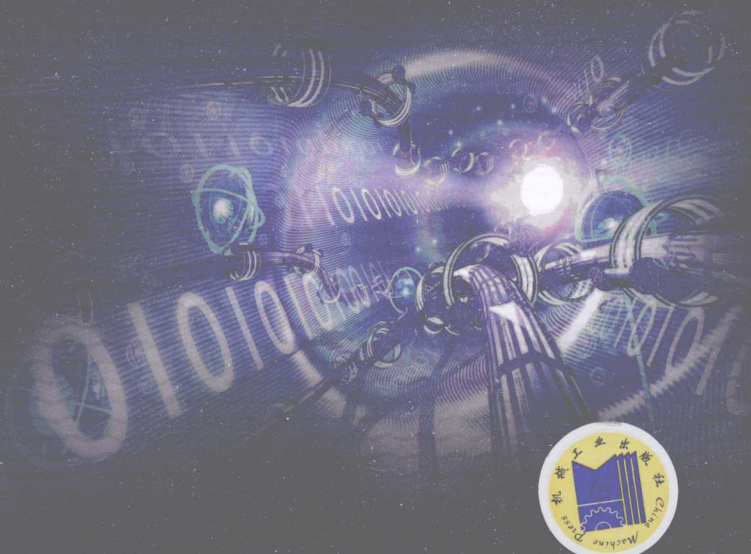
免费提供电子教案

高等院校规划教材
软件工程系列

软件测试实践教程

路晓丽　董云卫　编著



机械工业出版社
CHINA MACHINE PRESS

高等院校规划教材·软件工程系列

软件测试实践教程

路晓丽　董云卫　编著

机械工业出版社

本书全面、系统地阐述了软件测试的基础知识和应用技术，是一本非常实用的软件测试教材。全书共8章，第1章介绍了软件测试的基础知识，包括软件测试的概念、软件测试的分类和软件测试的背景等；第2章介绍了白盒测试的测试用例设计方法和典型案例；第3章介绍了黑盒测试的测试用例设计方法和典型案例；第4章介绍了一种新的测试方法——蜕变测试方法和典型案例；第5章介绍了测试流程和测试文档的基本知识，列举了测试计划、测试说明和测试报告等测试文档的例子；第6章介绍了功能测试工具WinRunner以及使用WinRunner的最佳实践；第7章介绍了压力测试工具LoadRunner以及使用LoadRunner的最佳实践；第8章介绍了测试管理工具TestDirector以及使用TestDirector的最佳实践。

本书可以作为大学本科软件测试课程的教材，也可以作为软件测试人员、软件项目经理和需要了解软件测试的各级管理人员的参考书。

图书在版编目（CIP）数据

软件测试实践教程/路晓丽，董云卫编著. —北京：机械工业出版社，2010.6
（高等院校规划教材·软件工程系列）
ISBN 978-7-111-30298-8

Ⅰ. ①软… Ⅱ. ①路… ②董… Ⅲ. ①软件—测试—高等学校—教材
Ⅳ. ①TP311.5

中国版本图书馆CIP数据核字（2010）第059160号

机械工业出版社（北京市百万庄大街22号 邮政编码100037）
策划编辑：唐德凯
责任编辑：唐德凯
北京外文印刷厂印刷
2010年6月第1版第1次印刷
184mm×260mm·14印张·342千字
0001—3000册
标准书号：ISBN 978-7-111-30298-8
定价：25.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务	网络服务
社服务中心：（010）88361066	门户网：http://www.cmpbook.com
销 售 一 部：（010）68326294	
销 售 二 部：（010）88379649	教材网：http://www.cmpedu.com
读者服务部：（010）68993821	封面无防伪标均为盗版

出 版 说 明

计算机技术的发展极大地促进了现代科学技术的发展，明显地加快了社会发展的进程。因此，各国都非常重视计算机教育。

近年来，随着我国信息化建设的全面推进和高等教育的蓬勃发展，高等院校的计算机教育模式也在不断改革，计算机学科的课程体系和教学内容趋于更加科学和合理，计算机教材建设逐渐成熟。在“十五”期间，机械工业出版社组织出版了大量计算机教材，包括“21 世纪高等院校计算机教材系列”、“21 世纪重点大学规划教材”、“高等院校计算机科学与技术‘十五’规划教材”、“21 世纪高等院校应用型规划教材”等，均取得了可喜成果，其中多个品种的教材被评为国家级、省部级的精品教材。

为了进一步满足计算机教育的需求，机械工业出版社策划开发了“高等院校规划教材”。这套教材是在总结我社以往计算机教材出版经验的基础上策划的，同时借鉴了其他出版社同类教材的优点，对我社已有的计算机教材资源进行整合，旨在大幅提高教材质量。我们邀请多所高校的计算机专家、教师及教务部门针对此次计算机教材建设进行了充分的研讨，达成了许多共识，并由此形成了“高等院校规划教材”的体系架构与编写原则，以保证本套教材与各高等院校的办学层次、学科设置和人才培养模式等相匹配，满足其计算机教学的需要。

本套教材包括计算机科学与技术、软件工程、网络工程、信息管理与信息系统、计算机应用技术以及计算机基础教育等系列。其中，计算机科学与技术系列、软件工程系列、网络工程系列和信息管理与信息系统系列是针对高校相应专业方向的课程设置而组织编写的，体系完整，讲解透彻；计算机应用技术系列是针对计算机应用类课程而组织编写的，着重培养学生利用计算机技术解决实际问题的能力；计算机基础教育系列是为大学公共基础课层面的计算机基础教学而设计的，采用通俗易懂的方法讲解计算机的基础理论、常用技术及应用。

本套教材的内容源自致力于教学与科研一线的骨干教师与资深专家的实践经验和研究成果，融合了先进的教学理念，涵盖了计算机领域的核心理论和最新的应用技术，真正在教材体系、内容和方法上做到了创新。同时本套教材根据实际需要配有电子教案、实验指导或多媒体光盘等教学资源，实现了教材的“立体化”建设。本套教材将随着计算机技术的进步和计算机应用领域的扩展而及时改版，并及时吸纳新兴课程和特色课程的教材。我们将努力把这套教材打造成为国家级或省部级精品教材，为高等院校的计算机教育提供更好的服务。

对于本套教材的组织出版工作，希望计算机教育界的专家和老师能提出宝贵的意见和建议。衷心感谢计算机教育工作者和广大读者的支持与帮助！

机械工业出版社

前　言

软件测试是软件开发过程中的一个重要环节，是保证软件质量的关键技术之一。软件测试工作开展的好坏，直接决定着软件产品质量的优劣。在实际软件开发过程中，国际化大型软件公司在软件测试上投入了大量的人力和物力，以确保软件质量能够满足软件用户的要求，以赢得软件产品质量的美誉度。近年来，随着国内软件产业的蓬勃发展，专业化软件产品及服务软件企业的出现和产业规模的扩大，软件测试的重要性日益凸显，软件测试行业在国内也渐渐兴起，并逐步扩大。2007 年第四季度以后，我国 IT 产业对软件测试人才的需求呈现进一步升级的态势。国家信息产业部提供的数据显示，目前软件测试人才的缺口已经达到 40～50 万，软件测试人才正在成为我国 IT 行业最紧缺的人才之一，软件测试能力不足已成为制约我国软件产业发展的重要因素。

虽然越来越多的大学毕业生和其他行业人员有志于从事软件测试行业，却苦于缺乏系统的软件测试理论知识和实用测试技术而被用人企业拒之门外，进一步加剧了软件测试业务的人才供需矛盾。本书正是针对该问题，总结了作者多年从事软件测试课程教学的经验和在软件公司从事软件测试的工作经历，在参阅了大量国内外相关文献资料，进行总结和充实后，完成了软件测试使用技术的编写工作，试图为软件人才的培养和锻炼提供帮助和指导。

本书全面、系统地阐述了软件测试的基础知识和应用技术，是一本非常实用的软件测试技术参考书。全书共分 8 章，其中，第 1 章介绍了软件测试的基础知识，包括软件测试的概念、软件测试的分类和软件测试的背景等；第 2 章介绍了白盒测试的测试用例设计方法和典型案例；第 3 章介绍了黑盒测试的测试用例设计方法和典型案例；第 4 章介绍了一种嵌入式软件测试的新方法——蜕变测试技术和典型案例；第 5 章介绍了测试流程和测试文档的基本知识，列举了测试计划、测试说明和测试报告等测试文档的例子；第 6 章介绍了功能测试工具 WinRunner 以及使用 WinRunner 的最佳实践；第 7 章介绍了压力测试工具 LoadRunner 以及使用 LoadRunner 的最佳实践；第 8 章介绍了测试管理工具 TestDirector 以及使用 TestDirector 的最佳实践。书中使用了大量的案例，由浅入深，理论与实践相结合。

软件测试作为软件工程方面一门专业学科，从软件工程中演化出来，贯穿于整个软件生命周期的验证和确认（V&V）活动，并随着软件开发技术的更新而不断发展。为了较好地阅读和理解本书的内容，掌握一门软件开发高级语言，了解一定的软件开发知识是十分必要的。本书作为高等院校计算机软件测试技术的教材，课程建议授课学时为 54 小时，实验学时 18 小时（折合 36 机时），并要求先修软件工程和一门高级语言，例如，C++或者 Java 语言。

本书工作的开展得到了 863 课题“构件化嵌入式软件测试方法及其工具研究”（课题号：2008AA01Z142）和西北大学“211 工程”的支持，主要由路晓丽和董云卫编写，同时课题组

其他成员李凤美、雒超等为本书的编写收集与整理了一些外文资料，西安 863 孵化中心为本书编写提供了一些测试文档的案例，在此对西安 863 孵化中心主任楼文晓高级工程师、西北大学席恒教授、西北工业大学计算机学院周兴社教授等为本书编写提供指导和帮助的人士一并表达我们的谢意。

由于作者水平有限，书中难免存在不妥之处，敬请广大读者原谅，并提出宝贵意见。

作者

目　录

出版说明

前言

第 1 章　软件测试概述 …… 1

1.1　软件危机 …… 1

1.1.1　什么是软件危机 …… 1

1.1.2　软件危机的内在原因 …… 1

1.1.3　软件工程和软件危机的解决 …… 2

1.2　软件测试的基本知识 …… 2

1.2.1　软件测试的背景和意义 …… 2

1.2.2　软件测试的定义 …… 4

1.2.3　软件测试的分类 …… 5

1.2.4　软件测试的过程 …… 13

1.2.5　软件测试与软件开发过程的关系 …… 13

1.3　正确认识软件测试 …… 15

1.4　软件测试职业 …… 17

1.4.1　软件测试职业和职位 …… 17

1.4.2　软件测试职业素质 …… 17

1.4.3　软件测试人才现状 …… 17

1.5　习题 …… 18

第 2 章　白盒测试 …… 19

2.1　白盒测试概述 …… 19

2.2　白盒测试的测试用例设计方法 …… 20

2.3　白盒测试的典型案例 …… 20

2.3.1　逻辑覆盖法典型案例 …… 20

2.3.2　路径覆盖法典型案例 …… 27

2.4　白盒测试的工具 …… 32

2.4.1　白盒测试工具的分类 …… 32

2.4.2　开放源码的 Junit 的使用 …… 33

2.5　习题 …… 34

第 3 章　黑盒测试 …… 35

3.1　黑盒测试概述 …… 35

3.2　黑盒测试的主要测试用例设计方法 …… 35

3.2.1　等价类划分 …… 35

3.2.2　边界值分析 …… 36

3.2.3　因果图 …… 37

3.2.4　判定表驱动测试 …… 38

3.2.5　正交实验设计法 …… 39

3.2.6　用例场景法 …… 40

3.3　黑盒测试的典型案例 …… 41

3.3.1　使用等价类划分法设计测试用例 … 41

3.3.2　使用边界值分析法设计测试用例 … 48

3.3.3　使用因果图法设计测试用例 …… 50

3.3.4　使用判定表法设计测试用例 …… 53

3.3.5　使用正交表法设计测试用例 …… 55

3.3.6　使用用例场景法设计测试用例 … 56

3.4　习题 …… 58

第 4 章　蜕变测试 …… 59

4.1　蜕变测试概述 …… 59

4.2　蜕变测试的典型案例 …… 60

4.2.1　数学应用案例 …… 60

4.2.2　图论应用案例 …… 61

4.2.3　搜索程序应用案例 …… 62

4.2.4　编译器应用案例 …… 62

4.2.5　图形图像应用案例 …… 63

4.2.6　其他应用案例 …… 64

4.3　习题 …… 64

第 5 章　测试流程和测试文档 …… 65

5.1　测试流程 …… 65

5.2　测试文档概述 …… 66

5.3　编写测试文档 …… 67

5.3.1　测试计划的内容和实例 …… 67

5.3.2　测试说明的内容和实例 …… 83

5.3.3　测试报告的内容和实例 …… 94

5.4　习题 …… 108

第 6 章 WinRunner 功能测试工具的运用 …… 109
6.1 WinRunner 介绍 …… 109
6.1.1 WinRunner 的测试模式 …… 109
6.1.2 WinRunner 的测试过程 …… 109
6.1.3 WinRunner 的样本软件 …… 110
6.2 WinRunner 使用概述 …… 110
6.2.1 WinRunner 主窗口 …… 110
6.2.2 WinRunner 测试窗口 …… 112
6.2.3 加载 WinRunner 插件窗口 …… 112
6.3 WinRunner 如何识别 GUI 对象 …… 113
6.3.1 识别 GUI 对象 …… 113
6.3.2 使用 GUI Spy 查看 GUI 对象的属性 …… 114
6.4 WinRunner 如何学习被测软件的 GUI 对象 …… 116
6.4.1 两种 GUI map 的工作模式 …… 116
6.4.2 WinRunner 学习 GUI 对象的方式 …… 117
6.5 录制测试脚本 …… 120
6.5.1 两种录制模式 …… 120
6.5.2 如何录制测试脚本 …… 121
6.5.3 如何阅读测试脚本 …… 123
6.5.4 执行测试脚本并分析结果 …… 124
6.6 同步点 …… 126
6.6.1 何时使用同步点 …… 126
6.6.2 如何建立同步点 …… 126
6.6.3 执行测试并分析结果 …… 129
6.7 GUI 对象检查点 …… 130
6.7.1 检查 GUI 对象 …… 130
6.7.2 插入 GUI 对象检查点 …… 130
6.7.3 执行测试并分析结果 …… 132
6.8 图像检查点 …… 135
6.8.1 检查应用程序的图像 …… 135
6.8.2 建立图像检查点 …… 135
6.8.3 检视结果 …… 137
6.9 使用 TSL 修改脚本 …… 138
6.9.1 录制基本脚本 …… 138
6.9.2 使用函数生成器在测试脚本中加入函数 …… 139
6.9.3 在测试脚本中加入判断式 …… 140
6.9.4 执行测试脚本 …… 140
6.10 建立数据驱动脚本 …… 141
6.10.1 数据驱动脚本概述 …… 141
6.10.2 将测试脚本转换为数据驱动测试脚本 …… 142
6.10.3 使用 regular expression 调整测试脚本 …… 145
6.10.4 执行脚本并分析结果 …… 145
6.11 文字检查点 …… 146
6.11.1 文字检查点概述 …… 146
6.11.2 建立文字检查点 …… 147
6.11.3 检查文字 …… 150
6.11.4 执行测试脚本 …… 151
6.12 建立批测试 …… 151
6.12.1 批测试概述 …… 151
6.12.2 建立批测试 …… 152
6.12.3 执行批测试 …… 152
6.12.4 分析测试结果 …… 153
6.13 维护测试脚本 …… 154
6.13.1 在 GUI Map 中改变 GUI 对象 …… 154
6.13.2 新增 GUI 对象到 GUI Map …… 156
6.13.3 使用 Run Wizard 自动更新 GUI Map …… 157
6.14 习题 …… 157
第 7 章 LoadRunner 负载测试工具的运用 …… 158
7.1 LoadRunner 负载测试工具概述 …… 158
7.1.1 LoadRunner 的重要组件 …… 158
7.1.2 LoadRunner 的例子应用程序 …… 159
7.2 LoadRunner 的测试过程 …… 160
7.3 制订负载测试计划 …… 160
7.3.1 分析应用程序 …… 160
7.3.2 确定测试目标 …… 161
7.3.3 计划怎样执行 LoadRunner …… 161

7.4 开发测试脚本 …… 161
7.4.1 录制基本的用户脚本 …… 161
7.4.2 完善测试脚本 …… 163
7.4.3 运行参数设定 …… 170
7.4.4 单机运行测试脚本 …… 171
7.5 创建运行场景 …… 172
7.5.1 选择场景类型 Manual Scenario …… 173
7.5.2 选择场景类型 Manual Scenario with Percentage Mode …… 177
7.5.3 选择场景类型 Goal-Oriented Scenario …… 177
7.6 分析以及监视场景 …… 179
7.7 分析实时监视图表 …… 181
7.8 利用 Analysis 分析结果 …… 182
7.8.1 分析事务的响应时间 …… 182
7.8.2 分解页面 …… 183
7.8.3 确定 WebServer 的问题 …… 184
7.9 习题 …… 184
第 8 章 TestDirector 测试管理工具的运用 …… 185
8.1 TestDirector 概述 …… 185
8.1.1 TestDirector 简介 …… 185
8.1.2 TestDirector 的窗口 …… 185
8.1.3 TestDirector 的测试管理过程 …… 186
8.2 需求定义 …… 187
8.2.1 定义测试范围 …… 188
8.2.2 创建测试需求大纲 …… 188
8.2.3 定义需求 …… 189
8.2.4 分析需求定义 …… 189
8.2.5 TestDirector 需求模块 …… 190
8.3 计划测试 …… 198
8.3.1 定义测试策略 …… 198
8.3.2 定义测试主题 …… 198
8.3.3 定义测试 …… 199
8.3.4 建立测试覆盖 …… 201
8.3.5 设计测试步骤 …… 201
8.3.6 自动测试 …… 202
8.3.7 分析测试计划 …… 202
8.4 执行测试 …… 203
8.4.1 建立测试集合 …… 203
8.4.2 测试执行表 …… 204
8.4.3 执行测试 …… 205
8.4.4 分析测试结果 …… 207
8.5 跟踪缺陷 …… 209
8.5.1 添加缺陷 …… 209
8.5.2 检查新缺陷 …… 211
8.5.3 修改开放缺陷 …… 212
8.5.4 验证缺陷 …… 212
8.5.5 分析缺陷数据 …… 212
8.6 习题 …… 212
参考文献 …… 213

第 1 章　软件测试概述

软件测试在软件生存期中占有非常突出的位置，是保证软件质量的重要手段。本章主要介绍软件测试技术的基础知识，包括软件危机的内在原因及软件工程的解决方法、软件测试的背景和意义、软件测试的概念、软件测试的过程、软件测试的职业和职位等基础理论知识，这是学习本书后续内容的必要准备。

1.1　软件危机

在 20 世纪 60 年代，由于软件的规模越来越大，软件变得越来越复杂，在软件的开发过程中出现了软件危机。下边分别介绍软件危机的概念、软件危机的内在原因和软件危机的解决方法。

1.1.1　什么是软件危机

随着计算机技术的飞速发展，计算机被广泛地应用。为了适应在广泛应用情况下出现的各种各样的复杂问题，需要严格地保证软件系统的质量。但是，软件系统的开发需要投入大量的人力、物力和财力，相对于硬件系统投资而言，软件投资所占比例越来越大。同时，开发软件本质上是一个“思考”的工程，开发人员有各自的编程习惯和思维方式，可凭个人的爱好进行工作，没有统一的标准可以遵循，因此系统的质量难以得到保证。大约在 20 世纪 60 年代，面对愈来愈复杂的大型软件系统开发，出现了“软件危机”，主要表现为以下几个方面。

1）软件项目经常无法按期完成，超出经费预算，软件质量难以控制。

2）开发过程管理不规范，约定不严密，文档书写不完整，使得软件维护费用高，有些系统甚至无法进行修改。

3）缺乏严密有效的质量检测手段，交付给用户的软件质量差，在运行中出现许多问题，甚至带来严重的后果。

4）系统更新换代难度大。

这些问题出现在很多具体的软件开发项目上，最为突出的便是美国 IBM 公司在 1963 年～1966 年开发的 IBM 360 机操作系统。这一项目在开发期间每年花费 5000 万美元，总共投入的工作量为 5000 人/年，参加工作人数最多时有 1000 人，总共写出了 100 万行源代码。这么大的开销，却拿不到开发成果，这是一个失败的记录。项目负责人 F.P.Brooks 事后总结了他在开发过程中的沉痛教训，写成《神秘的人-月》一书，这个反映软件危机的典型事例成为软件技术发展过程中一个历史性标志。

1.1.2　软件危机的内在原因

在软件系统的开发过程中，软件缺陷的积累与放大效应是导致软件危机的最主要原因，可谓失之毫厘，差之千里。在此情况下，反复无常的修改导致软件开发效率严重低下，带有缺陷的开发成果导致软件质量大幅度下降，不断投入的人员和其他资源导致软件开发成本急

剧增加。因而，迫切需要有规范化的工程来制约软件开发的无序性，即像处理“工程”一样来处理软件研制的全过程。1968 年，在北大西洋公约组织的学术会议上第一次提出了“软件工程”这个词，倡导按照工程化的原则和方法组织软件开发工作。

1.1.3 软件工程和软件危机的解决

软件工程主要是通过提供规范化的分析设计方法及相应的工具软件，来避免或减少软件错误的发生，为最终根除软件危机提供强有力的技术保障。在软件工程中，为了确保质量，软件的含义已不再仅仅是指程序了，而明确为“软件是程序以及开发、使用和维护程序所需的所有文档”；“研制软件”也不仅仅是“编程序”了，而明确为“研制软件过程中所涉及的所有活动，包括分析、设计、编码、测试和维护等”。

当用工程化的方式有效地管理软件开发的全过程时，程序的编写只是整个工程的一部分，在它的前后还有更重要的工作。一般来说，任何计算机软件都有它的生命周期，一般可分为 5 个阶段：需求分析（Requirement Analysis）、设计（Design）、程序编写（Coding）、测试（Testing）、运行和维护（Run and Maintenance）。每个阶段都有明确的任务，并生成一定规格的文档交送给下一个阶段。

表 1-1 软件工程各个阶段的基本情况

阶段		基本任务	工作成果	占开发期的工作量	参加者
开发期	需求分析	理解和表达用户的要求、对开发的软件进行详细的定义	系统需求说明书	20%	用户、系统分析员、高级程序员
	设计	概要设计和详细设计，建立系统的结构，明确系统的实现方式	系统设计说明书、数据说明	15%	系统分析员、高级程序员
	程序编写	写程序	程序	20%	高级程序员、初级程序员
	测试	发现错误和排除错误	可运行的系统	45% 模块测试 25% 其他测试 20%	测试工程师、测试员
运行期	运行和维护	维护	改进的系统		用户、程序员

从表 1-1 中可以看出，开发期中各阶段的工作量是不一样的，软件测试占有非常突出的地位，工作量及开销都要占整个工程的一半左右，软件测试成为保证软件质量的重要手段，越来越引起人们的重视。

1.2 软件测试的基本知识

软件测试作为软件工程中保证软件质量的重要环节，越来越受到大家的重视。随着当今软件规模和复杂性的日益增加，进行专业化高效软件测试的要求越来越迫切，挑战性极强。大量存在于软件中的各种缺陷需要在软件测试阶段被发现和修正，软件测试员的目标就是找出缺陷，并确保其得以修复，从而保证软件的质量。

1.2.1 软件测试的背景和意义

软件测试在软件生存期中占有非常突出的位置，是保证软件质量的重要手段。软件项目

的实践一再说明，为了确保软件产品能够符合用户的需要，必须着眼于整个软件生存期，在各个阶段进行验证、确认和测试活动，使软件不会在开发完成后，被发现和用户的需求有较大的差距。

软件在很多领域广泛使用，然而软件是人编的，所以并不完美，存在各种各样的缺陷。历史上有很多案例可以证明。

案例 1：迪斯尼的狮子王，1994～1995 年

1994 年秋天，迪斯尼公司发布了第一个面向儿童的多媒体光盘游戏 Lion King Animated Storybook（狮子王动画故事书）。公司进行了大力宣传，销售额非常可观。但是，12 月 26 日，迪斯尼公司的客户服务部就淹没在愤怒的家长和孩子的电话狂潮中。后来证实，迪斯尼公司并没有对市场上投入使用的各种机型的 PC 进行测试，软件在少数系统中工作正常，但在大众使用的常见系统中却不能正常工作。

案例 2：美国航天局火星极地登陆，1999 年

1999 年 12 月 3 日，美国航天局的火星极地登陆飞船在试图登陆火星表面时失踪。错误修正委员会观测到故障，并认定出现误动作的原因极可能是某一个数据位被意外修改。大家一致声讨，问题为什么没有在内部测试时解决。后来发现，登陆飞船经过了多个小组测试，其中一个小组测试飞船的脚落地位置，另一个小组测试此后的着陆过程。前一个小组没有注意着地数据位是否置位，后一个小组在开始测试之前重置计算机、清除数据位，双方独立工作都很好，但却从未协调在一起。

案例 3：爱国者导弹防御系统，1991 年

美国爱国者导弹防御系统首次应用在海湾战争中对抗伊拉克飞毛腿导弹的防御战争中，几次失利，其中一枚在沙特阿拉伯的多哈击中 28 名美军士兵。分析专家发现问题在于一个软件缺陷，一个很小的时钟错误积累起来就可能拖延 14 小时，造成跟踪系统失去准确度。

案例 4：千年虫，1974 年

20 世纪 70 年代的一位程序员，为了节省空间，开发工资系统时将 4 位日期缩减为两位，致使类似系统在 2000 年到来之前，更换或升级系统以解决 2000 年错误的费用超过了数亿美元。

案例 5　Intel 奔腾浮点除法，1994 年

在计算机中输入以下算式（4195835/3145727）*3145727-4195835，如果结果为零，则计算器没有问题，若不为零，则使用的是老式 Intel 芯片。

以上只是一部分软件失败时发生的历史事件，后果也许是不方便，也可能是灾难性的。在这些事件中，显然软件未按照预期目标运转，软件出现了缺陷。随着时间的推移，软件缺陷修复的费用会数十倍地增长，例如，若编写需求说明书时就发现了软件缺陷，费用可能只要几角钱；若在测试时才发现软件缺陷，费用可能要几元钱；若缺陷是客户发现的，费用可能达到几百元。比如“迪斯尼狮子王”案例，假如在编写需求说明书时，如果项目成员已经研究过什么机型的 PC 流行，并且明确指出软件需要在该种机型配置上设计和测试，则付出的代价非常小；如果没有这样做，就需要软件测试员去搜集流行 PC 样机并在其上验证，可能会发现软件缺陷，这时付出的代价要高很多。

因此，随着当今软件规模和复杂性的日益增加，进行专业化高效软件测试的要求越来越迫切，挑战性极强。软件测试员的目标就是找出缺陷，尽可能早一些，并确保其得以修复，从而保证软件的质量。

那么，到底什么是软件缺陷呢？以下给出符合软件缺陷的5个规则。

1）软件未达到产品说明书中已经标明的功能。

2）软件出现了产品说明书中指明不会出现的错误。

3）软件未达到产品说明书中虽未指出但应当达到的目标。

4）软件功能超出了产品说明书中指明的范围。

5）软件测试人员认为软件难以理解、不易使用，或者最终用户认为该软件使用效果不良。

例如，计算器的产品说明书可能明确地声明了它能够准确无误地进行加、减、乘、除运算。假如作为一个测试人员，就会按下加号“+”键。结果什么反应也没有，根据第一条，这是一个软件缺陷，假如得到了错误的答案，根据第一条规则，仍然是软件缺陷；产品说明书中声明计算器永远不会崩溃或者停止反应，假如狂敲键盘使得计算器不能接受输入，则根据第二条规则，这是一个软件缺陷；测试计算器时，电池没电或者电量不足会导致计算不正确，根据第三条规则，这是软件缺陷；假如计算器除了加、减、乘、除外，还可以求平方根，这一功能产品说明书中并没有说明，根据第四条规则，这是软件缺陷；测试人员使用不便的地方，例如，按键小，布局不好，根据第五条规则，这些都是软件缺陷。

1.2.2 软件测试的定义

软件测试在软件开发成本中占有很大的比例，是保证软件质量的主要手段，越来越受到人们的重视。那么，什么是软件测试呢？这一基本概念很长时间以来存在着不同的观点。

Glen Myers 认为“程序测试是为了发现错误而执行程序的过程”。这一定义明确指出“寻找错误”是测试的目的。相对于“程序测试是证明程序中不存在错误的过程”，Myers 的定义是对的。把证明程序无错当做测试的目的不仅是不正确的，是完全做不到的，而且对做好测试工作没有任何益处，甚至是十分有害的。从这方面讲，应该接受 Myers 的定义以及其所蕴涵的方法论和观点。不过，这个定义规定的范围似乎过于狭窄，使得它受到很大限制。因为如前所述，除去执行程序以外，还有许多方法去评价和检验一个软件系统。

另外，有些测试专家认为软件测试的范围应当包括更广泛些。J.B.Goodenough 认为测试除了要考虑正确性以外，还应关心程序的效率、健壮性（Robustness）等因素，并且应该为程序调试提供更多的信息。S.T.Redwine 认为，软件测试应该包括几种测试覆盖，分别为功能覆盖、输入域覆盖、输出域覆盖、函数交互覆盖、代码执行覆盖。关于测试的范围，A.E.Westley 将测试分为4个研究方向，即验证技术（目前验证技术仅用于特殊用途的小程序）；静态测试（应逐步从代码的静态测试往高层开发产品的静态测试发展）；测试数据选择；测试技术的自动化。

总的来说，软件测试就是在软件投入运行前，对软件需求分析、设计规格说明和编码实现的最终审查，它是软件质量保证的关键步骤。通常对软件测试的定义有以下两种描述。

定义 1：软件测试是为了发现错误而执行程序的过程。

定义 2：软件测试是根据软件开发各阶段的规格说明和程序的内部结构而精心设计的一批测试用例，并利用这些测试用例运行程序以发现错误的过程。

在 IEEE 提出的软件工程标准术语中，软件测试被定义为“使用人工和自动手段来运行或测试某个系统的过程，其目的在于检验它是否满足规定的需求或弄清楚预期结果与实际结果之间的差别。”

事实上，所有发布的软件产品都会因为缺陷而导致用户的困扰和开发者时间和金钱上的额外开支。而这些导致成本风险的软件问题可以通过在软件生命周期的每一个阶段中充分规划与行验证和确认（Verification and Validation ）而大大降低。

广义的软件测试是由确认、验证、测试3个方面组成。

1）确认：是评估将要开发的软件产品是否是正确无误、可行和有价值的。这里包含了对用户需求满足程度的评价，意味着确保一个待开发软件是正确无误的，是对软件开发构想的检测。

2）验证：是检测软件开发的每个阶段、每个步骤的结果是否正确无误，是否与软件开发各阶段的要求或期望的结果相一致。验证意味着确保软件会正确无误地实现软件的需求，开发过程是沿着正确的方向在进行。

3）测试：与狭隘的测试概念统一。通常是经过单元测试、集成测试、确认测试和系统测试4个环节。

在整个软件生存期，确认、验证、测试分别有其侧重的阶段。确认主要体现在计划阶段、需求分析阶段，也会出现在测试阶段；验证主要体现在设计阶段和编码阶段；测试主要体现在编码阶段。事实上，确认、验证、测试是相辅相成的，确认无疑会产生验证和测试的标准，而验证和测试通常又会帮助完成一些确认，特别是在系统测试阶段。因此，软件测试贯串于软件定义和开发的整个过程。软件开发过程中所产生的需求规格说明、概要设计规格说明、详细设计规格说明以及源程序都是软件测试的对象。

1.2.3 软件测试的分类

软件测试按照不同的划分方法，有不同的分类。按照程序是否执行，可以分为静态测试和动态测试；按照测试用例的设计方法，可以分为白盒测试和黑盒测试；按照开发阶段划分，软件测试可分为单元测试、集成测试、确认测试、系统测试和验收测试；按照测试实施组织划分，软件测试可分为开发方测试、用户测试（β测试）和第三方测试；按照是否使用工具软件，可以分为手工测试和自动测试。

1. 静态测试和动态测试

原则上讲，可以把软件测试分为两大类，即静态测试和动态测试。静态测试的主要特征是在用计算机测试源程序时，计算机并不真正运行被测试的程序。这说明静态方法一方面要利用计算机作为对被测程序进行特性分析的工具，它与人工测试有着根本的区别；另一方面它并不真正运行被测程序，只进行特性分析，这是和动态方法不同的方面。因此，静态测试常被称为“分析”，静态分析是对被测程序进行特性分析的一些方法的总称。

值得注意的是，静态分析并不等同于编译系统，编译系统虽也能发现某些程序错误，但这些错误远非软件中存在的大部分错误，静态分析的查询和分析功能是编译程序所不能代替的。目前，已经开发出一些静态分析系统作为软件测试的工具，静态分析已被当做一种自动化的代码校验方法。不同的方法有各自的目标和步骤，侧重点不同。

静态测试阶段的任务主要表现为以下方面：检查算法的逻辑正确性；检查模块接口的正确性；检查输入参数是否有合法性检查；检查调用其他模块的接口是否正确；检查是否设置了适当的出错处理；检查表达式、语句是否正确，是否含有二义性；检查常量或全局变量使用是否正确；检查标识符的使用是否规范、一致；检查程序风格的一致性、规范性；检查代码是否可

以优化，算法效率是否最高；检查代码注释是否完整，是否正确反映了代码的功能。

静态测试包括代码检查、静态结构分析 、代码质量度量等。它可以由人工进行，也可以借助软件工具自动进行。

1）代码检查包括代码走查、桌面检查、代码审查等，主要检查代码和设计的一致性，代码对标准的遵循、可读性，代码的逻辑表达的正确性，代码结构的合理性等方面。代码检查的具体内容有变量检查、命名和类型审查、程序逻辑审查、程序语法检查和程序结构检查等。代码检查的优点主要体现在实际使用中，代码检查比动态测试更有效率，能快速找到缺陷，发现 30%～70%的逻辑设计和编码缺陷；代码检查看到的是问题本身而非征兆。代码检查的缺点是非常耗费时间，而且代码检查需要知识和经验的积累。

2）静态结构分析主要是以图形的方式表现程序的内部结构。例如，函数调用关系图、函数内部控制流图。其中，函数调用关系图以直观的图形方式描述一个应用程序中各个函数的调用和被调用关系；控制流图显示一个函数的逻辑结构，由许多节点组成，一个节点代表一条语句或数条语句，连接结点的叫边，边表示节点间的控制流向。

3）代码质量度量针对软件的可维护性，目前业界主要存在 3 种度量参数：Line 复杂度、Halstead 复杂度和 McCabe 复杂度。其中 Line 复杂度以代码的行数作为计算的基准。Halstead 复杂度以程序中使用到的运算符与运算元数量作为计数目标（直接测量指标），然后可以计算出程序容量、工作量等。McCabe 复杂度一般称为圈复杂度，它将软件的流程图转化为有向图，然后以图论来衡量软件的质量。

动态测试的主要特征是计算机必须真正运行被测试的程序，通过输入测试用例，对其运行情况进行分析，判断期望结果和实际结果是否一致。动态测试包括以下内容。

1）功能确认与接口测试。

2）覆盖率分析。

3）性能分析。

4）内存分析。

2．黑盒测试和白盒测试

测试的关键是测试用例的设计，对任何工程产品都可用两种方法对其进行测试，第一是基于产品的功能来规划测试，检查程序各功能是否实现，并检查其中的错误，这种测试称为黑盒测试；第二是基于产品的内部结构来规划测试，检查内部操作是否按规定执行，各部分是否被充分利用，这种测试称为白盒测试。一般来说，这两类测试方法是从完全不同的起点出发，两类方法各有侧重，各有优缺点，构成互补关系，在测试的实践中都是有效和实用的，在规划测试时需要把黑盒测试和白盒测试结合起来。通常在进行单元测试时大都采用白盒测试，而在确认测试或系统测试中大都采用黑盒测试。

（1）黑盒测试

黑盒测试（Black-Box Testing）又称功能测试、数据驱动测试或基于规格说明书的测试，是一种从用户观点出发的测试。用这种方法进行测试时，把被测试程序当做一个黑盒，在不考虑程序内部结构和内部特性，测试者只知道该程序的输入和输出之间的关系或程序的功能的情况下，依靠能够反映这一关系和程序功能需求规格的说明书，来确定测试用例和推断测试结果的正确性。软件的黑盒测试被用来证实软件功能的正确性和可操作性。

黑盒测试主要根据规格说明设计测试用例，并不涉及程序的内部构造。它是一种传统的

测试方法，有严格的规定和系统的方式可供参考。应该说功能测试不仅能够找到大多数其他测试方法无法发现的错误，而且是一些外购软件、参数化软件包以及某些生成的软件的主要测试方法，由于无法得到源程序，用其他方法进行测试是完全无能为力的。如图 1-1 所示。

图 1-1　黑盒测试示意图

但任何软件作为一个系统都是有层次的，在软件的总体功能之下可能有若干个层次的功能，而且软件开发总是从把原始问题变换成计算机能处理的形式开始的，接着进行一系列变换，最后得到程序编码。在这一系列变换工程中，每一步都得到不同形式的中间成果，生成相应功能。因而，测试人员常常面临的一个实际问题是在哪个层次上进行测试，如果仅在高层上进行测试，就可能忽略一些细节，测试可能是不完全的和不充分的；若是在低层次上展开测试，又可能忽略各功能之间存在的相互作用和相互依赖关系。从策略上说，重要的是要发展可靠的并且高效的功能测试方法。

此外，如果想用黑盒测试发现程序中所有的错误，则必须用输入数据的所有可能值来检查程序是否都能产生正确的结果，这显然是做不到的。一方面在于输入和输出的结果是否正确本身无法全部事先知道；另一方面要穷举所有可能的输入更是天方夜谭。还需说明的是，黑盒测试的测试数据是根据规格说明书决定的，但实际上，并不能保证规格说明书是完全正确的，因而也就可能存在问题。例如，在规格说明书中规定了多余的功能，或是漏掉了某些功能，这对于黑盒测试来说是无能为力的。

局限于功能测试是不够的，因而还要花时间和精力来进行逻辑（结构）测试。

（2）白盒测试

白盒测试（White-Box Testing）又称结构测试、逻辑驱动测试或基于程序的测试。它依赖于对程序细节的严密检查，针对特定条件和循环集设计测试用例，对软件的逻辑路径进行测试。在程序的不同点检验“程序的状态”，以判定其实际情况是否和预期的状态一致。白盒测试如图 1-2 所示。

白盒测试主要是根据被测程序的内部结构设计测试用例的。有人可能会认为全面的白盒测试将产生“百分之百正确的程序”，只要保证程序中所有的路径都执行一次。这显然是不可能的。即使是一个非常小的控制流程，如循环语句（循环次数是 20 次）又嵌套 4 个 IF-THEN-ELSE 语句，则该程序可能的路径数达 5~10，如进行穷举测试，假设每毫秒内开发一个测试用例进行测试，并评估结果。每天运行 24 小时，每年运行 365 天，则需要 3170 年

的时间来测试这个程序。

因此，白盒测试要求对某些程序的结构特性做到一定程度的覆盖，或者说是“基于覆盖的测试”，并以此为目标，朝着提高覆盖率的方向努力，找出那些已被忽略的程序错误。为取得被测程序的覆盖情况，最为常用的方法是在测试前对被测程序进行预处理。预处理的主要工作是在其重要的控制点插入“探测器”——程序插装。必须说明，无论哪种测试覆盖，即使其覆盖率达到百分之百，都不能保证把所有覆盖的程序欠缺都揭露出来。

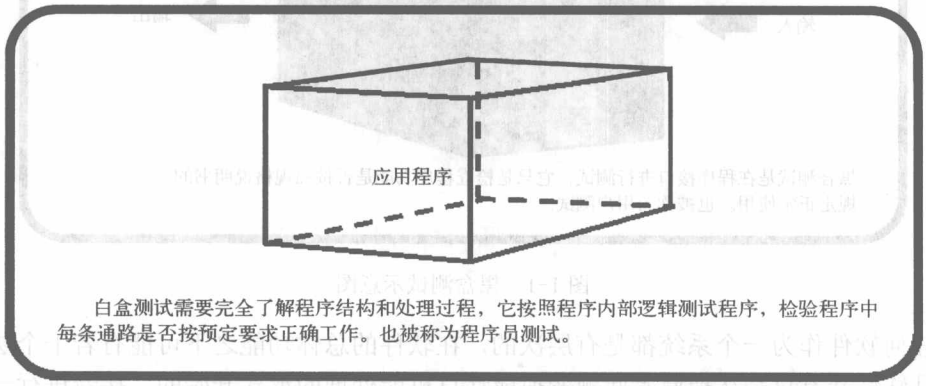


图 1-2　白盒测试示意图

（3）黑盒测试与白盒测试的比较

黑盒测试以用户的观点，从输入数据与输出数据的对应关系，即根据程序外部特性进行测试，而不考虑内部结构及工作情况。黑盒测试技术注重于软件的信息域（范围），通过划分程序的输入和输出域来确定测试用例。若外部特性本身存在问题或规格说明的规定有误，则应用黑盒测试方法是不能发现问题的。白盒测试只根据程序的内部结构进行测试，测试用例的设计要保证测试时程序的所有语句至少执行一次，而且要检查所有的逻辑条件。如果程序的结构本身有问题，比如说程序逻辑有错误或者有遗漏，那白盒测试也是无法发现的。黑盒测试与白盒测试的简单比较如表 1-2 所示。

表 1-2　黑盒测试与白盒测试的比较

项　　目	黑盒测试法	白盒测试法
规划方面	功能的测试	结构的测试
优点方面	能确保从用户的角度出发进行测试	能对程序内部的特定部位进行覆盖测试
缺点方面	无法测试程序内部特定部位；当规格说明有误，则不能发现问题	无法检查程序的外部特性 无法对未实现规格说明的程序内部欠缺部分进行测试
应用范围	边界分析法 等价类划分法 决策表测试	语句覆盖，判定覆盖 条件覆盖，判定/条件覆盖 路径覆盖，循环覆盖 模块接口测试

3. 单元测试、集成测试、确认测试、系统测试和验收测试

软件测试按照测试的不同阶段，可以分为单元测试、集成测试、确认测试、系统测试和验收测试。其中，单元测试是针对每个单元的测试，以确保每个模块能正常工作为目标。集