实例讲解
实训强化
培养技能
面向就业

全国高等职业教育计算机类规划教材 · 实例与实训教程系列

软件测试技术基础教程

◎ 顾海花　主编
◎ 雷　雁　史海峰　副主编

- 紧密结合软件测试工作岗位要求
- 以通俗易懂的语言讲述软件测试基础理论
- 通过经典实用的测试案例传授实战技术
- 详细介绍常用的测试工具

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

全国高等职业教育计算机类规划教材·实例与实训教程系列

软件测试技术基础教程

顾海花　主　编
雷　雁　史海峰　副主编

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

本书全面系统地介绍了软件测试理论及应用技术，内容全面实用，讲述浅显易懂。突出了软件测试的基础知识和理论的阐述，软件测试的发展脉胳及其与软件开发最新技术的结合和运用，使读者可以尽快地掌握软件测试的基础知识，并了解软件测试的最新动态。

本书既可作为高等院校软件测试课程的教材，也可作为软件测试爱好者的自学用书。对于那些希望增强软件测试方面知识的程序员、软件项目经理和软件开发团队的其他人员，本书也具有很好的参考价值。本书可作为高职、高专软件测试专业及计算机软件开发专业课程的教材，也可作为软件测试技术应用培训的基础教材，并供从事软件测试和开发行业的技术人员学习参考。

图书在版编目（CIP）数据

软件测试技术基础教程/顾海花主编.—北京：电子工业出版社，2011.8
全国高等职业教育计算机类规划教材·实例与实训教程系列
ISBN 978-7-121-13705-1

Ⅰ. ①软… Ⅱ. ①顾… 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 101186 号

策划编辑：程超群
责任编辑：郝黎明　　文字编辑：裴　杰　　特约编辑：罗树利
印　　刷：北京东光印刷厂
装　　订：三河市鹏成印业有限公司
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：18.75　字数：480 千字
印　　次：2011 年 8 月第 1 次印刷
印　　数：4 000 册　　定价：31.00 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

本书全面系统地介绍了软件测试基础理论及其应用技术，并介绍了软件测试的发展脉络及其与软件开发最新技术的结合和运用。本书旨在使读者可以很快地掌握软件测试的基础知识，引领读者进入软件测试这个新的领域，并了解软件测试的最新动态，对它有一个全面的认识。

本书重在培养读者软件测试工作的实践能力，适应软件企业的工作环境和业界标准，并和国际先进的软件开发理念和测试技术保持同步；通过本书的学习，读者应该了解并掌握软件产品质量保证的基本思想和科学体系，软件测试过程和策略，软件测试的方法、技术和工具的使用，为全面掌握软件技术和软件项目管理打下坚实的基础。

全书共分为两大部分，第 1 部分为软件测试基础理论部分，包括第 1～10 章，综合介绍了以下几个方面的内容：软件测试的发展历史和现状；软件测试的定义、目标、原则和分类，及其与软件质量保证的关系；软件开发过程和开发模式；白盒测试技术；黑盒测试技术；软件自动化测试；软件测试计划、文档及测试用例；面向对象的软件测试；Web 网站测试；软件测试技术前沿。这部分内容论述浅显易懂，知识点全面实用。特别是在白盒、黑盒测试技术介绍中运用了大量的案例来进行阐述。最后介绍了软件领域中的新技术对软件测试提出的新要求。第 2 部分为软件测试工具实践，包括第 11～13 章，介绍了 JUnit、WAS、WinRunner 等自动化测试工具的使用。

本书由顾海花担任主编，雷雁、史海峰担任副主编。其中第 3、4、5、6、9、10、12、13 章由顾海花编写，第 1、2、11 章由雷雁编写，第 7、8 章由史海峰编写，最后由顾海花负责统稿。同时，在本书的编写过程中，聂明院长给予了全力的支持和帮助，季飞主任也提出了宝贵的意见，在此表示衷心的感谢！

参加本书编写的人员均为高职院校的教学骨干，并将多年积累的具有实用价值的经验、知识点和操作技巧等毫无保留地奉献给广大读者。为了方便教师教学，本书配有教学演示文稿供下载使用。

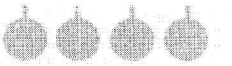
本书在编写过程中，参考和引用了大量专家学者的论著和研究资料，作者已经尽可能在参考文献中列出，谨在此向他们表示由衷的感谢。由于时间仓促和水平所限，书中难免存在一些不足之处，欢迎广大读者批评指正，联系信箱是 guhh@dma800.com。

编　者

目　录

第1部分　软件测试基础理论

第 2 部分　软件测试工具实践



第1部分

软件测试基础理论

第1章　软件测试概述

1.1　软件测试的背景

随着软件产业的日益发展，软件系统的规模和复杂性与日俱增，软件的生产成本和软件中存在的缺陷故障造成的损失也大大增加，甚至会带来灾难性的后果。软件产品不同于其他科技和生产领域的产品，它是人脑的高度智力化的体现，由于这一特殊性，软件与生俱来就有可能存在着缺陷。

在开发大型软件系统的漫长过程中，面对纷繁复杂的各种现实情况，人的主观认识和客观现实之间往往存在着差距，开发过程中各类人员之间的交流和配合也往往并不是尽善尽美的。

如果不能在软件正式投入运行之前发现并纠正这些错误，那么这些错误最终必然会在软件的实际运行过程中暴露出来。到那时，改正这些错误不仅要付出很大的代价，而且往往会造成无法弥补的损失。

软件的质量就是软件的生命，为了保证软件的质量，人们在长期的开发过程中积累了许多经验并形成了许多行之有效的方法。但是借助这些方法，我们只能尽量减少软件中的错误和不足，却不能完全避免所有的错误。

如何防止和减少这些可能存在的问题呢？答案是进行软件测试。测试是最有效的排除和防止软件缺陷与故障的手段，并由此促进了软件测试理论与技术实践的快速发展。新的测试理论、测试方法、测试技术手段在不断涌出，软件测试机构和组织也在迅速产生和发展，由此软件测试技术职业也同步完善和健全起来。

1.1.1　软件测试发展历史

软件测试是伴随着软件的产生而产生的。在软件行业发展初期，软件规模较小，复杂程度较低，软件开发的过程比较混乱、相当随意。这一阶段还没有系统意义上的软件测试，更多的是一种类似调试的测试，测试用例的设计和选取也都是根据测试人员的经验随机进行的，大多数测试的目的是为了证明系统可以正常运行。当时对测试的投入较少，测试介入的也较晚，一般是等到代码形成，产品已经基本完成才进行测试。

直到20世纪50年代后期，随着计算机软件的发展，用于计算机编程的各种高级语言相继诞生，程序的复杂性远远超过了以前，测试的重点也逐步转入到使用高级语言编写的软件系统中来。尽管如此，由于受到硬件的制约，在计算机系统中，软件仍然处于次要位置。软件正确性的把握仍然主要依赖于编程人员的技术水平，测试活动始终落后于开发活动。因此，这一时期软件测试的理论和方法发展比较缓慢。

到了20世纪70年代以后，计算机处理速度迅猛提高，存储器容量快速增加，软件在整个计算机系统中的地位也越来越重要。随着软件开发技术的成熟和完善，软件的规模也越来越大，复杂度也大大增加。因此，软件的可靠性面临着前所未有的危机，给软件测试工作带

来了巨大的挑战，很多测试理论和测试方法应运而生，逐渐形成了一套完整的体系，也涌现了一批出色的软件测试宗师。

1972 年，软件测试的先驱者 Bill Hetzel 博士（代表著“The Complete Guide to Software Testing”）在 North Carllina 大学举行了第一次以软件测试为主题的正式会议。此后软件测试的会议就如雨后春笋般出现。1981 年，Bill Hetzel 博士开设了一门公共课“结构化软件测试”（Structured Software Testing）。1983 年，他将软件测试定义为“评价一个程序和系统的特性或能力，并判断它是否达到预期的结果，软件测试就是以此为目的的任何行为”。他的思想的核心观点是：测试方法是试图验证软件是“工作的”，所谓“工作的”就是指软件的功能是按照预先的设计执行的，以正向思维，针对软件系统的所有功能点，逐个验证其正确性。软件测试业界把这种方法看做软件测试的第一类方法。

软件测试的第二类方法代表人物是 Glenford J. Myers（代表论著“The Art of Software Testing”）。他认为测试不应该着眼于验证软件是工作的，相反应该首先认定软件是有错误的，然后用逆向思维去发现尽可能多的错误。他还从人的心理学的角度论证，如果将 “验证软件是工作的”作为测试的目的，非常不利于测试人员发现软件的错误。1979 年，他提出了对软件测试的定义：“测试是为发现错误而执行的一个程序或系统的过程。（The process of executing a program or system with the intent of finding errors）”。这个定义，也被业界所认可，经常被引用。Myers 认为，测试是验证软件是“不工作的”，或者说是有错误的；一个成功的测试必须是发现 Bug 的测试，不然就没有价值。这就如同一个病人（假定此人确有病），到医院做一项医疗检查，结果各项指标都正常，那说明该项医疗检查对于诊断该病人的病情是没有价值的，是失败的。Myers 提出的“测试的目的是证伪”这一概念，推翻了过去“为表明软件正确而进行测试”的错误认识，为软件测试的发展指出了方向，软件测试的理论、方法在之后得到了长足的发展。第二类软件测试方法在业界也很流行，受到很多学术界专家的支持。

然而，对 Glenford J. Myers 先生“测试的目的是证伪”这一概念的理解也不能太过于片面。在很多软件工程学、软件测试方面的书籍中都提到一个概念：“测试的目的是寻找错误，并且是尽最大可能找出最多的错误”。这很容易让人们认为测试人员就是“挑毛病”的，而由此带来诸多问题。大家熟悉的 Ron Patton 在《软件测试》一书中有一个明确而简洁的定义：“软件测试人员的目标是找到软件缺陷，尽可能早一些，并确保其得以修复。”这样的定义具有一定的片面性，带来的结果是：若测试人员以发现缺陷为唯一目标，而很少去关注系统对需求的实现，测试活动往往会存在一定的随意性和盲目性；若有些软件企业接受了这样的方法，以 Bug 数量来作为考核测试人员业绩的唯一指标，也不太科学。

总的来说，第一类测试可以简单抽象地描述为这样的过程：在设计规定的环境下运行软件的功能，将其结果与用户需求或设计结果相比较，如果相符则测试通过，如果不相符则视为 Bug。这一过程的终极目标是将软件的所有功能在所有设计规定的环境全部运行，并通过。在软件行业中一般把第一类方法奉为主流和行业标准。第一类测试方法以需求和设计为本，因此有利于界定测试工作的范畴，更便于部署测试的侧重点，加强针对性。这一点对于大型软件的测试，尤其是在有限的时间和人力资源情况下显得格外重要。

而第二类测试方法与需求和设计没有必然的关联，更强调测试人员发挥主观能动性，用

逆向思维方式，不断思考开发人员理解的误区、不良的习惯、程序代码的边界、无效数据的输入及系统的各种弱点，试图破坏系统、摧毁系统，目标就是发现系统中各种各样的问题。这种方法往往能够发现系统中存在的更多缺陷。

在产业界，从20世纪70年代后期到20世纪80年代中期，很多软件企业成立了QA或者SQA部门。后来QA的职能转变为流程监控（包括监控测试流程），而测试（Testing）则从QA中分离出来成为独立的组织职能。

到了20世纪80年代初期，软件和IT行业进入了大发展时代，软件趋向大型化、高复杂度，软件的质量越来越重要。这时，一些软件测试的基础理论和实用技术开始形成，并且人们开始为软件开发设计了各种流程和管理方法，软件开发的方式也逐渐由混乱无序的开发过程过渡到结构化的开发过程，以结构化分析与设计、结构化评审、结构化程序设计及结构化测试为特征。人们还将“质量”的概念融入其中，软件测试定义发生了改变，测试不单纯是一个发现错误的过程，而且将测试作为软件质量保证（SQA）的主要职能，包含软件质量评价的内容，Bill Hetzel在《Complete Guide of Software Testing》一书中指出：“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。”这个定义至今仍被引用。软件开发人员和测试人员开始坐在一起探讨软件工程和测试问题。软件测试已有了行业标准（IEEE/ANSI），1983年IEEE提出的软件工程术语中给软件测试下的定义是：“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”。这个定义明确指出：软件测试的目的是为了检验软件系统是否满足需求。它再也不是一个一次性的、只是开发后期的活动，而是与整个开发流程融合成一体。软件测试已成为一个专业，需要运用专门的方法和手段，需要专门人才和专家来承担。

如今在软件产业化发展的大趋势下，人们对软件质量、成本和进度的要求也越来越高，软件质量的控制已经不仅仅是传统意义上的软件测试。传统软件的测试大多是基于代码运行的，并且常常是软件开发的后期才开始进行，但大量研究表明，设计活动引入的错误占软件开发过程中出现的所有错误数量的50%～65%。因此，越来越多的声音呼吁，要求有一个规范的软件开发过程。而在整个软件开发过程中，测试已经不再只是基于程序代码进行的活动，而是一个基于整个软件生命周期的质量控制活动，贯穿于软件开发的各个阶段。

1.1.2 软件测试的现状

在我国，软件测试目前还没有形成一个真正的产业，尚处于起步阶段，根据51testing组织得到的《2009年中国软件测试从业人员调查报告》可以看出：软件测试从业人员所在公司成立的时间在5年以上的比例为58%，集中分布在应用软件行业、电信/互联网服务行业，公司多为私营或集体所有制企业，且比例逐年增加。调查显示虽然国内IT软件开发企业对软件测试认识比较淡薄，公司测试人员与开发人员的比例主要分布在1∶3～1∶4之间，较国外1∶1比例相距甚远，但是国内IT企业也逐步开始重视软件测试团队的建设，在IT企业中，一些知名企业已经将软件测试作为企业未来发展的一个板块，参与2009年调查的软件测试从业人员中，73%的人所在公司具有独立的测试部门，专职测试人员也呈逐年上升趋势。但是，在国内，现在在软件测试行业中各种软件测试的方法、技术和标准都还在探索阶段。

总之，国内软件行业普遍规模偏小，缺乏大型软件产品经验，开发过程不够规范，这决定了国内软件测试行业与一些发达国家相比还存在一定的差距。其实，这与中国整体软件的发展水平是一致的，因为我国整体的软件产业水平和软件发达国家的水平相比有较大的差距，而作为软件产业重要一环的软件测试，必然也存在着不小的差距。但是，我们在软件测试实现方面并不比国外差，国际上优秀的测试工具，我们基本都有，这些工具所体现的思想我们也有深刻的理解，很多大型系统在国内都得到了很好的测试。

1.2 软件缺陷

1.2.1 软件缺陷案例分析

软件是由人编写开发的，是一种逻辑思维的产品，尽管现在软件开发者采取了一系列有效措施，不断地提高软件开发质量，但仍然无法完全避免软件（产品）会存在各种各样的缺陷。软件中存在的缺陷有时会造成相当严重的损失和灾难。

下面以 4 个软件缺陷的案例来说明。

1. 美国迪士尼公司的狮子王游戏软件缺陷

1994 年秋天，美国迪士尼公司发布了一个面向儿童的多媒体游戏软件“狮子王动画故事书（The Lion King Animated Storybook）”。迪士尼公司为了打开儿童游戏市场，进行了大量促销宣传。结果，销售非常火爆，销售额非常可观，该游戏成为美国孩子们当年的“必买游戏”。然而好景不长，圣诞节过后，迪士尼公司的客户投诉电话便开始响个不停，愤怒的家长和玩不成游戏的孩子们对这款游戏软件的缺陷进行了大量的投诉。报纸和电视新闻进行了大量的报道。

后经调查证实，迪士尼公司在软件上市前没有将软件在市面上投入使用的许多不同类型的 PC 上进行广泛的测试，也就是说游戏软件对硬件的兼容性没有得到保证，造成了该软件只能在少数系统中正常工作，即在迪士尼程序员用来开发游戏的系统中可以正常运行，但在大多数公众使用的系统中却不能运行。该软件故障使迪士尼公司声誉大损，并且为了改正软件缺陷付出了沉重的代价。

2. 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突然失踪。事故评估委员会对这起事故进行了调查，认定出现故障的原因极可能是一个数据位被意外置位。而该问题本应该在内部测试时就被发现并予以解决。

从理论上看，着陆的过程是这样的：当探测器向火星表面降落时，它将打开降落伞减缓探测器的下降速度。降落伞打开几秒后，探测器的 3 条腿将迅速撑开，并锁定位置，准备着陆。当探测器离地面 1 800 m 时，它将丢弃降落伞，点燃着陆推进器，缓缓地降落到地面。

然而美国航天局为了节省研制经费，简化了确定何时关闭着陆推进器的装置。为了替代其他太空船上使用的贵重雷达，他们在探测器的脚部安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。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。

遗憾的是，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，计算机极有可能关闭着陆推进器，这样火星极地登录者号探测器飞船下坠 1 800 m 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。登录探测器在发射前经过了多个小组测试，其中一个小组测试飞船的脚折叠过程，另一个小组测试此后的着陆过程。前一个小组不去注意着陆数据是否置位，因为这不是他们负责的范围；后一个小组总是在开始测试之前复位计算机，清除数据位。双方独立工作都做得很好，但没有在一起进行集成测试，使系统中的衔接问题隐藏起来，最终导致了灾难性事故的发生。

3．北京奥运门票被迫暂停销售

2007 年 10 月 30 日上午 9 点，北京奥运会门票面向境内公众的第二阶段预售正式启动。然而，为让更多的公众实现奥运梦想“先到先得，售完为止”的销售政策适得其反，公众纷纷抢在第一时间订票，致使票务官网压力激增，承受了超过自身设计容量 8 倍的流量，导致系统瘫痪。为此，北京奥组委票务中心对广大公众未能及时、便捷地实现奥运门票预订表示歉意，同时宣布奥运门票暂停销售 5 天。

为此次门票销售提供技术系统平台的是北京歌华特玛捷票务有限公司。该公司工作人员透露，此次票务官网设计的流量容量是每小时 100 万次，但瞬间承受了每小时 800 万次的流量压力，访问数量过大造成网络堵塞，技术系统应对不畅，造成很多申购者无法及时提交申请，所以系统在启动不久就出现了处理能力不足的问题。

Web 压力测试可以有效地测试一些 Web 服务器的运行状态和响应时间等，对于 Web 服务器的承受力测试是个非常好的手法。如果在网站建设时就进行压力测试，并考虑网站在访问量过大时如何保证网站的正常运行，就会增强网站的鲁棒性。

4．诺基亚 Series 40 手机平台存在缺陷

2008 年 8 月诺基亚承认该公司销售量超过 1 亿部的 Series 40 手机平台存在严重缺陷：旧版 J2ME 中的缺陷使黑客能够远程访问本应受到限制的手机功能；Series 40 中的缺陷使黑客能够秘密地安装和激活应用软件。Series 40 是一款非常普及的手机平台，主要用于诺基亚的低端手机。这些问题给数量众多的手机用户造成潜在的重大威胁。

从上述案例中可以看到软件缺陷就在我们身边，它普遍存在，并将对产品造成不同程度的危害，甚至对用户的使用产生灾难性的影响。

1.2.2 软件缺陷的定义

对于软件存在的各种问题，在软件工程或软件测试中都可以称为软件缺陷或软件故障。作为软件测试员，可能发现的缺陷大多数都不如上面所列举的实例那么明显，而软件测试员的任务就是要发现软件中所隐藏的错误，其中的一些简单而细微的错误，很难做到真正区分哪些是真正的错误，哪些不是，这对于软件测试员是一个最大的挑战。

软件缺陷即计算机系统或程序中存在的任何一种破坏正常运行能力的问题、错误，或者隐藏的功能缺陷、瑕疵。缺陷会导致软件产品在某种程度上不能满足用户的需要。对于软件缺陷的精确定义，通常有以下 5 条描述：

① 软件未达到产品说明书要求的功能。

② 软件出现了产品说明书指明不会出现的错误。

③ 软件功能超出了产品说明书规定的功能。

④ 软件未实现产品说明书虽未明确指出但应该实现的目标。

⑤ 软件难以理解，不易使用，运行缓慢或者最终用户认为使用效果不好。

为了更好地理解上述 5 条描述，我们以计算器软件为例进行说明。

计算器的产品说明书声明它能够准确无误地进行加、减、乘、除运算。当你拿到计算器后，按下“+”键，结果什么反应也没有，根据第①条规则，这是一个缺陷。假如得到错误答案，根据第①条规则，这同样是一个缺陷。

若产品说明书声明计算器永远不会崩溃、锁死或者停止反应，当你任意敲键盘时，计算器停止接受输入，根据第②条规则，这是一个缺陷。

若用计算器进行测试，发现除了加、减、乘、除之外它还可以求平方根，说明书中从没提到这一功能，根据第③条规则，这是软件缺陷。软件实现了产品说明书未提到的功能。

若在测试计算器时，发现电池没电会导致计算不正确，但产品说明书未指出这个问题。根据第④条规则，这是个缺陷。

第⑤条规则是全面的。如果软件测试员发现某些地方不对劲，无论什么原因，都要认定为缺陷。如“＝”键布置的位置极其不好按；在明亮光下显示屏难以看清。根据第⑤条规则，这些都是缺陷。

美国商务部国家标准和技术研究所（NIST）进行的一项研究表明，软件中的缺陷每年给美国经济造成的损失高达 595 亿美元，说明软件中存在的缺陷所造成的损失是巨大的，从反面又一次证明软件测试的重要性。如何尽早彻底地发现软件中存在的缺陷是一项非常复杂、需要创造性和高度智慧的工作。同时，软件的缺陷是软件开发过程中的重要属性，反映软件开发过程中需求分析、功能设计、用户界面设计、编程等环节所隐含的问题，也为项目管理、过程改造提供了许多信息。

1.2.3 软件缺陷产生的原因

在软件开发的过程中，软件缺陷的产生是不可避免的。造成软件缺陷的原因有哪些？我们可以从软件本身、团队工作和技术问题等多个方面分析，将造成软件缺陷的原因归纳如下。

1. 软件本身

- ✧ 文档错误、内容不正确或拼写错误。
- ✧ 数据考虑不周全引起强度或负载问题。
- ✧ 对边界考虑不够周全，漏掉某几个边界条件造成的错误。
- ✧ 对一些实时应用系统，保证精确的时间同步，否则容易引起时间上不协调、不一致性带来的问题。
- ✧ 没有考虑系统崩溃后在系统安全性、可靠性方面的隐患。
- ✧ 硬件或系统软件上存在的错误。
- ✧ 软件开发标准或过程上的错误。

2. 团队工作

✧ 系统分析时对客户的需求不是十分清楚，或者和用户的沟通存在一些困难。

✧ 不同阶段的开发人员相互理解不一致，软件设计对需求分析结果的理解偏差，编程人员对系统设计规格说明书中某些内容重视不够，或存在着误解。

✧ 设计或编程上的一些假定或依赖性，没有得到充分地沟通。

3. 技术问题

✧ 算法错误。

✧ 语法错误。

✧ 计算的精度问题。

✧ 系统结构不合理，造成系统性能问题。

✧ 接口参数不匹配，导致模块集成出现问题。

软件缺陷是由很多原因造成的，将诸多原因如规格说明书、系统设计结果、编程的代码等归类起来比较后发现，规格说明书是软件缺陷出现最多的地方，如图 1-1 所示。

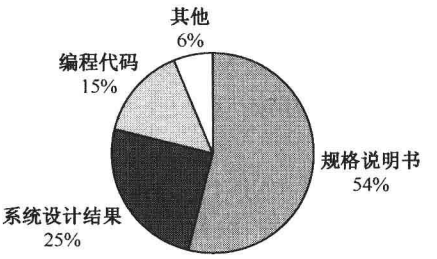


图 1-1　软件缺陷构成图

软件产品规格说明书为什么是软件缺陷存在最多的地方？主要原因有以下几种：

✧ 由于软件产品还没有设计、开发，完全靠想象去描述系统的实现结果，所以有些特性还不够清晰。

✧ 用户一般是非计算机专业人员，软件开发人员和用户的沟通存在较大困难，对要开发的产品功能理解不一致。

✧ 需求变化的不一致性。用户的需求总是在不断变化的，这些变化如果没有在产品规格说明书中得到正确的描述，容易引起前后文、上下文的矛盾。

✧ 对规格说明书不够重视，在规格说明书的设计和写作上投入的人力、时间不足。

✧ 没有在整个开发队伍中进行充分沟通，有时只有设计师或项目经理得到比较多的信息。

1.2.4　软件缺陷的修复费用

软件通常要靠有计划、有条理的开发过程来实现。从开始到计划、编程、测试，到公开使用的过程中，都有可能发现软件缺陷。软件缺陷造成的修复费用随着时间的推移呈指数级地增长。当早期编写产品说明书时发现并修复缺陷，费用可能只要 10 美元甚至更少。同样的缺陷如果直到软件编写完成开始测试时才发现，费用可能要 100～1 000 美元。如果是客户发现的，费用可能达到数千甚至数百万美元。图 1-2 所示为软件缺陷在不同阶段发现时错误

修正后的费用情况。

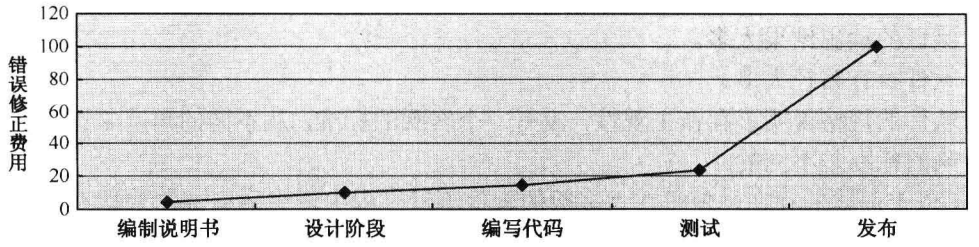


图 1-2　软件缺陷在不同阶段发现时错误修复的费用

以前面的迪士尼狮子王软件为例，它的问题的根本原因是软件无法在流行的家庭 PC 平台上运行。假如早在编写产品说明书时，有人已经研究过在普通家庭用什么型号的 PC，并且明确指出软件需要在该种配置上设计和测试，付出的代价小得几乎可以忽略不计。如果没有这样做，可以补救的措施是，软件测试员去搜集家庭流行 PC 样机并在其上验证。他们会发现软件缺陷，但是修复费用要高得多，因为软件必须调试、修改、再测试。开发小组还应当把软件的初期版本分发给一小部分客户试用进行 Beta 测试，那些被挑选出来代表庞大市场的客户可能会发现问题。然而实际的情况却是这个缺陷被完全忽视，根本没有进行这方面的测试，直到成千上万的光盘被压制和销售出去。迪士尼公司最终支付了客户投诉电话费、产品召回、更换光盘，以及又一轮调试、修改和测试的费用，付出了昂贵的代价。如果严重的软件缺陷保留到了客户那里，就足以耗尽整个产品的利润。

1.3　软件测试的复杂性与经济性分析

人们通常认为软件工程中程序的开发是一个复杂而困难的过程，需要投入大量的人力、物力和时间，而测试程序则比较容易，不需要花费太多的精力，并且通过测试可以找到所有的软件故障。这其实是人们对软件开发过程理解上的一个误区。在实际的软件开发过程中，软件测试作为现代软件开发工业的一个非常重要的组成部分，正扮演着越来越重要的角色。随着软件规模的不断扩大，如何在有限的条件下对被开发的软件进行有效的测试已经成为软件工程中一个非常关键的课题。

1.3.1　软件测试的复杂性

软件测试是一项细致并且需要具备高度技巧的工作，稍有不慎就会顾此失彼，发生不应有的疏漏。针对下面几个问题的分析充分说明了软件测试的复杂性。

1．不可能对程序实现完全测试

完全测试是不现实的，在实际的软件测试工作中，由于测试的数量极其巨大，不论采用什么方法，都不可能进行完全的测试。所谓完全的测试，就是让被测程序在一切可能的输入情况下全部执行一遍。通常也称这种测试为“穷举测试”。

穷举测试会引起以下几种问题：

- ✧ 测试所需要的输入量太大。
- ✧ 测试的输出结果太多。
- ✧ 软件执行路径太多。
- ✧ 软件的规格说明书存在主观性，没有一个客观的标准，从不同的角度来评判，软件缺陷的标准是不同的。

由于以上问题的存在，使得在大多数的软件测试过程中，穷举测试几乎是不可能的。以Windows 系统中最简单的计算器程序为例，假如要对它进行穷举测试，首先进行加法的测试，那就要输入 1+0=，1+1=，1+2=，…，计算器能处理的数字是 32 位，所以要一直输入到1+99 999…99 999（共 32 个 9）=。接下来，继续输入 2+0=，2+1=，2+2=，…，直到输入2+99 999…99 999（共 32 个 9）=。依次类推，加法的输入还在继续……

在软件的使用过程中，人们不仅要进行合法的输入，若出现某些意外情况，可能还要发生种种不合法的输入。所以测试人员既要测试所有合法的输入，还要对那些不合法但是可能的输入进行测试。比如，在测试加法计算时输入：1+a，jpkl+o9，jsfw+16，…，这样的测试情况可能出现无穷多个。

按照上述思路一个一个的测试，单是合法输入就接近无穷多个，使得在理论上根本无法进行穷举测试。在实际的使用过程中，测试人员还要考虑到包括随机出现的各种突发情况，比如用户不小心撞到键盘引起某个误操作。Glenford J. Myers 在 1979 年描述了一个只包含 loop 循环和 if 语句的简单程序，可以使用不同的语言将其写成 20 行左右的代码，但是这样简短的语句却有着十万亿条路径。面对这样一个庞大的数字，即便是一个有经验的优秀的软件测试员也需要十亿年才能完成全部测试，而且在实际应用中，此类程序是非常有可能出现的。

E.W.Dijkstra 的一句名言对测试的不彻底性作了很好的注解：“程序测试只能证明错误的存在，但不能证明错误的不存在”。由于穷举测试工作量太大，实践上行不通，这就注定了一切实际测试都是不彻底的，也就不能够保证被测试程序在理论上不存在遗留的错误。

2. 杀虫剂现象

软件中存在的故障现象与发现的故障数量成正比。1990 年，BorisBeizer 在其编著的“Software Testing Techniques”（第二版）中提到了“杀虫剂怪事”一词，同一种测试工具或方法用于测试同一类软件越多，则被测试软件对测试的免疫力就越强。这与农药杀虫是一样的，老用一种农药，则害虫就有了免疫力，农药就失去了作用。

在现实当中，往往是发现了一个故障以后，很可能会接二连三地发现更多的软件故障。有这样一个现象值得我们去重视：47%的软件故障（是由用户发现的）是与系统中的 4%的程序模块有关。因此，经测试后的程序中隐藏的故障数目与该程序中发现的故障数目成正比。

产生杀虫剂现象的可能原因是由于开发过程中各种各样的主客观因素，再加上不可预见的突发性事件，软件测试员采用同一种测试方法或者工具不可能检测出所有的缺陷。为了克服被测试软件的免疫力，软件测试员必须不断编写新的测试程序，对程序的各个部分进行不断测试，以避免被测试软件对单一的测试程序具有免疫力而使软件缺陷不被发现。