软件项目管理培训讲座

系列之六——Bug管理规范

内容提要

- Bug管理的基本概念和内容
- Bug管理的原则与方法
- Bug管理的工具和技巧

软件Bug的基本概念

软件系统中所有未能够满足功能要求，或未达到性能目标的缺陷，都称为**Bug**。

描述一个**Bug**，需要完整的属性和类型定义

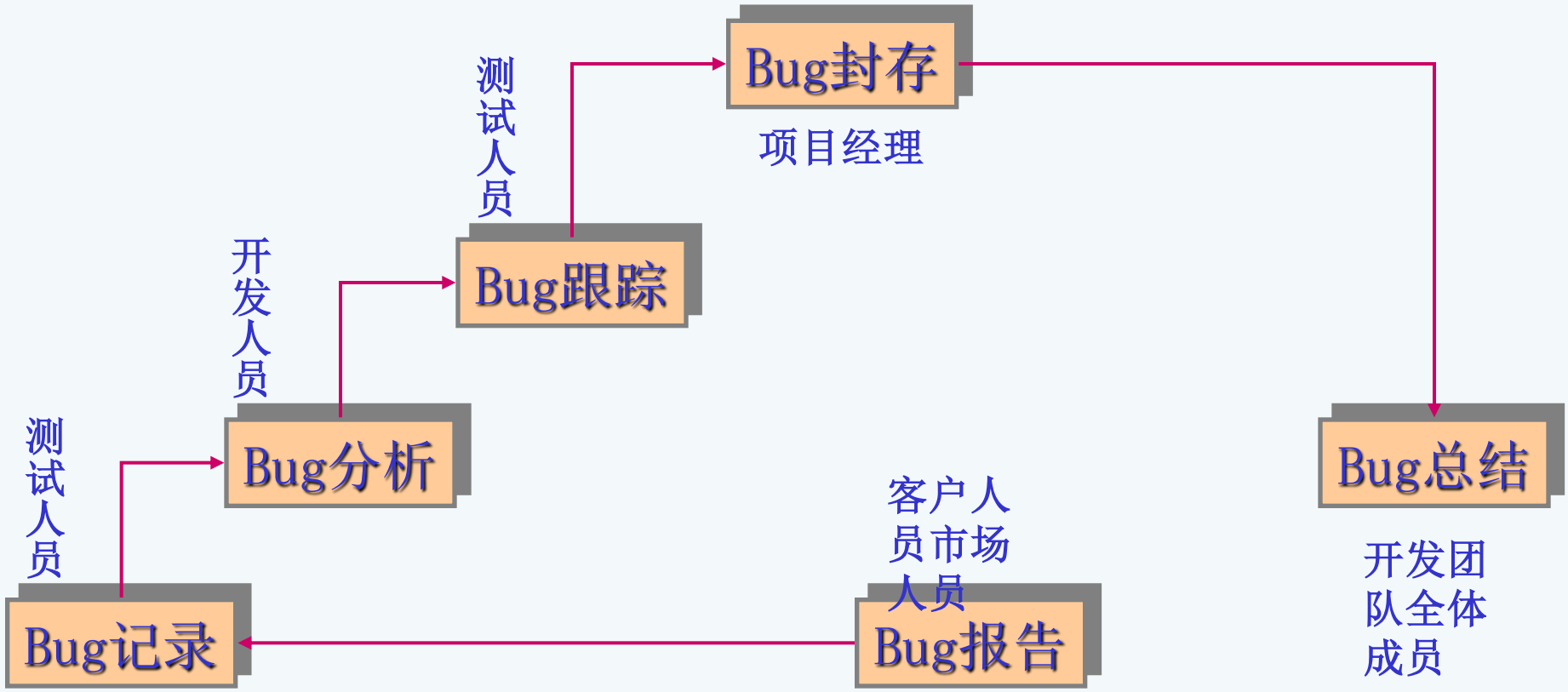
对Bug的客观认识

- 软件开发的过程中，Bug无处不再
  - 需求分析和设计方案中的Bug
  - 代码的功能和性能Bug
  - 系统配置、发布过程中的Bug
- Bug的形式丰富多样
  - 设计类Bug：错误的需求理解或系统结构
  - 技术类Bug：笔误、流程错误、算法错误
  - 文档类Bug：版本陈旧、配置不一致……

Bug管理的内容

- 实施测试过程，及时发现各种Bug
  - Bug发现越早，团队损失越小
- 进行分析评估，指导Bug的解决
  - 以最小的代价解决Bug，驱动开发工作
- 全程跟踪检查，保证系统的稳定
  - “山重水复疑无路，柳暗花明又一村”
- 适时总结分析，提升开发经验
  - 温故而知新，从教训中汲取经验

Bug管理的工作流程


Bug封存
项目经理
测试人员
Bug跟踪
开发人员
Bug分析
测试人员
Bug记录
客户人员市场人员
Bug报告
Bug总结
开发团队全体成员

Bug管理的环境配置

- 规范统一的名称和角色分工
  - 防止开发人员和测试人员的交流障碍
- 完整严谨的Bug格式记录说明
  - 发现问题只是解决问题的第一步
- 可记录的Bug跟踪过程（文档或数据库环境）
  - 切忌头痛医头、脚痛医脚
- 稳定通畅的信息交流渠道
  - 测试——开发、测试——管理之间的信息交流

内容提要

- Bug管理的基本概念和内容
- Bug管理的原则与方法
- Bug管理的工具和技巧

Bug管理的基本原则

- 明确团队分工，确定责、权、利
  - 心理因素：开发人员的接受程度
- 了解Bug周期，设定合理策略
  - 生生不息的Bug，是软件团队心头永远的痛
- 建立规范完整的内部交流机制
  - 及时发现、及时分析、及时解决
- 保持质量、时间、成本的平衡
  - 软件开发的艺术性决定了Bug解决的艺术性

Bug的描述方法

- 标识
  - 唯一标记Bug的序号
- 类型
  - Bug表现形式的分类
- 严重程度
  - Bug对开发成果的影响
- 优先级
  - Bug被修复的紧急程度
- 状态
  - Bug生命周期
- 起源
  - Who，What，Description
- 来源
  - 原始操作或者原始数据
- 根源
  - 分析结论，影响Bug状态变化

对Bug描述的理解

- 谁来负责描述Bug
  - 测试人员：标识、严重程度、优先级、起源、来源
  - 技术开发人员：类型、根源
  - 共同维护：Bug状态
- 如何看待Bug描述
  - 严重程度和优先级的区别：一粒老鼠屎坏了一锅粥
  - Bug状态的变化过程：完整的生命周期
- Bug描述的一些误区
  - 可重现的才是Bug：说明起源和来源很重要
  - 分析也是一种描述：根源往往藏的很深
  - 完整的描述是财富：切忌语焉不详，造成资源浪费

Bug的类型

缺陷类型编号	缺陷类型	描述
10	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。如逻辑，指针，循环，递归，功能等缺陷。
20	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。
30	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。
40	C- Checking	提示的错误信息，不适当的数据验证等缺陷。
50	B- Build/package/merge	由于配置库、变更管理或版本控制引起的错误。
60	D- Documentation	影响发布和维护，包括注释。
70	G- Algorithm	算法错误。
80	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。
90	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。
100	N-Norms	不符合各种标准的要求，如编码标准、设计符号等。

Bug分类统计

- 一份对6877000行源代码进行测试的报告分析结果：

错误分类	百分比	错误分类	百分比
需求错误	8.1%	集成错误	9.0%
功能和性能错误	16.2%	系统结构错误	1.7%
结果错误	25.2%	测试错误	2.8%
数据错误	22.4%	其他错误	4.7%
实现和编码错误	9.95%		

Bug的严重程度

#	缺陷严重等级	描述
1	Critical 严重缺陷	不能执行正常工作功能或重要功能。或者危及人身安全。
2	Major 较大缺陷	严重地影响系统要求或基本功能的实现，且没有办法更正。（重新安装或重新启动该软件不属于更正办法）
3	Minor 较小缺陷	严重地影响系统要求或基本功能的实现，但存在合理的更正办法。（重新安装或重新启动该软件不属于更正办法）
4	Cosmetic 轻微缺陷	使操作者不方便或遇到麻烦，但它不影响执行工作功能或重要功能。
5	Other 其他缺陷	其它错误。

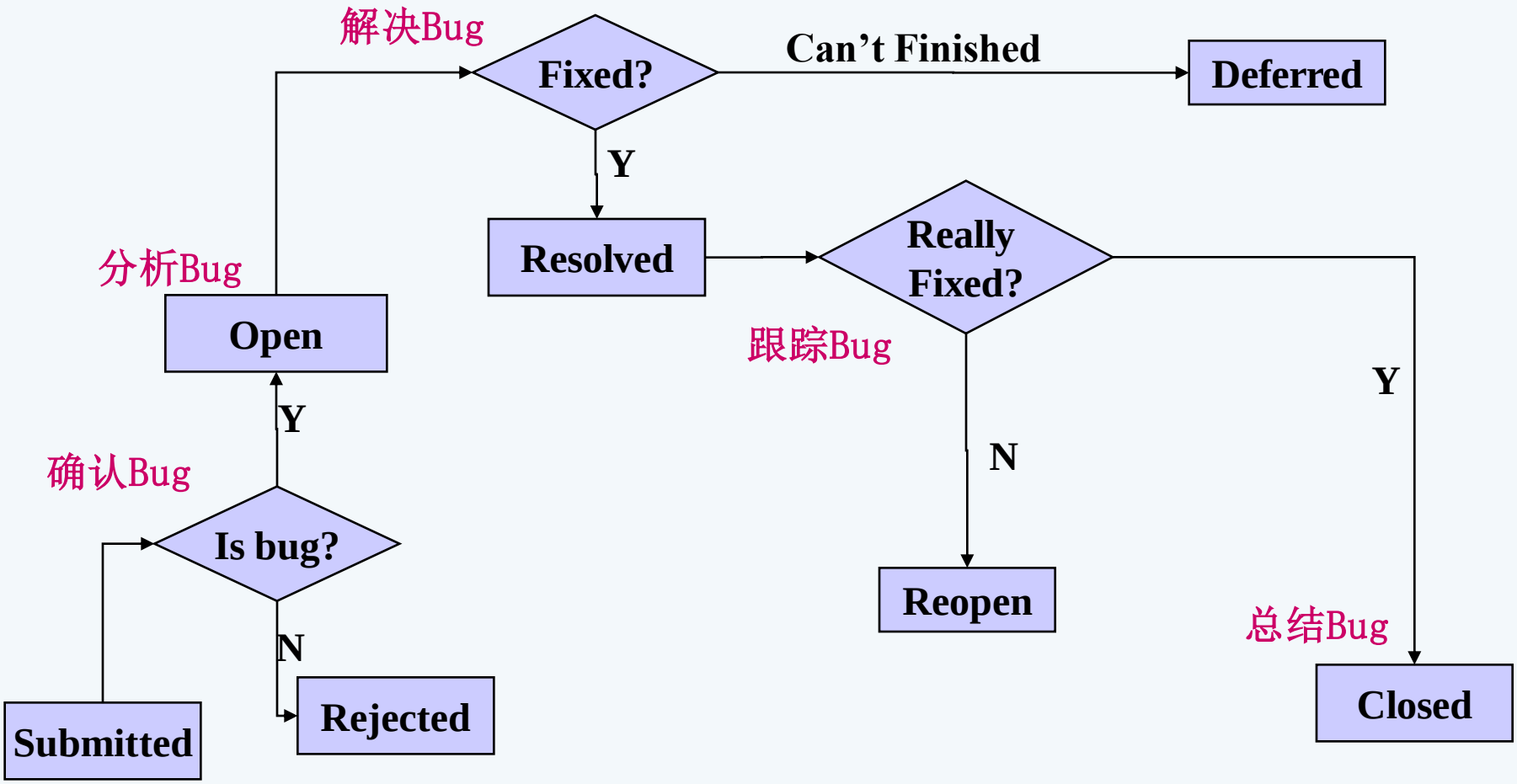
Bug的优先级

#	缺陷优先级	描述
1	Resolve Immediately 理解解决	缺陷必须被立即解决。
2	Normal Queue 正常排队	缺陷需要正常排队等待修复或列入软件发布清单。
3	Not Urgent 不紧急	缺陷可以在方便时被纠正。

Bug的状态

缺陷状态	描述
Submitted	已提交的缺陷
Open	确认“提交的缺陷”，等待处理
Rejected	拒绝“提交的缺陷”，不需要修复或不是缺陷
Resolved	缺陷被修复
Reopen	缺陷再一次出现（反复出现，说明没有真正解决）
Deferred	推迟修复（不重要或者难以解决）
Closed	确认被修复的缺陷，将其关闭

Bug生命周期


解决Bug
Fixed?
Can't Finished
Deferred
Y
Resolved
Really Fixed?
分析Bug
Open
跟踪Bug
Y
N
Y
确认Bug
Is bug?
Reopen
总结Bug
N
Closed
Submitted
Rejected

软件开发周期与Bug生命周期

- 越早发现Bug，其解决代价越小

阶段	代价
需求分析	1
设计	5
编码	20
测试	50
维护	100

Bug管理的基本方法

- 需求分析的评审
  - 项目经理、技术人员、测试人员、客户人员共同参与
  - 降低需求分析理解错误造成的Bug
- 设计方案的讨论
  - 技术人员和测试人员讨论，项目经理负责
  - 降低设计缺陷造成的Bug
- 代码自查与互查
  - 技术人员：互相检查代码，剔除笔误、流程和算法错误
  - 向测试人员提供看起来没有错误的程序
- Bug的过程跟踪
  - 有经验的测试人员：严格保证每个Bug的真实性
  - 项目经理、测试人员和技术人员：测试驱动开发

内容提要

- Bug管理的基本概念和内容
- Bug管理的原则与方法
- Bug管理的工具和技巧

Bug管理的一些技巧

- Bug Base的建设
  - 不要轻易的相信Bug
  - 真实案例：ScanSoft项目的教训
- 对Bug完整清晰的描述
  - 语焉不详的描述会带来不必要的浪费
  - 真实案例：掌纹自动鉴别系统的教训
- 公正、友好的团队气氛
  - 指出Bug并不会伤害开发人员的自尊
  - 真实案例：Table项目中的教训
- 最重要的一点——开发人员和测试人员的协作
  - 这个世界不是缺少Bug，而是缺少发现

Bug管理的一些技巧

- 一定要选用Bug管理工具吗？
  - 长期维护或者重复出现的项目，最好使用Bug管理工具
  - 一般的小型项目可用Excel、Mail来记录Bug
- Bug管理与代码管理的协同
  - 重要原则1：进行测试时，必须进行Code Freezing
  - 重要原则2：分析未完成时，不应急于修改代码
- Bug解决的策略
  - 尽可能的消除Bug出现的原因，而不是增加处理解决Bug
  - 不到万不得已，不应Defer Bug，后患无穷
  - 必须有良好的代码规范，防止Bug的Reopen

Bug管理工具

- 开发过程中使用一套BUG管理软件非常必要
- 常用工具
  - CompuWare: TrackRecord
  - Mozilla: Buzilla
  - Microsoft: ATS
  - 微创：BMS
  - Rational ClearQuest
  - 可自主研发小型的Bug管理工具

Bug Base的案例剖析

- Table处理的Bug Base
- MiniCheck的Bug描述

结束语

- Bug管理是保证软件开发质量的核心工作
- Bug管理需要团队全体人员的通力配合
- 提高软件质量的重要手段——使Bug尽量少

Thanks for your time!

Questions & Answers