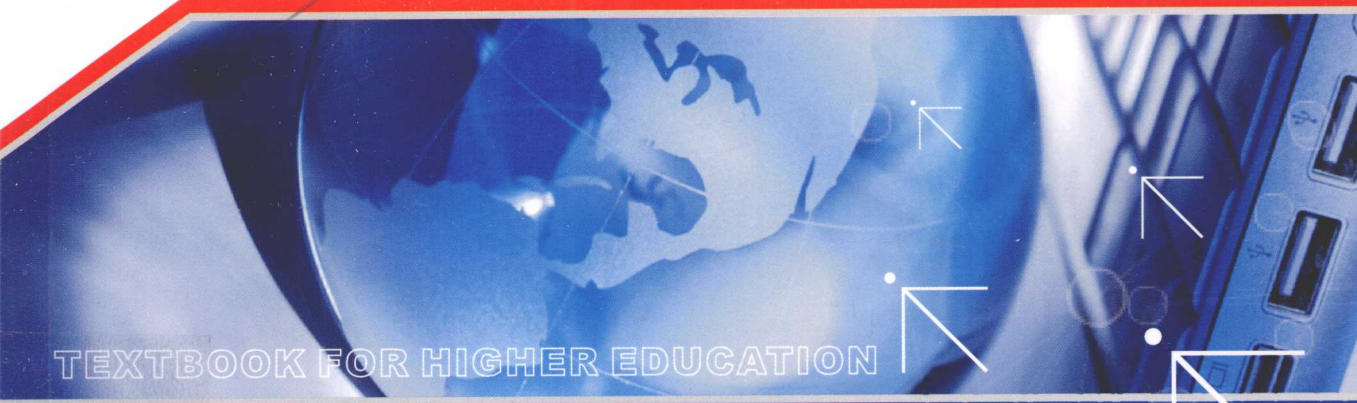
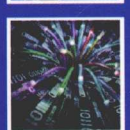
软件系统开发指导教程系列丛书

软件测试

郑 炜 主编



TEXTBOOK FOR HIGHER EDUCATION



西北工業大學出版社
NORTHWESTERN POLYTECHNICAL UNIVERSITY PRESS

软件系统开发指导教程系列丛书

软件测试

主　编　郑　炜

副主编　汪　芳　蔡　璐　吴潇雪

西北工业大学出版社

【内容简介】 本书系统回顾了软件测试的发展与软件测试的定义，深入讲解了软件测试方法与测试过程，全面介绍了测试管理与主流测试工具。

全书共分3篇13章，其中第一篇基础篇（第1～4章）介绍了软件测试的基本理论、技术和准则；第二篇实用篇（第5～9章）介绍了静态测试、单元测试、功能测试、Web测试、其他测试等测试基础知识和多种应用技术；第三篇提高篇（第10～13章）介绍了自动化测试、基于模型的测试、测试管理、成熟软件测试。

本书可以作为计算机、软件工程、软件测试及相关专业的本科、硕士研究生教材，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工作参考书。对于希望增强软件测试方面知识的程序员及软件开发团队的其他人员，本书也具有很好的参考价值。

图书在版编目(CIP)数据

软件测试/郑炜主编. —西安：西北工业大学出版社，2011.9
ISBN 978-7-5612-3194-4

Ⅰ.①软… Ⅱ.①郑… Ⅲ.①软件—测试 Ⅳ.①TP311.5

中国版本图书馆CIP数据核字(2011)第182958号

出版发行：西北工业大学出版社
通信地址：西安市友谊西路127号　　**邮编**：710072
电　　话：(029)88493844　88491757
网　　址：www.nwpup.com
印 刷 者：陕西向阳印务有限公司
开　　本：787 mm×960 mm　1/16
印　　张：21.25
字　　数：465千字
版　　次：2011年8月第1版　　2011年8月第1次印刷
定　　价：39.00元

前　言

软件测试是软件工程学科的一个重要分支。随着软件的发展，软件测试已经成为软件质量保证的关键技术之一，也是软件开发过程中的一个重要环节，它对测试人员的专业知识要求很全、专业技术要求很强、专业能力要求很高。目前企业对测试人员的要求也是要有较丰富的测试经验及较强的测试工具应用能力。

本书是在笔者参阅大量国内外相关文献资料，深入研究软件测试理论，并结合多年从事软件工程、软件测试和软件质量保证工作的实践和教学经验不断总结和充实的基础上，精心编写而成的。同时，本书在编写时注意理论结合实际，书中许多涉及理论的内容都使用例子进行阐述和说明，举一反三、抛砖引玉的内容的设计，能够使学生更有效地巩固所学的软件测试知识，是一本非常实用的软件测试教材。本书的特色在于：

(1)系统地论述了软件测试的理论知识和应用技术，内容完整、全面，包括软件测试的基本理论、技术和准则、静态测试、单元测试、Web 测试、面向对象的测试、可使用性测试、易获得性测试以及自动化测试、基于模型的测试、测试管理、成熟软件测试等重要内容，基本上涉及了软件测试的各个测试阶段和各种测试类型，涵盖了业界出现的大部分测试领域内的知识。

(2)既论述了软件测试基本知识以及相关的测试方法和技术，又介绍了典型的软件测试工具如 Visual Studio 2010，并突出了软件测试工具在实际测试项目中的运用，理论结合实际，提高学生软件测试的实战能力。

(3)内容全面、条理清晰、结构严谨、实用性强，注重理论结合实际，配合大量的例子进行阐述和说明，有助于学生全面掌握软件评测的方法和技术。对高校软件测试课程的开设、培训机构软件测试的实战培训以及开发人员和测试人员自学是非常有现实意义的。

本书由西北工业大学郑炜主编，汪芳、蔡璐、吴潇雪任副主编，西北工业大学马春燕，西北农林科技大学毛锐，西安武警学院武光明、来爽、杨威、黄鲜鸽、周峰安、邹鹏里、刘再宏、刘田、黄磊等参与编写。但由于软件测试覆盖面太大，涉及领域太多，测试工具种类繁多、应用复杂，加之水平有限，书中疏漏之处在所难免，敬请广大专家和读者多提宝贵意见。

编　者

2011 年 6 月

目　录

第一篇　基　础　篇

试读结束：需要全本请在线购买：www.ertongbook.com

第二篇　实　用　篇

第三篇　提　高　篇

第一篇 基 础 篇

第1章 软件测试基础

软件系统的复杂是安全的巨大威胁。在信息系统开发中，软件测试已成为一项不可或缺的内容。软件测试在软件生命周期的地位和意义，软件测试的概念（定义、对象、目的、原则），软件测试的方法和分类构成了软件测试的基础。

1.1 软件缺陷

1.1.1 软件缺陷导致的事故

1. 首个 Bug

故事发生在 1945 年 9 月 9 日，一个炎热的下午。当时的机房是一间第一次世界大战时建造的老建筑，没有空调，所有窗户都敞开着。Grace Hopper 正领导着一个研究小组夜以继日地工作，研制一台称为“MARK Ⅱ”的计算机，它使用了大量的继电器（电子机械装置，那时还没有使用晶体管），一台不是纯粹的电子计算机。突然，MARK Ⅱ死机了 ……

技术人员尝试了很多办法，最后定位到 70 号继电器出错。Hopper 观察这个出错的继电器，有一只飞蛾躺在中间，已经被继电器打死。他小心地用摄子将蛾子夹出来，用透明胶布粘到“事件记录本”中，并注明“第一个发现虫子的实例。”

2. Therac 案例

Therac 系列仪器是一种医用高能电子线性加速器，用来杀死病变组织癌细胞，同时使其对周围健康组织影响尽可能降低。Therac20 具有独立的合乎工业标准的硬件控制系统，配置有 PDP11 计算机，但计算机控制属辅助性质，计算机软件控制仅为在某些情况下方便操作而设置。

在实际应用中，该机器致命地超过剂量设定导致在1985年6月到1987年1月之间，发生6起已知的医疗事故，造成患者死亡或严重辐射灼伤。

经事后的调查发现，在机器的编辑阶段，数据的输入速度是该错误出现的关键因素，也就是说，对于一个熟练的操作人员，在重复同样的操作千百次之后，编辑速度越来越快，最终将使出错时的“Malfunction 54”信息出现，即超剂量辐射事故发生。测量这时的辐射剂量已经达到饱和的25 000拉德。对人体而言，辐射剂量达到1 000拉德就已经是致命的了。

由于软件错误导致系统失效，酿成重大损失的事例不胜枚举。这些惨痛教训增加了软件业界对软件质量在软件工程中的思考。设计人员和使用人员都希望在将软件系统投入运行之前，能得到系统正确性的保证，或能将系统正确性提高到比较高的程度。

1.1.2 软件缺陷定义

软件在其生命周期的各个阶段都有可能发生问题，其情况和形式各不相同，这就是缺陷，通常称为Bug。IEEE Standard 729对软件缺陷的定义是：软件缺陷就是软件产品中所存在的问题，最终表现为用户所需要的功能没有完全实现，不能满足或不能全部满足用户的需求。从产品的内部看，软件缺陷是软件产品开发或者维护过程中所存在的错误、毛病等各种问题；从外部看，软件缺陷是系统所需要实现的某种功能的失效或违背。

软件缺陷的表现形式有多种，具体见表1-1。

表1-1 软件缺陷的表现形式

表现形式	描　述
功能	属性没有实现或者只有部分实现
设计	不合理，导致存在潜在缺陷
实际结果	与客户文档的预期不一致
运行	出错，包括安装出错，运行中断，系统不兼容崩溃，界面混乱
数据结果	不正确，不精确
用户	对软件不满意，界面不美观，无法保存，或者运行不流畅，启动速度慢等

一般来说，可以参照以下规则来判别出现的问题是否是缺陷：

(1)软件是否已全部实现需求说明文档要求的功能。

(2)软件是否出现了需求说明文档指明不应该出现的错误。

(3)软件是否多实现了需求说明文档未提到的冗余功能。

(4)软件是否没实现需求说明未提及，但是应该实现的功能和目标。

(5)软件是否使用不便，运行速度缓慢，难以理解，导致客户不满意。

软件缺陷产生模型如图1.1所示。

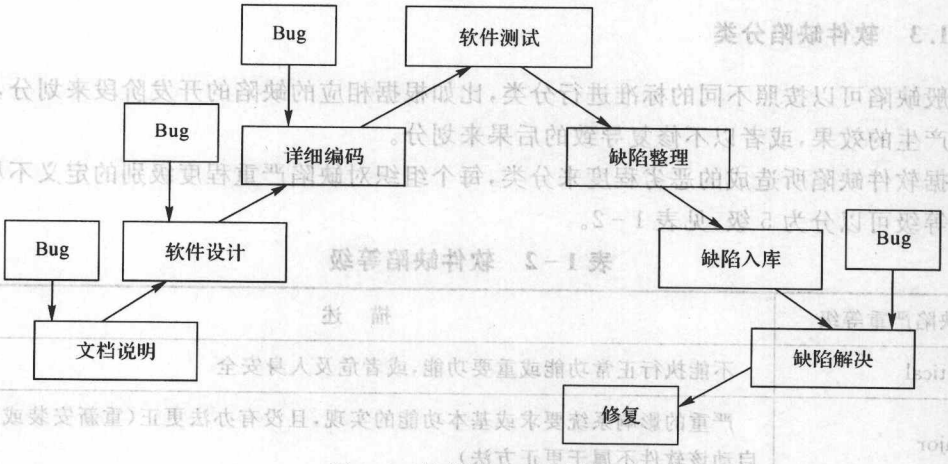


图 1.1　软件缺陷产生模型

在软件的开发阶段，有很多机会引入缺陷，并在开发的其他过程中将这些缺陷传播演变成其他缺陷，新的缺陷也有可能随着旧缺陷的修复而产生。

最有效的缺陷排查应始于项目的最初阶段，远早于程序编码。首先验证需求文档，检查需求在一致性、可测试性、可行性、明确性和可追溯性上存在的错误，往往使得缺陷预防工作在需求阶段的效率最高。其次，编程技术的不足或设计出现逻辑上的错误，比如算法错误、语法错误、计算和精度问题、接口参数传递不匹配等也会为程序带来缺陷。另外的因素包括人为的各种行为、软件本身的因素等。团队中的误解、沟通不畅将导致负责不同阶段的人员对程序涉及的含义有不同的理解。文档错误、时间上不协调或不一致、系统的自我恢复或数据的异地备份、灾难性恢复等问题，都将为后续的工作埋下隐患。软件缺陷产生的原因的统计数据如图 1.2 所示。

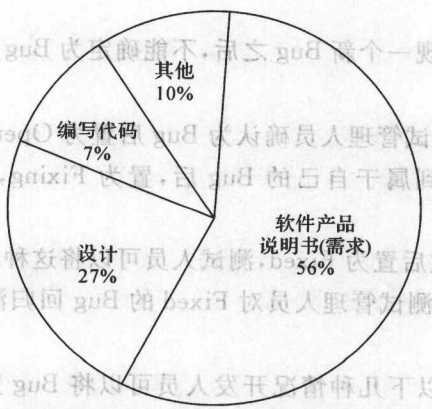


图 1.2　软件缺陷产生的原因分布

1.1.3 软件缺陷分类

一般缺陷可以按照不同的标准进行分类，比如根据相应的缺陷的开发阶段来划分，根据缺陷失效产生的效果，或者以不修复导致的后果来划分。

根据软件缺陷所造成的恶劣程度来分类，每个组织对缺陷严重程度级别的定义不尽相同，具体的等级可以分为5级，见表1-2。

表1-2 软件缺陷等级

缺陷严重等级	描 述
Critical	不能执行正常功能或重要功能，或者危及人身安全
Major	严重的影响系统要求或基本功能的实现，且没有办法更正（重新安装或重新启动该软件不属于更正方法）
Minor	严重地影响系统要求或基本功能的实现，但存在合理的更正方法（重新安装或重新启动该软件不属于更正方法）
Cosmic	使操作不方便或遇到麻烦，但它不影响执行工作功能或重要功能
Other	其他错误

根据软件缺陷产生的技术原因进行分类，一般可以概括为5种类型：输入/输出缺陷，逻辑缺陷，计算缺陷，接口缺陷，数据缺陷等。

1.1.4 软件缺陷生存周期

1. 定义缺陷的状态

缺陷的7种状态：

(1)New：指测试人员发现一个新Bug之后，不能确定为Bug，置为New，等待测试组长或项目经理确认。

(2)Open：测试人员或测试管理人员确认为Bug后置为Open。

(3)Fixing：开发人员看到属于自己的Bug后，置为Fixing，表明自己已经看到，正在修改Bug。

(4)Fixed：开发人员修改后置为Fixed，测试人员可以将这种状态的Bug进行回归测试。

(5)Reopen：测试人员或测试管理人员对Fixed的Bug回归测试后，发现问题仍然没有解决，将Bug置为Reopen。

(6)Rejected：一般出现以下几种情况开发人员可以将Bug置为Rejected，但开发人员必须写明拒绝修改该的原因：暂不修改，开发人员由于某些原因暂不修改，但必须得到项目经理的批准；无法再现，有些问题无法再现，开发人员无法修改；使用错误，开发人员认为是由于测

试读结束：需要全本请在线购买：www.ertongbook.com

试人员使用不当造成错误，拒绝修改；系统限制，该功能对系统有特殊要求和限制，本身并无缺陷。

(7)Closed：Bug 已经解决，测试人员可将其置为 Closed。

上述 Open，Fixing 和 Fixed 称为缺陷的活动态，Closed 和 Rejected 称为缺陷的终结态。

2. 缺陷管理流程

缺陷跟踪管理是测试工作的一个重要部分，测试的目的是为了尽早发现软件系统中的缺陷，因此，对缺陷进行跟踪管理，确保每个被发现的缺陷都能够及时得到处理是测试工作的一项重要内容。缺陷跟踪管理的目标是：确保每个被发现的缺陷都能够被解决，不一定都被修正，也可能以其他方式解决，总之，对每个被发现的 Bug 的处理方式必须能够在开发组织中达到一致；收集缺陷数据并根据缺陷趋势曲线识别测试过程阶段，决定测试过程是否结束有很多种方式，通过缺陷趋势曲线来确定测试过程是否结束是常用并且较为有效的一种方式；收集缺陷数据并在其上进行数据分析，作为组织的过程财富。

缺陷管理流程如图 1.3 所示，具体实施必须使用开发小组制定的缺陷管理工具，该工具将记录所有缺陷的状态信息，并可以自动产生缺陷管理报告，一切测试工作都应有计划有步骤地进行，尽早进行管理，并尽可能多地发现 Bug，保证测试工作的顺利进行。具体遵循以下流程：

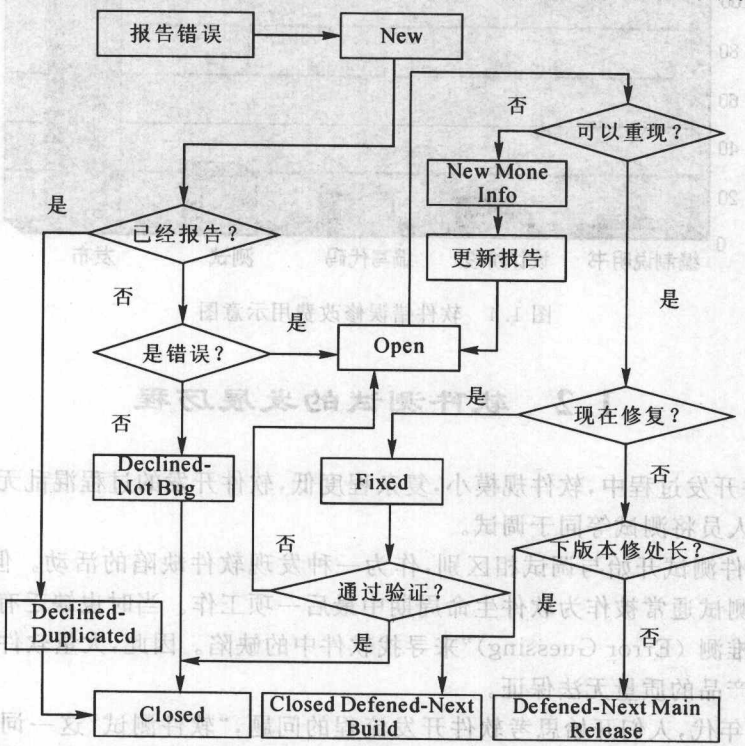


图 1.3　缺陷管理流程图

(1)测试人员提交新的 Bug 入库,错误状态为 New。

(2)高级测试人员验证错误,如果确认是错误,分配给相应的开发人员,设置状态为 Open;如果不是错误,则拒绝,设置为 Declined(拒绝)状态。

(3)开发人员查询状态为 Open 的 Bug,如果不是错误,则置状态为 Declined;如果是 Bug,则修复并置状态为 Fixed。不能解决的 Bug,要留下文字说明及保持 Bug 为 Open 状态。对于不能解决和延期解决的 Bug,不能由开发人员自己决定,一般要经某种会议(评审会)通过才能认可。

(4)测试人员查询状态为 Fixed 的 Bug,然后验证 Bug 是否已解决,如解决置 Bug 的状态为 Closed,如没有解决置状态为 Reopen。

1.1.5 软件缺陷修复代价

软件在从计划、编制、测试,一直到交付用户公开使用的过程中,都有可能产生和发现错误。随着整个开发过程的时间推移,修复软件的费用呈几何级数增长。图 1.4 是软件错误在不同阶段发现时修改的费用示意图。

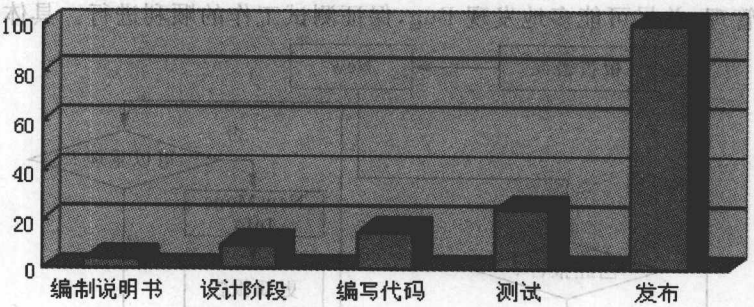


图 1.4 软件错误修改费用示意图

1.2 软件测试的发展历程

早期的软件开发过程中,软件规模小,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,开发人员将测试等同于调试。

1957 年,软件测试开始与调试相区别,作为一种发现软件缺陷的活动。但测试活动始终后于开发活动,测试通常被作为软件生命周期中最后一项工作。当时也缺乏有效的测试方法,主要依靠"错误推测 (Error Guessing)"来寻找软件中的缺陷。因此,大量软件交付后,仍存在很多问题,软件产品的质量无法保证。

20 世纪 70 年代,人们开始思考软件开发流程的问题,"软件测试"这一词条已频繁出现,一些软件测试的探索者们建议在软件生命周期的开始阶段就根据需求制订测试计划,这时涌现出一批软件测试大师。

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。它不再是一个一次性的,而且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。

从 20 世纪 80 年代中后期开始,软件生产进入以个体软件过程 PSP(Personal Software Process)、过程成熟度模型 CMM 和群组软件过程 TSP(Team Software Process)为标志,以过程为中心的第二阶段。

1996 年,测试支持度 TSM(Testing Support Model)和测试成熟度 TMM(Testing Maturity Model)的概念相继提出。其中 TMM 被分解为 5 个级别。

1 级(初始级):测试过程无序,甚至是混乱的,几乎没有妥善定义。测试和调试没有区别,除了支持调试外,测试没有其他目的,测试过程不可重复,且软件发布后没有质量保证。

2 级(定义级):测试被定义为软件生命周期中的一个阶段,在编码阶段之后。

3 级(集成级):测试贯穿于软件开发生命周期。不过,尽管处于该阶段的公司认识到评审在质量控制中的重要性,但并没有建立起有效的质量控制标准,以在软件开发生命周期中的各阶段实施有效的评审。

4 级(管理和度量级):测试活动除了测试被测程序外,还包括软件生命周期中各个阶段的评审、审查和复查,使测试活动涵盖了软件验证和软件确认活动。

5 级(优化级):具有缺陷预防和质量控制的能力,优化调整和持续改进测试过程。测试过程的管理为持续改进产品质量和过程质量提供指导,并提供必要的基础设施。

软件测试与质量保证体系有机融合,测试方法越来越细化,测试技术不断更新,软件测试正以走向专业化的发展趋势,蓬勃发展。

1.3　软件测试的定义

1.3.1　软件测试狭义定义

软件测试是使用人工的或自动的手段来运行或检测某个系统的过程,其目的在于检验它是否满足约定的需求或是比较预期结果与实际结果之间的差别。随着软件业界对软件工程的不断反思和总结,人们对软件测试有了更深入的理解,重新定义了软件测试:软件测试是在既定的状况条件下,运行一个系统或组件,观察记录结果,并对其某些方面进行评价的过程。

上述是 IEEE 的定义,一般也称为第一类测试。还存在一种被称为第二类测试的 Myers 的定义:软件测试是为了发现错误而运行程序的过程。这一定义明确指出软件测试的目的是“发现错误”。

第一类测试可简单地描述为:在设计规定的环境下运行软件,其功能及性能与用户需求或

设计相比较，如果相符，则测试通过；否则，视为缺陷。这一过程的目标是将软件的所有功能在设计规定的环境下全部运行并通过。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。

第二类测试中，测试方法与需求和设计没有必然的联系，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、无效数据的输入以及系统各种弱点，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。这种方法往往能发现系统中存在的更多缺陷。

1.3.2 软件测试广义定义

广义的软件测试是由确认、验证、测试 3 个方面组成的。

(1)确认(Validation)：评估将要开发的软件产品是否正确无误、可行和有价值。确认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，确保产品实现的功能满足了用户所有的需求。

(2)验证(Verification)：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与规格说明书保持一致性，开发过程是沿着正确的方向进行的。

(3)测试：与狭义的测试概念统一。

1.3.3 软件测试的对象

软件测试并不等于程序测试，软件测试应该贯穿整个软件定义与开发期间。因此需求分析、概要设计、详细设计以及程序编码等各阶段所得到的文档，包括需求规格说明、概要设计规格说明、详细设计规格说明以及源程序，都应该是软件测试的对象，如图 1.5 所示。

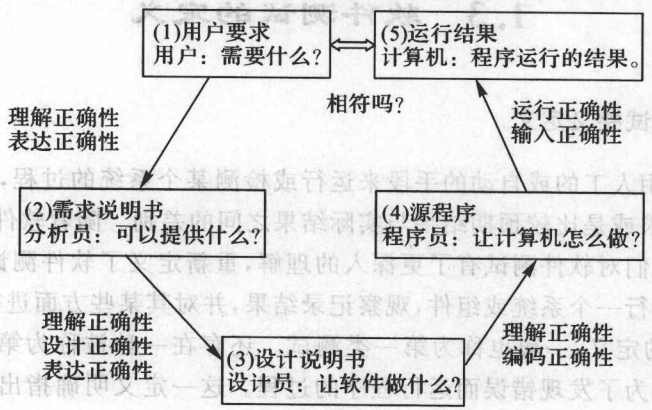


图 1.5 软件测试对象

图 1.5 中，在对需求理解与表达的正确性、设计与表达的正确性、实现的正确性以及运行的正确性的验证中，任何一个环节发生了问题都可能在软件测试中表现出来。

1.4　软件测试原则

Glenford J. Mayers 在其经典著作《The Art of Software Testing》中，提出了软件测试的十大原则，见表 1-3。

表 1-3　软件测试原则

编号	原　则
1	测试用例中一个必需部分是对预期输出或结果进行定义
2	程序员应当避免测试自己编写的程序
3	编写软件的组织不应当测试自己编写的软件
4	应当彻底检查每个测试的执行结果
5	测试用例的编写不仅应当根据有效和遇到的输入情况，而且也应当根据无效或者未遇到的输入情况
6	检查程序是否“未做其应该做的”仅是测试的一半，测试的另一半是检查程序是否“做了其不应该做的”
7	应该避免测试用例用后即弃，除非软件本身就是一个一次性的软件
8	计划测试工作时不应默许假定不会发生错误
9	程序某部分存在更多错误的可能性，与该部分已发生错误的数量成正比
10	软件测试是一项极富创造性、极具智力挑战的工作

下面就其中的几条进行阐述。

测试用例应由测试输入数据和与之对应的预期输出结果这两部分组成。测试以前应当根据测试的要求选择在测试过程中使用的测试用例(Test case)。测试用例主要用来检验程序员编制的程序，因此不但需要测试的输入数据，而且需要针对这些输入数据的预期输出结果。如果对测试输入数据没有给出预期的程序输出结果，那么就缺少了检验实测结果的基准，就有可能把一个似是而非的错误结果当成正确结果。

程序员应避免检查自己的程序，编写软件的组织也不应当测试自己编写的软件，与此同时，计划测试工作时不应默许假定不会发生错误。测试工作需要严格的作风、客观的态度和冷静的情绪。人们常由于各种原因具有一种不愿否定自己工作的心理，认为揭露自己程序中的问题总不是一件愉快的事。这一心理状态就成为测试自己程序的障碍。另外，程序员对软件

规格说明理解错误而引入的错误则更难发现。如果由别人来测试程序员编写的程序，可能会更客观、更有效，并更容易取得成功。要注意的是，这点不能与程序的调试(debuging)相混淆。调试由程序员自己来做可能更有效。

应当彻底检查每个测试的执行结果。这是一条最明显的原则，但常常被忽视。有些错误的征兆在输出实测结果时已经明显地出现了，但是如果不仔细、全面地检查测试结果，就会使这些错误被遗漏掉。所以必须对预期的输出结果明确定义，对实测的结果仔细分析检查，抓住征候，暴露错误。

测试用例的编写不仅应当根据有效和遇到的输入情况，而且也应当根据无效或者未遇到的输入情况。合理的输入条件是指能验证程序正确的输入条件，而不合理的输入条件是指异常的、临界的、可能引起问题异变的输入条件。在测试程序时，人们常常倾向于过多地考虑合法的和期望的输入条件，以检查程序是否做了它应该做的事情，而忽视了不合法的和预想不到的输入条件。事实上，软件在投入运行以后，用户的使用往往不遵循事先的约定，使用了一些意外的输入，如用户在键盘上按错了键或打入了非法的命令。如果开发的软件遇到这种情况时不能做出适当的反应，给出相应的信息，那么就容易产生故障，轻则给出错误的结果，重则导致软件失效。因此，软件系统处理非法命令的能力也必须在测试时受到检验。用不合理的输入条件测试程序时，往往比用合理的输入条件进行测试能发现更多的错误。

检查程序是否“未做其应该做的”仅是测试的一半，测试的另一半是检查程序是否“做了其不应该做的”。在项目启动时会定义需求规格说明，坚持对需求的验证，使得程序是可预测的。一方面，验证程序的正确性，根据用户的需要来进行检验，同时也应遵从某个标准进行检验；另一方面，验证程序对需求实现的完整性，用于保证需求中没有遗漏任何的功能需求，避免由于没有人提出恰当的问题或者没有人检查相关的原始文档而引起的需求遗漏；另外，还需考虑每个功能性需求附带的非功能性要求，包括性能、安全性、可使用性、兼容性和可访问性。

应该避免测试用例用后即弃，除非软件本身就是一个一次性的软件。妥善保存测试计划、测试用例、出错统计和最终分析报告，为维护提供方便。

程序某部分存在更多错误的可能性，与该部分已发生错误的数量成正比。经验表明，测试后程序中残存的错误数目与该程序中已发现的错误数目或检错率成正比，这也被称为群集现象。根据这个规律，应当对错误群集的程序段进行重点测试，以提高测试投资的效益。在所测程序段中，若发现错误数目多，则残存错误数目也比较多。这种错误群集性现象，已为许多程序的测试实践所证实。

根据中国软件评测中心的测试理念，提出的测试原则就是要从用户和开发者的角度出发，进行软件产品的测试，通过测试，为用户提供高质量的产品，并对优秀产品进行认证。其测试原则有以下 8 点：

(1)应当尽早和不断地进行软件测试。

(2)程序员应当避免检查自己的程序，测试工作应该交由独立的专业的软件测试机构来完成。

试读结束：需要全本请在线购买： www.ertongbook.com