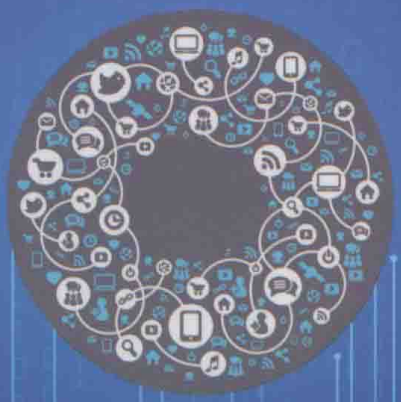
高等院校电子信息类规划教材



Web网站自动化测试

邢颖　周晓光　宋海宁　主编

非外借

北京邮电大学出版社
www.buptpress.com



高等院校电子信息类规划教材

Web 网站自动化测试

邢　颖　周晓光　宋海宁　主编



北京邮电大学出版社
www.buptpress.com

内 容 简 介

本书作为邮政快递相关专业创新实践课的配套教材，旨在紧跟软件测试的潮流与风向，基于实际的自动化测试框架项目的设计理念和开发经验，全面系统地介绍与 Web 测试相关的知识、技术和方法，帮助学生掌握 Web 测试中所需的各项知识和技能，从而保证该课程的顺利开展，为相关专业输送具备实战能力的技术人才。

图书在版编目(CIP)数据

Web 网站自动化测试 / 邢颖，周晓光，宋海宁主编. -- 北京：北京邮电大学出版社，2021.6
ISBN 978-7-5635-6389-0

Ⅰ. ①W… Ⅱ. ①邢… ②周… ③宋… Ⅲ. ①网站—测试—高等学校—教材 Ⅳ. ①TP393.092.1

中国版本图书馆 CIP 数据核字(2021)第 117147 号

策划编辑：彭　楠　　**责任编辑**：王晓丹　米文秋　　**封面设计**：七星博纳

出版发行：北京邮电大学出版社
社　　址：北京市海淀区西土城路 10 号
邮政编码：100876
发 行 部：电话：010-62282185　传真：010-62283578
E-mail：publish@bupt.edu.cn
经　　销：各地新华书店
印　　刷：唐山玺诚印务有限公司
开　　本：787 mm×1 092 mm　1/16
印　　张：13.25
字　　数：330 千字
版　　次：2021 年 6 月第 1 版
印　　次：2021 年 6 月第 1 次印刷

ISBN 978-7-5635-6389-0　　定价：35.00 元

·如有印装质量问题，请与北京邮电大学出版社发行部联系·

前　言

本书以培养高层次的 Web 测试专业人才为出发点，结合当前的 Web 测试人才需求，紧跟行业发展方向，有针对性地设置了教学内容。本书的主要内容包括软件测试概述、软件缺陷、黑盒测试、Web 自动化测试概述、Web 自动化测试实现原理、自动化测试平台 ATF (Automatic Testing Framework)简介、ATF 概述、ATF 测试基础设施建设、ATF 项目测试流程、软件评审、电子商务网站实战演练和智慧校园网站实战演练，共 12 章。

其中，第 1 章，软件测试概述，介绍了软件测试的背景、基本概念以及测试流程等。第 2 章，软件缺陷，先介绍了软件缺陷的基本概念，然后重点介绍了如何管理软件缺陷。第 3 章，黑盒测试，主要介绍了黑盒测试方法以及黑盒测试工具。第 4 章，Web 自动化测试概述，主要介绍了 Web 测试的基本概念及其技术的实践与发展历程，并进一步介绍了 Web 自动化测试的概念。第 5 章，Web 自动化测试实现原理，先介绍了目前常见的 Web 自动化测试工具以及先进的 ATF 自动化测试框架，之后讲解了自动化测试的基本流程和自动化模拟技术。第 6 章，ATF 简介，在宏观角度上对 ATF 做了整体介绍，包括产生背景、设计理念、支持的浏览器和平台、辅助工具和能力要求。第 7 章，ATF 概述，着重介绍了 ATF，并对其系统架构、创新点和优势进行了概述。第 8 章，ATF 测试基础设施建设，介绍了 ATF 中的测试基础设施建设模块，包含自动化构件管理与维护、元素库、执行代码管理、基础脚本、执行机管理、用户权限等内容。第 9 章，ATF 项目测试流程，具体介绍了使用 ATF 进行 Web 测试工作流程的全周期。第 10 章，软件评审，主要介绍了软件评审的组织、内容、方法和要点。第 11 章，电子商务网站实战演练，介绍了在电子商务网站中使用 ATF 进行测试的步骤及流程，给读者提供了工具使用的范例。第 12 章，智慧校园网站实战演练，介绍了在更为复杂的智慧校园网站中使用 ATF 进行测试的步骤及流程，使读者能够更加熟练地使用 ATF 测试工具。

本书是根据北京邮电大学教育部信息网络工程研究中心的研发成果编写而成的，由邢颖、周晓光、宋海宁主编，亦得益于钱晓萌、吴佳轩、王兴德、付寿东等团队成员的努力和贡献，在此对他们的付出表示感谢。

目　录

第 1 章　软件测试概述 …… 1

1.1　软件测试的背景 …… 1
1.1.1　软件缺陷的简单介绍 …… 1
1.1.2　软件测试的重要性 …… 3
1.2　软件测试的基本概念 …… 4
1.2.1　软件测试的定义 …… 4
1.2.2　软件测试的分类 …… 5
1.2.3　软件测试的目的 …… 6
1.2.4　软件测试的基本原则 …… 6
1.3　软件开发 …… 7
1.3.1　软件开发的流程 …… 7
1.3.2　软件开发的模型 …… 9
1.4　软件测试的过程 …… 13
1.4.1　软件测试的流程 …… 13
1.4.2　软件测试过程模型 …… 16
1.5　软件测试人员 …… 18

第 2 章　软件缺陷 …… 20

2.1　软件缺陷概述 …… 20
2.1.1　软件缺陷的定义 …… 20
2.1.2　软件缺陷的产生原因 …… 21
2.1.3　软件缺陷的分析 …… 23
2.1.4　软件缺陷的分类 …… 25
2.1.5　软件缺陷的数目估计 …… 30
2.1.6　软件缺陷的分布 …… 32
2.1.7　软件缺陷的效率分析 …… 33
2.2　软件缺陷管理 …… 35

2.2.1 软件缺陷管理的概念…… 35
2.2.2 软件缺陷管理的目标…… 35
2.2.3 软件缺陷管理的过程…… 35
2.2.4 软件缺陷管理的工具…… 37
2.2.5 软件缺陷报告…… 39

第 3 章 黑盒测试 …… 42

3.1 黑盒测试的基本概念…… 42
3.2 黑盒测试方法概述…… 43
3.3 等价类划分法…… 43
3.3.1 等价类划分法的概念…… 43
3.3.2 等价类划分法的具体实施…… 44
3.3.3 等价类划分法的实例分析…… 45
3.4 边界值分析法…… 47
3.4.1 边界值分析法的概念…… 47
3.4.2 边界值分析法的具体实施…… 48
3.4.3 边界值分析法的实例分析…… 49
3.5 决策表分析法…… 50
3.5.1 决策表分析法的概念…… 50
3.5.2 决策表分析法的具体实施…… 51
3.5.3 决策表分析法的实例分析…… 52
3.6 黑盒测试方法的比较和选择…… 53
3.6.1 测试工作量…… 54
3.6.2 测试效率…… 55
3.6.3 总结…… 56
3.7 黑盒测试工具…… 56
3.7.1 黑盒测试工具概要…… 56
3.7.2 黑盒功能测试工具——Selenium …… 57
3.7.3 黑盒功能测试工具——QTP …… 58
3.7.4 黑盒功能与性能测试工具——Robot …… 58
3.7.5 黑盒性能测试工具——JMeter …… 59

第 4 章 Web 自动化测试概述 …… 60

4.1 Web 测试的基本概念 …… 60
4.2 Web 测试技术的实践与发展 …… 60
4.2.1 传统软件测试…… 60
4.2.2 Web 测试与传统软件测试的区别 …… 61

4.2.3 基本的 Web 测试技术 …… 61
4.2.4 Web 2.0 下新增的 Web 测试技术 …… 62
4.3 Web 自动化测试的基本概念 …… 63
4.3.1 自动化测试介绍 …… 63
4.3.2 自动化测试的优点 …… 64
4.3.3 使用自动化测试的条件 …… 65
4.3.4 Web 自动化测试的简单介绍 …… 65

第 5 章 Web 自动化测试实现原理 …… 67

5.1 自动化测试框架介绍 …… 67
5.1.1 自动化测试框架概述 …… 67
5.1.2 四种常见的自动化测试框架模式 …… 67
5.1.3 自动化测试框架的作用 …… 68
5.1.4 自动化测试框架设计的核心思想 …… 68
5.2 自动化测试的基本流程 …… 69
5.2.1 用例管理 …… 69
5.2.2 数据管理 …… 70
5.2.3 脚本管理 …… 70
5.2.4 执行管理 …… 71
5.2.5 结果统计分析 …… 71
5.3 自动化测试页面元素的定位 …… 71

第 6 章 ATF 简介 …… 77

6.1 ATF 的产生背景 …… 77
6.2 ATF 的设计理念 …… 77
6.3 ATF 支持的浏览器 …… 78
6.4 自动化测试辅助工具 …… 79
6.4.1 Firefox 浏览器或 Google Chrome 的安装 …… 79
6.4.2 浏览器自带的辅助开发工具 …… 79
6.5 学习 ATF 工具的能力要求 …… 80

第 7 章 ATF 概述 …… 81

7.1 ATF 的系统构架 …… 81
7.1.1 ATF 的系统结构 …… 81
7.1.2 ATF 的测试界面 …… 83
7.2 ATF 的创新点及优势 …… 85
7.2.1 与传统工具的特点比较 …… 85

7.2.2 ATF 的创新点 …… 85
7.2.3 ATF 的优势 …… 88

第 8 章 ATF 测试基础设施建设 …… 90

8.1 自动化构件管理与维护 …… 90
8.1.1 自动化构件管理 …… 90
8.1.2 自动化构件维护 …… 92
8.2 元素库 …… 93
8.2.1 元素库的设计思想 …… 93
8.2.2 元素库的配置方法 …… 94
8.3 执行代码管理 …… 95
8.4 基础脚本 …… 98
8.4.1 基础脚本的参数化原理 …… 98
8.4.2 目标代码的生成 …… 99
8.5 执行机管理 …… 100
8.6 用户权限 …… 101

第 9 章 ATF 项目测试流程 …… 103

9.1 测试基础设施的维护 …… 103
9.1.1 自动化构件的维护 …… 103
9.1.2 元素库的维护 …… 103
9.1.3 基础脚本的维护 …… 105
9.2 测试项目的建立与管理 …… 105
9.3 测试用例管理 …… 107
9.4 测试资源管理 …… 112
9.4.1 测试资源管理简介 …… 112
9.4.2 测试资源编辑 …… 113
9.5 测试场景的配置 …… 116
9.5.1 配置测试用例 …… 117
9.5.2 设置定时执行 …… 117
9.5.3 管理定时任务 …… 117
9.5.4 触发器设置 …… 118
9.5.5 执行过程控制 …… 119
9.5.6 数据资源池配置 …… 120
9.6 测试计划及执行配置 …… 121
9.7 执行机调度管理 …… 123
9.8 批次执行查询 …… 125

9.9　测试记录单的查看与分析 …… 126
9.10　测试结果的统计分析 …… 127

第 10 章　软件评审 …… 128

10.1　概述 …… 128
10.2　软件评审的组织 …… 128
10.2.1　内部评审 …… 128
10.2.2　外部评审 …… 129
10.3　软件评审的内容 …… 130
10.3.1　管理评审 …… 130
10.3.2　技术评审 …… 130
10.3.3　文档评审 …… 131
10.4　软件评审方法 …… 131
10.4.1　走查法 …… 131
10.4.2　结构走查法 …… 132
10.4.3　审查法 …… 133
10.4.4　三种评审方法的比较 …… 134
10.5　软件评审的要点 …… 134

第 11 章　电子商务网站实战演练 …… 136

11.1　被测系统的添加与配置 …… 137
11.1.1　添加被测系统 …… 137
11.1.2　配置被测系统 …… 138
11.2　测试项目的添加 …… 140
11.3　登录功能测试 …… 141
11.3.1　新建功能点 …… 141
11.3.2　元素库的添加 …… 141
11.3.3　基础脚本的配置 …… 142
11.3.4　用例的添加 …… 144
11.3.5　测试资源的配置 …… 144
11.3.6　测试计划及执行 …… 145
11.4　注册功能测试 …… 147
11.4.1　新建功能点 …… 147
11.4.2　元素库的添加 …… 147
11.4.3　基础脚本的配置 …… 148
11.4.4　用例的添加 …… 149
11.4.5　测试资源的配置 …… 149

11.4.6 测试计划及执行…… 149
11.5 商品搜索功能测试…… 150
11.5.1 新建功能点…… 150
11.5.2 元素库的添加…… 150
11.5.3 基础脚本的配置…… 151
11.5.4 用例的添加…… 151
11.5.5 测试资源的配置…… 151
11.5.6 测试计划及执行…… 152
11.6 下单功能测试…… 152
11.6.1 新建功能点…… 152
11.6.2 元素库的添加…… 153
11.6.3 基础脚本的配置…… 153
11.6.4 用例的添加…… 154
11.6.5 测试资源的配置…… 154
11.6.6 测试计划及执行…… 154
11.7 订单查询功能测试…… 155
11.7.1 新建功能点…… 155
11.7.2 元素库的添加…… 155
11.7.3 基础脚本的配置…… 155
11.7.4 用例的添加…… 156
11.7.5 测试资源的配置…… 157
11.7.6 测试计划及执行…… 157

第 12 章 智慧校园网站实战演练 …… 159

12.1 测试系统的添加与配置…… 159
12.1.1 添加测试系统…… 159
12.1.2 配置测试系统…… 161
12.2 测试项目的添加…… 163
12.3 基础配置功能模块测试…… 164
12.3.1 区域配置功能测试…… 164
12.3.2 项目配置功能测试…… 171
12.3.3 处理规则功能测试…… 176
12.3.4 通用配置功能测试…… 181
12.4 系统配置功能模块测试…… 186
12.4.1 组织管理功能测试…… 187
12.4.2 人员管理功能测试…… 191
12.4.3 角色管理功能测试…… 197

第1章　软件测试概述

1.1　软件测试的背景

随着计算机信息技术的飞速发展，计算机在人们的生活中起着越来越重要的作用，软件质量问题成为人们关注的焦点问题。急剧增长的计算机系统的规模和复杂性，以及不断增长的软件开发成本和软件故障造成的经济损失成为当前亟待解决的重要问题。在展望21世纪计算机科学发展方向与策略时，许多科学家把提高软件功能和提高性能放在提升软件质量的优先位置。

1.1.1　软件缺陷的简单介绍

软件缺陷是指在计算机软件或程序中存在的某种破坏正常运行能力的问题、错误，或者存在的隐藏功能缺陷。一般来说，满足下列5个规则之一才称发生了软件缺陷：

- 软件未实现产品说明书要求的功能；
- 软件出现了产品说明书指明不应该出现的错误；
- 软件实现了产品说明书未提到的功能；
- 软件未实现产品说明书虽未提到，但应该实现的目标；
- 软件难以理解，不易使用，运行缓慢，或者从测试员的角度看，用户最终会认为不好。

目前软件规模和复杂性急剧增加，具有上千万、上亿乃至数十亿行代码的软件非常普遍，软件故障逐渐成为导致计算机系统失效和停机的最主要因素。下面介绍几个实例。

1. 迪士尼狮子王游戏的软件缺陷

1994年秋天，迪士尼公司发布了第一个面向儿童的多媒体光盘游戏——《狮子王动画故事书》(*The Lion King Animated Storybook*)。当时已有许多其他公司在儿童游戏市场上运作多年，但是这次是迪士尼公司首次进军这个市场，所以其进行了大量的促销宣传。最终，狮子王游戏的销售额非常可观，该游戏成为孩子们那年节假日必买的游戏之一。然而，没过多久，迪士尼公司就面临了市场危机。12月26日，迪士尼公司的客户支持电话开始响个不停，很快，电话支持技术员们就被淹没在来自哭着完不成游戏的孩子们的愤怒家长的电话中，报纸和电视新闻进行了大量的报道。

后来证实，迪士尼公司未能对市面上投入使用的许多不同的 PC 机型进行广泛的测试。软件只有在迪士尼程序员使用的用来开发游戏的系统等极少数系统中可以正常使用，但在大多数公众使用的系统中却不能正常运行。

2. “爱国者”导弹防御系统缺陷

“爱国者”导弹防御系统是美国总统里根提出的战略防御计划（即星球大战计划）的简单版本，它首次应用在海湾战争对抗伊拉克的“飞毛腿”导弹的防御战中。尽管对该系统进行赞誉的报道不绝于耳，但是它确实在几次导弹对抗中失利，包括一次导致在沙特阿拉伯多哈的 28 名美国士兵被击毙。分析发现症结在于一个软件缺陷，系统时钟的一个很小的计时错误积累到 14 小时后，跟踪系统就不再准确，而在多哈的这次袭击中，系统已经运行了 100 多小时。

3. “千年虫”问题

20 世纪 70 年代早期的某个时间，某位程序员为其公司设计开发工资系统。因为工资系统相当依赖于日期的处理，所以需要节省大量的存储空间。他使用的计算机存储空间很小，迫使他尽量节省每一个字节。他将自己的程序压缩得比其他任何人的都紧凑，使用的方法之一是把 4 位数年份，如 1973 年，截取后两位，将年份缩减为 2 位数 73。他简单地认为只有到达 2000 年，他的程序开始计算 00 或 01 这样的年份时问题才会产生。虽然他知道会出现这样的问题，但是他认定在 25 年之内程序肯定会升级或者更换，而眼前的任务比计划遥不可及的未来更加重要。然而这一天毕竟到来了。1995 年，他的程序仍在使用，而他退休了，谁也不会想到要深入程序中，检查 2000 年的兼容问题，更不用说去修改了。全球各地更换或升级类似于前者的程序以解决潜在的 2000 年问题的费用达数千亿元。

4. 美国航天局火星登陆探测器缺陷

1999 年 12 月 3 日，美国航天局的“火星极地登陆者”号探测器试图在火星表面着陆时失踪。一位故障评估委员会成员在调查该故障后，认定出现故障的原因极可能是一个数据位被意外置位。最令人警醒的问题是，该故障为什么没有在内部测试时被发现呢？

着陆的计划是这样的：探测器向火星表面降落时，它将打开降落伞减缓下降速度。降落伞打开几秒后，探测器的三条腿将迅速展开，并锁定位置，准备着陆。探测器在离地面 1 800 米时，将丢弃降落伞，点燃着陆推进器，缓缓地降落到地面。

美国航天局为了节省资金，简化了确定何时关闭着陆推进器的装置。他们在探测器的脚部装了一个廉价的触点开关，在计算机中设置一个数据位来控制触点开关关闭燃料。想法很简单，探测器的发动机需要一直点火工作，直到脚“着地”为止。

故障评估委员会在测试中发现：许多情况下，当探测器的脚迅速撑开准备着陆时，机械振动也会触发着陆触点开关，设置致命的错误数据位。设想探测器开始着陆时，计算机极可能关闭着陆推进器，“火星极地登陆者”号探测器飞船下坠 1 800 米之后冲向地面，撞成碎片。

结果是灾难性的，但背后的原因却很简单：登陆探测器经过了多个小组的测试，其中一个小组测试飞船的脚的展开过程，另一个小组测试此后的着陆过程。前一个小组不会注意着地数据是否置位——这不是他们负责的范围；后一个小组总是在开始复位之前复位计算机，清除数据位。双方独立工作都做得很好，但合在一起就不是这样了。

5. Intel 奔腾芯片缺陷

在计算机的“计算器”中计算算式：

$$(4\,195\,835/3\,145\,727)\times 3\,145\,727-4\,195\,835$$

如果结果不为 0，就说明该计算机使用的是带有浮点除法软件缺陷的老式 Intel 奔腾处理器。

1994 年，美国弗吉尼亚州 Lynchburg 学院的一位博士在用奔腾 PC 解决一个除法问题时，发现了这个问题。之后，他将发现的问题发布在互联网上，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。不过，这种情况仅在精度要求很高的数学、科学和工程计算中才会出现。

这个事件引起人们关注的原因并不是这个软件缺陷，而是 Intel 公司解决问题的态度。

- Intel 公司的测试工程师在芯片发布之前就已经发现了这个问题，但管理层认为还没严重到一定要修正甚至公开的程度。
- 当这个软件缺陷被发现时，Intel 公司通过新闻发布和公开声明试图弱化问题的严重性。
- 当来自外界各方的压力增大时，Intel 公司承诺可以更换有问题的芯片，但要求用户必须证明自己受到缺陷的影响。

之后，互联网上充斥着客户要求 Intel 公司解决问题的呼声，新闻报道将 Intel 公司描述成不诚信者。最后，Intel 公司为自己处理软件缺陷时不当的行为道歉，并拿出 4 亿多美元来支付更换芯片的费用。由此可见，一个小小的软件缺陷造成的损失有多大。

产生软件缺陷的原因是多方面的，下面列举一些产生软件缺陷的原因。

- 技术问题：算法错误、语法错误、计算和精度问题、接口参数传递不匹配等。
- 团队工作：在软件开发过程中，负责不同部分的团队之间沟通不充分，出现前后不兼容等问题。
- 软件本身：文档错误、用户的使用方法不对；需求与设计不协调或不一致所带来的问题；系统的自我恢复或者数据的异地备份、灾难性恢复等问题。

不管采用什么样的技术和方法，软件中总会有故障产生，因为人的主观认识常常难以完全符合客观现实，而且与工程密切相关的工作人员之间的沟通和配合也不可能完美无缺。即使是标准商业化软件中也有故障存在，只是严重程度不同而已。为减少故障的引入，可以采用新的编程语言、先进的开发方式、完善的开发过程，但是我们无法完全杜绝软件中的故障。这就需要通过软件测试来发现软件中的故障及故障密度。

1.1.2　软件测试的重要性

“在发布新产品之前做好测试，提高软件产品质量，减少漏洞，是预防遭受木马攻击的根本途径。”北京康赛普特信息技术有限公司高级测试总监王亚智指出，零缺陷的软件是不存在的。但通过必要的测试，软件缺陷可减少 75%，降低软件使用风险。有关机构研究表明，国外软件开发商 40%的工作量要花在测试上，对一些可靠性、安全性要求较高的软件更是不惜人力、物力。以微软为例，早在 1999 年发布 Windows 2000 操作系统时，微软就投入了 250 多个项目经理，1 700 多个开发人员，而内部测试人员达到 3 200 人，比前两者之和还要

多。相比之下，国内 IT 产业还在软件测试人才稀缺这一基础难题上挣扎。由于人才供需失衡，国内 120 万软件从业人员中，真正能担任软件测试职位的不超过 5 万人，具有 3 年以上经验的软件测试工程师不足 1 万人，大多数软件厂商测试人员的数量不足开发人员总数量的五分之一，远落后于国外的先进水平。人才短缺使企业测试能力不足，限制了产品开发和行业发展。有专家分析指出：我国测试人才稀缺的主要原因是人才培养途径不健全。

目前，我国高等教育体系还没有开设软件测试的相关专业，仅有几家知名职业培训机构开设了相关课程，但每年培养的人才数量相对于市场的巨大缺口无异于杯水车薪，现阶段我国软件测试人才供需严重不平衡，人才培养迫在眉睫。从将消费者追求安全可靠的心态作为衡量软件产品是否合格的标准来看，软件测试不应仅仅是检验质量的工具，更应成为验证软件产品是否符合用户需求的保障。对软件厂商来说，只有拥有足够的软件测试人才才能对产品进行全面的安全测试，业务才有可能进一步拓展。

1.2 软件测试的基本概念

1.2.1 软件测试的定义

软件测试是使用人工或自动化手段来运行或测试某个系统的过程，其目的在于检验系统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。关于软件测试的定义，不同学者有不同的观点。

- IEEE 标准的定义：使用人工或自动化手段来运行或测试某个系统的过程。其目的在于检验系统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。
- G. J. Myers 给出的定义：程序测试是为了发现错误而执行程序的过程。这个定义被软件测试业界认可，并经常被引用。

但实际上，这样的定义还不能完全反映软件测试的内涵，它仍局限于“程序测试”。随后，G. J. Myers 进一步提出了有关程序测试的 3 个重要观点：

① 测试是为了证明程序有错，而不是证明程序无错。

② 一个好的测试用例在于它能发现至今未被发现的错误。

③ 一个成功的测试是发现了至今未被发现的错误的测试。

要完整地理解软件测试，就要从不同方面和视角去辩证地审视软件测试。概括起来，软件测试就是贯穿整个软件开发生命周期、对软件产品（包括阶段性产品）进行验证和确认的活动过程，其目的是尽快尽早地发现在软件产品中存在的与用户需求、预先的定义不一致的地方。

软件测试是一项花费昂贵的活动，客户希望通过软件测试来发现并改正程序中的错误，从而提高软件的质量和可靠性。在进行测试时不应该假定待检测软件没有故障，而应该从软件含有故障这个假定出发去测试程序，从中发现尽可能多的软件故障。因此，一个成功的测试是发现了至今未被发现的故障的测试，一个好的测试用例在于能发现至今尚未被发现的故障。

1.2.2　软件测试的分类

从不同的角度看，软件测试有不同的分类，如下所述。

根据是否针对系统的内部结构和具体实现算法来完成测试，可分为以下几类。

- 黑盒测试：通过软件的外部表现来发现其缺陷和错误。黑盒测试法把测试对象看成一个黑盒子，完全不考虑程序内部结构和处理过程。黑盒测试是在程序界面处进行测试，它只是检查样序是否按照需求规格说明书的规定正常实现。
- 白盒测试：又称结构测试，通过程序内部结构的分析、检查来寻找问题。白盒测试可以把程序看成装在一个透明的盒子里，也就是清楚了解程序结构和处理过程，检查是否所有的结构及路径都是正确的，检查软件内部动作是否按照设计说明的规定正常进行。
- 灰盒测试：介于白盒测试与黑盒测试之间的测试。灰盒测试关注输出对于输入的正确性；同时也关注内部表现，但这种关注并不像白盒测试那样详细、完整，只是通过一些表征性的现象、事件、标志来判断内部的运行状态。

根据测试过程中被测软件是否被执行可分为以下几类。

- 静态测试(static testing)：不实际运行被测软件，而只是静态地检查程序代码、界面或文档中可能存在的错误的过程。
- 动态测试(dynamic testing)：实际运行被测软件，输入相应的测试数据，检查输出结果和预期结果是否相符的过程。

按测试的阶段可分为以下几类。

- 单元测试：又称模块测试，是针对软件设计的最小单位——程序模块——进行正确性检验的测试工作。其目的在于检查每个程序单元能否正确实现详细设计说明中的模块功能、性能、接口和设计约束等要求，发现各模块内部可能存在的各种错误。单元测试需要从程序的内部结构出发设计测试用例。多个模块可以平行地独立进行单元测试。
- 集成测试：又称组装测试。通常在单元测试的基础上，将所有的程序模块进行有序的、递增的测试。集成测试用于检验程序单元或部件的接口关系，使其逐步成为符合概要设计要求的程序或整个系统。
- 确认测试：又称有效测试，是指在模拟的环境下，验证软件的所有功能和性能以及其他特性是否与用户的预期要求一致。通过了确认测试的软件才具备进入系统测试阶段的资质。
- 系统测试：在真实的系统运行环境下，检查完整的程序系统能否和系统(包括硬件、外设、网络和系统软件、文件平台等)正确配置、连接，并最终满足用户的所有需求。
- 验收测试：软件产品检验的最后一个环节，按照项目任务书或合同、供需双方约定的验收依据文档进行对整个系统的测试与评审，决定接收或拒绝系统。

按软件特性可分为以下几类。

- 功能测试：黑盒测试的一方面，检查实际软件的功能是否符合用户的需求。
- 性能测试：功能的另一个指标，主要关注软件中的某一功能在指定的时间、空间条件

下是否使用正常。

其他分类方式如下。

- 回归测试：修改了旧的代码之后重新进行测试，以确认修改没有引入新的错误或导致其他代码产生错误。
- 冒烟测试：在对一个新版本进行系统大规模的测试之前，先验证一下软件的基本功能是否实现，是否具备可测性，所以也称可测性测试。
- 随机测试：也称随意性测试，是指测试人员基于经验和直觉的探索性测试。其目的是模拟用户的真实操作，并发现一些边缘性的错误。
- 自动化测试：利用软件测试工具自动实现全部或部分测试。它是软件测试的一个重要组成部分，能完成许多手工测试无法实现或难以实现的测试。正确、合理地实施自动化测试，能够快速、全面地对软件进行测试，从而提高软件质量，节省经费，缩短软件发布周期。

1.2.3 软件测试的目的

测试方案设计的决定权在于软件测试的目的，如果软件测试的目的是要证明程序中没有隐藏的故障存在，就会不自觉地回避可能出现故障的地方，从而设计出一些不易暴露故障的测试方案，使程序的鲁棒性受到极大的影响。相反，如果测试的目的是要证明程序中有故障存在，就会力求设计出最能暴露故障的测试方案。软件测试的目的如下：

- 软件测试是为了发现错误而执行程序的过程。
- 测试是为了证明程序有错，而不是证明程序无错（发现错误不是唯一目的）。
- 一个好的测试用例在于它能发现至今未被发现的错误。
- 一个成功的测试是发现了至今未被发现的错误的测试。

注意 测试并不仅仅是为了找出错误。分析错误产生的原因和错误的分布特征，可以帮助项目管理者发现当前所采用的软件过程的缺陷，以便改进。同时，分析也能帮助我们设计出有针对性的检测方法，改善测试的有效性。没有发现错误的测试也是有价值的，完整的测试是评定测试质量的一种方法，详细而严谨的可靠性增长模型可以证明这一点。例如，Bev Littlewood 发现一个经过测试而正常运行了 n 小时的系统有继续正常运行 n 小时的可能。

1.2.4 软件测试的基本原则

从不同的角度出发，对软件测试的期望也不一样。用户希望软件测试充分暴露软件中存在的缺陷，从而考虑是否接受该产品；开发者希望软件测试能够充分发现软件产品中未被发现的错误，正确地实现用户的需求，确立用户对软件质量的信心。

一般情况下，要从用户和开发者的角度出发进行软件测试。软件测试需要达到为开发者提供修改意见，提高软件质量，以及为用户提供放心的产品，并对优秀的产品进行认证的目的。为了达到上述目标，软件测试应遵循以下基本原则：

① 软件测试工作应贯穿软件开发的各个阶段。在软件项目的需求分析和设计阶段，就

应该同步考虑测试问题，分析测试需求并制订相应的测试计划，对软件文档进行评审。发现问题的时间越早，所需要花费的代价越小。

② 软件测试工作应针对用户需求进行。要根据软件需求和设计文档进行测试需求分析和设计工作，并建立用户需求、测试项和测试用例之间的对应关系。

③ 软件测试工作不应由开发人员承担，而应该由独立的软件测评机构或测评小组来完成。软件测试需要反向思维，采用异常数据和边界数据，这样才容易发现问题。但是开发者有一定的思维定式，其设计的测试用例偏重于采用正常数据，将可能出现问题的地方规避掉，从而导致不能全面地发现软件中的问题。

④ 软件测试中的测试用例应考虑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，要制造一些极端状态或意外状态，如网络异常中断、电源断电等情况。

⑤ 软件测试总是不全面的。由于软件程序中循环的存在，程序中存在海量的路径，因此进行穷举测试是不可能的，不要试图通过穷举测试来验证程序的正确性。

⑥ 软件测试要注意测试中错误集中的现象。这与软件开发员的编程水平和习惯有很大的关系。

⑦ 软件测试应明确测试的目的，制订详细的测试计划，并严格执行，尽可能避免测试的随意性。要把软件测试的时间尽可能安排得比较合适，在极短的时间内完成一个高水平的测试可能会出现很多不确定性。

⑧ 软件测试一定要充分注意回归测试的关联性。修改一个错误而引起更多错误的现象并不少见。

⑨ 软件测试要妥善保存测试过程中的一切文档，包括测试需求规格说明、测试计划、测评结果、原始记录、问题报告以及测评报告等，这有利于重现问题，为维护提供方便。

1.3　软 件 开 发

软件，简单地说就是那些在计算机中能看得到，但摸不着的东西。概念性地说，软件也称“软设备”，广义地说，软件是指系统中的程序以及开发、使用程序所需要的所有文档的集合。软件不只包括可以在计算机上运行的程序，与这些程序相关的文件一般也被认为是软件的一部分。软件被应用于各个领域，对人们的生活和工作都产生了深远的影响。

1.3.1　软件开发的流程

一个软件产品的开发过程与计算机程序爱好者编写一个小程序的过程是完全不同的。软件开发可能需要几十、几百甚至几千人的协同工作，例如，大约有 6 000 人参与开发 Windows 2000 Server。正规的软件开发过程一般包括软件计划、需求分析、软件设计、程序编码、软件测试和软件维护六个阶段。这六个阶段构成了软件的生命周期，以下给出各阶段的主要任务。