获选“十二五”职业教育国家规划教材

高等应用型人才培养规划教材

软件测试教程

（第3版）

贺 平 编著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

获选“十二五”职业教育国家规划教材

高等应用型人才培养规划教材

软件测试教程

（第3版）

贺　平　编著

電子工業出版社

Publishing House of Electronics Industry

北京 · BEIJING

内 容 简 介

本书全面阐述了软件测试的基本理论和业界主流的技术方法，并从软件生命周期的最新视角展开和分析软件测试的知识、技术及应用的策略、过程及方法。全书共 10 章：软件测试概述、软件生命周期的测试、软件静态测试技术、软件动态测试技术、软件自动化测试、软件项目的组件测试、软件系统性功能测试、软件系统性能测试、软件系统安全性测试、软件测试管理，基本涵盖了目前软件测试的知识体系、技术体系和应用体系。本书使读者能系统、较快地掌握软件测试的系统知识，获得解决实际测试问题的思路和基本的工程实践方法。

本书可作为高等院校、高职高专院校的软件工程、软件技术、软件测试及相关的信息技术类专业教材，也可作为参加国际软件测试工程师认证（ISTQB）的主选参考资料。

图书在版编目（CIP）数据

软件测试教程/贺平编著. —3 版. —北京：电子工业出版社，2014.8
高等应用型人才培养规划教材
ISBN 978-7-121-23818-5

Ⅰ. ①软… Ⅱ. ①贺… 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（2014）第 156910 号

策划编辑：吕 迈
责任编辑：吕 迈 特约编辑：张燕虹
印 刷：三河市鑫金马印装有限公司
装 订：三河市鑫金马印装有限公司
出版发行：电子工业出版社
北京市海淀区万寿路 173 信箱 邮编 100036
开 本：787×1 092 1/16 印张：23.75 字数：608 千字
版 次：2005 年 6 月第 1 版
2014 年 8 月第 3 版
印 次：2014 年 8 月第 1 次印刷
印 数：4 000 册 定价：46.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

本书此次修订较第一版、第二版在体系结构上做了较大调整与变动。全书在内容上做了整合与更新，使教材的容量基本保持不变而内涵进一步提升，突出软件测试课程教程的功能，以更加符合学习规律、认识规律，全面阐述软件测试基本理论和业界主流的技术方法，并从软件生命周期的最新视角阐述和分析软件测试的知识、技术及应用，运用于工程中的策略、过程及方法，以期能基本涵盖目前软件测试领域的知识体系、技术体系和应用体系。

本书由软件测试概述、软件生命周期的测试、软件静态测试技术、软件动态测试技术、软件自动化测试、软件项目的组件测试、软件系统性功能测试、软件系统性能测试、软件系统安全性测试、软件测试管理共 10 章构成。其中，软件测试概述、软件生命周期的测试两章作为软件测试的基本理论知识加以系统介绍；软件静态测试技术、软件动态测试技术、软件自动化测试三章作为测试方法及技术应用分析的重点；软件项目的组件测试、软件系统性功能测试、软件系统性能测试、软件系统安全性测试，软件测试管理五章则从测试工程的角度进行详细深入的阐述。每章除了正文之外，还增加了有针对性的内容提要与学习目标的导学、本章小结等内容。为进一步体现和完善教材的功效，全书配备了例题、案例、习题、作业、应用实践项目共 300 多个，并为每章的相关内容加入了专业术语栏目与参考资料的索引。

本书内容丰富、层次清晰、阐述简明，较好地把握了所编选内容的深度及广度，使之不仅反映出软件测试的发展脉络、最新的研究与应用成果，而且更加注重将理论知识、技术基础与工程实践进行有机融合，使读者能系统、较快地学习到软件测试的系统知识与主流技术，获得解决实际测试问题的思路，掌握基本的工程方法与运用的策略。

本书是中国大学精品共享资源课程（爱课网）软件测试的配套主教材，以期作为高等学校软件测试课程的一本专门教程，成为软件测试工程师的首选读本，为进一步掌握、提高测试专业能力和职业发展起到奠基的作用。

本书涵盖了国际软件测试工程师认证（ISTQB FL/AL）大纲 2007 版所规定的许多内容，因此可作为准备参加此项专业认考的参考资料。

本书含有大量的算法语句、程序语句及计算公式，对于其中的变量，为了方便读者阅读，避免歧义，不再区分正、斜体，而是统一采用正体，特此说明。

贺萌、吴晓园、杨艳、夏辉、徐芳、李健、李晓兵、赵琳也参与了本书的编写与资料整理工作，在此一并感谢。

本书从首版编写到第 3 版的出版，始终得到电子工业出版社的大力、持久的支持和帮助，在此谨表敬意与衷心感谢。

因作者水平所限，书中的错误和不妥在所难免，恳请读者批评指正和提出改进建议。

联系电子邮箱：he_ping2002@163.com

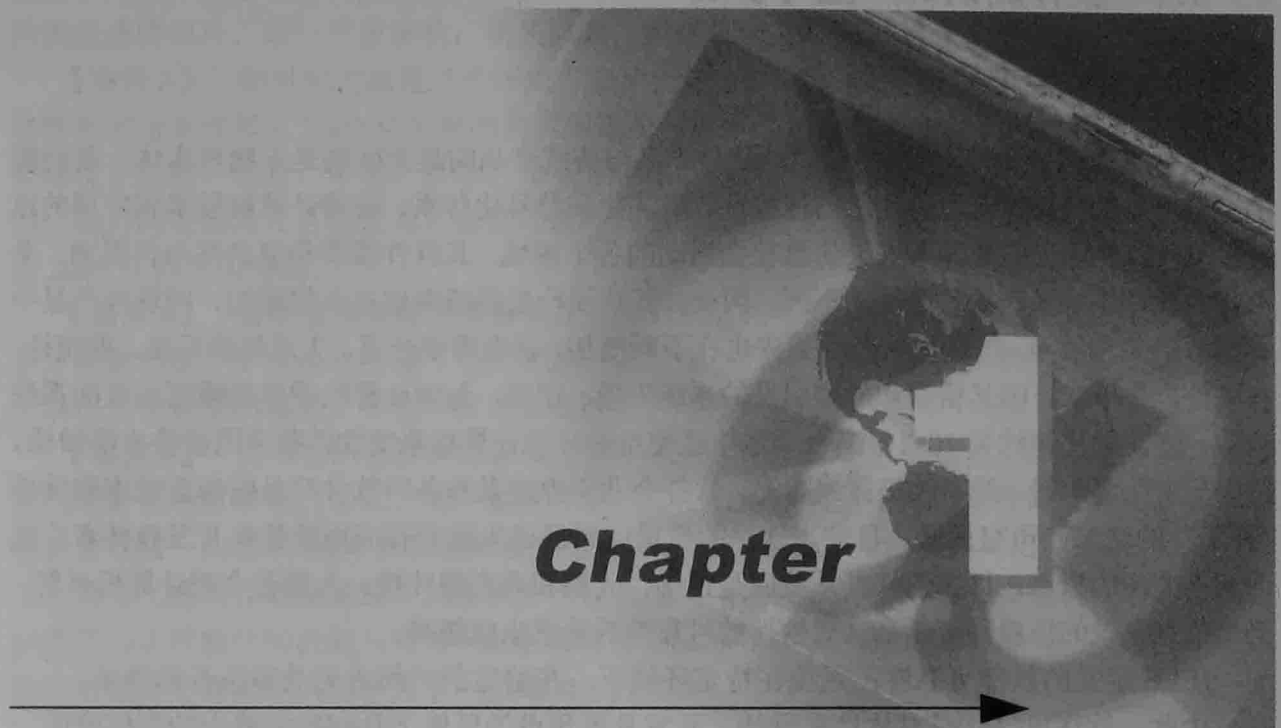
编　著　者

目　录

第 1 章　软件测试概述 …… 1
1.1　软件测试的产生与发展 …… 2
1.1.1　软件可靠性问题 …… 2
1.1.2　软件缺陷与故障 …… 2
1.1.3　软件测试的产生与发展 …… 4
1.1.4　软件测试的发展趋势 …… 7
1.2　软件测试基础知识与理论 …… 8
1.2.1　软件测试的目的与原则 …… 8
1.2.2　软件测试的基本原理与特性准则 …… 8
1.2.3　软件测试的基本策略 …… 9
1.3　软件开发模式与软件测试 …… 11
1.3.1　软件开发模式 …… 11
1.3.2　软件开发与软件测试 …… 17
1.3.3　软件测试模型分析 …… 18
1.4　软件质量及其保证 …… 21
1.4.1　软件质量体系 …… 21
1.4.2　软件测试成熟度 …… 24
本章小结 …… 28
习题与作业 …… 28
第 2 章　软件生命周期的测试 …… 32
2.1　软件生命周期中的测试 …… 33
2.1.1　软件生命周期 …… 33
2.1.2　软件生命周期中的测试策略 …… 34
2.1.3　软件测试通用流程 …… 35
2.2　软件测试技术分析 …… 36
2.2.1　基于动态测试分析与静态测试分析 …… 36
2.2.2　基于规格说明的测试技术 …… 37
2.2.3　基于结构的测试技术 …… 38
2.2.4　基于经验的测试技术 …… 39
2.2.5　手工测试与自动化测试 …… 42
2.2.6　基于风险的测试 …… 43
2.2.7　软件测试的分类 …… 44
2.3　组件测试 …… 46
2.3.1　组件测试的类别及模式 …… 46
2.3.2　组件测试的任务 …… 47
2.3.3　组件测试的过程 …… 49
2.3.4　组件测试管理 …… 49
2.4　集成测试 …… 50
2.4.1　集成测试概念 …… 50
2.4.2　集成测试策略 …… 50
2.5　系统测试 …… 52
2.5.1　系统测试的概念、对象、环境与目标 …… 52
2.5.2　系统的功能性测试 …… 52
2.5.3　系统测试的非功能性测试 …… 54
2.6　确认测试 …… 57
2.6.1　确认测试的准则 …… 57
2.6.2　程序修改后的确认测试 …… 58
2.6.3　配置与审查 …… 58
2.7　验收测试 …… 58
2.7.1　验收测试的含义 …… 58
2.7.2　验收测试的任务及内容 …… 59
2.7.3　软件文档验收测试 …… 60
2.8　软件新版本的测试 …… 60
2.8.1　软件维护测试 …… 60
2.8.2　软件版本开发的测试 …… 60
2.8.3　软件增量开发中的测试 …… 61
本章小结 …… 63
习题与作业 …… 64
第 3 章　软件静态测试技术 …… 72
3.1　软件静态测试 …… 73
3.1.1　静态测试技术概要 …… 73
3.1.2　静态测试技术 …… 74
3.2　程序数据流分析方法 …… 76
3.2.1　数据流测试 …… 76
3.2.2　数据流测试的应用举例 …… 77
3.3　程序控制流分析方法 …… 78
3.3.1　程序的控制流图 …… 78
3.3.2　将程序流程图转换为控制流图 …… 80

3.3.3 控制流图分析的测试应用 …… 81
3.4 软件的复杂性度量 …… 82
3.4.1 静态检查与测试对象的规范、标准的一致性 …… 82
3.4.2 软件复杂度的度量 …… 82
3.4.3 Logiscope 静态分析测试应用 … 90
3.5 软件评审 …… 100
3.5.1 软件评审的概念 …… 100
3.5.2 评审的组织 …… 101
3.5.3 评审过程 …… 101
3.5.4 评审类型 …… 102
本章小结 …… 103
习题与作业 …… 104
第 4 章 软件动态测试技术 …… 109
4.1 软件动态测试技术 …… 110
4.1.1 动态测试 …… 110
4.1.2 动态测试（黑盒技术）的测试模型 …… 112
4.2 等价类划分法与边界值分析法 … 113
4.2.1 等价类划分法简介 …… 113
4.2.2 边界值测试 …… 117
4.2.3 等价类划分测试法与边界值测试法结合设计测试用例 …… 119
4.3 因果图/决策表法 …… 121
4.3.1 因果图法 …… 121
4.3.2 决策表法 …… 123
4.3.3 因果图/决策表法的测试应用 … 126
4.4 状态转换法 …… 131
4.4.1 状态转换法原理 …… 131
4.4.2 运用状态转换法设计测试用例 …… 133
4.5 全配对法 …… 135
4.5.1 全配对法测试原理 …… 135
4.5.2 全配对测试法应用 …… 138
4.6 覆盖测试法 …… 142
4.6.1 逻辑覆盖 …… 142
4.6.2 路径覆盖 …… 147
4.6.3 循环的路径测试 …… 151
本章小结 …… 153
习题与作业 …… 154
第 5 章 软件自动化测试 …… 166
5.1 软件自动化测试概念 …… 167
5.1.1 自动化测试的原理 …… 167
5.1.2 自动化测试的优势与特点 …… 168
5.2 软件自动化测试生存周期方法学及应用 …… 170
5.2.1 自动化测试决策 …… 170
5.2.2 测试工具获取 …… 170
5.2.3 自动化测试引入 …… 171
5.2.4 测试计划、设计、开发 …… 171
5.2.5 测试执行与管理 …… 172
5.2.6 测试评审与评估 …… 173
5.3 自动化测试用例与脚本 …… 174
5.3.1 自动化测试用例的生成要求 … 174
5.3.2 自动化测试脚本 …… 174
5.4 自动化测试工具 …… 176
5.4.1 自动化测试的专项工具 …… 176
5.4.2 自动化测试套件 …… 181
本章小结 …… 190
习题与作业 …… 191
第 6 章 软件项目的组件测试 …… 193
6.1 软件项目的组件测试介绍 …… 194
6.1.1 组件测试的范围及内容 …… 194
6.1.2 软件项目的组件测试解决方案 …… 194
6.2 软件 GUI 的测试 …… 197
6.2.1 页面元素测试 …… 197
6.2.2 对窗体操作的测试 …… 198
6.2.3 对下拉式菜单与鼠标操作的测试 …… 198
6.2.4 对数据项操作的测试 …… 198
6.3 面向对象软件类的测试 …… 199
6.3.1 类、对象、消息及接口 …… 199
6.3.2 类的测试设计 …… 202
6.4 Logiscope 组件测试应用 …… 208
6.4.1 Logiscope 概况 …… 208
6.4.2 Logiscope 功能 …… 210
6.4.3 Logiscope 的安装与配置 …… 211
6.4.4 TestChecker 测试应用 …… 212
6.5 运用 JUnit 进行组件测试 …… 220

6.5.1 JUnit 的基本概要 …… 220
6.5.2 运用 JUnit 进行组件测试 …… 224
本章小结 …… 230
习题与作业 …… 231
第 7 章 软件系统性功能测试 …… 233
7.1 软件系统性功能测试 …… 234
7.1.1 软件系统性功能测试的内容 …… 234
7.1.2 软件系统性功能测试的基本要素 …… 235
7.2 软件功能测试工具及应用 …… 236
7.2.1 RFT 的一般概况 …… 236
7.2.2 RFT 的基本运用方法 …… 241
7.2.3 RPT 的测试应用 …… 254
本章小结 …… 260
习题与作业 …… 260
第 8 章 软件系统性能测试 …… 262
8.1 软件系统性能测试概述 …… 263
8.1.1 软件系统性能测试的概念 …… 263
8.1.2 软件系统性能测试规划与设计 …… 267
8.1.3 软件系统性能测试管理 …… 270
8.2 Web 性能测试 …… 272
8.2.1 Web 性能测试模型 …… 272
8.2.2 Web 性能测试用例设计 …… 273
8.2.3 Web 性能测试过程管理 …… 278
8.3 软件系统性能测试工具 …… 279
8.3.1 RPT 功能简介 …… 279
8.3.2 RPT 的基本测试应用分析 …… 281
8.3.3 RPT 性能测试工程应用 …… 286
本章小结 …… 293
专业术语 …… 293
习题与作业 …… 294
第 9 章 软件系统安全性测试 …… 296
9.1 软件系统安全性测试的问题 …… 297
9.1.1 软件系统安全性概述 …… 297
9.1.2 软件系统安全性测试策略 …… 302
9.1.3 软件系统安全性测试方法 …… 303
9.2 Web 应用系统的安全性测试 …… 305
9.2.1 Web 应用安全的背景 …… 305
9.2.2 Web 应用安全测试 …… 306
9.3 软件系统安全测试工具及测试应用 …… 310
9.3.1 AppScan 概要 …… 311
9.3.2 AppScan 功能特性 …… 312
9.3.3 AppScan 的基本使用 …… 320
9.3.4 AppScan 安全性测试应用 …… 328
本章小结 …… 330
习题与作业 …… 330
第 10 章 软件测试管理 …… 332
10.1 软件测试管理的概念 …… 333
10.1.1 测试管理的基本要素 …… 333
10.1.2 测试组织管理 …… 334
10.2 测试过程管理 …… 337
10.2.1 测试计划管理 …… 337
10.2.2 测试流程管理 …… 340
10.3 测试事件管理 …… 348
10.3.1 缺陷管理 …… 348
10.3.2 测试用例管理 …… 351
10.4 软件配置管理 …… 354
10.4.1 软件配置管理的内涵 …… 354
10.4.2 配置管理策略与方法 …… 356
10.4.3 配置管理的应用 …… 359
10.5 测试管理工具及应用 …… 360
10.5.1 TestDirector 测试管理工具 …… 360
10.5.2 Rational Test Manager 测试管理工具 …… 366
本章小结 …… 367
专业术语 …… 367
习题与作业 …… 368
参考文献 …… 372



Chapter 1

第 1 章　软件测试概述

本章导学

内容提要

本章为软件测试的概要阐述。包括软件测试基本理论知识，包含软件测试产生与发展、软件测试定义、软件开发模式（过程）与测试策略的关联关系、软件测试过程、软件质量及保证体系等主要内容，并简要介绍了软件测试的新技术、新应用及技术发展方向。

学习目标

通过本章学习，读者将能正确理解软件测试产生及背景、软件缺陷及故障、软件测试定义等概念，理解软件测试的基本思想与实施策略，能初步认识软件开发与软件测试相辅相成、相互依赖的关系，对软件测试建立概要性、框架性的整体认识和为后续学习软件测试策略、流程与工程方法奠定坚实基础。要求：

- ☒ 正确理解软件测试的背景及定义、软件缺陷、故障和软件失效等概念
- ☒ 正确理解和认识软件测试的目的及测试的意义
- ☒ 正确理解和认识软件测试的基本原理与实施策略
- ☒ 正确理解和认识软件开发过程与软件测试的依存关系
- ☒ 正确理解和认识软件质量的概念及质量保证体系

1.1 软件测试的产生与发展

1.1.1 软件可靠性问题

软件是人类智力的一种“产品”。这种产品与传统产品的最大区别是无物理实体，是纯粹的逻辑思维的结果，表现为高度的抽象性、形式化和符号化体系。随着计算机技术和应用的迅速发展，软件现已广泛深入人类信息社会活动的各个领域，其软件规模和复杂性与日俱增，千万行的软件系统（产品）已屡见不鲜。因此，其错误产生的概率也在大幅增加，因软件产品中存在的缺陷和故障，所造成的各类损失也在不断增加，甚至带来严重、灾难性的后果。据统计，自软件产生以来，因其错误和故障引发的系统失效、崩溃，与因计算机硬件故障而引发的系统失效、崩溃的比例约为 10:1。当今全球广泛使用的一些计算机系统软件和应用软件中的缺陷、错误与安全漏洞等，经常被披露或曝光，软件企业不停地发布各种软件产品的修正版本和软件“补丁”的现象已司空见惯。目前，软件的质量问题已成为被所有应用软件和开发软件者广泛而深入关注的焦点。可靠的软件系统应是正确、完整和具有健壮性。人类社会对计算机系统，乃至信息系统的依赖程度越高，对软件的可靠性的要求也就越高。

IEEE 定义的软件可靠性：系统在特定环境下，在给定的时间内无故障运行的概率。

由此，软件可靠性是对软件在设计、开发及所预设的环境下具有特定能力的置信度的一个度量，是衡量软件质量的主要指标。

软件产品的“特殊”性质，使其可靠性必须依赖软件工程的各个环节来保证，经过几十年的经验与总结，提高可靠性的最重要的策略与技术手段是在软件的生命周期中不断进行软件测试。

1.1.2 软件缺陷与故障

1. 软件缺陷与故障案例及其意义

尽管软件工程中采取一系列有效措施，不断提高软件的质量，但仍然无法完全避免软件会存在各种各样的缺陷，即软件中存在缺陷几乎不可避免。

软件缺陷或故障（在运行时发生），可依据其可能造成的危害及风险程度，分为极其严重、严重、一般、轻微等不同级别。这里，将通过比较典型的软件缺陷与故障的案例分析，来说明软件缺陷与故障问题所造成的严重后果及灾难。

1）软件缺陷与故障的典型案例

【事件 1】 美国迪士尼公司狮子王游戏软件 Bug 事件。这是一起典型的软件兼容性缺陷问题。造成这一严重问题的起因是，这个公司没有在已投入市场的各种 PC 上进行该游戏软件与硬件环境的兼容性的完整测试，使兼容性没有得到保障。虽然该游戏软件在开发者机器的硬件系统上工作正常，但在公众使用的其他各类硬件系统中存在不兼容问题，造成了该软件无法正常运行。用户大量的投诉，致使迪士尼遭受经济与声誉双重重大损失。

【事件 2】 美国航天局火星极地飞船登陆事故。1999 年 12 月 3 日，美国航天局火星极地登陆飞船在试图登陆火星表面时突然失踪。负责这一项目的错误修正委员会的专家们观测到了这一幕并分析了事故原因，确定事故可能是由于某一数据位被意外更改而造成灾难性的后果的，并得出造成该事故的问题应在内部测试时就予以解决的结论。事后分析测试发现，机械振动很容易触发飞船的着地触发开关，导致程序设置了错误的数据位，关闭了登陆推进

器燃料开关，切断了燃料供给，使推进器提前停止工作，飞船加速下坠1800m直接冲向火星的表面撞成碎片。这一严重事故，损失巨大，但是仅起因于软件的一个小缺陷。

【事件3】 跨世纪的软件“千年虫”缺陷问题。20世纪末最后几年，全球计算机硬件、软件和应用系统都为2000年的时间兼容问题及与此年份相关的其他问题付出代价。据统计，全球仅在金融、保险、军事、科学、商务等领域，对现有程序进行检查、修改，所花费的人力、物力耗资高达几百亿美元。而这些缺陷问题根源在于早期的软件设计，并未考虑跨世纪的2000年的时间问题。

【事件4】 美军爱国者导弹防御系统狂炸自己。美军爱国者导弹系统首次应用于海湾战争并屡建功勋，多次成功拦截飞毛腿导弹。但因很小的系统时钟错误积累的延时缺陷，造成了跟踪系统精度的偏差，导致一枚导弹在沙特多哈炸死了28名美军士兵。

【事件5】 美国加州监狱计算机程序缺陷致使上千高危犯人被错放。2011年5月，美国加利福尼亚州监察部门表示，由于监狱计算机程序缺陷，信息不完整致使误放近450名“高度暴力危险”囚犯和1000多名很可能实施毒品、财产犯罪的囚犯被假释出狱。

诸如上述软件缺陷或错误的案例并不仅为这几项，类似的问题数不胜数。现今，全球正在使用的多种软件的缺陷与错误（如系统的安全漏洞等），常常被披露和曝光，不时发布这些软件的修订升级版本或程序补丁，已成为一种常态。

2）软件缺陷与故障定义

从缺陷或故障的案例中已能够认识和理解什么是软件的缺陷和错误了，并观察出软件发生故障或事件的共同特点。软件开发可能未按预期规则或目标要求进行，虽经过测试，但不能保证完全发现和排除了已发现存在的或潜在的错误，甚至有一些是简单而细微的错误。

不论软件存在问题的规模与危害是大还是小，都会产生软件使用上的各种障碍，所以将这些问题统称为软件缺陷。对软件缺陷的精确定义，业界通常认同下列5条描述。

（1）软件未达到产品说明书中已标明的功能。

（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（隐含）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）测试者认为软件难以理解、不易使用，或最终用户认为软件使用效果不良。

3）软件缺陷的特征

软件测试理论研究与测试实践都表明，软件缺陷的特征有两个。

（1）软件系统的逻辑性与复杂性等特殊性质决定了其缺陷不易直接被肉眼观察到，即“难以看到”。

（2）即使在运行与使用中发现了缺陷或发生故障，仍不容易找到问题产生的原因，即“看到但难以抓到”。

4）软件缺陷产生原因

软件测试是在对软件需求分析、设计规格说明、编码实现和发布运行之前的最终审定。为何还会存在缺陷呢？研究表明，故障并不一定是由编码过程所引起的，大多数缺陷并非来自编码过程中的错误。因其缺陷很可能在系统概要或详细设计阶段、甚至在需求分析阶段就存在问题而导致故障发生，针对源程序进行的测试所发现的故障根源也可能存在于开发前期。事实上，导致缺陷最大原因在于软件产品的设计文档（设计、规划文本及说明书等）。

在多数情况下，软件产品的设计可能存在着没有文档，或描述不清楚、不准确，或在开发中对产品的需求及产品的功能经常变更，或因开发人员之间没有充分地进行交流与沟通，

或没有组织执行测试的流程等情形。因此，制定软件产品开发计划非常重要，如果产品计划没有制订好，缺陷就会潜伏，软件运行时出现故障问题就在所难免。

根据统计，软件缺陷产生的第二大来源是设计方案，这是实施软件计划的关键环节。

典型的软件缺陷产生原因大致归纳为以下 10 类。

（1）软件需求解释有错误或不明确。

（2）用户需求的定义中存在错误。

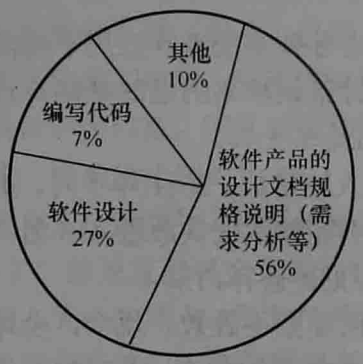


图 1-1　软件缺陷产生原因分布概率

（3）软件需求中记录了错误。

（4）软件设计说明中有错误。

（5）软件编码说明中有错误。

（6）软件程序代码存在错误。

（7）数据输入有错误。

（8）软件测试过程有错误。

（9）软件问题修改不正确或不彻底。

（10）有时，错误结果是因其他软件缺陷而引起或产生的。

图 1-1 所示是软件缺陷产生原因分布概率，软件产品的设计文档规格说明（如需求分析等）占 50%以上。

2. 软件测试定义

软件测试是发现缺陷的过程。在确定关于软件测试的概念或定义时，通常都会引用《软件测试的艺术》（Myers）一书中的观点。

（1）软件测试是为了发现错误而执行程序的过程。

（2）测试是为了证明程序有错，而不是证明程序没有错误。

（3）一个好的测试用例是它能发现至今未发现的错误。

（4）一个成功的测试是发现了至今未发现错误的测试。

软件测试就是在软件开发和投入运行前的各阶段，对软件的需求分析、设计规格说明和程序编码等过程的阶段性隐藏的错误或缺陷的最终检查，是软件质量保证的关键过程。通常，对软件测试的描述性定义有以下两种。

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（输入数据及其预期的输出结果），并利用这些测试用例运行程序以及发现错误的过程，即执行测试步骤。

1.1.3　软件测试的产生与发展

软件测试在 20 世纪 60 年代后正式建立。1961 年，一个简单的软件错误导致了美国大力神洲际导弹助推器的毁灭，致使美国空军强制要求在其后的关键性发射任务中，必须进行独立的测试验证，从而建立了软件验证与确认的方法论，软件测试就此正式产生。

随着软件迅速发展和广泛而深入应用于人类社会与生活各领域，系统规模和复杂性与日俱增，其错误产生的概率大为增加。软件存在的缺陷与故障所造成的损失也在不断发生，一些重要软件系统，如航空航天与高速列车的自动控制软件、银行结算系统与证券交易系统、国家军事防御系统、核电站的安全控制系统，以及涉及公众的交通订票系统、电子商务系统、

生命科学和医疗诊断系统等，因软件系统的质量问题，可能会造成严重损失或带来灾难性的后果。当今，在信息社会的生态系统中，软件质量问题已成为所有开发软件和应用软件者关注的焦点。软件是人脑智能化的一种典型体现，并以思维逻辑的形式呈现为一种“抽象产品”，并且具有生命周期的特征，从而有别于其他科技、生产领域及其产品的形态。软件的这个特性，使其“与生俱来”就可能存在着缺陷，且不易被发现或难以彻底根除。

软件具有“看不见，摸不着”的非有形产品的特征，从而有别于其他传统工业产品的外形特征，因此，软件的质量无法或难以采用传统的工业品的检验方法。在软件开发中的中间（过程）产品及最终产品的质量检验则更为复杂和困难。

软件工程的几十年的发展里程表明，对软件缺陷或错误的检验，预防软件运行发生故障的最有效的措施，就是通过软件测试来发现缺陷或错误，从而控制其质量。

软件测试始终都是软件质量理论研究与工程实践的重要内容。

20 世纪 60 年代，在软件工程建立之初，软件测试是为表明程序的正确性而进行的一项工作。1972 年，在美国北卡罗来纳（North Carolina）大学举行了首次以软件测试为主题的正式学术会议。1973 年，Bill Hetzel 给出软件测试的第一个定义：“软件测试就是对程序能够按预期的要求运行建立起的一种信心。”1975 年，John Good Enough 和 Susan Gerhart 在 IEEE 发表了《测试数据选择的原理》一文，从而使软件测试开始被确定为软件领域的一个新的研究方向与工程实践。1979 年，Glenford Myers 在《软件测试的艺术》一书中提出软件测试的目的是证伪，即“测试是以发现错误为目的而运行的程序或系统的执行过程”。在这个阶段，对软件测试的理解是：“软件测试用于验证软件（产品）能否正确工作并符合要求。”

20 世纪 80 年代后期，全球软件业迅速发展，软件规模越来越大，复杂程度越来越高。如研发的各种操作系统、大型商务软件、航天飞船控制系统、复杂工业流程控制系统等。在此阶段，某些系统的软件开发团队人员达到了几千或上万人的规模，程序也达到了几十万乃至几千万行的数量级。此时，软件的开发成本、效率和质量受到高度的重视，整个过程需要控制。同时，在该阶段，软件测试理论研究和技术应用也得到发展，其最主要的表现就是软件测试定义发生改变。软件测试不再单纯为一个“发现程序缺陷和错误的过程”，而且开始包含对软件质量评价的内容，软件测试被作为软件质量保证的一个重要手段，用以控制、保障和评价软件的质量，并为此制定了软件测试工程与技术的标准。1983 年，Bill Hetzel 在《软件测试完全指南》一书中提出：“测试是以评价一个程序或系统属性为目标的任何一种活动，测试是对软件质量的度量。”与此同时，IEEE 对软件测试的定义是“使用人工或自动的手段来运行或测量软件系统的过程，目的是检验软件系统是否满足规定的要求，并找出与预期结果之间的差异。软件测试是一门需要经过设计、开发和维护等完整阶段的软件工程。”从此，软件测试进入新的发展时期，成为软件领域的专门学科，并开始形成较完整的理论体系与技术方法，测试已具有高度的独立性，测试被正式列入软件工程范畴，并逐渐实现工程化。

进入 20 世纪 90 年代后，软件工程发展迅速，形成各种各样的软件开发模式，同时关于软件质量的研究和技术实践也不断被理论化和工程化，软件开发过程得到规范性和约束性的要求，软件测试与之相辅相成，测试技术规范和软件质量度量建立和逐步形成。1996 年，建立了软件测试能力成熟度模型（Testing Capability Maturity Model，TCMM），其后，软件业界又提出和制定了软件测试支持度量模型（Testability Support Model，TSM）、软件测试成熟度模型（Testing Maturity Model，TMM）和 ISO/IEC 9126 软件质量模型等一系列技术规范及质量标准。与此同时，开始产生与开发了软件测试工具，并开始在测试工程实践中运用测试工具，以辅助手工（人工）测试，加强测试力度和提高测试效率，开始探索软件测试的自动化形式与实施。

21 世纪后，软件测试应用促进了其理论的进一步发展与技术应用的深入。Rick 和 Stefan 在《系统的软件测试》中对软件测试做了进一步的定义："测试是为了度量和提高被测软件的质量，是对被测软件进行工程设计、实施和维护的整个生命周期过程。"新的软件测试理论、测试策略、测试技术、测试领域应用不断涌现，测试活动渗透和深入到各类软件系统中，如基于模型的测试、Web 应用系统的测试、嵌入式系统的软件测试、游戏软件测试等。由此带来测试活动及过程对软件开发技术的相互影响与作用反馈，以及对软件测试的重新评价。

近年来，软件测试与软件开发由相对独立性逐渐开始出现既独立又融合的特性，这是一个显著的变化。开发人员将承担软件测试的责任，同时，测试人员也将更多参与测试代码的开发工作，软件开发与测试的边界变得十分清晰，但过程又融为一体。以敏捷开发模式为代表的新一代软件开发模式，产生和融入了软件开发的新思想、新模式、新策略。极限编程、测试驱动、角色互换、团队模式等，在一流先进软件企业探索和实施，并赢得众多软件开发团队的青睐，不断获得成功。例如，测试驱动开发技术（Test-Driven Development，TDD）就是将测试作为开发工作起点和首要任务的一项新方法，是敏捷开发的一项核心实践和技术，也是一种新的设计方法论。TDD 的基本原则是通过测试来推动整个开发的进行，在开发功能代码之前，先编写单元测试用例的代码，测试代码确定了需要编写什么样的软件代码。TDD 测试驱动的开发技术不仅仅单纯运用于测试工作，而是把需求分析、设计与质量控制量化进行一体化的全过程。

软件测试模型、测试方法和测试服务模式等，也是进入 21 世纪软件测试研究的主要内容与方向。基于测试模型研究与应用，基于云计算、大数据的测试，基于 Web 2.0 软件的测试，基于安全性的测试，基于虚拟技术创建、维护、优化测试环境，乃至测试执行等都成为软件测试领域的新热点、新应用。

对测试质量的衡量已从计算缺陷数量、测试用例数量转到需求覆盖、代码覆盖方面；基于模型的软件测试针对软件中一些常见模型而提出，例如，软件模型分为故障模型、安全漏洞模型、差性能模型、并发故障模型、不良习惯模型、代码国际化模型和诱骗代码模型等。基于模型的测试机理首先提出软件模型，然后通过检测算法对其进行检测。若检测算法是完全的，则能从软件中排除问题。

软件测试的重要性和理论、技术体系和应用的发展，促使软件企业中产生了软件测试的专门组织与机构，组织形式与结构得到规范，测试策略与技术得到更新，自动化测试程度得到提高与完善。与此同时，产生了专门从事软件测试专业工作的机构或企业，使软件测试工作呈现出职业化的特征。2003 年，由德、英、美等国的软件测试工作者分别成立了各自国家的软件测试专业委员会，并在此基础上，成立了国际软件测试专业认证委员会（International Software Testing Qualifications Board，ISTQB），到 2009 年，该组织已扩大到 40 多个国家，成为专门制定软件测试规范标准、开展技术咨询、进行测试专门人才认证与指导的国际性专业组织机构。

软件测试大致经历了软件调试、独立的软件测试、首次被定义、成为专门学科与技术、与软件开发实现融合几个重要阶段。模型软件测试的产生和运用促进了软件测试的学科理论与技术运用的快速发展，新的软件测试理论、新的软件测试策略与方法、新的软件测试技术应用在不断地创新和涌出，软件测试已成为软件领域的专门学科，其测试应用与实践蓬勃发展，软件企业已建立软件测试的专门组织和机构，同时伴随着软件测试工作的专业化和职业化。

软件测试的发展大致经历了如图 1-2 所示的几个重要阶段。

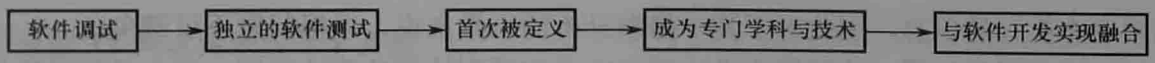


图 1-2　软件测试的发展历程

1.1.4　软件测试的发展趋势

1．软件测试领域的动态变化

近十年，软件测试学科研究和技术研发得到快速发展，主要表现为：有了比较完善的针对不同软件系统和软件类型的测试解决方案与测试方法论；研发出的软件测试工具的“智能化”程度越来越高；软件测试的组织形式、测试策略与测试技术不断地发生变化。著名软件专家 Harry Robinson 在 2004 年曾预测，软件测试领域将会发生下列变化。

（1）软件需求工程师、开发工程师将成为软件测试团队成员，并与测试人员合作。

（2）测试方法将日臻完善，Bug 的预防和早期检查将成为测试工具的主流。

（3）通过仿真工具模拟软件正式运行环境进行测试。

（4）对测试用例的更新将变得更为容易（以适应软件需求的变更）。

（5）自动化测试将由机器替代人做更多的工作。

（6）测试执行和测试开发的界限将变得模糊。

（7）对测试质量的衡量将从计算缺陷数量、测试用例数转为需求覆盖、代码覆盖方面。

某些预见，在 2009 年就已实现，如对软件模型的研究取得重大突破，基于模型的软件测试工具应运而生，等等。

2．基于模型的软件测试技术

基于模型的软件测试技术是针对软件中的常见软件模型而提出的一种测试技术。

针对软件模型的分类如下。

（1）故障模型：会引起软件错误或故障的常见模型。

（2）安全漏洞模型：为非法者攻击软件提供的可能性。

（3）差性能模型：软件动态运行时效率低下。

（4）并发故障模型：针对多线程编码。

（5）不良习惯模型：因编码的不良习惯而造成的一些错误。

（6）代码国际化模型：存在于以不同语言进行国际化过程中。

（7）诱骗代码模型：容易引起歧义、迷惑人的编写方式。

基于模型的测试机理：首先提出软件模型，然后通过检测算法进行检测，若检测算法结果符合质量的要求，则能从软件中排除该类模型。基于模型的软件测试工具将能够自动检测软件中的故障，并在对一些大型商业软件和开源软件的测试中发现大量的、以往测试并没有发现的一些软件故障及存在的隐患。例如，IBM AppScan，可自动测试 Web 应用软件系统中的编程安全漏洞。

基于模型的测试技术的优势：

（1）工具自动化程度高，测试效率高，测试过程所需时间较短。

（2）基于模型的测试技术往往能发现其他测试方法难以发现的故障。

基于模型的测试技术存在的不足：

（1）存在误报问题。通常基于模型的测试技术属于静态分析技术，而某些软件故障的确

定需要动态执行的信息，因此对于基于静态分析的工具来说，误报问题不可避免。

（2）存在漏报问题。该问题主要由模型定义和模型检测的算法引起，因为目前对软件模式还无规范、统一与形式化的定义。

（3）模型呈多样性。由于软件模型的多种多样，很难用一种或几种策略方法适应多种模型。

预见软件测试模型、测试方法、测试服务模式在未来将成为软件测试研究的主要内容与方向。

1.2 软件测试基础知识与理论

1.2.1 软件测试的目的与原则

1. 软件测试的目的

软件测试的目的是发现软件存在的故障或缺陷，并借此对软件的质量进行度量。为达此目的，测试活动的目标是尽最大可能找出最多的错误。测试是从软件含有缺陷和故障的假设而进行的，实现这个目标的关键是科学、合理地设计出最能暴露问题的测试用例。

（1）测试是程序执行过程，并限于执行处理有限的测试用例与情形且发现了错误。

（2）检测软件是否满足软件定义的各种需求目标。

（3）执行的测试用例发现了未曾发现的错误，实现成功的测试。

2. 软件测试原则

依据软件测试的目的，有以下测试原则。

（1）尽早地和及时地进行测试。测试活动应从软件产品开发的初始阶段就开始。

（2）测试用例要由测试数据与预期结果两个部分组成，并包括测试前置条件或后置条件。

（3）测试根据其需求和风险，可由专业测试人员进行或程序开发者自行检测。

（4）需要严格执行测试计划，并排除测试工作随意性。

（5）充分注意测试中的集群效应，经验表明软件约 80%的错误仅与 20%的程序有关。

（6）应对测试结果做核查，存档测试计划、测试用例、缺陷统计和分析报告等文档，为软件维护提供资料及条件。

1.2.2 软件测试的基本原理与特性准则

软件测试发展史已达 40 多年。经过长期实践，总结归纳出了一些测试的基本原理与特性准则，并被业界普遍接受和遵循，对测试的设计、执行和管理均具有工程指导意义。

1. 测试的基本原理

【原理 1】测试可以证明缺陷存在，但不能证明缺陷不存在

测试可以证明软件系统（产品）是失败的，即说明软件中有缺陷，但测试不能证明软件中没有缺陷。适当的软件测试可以减少测试对象中的隐藏缺陷。即使在测试中没有发现失效，也不能证明其没有缺陷。

【原理 2】穷尽测试是不可能的

测试若考虑所有可能的输入值及其组合，并结合所有的前置条件进行穷尽测试是不可能的。在实际测试过程中，软件测试基本上是抽样测试。因此，必须根据风险和优先级，控制测试工作量。

【原理 3】测试活动应尽早开始

在软件生命周期中，测试活动应尽早实施，并聚焦于定义的目标上，这样可以尽早地发现缺陷。

【原理 4】缺陷集群性

在通常情况下，缺陷并不是平均的，而是集群分布的，大多数的缺陷只存在于测试对象的极小部分中。因此，如在一个地方发现了较多缺陷，通常在附近会有更多的缺陷，这就是所谓的缺陷集群性，也就是经常所说的“80/20 现象”，80%的缺陷集中在 20%的程序模块中。因此，在测试中，应机动灵活地应用这个原理。

【原理 5】杀虫剂悖论

若同样的测试用例被一再重复执行，则会减少测试的有效性。先前没有发现的缺陷反复使用同样的测试用例也不会被重新发现。因此，为了维护测试的有效性，战胜这种“抗药性”，应对测试用例进行修正或更新。这样，软件中未被测试过的部分或先前没有被使用过的输入组合会被重新执行，从而发现更多的缺陷。

【原理 6】测试依赖于测试内容

测试必须与应用系统的运行环境及使用中固有的风险相适应。因此，对于存在差异的两个系统可以用完全相同的方式进行测试。对于每个软件系统，测试出口准则等应依据其使用的环境分别量体定制。例如，对安全起关键作用的系统与一个电商应用系统所要求的测试是不尽相同的。

【原理 7】没有发现失效就是有用的系统的说法是一种谬论

测试找到了导致失效的 Bug 且修正了缺陷，并不能保证整个系统达到了用户的预期要求和需要。因此，没有发现失效就是有用的系统的说法是一种谬论。

2．测试的特性准则

（1）对任何软件（产品）系统都存在有限的充分测试集合。

（2）若一个软件系统在一个测试数据（测试用例）集合上的测试是充分的，那么再执行一些测试用例也是充分的，这一特性称为测试的单调性。

（3）即使对软件所有的组成成分都进行了充分测试，也并不能表明整体软件系统的测试已经充分，这一特性称为测试的非复合性。

（4）即使对软件系统整体的测试是充分的，也并不能证明软件系统中各组成成分都已得到了充分测试，这个特性称为测试的非分解性。

（5）软件测试的充分性应与软件需求和软件实现相关。

（6）软件越复杂，测试数据就越多，这一特性称为测试的复杂性。

（7）测试越多，进一步测试所获充分性增长就越少，这一特性称为测试回报递减率。

（8）软件测试的特性准则对测试的设计与执行均具有工程指导意义。

1.2.3 软件测试的基本策略

软件具有生命周期的特性。软件测试贯穿整个软件生命周期，因此，测试的基本策略是在其生命周期的每个阶段中确定测试目标、确认测试对象、建立测试生命周期、制定和实施测试策略、选择测试类型和运用测试方法 6 项。

1. 确定测试目标

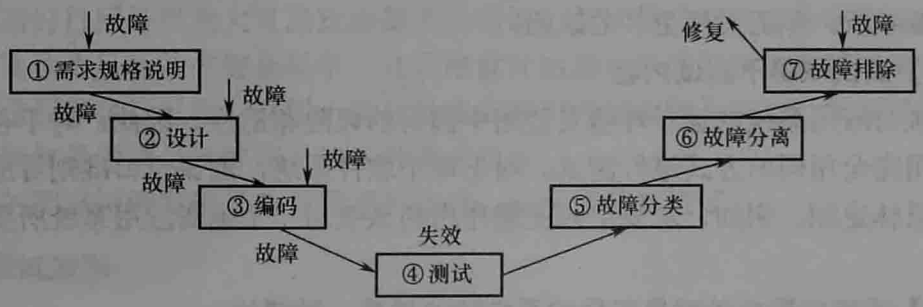
根据软件生命周期划分的几个阶段，以及软件质量包含的各项属性特征，测试需要对每一阶段和每个部分的目标进行确定。

2. 确认测试对象

软件测试是对程序的测试，并贯穿于软件生命周期整个过程。因此，软件开发过程中产生的需求分析、概要设计、详细设计以及编码等各个阶段所获文档，如需求规格说明、概要设计规格说明、详细设计规格说明以及源程序等都是软件测试的对象。

3. 建立测试生命周期

软件测试生命周期包括在软件生命周期中。测试生命周期从大的方面看，主要横跨两个历程，分为软件生产阶段的测试历程和软件运行维护阶段的测试活动。软件测试生命周期在软件生产阶段的测试活动及运行维护阶段的测试活动过程如图 1-3 所示。



注：①～③ 可能引入故障或导致其他阶段故障；④ 测试发现了失效；⑤～⑦故障排除。故障排除过程有可能使原本正确执行的程序又出现错误，即排除旧故障、引入了新故障。

图 1-3　软件测试生命周期模型

4. 制定和实施测试策略

制定和实施测试策略包含以下四项内容。

（1）确定测试由谁执行。在软件产品开发中，通常有开发者和测试者两种角色。开发者通过开发形成产品，如分析、设计、编码调试或文档等可交付物。测试者通过测试检测产品中是否存在缺陷，包括根据特定目的而设计的测试用例、测试过程构造、执行测试和评价测试结果。通常的做法是，开发者负责完成组件级别的测试，而集成级别和系统级别的测试则由独立的测试人员或专门测试机构完成。但也可有其他策略，如在组件级别的测试中，专门测试人员加入。

（2）确定测试什么。测试经验表明，通常表现在程序中的故障不一定由编码引起。它可能由软件详细设计、概要设计阶段，甚至需求分析阶段的问题所致。要排除故障、修正错误必须追溯到前期工作。事实上，软件需求分析、设计和实施阶段是软件故障的主要来源。

（3）确定何时进行测试。确定测试是与开发并行的过程，还是在开发完成某个阶段任务之后的活动或是在整个开发结束后的活动。软件开发经验和事实证明，随着开发过程深入，越是在早期没有进行测试的模块对整个软件的潜在隐患及破坏作用就越明显。

（4）确定怎样进行测试。软件的“规格说明”界定了软件本身应达到的目标，而程序“实现”则是对应各种输入并如何产生输出的一种算法，即规格说明界定软件要做什么，而程序实现表达了软件怎样做。确定怎样进行测试，也就是要根据软件“规格说明”和程序实现的对应关系，确定采用何种测试策略与技术方法，设计并生成有效测试用例，对其进行检验。