

Software Testing Training

Training Department



吉
祥

提要

第一章 概论 (Basic Concept)

第二章 测试文档的编写 (Test Documents)

第三章 Bug管理系统 (Bug Trace System)



吉祥如意

第一章 概 论

■ 第一节. 软件测试的概念和术语

Concept & Glossary

■ 第二节. 软件测试的目的和原则

Purpose & Fundamental

■ 第三节. 软件测试的分类和比较

Test types



第一节 软件测试的概念和术语

软件测试的概念 (Definition of SW Testing)

IEEE给出的定义是：

是使用人工或自动手段来运行或测定某个系统的过程，其目的在于检验它是否满足规定的需求或是弄清预期结果与实际结果之间的差别。





软件测试的术语

- 软件质量(SW Quality): 软件的功能和性能满足用户需要的程度
- 软件Build: 用于测试的软件中间版本程序。
- 软件缺陷(SW Defect/bug/error): 软件的功能/性能/界面/文档与软件需求文档和用户的需要不一致的现象
- 软件缺陷生命周期(SW defect lifecycle): 报告、确认、修正、验证、关闭

软件测试的术语（续）

吉祥如意

- 测试用例 (Testcase)：包含输入条件、执行步骤和测试期望的正确结果的文档
- 缺陷跟踪系统 (DTS)：管理软件缺陷的整个生命周期的工具
- 静态测试 (Statistic testing)：不执行被测程序而进行的测试
- 动态测试 (dynamic testing)：执行被测程序而进行的测试



吉祥如意

软件测试的术语（续）

- 白盒测试(White box testing): 测试软件代码结构的测试
- 黑盒测试(Black box testing) : 不关心软件代码结构, 以软件输入和输出来测试软件功能的测试。
- 回归测试(Regression testing): 在新的软件Build上验证修正的缺陷是否不再现
- 冒烟测试(Smoke testing): 在大规模测试前, 快速执行的基本功能测试。

软件测试的术语（续）

吉祥如意

- 软件里程碑 (SW Milestone) :
- 软件项目开发的各个关键过程。



第二节 软件测试的目的和原则

- 软件测试的目的 (Purpose of SW Testing)
 - 1 寻找软件的缺陷 (Find bugs) ;
 - 2 跟踪修正软件缺陷 (Trace bugs) ;
 - 3 验证修正的软件缺陷 (Verify bugs) ;

吉
祥

软件测试的原则

- 1 尽早地和不断地进行软件测试。
- 2 程序员应该避免检查自己的程序。
- 3 设计测试用例时应该考虑到合法的输入和不合法的输入以及各种边界条件，特殊情况下要制造极端状态和意外状态。





软件测试的原则（续）

- 4 注意测试中的错误集中发生现象。
- 5 对测试错误结果一定要有一个确认的过程，严重的错误可以召开评审会进行讨论和分析。
- 6 制定严格的测试计划。
- 7 注意回归测试的关联性。
- 8 妥善保存一切测试过程文档。



第三节 软件测试的分类和比较 (Test Type)

- 从测试方式上进行分类和比较
 - 白盒测试(White box): 关心软件内部设计和程序实现, 主要测试依据是设计文档
 - 黑盒测试(Black box): 不关心软件内部, 只关心输入输出, 主要测试依据是需求文档



从测试阶段上进行分类和比较

- 单元测试(Unit Test)的粒度最小，一般由开发小组采用白盒方式来测试，主要测试单元是否符合“设计”。
- 集成测试(Integration Test)界于单元测试和系统测试之间，起到“桥梁作用”，一般由开发小组采用白盒加黑盒的方式来测试，既要验证“设计”又要验证“需求”。
- 系统测试(System Test)的粒度最大，一般由独立测试小组采用黑盒方式来测试，主要测试系统是否符合“需求规格说明书”。
- 验收测试(Validation Test)与系统测试非常相似，主要区别是测试人员不同，验收测试由用户执行。



问题解答(Questions)

问题1 有了“黑盒”测试为什么还要“白盒”测试？

■ 黑盒测试只能观察软件的外部表现，即使软件的输入输出都是正确的，却并不能说明软件就是正确的。因为程序有可能用错误的运算方式得出正确的结果，例如“负负得正，错错得对”，只有白盒测试才能发现真正的原因。



■ 白盒测试能发现程序里的隐患，象内存泄漏、误差累计问题。在这方面，黑盒测试存在严重的不足。



问题解答（续）Questions Cont.

问题2 由于单元测试要写测试驱动程序，非常麻烦，能否等到整个系统全部开发完后，再集中精力进行一次性地单元测试呢？

■ 如果这样做，在开发过程中，缺陷会越积越多并且分布得更广、隐藏得更深，反而导致测试与改错的代价大大增加。最糟糕的是无法估计测试与改错的工作量，使进度失去控制。因此为图眼前省事而省略单元测试或者“偷工减料”，是“得不偿失”的做法。





问题解答（续）Questions Cont.

问题3 如果每个单元都通过了测试，把它们集成一起难道会有什么不妥吗？集成测试是否多此一举？

- 要把N个单元集成一起肯定靠接口耦合，这时可能会产生在单元测试中无法发现的问题。例如：数据通过不同的接口时可能出错；几个函数关联在一起时可能达不到预期的功能；在某个单元里可以接受的误差可能在集成后被扩大到无法接受的程度。所以集成测试是必要的，不是多此一举。



问题解答（续）Questions Cont.



问题4 在集成测试的时候，已经对一些子系统进行了功能测试性能测试等等，那么在系统测试时能否跳过相同内容的试？

- 不能！因为集成测试是在仿真环境中开展的，那不是真正的目标系统。再者，单元测试和集成测试通常由开发小组执行。根据测试心理学的分析，开发人员测试自己的工作成果虽然是必要的，但不能作为成果已经通过测试的依据。



问题解答（续）Questions Cont.

问题5 既然系统测试与验收测试的内容几乎是相同的，为什么还要验收测试？

- 首先是“信任”问题。对于合同项目而言，如果测试小组是开发方的人员，客户怎么能够轻易相信“别人”呢？所以当项目进行系统测试之后，客户再进行验收测试是情理之中的事。否则，那是客户失职。
- 不论是合同项目还是非合同项目，软件的最终用户各色各样（如受教育程度不同、使用习惯不同等等）。测试小组至多能够模仿小部分用户的行为，但并不具有普遍的代表性。

问题解答（续）Questions Cont.



问题6：能否将系统测试和验收测试“合二为一”？

■ 系统测试不是一会儿就能做完的，比较长时间的用户测试很难组织。用户还有自己的事情要做，他们为什么要为别人测试呢？即使用户愿意做系统测试，他们消耗的时间、花费的金钱大多比测试小组的高。

■ 系统测试时会找出相当多的软件缺陷，软件需要反反复复地改错。如果让用户发现“内幕”，一是丢脸，二是会吓跑买主。所以还是关起门来，先让测试小组做完系统测试的好。



吉
祥
如意

提 要

第一章 概论 (Basic Concept)

第二章 测试文档的编写 (Test Documents)

第三章 Bug 管理系统 (Bug Trace System)





第二章 测试文档的编写

- 每个测试过程的基本文档包括：
 - 《测试计划》(Test Plan)：指明测试范围、方法、资源，以及相应测试活动的时间进度安排表的文档。
 - 《测试方案》：指明为完成软件或软件集成特性的测试而进行的设计测试方法的细节文档。
 - 《测试用例》(Test Case)：指明为完成一个测试项的测试输入，预期结果，测试执行条件等因素的文档。
 - 《测试规程》：指明执行测试时测试活动序列的文档。
 - 《测试报告》(Test Report)：指明执行测试结果的文档。



- 第一节 测试计划(Test Plan)的编写

- 测试计划的概念
- 测试计划文档的内容
- 测试计划文档举例

- 第二节 测试用例(Test Case)的编写

- 什么是测试用例
- 良好测试用例的特征
- 设计测试用例的策略选择
- 测试类型与测试用例设计
- 测试用例设计工具
- 黑盒测试用例的设计方法
- 白盒测试用例的设计方法
- 测试用例文档
- 案例研究

- 第三节 测试文档模板(Templates)

- 测试计划参考模板
- 测试用例参考模板
- 测试报告参考模板



第一节 测试计划(Test Plan)的编写

- 测试计划的概念(Definition of Test Plan)
- 测试计划文档的内容(Content of Test Plan)
- 测试计划文档举例(Samples of Test Plan)

测试计划的概念 (Definition of Test Plan)

吉祥如意

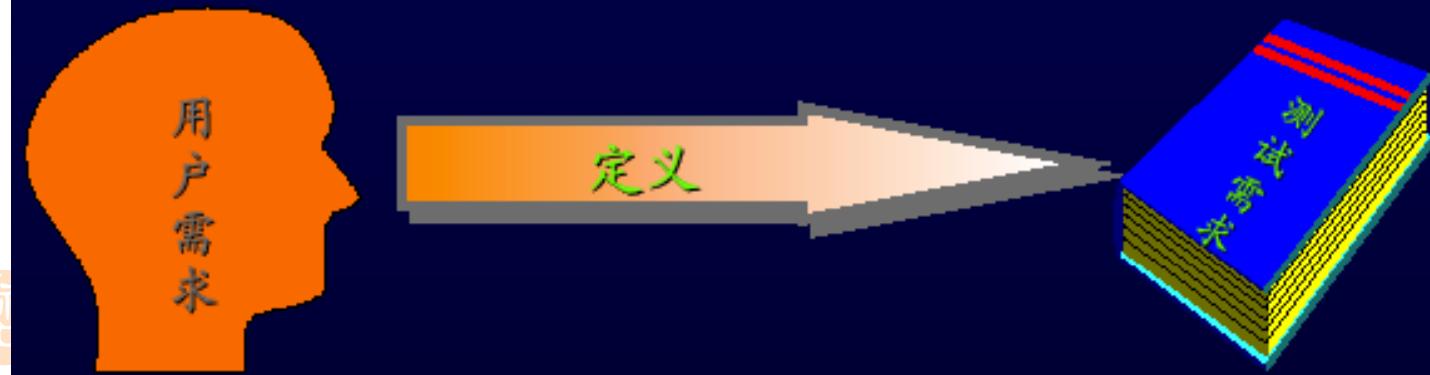
- 测试计划是描述软件测试努力的目标、范围、方法和焦点的文档。
- 专业的测试必须以一个好的测试计划作为基础。尽管测试的每一个步骤都是独立的，但是必定要有一个起到框架结构作用的测试计划。测试的计划应该作为测试的起始步骤和重要环节。



吉祥如意

测试计划来源于测试需求

测试计划：基于测试需求



根据用户需求定义并完善测试需求，以作为整个测试的标准

测试计划文档内容

Content of Test Plan

- 《测试计划》文档应该包括如下内容
 - 目标
 - 概述
 - 组织形式
 - 角色及职责
 - 测试对象
 - 测试通过/失败的标准
 - 测试挂起的标准及恢复的必要条件
 - 测试任务安排
 - 应交付的测试工作产品
 - 工作量估计

吉
祥
如意



《测试计划》文档内容（续）

Content of Test Plan Cont.

■ 目标

- 表示该测试计划所应该达到的目标

■ 概述：包括

- 项目背景：如项目的主要功能特征，体系结构，简要历史等

- 范围：指明该计划的使用对象，范围组织形式

- 表示测试计划执行过程中的组织结构及之间的关系，以及所需要的组织的独立程度。

- 同时指出测试过程与其他过程（开发，管理，...）之间的关系。

- 还包括沟通渠道等

《测试计划》文档内容 (续)

Content of Test Plan Cont.



- 角色与职责
 - 定义角色及其职责，在每个角色与测试任务之间建立关联。

- 测试对象
 - 列出所有将被作为测试目标的测试项（功能需求，非功能需求...）



《测试计划》文档内容（续）

Content of Test Plan Cont.

- 测试通过/失败的标准

- 指明了判断/确认测试何时结束，以及所测试的应用程序的质量。可以直接陈述，也可以引用其他的文档。

- 至少应该说明：

- 什么将被测试？
 - 度量尺度是如何建立的？
 - 使用了哪些标准对度量进行评价？

- 测试挂起的标准以及恢复的必要条件

- 指明挂起全部或部分测试项的标准，并指明恢复测试的标准及其必须重复的测试活动。

《测试计划》文档内容 (续)

Content of Test Plan Cont.

- 测试任务安排

➤ 明确测试任务。对于每个任务说明：

- 任务
- 方法和标准
- 输入/输出
- 时间安排
- 资源
- 风险和假设
- 角色和职责

《测试计划》文档内容（续）

Content of Test Plan Cont.

- 应交付的测试工作产品
 - 指明应该交付的文档，测试代码及测试工具，一般包括测试计划，测试方案，测试用例，测试规程，测试日志，测试事故报告，测试总结报告，测试输入以及测试输出，测试工具。

工作量估计

- 给出前面定义任务的人力需求及总计。

测试计划文档举例

Sample of Test Plan



QE Test Plan: 2. Create Local Mail file Replica via Setup & Desktop Policies

Functional Area: Server Admin Policies - Setup & Desktop

Project Group: Server: Admin

Test Phase:	Regression Test
Release:	7.0
Owner:	Jun ZJ Zheng/China/Contr/IBM
Link to spec:	
TER location:	◆

Status:	Ready for Review
Approver:	Kristen Brouillette/Westford/IBM
Reviewers:	Shobha Hiremath/Westford/IBM, Kristen Brouillette/Westford/IBM
# of test cases:	6
# of automated cases:	

Purpose

The purpose of the **Dynamic Configuration: Create Local Mail file Replica (in Setup & Desktop Policies)** test is to ensure the ability of dyn config to create a local mail file replica upon initial client setup and subsequent authentication with the home server for users with an explicit or organizational Setup/Desktop policy setup for this action. The timing of the replica creation should correspond to the rules of Setup & Desktop Policies & dyn config.

测试计划文档举例（续）

Sample of Test Plan Cont.

Requirements

Please reference the primary Test Plan: [Setup and Desktop Policies: Explicit, Organizational, Both](#) () for domain configuration, Policy conceptual information, and the full Test Plan.

All test must be run on all 7.0 supported and certified platforms server platforms, and access to DB2 builds/servers for DB2 enable environment:

Win2k/Win2003	AIX5.2/AIX5.3	United Linux 1.0/SuSe8	SuSe9	Solaris
<input type="checkbox"/> non-DB2 env. <input type="checkbox"/> Local DB2 env. <input type="checkbox"/> Remote w32 DB2 env. <input type="checkbox"/> Remote AIX DB2 env.	<input type="checkbox"/> non-DB2 env. <input type="checkbox"/> Local DB2 env. <input type="checkbox"/> Remote AIX DB2 env.	<input type="checkbox"/> non-DB2 env.	<input type="checkbox"/> non-DB2 env.	<input type="checkbox"/> non-DB2 env.

The following table is estimated test time for the different types of testing, the Total Test Days is estimated by testing against two client platforms(Winxp and Win2000) and all supported server platforms.

Test Coverage Type	Total Test Days	Total Test Cases	Comments
Feature test (e.g., full pass regression testing)	4	6	This assumes that all included tests are run.
Regression test (e.g., milestone testing)	1	2	Ideally there will be some way to indicate which pieces of the test plan should be run for a milestone/regression type test -- let the team know if you have good suggestions.
Smoketest (e.g., gold candidate testing)	0.5	2	Ditto for a smoketest.



测试计划文档举例（续）

Sample of Test Plan Cont.

Procedure

The 'Create Local Mail file Replica' is a featurette of the Setup Settings & Desktop Settings features. The information below describes the featurette & how to test it. Please incorporate the information below into your testing via the primary Test Plan:

Setup and Desktop Policies: Explicit, Organizational, Both: 

Create Local Mail file Replica

Test procedure:

- launch Admin Client
- open an existing Setup or Desktop Settings doc in edit mode (or create a new one)
- select the **Create local mail file replica** checkbox in the **Local mail file:** field (on the Basics tab - Server Options section)
- save & close the policy

For a Setup Policy with this feature enabled, verify:

1. a local mail file replica is created **during** setup (you'll see the file copying status bar during setup or a status in the Notes Status bar)
2. upon completion of client setup, verify the mail file bookmark points to both *Local* & *<home server>*
3. verify the mail file is listed on the replicator page along w/a line item for "Incoming Mail"
4. CTRL-O to verify there is a *Local* mail directory & that the user's mail file replica exists in this directory
5. open the local mail file replica to assure the ability to open/access the local db



第二节 测试用例(Test Case)的编写

- 什么是测试用例
- 良好测试用例的特征
- 设计测试用例的策略选择
- 测试类型与测试用例设计
- 测试用例设计工具
- 黑盒测试用例的设计方法
- 白盒测试用例的设计方法
- 测试用例文档
- 案例研究



什么是测试用例

Definition of Test Case



■ 测试内容的一系列情景和每个情景中必须依靠输入和输出，而对软件的正确性进行判断的测试文档，称为测试用例。

■ 测试用例就是将软件测试的行为活动，做一个科学化的组织归纳。



良好测试用例的特征



How to write a good Test Case

- 可以最大程度地找出软件隐藏的缺陷
- 可以最高效率的找出软件缺陷
- 可以最大程度地满足测试覆盖要求
- 既不过分复杂、也不能过分简单使软件缺陷的表现可以清楚的判定
 - 测试用例包含期望的正确的结果
 - 待查的输出结果或文件必须尽量简单明了
- 不包含重复的测试用例
- 测试用例内容清晰、格式一致、分类组织



设计测试用例的策略选择

- 测试用例的设计方法不是单独存在的，具体到每个测试项目里都会用到多种方法，每种类型的软件有各自的特点，每种测试用例设计的方法也有各自的特点，针对不同软件如何利用这些黑盒方法是非常重要的。
 - 在实际测试中，往往是综合使用各种方法才能有效提高测试效率和测试覆盖度，这就需要认真掌握这些方法的原理，积累更多的测试经验，以有效提高测试水平。
 - 首先进行等价类划分，包括输入条件和输出条件的等价划分，将无限测试变成有限测试，这是减少工作量和提高测试效率的最有效方法。
 - 在任何情况下都必须使用边界值分析方法。经验表明用这种方法设计出测试用例发现程序错误的能力最强。
 - 对照程序逻辑，检查已设计出的测试用例的逻辑覆盖程度。如果没有达到要求的覆盖标准，应当再补充足够的测试用例。
- 对于业务流清晰的系统，可以利用场景法贯穿整个测试案例过程，在案例中综合使用各种测试方法。



测试类型与测试用例设计

Test Types & Test Case Design

根据测试类型设计

功能测试

回归测试

易用性测试

界面测试

配置测试

文档测试

压力测试

国际化测试

- 测试用例1
- 测试用例2
- 测试用例3



根据程序功能模块设计

安装/卸载测试

联机注册测试

联机帮助测试

文件操作测试

软件更新测试

数据备份测试

- 测试用例1
- 测试用例2
- 测试用例3

- 测试用例1
- 测试用例2
- 测试用例3



测试用例设计工具

吉祥如意

- 设计依据:
- 软件需求文档
- 软件设计文档

常见问题:

- 软件文档不全或没有文档
- 没有完成编码就开始设计测试用例

通用设计工具:

- Microsoft Word
- Microsoft Excel
- Microsoft Access

设计工具专用:

- IBM Rational TestManager
- Mercury Interactive TestDirector



黑盒测试用例的设计方法

Methodologies of Test Case writing

1. 等价类划分方法
2. 边界值分析方法
3. 错误推测方法
4. 因果图方法
5. 判定表驱动分析方法
6. 正交实验
7. 设计方法
8. 功能图分析方法



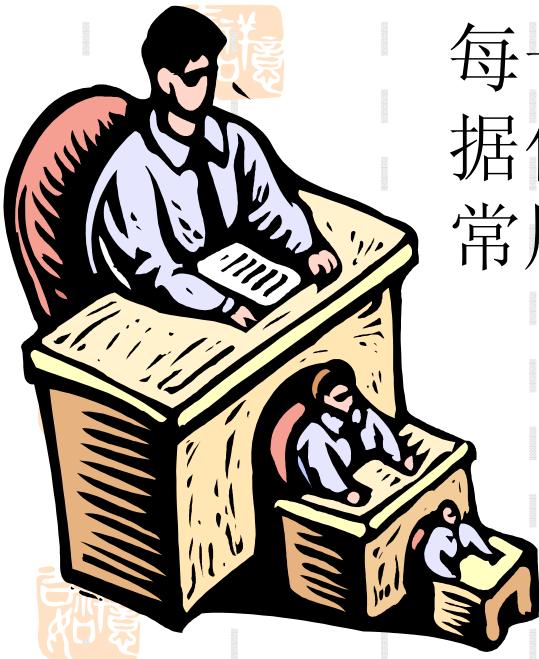


1. 等价类划分方法

Equivalence partition testing

定义：

是把所有可能的输入数据，即程序的输入域划分成若干部分（子集），然后从每一个子集中选取少数具有代表性的数据作为测试用例。该方法是一种重要的，常用的黑盒测试用例设计方法。



1. 等价类划分方法（续）

Equivalence partition testing

方法：

- 等价类划分的办法是把程序的输入域划分成若干部分，然后从每个部分中选取少数代表性数据当作测试用例。
- 每一类的代表性数据在测试中的作用等价于这一类中的其他值，也就是说，如果某一类中的一个例子发现了错误，这一等价类中的其他例子也能出现同样的错误。
- 使用这一方法设计测试用例，首先必须在分析需求规格说明的基础上划分等价类，列出等价类表。
- 在考虑等价类划分时，先从程序的功能说明中找出每个输入条件，然后为每个输入条件划分两个或更多个等价类。

1. 等价类划分方法（续）

Equivalence partition testing

两种等价类:有效等价类和无效等价类

- 有效等价类是指对程序的规格说明是有意义的、合理的输入数据所构成的集合；
- 无效等价类是指对程序的规格说明是不合理的或无意义的输入数据所构成的集合。

注意:设计测试用例时,要同时考虑这两种等价类.因为,软件不仅要能接收合理的数据,也要能经受意外的考验.这样的测试才能确保软件具有更高的可靠性.

吉祥如意

2. 边界值分析方法 boundary value testing

- 边界值分析：是一种补充等价划分的测试用例设计技术，它不是选择等价类的任意元素，而是选择等价类边界的测试用例。
- 实践证明，在设计测试用例时，对边界附近的处理必须给予足够的重视，为检验边界附近的处理专门设计测试用例，常常可以取得良好的测试效果。边界值分析不仅重视输入条件边界，而且也从输出域导出测试用例。





用等价类划分法和边界值分析法设计用例

例如：学生成绩管理系统中的学生分数的输入（不计小数点）。

这时，我们可以确定输入数据的最小值Nmin和最大值NMax，则有效的数据范围是 $N_{min} \leq N \leq N_{Max}$ ，学生分数的输入范围是 $0 \leq N \leq 100$ ，这个范围就是有效数据区域。

除此之外，就是无效数据区域，即 $N < N_{min}$ 或 $N > N_{Max}$ ，如 $N < 0$ 或 $N > 100$ 。

这时测试的数据从近乎无限的数据简化为5个输入数据，就是：



用等价类划分法和边界值分析法设计用例 (续)

边界值两个：Nmin和NMax，如0和100

有效数据的等价输入值 Ni，如75

无效数据的等价输入值两个：NLm1和NLm2，如 -999和999

为了得到更好的覆盖率，可以在最靠近边界取一些值，共四个，即： Nmin +1， Nmin -1， NMax +1， NMax -1，如 -1， 1， 99， 101

所以一个有效的测试数据集合是 {-1, 0, 1, 99, 100, 101}；
更完整的测试数据集合是 {-999, -1, 0, 1, 75, 99, 100, 101, 999}。



3. 错误推测方法

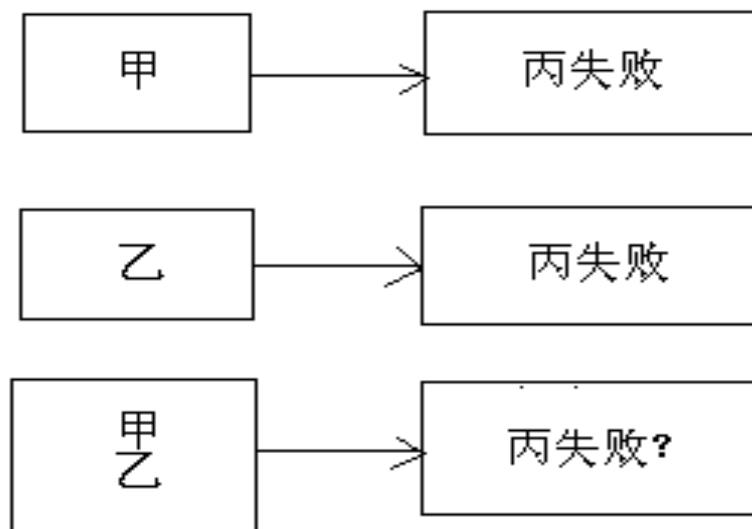
- 错误推测法：基于经验和直觉推测程序中所有可能存在的各种错误，从而有针对性的设计测试用例的方法。
- 错误推测方法的基本思想：列举出程序中所有可能有的错误和容易发生错误的特殊情况，根据他们选择测试用例。
- 例如：数据测试中的 缺省值、空白、空值、零值、无





4. 因果图方法 cause-effect graph

- 因果图方法：多种输入条件的组合，产生多种结果。





白盒测试用例的设计方法

■ 什么是白盒测试

- 白盒测试也称为结构测试，把程序看作一个透明的盒子，测试程序的代码书写结构和逻辑问题

■ 白盒测试用例的设计方法

- 逻辑覆盖：以程序的内部逻辑结构为基础，分为语句覆盖、判定覆盖、判定-条件覆盖、条件组合覆盖等

- 基本路径测试：在程序控制流程的基础上，分析控制构造的环路复杂性，导出基本可执行路径集合，从而设计测试用例。

■ 白盒测试用例注意事项

- 由于测试路径可能非常多，由于时间和资源问题，选出足够多的路径测试

- 由于深入到程序编码，通常开发人员协助测试人员书写白盒测试用例



测试用例文档

Test Case Samples

吉祥如意

- 测试用例列表

测试项目	测试子项目	用例编号	用例级别	输入值	预期输出	实测结果	备注
测试项目1	测试子项目1						
总数		—			—	—	—



测试用例列表解释

Test Case Samples



■ 测试用例列表解释

- 测试项目：指明并简单描述本测试用例是用来测试那些软件项目，软件子项目或者软件特性。
- 用例编号：对该测试用例分配唯一的标志编号。
- 用例级别：表明该用例的重要程度。（如果该用例没有通过的话，软件质量受到的影响。）



测试用例列表解释（续）

Test Case Samples



■ 测试用例级别解释：

- 1级：基本。用例涉及系统的基本功能，作为版本通过准则。项目如果不能通过这样的案例，需要考虑重新提交版本。
- 2级：重要。用例涉及单个版本特性。如果这个版本通不过，可以作为重要或者一般问题提交报告。
- 3级：详细。用例反应的是某个单项功能的某个细节方面。比如：有关性能，极限，用户界面，...
- 4级：生僻。对应于比较生僻的前置条件和数据设置。对系统质量的影响不大。





测试用例列表解释（续）

■ 输入值

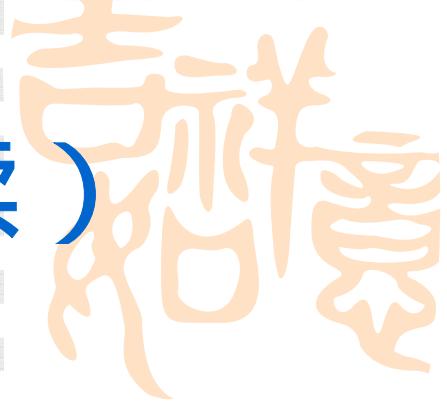
- 列出执行本测试用例所需要的具体的输入值。可能的表示方式包括：

直接输入；指明输入范围；引用常量表或事务文件；指明相关数据库；...

■ 预期输出值

- 描述被测项目或被测特性所希望或要求达到的输出或指标。如有需要，也要列出描述预期输出数据的允许误差范围。对于复杂的数据，如数据库，也需要相应的描述。





测试用例列表解释（续）

■ 实测结果

在测试执行的时候填写，指明该测试用例是否通过。若否，需要列出实际测试时的测试输出值。

■ 备注

必要时，需要填写：特殊环境需求，特殊测试步骤要求，相关测试用例等信息。



测试用例列表解释（续）

Test Case Samples



- 特殊环境要求
 - 包括：硬件需求，软件需求，其他需求。
- 特殊测试步骤要求：
 - 指明执行本测试用例对相应的测试规程的任何特别的限制或要求。比如：初始化要求，人为干预，人观测和判断的步骤，特殊的恢复步骤等。
- 相关测试用例
 - 列出必须先于本测试用例执行的测试用例。指明本用例和它们之间的依赖关系。



案例研究：判断三角形的形状

Test Case Samples



测试场景：

一个程序读入3个整数，把这三个数值看作一个三角形的3条边的长度值。这个程序要打印出信息，说明这个三角形是不等边的、是等腰的、还是等边的。

确定输入数据与三角形形状的关系：

- 设三角形的3条边分别为A, B, C。如果它们能够构成三角形的3条边，必须满足：
 - $A > 0, B > 0, C > 0$, 且 $A+B > C, B+C > A, A+C > B$;
 - 如果是等腰的，还要判断 $A=B$, 或 $B=C$, 或 $A=C$;
 - 如果是等边的，则需判断是否 $A=B$, 且 $B=C$, 且 $A=C$ 。



案例研究：判断三角形的形状（续）

Test Case Samples

创建等价类表：

输入条件	有效等价类	无效等价类
是否三角形的三条边	$(A>0)$, (1) $(B>0)$, (2) $(C>0)$, (3) $(A+B>C)$, (4) $(B+C>A)$, (5) $(A+C>B)$, (6)	$(A\leq 0)$, (7) $(B\leq 0)$, (8) $(C\leq 0)$, (9) $(A+B\leq C)$, (10) $(B+C\leq A)$, (11) $(A+C\leq B)$, (12)
是否等腰三角形	$(A=B)$, (13) $(B=C)$, (14) $(C=A)$, (15)	$(A\neq B)$ and $(B\neq C)$ and $(C\neq A)$ (16)
是否等边三角形	$(A=B)$ and $(B=C)$ and $(C=A)$ (17)	$(A\neq B)$, (18) $(B\neq C)$, (19) $(C\neq A)$, (20)

案例研究：判断三角形的形状(续)

Test Case Samples

确定等价类输入数据：

序号	【A, B, C】	覆盖等价类	输出
1	【3, 4, 5】	(1), (2), (3), (4), (5), (6)	一般三角形
2	【0, 1, 2】	(7)	
3	【1, 0, 2】	(8)	
4	【1, 2, 0】	(9)	
5	【1, 2, 3】	(10)	不能构成三角形
6	【1, 3, 2】	(11)	
7	【3, 1, 2】	(12)	
8	【3, 3, 4】	(1), (2), (3), (4), (5), (6), (13)	
9	【3, 4, 4】	(1), (2), (3), (4), (5), (6), (14)	等腰三角形
10	【3, 4, 3】	(1), (2), (3), (4), (5), (6), (15)	
11	【3, 4, 5】	(1), (2), (3), (4), (5), (6), (16)	非等腰三角形
12	【3, 3, 3】	(1), (2), (3), (4), (5), (6), (17)	是等边三角形
13	【3, 4, 4】	(1), (2), (3), (4), (5), (6), (14), (18)	
14	【3, 4, 3】	(1), (2), (3), (4), (5), (6), (15), (19)	非等边三角形
15	【3, 3, 4】	(1), (2), (3), (4), (5), (6), (13), (20)	

第三节 测试文档模板

Templates of Test Documents

- 测试计划(Test Plan)参考模板
- 测试用例(Test Case)参考模板
- 测试报告(Test Report)参考模板

测试计划(Test Plan)的参考模板

类别	递交时间	建议制定者	建议审批者
《单元测试计划》	在系统设计阶段就可以起草，最迟可在实现阶段之初递交。	开发小组的技术负责人	项目经理
《集成测试计划》	在系统设计阶段就可以起草，最迟可在实现阶段之初递交。	开发小组的技术负责人	项目经理
《系统测试计划》	在需求开发阶段就可以起草，最迟可在开发工作完成之际递交。	独立测试小组的负责人	项目经理
《验收测试计划》	在需求开发阶段就可以起草，最迟可在系统测试工作完成之际递交。	项目经理	客户代表或上级领导

1. 测试范围与目的
2. 测试方法
3. 测试环境与测试辅助工具的描述
4. 测试启动准则与结束准则
5. 应递交的文档

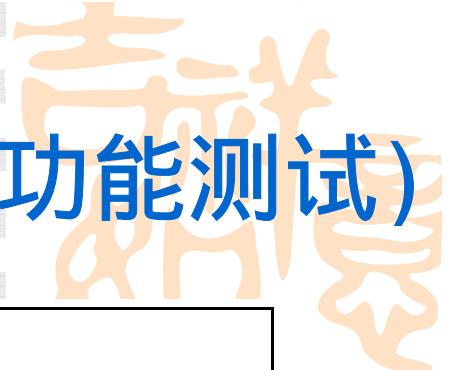
文档名称	预计递交时间
《测试计划》	
《测试用例》	
《测试报告》	

6. 测试人员的组织
7. 任务表

测试任务	开始时间	结束时间	执行人员

8. 可能存在的问题与对策
- 附录：本计划的审批意见

测试用例(Test Case)的参考模板(功能测试)



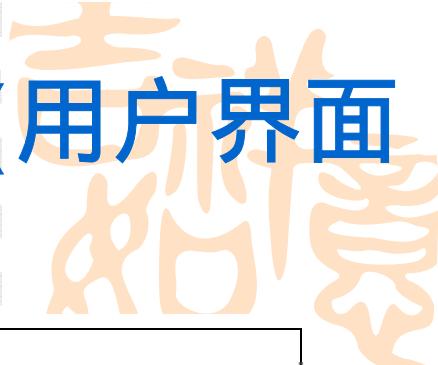
1. 被测试对象的介绍
2. 测试范围与目的
3. 测试环境与测试辅助工具的描述
4. 测试驱动程序的设计
5. 功能测试用例列表

功能 A 描述		
用例目的		
前提条件		
输入/动作	期望的输出/相应	实际情况
示例：典型值...		
示例：边界值...		
示例：异常值...		
功能 B 描述		
~~~~~		

~~~~~



测试用例(Test Case)的参考模板(用户界面 测试)



1. 被测试对象的介绍
2. 测试范围与目的
3. 测试环境与测试辅助工具的描述
4. 测试驱动程序的设计
5. 测试人员的分类

| 类别 | 特征 |
|-------|----|
| A类 | |
| B类 | |
| | |

6. 用于测试用户界面的检查表

| 检查项 | 测试人员的类别及其评价 |
|-------------------------------|-------------|
| 重点检查正确性、易用性和美观程度，参见
表 7-16 | |
| | |
| | |

.....



测试用例(Test Case)的参考模板(安装/反安装测试)



1. 被测试对象的介绍
2. 测试范围与目的
3. 测试环境与测试辅助工具的描述
4. 安装/反安装测试用例

| 配置说明 | | |
|-------|------|------|
| 安装选项 | 是否正常 | 难易程度 |
| 全部 | | |
| 部分 | | |
| 升级 | | |
| 其它 | | |
| 反安装选项 | 是否正常 | 难易程度 |
| | | |
| | | |

XXXXXX



测试报告(Test Report)的参考模板

1. 基本信息

| | |
|----------|----------------------|
| 测试计划的来源 | 提示：填写《测试计划书》名称，版本，时间 |
| 测试用例的来源 | 提示：填写《测试用例》名称，版本，时间 |
| 测试对象描述 | |
| 测试环境描述 | |
| 测试驱动程序描述 | 提示：可以把测试驱动程序当作附件 |
| 测试人员 | |
| 测试时间 | |

2. 实况记录

| 测试用例 | 测试情况 | 错误严重程度 |
|------|------|--------|
| | | |
| | | |
| | | |
| | | |

3. 分析与建议

提示：对测试结果进行分析，提出建议。

4. 错误修改记录

| 错误名称 | 原因 | 修改人 | 修改时间 | 是否回归测试 |
|------|----|-----|------|--------|
| | | | | |
| | | | | |

▪ 附件

吉
祥
如
意

提要

第一章 概论 (Basic Concept)

第二章 测试文档的编写 (Test Documents)

第三章 Bug管理系统 (Bug Trace System)



第三章 Bug管理系统

Bug Trace System

吉祥如意

- 第一节 缺陷分类和缺陷跟踪系统的作用
- 第二节 缺陷跟踪文档的具体组成结构和功能
- 第三节 缺陷报告的生命周期和缺陷追踪





第一节 缺陷分类和缺陷跟踪系统的作用

- 软件缺陷(Bug)的定义
- 为什么会出现软件缺陷
- 软件缺陷的分类
- 缺陷跟踪系统的作用



吉
祥



吉
祥



吉
祥

软件缺陷的定义

Definition of Bug

吉
祥
如
意

符合下列五种情况之一的就可以认为是软件缺陷(bug)：

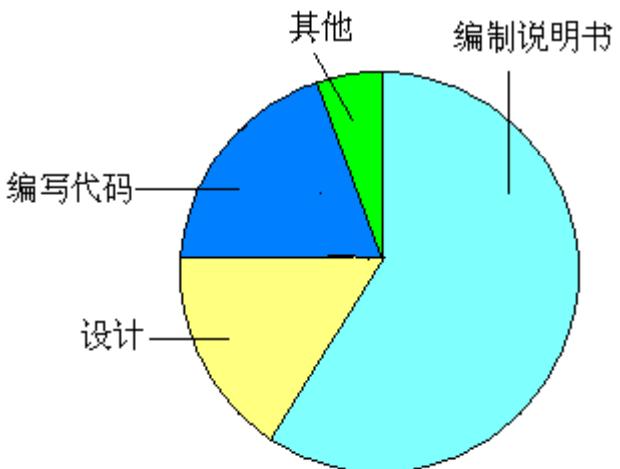
1. 软件未达到软件产品需求说明书指明的要求。
2. 软件未达到软件产品需求说明书虽未指明但应达到的要求。
3. 软件出现了软件产品需求说明书指明不会出现的错误。
4. 软件功能超出软件产品需求说明书指明的范围。
5. 软件测试人员认为难以理解、不易使用、运行速度缓慢、或者最终用户认为不好的问题。



吉祥如意

为什么会出现软件缺陷

- 1. 导致软件缺陷最大的原因是产品说明书。
- 2. 第二大来源是设计方案。
- 3. 某些软件缺陷产生的条件被错误认定。



软件缺陷的分类

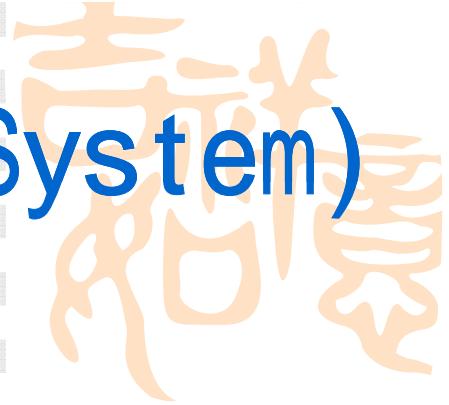
The types of bugs

吉
祥
如
意

- 1. 软件需求缺陷
- 2. 功能和性能缺陷
- 3. 软件结构缺陷
- 4. 数据缺陷
- 5. 实现和编码缺陷
- 6. 软件集成缺陷
- 7. 操作系统调用缺陷
- 8. 测试定义和测试执行缺陷



缺陷跟踪系统(Bug Trace System) 的作用



1. 便于集中管理，提高效率；
2. 有利于缺陷的清楚传达；
3. 便于查找和跟踪。对于大型软件的测试，报告的缺陷总数可能成百上千个，如果没有缺陷跟踪系统的情况下，要求查找某个缺陷，简直是搜索者的恶梦，其难度和效率可想而知的。
4. 便于跟踪和监控缺陷的处理过程和方法。可以方便地检查处理方法是否正确，可以确定处理者的姓名和处理时间，作为工作质量的统计和考核的参考。



缺陷跟踪系统(Bug Trace System)的作用(续)

5. 安全性高，通过权限设置，不同权限的用户能执行不同操作，保证只有适当的人员才能执行正确的处理。
6. 保证处理顺序的正确性，根据当前缺陷的状态，决定当前缺陷的处理方法。
7. 便于项目结束后的存档。缺陷跟踪系统具有方便存储的特点，可以随时或在项目结束后存储，以备将来参考。



第二节 缺陷跟踪文档(Bug Report)的具体组成 结构和功能

根据测试项目的特点和测试要求，确定缺陷跟踪系统的具体组成结构和功能。



吉
祥
如
意

缺陷的填写原则

- 首先，在数据库中查询该缺陷是否已存在。
 - 检查该缺陷对于同一语言是否已存在
 - 检查该缺陷对于其它语言是否已存在
- 不要在同一缺陷中描述不同的问题
- 缺陷的提出是根据语言而不是操作平台
 - 如果一个缺陷出现在一种语言的四个操作平台中，那么只需报告这种语言的一个缺陷而不是四个缺陷。
 - 如果一个缺陷出现在五种语言的十二个操作平台中，那么需要报告五个缺陷而不是十二个缺陷。
- 对于多种语言的同一类缺陷，总是维护主缺陷，无论这个主缺陷是哪种语言，也无论其它语言是什么。
- 主缺陷总是包含英语版本的附图，无论这个主缺陷是否在英语版本中出现。
- 写缺陷时一定要注意到描述的精确度和可读性。



[New Slave Bug Track](#)[Save & Exit](#)**Title:**\*
『』**Basics****Project:**\*
『』**Severity:**\*
『3=Minor Functionality』**OS:**\*
『』**Bug in US:**\*
『No』**Source:**\*
『China』**How Found:**
『Manual Test』**Default Settings:**
『Save』**Build#Ver:**\*
『』**Priority:**\*
『2 = Should fix soon, before product release』**Language:**\*
『Simplified Chinese』**Master Bug:**
『』**Site:**
『』**Email alert to author on state change:**
 yes**Email alert options:**
『』**Body**Guideline: include detail description followed by: (1) [ENV] (2) [Repro Steps] (3) [Result] (4) [Expected] (5) [Comment]

『』

Attachments**[Screen shot(GIF/JPEG)]:**

『』

How to write bugs



▪ Bug Title:

<lang>:<product>:<version>: <platform> :<component>:<description>

➤ Description should be accurate, short; better using keyword, phrase instead of long sentence which is easy for your search later.

Examples



CHS : Voyager: 4374 : Administration: New Host Instance: Duplicate Hotkey 'H'



IT: Rosetta : 677 : Report Designer : TableProperty : Visible : MissingHotkey



TW: Rosetta: 677: Report Designer: misalignment in 'Advanced Textbox Properties' window



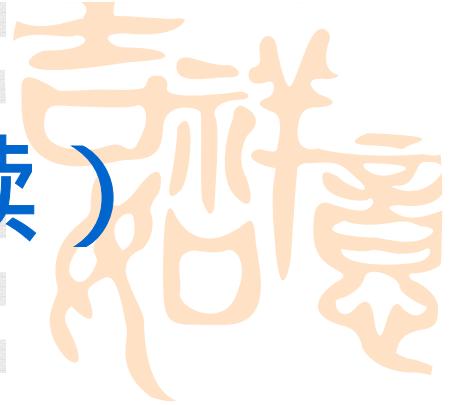
How to write bugs (续)



□Title - lang (East asian)

| | English | Chinese | Chinese | Korean | Japanese |
|-----------------|-------------------------|---------------|------------------|--------|----------|
| Lang Abbrev.(3) | ENU | CHS | CHT | KOR | JPN |
| Lang Abbrev.(2) | US | SC | TC | CH | JP |
| Country Abbrev. | USA | CHN | TWN | KOR | JPN |
| Locale | English (United States) | Chinese (PRC) | Chinese (Taiwan) | Korean | Japanese |
| Win-CodePage | 1252 | 936 | 950 | 949 | 932 |
| LangID | 0x0409 | 0x0804 | 0x0404 | 0x0412 | 0x0411 |

How to write bugs (续)



□ Title – lang (European-1)

| | German | French | Spanish | Italian |
|--------------------|-----------------------------|----------------------------|------------------------------------|----------------------------|
| Lang
Abbrev.(3) | DEU | FRA | ESN | ITA |
| Lang
Abbrev.(2) | DE | FR | ES | IT |
| Country
Abbrev. | DEU | FRA | ESP | ITA |
| Locale | German
(Germany) | French
(France) | Spanish
(International) | Italian
(Italy) |
| Win-CodePage | 1252 | 1252 | 1252 | 1252 |
| LangID | 0x0407 | 0x040C | 0x040A | 0x0410 |

How to write bugs (续)



>Title – *lang* (European-2)

| | | | |
|--------------------|------------------------|------------------------|---------------|
| | Portuguese
(BR) | Dutch | Swedish |
| Lang
Abbrev.(3) | PTB | NLD | SVE |
| Lang
Abbrev.(2) | BR | NL | SV |
| Country
Abbrev. | BRA | NLD | SWE |
| Locale | Portuguese
(Brazil) | Dutch
(Netherlands) | Swedish |
| Win-CodePage | 1252 | 1252 | 1252 |
| LangID | 0x0416 | 0x0413 | 0x041D |

How to write bugs (续)



▪ Bug Body

- **[ENV]** : including basic OS, SP if any, APP, APP build number etc. AND language for each.
- **[CON]** : any pre-condition that repro the bug, such as special configuration etc.
- **[Repro Steps]** : listing step by step in detail
- **[Result]** : record what you see, but not the judgment
- **[Expectation]** : write down the right result, if you know the principle definitely, write it also.
- **[Comment]** or **[Notes]** : record what you find honestly beyond the bug itself, such as whether repro on other languages or other platform, and the bug number.

Tip: In Repro Steps, try to use *click 'Start' -> 'Program' -> 'SQLXML 3.0'*, instead of statement

"Click SQLXML 3.0 on Start menu"



How to write bugs (续)

▪ Attachments

Picture :

Name(IT\_B3450\_Dupkey\_A.png, EN pic)

- Format of Picture.
- Tools (Paint and HyperSnap).
- Store (

\testserver\bugs\SQLxml3\3457\cn)



More Examples

[Repro Steps]

1. Start SQLXML3.0, create a new virtual machine
2. Click the property of the default virtual machine
3. choose "virtual names" page, choose type "dbobject"
4. press hotkey ALT+P
5. Result: The 'properties' dialog be closed.

Compare to:

1. Click 'Start'->'Program'->'SQLXML 3.0' -> 'Configure IIS Support'
2. Click the virtual directory on your machine
3. Right click 'Default Web Site', Click 'New'
4. Create a new virtual directory
5. In 'New Virtual Directory Property' Dialog, click 'Virtual Name' Tab
6. Input the name in 'Name' control, select 'dbobject' in 'Type' control list.
7. Then Press hotkey 'Alt +P'
8. Result: 'New Virtual Directory Property' dialog crashes.

More Examples (续)



[ENV]

Install Windows 2000 ADV Server + Windows Updates
Install MDAC 2.7 SP1
Install SQL Server 2000 ENT edition (Server + Client Tools)
Install Visual Studio .NET 7.0
Install .NET Framework 1.1

[Repro Step]

1. Double click the setup icon of N.S. 2.0 SP1 RC24
2. When Setup wizard is appeared, then click next and stop on the page "User infomation"

[Result]

The hotkey is in the wrong position. The hotkey is located in the middle of the sentence.

[Expect Result]

The hotkey should be the end of the sentence. Please see the attached picture cht2kad\_hotkey.jpg.

[comment]

This bug is valid also for Windows 2003, win 2000 adv, and Windows XP.

This bug is also valid for CHS language for all 3 OS.

Comments:

1. Should indicate lang in [ENV]
2. 'Double click' need to indicate file name
3. Description is repeated
4. 'Please see the attached picture cht2.kad\_hotkey.jpg' should be written in [Result].
5. This is not a functional bug, so should file another bug for CHS.
6. Typo and writing style (punctuation and abbreviation)

More Examples (续)



[ENV]

Simplified Chinese windows XP + SP1

[Repro Step]

1. Install SP3a BOL to default directory.
2. Click start —> All Program —> Microsoft SQL Server 2000 Book Online(Updated-sp3) to bring up SQL Server Book Online.
3. Click "send feedback about this page " icon
4. Click "Submit Feedbck" link to bring up new mail dialog.
5. Be sure the outlook express has not been configued before.
6. Follow the outlook express wizard until you get the feedback mail.
7. See title, subject and default content in the mail,please.



[Result]



There are garbage strings in title,subject and content. But when click "Submit Feedbck" link again,the issue disappears.

[Expect Result]

There should be in the right version of Simplified Chinese.



Note: This issue occurs in other platform such as Windows 2000 advanced server+SP3,window 98,NT4.0 SP6.



Comments:

1. the condition should be written at the top of the bug
2. 'disappear' is a condition when the bug disappear, so should be in Note part.
3. Typo and writing style (punctuation and abbreviation)



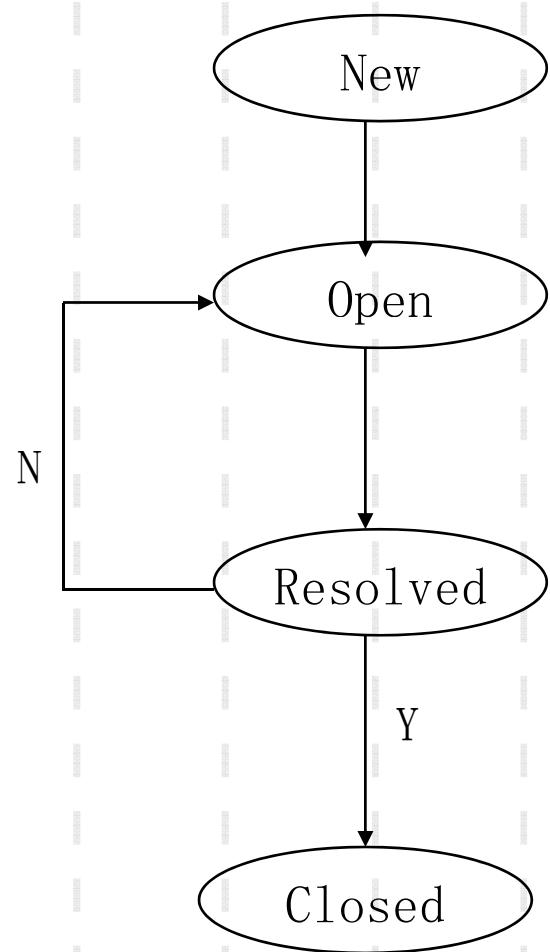
Severity level

- 1 – Critical – Missing Content, Un-localized content, corrupt TOC....
- 2 - Major - Mainly Functionality (Broken links, incorrect hyperlink) unacceptable artwork...
- 3 - Minor – Formatting or style differences.
- 4 - Trivial - Very Small bug. Missing Full stop / Period Etc



第三节 缺陷报告的生命周期和缺陷追踪

缺陷报告的生命周期





缺陷追踪 (Bug Tracking)

- 测试工程师通过缺陷缺陷报告把问题提交给开发人员；开发人员根据测试人员、客户提交的 BUG 的现象、描述、重现条件等数据对该缺陷进行分析，修正完成后，除了将修正的代码提交，编译成产品，还要修改缺陷追踪软件中记录的对应的缺陷的状态，将其状态改为“Resolved”；现在又传到了测试人员手中，测试人员对新编译的产品进行测试，看是否真的已经修正了这个缺陷，如果没有完成修正，重新将缺陷的状态改为“Active”；如果经过测试发现已经修正，则将缺陷的状态改正为“closed”
- “Active” 代表新提交的 bug，
- “Closed” 代表已经修正的bug, 被关闭。
- “Resolved” 代表已经解决，但还需测试工程师验证。

吉祥

謝!

謝

