；
* 时间急迫时影响版本上线的问题等；
 |
| 紧急 | 不影响测试工作的严重问题，下个测试版本发版前必须解决。 * 所有严重问题；
* 常用模块功能、业务逻辑或数据错误；
* 明显的性能问题；
 |
| 尽快 | 一般性功能错误，版本发布前应该解决的问题。* 大多数一般问题；
* 页面显示，页面的字符，界面图标、文字显示、链接有误，不影响使用。如错别字，图标显示异常等；
 |
| 一般 | 不影响版本上线的问题，部分问题允许不修改。* 非常用界面或字符的显示错误或不恰当；
* 用户使用习惯，语言表达等优化建议；
 |

注：立刻、紧急、尽快的问题都要求在系统上线前解决。

(6)发生概率定义

|  |  |  |
| --- | --- | --- |
| **发生概率** | **定义说明** | **验证问题最小次数** |
| 必现 | 100%。 测试5次，出现5次。 | 5次 |
| 经常 | 100%>&>=30%。测试5次，出现3~4次；或测试10次，出现3次及以上；或测试15次，出现5次及以上。 | 10次 |
| 偶尔 | 30%>&>=10%。 测试10次，出现2次； 或测试15次，出现2~4次。 | 20次 |
| 随机 | <10%。 测试15次，出现1次。 | 30次 |

(7)处理状态说明

|  |  |
| --- | --- |
| **处理状态** | **相关规则** |
| 确认中  | 当问题确认过程时，可以选择这状态来说明。 |
| 解决中  | 开发工程师设置此状态。当Bug正在分析或解决时，可选这状态来说明。 |
| 复现中  | 当正在复现问题，或者正在跟踪测试问题，可以选择这状态来说明。 |
| 验证中  | 当验证随机问题等需要长时间，可以选择这状态来说明。 |
| 延期解决 | 项目经理才能设置此状态，参考5.2 |

(8) 解决方案定义与规则

|  |  |  |
| --- | --- | --- |
| **解决方案** | **方案说明** | **相关规则** |
| 已经解决 | 缺陷被修复或更正，并通过其验证测试。 | 开发工程师权限，解决时需要填写“解决版本”和注明Bug原因等。 |
| 重复缺陷  | 相同的缺陷别人已经提交，或者开发认为原因是相同的 | 开发工程师权限，解决时需要填写正确的重复缺陷 ID。 |
| 无效缺陷 | 设计如此，不是问题，优化建议不采纳，没法解决的第三方问题。 | 开发工程师需要与创建者沟通说明，直到创建者同意，开发工程师才能选择此方案。 |
| 无法复现 | 开发和测试工程师没法复现又不能解决的问题，并且跟踪测试二个以上版本也不能复现。 | 开发和测试工程师经过努力也不能复现，并由测试工程师跟踪测试二个以上版本，开发工程师才能选择此方案。 |
| 延期解决 | 开发工程师Bug进行解决时，发现或评估其解决时间紧或风险比较大等，向项目经理说明原因。 | 项目经理的权限，项目经理综合通过考虑解决问题的时间、风险、市场需求等多方面要素决定是否选择此方案 |

注：无法复现问题将作为风险评估点。

5.5 缺陷描述规范

（1）缺陷标题

 缺陷标题是对所提交缺陷的概述，需要简短而准确的描述出缺陷概要信息，并使用一些精炼的关键词，主要内容包括：位置+对象+动作+现象。

1. 环境关键词：包括数据环境，时间环境，地点环境条件环境，描述缺陷发生时所处的背景环境，或正在执行的操作或所处的界面环境，如“在…”页面，“当…时…”，“在…过程中…”等；
2. 动作关键词：引发缺陷所执行的操作，如“点…键”“选…选项”等；
3. 对象的关键词：描述缺陷出现的对象，“页面”，“显示框” ，“图表”等；
4. 现象的关键词：描述缺陷出现的现象，如“显示为负数”，“卡死”等。

根据上述关键词的组合来描写缺陷标题。

（2）重现步骤

在描述缺陷重现步骤的过程中，通常需要通过描述前提条件，测试步骤，实际结果，期望结果这四个方面清楚详细的描述缺陷。

1. 前提条件

外部环境，这里包括网络环境，硬件环境和软件环境的状态。默认情况下，无需特殊说明，前提条件均为“系统正常运行”，其含义为网络正常，电脑硬件环境能支撑软件运行，系统软件配置情况正常。需要注意，软件环境有可能引发缺陷的功能模块所处的状态，以及重现该缺陷需要的模块相关状态或者特殊设置情况应该前在前提条件中做说明。

数据环境，对缺陷产生的所在案件或引发bug现象的数据输入和数据设计等应该在前提条件中做相应说明。

总之，这里对缺陷现象重现紧密相关的预先设置，或与缺陷模块相关联的预先设置都应该在前提条件中说明。

1. 测试步骤

这里需要详细描述出重现缺陷的操作步骤，以便于重现缺陷，修复和验证缺陷。在描写测试步骤时应该注意以下几点：

精简：只描述缺陷必须的细节；

单一：每个缺陷单只报告一个缺陷；

步骤清晰：详细的、有序的描述出每一个步骤，包括输入的数据情况，执行的操作以及执行操作的界面。

操作量化：对操作次数的描述需要量化，如“连续3次点确认键”等，尽量避免出现“多次”等模糊的词。

1. 实际结果

实际结果是指按照测试步骤操作后实际反映的情况，这里指的是缺陷的表现现象。这里应该详细描述出错误的现象，包括页面错误，连接错误，字符错误，业务流程错误，数据错误等。

1. 期望结果

期望结果是指按照测试步骤操作，正常情况下正确执行测试步骤后应该满足设计要求的结果。对于不确定的期望结果就需要用到如建议，提示等关键字。

（3） 附件

为了方便开发工程师分析和解决，创建缺陷时需要添加必要的附件，包括缺陷现象图、测试的数据文档，测试时用到的其他特殊文件，测试脚本，操作步骤的录像等。

所以，测试人员在测试的过程中，要求每个缺陷有现象截图，以便协助开发人员快速定位问题。