SOFTWARE TESTING

Software Testing

Software Defect Management (1)

School of Computer Science & Engineering
Sun Yat-sen University

Instructor: Guoyang Cai
email: isscgy@mail.sysu.edu.cn

Approaches & Technologies

OUTLINE

- 8.1 概述
- 8.2 软件缺陷描述与分类
- 8.3 软件缺陷的处理与跟踪
- 8.4 软件缺陷报告
- 8.5 软件缺陷的度量与分析
- 8.6 软件缺陷管理工具

软件缺陷的概念

软件缺陷定义

- 软件缺陷是软件 (文档、数据、程序) 之中存在的不希望、或不可接受的偏差，这些偏差导致软件产生质量问题。
  - 软件缺陷是对软件产品预期属性的偏离现象：软件缺陷导致软件产品在某种程度上不能满足用户的需求。
- *Ron Patton* (Software Testing, 2005)：至少满足以下5个规则之一，才称为发生了一个软件缺陷：

(1) 软件未实现产品说明书要求的功能；
(2) 软件出现了产品说明书指明不应该出现的错误；
(3) 软件实现了产品说明书未提到的功能；
(4) 软件未实现产品说明书虽未明确提及但应该实现的目标；
(5) 软件难以理解，不易使用，运行缓慢或者—从测试员的角度看—最终用户会认为不好。

- 软件缺陷的概念
  - 软件缺陷定义 (续)
    - 软件缺陷的主要表现：
      - 约定的功能、特性没有实现或只是部分实现。
      - 设计不合理。
      - 实际结果和预期结果不一致。
      - 数据结果不正确、精度不够。
      - 运行出错，包括运行中断、系统崩溃、界面混乱。
      - 用户界面不友好。
      - 性能低下。
      - 用户不能接受的其他问题。
        - 如存取时间过长、界面不美观。
    - 参见 Lec.6-Chap.2.1-2.2 Introduction to Software Defect & Testing

软件缺陷的概念

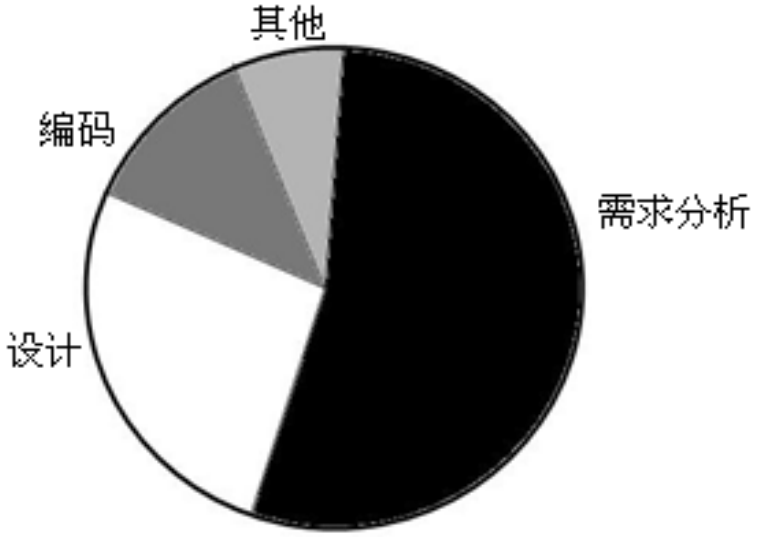
软件缺陷定义 (续)

- 软件缺陷既指程序中存在的错误 (例如语法错误、拼写错误或语义错误)，也指可能出现在需求规格说明、设计说明以及其他的文档中的错误。这些错误导致软件故障和失效。
- 软件缺陷包括检测缺陷和残留缺陷两大类。
  - 检测缺陷 (Detected Defect) 指软件在进入用户使用阶段之前被检测出的缺陷。
  - 残留缺陷 (Residual Defect) 指软件发布后存在的缺陷，包括在用户安装前未被检测出的缺陷以及检测出但未被修复的缺陷。

- 软件缺陷的概念
  - 与软件缺陷有关的系统风险分析内容
    - 系统的某一部分产生了错误所导致的结果
    - 未被验证的数据交换被接受所导致的结果
    - 文件的完整性被破坏所导致的结果
    - 系统是否能被安全恢复 (完全恢复成备份时的状态)
    - 系统是否能被暂停运行
    - 系统的操作流程是否符合用户的组织策略和长远规划
    - 系统性能在进行维护工作时是否会下降到不能接受的水平
    - 系统的安全性是否有保证
    - 系统是否可靠、稳定
    - 系统是否易于使用
    - 系统是否方便维护
    - 系统是否易于与其它系统相连

软件缺陷的产生原因

- 导致软件产生缺陷的九类原因
  - (1) 客户—开发者沟通失败/失误
  - (2) 软件需求定义不完善
  - (3) 软件需求偏离
  - (4) 设计的逻辑性错误
  - (5) 编码错误
  - (6) 文档编制与编码不符合规定
  - (7) 测试过程不足
  - (8) 规程错误
  - (9) 文档编制错误



软件缺陷的来源

- 研究表明：大多数软件缺陷并不是由于编码造成的。软件缺陷主要产生于需求分析阶段 (54%)，其次是软件设计阶段 (25%)，再次才是编码阶段 (15%)。

软件缺陷的产生原因

- 需求分析缺陷是产生软件缺陷的最大来源
  - 软件需求规格说明书描述了系统应该具备和不应该具备的功能清单，以及功能的操作性和性能、接口等具体规格，它是开发流程与测试流程的输入源头。
  - 客户—开发者之间的沟通失败，可以造成需求规格说明的不完善或者是对软件需求的偏离；在开发过程中需求规格说明的变更，会造成设计、编码与需求之间的偏离。
  - 需求缺陷经过开发过程的处理后被放大，修复需要耗费大量人力、物力和时间成本，总体成本显著上升。因此必须从早期就开始对需求规格说明书进行审查并建立基线。
    - 测试人员应该在需求被基线化前介入开发流程，参与需求评审，尽早从客户/测试的角度找出所有不合理/不明确/不可行的需求，以减少后期的开发与测试成本。

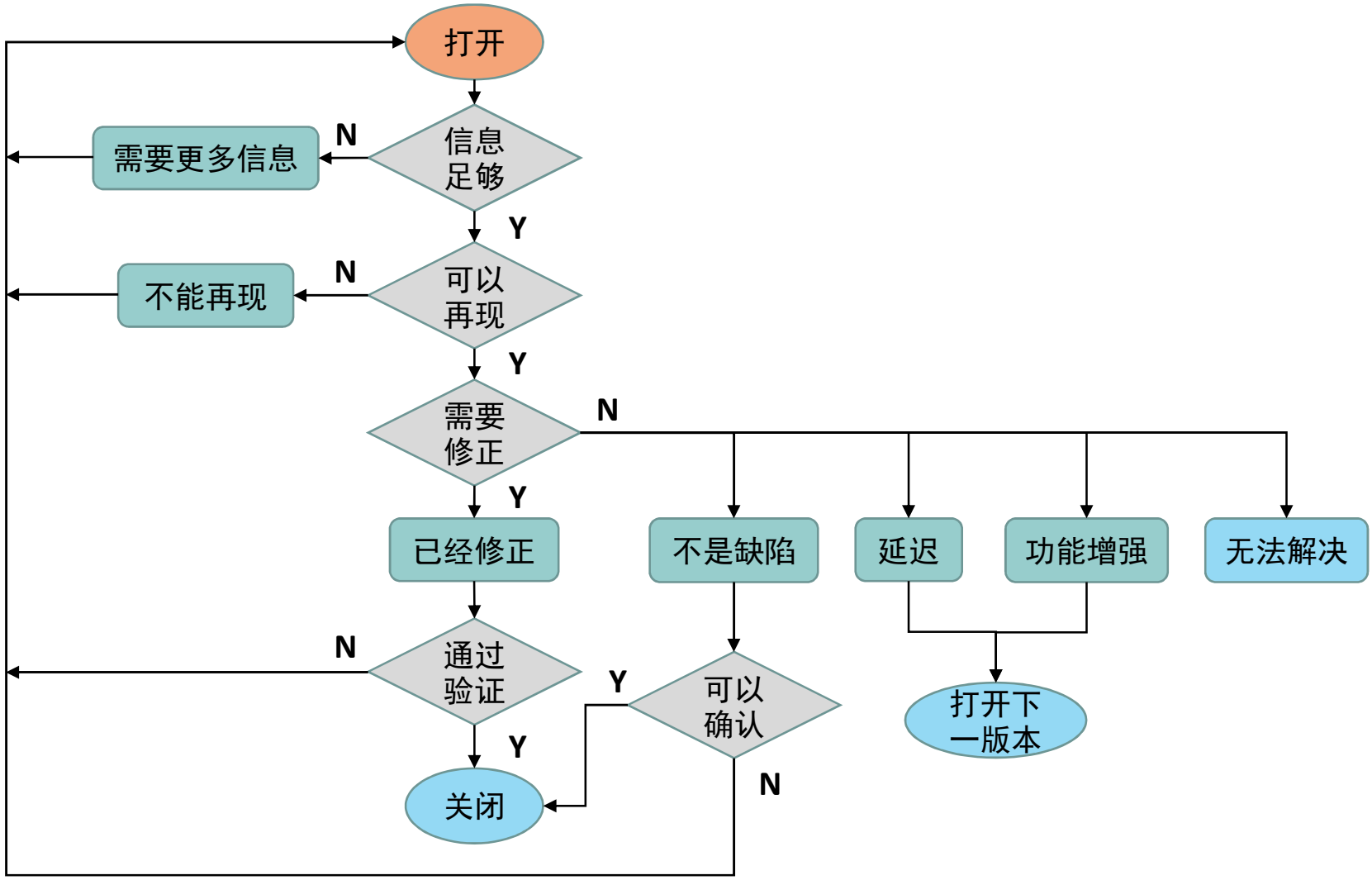
- 软件缺陷的产生原因
  - 设计阶段是产生软件缺陷的另一个主要来源
    - 系统设计阶段是软件系统的技术规划过程，设计人员的专业技术训练水平、对业务逻辑的认知能力等的不足都可能导致设计阶段出现逻辑性错误。
    - 设计方案的变更，加上开发团队的沟通问题，都会导致在设计阶段引入软件缺陷。
  - 编码阶段是产生软件缺陷的来源之一
    - 编码阶段通常由于代码错误而造成软件缺陷。代码错误可能源于软件设计复杂、文档不足、进度压力、易犯的低级错误，或者是程序员的习惯性错误。

- 软件缺陷的典型类型
  - 典型的软件缺陷
    - 需求解释有误、用户定义的需求有误、需求记录错误
    - 设计说明有误
    - 编码说明有误、程序代码有误、数据输入有误
    - 测试过程有误、回归修改新增错误
    - 正确的结果是由于多个缺陷合并产生的 (多缺陷错误)

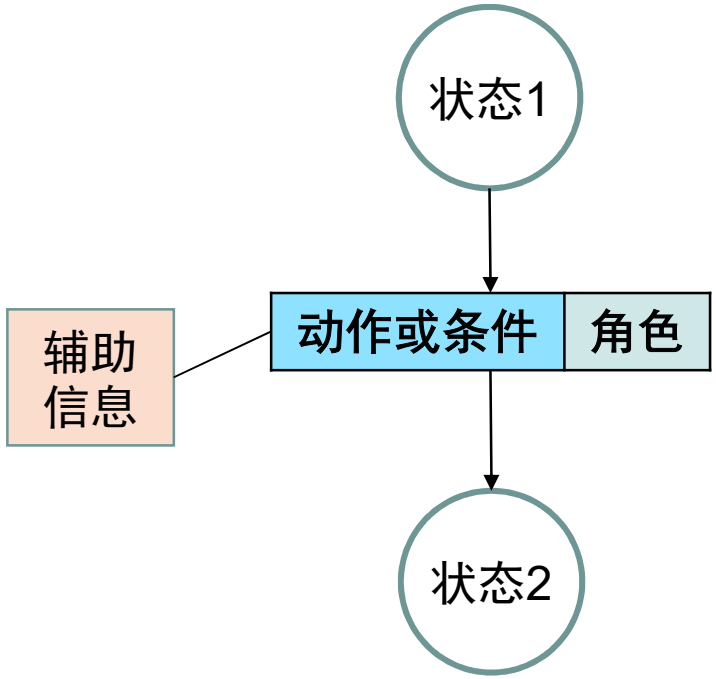
- 软件缺陷的检测和修复
  - 软件缺陷的检测和修复
    - 存在着无法发现的软件缺陷。
      - 软件测试并不能保证发现所有的缺陷。
    - 存在着无法重现的软件缺陷。
      - 有些软件缺陷虽然在测试过程中被发现，但是测试人员无法使其重现。这样的软件缺陷不能得到修复。
    - 存在着无法修复的软件缺陷。
      - 软件缺陷原则上必须修复。但因时间、风险、成本等管理上的原因，产品开发团队可能决定对一些软件缺陷不作修复。

- 软件缺陷的跟踪
  - 软件缺陷跟踪
    - 软件缺陷的跟踪是软件测试工作的重要组成部分。
      - 软件测试的目的就是为了尽早发现软件系统中的缺陷，软件测试过程围绕着缺陷进行。
      - 对软件缺陷进行跟踪管理，确保每一个被发现的缺陷都能够及时得到处理是软件测试工作的重要内容。
    - 软件缺陷的跟踪需要达到以下的目标：
      - 确保每一个被发现的缺陷都能够得到解决；
      - 收集缺陷数据并根据缺陷趋势曲线识别测试过程的阶段；
      - 收集缺陷数据并进行数据分析，从中获得与软件质量相关的数据。

软件缺陷的生命周期

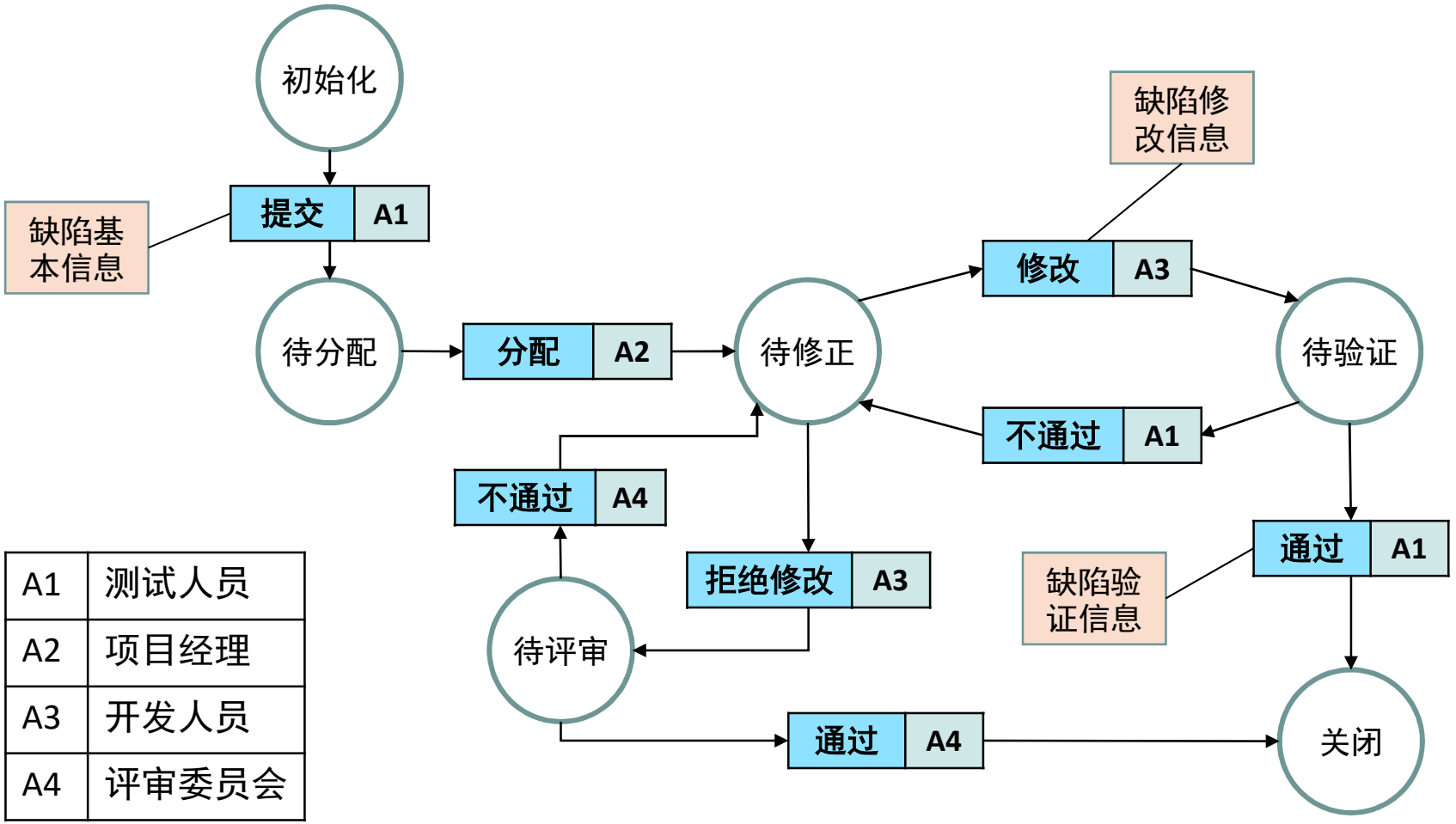

打开
信息足够
N
需要更多信息
Y
可以再现
N
不能再现
Y
需要修正
N
Y
已经修正
不是缺陷
延迟
功能增强
无法解决
通过验证
N
Y
可以确认
Y
打开下一版本
关闭
N

- 软件缺陷的管理流程
  - 状态转移图例



软件缺陷的管理流程

软件缺陷的一般管理流程



- 软件缺陷的管理流程
  - 软件缺陷的状态
    - 初始化
      - 缺陷的初始状态。
    - 待分配
      - 缺陷等待分配给相关开发人员处理。
    - 待修正
      - 缺陷等待开发人员修正。
    - 待验证
      - 开发人员已完成修改，等待测试人员验证。
    - 待评审
      - 开发人员拒绝修改缺陷，需要评审委员会评审。
    - 关闭
      - 缺陷已经处理完毕。

- 软件缺陷的管理流程
  - 软件缺陷管理流程中的角色
    - A1：测试人员
      - 执行软件测试的人员，是缺陷的发现者。
    - A2：项目经理
      - 对整个项目负责，对产品质量负责的人员。
    - A3：开发人员
      - 执行开发任务的人员，完成实际的设计和编码工作。
    - A4：评审委员会
      - 对缺陷进行最终确认，在项目成员对缺陷达不成一致意见时，行使仲裁权力。

软件缺陷的管理流程

- 软件缺陷管理流程的要点
  - 软件错误的确认
    - 由具备丰富测试经验的测试人员验证和确认发现的错误是否是真正的错误，必要时需要与委托方做充分的交流。
  - 软件错误的处理
    - 每次对软件错误的处理都要保留处理信息，包括处理者姓名、时间、处理方法、处理步骤、错误状态、注释等。
  - 软件错误的拒绝
    - 由项目经理，测试经理和设计经理共同决定，而不能由程序员单方面决定。
  - 软件错误的延期处理由软件开发方决定。
  - 软件错误修复后必须由报告该错误的测试人员验证并确认已经修复，才能关闭。
  - 由独立的评审委员会对争议行使裁决权。

- 软件缺陷的描述
  - 软件缺陷描述的意义
    - 软件缺陷的描述是软件缺陷报告中测试人员对问题的陈述的一部分，并且是软件缺陷报告的基础部分。
    - 软件缺陷的描述也是测试人员就一个软件问题与开发小组交流的最初而且最好的机会。一个好的描述，需要使用简单的、准确的、专业的语言来抓住缺陷的本质。

- 软件缺陷的描述
  - 软件缺陷的有效描述规则
    - 单一缺陷准则
      - 每个报告只针对一个软件缺陷。
      - 在一个报告中报告多个软件缺陷可能导致只有其中一个缺陷得到注意和修复。
        - 多缺陷报告的例：联机帮助文档中下述不同的15页中的单词拼写错误。登录对话框不接受大写字母输入的口令或者登录 ID。
    - 可再现性
    - 完整性和统一性
      - 提供完整、前后统一的软件缺陷的修复步骤和信息。
    - 特定条件描述
      - 例：搜索功能在没有找到结果时返回的跳转页面不对。

- 软件缺陷的描述
  - 软件缺陷的有效描述规则
    - 简洁性
    - 时效性
      - 尽快报告发现的软件缺陷。
    - 中立性
      - 软件缺陷报告是针对产品的，软件缺陷描述不要带有个人观点，不要对开发人员进行评价。

- 软件缺陷的描述
  - 软件缺陷描述内容
    - 缺陷 ID
      - 缺陷 ID 具有唯一性，可以根据该 ID 追踪缺陷。
    - 缺陷基本信息
      - 缺陷标题：描述缺陷的标题。
      - 缺陷严重程度：描述缺陷的严重程度，比如分为“致命”、“严重”、“一般”、“建议”四种。
      - 缺陷紧急程度：描述缺陷的紧急程度，比如分为1-4级，1级是优先级最高的等级，4级是优先级最低的等级。
      - 缺陷提交人：缺陷提交人的信息 (名字、邮件地址)。
      - 缺陷提交时间：缺陷首次提交的时间。
      - 缺陷所属项目/模块：缺陷所属的项目或模块，最好能精确定位到模块。

- 软件缺陷的描述
  - 软件缺陷描述内容
    - 缺陷基本信息 (续)
      - 缺陷指定解决人：缺陷状态为“提交”时为空；缺陷状态为“分发”时由项目经理指定的开发人员。
      - 缺陷指定解决时间：项目经理指定的开发人员解决此缺陷的截止时间。
      - 缺陷处理人：最终处理缺陷的人员。
      - 缺陷处理结果描述：对处理结果的描述。如果对代码进行了修改，要求在此处体现出修改的具体细节。
      - 缺陷处理时间：对缺陷处理的实际耗费时间。
      - 缺陷验证人：对被缺陷处理结果进行验证的人员。
      - 缺陷验证结果描述：对验证结果的描述 (通过、不通过)。
      - 缺陷验证时间：对缺陷进行验证的时间。

- 软件缺陷的描述
  - 软件缺陷描述内容
    - 缺陷的详细描述
      - 缺陷描述应该尽可能详细。缺陷描述的详细程度将直接影响开发人员对缺陷的修改。
    - 测试环境说明
      - 对测试环境的描述。
    - 必要的附件
      - 对于某些文字很难表达清楚的缺陷，有必要使用图片等附件。
    - 其它
      - 从统计的角度出发，还可以添加上“缺陷引入阶段”、“缺陷修正工作量”等项目。

- 软件缺陷的描述
  - 例：手工缺陷报告样本

WIDGETS SOFTWARE INC. **BUG REPORT** BUG#:_____

SOFTWARE:_____RELEASE:_____VERSION:_____

TESTER:_____DATE:_____ASSIGNED TO:_____

SEVERITY: 1 2 3 4 PRIORITY: 1 2 3 4 REPRODUCIBLE: Y N

TITLE:_____

DESCRIPTION:_____

RESOLUTION: FIXED DUPLICATE NO-REPRO CAN'T FIX DEFERRED WON'T FIX

DATE RESOLVED:_____RESOLVED BY:_____VERSION:_____

RESOLUTION COMMENT:_____

RETESTED BY:_____VERSION TESTED:_____DATE TESTED:_____

RETEST COMMENT:_____

SIGNATURES:

ORIGINATOR:_____ TESTER:_____

PROGRAMMER:_____ PROJECT MANAGER:_____

MARKETING:_____ PRODUCT SUPPORT:_____

- 软件缺陷的描述
  - 软件缺陷属性
    - 缺陷属性描述
      - 缺陷标识 (Identifier)：标记某一个缺陷的一组符号。每个缺陷必须有一个唯一的标识。
      - 缺陷类型 (Type)：根据缺陷的自然属性划分的缺陷种类。
      - 缺陷严重程度 (Severity)：缺陷引起的故障对软件产品的影响程度。
      - 缺陷优先级 (Priority)：缺陷必须被修复的紧急程度。
      - 缺陷状态 (Status)：缺陷通过跟踪修复过程的进展情况。
      - 缺陷起源 (Origin)：缺陷引起的故障或失效事件首次被检测到的开发阶段。
      - 缺陷来源 (Source)：引起缺陷的直接原因。
      - *缺陷根源 (Root Cause)：发生错误的根本因素。

- 软件缺陷的分类
  - 软件缺陷分类方法
    - *Putnam* 分类法
      - *Lawrence H. Putnam* (1992) 提出按照软件生命周期阶段将软件缺陷分为六类：需求类、设计类、文档类、算法类、界面类和性能类等六类缺陷。
    - GJB 437-1988/GJB 437A-97
      - GJB 437 标准将软件错误按照来源不同分为三类：程序错误、文档错误、设计错误。
    - *Putnam* 分类法和 GJB 437 标准分类比较简单，可以分析软件缺陷的来源和出处，指明修复缺陷的努力方向，为软件开发过程各项活动的改进提供线索。

- 软件缺陷的分类
  - 软件缺陷分类方法 (续)
    - *Thayer* 分类法
      - *Thayer* 分类法将软件缺陷按软件错误性质分类，利用测试人员在软件测试过程填写的问题报告和用户使用软件过程反馈的问题报告作为错误分类的信息。
      - *Thayer* 分类法包括了16个大类以及164个子类。16个大类包括：计算错误；逻辑错误；I/O 错误；数据加工错误；操作系统和支持软件错误；配置错误；接口错误；用户需求改变；预置数据库错误；全局变量错误；重复的错误；文档错误；需求实现错误；不明性质错误；人员操作错误；问题 (需要答复的问题)。
      - *Thayer* 分类法有利于分析错误分布状况，不适合用于过程改进。

- 软件缺陷的分类
  - 软件缺陷分类方法 (续)
    - IEEE 软件异常分类法 (IEEE Std 1044-2009 *IEEE Standard Classification for Software Anomalies*)
      - 提供一个统一的方法对软件和文档中发现的异常情况进行详细的分类，并提供异常的相关数据项帮助异常识别和异常跟踪活动。
      - Withdrawn Date:2020-03-05

- 软件缺陷的分类
  - 正交缺陷分类分析法 (ODC 分析法)
    - 正交缺陷分类 (ODC, Orthogonal Defects Classification, 1992) 是 IBM 提出的缺陷分析方法，包括分类、校验、评估、行动四个流程。它通过给每一个缺陷添加一些额外的属性，利用对这些属性的归纳和分析，反映产品的设计、代码质量、测试水平等各方面的问题。
      - 正交分类的含义是分类不重叠，而且具有统计的相对独立性。不同阶段的分类方案应该保持一致，才能看出各阶段间的变化趋势；分类应该与产品或组织的特性无关。
      - 正交缺陷分类分析法简称正交缺陷分析法，或 ODC 方法。
    - ODC缺陷分类方法提供一个从缺陷中提取关键信息的测量范例，将缺陷的多维语义信息与缺陷的原因和结果关联，用于对软件开发过程进行评价，提出正确的过程改进方案。

- 软件缺陷的分类
  - 正交缺陷分类分析法
    - ODC 方法针对一个缺陷采用多个属性来描述其缺陷特征。这些属性包括：
      - Activity 活动
      - Trigger 触发事件
      - Impact 影响
      - Target 载体/对象/修复目标
      - Defect Type 类型
      - Qualifier 限定词
      - Source 来源
      - Age 生存期
      - *Content Type 文档

- 软件缺陷的分类
  - 正交缺陷分类分析法
    - ODC 定义的缺陷属性
      - Activity：发现缺陷的活动。描述发现该缺陷的测试活动阶段，如：UT (单元测试)、FVT (功能测试)、SVT (系统测试)等。
      - Trigger：触发缺陷的事件。描述引发该缺陷所采取的测试方式。
      - Impact：缺陷影响。描述该缺陷的发生所造成的影响。
      - Target：缺陷载体。描述该缺陷的载体，即为了修复该缺陷所需要确定的工作对象，如产品设计、相应的代码和文档等。
      - Defect Type：缺陷类型。ODC 将缺陷分为赋值、检验、算法、时序、接口、功能、关联、文档 8 大类型。

- 软件缺陷的分类
  - 正交缺陷分类分析法
    - ODC 定义的缺陷属性 (续)
      - Qualifier：缺陷限定词。指出该缺陷是由于丢失相关代码、还是代码不正确造成的，或者是由于第三方提供的代码造成的。
      - Source：缺陷来源。指出该缺陷的来源是由机构内部编写代码引起的，还是由外包公司提供的代码引起等（注意这里的定义与之前的缺陷来源的定义不同）。
      - Age：缺陷生存期。指出该缺陷是由新代码产生的还是由于修改其它缺陷而引发的，或是在上一个发布版本中就已经有的问题等。
      - *Content Type：文档类型。表示修复文档的类型，仅对文档类的缺陷有效。

- 软件缺陷的分类
  - 正交缺陷分类分析法
    - ODC 缺陷属性对缺陷的消除和预防起到关键作用。在 8 种缺陷属性中，缺陷触发事件和缺陷类型是最为重要的信息，作为缺陷分类方案的认识基础。
    - 对应缺陷的发现和修复两类特定的活动，ODC 缺陷属性可分为两部分：
      - 发现缺陷时，导致缺陷暴露的环境和缺陷对用户可能的影响是易见的，此时可以确定缺陷的 3 个属性，它们为验证过程提供反馈：
        - 活动 (Activity)
        - 触发事件 (Trigger)
        - 影响 (Impact)

- 软件缺陷的分类
  - 正交缺陷分类分析法
    - 对应缺陷的发现和修复两类特定的活动，ODC 缺陷属性可分为两部分：(续)
      - 修复/关闭缺陷时，缺陷的性质和修复范围已得到确认，此时可以确定缺陷的其余 5 个属性，它们为开发过程提供反馈：
        - 缺陷载体 (Target)
        - 缺陷类型 (Type)
        - 缺陷限定词 (Qualifier)
        - 缺陷生存期 (Age)
        - 缺陷来源 (Source)

- 软件缺陷的分类
  - 正交缺陷分类分析法
    - ODC 的缺陷类型 (主要是设计、编码方面) 被分为 8 大类：
      - 赋值 (Assignment)
      - 检验 (Checking)
      - 算法 (Algorithm)
      - 时序 (Timing/Serialization)
      - 接口 (Interface)
      - 功能 (Function)
      - 关联 (Relationship)
      - 文档 (Documentation)

- 软件缺陷的分类
  - 正交缺陷分类分析法
    - ODC 方法对缺陷进行细致分类，用于缺陷的定位、排除、原因分析和缺陷预防。由缺陷属性描述的缺陷特征提供的丰富信息为缺陷的消除、预防和软件过程的详细分析改进奠定了基础。
    - ODC 方法的缺点在于分类过于复杂，分类标准难以把握，缺陷分析人员的主观意见会影响属性的确定。
    - ODC 方法既是缺陷分类方法，同时也是软件缺陷度量分析方法。它介于缺陷统计模型和因果分析方法之间，保持了类似于定量方法的低成本，效果上又可以达到定性分析的力度。
    - ODC 方法自提出后得到广泛的应用，并得到多个国际化软件组织的接受和使用。作为定量的测量和分析方法， ODC 方法已经成为 CMM 4/5 的支撑工具之一，为其定量过程管理、缺陷预防等提供有力支持。

- 软件缺陷的分类
  - 软件缺陷分类方法的应用
    - 参照 CMM、GJB 5000A-2008 军用软件研制能力成熟度模型、ODC 分析方法等质量过程控制标准，按照缺陷的表现形式、严重程度、优先级、起源和来源、根源、生命周期等进行缺陷分类。
    - 按缺陷类型标准 (表现形式/自然属性) 的分类：
      - 10 F-Function (功能)
        - 影响了重要特性、用户界面、产品接口、硬件结构接口和全局数据结构，并且导致需要正式变更设计文档的缺陷。
        - 例如：逻辑、指针、循环、递归、功能等缺陷。
      - 20 A-Assignment (赋值)
        - 需要修改少量代码 (如初始化或控制块) 的缺陷。
        - 例如：声明、重复命名、范围、限定等缺陷。

- 软件缺陷的分类
  - 软件缺陷分类方法的应用
    - 按缺陷类型标准 (表现形式/自然属性) 的分类 (续)
      - 30 I-Interface (接口/时序)
        - 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 (检查)
        - 提示信息错误、不适当的数据验证等缺陷。
      - 50 B-Build/package/merge (联编/打包)
        - 因配置库、变更管理或版本控制引起的错误。
      - 60 D-Documentation (文档)
        - 影响发布和维护的文档错误，包括注释。
      - 70 G-Algorithm (算法)
        - 算法错误。

- 软件缺陷的分类
  - 软件缺陷分类方法的应用
    - 按缺陷类型标准 (表现形式/自然属性) 的分类 (续)
      - 80 U-User Interface (用户接口)
        - 人机交互特性如屏幕格式、 用户输入确认、功能有效性、页面排版等方面的缺陷。
      - 90 P-Performance (性能)
        - 系统可测量的属性值得不到满足的缺陷，如：执行时间、事务处理速率等。
      - 100 N-Norms (规范)
        - 不符合各种标准的要求 (如编码规范、设计符号等) 的缺陷。

- 软件缺陷的分类
  - 软件缺陷分类方法的应用
    - 按缺陷严重程度的分类
      - 软件测试错误的严重程度

(1) Critical：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：严重地影响系统要求或基本功能的实现，且没有更正办法。

(3) Minor：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。

(4) Cosmetic：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：其它错误。

- 软件缺陷的分类
  - 软件缺陷分类方法的应用
    - 按缺陷严重程度的分类 (续)
      - 同行评审错误的严重程度

        (1) Major：主要的，较大的缺陷

        (2) Minor：次要的，小的缺陷

- 软件缺陷的分类
  - 软件缺陷分类方法的应用
    - 按缺陷优先级的分类

(1) Resolve Immediately

- 缺陷导致系统几乎不能使用或测试不能继续，需立即修复。

(2) High Queue

- 缺陷严重，影响测试，需要优先考虑修复。

(3) Normal Queue

- 缺陷需要正常排队等待修复或列入软件发布清单。

(4) Not Urgent

- 缺陷可以在方便时加以纠正。

- 软件缺陷的分类
  - 软件缺陷分类方法的应用
    - 按缺陷状态的分类
      - Submitted 已提交
        - 已提交的缺陷。
      - Open/Active 打开/活动
        - 该缺陷还没有解决；该缺陷被提交并得到确认，如新报告的缺陷。
      - Rejected/NotaBug 拒绝/不是缺陷
        - 该缺陷被提交并被拒绝，不需要修复或不是缺陷。
      - Resolved/Fixed 已修复
        - 该缺陷已被开发人员检查、修复过，通过单元测试，认为已解决但还没有被测试人员验证 。

- 软件缺陷的分类
  - 软件缺陷分类方法的应用
    - 按缺陷状态的分类 (续)
      - Closed/Inactive 关闭/非活动
        - 测试人员验证确认该缺陷被修复/不存在。
      - Reopen 重新打开
        - 测试人员验证发现该缺陷依然存在，等待开发人员进一步修复。
      - Deferred 推迟
        - 该缺陷可以在下一个版本中解决 。
      - Onhold 保留
        - 该缺陷由于技术原因或第三者软件的原因，开发人员不能修复。

- 软件缺陷的分类
  - 软件缺陷分类方法的应用
    - 按缺陷状态的分类 (续)
      - Cannotduplicate 不能重现
        - 开发人员不能复现该缺陷，需要测试人员检查缺陷复现的步骤。
      - Needmoreinfor 需要更多信息
        - 开发人员能重现该缺陷，但开发人员需要一些进一步的信息，如：缺陷的日志文件、图片等。
      - Duplicate 重复
        - 该缺陷已经被其他的软件测试人员发现。
      - Specmodified 需要修改软件规格说明书
        - 由于软件规格说明书对软件设计的要求，开发人员无法修复该缺陷，必须修改软件规格说明书。

- 软件缺陷的分类
  - 软件缺陷分类方法的应用
    - 按缺陷起源 (Origin) 的分类：缺陷起源是缺陷引起的故障或失效事件第一次被检测到的阶段。
      (1) Requirement：在需求阶段发现的缺陷
      (2) Architecture：在架构设计阶段发现的缺陷
      (3) Design：在设计阶段发现的缺陷
      (4) Code：在编码阶段发现的缺陷
      (5) Test：在测试阶段发现的缺陷
      (6) Use：在用户使用阶段发现的缺陷

- 软件缺陷的分类
  - 软件缺陷分类方法的应用
    - 按缺陷来源 (Source) 的分类：缺陷来源即缺陷的直接起因，指导致缺陷的错误所在的位置 (某阶段生成的文档、代码等)。
      (1) Requirement：需求描述缺陷
      (2) Architecture：架构缺陷
      (3) Design：设计缺陷
      (4) Data flow/Database：数据字典、数据库中的错误引起的缺陷
      (5) Code：编码缺陷
      (6) Test：测试缺陷
      (7) Integration：集成问题引起的缺陷

- 软件缺陷的分类
  - 软件缺陷分类方法的应用
    - 按缺陷根源 (Root Cause) 的分类：缺陷根源是造成软件错误的根本因素。
      - 测试策略
        - 错误的范围、误解的目标、超越能力的目标等。
      - 过程、工具和方法
        - 无效的需求收集过程、过时的风险管理过程、不适用的项目管理方法、缺少的估算规程、无效的变更控制过程等。
      - 人的因素
        - 项目团队职责交叉、缺乏培训、缺乏经验、缺乏士气、动机不纯等。

- 软件缺陷的分类
  - 软件缺陷分类方法的应用
    - 按缺陷根源的分类 (续)
      - 组织和通讯
        - 缺乏用户参与、职责不明确、管理失败等。
      - 其它
        - 硬件：例如处理器缺陷导致算术精度丢失、内存溢出等。
        - 软件：例如操作系统错误导致无法释放资源、工具软件错误、编译器错误等。
        - 环境：例如组织机构调整、预算改变、工作环境恶劣等。

- 软件缺陷的分类
  - 软件缺陷分类方法的应用
    - 按缺陷的生命周期的分类
      - New 新建
        - 每一个缺陷都是由测试人员发现并提交的，这个状态标注为 new。
      - Confirmed 已确认
        - 缺陷被提交后，由相应的负责人接受，即 confirmed。
      - Fixed 已解决
        - 相应的负责人员解决了该缺陷后，该缺陷的状态就改为 fixed，并且将其分发给测试人员进行回归测试，防止产生其他错误。

- 软件缺陷的分类
  - 软件缺陷分类方法的应用
    - 按缺陷的生命周期的分类 (续)
      - Closed 关闭
        - 测试人员对已解决的缺陷进行回归测试，如果确定已经解决，那么缺陷的状态就改为 closed，否则需要返还给该缺陷的负责人重新修正。
      - Reopen 重新打开
        - 有的缺陷在以前的版本中已经关闭，但是在新的版本中又重新出现，则需要将其状态改为 reopen。

- 软件缺陷的分类
  - 软件缺陷分类方法的应用
    - 按缺陷产生的可能性 (发生频率) 的分类
      - Always 总是
        - 总是产生这个软件缺陷，其产生的频率是 100%。
      - Often 经常
        - 按照测试用例，通常情况下会产生这个软件缺陷，其产生的频率大概是 80-90%。
      - Occasionally 有时
        - 按照测试用例，有时会产生这个软件缺陷，其产生的频率大概是 30-50%。
      - Rarely 很少
        - 按照测试用例，很少会产生这个软件缺陷，其产生的频率大概是 1-5%。

软件缺陷的分类

软件缺陷分类方法的应用

缺陷类型标准与软件测试阶段的关系

缺陷类型名称		功能	赋值	接口/时序
缺陷类型描述		功能实现，全局数据，算法	说明，重名，作用域，限制，初始化	过程调用和引用，函数调用，数据块共享，消息传递
缺陷检测阶段	设计评审	需求满足性、数据库设计	函数说明	过程接口
	单元测试	功能、程序逻辑	赋值	过程接口
	功能测试	程序错误	边界值	
	系统测试	程序逻辑、数据库、需求遗漏	边界值、无效数据域	
关联注入阶段		需求分析、设计	编码	设计

软件缺陷的分类

软件缺陷分类方法的应用

缺陷类型标准与软件测试阶段的关系

缺陷类型名称		联编打包	文档	用户接口/检查
缺陷类型描述		变更管理，配置库，版本控制	注释，需求分析及设计类文档	人机交互，屏幕控制，出错信息，日志，输入/输出
缺陷检测阶段	设计评审		设计说明	检查
	单元测试		设计问题	检查
	功能测试	打包问题	手册问题、需求问题、设计问题	报表格式、界面控制、权限错误、提示信息
	系统测试	安装	手册及联机帮助	界面控制
关联注入阶段		集成	所有阶段	设计、编码

软件缺陷的分类

软件缺陷分类方法的应用

缺陷类型标准与软件测试阶段的关系

缺陷类型名称		算法	标准/语法	性能
缺陷类型描述		算法，局部数据结构，逻辑，指针，循环，递归	程序设计规范，编码标准，指令格式等	不满足系统可测的属性值：执行时间、处理速率等
缺陷检测阶段	设计评审	算法	规范	
	单元测试	算法	语法、规范	
	功能测试			
	系统测试			性能
关联注入阶段		设计	设计、编码	设计

软件缺陷的分类

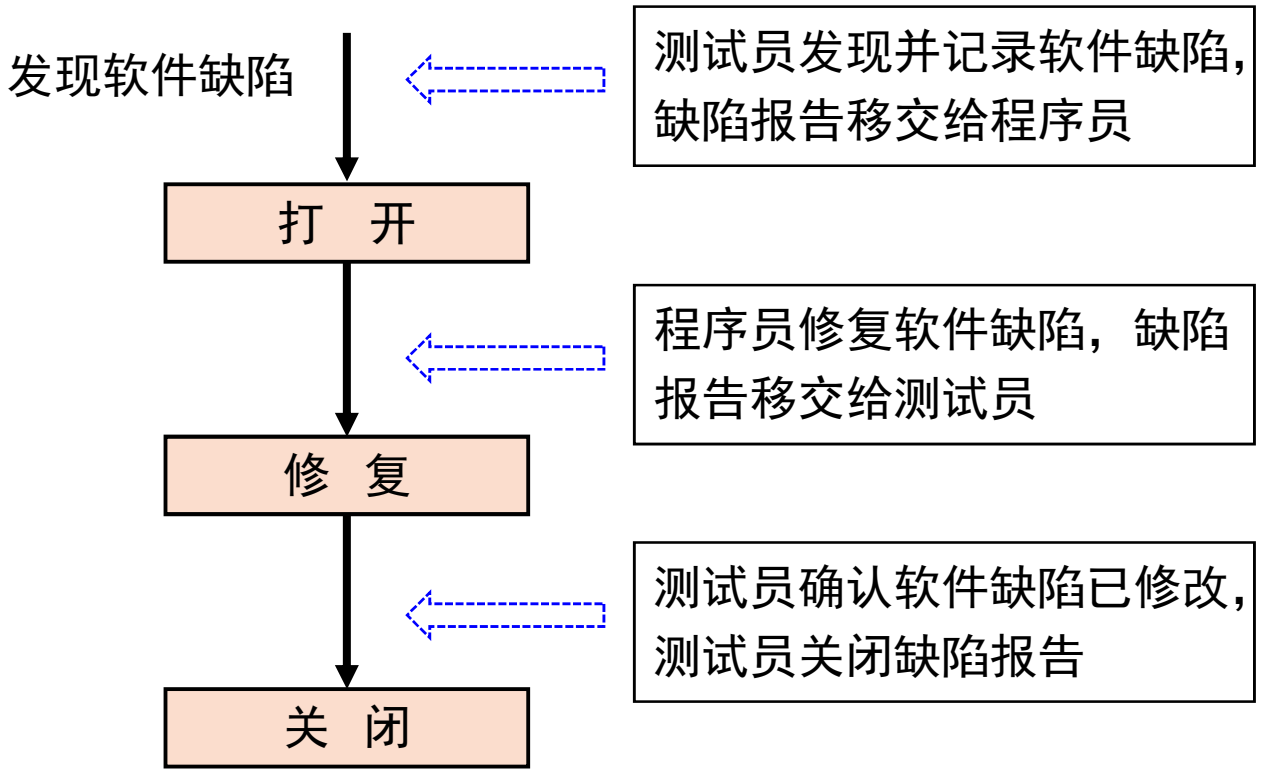
软件缺陷分类方法的应用

缺陷类型标准与软件测试阶段的关系

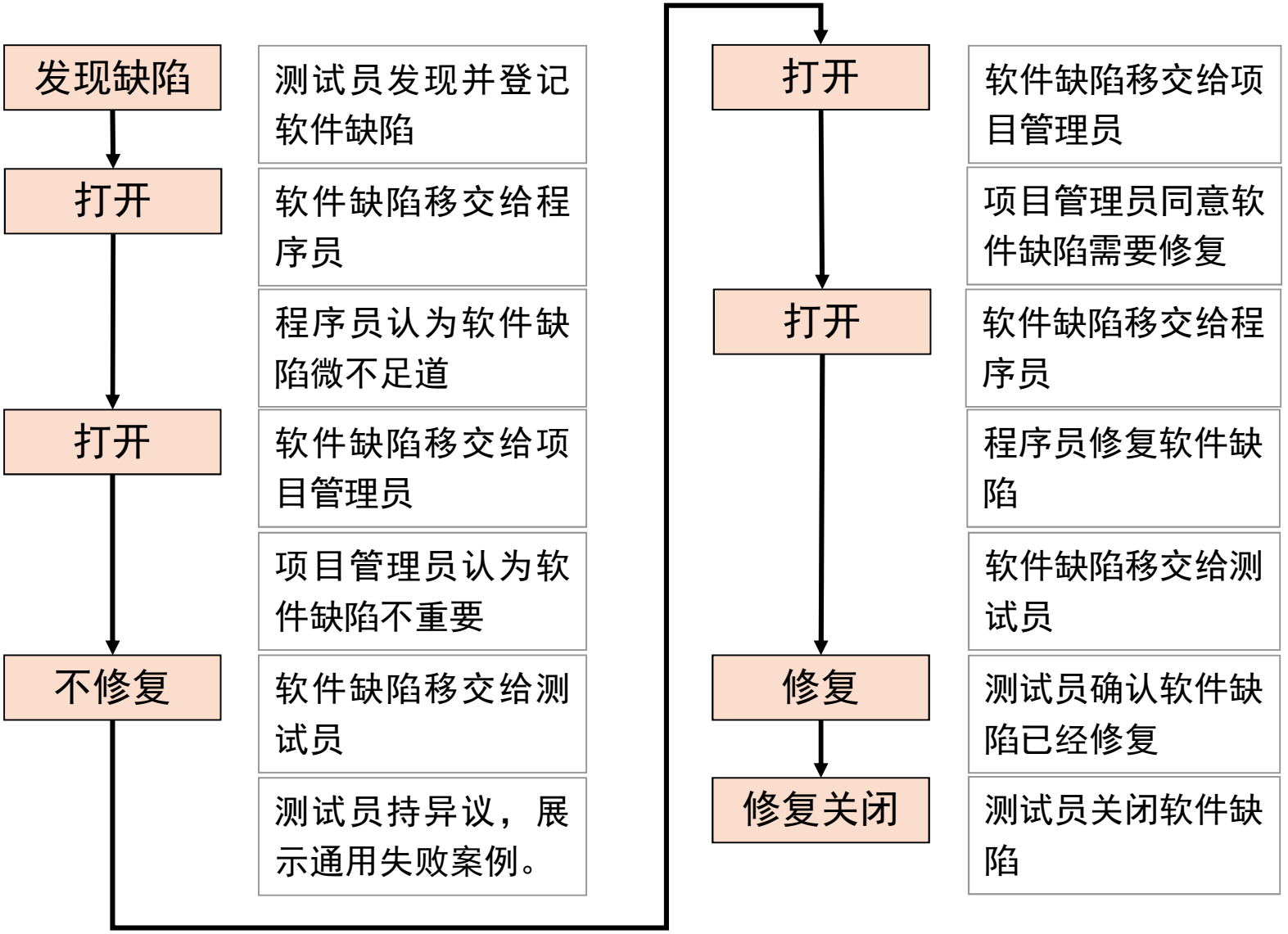
缺陷类型名称		环境		
缺陷类型描述		设计，编译，测试，以及其它支持系统问题		
缺陷检测阶段	设计评审			
	单元测试			
	功能测试	测试环境		
	系统测试	安装		
关联注入阶段		集成、运行		

软件缺陷的生命周期

一个简化的软件缺陷生命周期

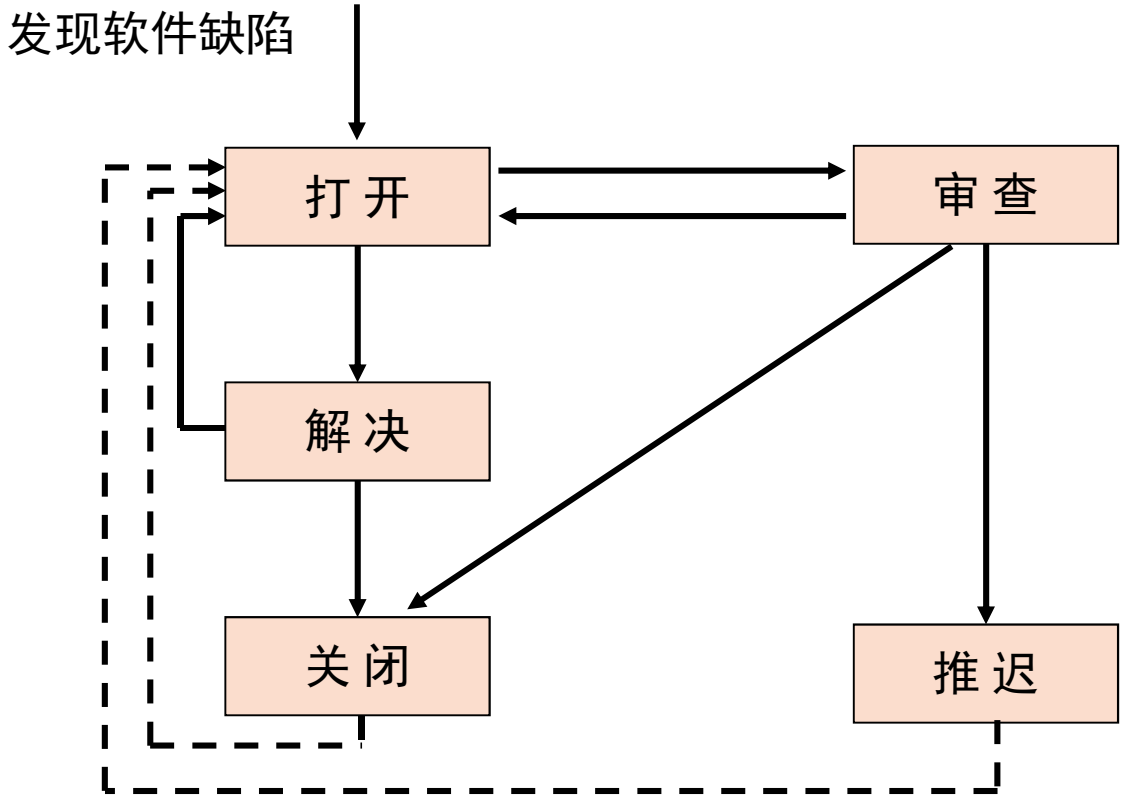


软件缺陷的处理与跟踪


发现缺陷
测试员发现并登记软件缺陷
打开
软件缺陷移交给程序员
程序员认为软件缺陷微不足道
打开
软件缺陷移交给项目管理员
项目管理员认为软件缺陷不重要
不修复
软件缺陷移交给测试员
测试员持异议，展示通用失败案例。
打开
软件缺陷移交给项目管理员
项目管理员同意软件缺陷需要修复
打开
软件缺陷移交给程序员
程序员修复软件缺陷
软件缺陷移交给测试员
修复
测试员确认软件缺陷已经修复
修复关闭
测试员关闭软件缺陷

软件缺陷的生命周期

- 一个复杂的软件缺陷生命周期

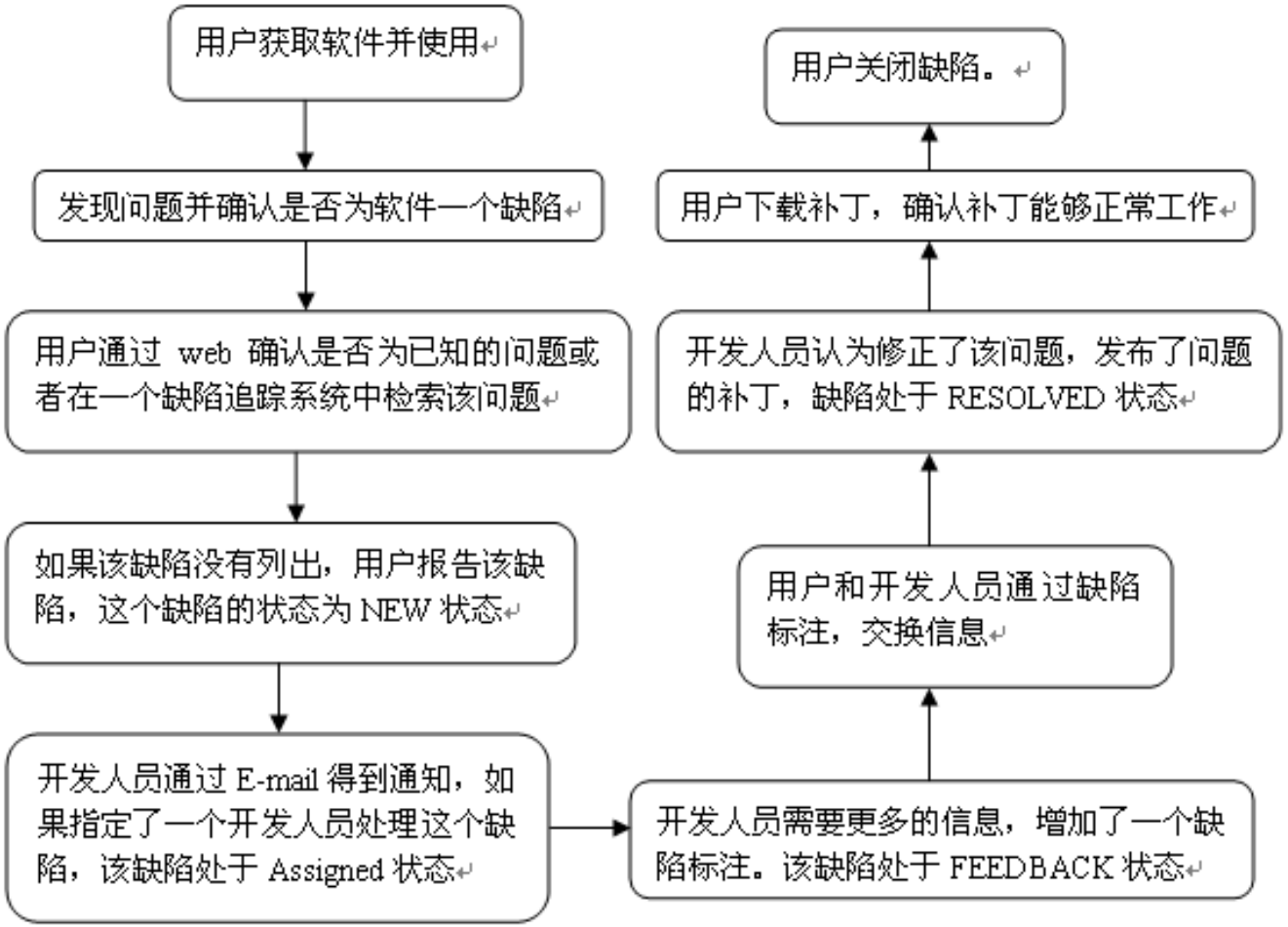


软件缺陷的生命周期

- Mantis 典型的缺陷周期
  - Mantis Bug Tracker, or MantisBT is a free popular web-based bug tracking system (feature list). It is written in the PHP scripting language and works with MySQL, MS SQL, and PostgreSQL databases and a webserver. MantisBT has been installed on Windows, Linux, Mac OS, OS/2, and others. Almost any web browser should be able to function as a client. It is released under the terms of the GNU General Public License (GPL).

软件缺陷的生命周期

- Mantis 典型的缺陷周期



软件缺陷的生命周期

小结

- 软件缺陷在生命周期中经历了数次的审阅和状态变化，最终由测试人员关闭软件缺陷，结束软件缺陷的生命周期。
- 软件缺陷生命周期中的不同阶段是测试人员、开发人员和管理人员一起参与、协同测试的过程。软件缺陷一旦发现，便进入测试人员、开发人员、管理人员的严密监控之中，直至软件缺陷生命周期终结，这样既可保证在较短的时间内高效率地关闭所有的缺陷，缩短软件测试的进程，提高软件质量，同时降低软件的开发、测试和维护成本。

- 软件缺陷的处理阶段
  - 报告与审阅
    - 测试人员在缺陷跟踪数据库中输入一个新的缺陷，并及时提交缺陷报告。缺陷报告的审阅可以由测试管理员、项目管理员或其他人员进行，主要检查缺陷报告的质量水平。
  - 拒绝
    - 审阅者决定需要对一份缺陷报告进行重大修改时，例如需要添加更多的信息或者需要改变缺陷的严重等级，则拒绝该报告，并和报告提交者一起讨论，纠正缺陷报告后再次提交。
  - 完善
    - 审阅者对测试人员已经完整地描述了问题的特征并将其分离的缺陷报告加以肯定。

- 软件缺陷的处理阶段
  - 分配
    - 测试人员将特征得到完整描述并被分离的问题分配给适当的开发人员；不能确定具体开发人员时应分配给项目开发组长，由开发组长再分配给对应的开发人员。
  - 测试
    - 一个缺陷一旦被开发人员修复，它就将进入测试阶段。缺陷的修复需要得到测试人员的验证，同时还要进行回归测试，检查这个缺陷的修复是否会引入新的问题。
  - 重新打开
    - 如果修复没有通过验证测试，测试人员将重新打开这个缺陷报告。重新打开一个缺陷，需要加注释说明，否则会引起“打开-修复”多个来回，造成测试人员和开发人员不必要的矛盾。

软件缺陷的处理与跟踪

- 软件缺陷的处理阶段
  - 关闭
    - 如果修复通过验证测试，测试人员将关闭这个缺陷。只有测试人员具有关闭缺陷的权限。
  - 暂缓
    - 如果每个人都同意将确实存在的缺陷延迟处理，应该指定下一个版本号或修改的日期。一旦新的版本开始时，这些暂缓的缺陷应该重新被打开。

软件缺陷的跟踪

- 软件缺陷跟踪系统
  - 软件缺陷跟踪系统包括了软件缺陷跟踪数据库，它不仅有利于对软件缺陷进行清楚描述，还提供统一的、标准化的报告，使所有人的理解一致。
  - 缺陷跟踪数据库允许自动建立连续的软件缺陷编号，还提供了大量供分析和统计的选项，这是手工方法难以实现的。
  - 基于缺陷跟踪数据库，可以快速生成满足各种查询条件所必要的缺陷报表、曲线图等，开发小组和主管可以随时掌握软件产品质量的整体状况或测试/开发的进度。
  - 缺陷跟踪数据库提供了软件缺陷属性，并允许开发小组根据项目的相对和绝对重要性来修复缺陷。

软件缺陷的跟踪

软件缺陷跟踪系统 (续)

- 在软件缺陷跟踪数据库中记录每一份关闭的缺陷报告。产品交付时，每一份未关闭的缺陷报告都提供了预先警告的有效技术支持，并且证明测试人员找到了在特殊领域突发事件中的软件缺陷。
- 缺陷跟踪系统对缺陷的管理贯穿从最初报告到最后解决的软件缺陷生命周期，确保了每一个缺陷不会被忽略。同时，它还可以使注意力保持在那些必须尽快修复的重要缺陷上。
- 当缺陷在它的生命周期中变化时，开发人员、测试人员以及管理人员能够熟悉新的软件缺陷信息。一个设计良好的缺陷跟踪系统可以获取历史记录，并在检查缺陷状态时提供历史记录。

软件缺陷的处理与跟踪

- 软件缺陷的跟踪
  - 一些软件缺陷跟踪系统
    - TestTrack Pro

      http://www.seapine.com
    - DevTrack

      http://www.techexcel.com
    - JIRA

      http://www.atlassian.com
    - Mantis

      http://www.mantisbt.org
    - Bugzilla

      http://www.bugzilla.org

软件缺陷的处理与跟踪

软件缺陷的跟踪

- 一些开源软件缺陷跟踪系统
  - Mantis
    http://mantisbt.sourceforge.net/
  - Bugzilla
    http://www.mozilla.org/projects/bugzilla/
  - Bugzero
    http://bugzero.findmysoft.com/
  - Scarab
    http://scarab.tigris.org/
  - TrackIT
    http://trackit.sourceforge.net/
  - Itracker
    http://www.itracker.org/

软件缺陷的处理与跟踪

软件缺陷的跟踪

- 一些商业软件缺陷跟踪系统
  - JIRA

    http://www.atlassian.com
  - IBM ClearQuest

    http://www-01.ibm.com/software/awdtools/clearquest
  - Compuware TrackRecord

    http://www.compuware.com/trackrecord.htm
  - HP TestDirector

    http://www.hp.com/
  - TestTrack Pro

    http://www.seapine.com/ttpro.html
  - DevTrack

    www.techexcel.com/products/devsuite/devtrack.html
  - Borland Segue SilkCentral™ Issue Manager

Lecture 24. Software Defect Management (1)

End of Lecture