第2章 软件缺陷

张圣林
zhangsl@nankai.edu.cn
http://nkcs.iops.ai/courses/softwaretesting/

2 软件缺陷

- 2.1 软件缺陷概述
  - 2.1.1 软件缺陷的定义
  - 2.1.2 软件缺陷分析
  - 2.1.3 软件缺陷的种类
  - 2.1.4 软件缺陷的产生
  - 2.1.5 软件缺陷数目估计
- 2.2 软件缺陷管理
  - 2.2.1 缺陷管理的目标
  - 2.2.2 缺陷报告
  - 2.2.3 软件缺陷管理流程
  - 2.2.4 缺陷管理工具

2.1 软件缺陷概述

- 2.1.1 软件缺陷的定义
  - 软件没有达到产品说明书表明的功能
  - 程序中存在语法错误
  - 程序中存在拼写错误
  - 程序中存在不正确的程序语句
  - 软件出现了产品说明书中不一致的表现
  - 软件功能超出产品说明书的范围
  - 软件没有达到用户期望的目标
  - 测试员或用户认为软件的易用性差

2.1 软件缺陷概述

- 2.1.1 软件缺陷的定义
  - 按照缺陷的来源，软件缺陷分为文档缺陷、代码缺陷、测试缺陷、过程缺陷
  - 文档缺陷
    - 文档在静态检查过程中发现的缺陷，通过测试需求分析、文档审查对被分析或被审查的文档发现的缺陷
  - 代码缺陷
    - 对代码进行同行评审、审计或代码走查过程中发现的缺陷

2.1 软件缺陷概述

- 2.1.1 软件缺陷的定义
  - 测试缺陷
    - 由测试执行活动发现的被测对象（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
      - 内部测试、连接测试、系统集成测试、用户验收测试，测试活动中发现的缺陷为测试缺陷。
  - 过程缺陷
    - 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。过程缺陷发现者一般是质量经理、测试经理、管理人员等。

2.1 软件缺陷概述

- 2.1.2 软件缺陷分析
  - 软件缺陷是影响软件质量的重要与关键因素之一，发现与排除软件缺陷是软件生命周期中的重要工作之一。
  - 每一个软件组织都知道必须妥善处理软件中的缺陷。这是关系到软件组织生存、发展的质量根本。
  - 发现与排除软件缺陷需要大量的花费。
  - 美国国防部的数据表明，在IT产品中，大约42%的资金是用于与软件缺陷相关的工作上。
  - 目前在美国，软件测试的花费占整个软件费用的53%～87%。因此，对软件缺陷及其相关问题进行研究是极为有价值的。

2.1 软件缺陷概述

- 2.1.2 软件缺陷分析
  - 软件工程师在工作中一般会引入大量的缺陷。
    - 统计表明，有经验的软件工程师的缺陷引入率一般是50～250个缺陷/KLOC，平均的缺陷引入率在100个缺陷/KLOC以上。即使软件工程师学过软件缺陷管理之后，平均的缺陷引入率也在50个缺陷/KLOC。
  - 目前高水平的软件组织所生产的软件可以达到的缺陷密度为2～4个缺陷/KLOC，一般的软件组织所生产的软件其缺陷密度为4～40个缺陷/KLOC，NASA的软件的缺陷密度可以达到0.1个缺陷/KLOC。

2.1 软件缺陷概述

- 2.1.2 软件缺陷分析
  - 开发低缺陷密度的软件需要大量的花费。在上世纪九十年代，NASA的软件平均一行代码需要$1000
  - 影响软件缺陷数目的因素很多。在不同的软件阶段，软件的缺陷密度是不同的。
    - 从宏观上看，包括管理水平、技术水平、测试水平等。
    - 从微观上看，软件规模、软件复杂性、软件类型、测试工具、测试自动化程度、测试支撑环境、 开发成本等。初始的软件缺陷密度一般是靠经验来估计的。

2.1 软件缺陷概述

- 2.1.3 软件缺陷的种类–按照缺陷本质
  - 输入/输出缺陷

类型	举例
输入	不接受正确输入 接受不正确输入 描述有错或遗漏 参数有错或遗漏
输出	格式有错 结果有错 在错误的时间产生正确的结果 不一致或遗漏结果 不合逻辑的结果 拼写/语法错误 修饰词错误

2.1 软件缺陷概述

- 2.1.3 软件缺陷的种类
  - 逻辑缺陷
    - 遗漏情况
    - 重复情况
    - 极端条件出错
    - 解释有错
    - 遗漏条件
    - 外部条件有错
    - 错误变量的测试
    - 不正确的循环迭代
    - 错误的操作符

2.1 软件缺陷概述

- 2.1.3 软件缺陷的种类
  - 计算缺陷
    - 不正确的算法
    - 遗漏计算
    - 不正确的操作数
    - 不正确的操作
    - 括号错误
    - 精度不够
    - 错误的内置函数

2.1 软件缺陷概述

- 2.1.3 软件缺陷的种类
  - 接口缺陷
    - 不正确的中断处理
    - I/O时序有错
    - 调用了错误的过程
    - 调用了不存在的过程
    - 参数不匹配
    - 不兼容的类型
    - 过量的包含

2.1 软件缺陷概述

- 2.1.3 软件缺陷的种类
  - 数据缺陷
    - 不正确的初始化
    - 不正确的存储/访问
    - 错误的标志/索引值
    - 不正确的打包/拆包
    - 使用了错误的变量
    - 错误的数据引用
    - 缩放数据范围或单位错误
    - 不正确的数据维数
    - 不正确的下标
    - 不正确的类型
    - 不正确的数据范围
    - 传感器数据超出限制
    - 出现1次断开
    - 不一致数据

2.1 软件缺陷概述

- 2.1.4 软件缺陷的产生
  - 疏忽造成的错误（Carelessness defect，CD）
  - 不理解造成的错误（Misapprehend defect，MD）
  - 二义性造成的错误（Ambiguity defect，AD）
  - 遗漏造成的错误（Skip defect，SD）

2.1.4 软件缺陷的产生

- 程序编写错误
  - Bug的难以避免性
- 需求变更过于频繁
  - 需求变更所造成的结果就是变更程序代码，程序代码只要稍做变更就必须经过测试来确保运行正常，所以这个影响是一个连锁反应或称为依存问题
- 软件的复杂度
  - 图形用户界面（GUI）、B/S 结构、面向对象设计、分布式运算、底层通信协议、超大型关系型数据库以及庞大的系统规模，都体现了软件复杂度大大高于以前，Bug出现可能性就更高

2.1.4 软件缺陷的产生

- 交流不充分或者沟通出问题
  - 大部分项目人员在同客户进行交流时常常存在着各种各样的问题，究其原因，还是因为项目人员、参与人员和客户之间没有详细、充分、谨慎地进行交流
- 测试人员的经验与技巧不足
- 时间过于紧迫
- 缺乏文档
  - 贫乏或者差劲的文档使得代码维护和修改变得非常困难，结果会导致其他开发人员或客户有许多错误的理解
- 管理上的缺陷

2.1 软件缺陷概述

- 2.1.5 软件缺陷数目估计
  - 1 撒播模型
  - 2 静态模型
  - 3 覆盖率预测模型

1 撒播模型

- 乒乓球法（参入黑球）
  - 撒播模型是利用概率论的方法来估算程序中的错误数目，其基本原理类似于估计一个大箱子中存放的乒乓球的数目。
  - 假设一个大箱子里有许多白色的乒乓球N，向箱子里置入M个黑色的乒乓球，并将箱中的球搅拌均匀，然后，从箱子中随机的取出足够多的球，假设白色的乒乓球有n个，黑色的乒乓球m个，则可以根据下列公式估计N

$$\frac{N}{N+M}=\frac{n}{n+m}$$

$$N=\frac{n}{m}*M$$

1 撒播模型

- Mills缺陷预测（植入缺陷）
  - 用人工随机的向待估算的软件置入错误；然后进行测试，待测试到足够长的时间后，对所测试到的错误进行分类，看看哪个是人工置入的错误，哪个是程序中固有的错误；最后，根据上述公式即可估算出程序中所有的错误。
  - 缺点
    - 程序中固有的缺陷是未知的，每个错误被检测的难易程度也同样是未知的。
    - 人工置入的缺陷是否和程序中存在缺陷检测的难易程度一致也是未知的。

1 撒播模型

- Hyman缺陷预测（相同缺陷）
  - 假设软件总的排错时间是X个月，即经过X个月的排错时间，假设程序中将不再存在错误。
  - 让两个人共同对程序进行排错，假设经过足够长(X的一半或更少)的排错时间后，第一个人发现了n个错误，第二个人发现了m个错误，其中属于两个人共同发现的错误有m_1个，则公式为：

$$\frac{N}{n} = \frac{m}{m_1}$$

$$N = \frac{m}{m_1} * n$$

由于软件缺陷的复杂性，简单的撒播模型所估计的数值仅作为参考。

2 静态模型

- Akiyama模型
- 谓词模型
- Halstead模型
- Lipow模型
- Gaffnev模型
- Compton and Withrow模型

2 静态模型

- Akiyama模型
  - N=4.86+0.018*L
  - 其中：N是缺陷数；L是可执行的源语句数目。
  - 基本的估计是1KLOC有23个缺陷。这是早期的研究成果。该模型可能对某个人或某类专门的程序是有效的，模型的提出明显是一种实践上的统计结果。该模型太简单了，实际价值不大。

2 静态模型

- 谓词模型
  - N=C+J
  - 其中：C是谓词数目，J是子程序数目。
  - 程序的许多错误来自于程序中包含的谓词。但将每个谓词和子程序都假定一个错误也显然是不合适的，但也没有其他更好的办法来准确地描述它们之间的关系。总之，该模型也有一定的参考价值。

2 静态模型

- Halstead模型
  - N=V/3000
  - 其中：V=x*lny，x=x1+x2，y=y1+y2
  - x1：程序中使用操作符的总次数；
  - x2：程序中使用操作数的总次数；
  - y1：程序中使用操作符的种类；
  - y2：程序中使用操作数的种类；
  - 根据Halstead的理论，V是程序的体积，即程序占内存的比特数目，该模型认为，平均3000bit就有一个错误。该模型和Akiyama模型有些类似，也完全是大量程序的统计结果，但难以说清楚哪一个更好。

2 静态模型

- Lipow模型
  - $N=L*(A0+A1*InL+A2*\ln^2 L)$
  - Fortran语言：A0=0.0047，A1=0.023，A2=0.000043。
  - 汇编语言：A0=0.0012，A1=0.0001，A2=0.000002。
  - 显然，这也是一个统计结果。不同的是，该模型区分了高级语言和低级语言。

2 静态模型

- Gaffnev模型
  - $N=4.2+0.0015L^{4/3}$
  - 解此方程可推断出一个模块的最佳尺寸是877LOC。

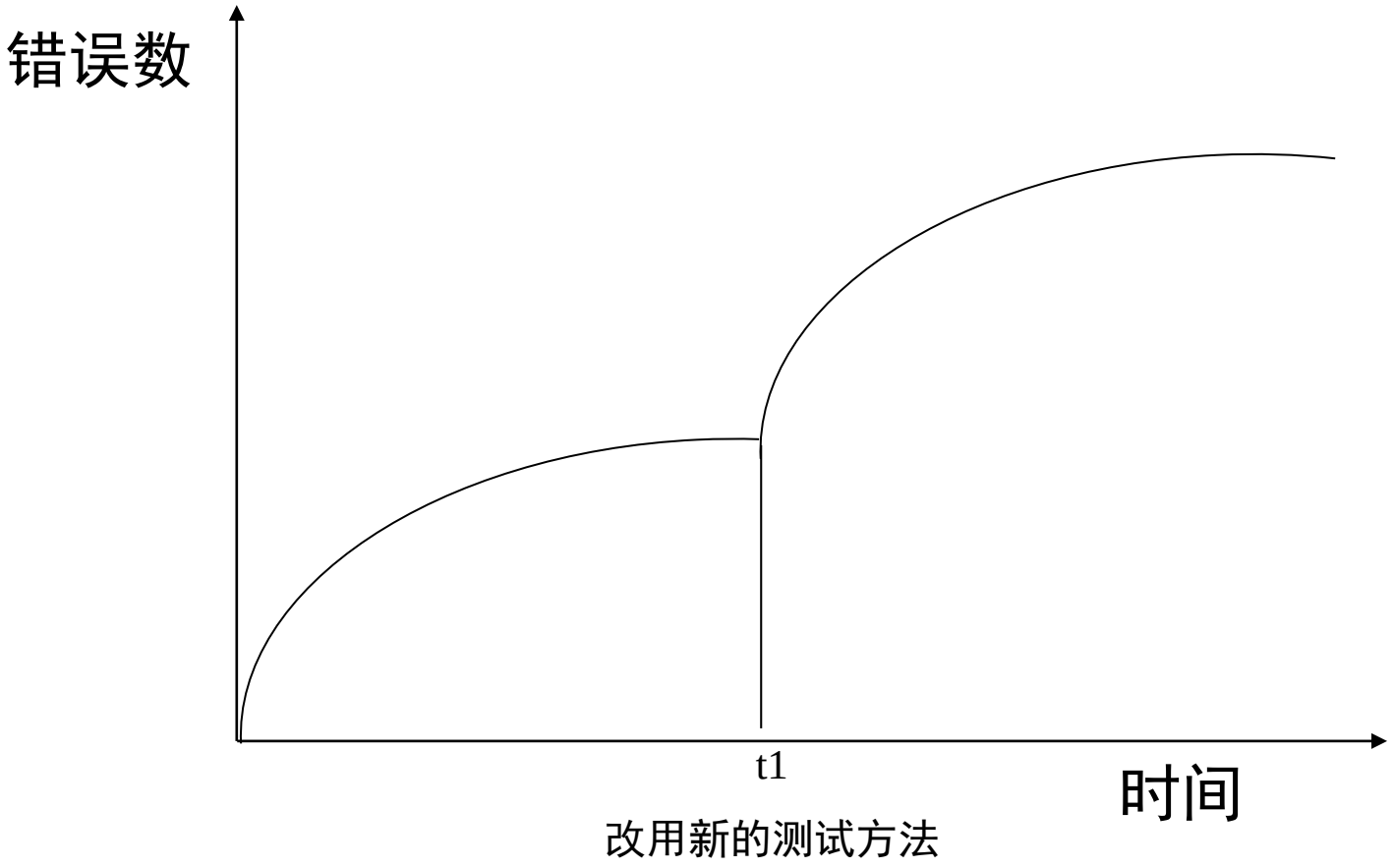
2 静态模型

- Compton and Withrow模型
  - $N=0.069+0.00156L+0.00000047L^2$
  - 由该方程可推断出一个模块的最佳尺寸是83LOC

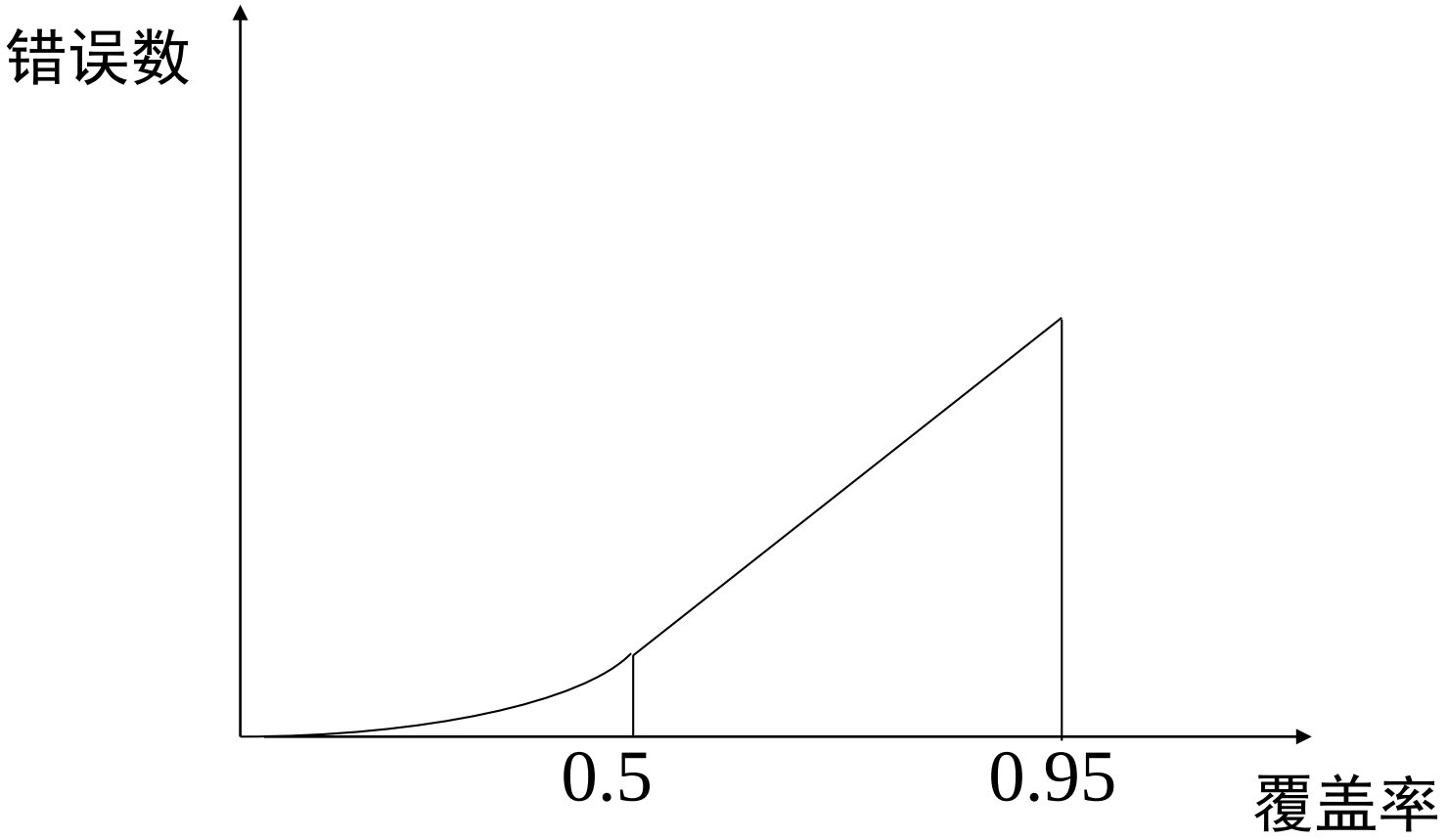
3 覆盖率预测模型

- 错误与时间曲线
- 错误与覆盖率曲线
- 覆盖率与时间曲线

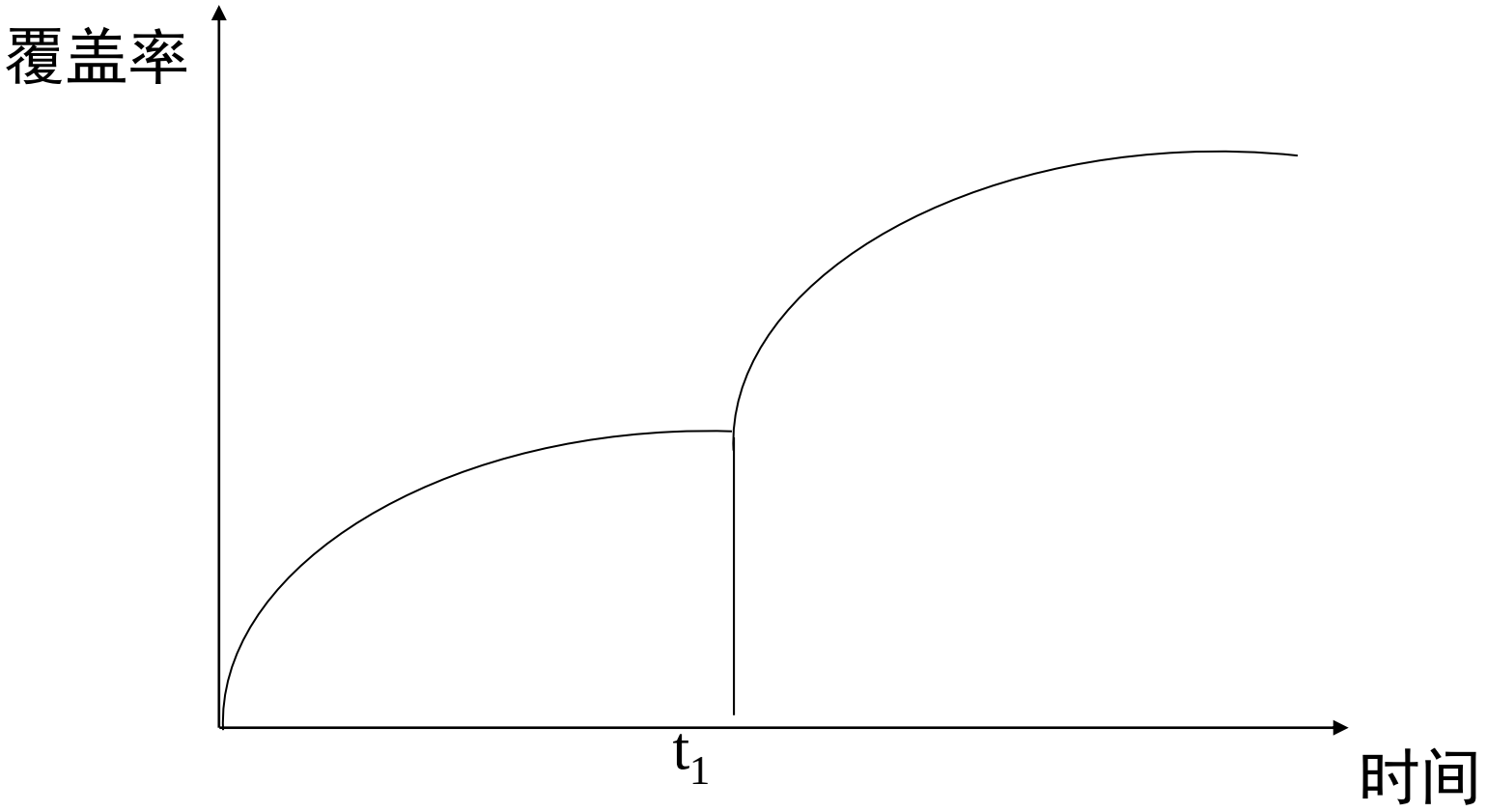
错误与时间



错误与覆盖率



覆盖率与时间



2.2 软件缺陷管理

- 2.2.1 缺陷管理的目标
  - 确保每个被发现的缺陷都能够被解决
    - 这里解决的意思不一定是被修正，也可能是其他处理方式(例如，在下一个版本中修正或不修正)。
  - 收集缺陷数据并根据缺陷趋势曲线识别测试过程的阶段
    - 决定测试过程是否结束有很多种方式，通过缺陷趋势曲线来确定测试过程是否结束是常用并且较为有效的一种方式。

2.2 软件缺陷管理

- 2.2.1 缺陷管理的目标
  - 收集缺陷数据并在其上进行数据分析，作为组织的过程财富
    - 在对软件缺陷进行管理时，必须先对软件缺陷数据进行收集，然后才能了解这些缺陷，并且找出预防和修复它们的方法，以及预防引入新的缺陷。

2.2.2 缺陷报告

- 缺陷报告的用途是什么？
  - 记录缺陷
  - 缺陷分类
  - 缺陷跟踪
- 为什么要尽早的报告缺陷？
- 是不是所有的缺陷都会被修复？

缺陷报告的内容

- 缺陷报告的内容
  - 缺陷报告也叫缺陷报告单或者问题报告单
- 问题报告单所需的基本信息类型是大同小异的，不同的只是组织和标志
  - 问题报告编号
  - 程序名
  - 版本标识：发布号和版本号
    - 用来识别被测的代码。
    - 例如：某个版本号可能是1.01m，发布号是1.01，“m”指1.01版本的第13稿

缺陷报告的内容

- 报告类型：描述了发现的问题类型。包括：编码错误 、设计问题 、建议 、文档 、硬件、质疑
- 严重性：为问题严重程度评分
  - 三个等级：轻微的、严重的和致命的
- 附件
  - 存有测试数据的软盘、键盘捕获记录或一组可产生测试用例的宏、程序的打印输出、内存dump或一份注释，里面详细描述了你所做的操作，以及你认为该问题很重要的原因
- 问题概要：对问题进行描述，有助于评审突出的问题，并找到相应的问题报告

缺陷报告的内容

- 问题能否重现：要多次测试看能否再次出现
- 问题描述及如何重现：描述所有的步骤和现象，包括错误信息，一定要提交报告
- 建议的改正措施
- 报告人
- 日期：指的是报告人员发现问题的日期
- 功能域：对问题进行大体分类
- 承办人
- 注释：程序员在这里简短地说明为什么要推迟处理，或说明是如何改正问题的

缺陷报告的内容

- 状态
  - 三个状态码：开放、关闭和已解决
- 优先级：只由项目经理设置
- 处理状态与处理版本
  - 处理状态定义了问题的当前状态
    - 未解决：初始化状态或仍有冲突状态
    - 已改正
    - 不能重现
    - 暂缓：对存在的问题在下个版本改正
    - 符合设计
    - 由报告人撤回

缺陷报告的内容

- 需要进一步信息
- 不同意建议
- 重复：关闭重复上报的缺陷

- 签名：签署以表明已经对改动进行了测试，对结果表示满意
- 暂缓处理：对缺陷的推迟处理

缺陷报告的特点

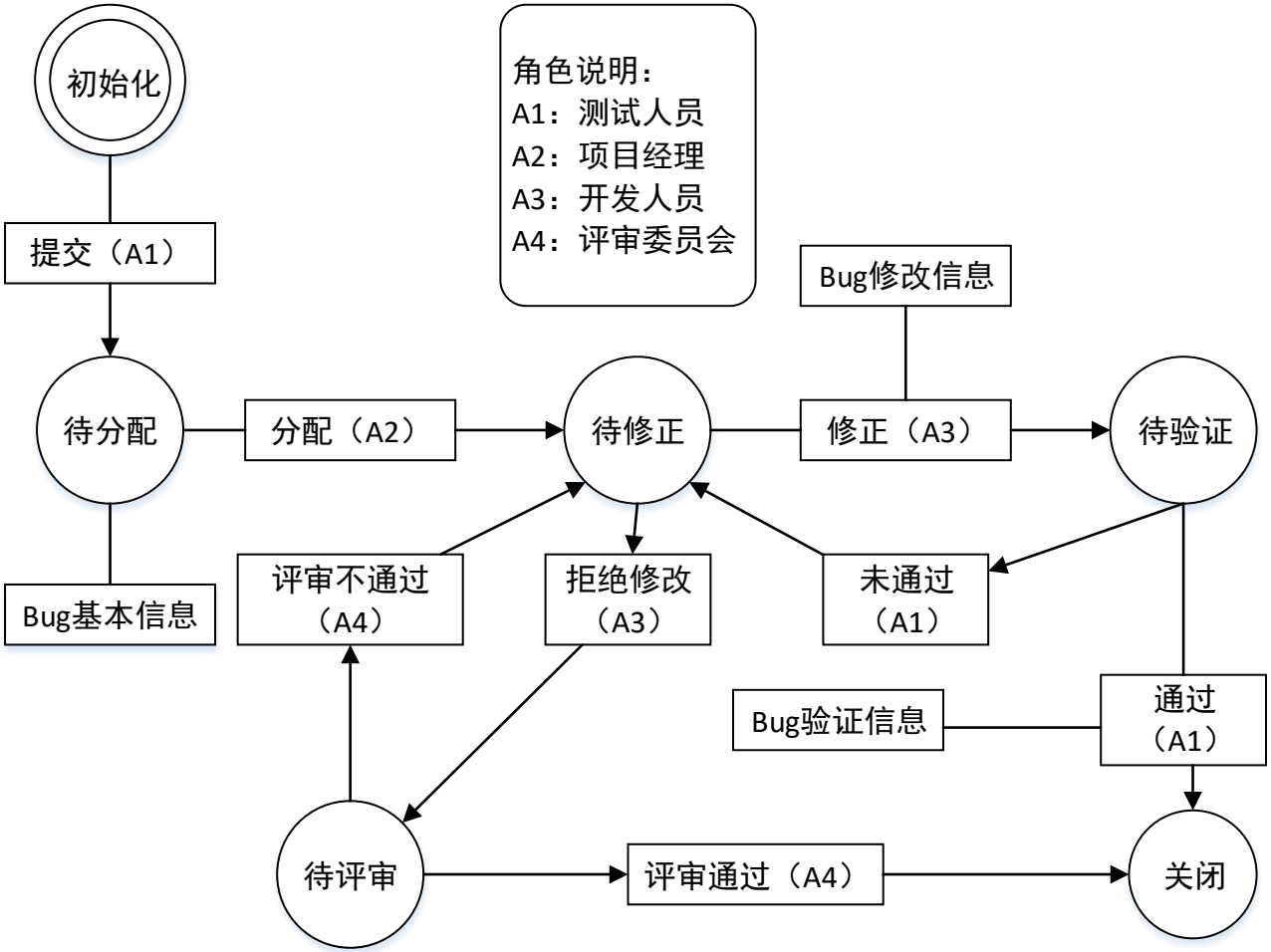
- 书面的
  - 一份书面的报告是必要的，供日后对修改后的程序进行测试时使用；让管理层、销售人员和产品支持人员检查
- 已编号的
  - 依据惟一的编号跟踪问题报告
- 简单的
  - 一份报告应只描述一个问题，不要在一份报告中记录多个缺陷

缺陷报告的特点

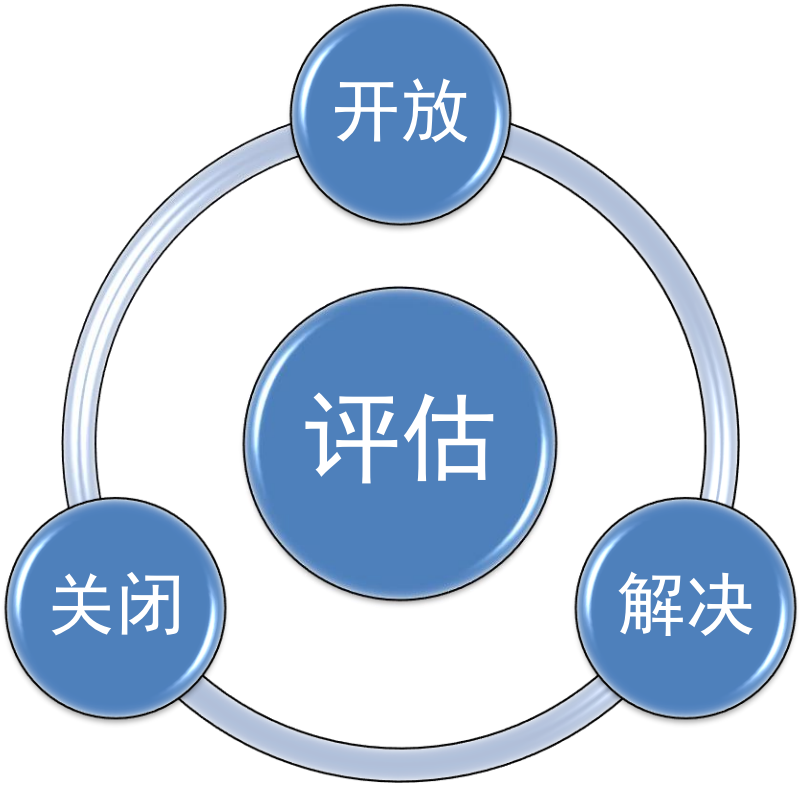
- 可重现的
  - 一定要强调BUG的可重现性
- 不做判断的
  - 对程序员的评价要三思而后行，本着合作的精神，作出合理的判断

2.2 软件缺陷管理

- 2.2.3 软件缺陷管理流程

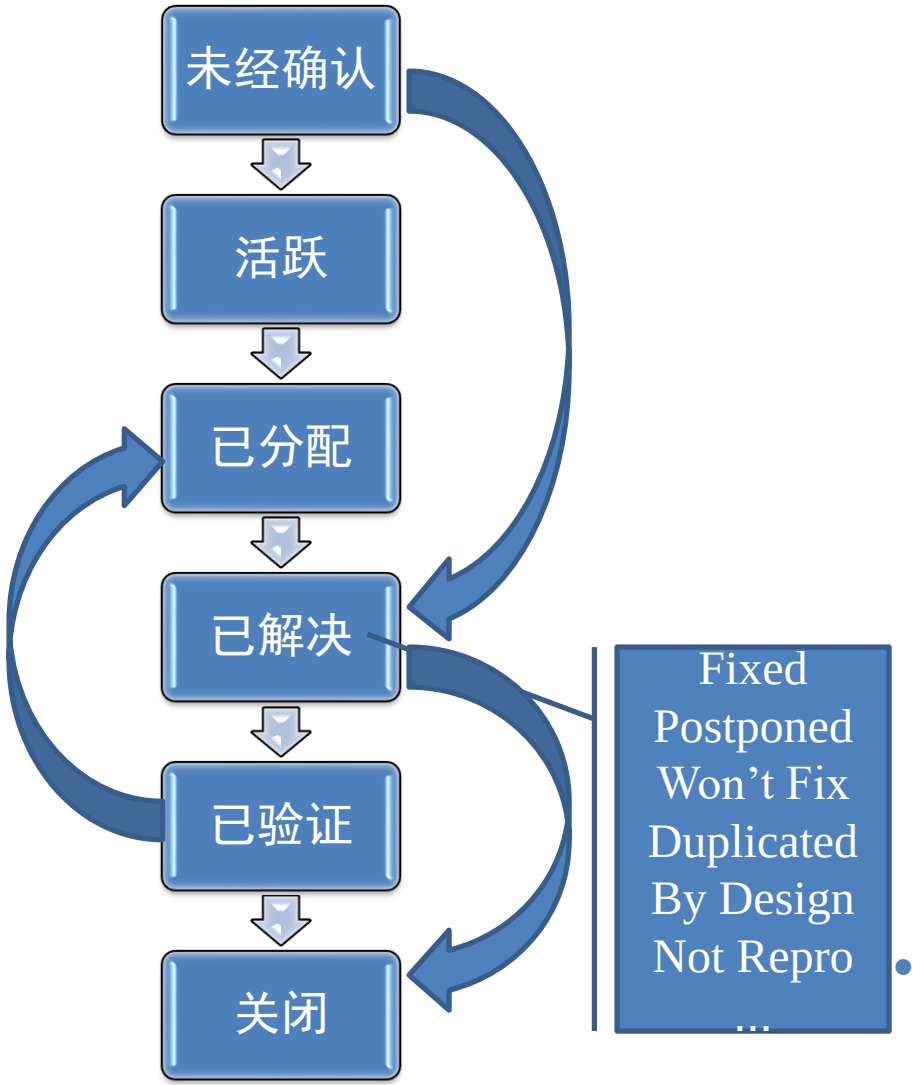


缺陷的生命周期



- 评估（Review）是缺陷处理的核心。由项目经理或者委员会组决定缺陷如何处理
- 缺陷一旦被发现并且记录下来，就处于开放（Open）状态
- 经过评估，决定是否解决，由谁来解决
- 找到解决方案的软件缺陷（Resolved），必须经过评估和测试验证，才能最终关闭（Closed）

基本流程



- 关键状态
  - 未经确认（Unconfirmed）
    - 直接来自最终用户或系统外部
  - 活跃(Active/Open)
    - 新的缺陷
    - 直接来自内部（开发或测试）团队
    - 曾经关闭的缺陷，被重新激活
  - 已分配（Assigned）
    - 经过评估，分配给相关开发人员处理
    - 开发人员自己发现的缺陷
    - 未通过测试的解决方案，需要进一步研究
  - 已解决（Resolved）
    - 经过评估确认的解决方案
  - 已验证（Verified）
    - 测试通过，但是需要后续处理 - 归档，完善测试覆盖，原因分析
    - 测试不通过，重新分配给开发人员
  - 关闭（Close）
    - 处理完毕
- 实际情况不必经过每一个状态

2.2 软件缺陷管理

- 2.2.4 缺陷管理工具
  - 缺陷管理工具用于集中管理软件测试过程中发现的错误，是添加、修改、排序、查寻、存储软件测试错误的数据库程序。
  - 缺陷管理工具的使用使得跟踪和监控错误的处理过程和方法更加容易，既可以方便地检查处理方法是否正确，也可以确定处理者的姓名和处理时间，作为统计和考核工作质量的参考。

2.2 软件缺陷管理

- 2.2.4 缺陷管理工具
  - TrackRecord（商用）
    - 作为Compuware项目管理软件集成的一个重要组成部分，TrackRecord目前已经拥有众多的企业级用户，它基于传统的缺陷管理思想，整个缺陷处理流程完备，界面设计精细，并且对缺陷管理数据进行了初步的加工处理，提供了一定的图形表示。

2.2 软件缺陷管理

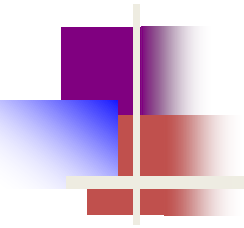
- 2.2.4 缺陷管理工具
  - ClearQuest（商用）
    - IBM Rational C1earQuest是基于团队的缺陷和变更跟踪解决方案，是一套高度灵活的缺陷和变更跟踪系统，适用于在任何平台上、任何类型的项目中，捕获各种类型的变更。
    - http://www03.ibm.com/software/products/zh/clearquest/

2.2 软件缺陷管理

- 2.2.4 缺陷管理工具
  - Bugzilla （开源）
    - Bugzilla 作为一个开源免费软件，拥有许多商业软件所不具备的特点，现在已经成为全球许多组织喜欢的缺陷管理软件。
    - http://www.bugzilla.cn/

2.2 软件缺陷管理

- 2.2.4 缺陷管理工具
  - Buggit（开源）
    - Buggit是一个十分小巧的C/S 结构的Access 应用软件，仅限于Intranet. 使用十分简单，能满足基本的缺陷跟踪功能
  - Mantis（开源）
    - Mantis是一款基于Web 的软件缺陷管理工具，配置和使用都很简单，适合中小型软件开发团队
    - http://www.mantis.org.cn/
  - HP Quality Center（简称QC，商用）中的缺陷管理模块



谢谢！