

21世纪高职高专系列规划教材·计算机类专业
高职高专“十二五”规划教材

软件测试技术与 案例实践教程

RUAJIAN CESHI JISHU
YU ANLI SHIJIAN
JIAOCHENG

主 编◎刘竹林
副主编◎韩 莉 曾金发



北京师范大学出版集团
BEIJING NORMAL UNIVERSITY PUBLISHING GROUP
北京师范大学出版社

21世纪高职高专系列规划教材·计算机类专业
高职高专“十二五”规划教材

软件测试技术与 案例实践教程

RUAJIAN CESHI JISHU
YU ANLI SHIJIAN
JIAOCHENG

主 编◎刘竹林
副主编◎韩 莉 曾金发



北京师范大学出版集团
BEIJING NORMAL UNIVERSITY PUBLISHING GROUP
北京师范大学出版社

图书在版编目(CIP)数据

软件测试技术与案例实践教程 / 刘竹林著. —北京: 北京师范大学出版社, 2011.5

21世纪高职高专系列规划教材

ISBN 978-7-303-12923-2

I. ①软… II. ①刘… III. ①软件—测试—高等职业教育—教材 IV. ①TP311.5

中国版本图书馆 CIP 数据核字(2011)第 090010 号

出版发行: 北京师范大学出版社 www.bnup.com.cn

北京新街口外大街 19 号

邮政编码: 100875

印 刷: 北京京师印务有限公司

装 订: 三河万利装订厂

经 销: 全国新华书店

开 本: 184 mm × 260 mm

印 张: 19

字 数: 420 千字

版 次: 2011 年 6 月第 1 版

印 次: 2011 年 6 月第 1 次印刷

定 价: 32.00 元

策划编辑: 周光明

责任编辑: 周光明

美术编辑: 高 霞

装帧设计: 华鲁印联

责任校对: 李 菡

责任印制: 孙文凯

版权所有 侵权必究

反盗版、侵权举报电话: 010-58800697

北京读者服务部电话: 010-58808104

外埠邮购电话: 010-58808083

本书如有印装质量问题, 请与印制管理部联系调换。

印制管理部电话: 010-58800825

前 言

伴随着 IT 行业的高速发展,软件开发方法越来越先进,实现的功能也越来越强大,应用软件几乎涉及了国计民生的各个领域。而与之相反的是,这些应用系统的整体性能却越来越低,尤其是一些涉及多用户的 Web 应用系统。在国内应用软件中出现了越来越多的“亚健康软件”。

亚健康不但威胁着 IT 人的生活质量,也威胁很多应用软件的性能,这就是软件的“亚健康”。

信息技术的飞速发展,使软件产品应用到社会的各个领域,软件产品的质量自然成为人们共同关注的焦点。软件质量是软件产品的根本,软件测试是保证软件质量的重要手段。

软件测试是开发一个软件必要的步骤,也是一种减少程序错误,证实软件质量的方法。然而,在软件测试中,如何选择测试用例,按什么样顺序设计测试路径一直是人们研究的重要问题。本书结合案例主要介绍了软件测试的流程、手段、方法。

软件也可按其性能高低划分为四类:超级健康软件、健康软件、亚健康软件、不健康软件。毫无疑问,我们做软件应该追求超级健康的软件,尽量不要开发出亚健康的软件。这些也是本书的目的。

本书共分 12 章,内容主要为:软件测试综述、软件测试过程与测试模型、黑盒测试与测试用例设计方法、白盒测试(静态分析与动态测试)的实用技术、单元测试(插桩技术、驱动模块等)方法、集成测试的方法、系统测试的方法、验收测试的方法、Web 应用系统测试方法、软件测试工具 LoadRunner 以及上机实训等内容,针对测试的过程中的面向过程编程的测试方法和面向对象编程的测试方法分别进行了介绍。书中给出了大量的测试用例。知识讲解通俗易懂,由浅入深。

本书课时是按照 64 课时(包括上机实践)安排的,读者可以根据具体情况对内容选择,其中“第 10 章”“第 11 章”可以作为选择内容。

本书特点有二:(1)知识面方面,遵照“够用”原则,紧扣高职高专的学生特点;(2)技术深度方面,遵照“适度”原则,使学生既达到了掌握软件测试技能的目的,又具备《全国计算机等级考试四级——软件测试工程师》考试能力。

另外,编者从多年来从事过的软件项目(军品——通信设备、银行软件系统研发、资金管理系统、校园一卡通系统、电子政务系统、ERP 系统、物流系统等)看,观察到一种现象,即“软件开发重编程、轻质量”,对软件的测试普遍不重视;很少有 SQA 这个角色,即使有也不知道该怎么做。本书出版不可能彻底改变这种局面,但是培养高质量软件测试工程师是本书的目的之一。

本书由刘竹林主编,韩莉和曾金发为副主编,其中韩莉对全书进行了认真阅读,提出了很多宝贵意见。

在这里首先感谢河北中信联信息技术有限公司的领导,是他们给了我机会,让我与公司的项目经理和软件测试工程师经常就关于软件测试问题进行探讨,因此本书的出

版有着他们的经验和智慧。

此外还要感谢石家庄信息工程职业学院！本书在出版前，讲稿已经（从2009年开始）在石家庄信息工程职业学院试用了2年，经过软件技术专业和信息管理专业的学生多次使用，对内容进行多次删、改、增，边使用边修改，力争使读者学习到实用的测试技术和测试方法。

最后感谢周光明老师和北京师范大学出版社所有为本书辛勤付出的编辑们！

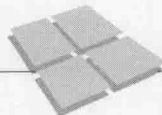
本书是软件技术专业、软件测试专业、多媒体开发专业、游戏开发专业、信息管理系统管理专业等（计算机偏软类专业）学生学习软件测试的必修书目，并且可作为计算机应用、电子信息、嵌入式开发等工程类学科的选修课书目，同时也适用于参加《全国计算机等级考试四级——软件测试工程师》考试的学生考试和计算机软件公司的软件测试人员和开发人员工作之用。

编者

2011年1月

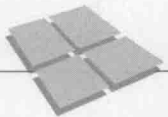
目 录

第 1 章 软件测试综述	(1)	本章小结	(21)
1.1 软件质量的定义及其特性	(1)	习题	(21)
1.2 软件质量保证与软件质量控制	(2)	第 3 章 黑盒测试与用例设计	(22)
1.3 软件测试的定义与范畴	(2)	3.1 软件测试方法分类	(22)
1.3.1 软件测试的定义	(2)	3.2 黑盒测试方法	(23)
1.3.2 软件测试的范畴	(4)	3.3 等价类划分法	(24)
1.4 软件测试的重要性	(4)	3.3.1 等价类划分法的思想	(24)
1.4.1 美国火星登陆探测器	(4)	3.3.2 等价类划分法和测试用例设计步骤	(24)
1.4.2 “爱国者”导弹防御系统	(5)	3.3.3 等价类划分的种类	(25)
1.4.3 英特尔奔腾芯片缺陷	(5)	3.4 边界值分析法	(26)
1.4.4 其他案例	(6)	3.4.1 边界值分析法的思想	(26)
1.5 软件测试的误区	(6)	3.4.2 边界值测试用例的设计方法	(26)
1.6 软件测试职业岗位分析	(7)	3.5 综合实例	(30)
本章小结	(8)	3.5.1 判断三角形形状	(30)
习题	(8)	3.5.2 成绩报告	(31)
第 2 章 软件测试过程与测试模型	(10)	3.5.3 测试程序的登录	(33)
2.1 软件开发过程	(10)	3.6 根据变量个数计算测试用例数	(34)
2.2 软件测试过程	(11)	本章小结	(36)
2.3 测试过程模型	(13)	习题	(36)
2.3.1 V 模型	(14)	第 4 章 静态白盒测试	(39)
2.3.2 W 模型	(14)	4.1 白盒测试概述	(39)
2.3.3 H 模型	(16)	4.1.1 白盒测试的定义	(39)
2.3.4 前置测试	(17)	4.1.2 白盒测试策略	(39)
2.4 测试理念	(18)	4.2 静态分析	(40)
		4.2.1 模块规范性测试	(40)
		4.2.2 模块逻辑性测试	(41)
		4.2.3 模块接口测试	(46)



4.2.4 模块局部数据结构测试	(47)	6.3.4 卡注册函数	(84)
4.2.5 模块全局数据结构测试	(47)	6.3.5 注销卡	(84)
4.3 同行评审	(47)	6.3.6 注册信息查询	(85)
4.3.1 同行评审的工作	(47)	6.3.7 按照书名查询	(86)
4.3.2 同行评审的问题	(48)	6.3.8 按照作者名字查询	(86)
本章小结	(49)	6.3.9 图书借阅管理函数	(87)
习题	(49)	6.3.10 还书函数	(88)
第5章 动态白盒测试	(51)	6.4 测试计划	(90)
5.1 画控制流图	(51)	6.4.1 概述	(90)
5.1.1 从程序代码导出控制流图	(51)	6.4.2 测试策略	(91)
5.1.2 从程序流程图导出控制流图	(54)	6.4.3 测试要点	(91)
5.2 动态白盒测试概述	(55)	6.4.4 测试通过/不通过标准	(92)
5.3 语句覆盖方法	(56)	6.4.5 计划进度	(92)
5.4 判定覆盖方法	(57)	6.4.6 人员与角色分配	(92)
5.5 条件覆盖	(60)	6.4.7 测试环境配置计划(硬件, 软件)	(93)
5.5.1 条件覆盖的定义	(60)	6.4.8 风险评估(尚未解决的问题和障碍)	(93)
5.5.2 例题	(61)	本章小结	(93)
5.6 判定/条件覆盖	(62)	习题	(93)
5.7 条件组合覆盖	(63)	第7章 单元测试	(94)
5.8 基本路径覆盖	(64)	7.1 单元测试的概念与必要性	(94)
5.8.1 圈复杂度及其计算	(64)	7.1.1 单元测试	(94)
5.8.2 圈复杂度和基本路径的关系	(67)	7.1.2 单元测试的必要性	(94)
5.8.3 基本路径测试步骤	(68)	7.2 单元测试的目标	(95)
5.8.4 例题	(68)	7.3 单元测试的任务	(95)
本章小结	(72)	7.4 单元测试的步骤	(96)
习题	(73)	7.4.1 代码审查	(97)
第6章 图书借阅管理系统项目介绍	(78)	7.4.2 桩模块和驱动模块	(98)
6.1 项目需求	(78)	7.4.3 编写驱动模块	(99)
6.2 项目软件设计	(79)	7.4.4 编写桩模块	(100)
6.3 项目代码结构	(81)	7.4.5 插桩技术	(100)
6.3.1 数据结构	(81)	7.4.6 编写测试用例	(102)
6.3.2 主函数代码	(82)	7.5 驱动测试思路	(103)
6.3.3 卡的操作与管理	(83)	7.5.1 驱动测试理论的引入	(103)
		7.5.2 测试驱动的原则	(104)

7.6 Junit 工具的使用	(104)	9.2.5 写系统测试总结与分析报告	(136)
7.6.1 被测试代码——连连看游戏	(104)	9.3 系统测试的内容	(136)
7.6.2 测试代码——创建 Junit 测试类	(113)	9.3.1 功能测试—— α 测试和 β 测试	(137)
7.6.3 运行测试类	(118)	9.3.2 软件攻击与系统接口测试	(137)
本章小结	(119)	9.3.3 恢复测试	(138)
习题	(119)	9.3.4 安全测试	(139)
第 8 章 集成测试	(121)	9.3.5 压力(负载)测试	(139)
8.1 集成测试基本概念	(121)	9.3.6 性能测试	(142)
8.2 集成测试方法	(121)	9.3.7 容量测试	(142)
8.2.1 非增值式集成测试	(122)	9.3.8 可靠性测试	(142)
8.2.2 增值式集成测试	(123)	9.3.9 兼容性测试	(143)
8.3 面向对象的集成测试	(126)	9.4 确认测试	(143)
8.3.1 面向对象程序的特点	(126)	9.4.1 确认测试的定义	(143)
8.3.2 面向对象的软件开发与面向过程的软件开发思路	(128)	9.4.2 确认测试的步骤	(144)
8.4 面向对象的集成测试	(129)	9.5 食堂消费系统测试案例	(144)
8.4.1 面向对象的集成测试流程	(130)	9.5.1 系统背景	(145)
8.4.2 撰写“集成测试计划”	(130)	9.5.2 食堂消费系统结构	(146)
8.4.3 设计集成测试用例	(130)	9.5.3 系统功能	(147)
8.4.4 集成测试用例评审	(130)	9.5.4 软件操作步骤与操作界面	(149)
本章小结	(131)	9.5.5 系统测试计划	(154)
习题	(131)	9.5.6 系统测试用例	(156)
第 9 章 系统测试与确认测试	(133)	本章小结	(158)
9.1 系统测试概念与目标	(133)	习题	(158)
9.2 系统测试步骤	(134)	第 10 章 Web 应用系统测试	(160)
9.2.1 写系统测试计划	(135)	10.1 Web 应用技术简介	(160)
9.2.2 设计系统测试用例	(135)	10.2 服务器端的测试	(160)
9.2.3 执行系统测试	(135)	10.2.1 性能测试	(160)
9.2.4 写缺陷报告	(136)	10.2.2 安全性测试	(162)
		10.2.3 负载和压力测试	(164)
		10.3 客户端的测试	(164)
		10.3.1 表单测试	(165)
		10.3.2 链接测试	(165)
		10.3.3 Cookie 测试	(165)
		10.3.4 用户界面测试	(165)



10.3.5	内容测试	(166)
10.3.6	兼容性测试	(166)
10.4	Web 应用系统的用例设计	(167)
10.5	基于场景的性能测试	(170)
	案例——图书网站下载	
	书籍	(170)
10.5.1	场景设计	(171)
10.5.2	用例设计	(171)
10.5.3	数据设计	(172)
	本章小结	(173)
	习题	(173)
第 11 章 性能测试工具 LoadRunner		
	操作指南	(174)
11.1	LoadRunner 的测试流程	(174)
	介绍	(174)
11.2	制订性能测试计划	(174)
	11.2.1 分析应用程序	(174)
	11.2.2 确定测试目标	(175)
11.3	创建负载测试	(175)
	11.3.1 创建 Web 脚本	(175)
	11.3.2 如何查看脚本	(180)
11.4	测试实例	(180)
	11.4.1 测试需求	(181)
	11.4.2 录制和编辑脚本	(181)
	11.4.3 回放脚本	(185)
	11.4.4 优化脚本	(186)
11.5	创建运行场景	(194)
	11.5.1 创建运行场景	(194)

11.5.2	在实际运行时如何查看脚本	(197)
11.6	运行负载测试	(197)
11.6.1	编译脚本	(198)
11.6.2	开始运行测试	(198)
11.7	分析以及监视场景	(200)
	本章小结	(202)
	习题	(203)
第 12 章	上机综合实训	(204)
附录 1	测试过程中的文档格式	(211)
附录 2	软件代码规范	(224)
全国计算机等级考试四级超级模拟		
试卷一软件测试工程师		(261)
全国计算机等级考试四级超级模拟		
试卷二四级软件测试工程师		(267)
全国计算机等级考试四级专家押题		
试卷一四级软件测试工程师		(272)
全国计算机等级考试四级专家押题		
试卷二四级软件测试工程师		(276)
全国计算机等级考试四级超级模拟		
试卷三四级软件测试工程师(部分题)		(282)
全国计算机等级考试四级专家押题		
试卷三四级软件测试工程师		(285)
2008 年 9 月全国计算机等级考试		
四级试卷四级软件测试工程师		(289)
参考文献		(294)

第1章 软件测试综述

学习目标

1. 正确理解软件质量的内涵。
2. 正确理解软件测试的内涵。
3. 了解软件测试职业岗位。

1.1 软件质量的定义及其特性

1. 什么是软件质量

ANSI/IEEE Std 729—1983 是这样定义的：“软件质量是与软件产品满足规定的和隐含的需求的能力有关的特征或特性的全体。”软件工程学者 M. J. Fisher 是这样定义的：“软件质量是描述计算机软件优秀程度的特性的组合。”

软件质量特性包括以下 11 个方面。

(1) 正确性(功能性)：正确性是指软件所实现的功能满足用户要求的程度，这是软件产品的首选质量特性。

衡量标准是：包括易追溯性、一致性、完备性。

(2) 易用性：易用性是指用户掌握软件操作所要付出的时间及努力程度。

衡量标准是：易操作性、培训、易交流性、输入和输出量、输入和输出速度。

(3) 完整性：保护软件不被未经同意的存储和使用的的能力。

衡量标准是：包括存储控制、存储审查。

(4) 效率：效率是指软件执行某项功能所需计算机资源(含时间)的有效程度。

衡量标准是：包括运行效率、存储效率。

(5) 可靠性：在给定的时间内，特定环境下软件正确运行的概率。或者说，是指在规定的条件和条件下，仍能维持其性能水准的程度。

衡量标准是：容错性、一致性、准确性、简洁性。

(6) 易维护性：易维护性指当环境改变或软件发生错误时，执行修改或恢复所做努力的程度。

衡量标准是：一致性、简洁性、简明性、模块性、自我描述性。

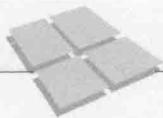
(7) 灵活性：当软件操作环境变化时，对软件作相应修改的难易程度。

衡量标准是：模块性、一般性、易扩展性、自我描述性。

(8) 易测试性：对软件测试以保证其无错误和满足其规约的难易程度。

衡量标准是：简洁性、模块性、检视、自我描述性。

(9) 易移植性：易移植性是指将一个程序从一个计算机系统或环境移到另一个计算机系统或环境的容易程度。



度量标准：(1)目标环境的个数；(2)与环境相关的模块数；(3)移植到目标环境的工作量；(4)移植目标环境的成本。

(10)易复用性：复用一个软件或其部分的难易程度。

衡量标准是：通用性、模块性、自我描述性、硬件独立性、软件独立性。

(11)互用性：将一个软件系统和其他软件系统组合在一起的难易程度。

衡量标准是：模块性、通信共同性、数据共同性。

2. 高质量软件具备的条件

可以总结出高质量软件应该具备的条件：

满足软件需求定义的功能和性能；

文档符合事先确定的软件开发标准；

软件的特点和属性遵守软件工程的目标与原则。

1.2 软件质量保证与软件质量控制

1. 软件质量保证(SQA)的定义

软件质量保证(Software Quality Assurance, SQA)是为保证产品和服务充分满足消费者要求的质量而进行的有计划、有组织的活动。它可以保证：

(1)依据组织经过文档化的开发计划和规程进行软件开发；

(2)这些计划和规程满足合同中有关质量的条款。

SQA的工作重点是对软件开发过程进行监督、管理和控制，它的实施是参照一定的质量标准、目标及各项软件流程、规范来监督和管理公司产品的质量。

很多公司有 SQA 这个角色，它和软件测试工程师的区别在于，件测试工程师的工作重点是对软件开发的成果进行检查、控制。

2. 软件质量控制

软件质量控制(Software Quality Control, SQC)的职责是验证产品的正确性，当发现与设计不一致的时候进行纠正。而质量保证充当支持执行全面质量管理角色。

1.3 软件测试的定义与范畴

软件测试是伴随着软件的产生而产生的。早期的软件开发过程中，软件规模都很小、复杂程度低，软件开发的过程混乱无序，测试的含义比较狭窄，开发人员经常将测试等同于“调试”。

软件测试属于软件质量控制的范畴，是软件质量保证的关键和重要内容。

1.3.1 软件测试的定义

随着人们对软件质量认识和重视的程度，不同的阶段软件测试的定义也不同。可见，人们对软件质量的重视程度在不断加深。

1972 年，软件测试的宗师 Bill Hetzel 博士在他的论著《The Complete Guide to Software Testing》(《软件测试完全指导书》)中指出：“软件测试就是建立一种信心，认

为程序能够按预期的设想运行。”

1979年,软件专家 Glenford J. Myers 博士在他的论著《The Art of Software Testing》(《软件测试的艺术》)中这样说:“测试是为发现错误而执行的一个程序或者系统的过程。”

1983年,IEEE这样给出了软件测试的定义:“软件测试是使用人工或自动的手段来运行或测定某个软件系统的过程,其目的在于检验它是否满足规定的需求或弄清预期结果与实际结果之间的差别。”

这个定义明确了两项内容:

1. 软件测试的目的

软件测试是以查找软件缺陷为手段而达到提高软件质量为目的的。或者说,软件测试的目的是以较少的用例、时间和人力找出软件产品潜在的各种错误和缺陷,以确保软件的质量。

这里提到的“软件缺陷”,包括哪些内容呢?下面列举了软件的缺陷。

- (1)谬误,失败,缺点:没有达到目标,但有危险。
- (2)异常,毛病,偏差:未预料的情况,不是全部失败。
- (3)问题,错误。
- (4)需求功能没有完全达到。
- (5)实现了需求中没有的功能。

那么,“错误”、“失效”、“故障”、“缺陷”有什么区别呢?在这里,我们有必要把这几个词语概念搞清楚。

错误:是指人们期望的和系统实际具有的状态或行为之间的偏差。软件错误包括由系统分析或者程序设计而产生的全部错误。

失效:是指在软件运行期间出现的错误。它是因软件中存在的错误引起的、并在执行期间动态表现出来的和实际不同的状态和行为。可以看出,失效只是错误的一部分。

故障:是指软件中的物理缺陷。故障可能引起软件的失效,也可能不引起软件的失效,人们总是通过修改软件中的故障来提高软件的可靠性。故障也是错误的一部分。

缺陷:凡是没有按照要求所进行的工作统称为缺陷。缺陷泛指系统中的一切错误。显然,缺陷是比错误范围更大的概念。例如:程序的冗余功能或模块、难以理解的代码、不易维护的程序、文档不完全等,这些可能并不能使软件失败。

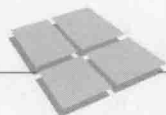
上面这4个概念在有些文献中经常混用,我们在实际使用时应区分它们之间的层次和范围。

2. 软件测试的准绳

查找软件缺陷的准绳是软件的需求说明书。就像法院定罪的标准就是法律一样,软件需求说明书就是软件测试工程师的“法律”。

为什么缺陷很难被找出?原因如下:

- (1)看不到;
- (2)看到但是抓不到;
- (3)典型的缺陷类型:需求解释有错误、用户定义错了需求、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、测试错误、问题修改不正确、正确的结果是由于其他的缺陷产生的。

因此,也可以通俗地说,软件测试是通过技术、流程、工具、人员以及管理手段,检测软件文档、软件中间产品和最终产品,查找和报告软件缺陷、错误以及隐患的软件研发环节之一,通过跟踪缺陷、错误及隐患的修正过程,以确保软件产品、中间产品和文档符合软件工程过程需求和用户的最终需求。

1.3.2 软件测试的范畴

软件测试属于软件工程范畴,研究范围有:

- (1)测试的基本概念和定义:包括测试术语、理论基础等;
- (2)测试层次:如单元/集成/系统测试;或验收/回归测试等;
- (3)测试技术:如黑盒/白盒测试;或基于错误/风险测试等;
- (4)测试度量:包括对于测试工作和被测测试程序的度量;
- (5)测试管理过程:包括宏观管理层面和具体的测试活动。

1.4 软件测试的重要性

软件测试是软件质量控制的手段,它的重要性在于,如果测试不充分,那么这些问题会潜伏在软件中,等到用户发现以后,再由开发人员进行维护,改错误的费用一般是开发阶段的 40 倍到 60 倍。

在软件测试原则中强调测试人员要在软件开发的早期进行测试,如需求分析阶段就应该介入,问题发现的越早越好。发现缺陷后要尽快修复缺陷,因为随着产品开发工作的进行,一个小错误会扩散成大错误,为了修改后期的错误所做的工作要大得多,即越到后来往前返工也越长,缺陷发现或解决得越迟,成本就会越高。修正错误的代价不是随时间线性增长,而几乎是呈指数增长的。

下面介绍几个重大的软件测试事故。

1.4.1 美国火星登陆探测器

火星极区着陆器是 1999 年 1 月 3 日发射的,在 1999 年 12 月 3 日进入大气层时失去通信联系。

按照计划,应该在美国东部时间约下午 3 时 15 分,探测器应在火星表面缓缓着陆。当飞船离地面 1800m 时,它将丢弃降落伞,点燃登陆推进器,在余下的高度缓缓降落地面。3 时 39 分应开始向地面发回第一次无线电信号,报告着陆成功。

但是,美国宇航局为了省钱,简化了关闭推进器的装置。为了替代其他太空船上使用的贵重雷达,在飞船的脚上装了一个廉价



图 1-1 火星登陆探测器

的触点开关,在计算机中设置一个数据位来关掉燃料。质量管理小组在事后的测试中发现,当飞船的脚迅速摆开准备着陆时,机械震动在大多数情况下也会触发着地开关,设置错误的数据位。设想飞船开始着陆时,计算机极有可能关闭推进器,而火星登陆飞船下坠 1800m 之后冲向地面,必然会撞成碎片。飞船的脚不“着地”,引擎就会点火。

为什么会出现这样的结果?原因很简单。登陆飞船经过了 2 个小组测试。

其中一个小组测试飞船的脚落地过程,但从没有检查那个关键的数据位,因为那不是这个小组负责的范围;

另一个小组测试着陆过程的其他部分,但这个小组总是在开始测试之前重置计算机、清除数据位。双方本身的工作都没什么问题,就是没有合在一起测试,其接口没有被测,而问题就在这里,后一个小组没有注意到数据位已经被错误设定。仅仅由于 2 个测试小组单独进行测试,没有进行很好沟通,缺少一个集成测试的阶段,结果导致 1999 年美国宇航局的火星基地登陆飞船在试图登陆火星表面时突然坠毁失踪。质量管理小组观测到故障,并认定出现误动作的原因极可能是某一个数据位被意外更改。

1.4.2 “爱国者”导弹防御系统

1991 年美国“爱国者”导弹防御系统首次被用在第一次海湾战争对抗伊拉克“飞毛腿”导弹的防御作战中,总体上看效果不错,赢得各界的赞誉。但它还是有几次没有成功拦截伊拉克“飞毛腿”导弹,其中一枚在科威特的多哈爆炸的飞毛腿导弹造成 28 名美国士兵死亡。

问题分析

拦截失败的症结在于一个软件缺陷,当“爱国者”导弹防御系统的时钟累计运行超过 14h 后,系统的跟踪系统就不准确。在多哈袭击战中,“爱国者”导弹防御系统运行时间已经累计超过 100 多个小时,显然那时系统的跟踪系统已经很不准确,从而造成这种结果。

1.4.3 英特尔奔腾芯片缺陷

1994 年,在计算机的“计算器”程序中输入以下算式: $(4195835/3145727) \times 3145727 - 4195835$ 。如果答案是 0,就说明该计算机浮点运算没问题。如果答案不是 0,就表示计算机的浮点除法存在缺陷。

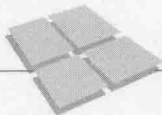
这个缺陷是美国弗吉尼亚州 Lynchburg 大学的 Thomas R. Nicely 博士发现的。他在奔腾 PC 上做除法实验时记录了一个没想到的结果。他把发现的问题放到因特网上,随后引发了一场风暴,成千上万的人发现了同样的问题,以及其他得出错误结果的情形。万幸的是,这种情况很少见,仅仅在进行精度要求很高的数学、科学和工程计算中才导致错误。大多数进行财会管理和商务应用的用户根本不会遇到此类问题。

问题分析

这个事件不仅说明软件缺陷所带来的问题,更重要的是说明对待软件缺陷的态度。

(1) 英特尔的软件测试工程师在芯片发布之前进行内部测试时已经发现了这个问题,但管理层认为这没有严重到一定要修正,甚至需要公布这个问题。

(2) 当软件缺陷被发现时,英特尔通过新闻发布和公开声明试图掩饰这个问题的严重性。



(3)受到压力时,英特尔承诺更换有问题的芯片,但要求用户必须证明自己受到软件缺陷的影响。结果舆论大哗。因特网新闻组充斥着愤怒的客户要求英特尔解决问题的呼声。得到这个教训之后,英特尔在网站上报告已发现的问题,并认真对待客户在因特网新闻组上的反馈意见。

在英特尔公司历史上,这是第二次由于芯片缺陷导致大规模产品回收。由于芯片的浮点运算区出现数值错误,该公司被迫更换成千上万片有缺陷的奔腾芯片。英特尔公司为此付出了5亿美元的代价。

1.4.4 其他案例

下面这些可能是由于软件的错误造成的重大事故:

(1)在伦敦希思罗国际机场,平均每3 min就有一架飞机起落,如果调度飞机场着陆次序的软件发生故障,将会造成飞机相撞的灾难。

(2)波音747等大型客机都使用计算机自动导航系统,如果它发生故障,将会造成机毁人亡的事故。

(3)在西方国家,铁路信号系统已计算机化。英国已计划在不久的将来实现铁路信号和扳道的全部自动化。铁路交通的安全性将进一步依赖计算机系统的正常运行。

(4)在巴黎地铁系统中,由于使用计算机进行有效的调度和安全控制,火车的行驶速度得以加快,火车之间的间隔得以缩短。

(5)欧洲航空航天局发射的阿丽亚那火箭,由于软件的错误致使发射失败,所携带的卫星报废,损失巨大。

下面这些是已经造成的危害:

(1)中国的×××卫星,由于软件的缺陷,在运行不到一年后就失去作用。根据事后的分析表明,仅仅是由于一个非常简单的软件错误而使耗资巨大的卫星报废。

(2)由于控制放射性治疗设备的软件错误,在加拿大已经造成了多起癌症病人因受到过量放射性辐射而死亡的事故。

(3)在英国,由于放射性治疗设备中计算辐射量的软件错误,发生了癌症病人因放射性辐射太低而得不到适当治疗的事故。

(4)在伦敦,救护车调度软件刚刚投入使用几小时就发生故障,造成急诊病人延误达十几小时。

(5)飞机自动控制软件的错误造成了飞机在特技飞行表演时坠毁。

(6)自动行李处理系统的一个很小的软件缺陷使丹佛国际机场飞机受阻,到达的行李空等九个月。

1.5 软件测试的误区

随着软件产业工业化、模块化的发展,在软件开发过程中软件测试人员的重要性也显得越来越突出。在国外,很多著名企业早已对软件测试工作十分重视。比如著名的微软公司,其软件测试人员与开发人员的比例已经达到2:1。

软件测试对于一个软件开发项目的成功与否具有十分重要的意义。但是在实际的

项目开发与管理中仍然存在很多管理上或者技术上的误区：

- (1)期望用测试自动化代替大部分人工劳动；
- (2)忽视软件测试人员在需求阶段的项目参与；
- (3)软件测试是技术要求不高的岗位；
- (4)测试与调试类似，是对调试的一种补充；
- (5)测试是比编程容易的工作。

1.6 软件测试职业岗位分析

随着软件业的迅猛发展，软件产品的质量控制与管理正逐渐成为软件企业生存与发展的关键。为保证软件出厂时的“健康状态”，几乎所有的IT企业在软件产品发布前，都需要进行大量的软件测试，以对软件质量进行严格控制，从而引发软件测试人才走俏。

1. 工作起点高

小公司对软件质量的重视程度没有大公司高。一般较大规模的软件企业需要大量测试工程师，所以只要能进入这些企业就是比较正规的企业。

2. 发展空间大

刚进入企业后，随着业务越来越熟练，有可能升为软件测试主管、项目主管、行业专家、自动化测试专家、QA、需求分析师、客户服务/支持等岗位。

3. 职业寿命长，越老越吃香

这个岗位是针对程序员来讲的，软件测试工程师经常使用专业测试工具，经验价值会逐步上升，对公司的业务也会越来越精、越“资深”。另外这个岗位对年龄没有限制。工作更稳定，不用频繁“跳”

- (1)工作时间越长，对业务需求就了解更深，这个岗位就很难有人能够替代；
- (2)由于进入的是大公司，因此制度全面，管理规范；
- (3)福利好。企业规模大，必然有发展前途，效益好；
- (4)有归属感。

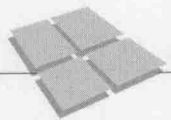
4. 薪水待遇好

真正资深的测试工程师凤毛麟角，所以薪酬空间就大。

5. 没有性别歧视

这个工作岗位不像程序员，公司对招聘的程序员实际上倾向于男性。

软件测试岗位对于性别没有特别要求，主要是要求能够有耐心和责任心的人去完成，所以女性恰恰具有这个优势。



本章小结

本章内容是软件测试综述,主要讲述软件测试的所涉及的一些基本知识和基本概念。

软件质量是软件产品的特性可以满足用户的功能、性能需求的能力。具体地说,软件质量是软件符合产品的功能需求和性能需求、规定的开发标准的程度。软件质量在软件开发过程中主要是由软件质量控制(SQC)和软件质量保证(SQA)来把关的。

软件测试是软件投入运行前,对软件需求分析、设计规格说明和编码实现的最终审查,是软件质量保证的关键步骤。可以说,一个好的、健康的软件的完成,必然有软件测试的伴随。如果不重视测试,也就是不重视软件的质量,那么很可能对软件用户造成不可想象的后果。

本章的最后两节讨论了软件测试中的误区以及软件测试职业的特点,软件测试在国内很多软件公司还没有充分认识到它的重要性,但相信在不久的将来会有很好的发展,这对于我们来说既是机遇又是挑战。

>>> 习 题

1. 写小论文:软件测试职业岗位趋势分析。

2. 调查:软件公司中的软件测试工具应用情况和职业需求情况。

3. 选择题(四选一)

(1)Myers 在 1979 年提出了一个重要观点,软件测试的目的是()。

A)证明程序正确

B)证明程序错误

C)改正程序错误

D)验证程序无错误

(2)下列哪项不是软件缺陷()。

A)软件没有实现需求说明书中的功能

B)软件出现了需求说明书中不应该出现的功能

C)软件界面不友好、可读性差、易用性差等

D)软件实现需求说明书中需求的功能,但是由于性能限制而未考虑可移植性问题

(3)软件生存周期过程中,修改错误代价最大的阶段是()。

A)需求阶段

B)设计阶段

C)编程阶段

D)发布运行阶段

(4)下列能表达程序未按照预期运行,但不会导致整体失效的是()。

A)故障

B)异常

C)缺点

D)失效

(5)功能或特性没有实现,主要功能部分丧失,次要功能完全丧失,或致命的错误声明,这属于软件缺陷级别中的()。

A)致命的缺陷

B)严重的缺陷