修复缺陷流程1.0.0

# 修复缺陷－核心流程图

开始

软件缺陷定义

确认缺陷的属性

定位缺陷原因

失败

修复缺陷问题

总结意见建议

缺陷修复结果

成功

# 修复缺陷备注

## 2.1 确认缺陷属性

### 2.1.1 输入

1. 文档 软件缺陷定义

2. 缺陷提交者的反馈信息

### 2.1.2 过程

1. 开发人员通过缺陷再现、代码调试等方法进行缺陷属性的确认。

2. 开发人员汇总缺陷属性与缺陷提交者进行沟通确认。

3. 保证开发人员与缺陷提交者持有统一的缺陷属性。

### 2.1.3 输出

1. 开发人员列出缺陷相关属性。

1.1 缺陷类型

1.2 缺陷级别

1.3 缺陷产生可能

1.4 缺陷的优先级

1.5 缺陷状态

1.6 缺陷再现流程

1.7 缺陷所在软件的版本号

2. 根据缺陷属性特征与项目经理进行沟通。

2.1 非常见缺陷类型或难以判断出来缺陷类型，通知项目经理。

2.2 缺陷级别太高，通知项目经理。

2.3 缺陷产生可能性小，通知项目经理对应的处理方法。

2.4 缺陷的优先级高，通知项目经理。

## 2.2 定位缺陷原因

### 2.2.1 输入

1. 软件缺陷定义

2. 解决问题－麦肯锡方法（可选）

3. 上一步的输出

### 2.2.2 过程

1. 100%确认缺陷原因：

1.1 通过观察缺陷现象、缺陷提交者的沟通，判断缺陷原因。

1.2 进行源代码调试，并定位到出现问题的代码段，判断缺陷原因。

2. 80%确认缺陷原因：

2.1 分析应用的日志／系统日志，判断缺陷原因。

2.2 使用源代码版本工具对比代码

2.2.1 回滚源代码版本到上一次没有缺陷的版本。

2.2.2 对比回滚版本与最新代码的差异。

2.2.2.1 功能代码的差异

2.2.2.2 工程配置文件的差异

2.2 对比软件环境

2.2.1 对比软件开发环境

2.2.1.1 依赖环境的版本号

2.2.1.1.1 第三方依赖版本号

2.2.2 对比软件的安装环境

2.2.2.1 依赖环境的版本号。

2.2.2.1.1第三方依赖版本号

2.2.2.1.2编译软件版本号

2.2.2.1.3系统版本号

2.2.2.1.4等等

2.2.3 对比环境变化后的相关影响因素

3. 20%确认缺陷原因

3.1 麦肯锡7步法

3.1.1进行头脑风暴搜集所有的可能性因素

3.1.1.1采用头脑风暴的方式

3.1.1.2并记录下来所有的可能性因素

3.1.2 按照一定的规则，将可能性因素进行优先排序

3.1.3 对可能性因素进行单个排除、多个排除测试

3.1.3.1 测试缺陷消失，则定位引发缺陷的原因，并记录测试结果

3.1.3.2测试缺陷没有消息，执行下一个或多个可能性因素的测试，并记录测试结果

3.1.4 直到问题解决、或测试完所有的可能性因素

3.1.5 汇报结果给项目经理

3.2 可替换的解决方案

3.2.1 从备用方案中选择最优方案，从新进行开发

3.2.2 创建基础的功能代码

3.2.3 功能逐个进行添加

3.2.4 确保每个功能添加时没有缺陷

3.3 放松放松

3.3.1 出去走一走，玩一玩

3.3.2 给大脑预留些时间整理思路

3.4 最终无法定位原因

3.4.1 向上级领导汇报，寻求协助

3.4.2 汇报自己的调研过程结果

4. 无法确认缺陷原因

4.1 同20%确认缺陷原因方法

### 2.2.3 输出

1. 阶段性结果汇报

1.1 缺陷的原因

1.2 缺陷的可能性因素及测试结果

1.3 缺陷的解决方法

## 2.3 修复缺陷问题

### 2.3.1 过程

1. 根据上一步的输出内容，进行缺陷的修复

## 2.4 汇总意见

1. 针对缺陷产生的原因，提出建设性的意见。

# 文档版本

1.0.0