IBM Software Group

使用Rational ClearQuest实现软件缺陷正交分析

Rational. software

IBM 软件部

ON DEMAND BUSINESS

议程

- 什么是缺陷正交分析
- 使用IBM Rational ClearQuest 实现缺陷正交分析
- 缺陷正交分析实施步骤

传统测试分类方法的缺点

- 无法全面了解缺陷产生的原因
  - 简单的使用严重等级、重要性等分类方法无法帮助开发人员快速定位问题
- 无法准确评估测试人员的效率
- 无法为每轮迭代或每个阶段定义精确有效的退出条件
  - 无法了解每个缺陷所属类型，只能根据总体测试通过率定义退出条件
- 无法有效反映缺陷对最终用户的影响及用户的感受

缺陷正交分析方法的历史

- ODC - Orthogonal Defect Classification
- IBM T.J.Watson Research Center的Ram Chillarege博士等人于1990年提出ODC概念，1997年基本完成ODC理论体系建设
- 1998年以后，IBM公司开始在其全球24个研发分支机构推广应用这项技术，取得了很好的经济效益
  - ODC技术的成本只有根原因分析法（Root Cause Analysis, RCA）的十分之一
  - 产生的控制措施可按优先级排序，因而可以大量节约质量管理成本和有效指导改进开发和测试过程

正交缺陷分类技术 (ODC)

- ODC 结合了根原因分析和统计建模 (Statistical Modeling) 两种软件缺陷分析技术的优势。
- ODC 提供了一套用于捕获缺陷数据关键特性的方案，并给出对分类的缺陷数据集进行分析的指导。
- ODC 可以帮助我们全面了解缺陷，从而采取最有效的措施来改进软件开发过程中的不足，不断地提高软件产品质量。

什么是正交缺陷分类

- ODC将每一个缺陷按不同维度分类
  - 当缺陷数量较多时，也可以对缺陷进行抽样分析
- 根据大量缺陷分类后产生的各类缺陷的统计数字，结合缺陷定位信息（所属子系统、模块、特性）进行多维度正交分析
  - 准确确定产品主要质量问题区域
  - 识别缺陷引入和去除过程的重点改进对象
  - 实现对过程和产品的精确改进指导
- 将传统度量手段和ODC技术相结合，可以实现对过程和产品的宏观评估和微观解剖。

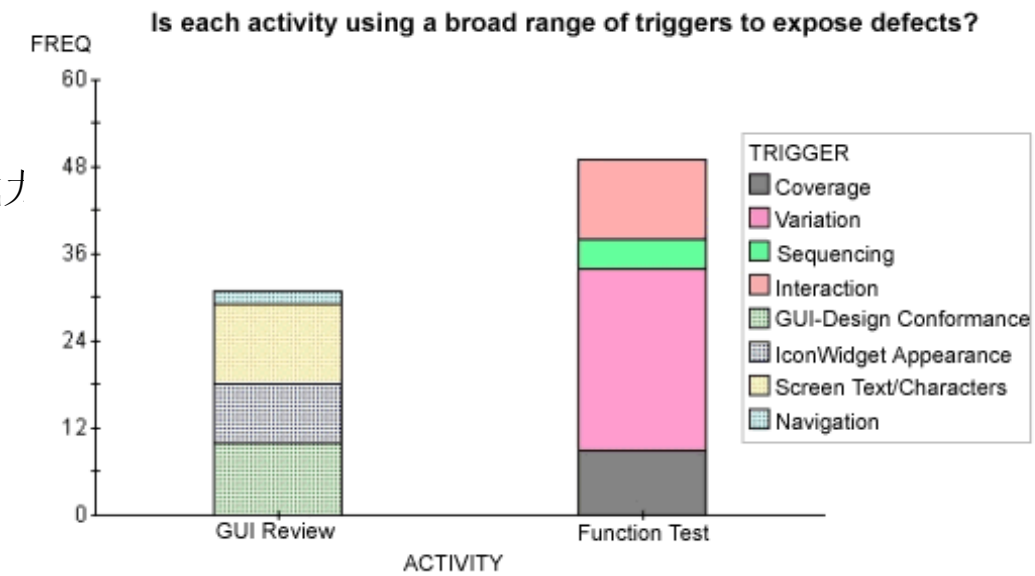
如何对软件缺陷进行正交分类

- 当发现缺陷时（测试人员）
  - Activity (活动)：它是指在哪种测试中发现的defect, 例如：UT、FVT、SVT
  - Trigger (触发器)：是什么样的测试发现了这个defect
  - Impact (影响)：对于客户有哪些方面的影响

- 当缺陷被修复时（开发人员）
  - Target (目标)：将要在哪里改正错误，例如：design、code 等等
  - Defect Type (缺陷类型)：defect的类型，例如：算法、初始化 等等
  - Qualifier (限定词)：定义了引起缺陷的原因
  - Source (源)：内部代码、outsource的代码等等
    - 内部代码、重用库代码、移植代码或外包代码
  - Age ： defect是新代码还是重写的代码
    - 新、旧（基类）、重写或再次修复缺陷

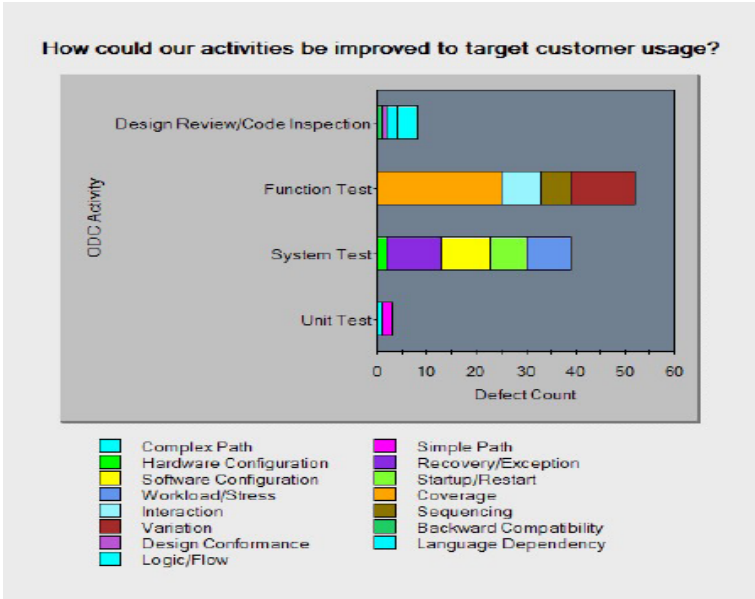
功能测试中应用ODC

- 在功能测试（FVT） 中，Trigger是帮助功能测试做得更好的主要指标
- 在ODC中，Trigger可以简单的理解为“是什么样的测试发现了这个缺陷”
- FVT中使用了如下4个触发器：
  - Coverage：任何用户都会用到的功能，基本的、简单的功能
  - Variation：使用不常用的创造性方法或者输入发现的错误
  - Sequencing：用与以前不同的操作流程来完成一种任务或功能时发现的错误
  - Interaction：当两个或者多个功能模块互相交互时发现的错误
- 通过对测试结果进行分析，可以
  - 了解产品的基本质量水平
  - 评估测试的有效性
  - 改进测试计划，调整不同类型测试的测试力
  - ……



系统测试中应用ODC

- 系统测试（SVT）使用了另外一组Trigger：
  - Blocked：哪些缺陷阻止了SVT的执行
  - Stress：压力测试
  - Recovery：数据恢复和出错处理
  - Restart：系统的启动和重启
  - Hardware Configuration：硬件配置
  - Software Configuration：软件配置
- 测试结果可以帮助项目组
  - 确定系统测试执行的时机
  - 改进测试计划
  - ……



开发中应用ODC

- Defect Type帮助开发人员定位开发过程中需要关注的环节
  - Assignment/Initialization：没有正确的初始化
  - Checking：条件语句中，参数没有或者被错误的校验
  - Algorithm/Method：算法问题
  - Function/Class/Object：功能性的错误，可能是模块内的功能没有被正确实现，也可能是模块与模块之间相联系的部分没有被正确实现
  - Timing/Serialization：与时间及共享资源的顺序处理有关的错误
  - Interface/Object-Oriented Messages：界面相关的错误
  - Relationship：代码之间相关联的错误，例如错误的继承关系
- 缺陷信息帮助开发人员了解
  - 开发过程哪个环节错误比较多
  - 从何/如何改正错误
  - ……

正交缺陷分类的好处

- 项目经理
  - ✓评估产品的稳定性
  - ✓评估测试的有效性
  - ✓风险评估
  - ✓管理测试过程
  - ✓设置测试阶段进入/退出的条件
- 开发团队
  - ✓在产品和组件级别上鉴别“热点”
  - ✓量化不正确的和丢失的代码
  - ✓更有效地单元测试
  - ✓减少逃逸缺陷数量
  - ✓更好地支持原因分析/趋势分析

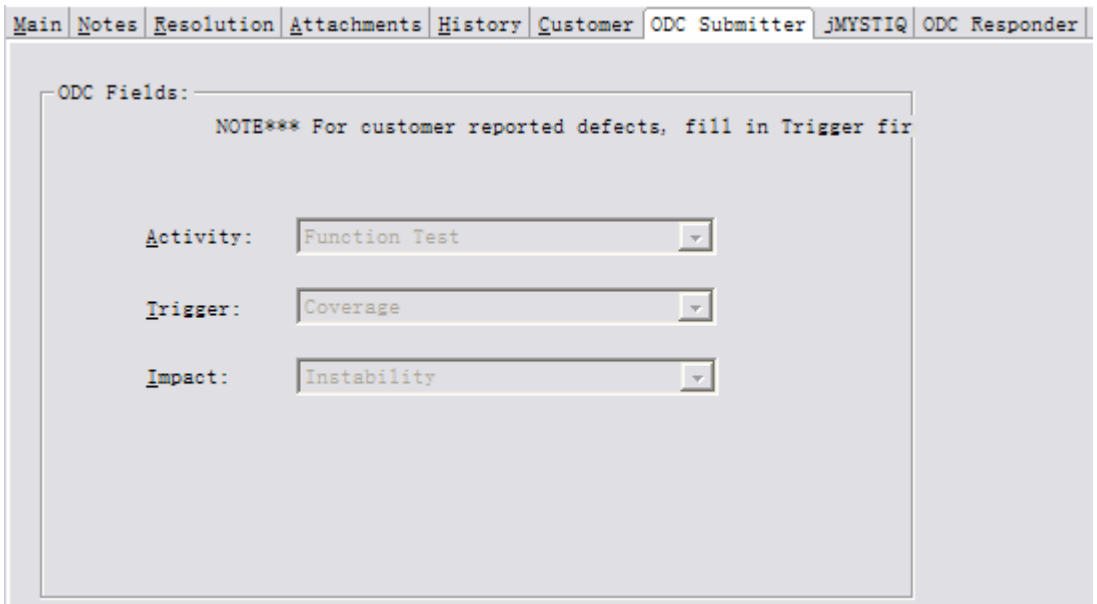
- 测试团队
  - ✓更有效的测试
  - ✓对测试的进度有更好的评估
- 服务人员
  - ✓鉴别需要开发人员帮助的“热点”
  - ✓推荐需要额外测试的地方
  - ✓计划资源和技术需求

议程

- 什么是缺陷正交分析
- 使用IBM Rational ClearQuest 实现缺陷正交分析
- 缺陷正交分析实施步骤

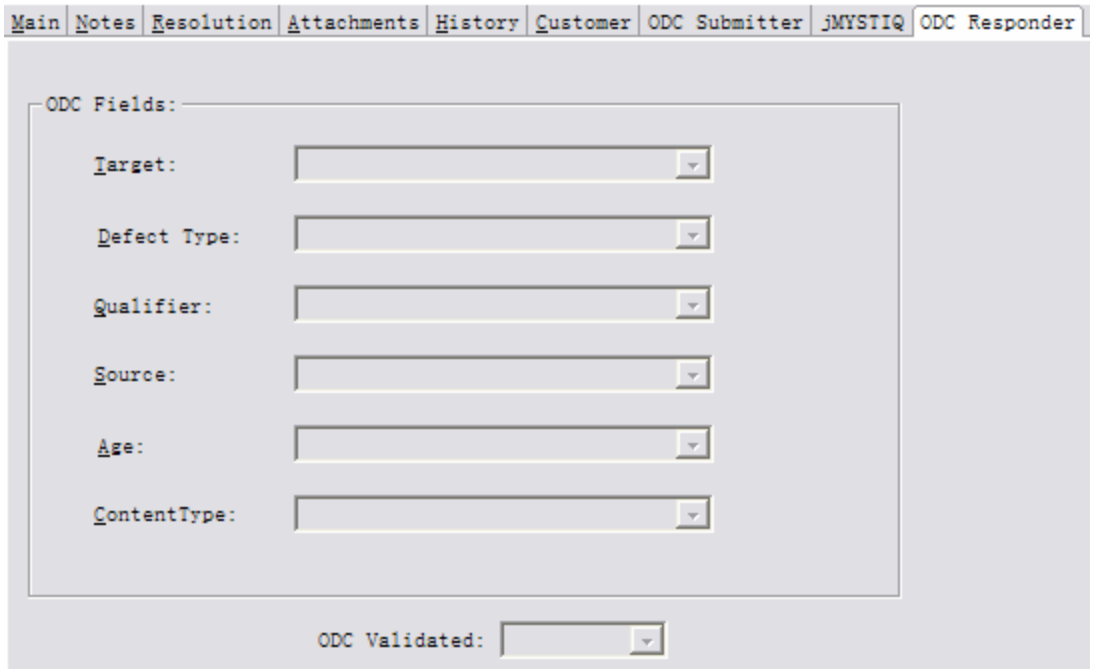
CQ Schema中为测试人员增加的属性

- Activity : 发现该缺陷所采用的检验活动
- Trigger : 曝露缺陷时采取存在的环境或条件
- Impact : 该缺陷可能在哪一方面对客户造成影响



CQ Schema中为开发人员增加的属性 (1)

- Target: 被修复目标对象的高层特性 (需求、设计、编码等)
- Defect Type: 所进行的缺陷修复的种类 (算法、初始化等)
- Qualifier: 缺陷是由于丢失、错误、无关的信息或代码引起的

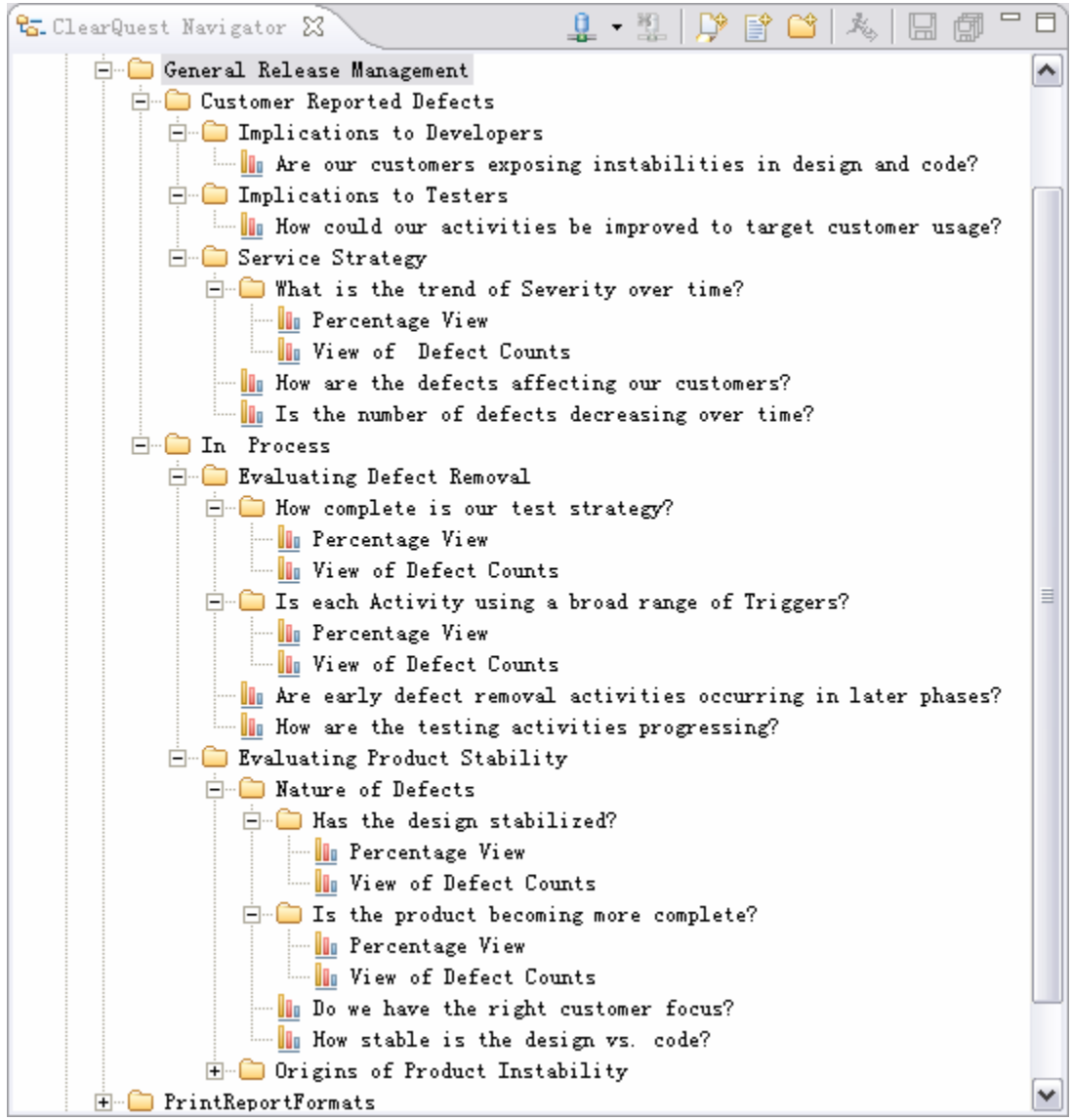


CQ Schema中为开发人员增加的属性 (2)

- Source: 缺陷的来源 (新开发代码、重用代码、移植或外包代码)
- Age: 缺陷历史 (新、旧基类、重写或再次修复缺陷)
- Content Type: Target为用户文档的缺陷独有的错误内容类型
- ODC Validated: 校验状态标志

通过图表分析数据

- ODC 通过各种统计图表来分析缺陷数据



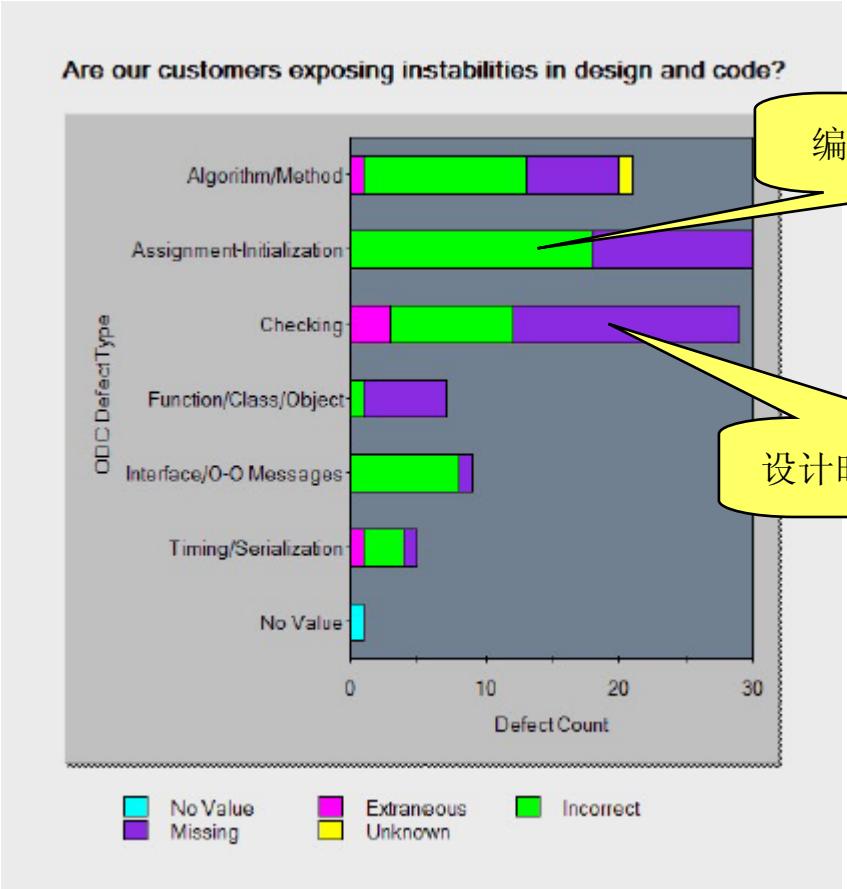
分析用户提交的缺陷

- 对于开发人员的反馈
- 对于测试人员的反馈
- 指导软件产品的支持策略
  - 产品支持人员的工作强度
  - 如何调整支持力度以保证客户满意度

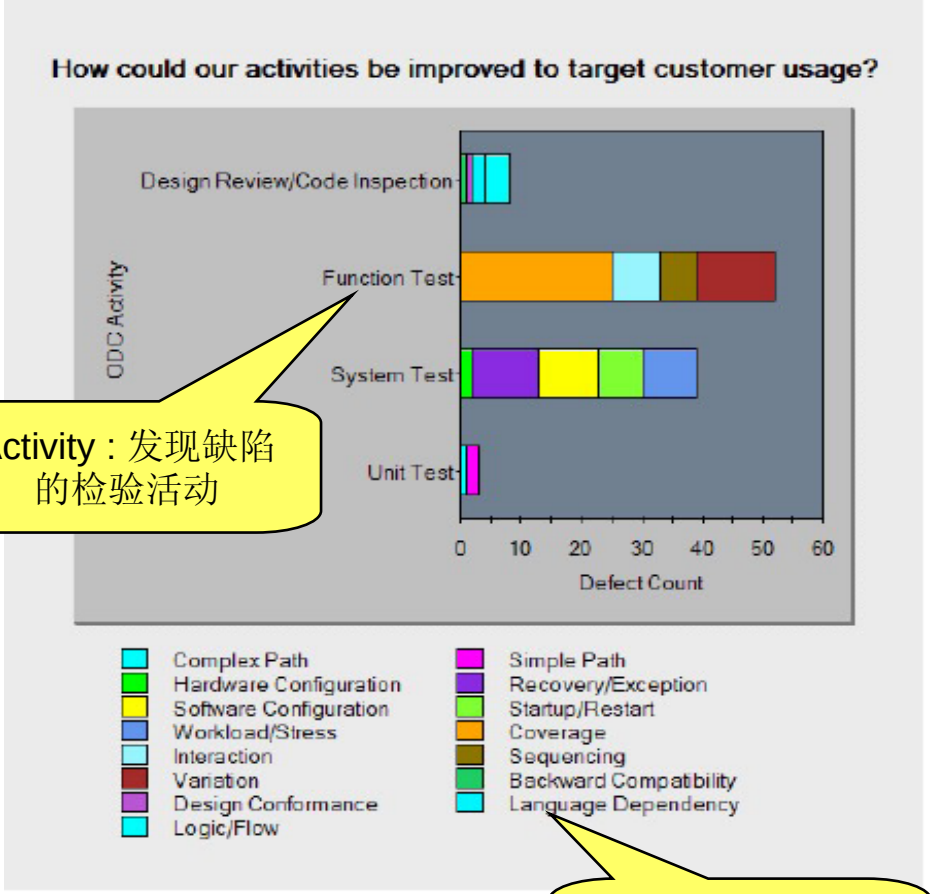
对于开发人员的反馈

Defect Type
Algorithm/Method
Assignment/Initialization
Checking
Function/Class/Object
Interface/O-O Messages
Timing/Serialization

Qualifier
Missing
Incorrect
Extraneous



对于测试人员的反馈

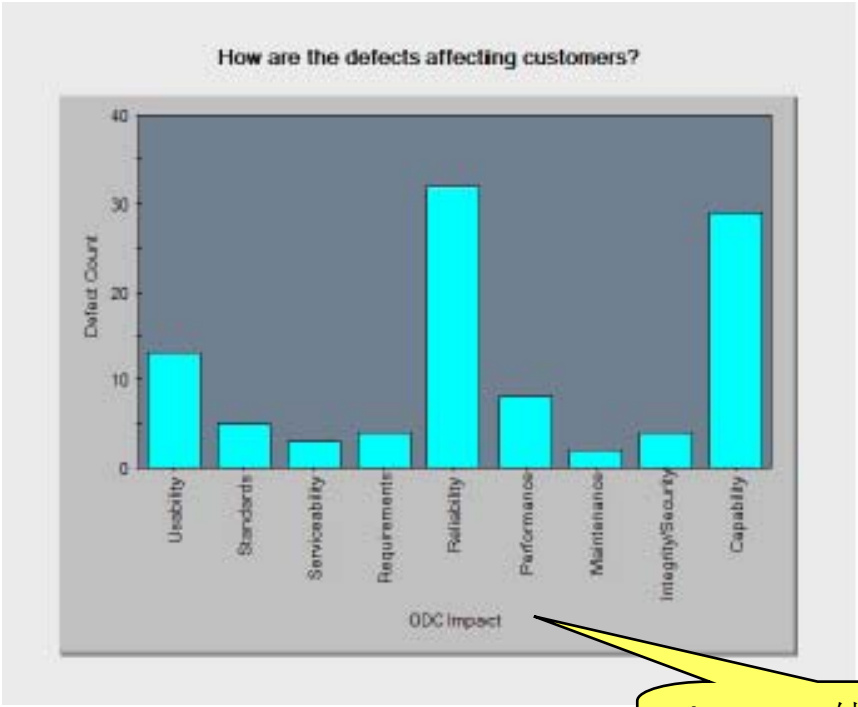


Activity：发现缺陷的检验活动

Trigger：发现缺陷的环境和条件

Activity	Trigger
Design Review/Code Inspection	• Design Conformance • Logic/Flow • Backward Compatibility • Lateral Compatibility • Concurrency • Language Dependency • Side Effects • Rare Situations
Unit Test (White Box)	• Simple Path • Complex Path
Function Test (Black Box)	• Coverage • Variation • Sequencing • Interaction
System Test	• Workload/Stress • Recovery/Exception • Startup/Restart • Hardware Configuration • Software Configuration • Blocked Test
User Documentation Review (Online or Hardcopy)	• Accuracy • Clarity • Completeness • Consistency • Organization • Retrievability • Style • Task Orientation • Graphic Design/Aesthetics
GUI Review (code implications)	• GUI Design Conformance • Icon/Widget Appearance • Screen Text/Characters • Input Devices • Navigation • Widget/GUI Behavior

软件缺陷可能在哪些方面影响客户满意度



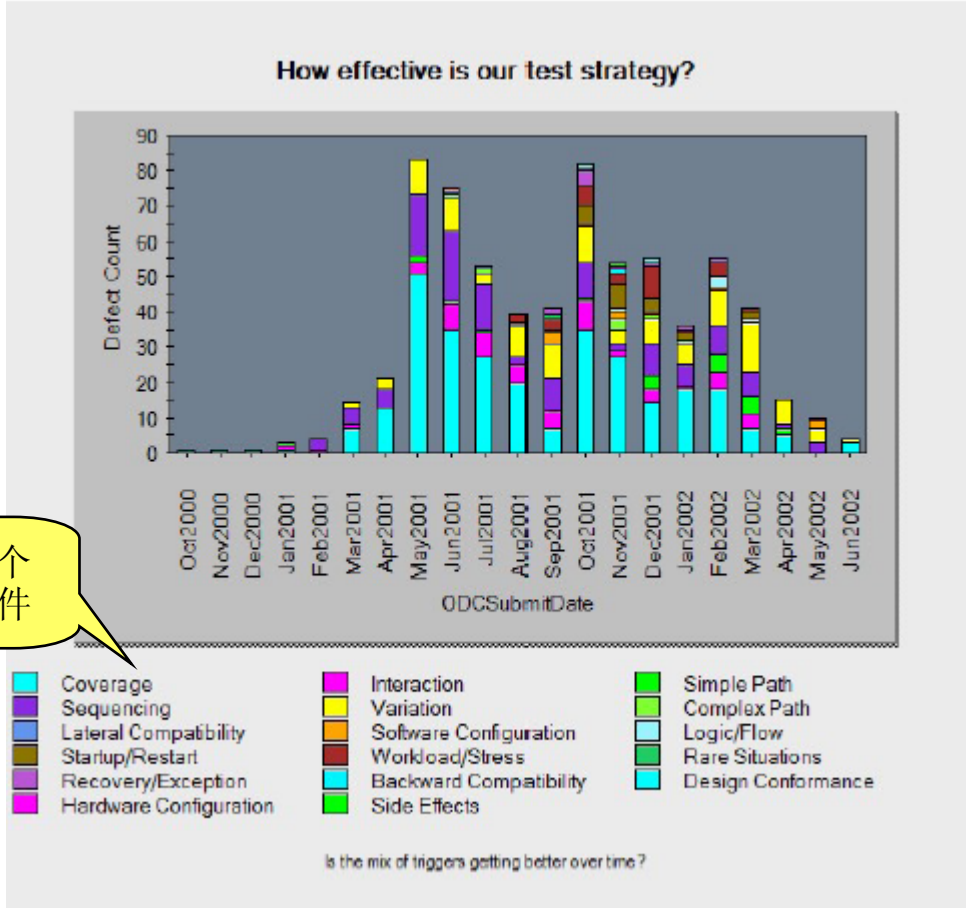
Impact
Installability
Serviceability
Standards
Integrity/Security
Migration
Reliability
Performance
Documentation
New Requirement
Maintenance
Usability
Accessibility
Capability

Impact : 缺陷对客户可能造成的影响

分析开发团队内部发现的缺陷分类

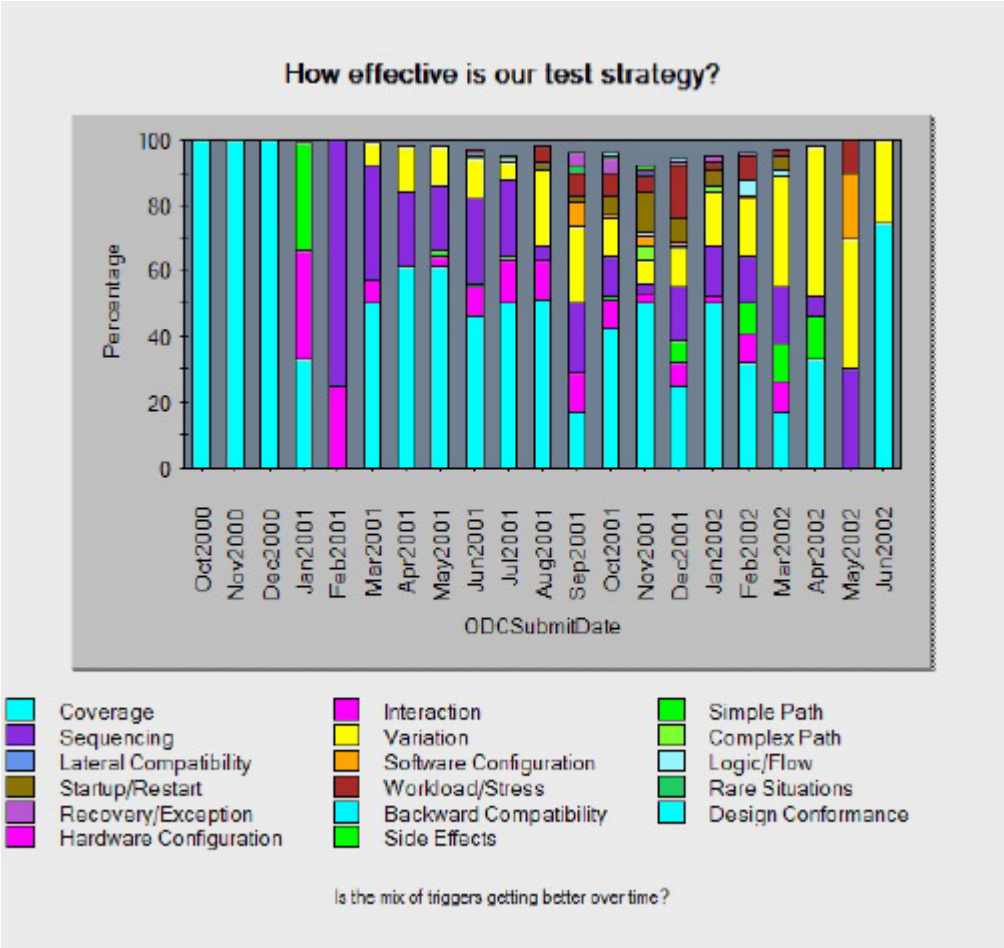
- 评估测试工作的有效性
- 评估产品的稳定性
- 鉴别设计和代码中的强处和不足
- 评估客户的使用方法
- 度量产品开发的进度

评估测试工作的效率

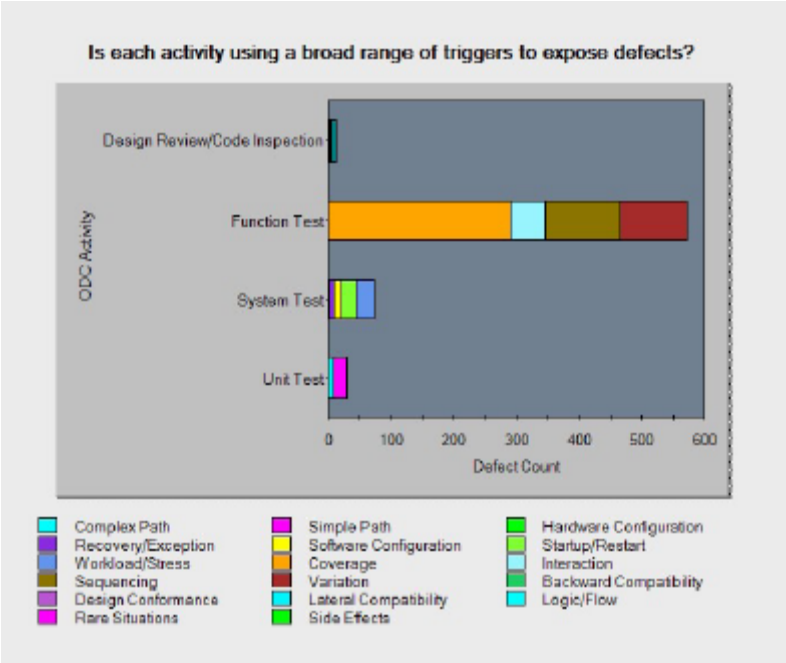


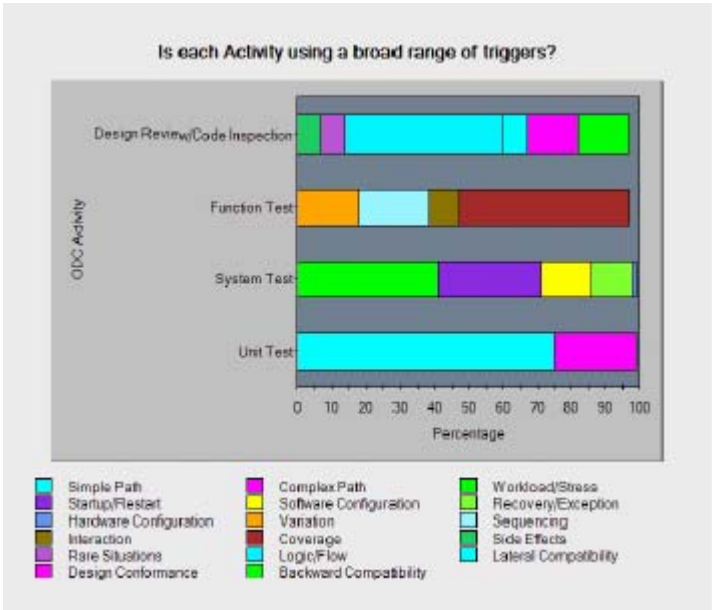
Trigger : 发现这个缺陷的环境和条件

缺陷类型随时间进展而发生的变化



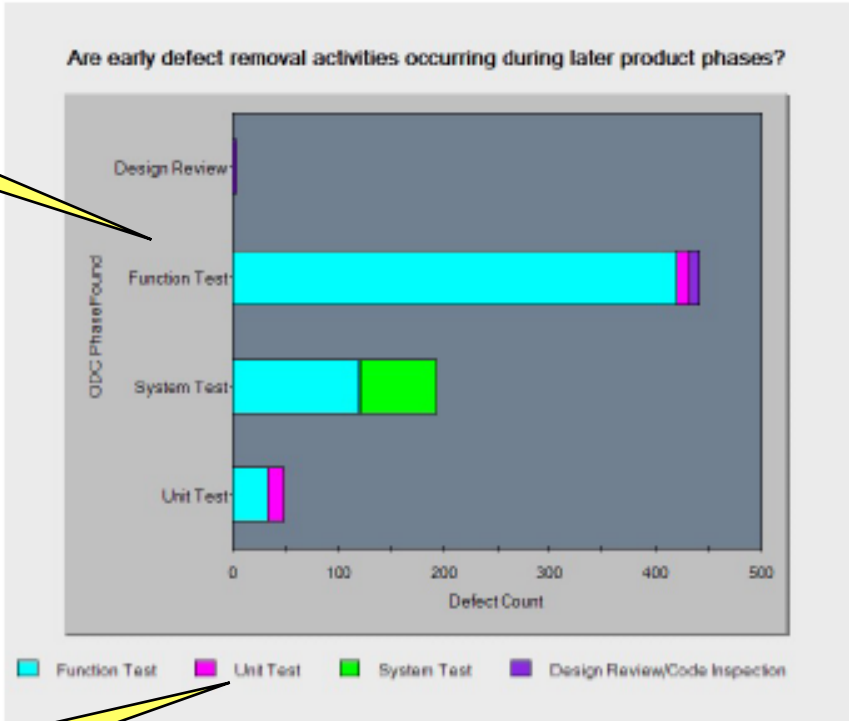
发现的缺陷是否覆盖了所有的 Trigger 类型





发现缺陷的检验活动分布在较晚的阶段中

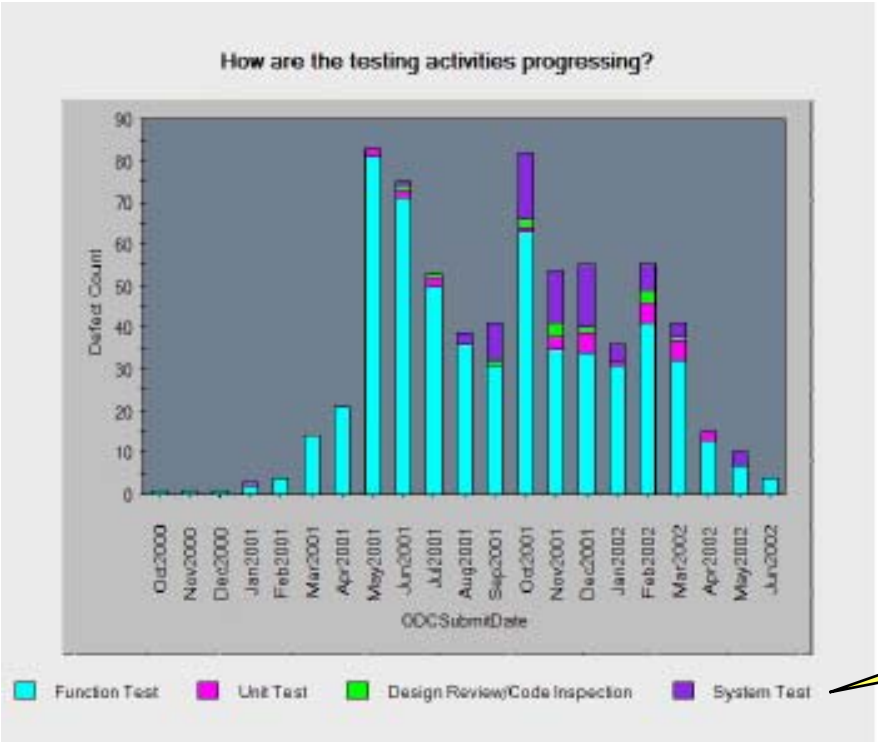
Phase : 开发的时间阶段



Activity : 发现这个缺陷的检验活动

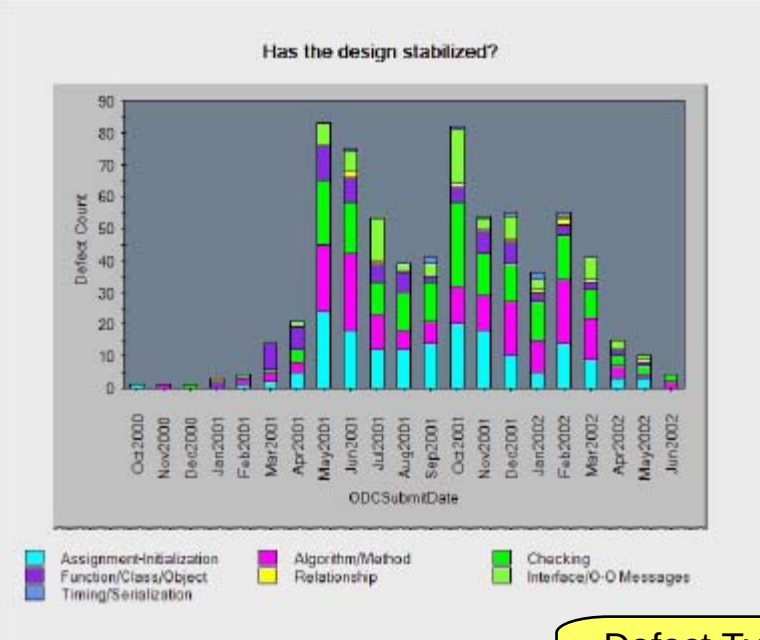
Activity
Design Review
Code Inspection
Unit Test
Function Test
System Test
GUI Review
Internal Documentation Review
Information Development Review

缺陷是通过哪些检验活动发现的



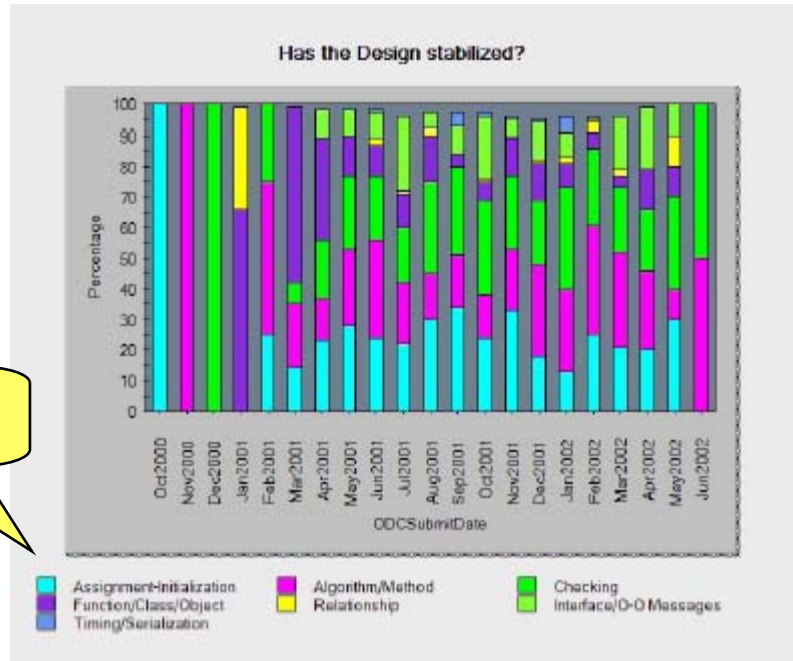
Activity：发现这个缺陷的检验活动

我们的设计稳定了吗？



Function 类型的缺陷应该随着时间进展而逐渐减少，这方面的缺陷逐步被改正；

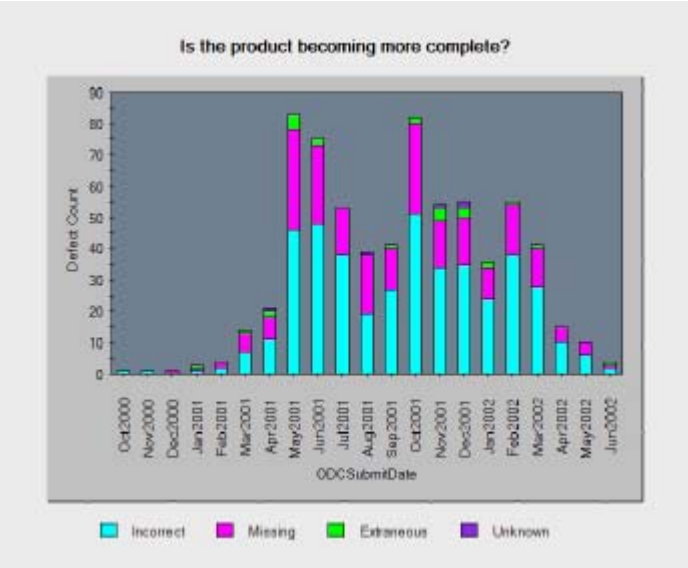
Assignment 和 Checking 类型的缺陷也是如此，它们是编码方面的问题，应该在早期测试中被发现。



Defect Type :
缺陷的类型

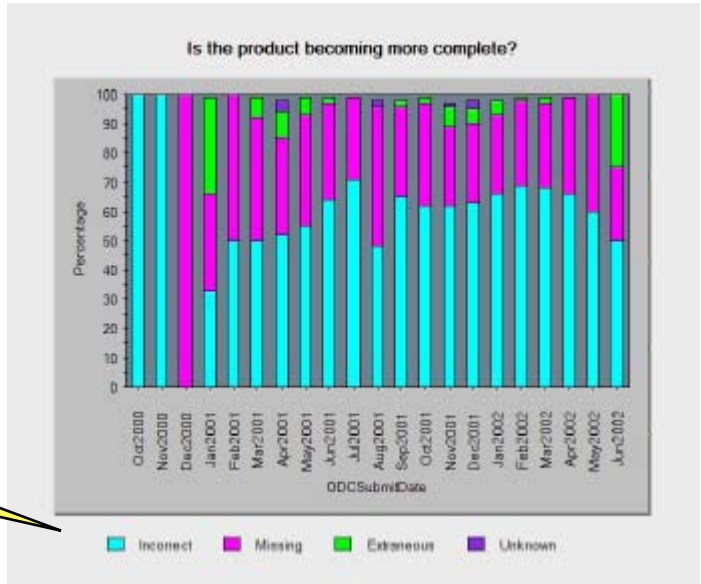
相反，Timing/Serialization 和 Interface/O-O Messages 类型的缺陷应该呈上升趋势，因为它们是被比较复杂的(系统)测试发现的。

产品是否趋向于完成



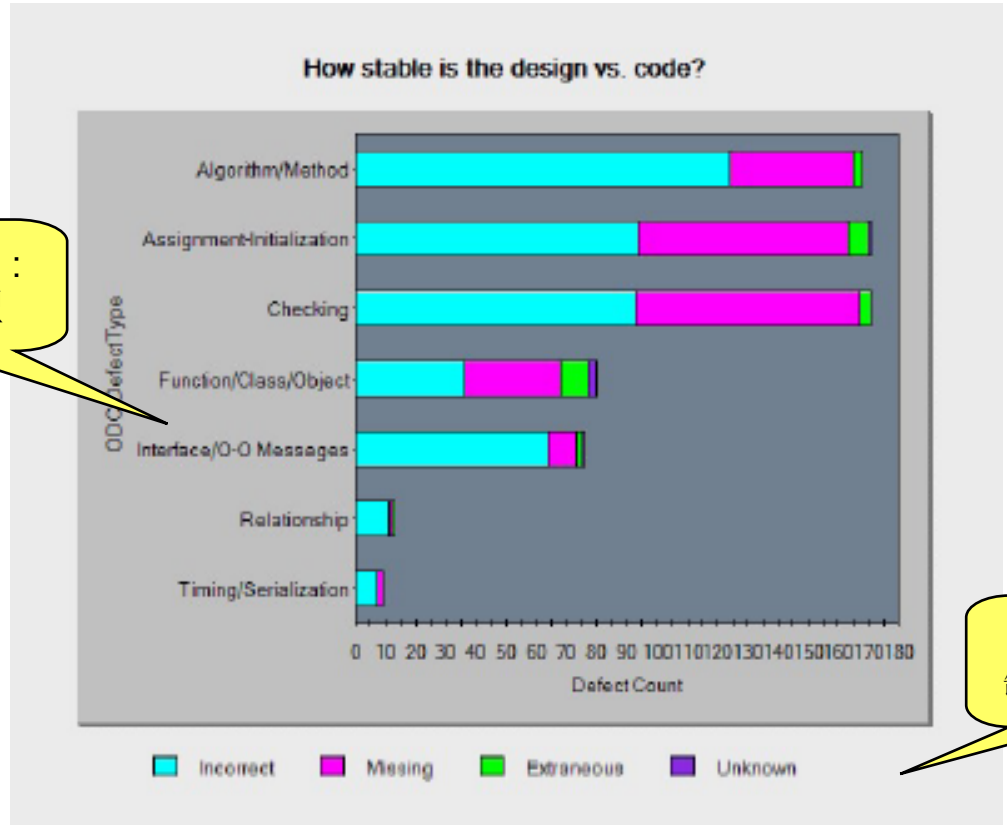
Missing 类型的缺陷应该随着时间进展而逐渐减少，这方面的缺陷逐步被改正；

如果该类型的呈增长趋势的话，需要针对 Defect Type 做进一步的分析



Qualifier :
缺陷的限定

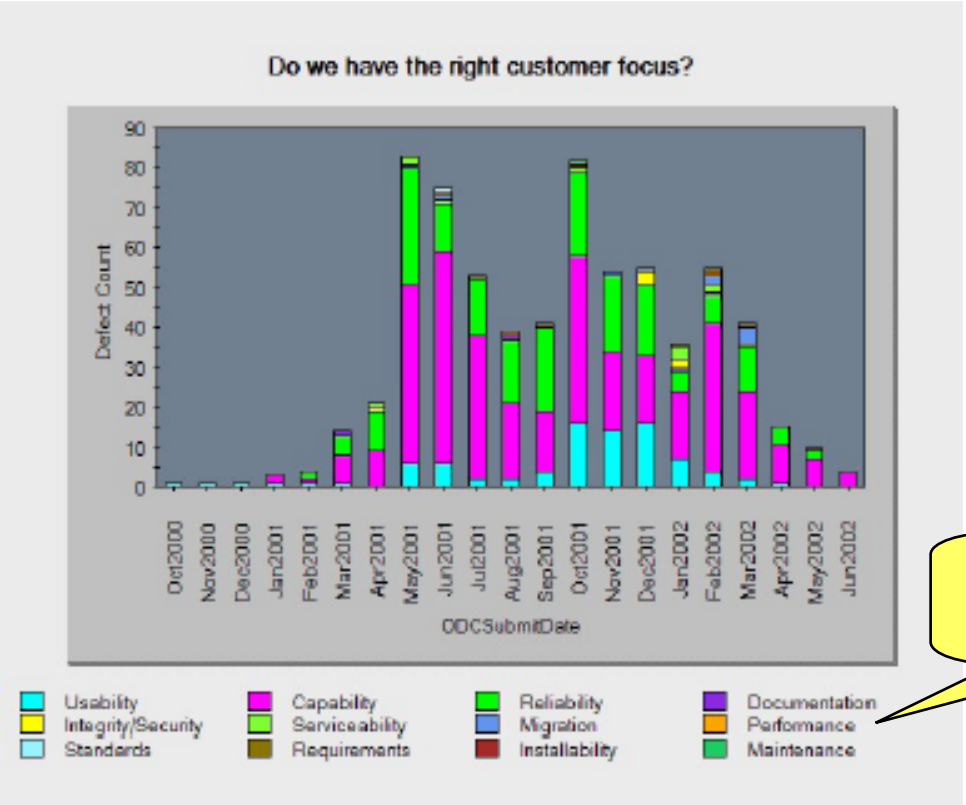
设计和代码的稳定性



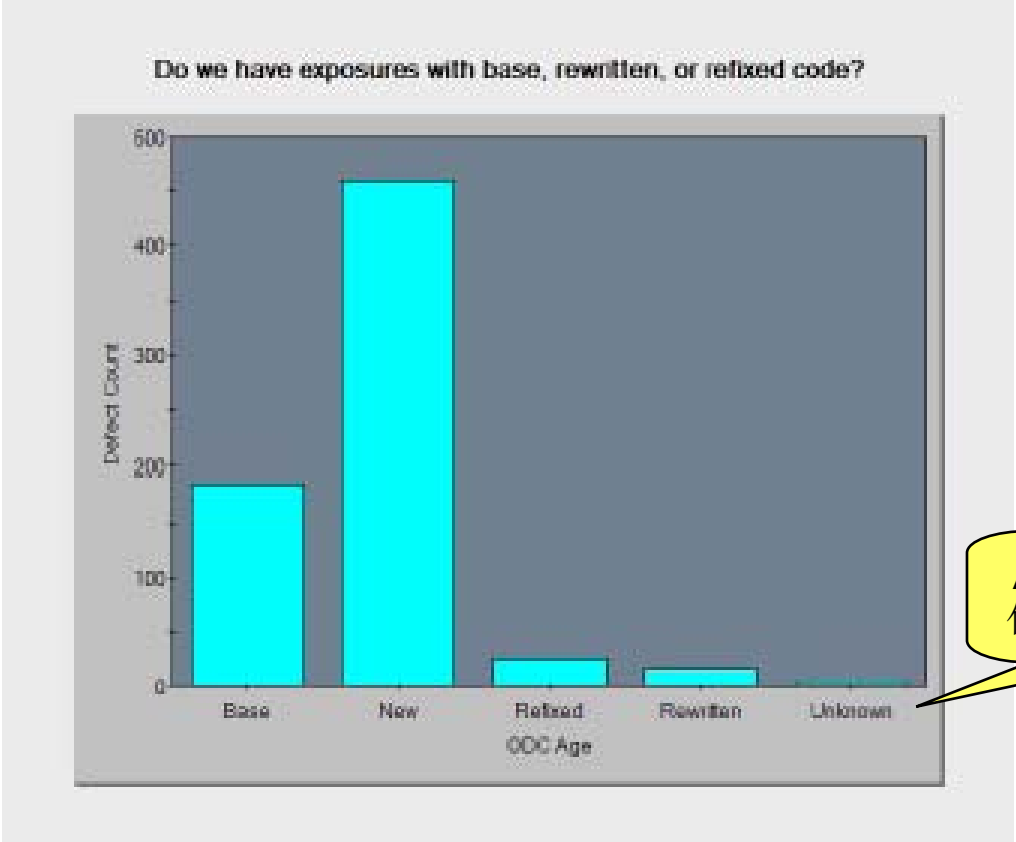
“Missing”、 ”Function” & ”Algorithm” & “Checking”类型的缺陷过多的话，很可能表明详细设计工作做得不够

“Incorrect”、“Checking” & “Assignment”类型的缺陷可能意味着编码时引入了很多缺陷

我们是否真正关注了客户需求



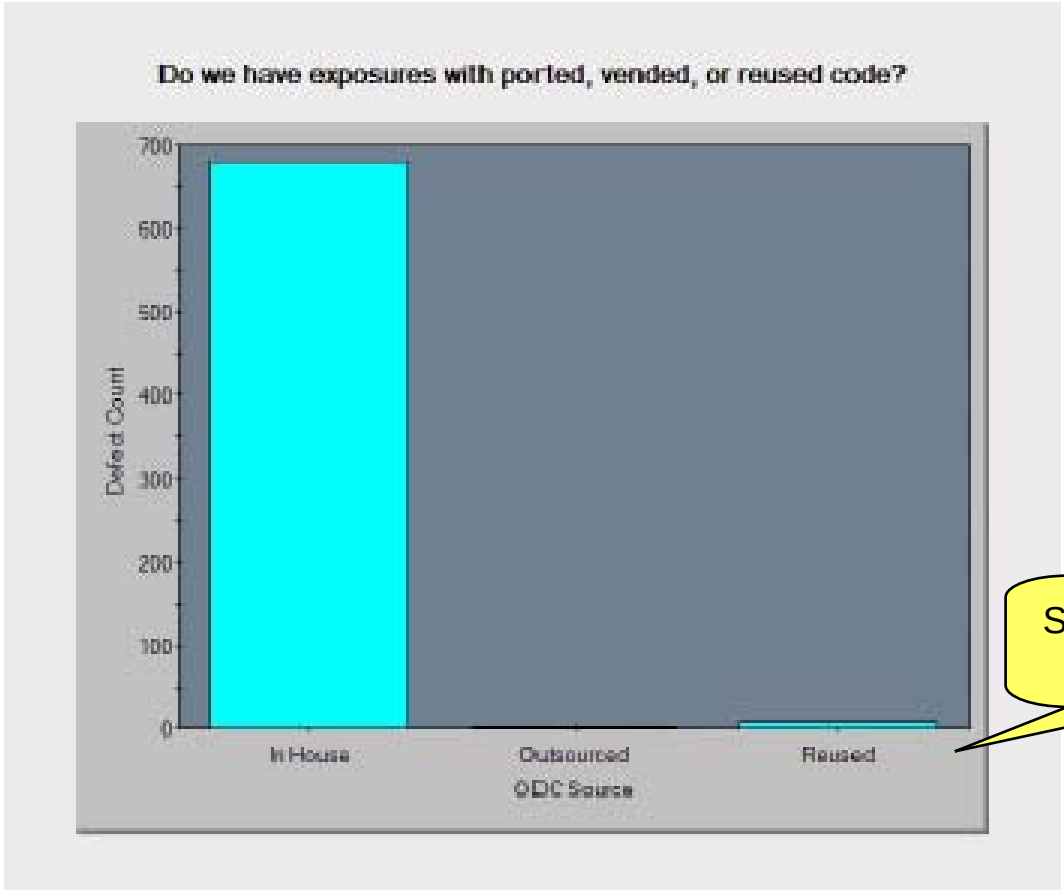
上一个版本的稳定性



Age
Base
New
Rewritten
ReFixed

Age : 造成缺陷的代码是何时开发的

我们移植或重用的代码存在风险吗



Source
Developed In-House
Reused From Library
Outsourced
Ported

Source : 造成缺陷的代码的来源

ODC分析的一些常见模式

主题	ODC属性	非ODC属性
评估测试的有效性	Activity, Qualifier, Trigger, Source, Age	Open date, severity, component, Phase
评估产品的稳定性	Defect type , Impact, Qualifier, Source, Age	Open date, severity, close date, component
鉴别设计和代码中的强处和不足	Defect type, Qualifier	Component
评估客户的使用方法	Impact , Trigger, Defect Type, Qualifier	Open date, Severity, Component
度量进度	Activity, Trigger	Severity, Component, Phase
……		

议程

- 什么是缺陷正交分析
- 使用IBM Rational ClearQuest 实现缺陷正交分析
- 缺陷正交分析实施步骤

ODC对满足如下条件的项目尤其有帮助

- 产品开发生命周期相对较长，质量需要持续改进
  - 如项目需要管理产品的多个软件版本或者同一个版本中的多次迭代
- 潜在缺陷数量特别大
  - 缺陷数量越多，分析结果的针对性和有效性越强
- 项目把“高质量”作为成功的首要目标

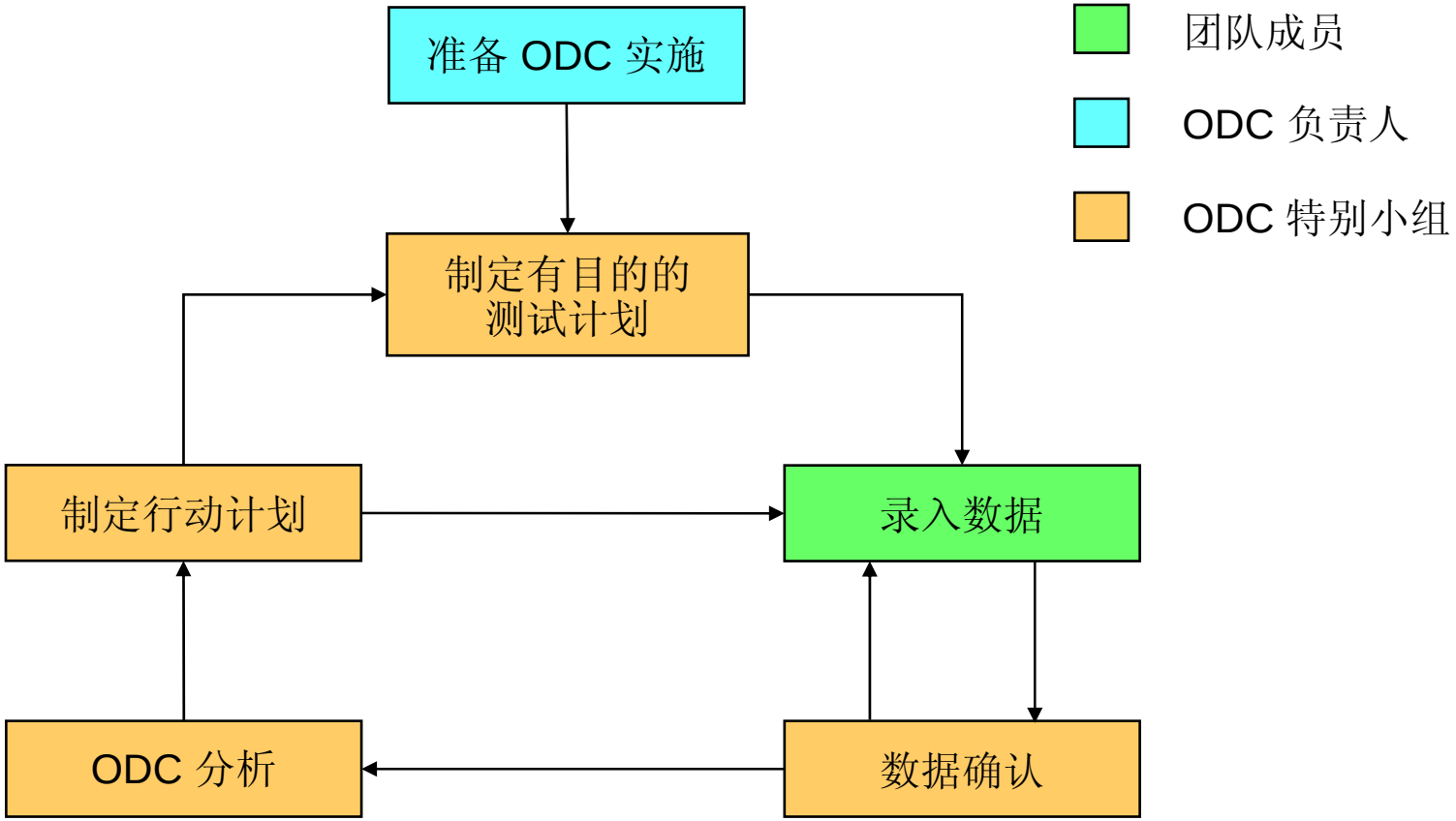
ODC 实施中的三种角色

- 团队成员
  - 团队成员包括开发人员、测试人员、用户，他们负责录入数据
- ODC 负责人 (ODC focal point)
  - ODC 负责人必须熟悉 ODC 分类的执行，他要制定 ODC 实施计划，指导团队成员进行 ODC 分析
  - ODC 负责人可以是项目团队之外的
- ODC 特别小组 (ODC special group)
  - ODC 特别小组由开发、测试人员的代表组成，主要负责制定行动计划、确认录入数据的正确性和进行 ODC 分析.

ODC实施的三个循环

- 根据ODC所需步骤的数量，它有三种可能的循环
  - 大循环： 除了预备步骤，该循环本身含有5个步骤
    - ODC计划步骤与ODC的评估是相关联的，它可以使评估更加有效
  - 中等循环：包含四个步骤
    - 包含ODC生命周期中的核心组成部分
    - 尽管无法进行完整的ODC分析，但仍然可以执行一些有用的评估
  - 小循环：只包含两个步骤
    - 只要找到一定数量的缺陷，随时可能发生确认的活动

ODC 使用模型


准备 ODC 实施
制定有目的的
测试计划
制定行动计划
录入数据
ODC 分析
数据确认
团队成员
ODC 负责人
ODC 特别小组

步骤 1：预备阶段

- 获得领导的批准和支持来实施 ODC 方法
- 获得开发团队和测试团队支持，他们愿意实施 ODC
- 确定一个 ODC 负责人，由他来提供培训和指导
- 分析项目当前的状态
  - 项目当前所处的阶段、如果不是新项目则还要考虑如何使用历史数据
- 确定 ODC 特别小组，由来自于开发和测试团队的代表组成
- 在 ClearQuest 上定制及部署 ODC schema
- 举办 ODC 培训
  - ODC 基本概念 + 数据录入和确认标准

步骤 2：计划

- 把项目分成多个组件
  - 每个缺陷都将追踪到相关组件，以供将来分析之用
  - 组件可以按照功能名称、物理分布或逻辑关系来确定
- 确定执行 ODC 分析的时间点
  - ODC 分析可以在功能测试和用户验收测试之后做
  - ODC 分析时间点的选择将影响后续质量改进的成效
- 对于迭代化的开发流程，可以在每个迭代结束的时候做一下 ODC 分析
- 建立一个 ODC 计划文档，其中应该记录关于资源的分配、所需工作量和分析步骤等

步骤 3 和 4：数据录入和确认

- 在数据录入之前，确保所有的开发和测试人员清楚地理解每一个属性的含义
- 在数据录入之后，需要做一个确认来保证录入数据的正确性.

步骤 5: 分析

- 在收集了一定量的数据之后，我们就可以通过各种统计图表来分析这些数据
- 分析工作可以在项目开发周期的任何时间点进行，我们需要从统计图表中分析出影响质量的原因

步骤 6：行动

- 制定一个正式的行动计划来帮助我们持续地提高产品质量
- 行动计划可以是针对设计文档、源代码、开发流程、测试方法等的改进建议
- 最重要的部分针对下一次迭代或发布所制定的行动计划
- 行动计划的目的必须定义清晰并且是可度量的


Questions


Thank
You