

TURING

图灵计算机科学丛书

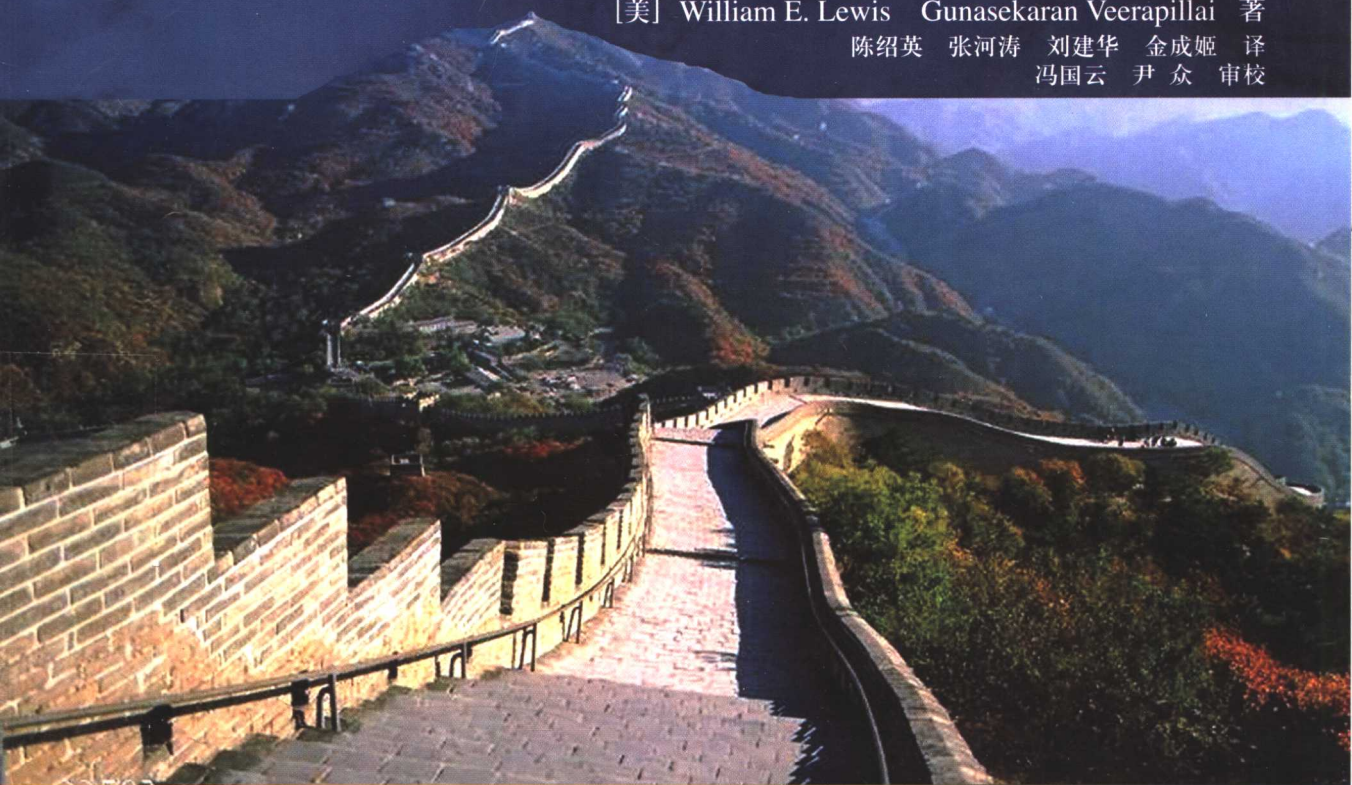
AUERBACH

软件测试与质量保证圣经

软件测试与持续质量改进 (第2版)

Software Testing and Continuous Quality Improvement
Second Edition

[美] William E. Lewis Gunasekaran Veerapillai 著
陈绍英 张河涛 刘建华 金成姬 译
冯国云 尹众 审校



人民邮电出版社
POSTS & TELECOM PRESS

TURING

图灵计算机科学丛书

TP311.5/216

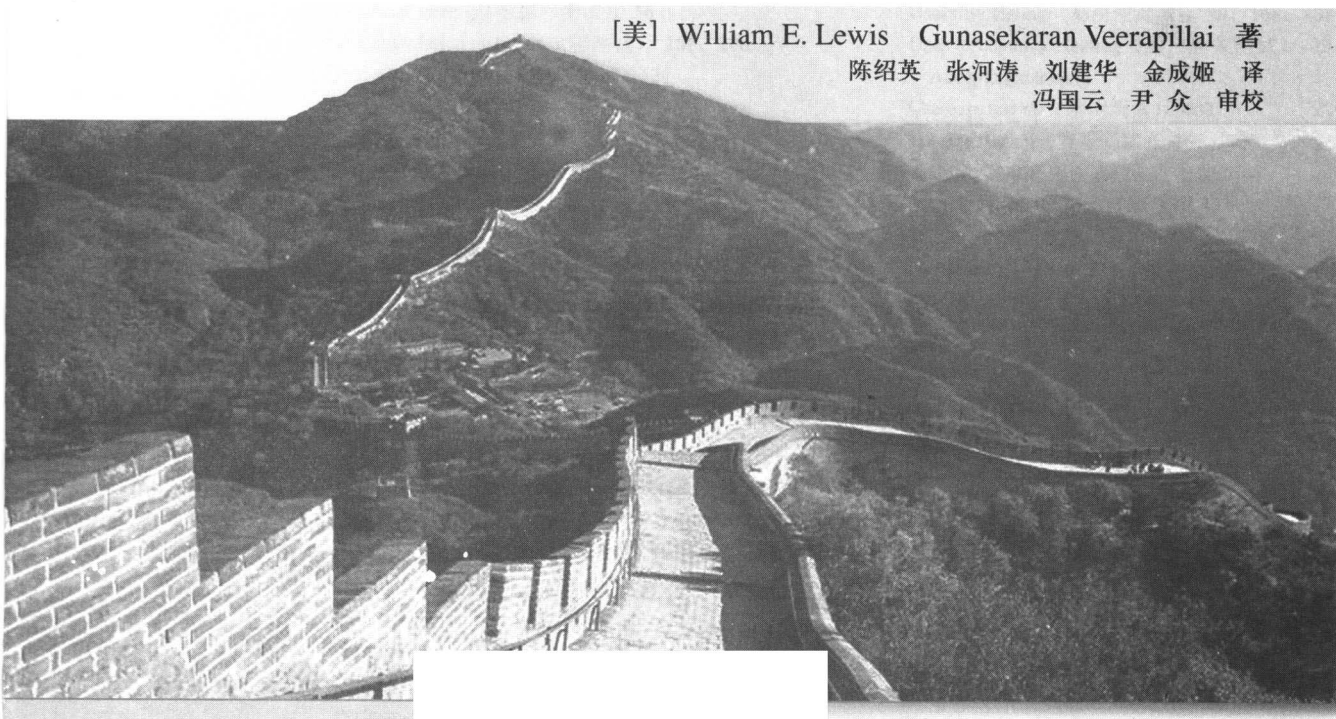
2008

软件测试与持续质量改进 (第2版)

Software Testing and Continuous Quality Improvement
Second Edition

[美] William E. Lewis Gunasekaran Veerapillai 著

陈绍英 张河涛 刘建华 金成姬 译
冯国云 尹众 审校



人民邮电出版社
北京

图书在版编目 (CIP) 数据

软件测试与持续质量改进: 第2版 / (美) 刘易斯 (Lewis, W. E.), (美) 维拉皮莱 (Veerapillai G.) 著; 陈绍英等译. —北京: 人民邮电出版社, 2008.2
(图灵计算机科学丛书)
ISBN 978-7-115-17161-0

I. 软… II. ①刘…②维…③陈… III. 软件-测试-教材 IV. TP311.5

中国版本图书馆 CIP 数据核字 (2007) 第 174734 号

内 容 提 要

本书旨在为软件测试过程提供一个质量框架, 其目的是提出一个持续改进软件质量的途径, 以提高测试效率。书中详细列举基本的软件测试技巧, 并基于一种持续改进过程介绍 Deming 的质量概念, 将“计划、执行、检查、改进”(Plan, Do, Check, Act, PDCA) 这样一个“质量轮”引入软件测试过程, 阐述现代质量保证理论及最佳实践方法。本书共分5个部分, 分别从软件质量、生命周期测试、软件测试方法、测试项目管理、现代软件测试工具的角度展开。此外, 附录中提供软件测试过程中可能涉及的各种文档的格式样本, 非常便于查阅和参考。

本书适合作为计算机及相关专业软件测试课程的教材, 也适合软件测试领域的专业技术人员作为参考手册。

图灵计算机科学丛书

软件测试与持续质量改进 (第2版)

-
- ◆ 著 [美] William E. Lewis Gunasekaran Veerapillai
译 陈绍英 张河涛 刘建华 金成姬
审 校 冯国云 尹 众
责任编辑 杨海玲
- ◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子函件 315@ptpress.com.cn
网址 <http://www.ptpress.com.cn>
北京艺辉印刷有限公司印刷
新华书店总店北京发行所经销
- ◆ 开本: 787×1092 1/16
印张: 22
字数: 607 千字 2008 年 2 月第 1 版
印数: 1-4 000 册 2008 年 2 月北京第 1 次印刷
著作权合同登记号 图字: 01-2006-4734 号

ISBN 978-7-115-17161-0/TP

定价: 55.00 元

读者服务热线: (010) 88593802 印装质量热线: (010) 67129223

反盗版热线: (010) 67171154

版 权 声 明

Authorized translation from the English language edition, entitled *Software Testing and Continuous Quality Improvement, Second Edition* by William E. Lewis; Gunasekaran Veerapillai, technical contributor, published by CRC Press LLC. Copyright © 2005 by CRC Press LLC.

All Right Reserved. Authorized translation from English Language edition published by Auerbach, part of Taylor & Francis Group LLC.

本书中文简体字版由 Auerbach 授权人民邮电出版社独家出版。未经出版者书面许可，不得以任何方式复制或抄袭本书内容。

版权所有，侵权必究。

译者序

用多如牛毛来形容中国IT公司的数量并不过分。除了一些软件外包企业外，多数企业的规模都在百人左右，过千的可谓凤毛麟角。而国内IT企业的研发领域，也多数以应用软件的研发为主，从事系统软件研发的企业则少之又少。国内企业在软件开发过程的管理方面相对落后无疑是形成这种局面的一个重要原因，而管理方面的相对落后尤其体现在测试过程管理和软件质量管理上。

我们在软件质量管理和测试领域落后于欧美先进国家很多，要想赶上国外水平，无疑需要积极地实践与学习。翻译本书的目的正是为学习国外先进的软件测试和质量管理办法提供有力的参考。

本书的主要作者在计算机行业有38年的从业经验，在软件测试与质量管理领域均有很深的造诣，并有着丰富的咨询与培训以及教学经验。这样一位资深人士的作品，对于测试与质量领域的工作有着重要的指导意义。

本书包含“软件质量透视”、“生命周期测试回顾”、“软件测试方法”、“测试项目管理”、“现代软件测试工具”5部分内容，共计30章。通过这5部分内容的学习，可以全面掌握软件测试与质量改进的理论思想。

目前，在国内招聘软件测试工程师的难度要超过招聘软件开发工程师，尤其是想要招聘一些技能全面的测试开发工程师更是如此，这一点国内的诸多软件外包服务公司都深有体会。我经常面试一些软件测试求职者，发现其中很多人是以为软件测试相对容易才决定加入这一领域的，这些求职者的软件测试知识非常贫乏，连基本的测试过程、测试方法都不了解。

一本内容全面的软件测试与质量管理图书，不但可以让新手快速入门，更可以让专业人士领悟其中的精髓，不断改进自己的工作。本书足可以满足这些方面的要求。通过学习本书，读者还可以全面掌握软件测试与质量管理的理论知识，不断改进企业质量管理流程。

本书由陈绍英、张河涛、刘建华、金成姬、纪云、王雯共同翻译完成，冯国云和尹众负责本书的校稿工作。

特别感谢金明姬同学进行了大量的录入工作。

陈绍英

2007年9月于北京

作者简介

William E. Lewis是享誉世界的软件测试与质量保证领域的专家。他拥有数学学士学位和运筹学硕士学位，有38年计算机行业从业经验。现在他是Smartware技术公司的创始人、董事长和CEO，该公司是一家专注于软件测试的质量保证咨询公司。他也是专利软件测试工具Test Smart™的发明者，这是一个可以根据需求生成优化了的测试用例/数据的测试工具。（关于作者的更多信息，请参见www.smartwaretechnologies.com。）

他拥有佛罗里达奥兰多质量保证协会（QAI）颁发的质量分析师（CQA）和软件测试工程师（CSTE）证书。在过去几年中，他在多次会议上宣讲论文。2004年他在QAI的年度国际信息技术质量会议上宣讲了名为*Cracking the Requirements/Test Barrier*（打破需求和测试的屏障）的论文。他还在美国质量学会的专题会议和信息技术从业者协会的会议上做过演讲。

Lewis先生曾经是花旗集团的质量保证经理，他管理测试团队，记录所有的软件测试、质量保证过程及规程，积极参与了花旗集团CMM的工作，并且设计了大量的WinRunner自动脚本。

Lewis先生还曾经供职于美国Technology Builders公司和沙特阿拉伯石油公司（Aramco），任测试顾问。

Lewis先生曾于1994年加入安永，负责该公司的软件配置管理、软件测试和应用程序演化手册，并且帮助开发一些应用程序的改进路线图。他是多个应用开发项目的质量保证经理，在测试计划、测试设计、执行、评估、报告以及测试自动化方面有非常丰富的经验。他负责该公司的ISO评审工作，最终促成了安永获得ISO9000国际认证。

1998年，Lewis在工作了28年后从IBM退休。他担任过12年的课程设计师和讲师，还做了多年的系统程序员/分析员和性能分析员。他曾在韩国首尔的韩国高等科技学院（KAIST）担任软件工程课程主管，该学院被誉为“韩国的MIT”。他还在加拿大多伦多的IBM加拿大总部工作过，在那里负责升级企业培训程序。另外，他还曾经在美国各地、罗马、阿姆斯特丹、南安普敦、香港和悉尼等大学，进行以软件测试为主的软件开发和质量保证课程的教学。在如此忙碌的情况下他还出版了关于计算机问题解决的5本书。

Gunasekaran Veerapillai 是持证的软件质量分析师（CSQA），也是美国PMI认证的PMP（项目管理专家）。他曾经在印度Canara银行工作，拥有15年的小额银行业务的经验，此后他在该银行位于班加罗尔的IT部的电子数据处理部门做了4年的经理。许多关键的内部软件开发、测试和维护项目都由他负责。他在与Thinksoft Global Services合作的测试项目中担任项目经理，该公司精于银行金融服务与保险（BFSI）方面的测试。

现在，他是HCL Technologies公司的高级测试中心项目经理，该公司是一个通过了CMM 5级认证的公司，并与知名测试自动工具厂商合作，如Mercury Interactive和IBM Rational。Guna曾经成功地为多家国际银行（如花旗银行、摩根士丹利和Discover Financial等）做了许多测试项目，他也为软件测试网站（如Sticky Minds）撰稿。

引言

许多教材都是在一个结构化的开发环境中来阐述软件测试的。“结构化”意味着明确地定义了开发周期以及开发周期中相应的一些开发步骤，每一步都提供了一些可度量的输出结果。这种做法有一个前提，那就是假设软件测试活动基于清晰定义的需求和软件开发标准，并且那些标准是用来制定和实施测试计划的。但是，现实情况通常并非如此。通常，软件测试所面对的需求经常是变化的，甚至是错误的。

本书旨在为传统的结构化以及非结构化环境下的软件测试过程提供一个质量框架，其目的是要提出一个持续改进软件质量的途径，以提高测试方法的效率，并提供一些技巧、技术和方法供用户选择。

持续质量框架的基础来源于Edward Deming的质量原理。Deming是质量改进领域的先驱，他曾经帮助日本的制造业起死回生。同样，在传统的“瀑布式”及“螺旋式”应用程序快速开发环境中进行软件测试时也应用了Deming的质量原理。所谓“瀑布式”方法是指在测试过程中遵循预先定义的顺序步骤，并且每一步都有明确定义的需求，而在“螺旋式”方法中，这些严格的顺序步骤可能就会不同程度地缺失或有所变化。

第一部分介绍现代质量保证原理及最佳实践。书中详细列举了基本的软件测试技术，并基于一种持续改进的过程介绍了Deming的质量概念，将“计划、执行、检查、改进”(Plan, Do, Check, Act, PDCA)这样一个“质量轮”的概念也引入软件测试过程。

持续改进过程的计划阶段从一开始就必须定义测试的目的，也就是测试在结束时结果是什么，必须描述测试策略的要素以及测试计划。测试策略是关于怎样满足测试目的的简要描述，必须在制定测试计划开发之前确定下来。书中提供了一个很好的测试计划提纲，包括引言、总体计划、测试需求、测试规程以及测试计划细节。

持续改进过程的执行阶段则阐述怎样设计和执行测试计划中包括的测试。书中以“食谱”的方式描述了怎样在一个“螺旋式”的环境中进行组件测试、集成测试以及系统验收测试。

持续改进过程的检查阶段主要强调度量及测试报告的重要性。测试小组必须严格地记录测试结果，并将其与测试计划和系统测试目标关联起来。书中提供了一份测试报告的示例，并介绍了几种图形分析技术。

持续改进过程的改进阶段则主要是更新测试用例和测试脚本。书中为读者讲述了如何为下一轮测试做准备，并从人员、过程及技术等方面提出了改进的建议。

第二部分介绍“瀑布式”开发方法，并描述了怎样通过技术评审和软件测试将持续质量改进的理念应用于软件开发的每个阶段中。书中依次讲解了需求分析、逻辑设计、物理设计、程序单元设计以及编码阶段，并且在每一阶段都应用了技术评审和软件测试。最后，还讨论了软件测试的心理学。

第三部分从技术上和心理上对比“瀑布式”开发方法和“螺旋式”快速开发方法。当需求频繁变化时，建议最好采用“螺旋式”方法进行测试。书中结合“PDCA质量轮”，并应用Deming的持续质量改进理论，将“螺旋式”方法分解成若干部分、步骤及任务进行讲解。

第四部分讨论维护和提高现有系统面临的基本挑战，描述和对比软件的变更，并结合软件维护活动的心理，讲述了管理软件维护的策略。书中同样基于“PDCA质量轮”，并应用Deming的持续质量改进过程，将维护测试方法分解成若干部分、步骤及任务进行介绍。

第五部分首先简要回顾了从20世纪50年代至今的测试工具的历史，并展望了未来的测试工具，然后描述了把握运用测试工具的时机的指导原则，并以问答的形式列举了选择测试工具的方法，还介绍了当今最流行的一些软件测试产品。最后，书中详细描述了评估测试工具的方法，范围涵盖很广，从最初的测试目标一直到培训和实施。

致谢

我要向我的妻子Carol表示我最诚挚的谢意，感谢她在我进行本书的准备工作中给予了极大的耐心和爱。同样也要感谢我的父母Bill和Joyce Lewis，我永远都不会忘记他们。

我要感谢John Wyzalek，Auerbach出版社的资深组稿编辑，因为他认识到为这本书出第2版的重要性。我也要感谢Gunasekaran Veerapillai，他是主要的技术撰稿人和编辑，他对于软件测试的概念和方法有着透彻的理解。

最后，我想要感谢许多软件测试厂商，他们为本书的第五部分中的相应测试工具提供了描述素材。

目 录

第一部分 软件质量透视

第 1 章 质量保证框架	2
1.1 什么是质量	2
1.2 预防与检测	2
1.3 验证与确认	3
1.4 软件质量保证	4
1.5 质量保证的组成	5
1.6 软件测试	5
1.7 质量控制	5
1.8 软件配置管理	6
1.9 软件配置管理的要素	6
1.10 组件确定	7
1.11 版本控制	7
1.12 配置构建	8
1.13 变更控制	8
1.14 软件质量保证计划	8
1.15 开发和实施软件质量保证计划的步骤	9
1.15.1 步骤1: 编写计划	9
1.15.2 步骤2: 获得管理层认可	10
1.15.3 步骤3: 获得开发人员认可	10
1.15.4 步骤4: 准备编写SQA计划	10
1.15.5 步骤5: 执行SQA计划	10
1.16 质量标准	10
1.16.1 ISO9000	11
1.16.2 能力成熟度模型	11
1.16.3 PCMM	13
1.16.4 CMMI	14
1.16.5 美国波多里奇国家质量奖	14
第 2 章 测试技术概述	16
2.1 黑盒测试(功能测试)	16
2.2 白盒测试(结构测试)	16
2.3 灰盒测试(功能与结构测试)	17
2.3.1 手工测试与自动测试	17
2.3.2 静态测试与动态测试	17
2.4 软件测试技术的分类	18
第 3 章 质量的持续改进过程	22
3.1 Edward Deming的贡献	22

3.2 统计方法的角色	22
3.2.1 因果图	22
3.2.2 流程图	23
3.2.3 帕累托图	23
3.2.4 运行图	23
3.2.5 直方图	23
3.2.6 散布图	23
3.2.7 控制图	23
3.3 Deming的14条质量原则	23
3.3.1 第1条: 树立始终如一的目标	23
3.3.2 第2条: 采用新的质量观念	24
3.3.3 第3条: 停止对大量检查的依赖	24
3.3.4 第4条: 结束仅靠价签来激励企业的实践活动	24
3.3.5 第5条: 坚持不懈地、永久地改进生产和服务系统	24
3.3.6 第6条: 组织培训及再培训	25
3.3.7 第7条: 确立领导职责	25
3.3.8 第8条: 驱除恐惧	25
3.3.9 第9条: 打破员工区域之间的栅栏	25
3.3.10 第10条: 为员工解除口号、训词及目标	26
3.3.11 第11条: 去除数字化目标	26
3.3.12 第12条: 消除员工自豪感的障碍	26
3.3.13 第13条: 组织有关教育和再培训方面的强有力的课程	26
3.3.14 第14条: 采取行动完成转变	26
3.4 通过“计划、执行、检查、改进”实现持续改进	27
3.5 遵循PDCA循环	27

第二部分 生命周期测试概述

第 4 章 概述	30
4.1 瀑布式开发方法	30
4.2 “阶段化”持续改进方法	31
4.3 生命周期测试的心理学	31

4.4 将软件测试作为持续改进过程	31	8.2 顺序结构	52
4.5 测试的圣经: 软件测试计划	33	8.3 选择结构	52
4.6 制定测试计划的主要步骤	34	8.4 循环结构	52
4.6.1 步骤1: 定义测试目标	35	8.5 编写单元测试用例	53
4.6.2 步骤2: 确定测试方法	35	第9章 代码编写阶段的检验	54
4.6.3 步骤3: 定义测试环境	35	9.1 用技术评审对代码编写进行测试	54
4.6.4 步骤4: 制定测试规约	35	9.2 执行测试计划	55
4.6.5 确定测试日程表	35	9.3 单元测试	55
4.6.6 评审及批准测试计划	35	9.4 集成测试	55
4.7 测试计划的组成	35	9.5 系统测试	56
4.8 将技术评审作为持续改进过程	37	9.6 验收测试	56
4.9 技术评审的动机	38	9.7 缺陷记录	56
4.10 评审的类型	38	第三部分 软件测试方法	
4.11 结构化走查	38	第10章 开发方法概述	60
4.12 审查	38	10.1 生命周期开发的局限性	60
4.13 参与者角色	40	10.2 客户端/服务器架构的挑战	60
4.14 有效评审的步骤	40	10.3 客户端/服务器架构中螺旋测试的心理 学	61
4.14.1 步骤1: 规划评审过程	40	10.3.1 新思想	61
4.14.2 步骤2: 安排评审进度	41	10.3.2 对测试人员/开发人员的理解	62
4.14.3 步骤3: 制定评审议程	41	10.3.3 项目的目标: 把质量保证和 开发结合起来	62
4.14.4 步骤4: 创建评审报告	41	10.3.4 迭代/螺旋式开发方法	63
第5章 验证需求阶段	42	10.4 JAD的角色	64
5.1 通过技术评审测试需求	43	10.5 原型角色	65
5.2 审查和走查	43	10.6 开发原型的方法	66
5.3 检查表	43	10.6.1 开发原型	66
5.4 方法检查表	43	10.6.2 向管理层演示原型	67
5.5 需求可追溯性矩阵	44	10.6.3 向用户演示原型	67
5.6 制定系统/验收测试计划	44	10.6.4 修订并定稿规约	67
第6章 验证逻辑设计阶段	46	10.6.5 开发产品系统	68
6.1 数据模型、过程模型及其联系	46	10.7 持续改进螺旋测试方法	68
6.2 通过技术评审测试逻辑设计	47	第11章 信息收集(计划)	71
6.3 细化系统/验收测试计划	48	11.1 步骤1: 准备访谈	72
第7章 验证物理设计阶段	49	11.1.1 任务1: 确定参加访谈的人	72
7.1 通过技术评审测试物理设计	49	11.1.2 任务2: 确定议程	72
7.2 创建集成测试用例	50	11.2 步骤2: 执行访谈	73
7.3 集成测试方法	50	11.2.1 任务1: 理解项目	73
7.3.1 步骤1: 标识出单元接口	50	11.2.2 任务2: 理解项目的目标	73
7.3.2 步骤2: 全面协调接口	50	11.2.3 任务3: 理解项目的状态	74
7.3.3 步骤3: 创建集成测试条件	51	11.2.4 任务4: 理解项目的计划	75
7.3.4 步骤4: 评估集成测试条件的 完整性	51	11.2.5 任务5: 理解项目的开发方法	75
第8章 验证程序单元设计阶段	52		
8.1 通过技术评审测试程序单元设计	52		

11.2.6	任务6: 确定高级业务需求	75	13.3	步骤3: 定义系统/验收测试	105
11.2.7	任务7: 进行风险分析	76	13.3.1	任务1: 确定可能的系统 测试	105
11.3	步骤3: 总结访谈成果	77	13.3.2	任务2: 设计阶段性系统 测试	105
11.3.1	任务1: 总结访谈	77	13.3.3	任务3: 确定潜在的验收 测试	107
11.3.2	任务2: 确认访谈成果	78	13.4	步骤4: 测试设计的评审和批准	107
第 12 章	测试计划 (计划)	79	13.4.1	任务1: 评审的日程安排/ 准备	107
12.1	步骤1: 建立测试计划	80	13.4.2	任务2: 获得批准	107
12.1.1	任务1: 准备引言部分	80	第 14 章	测试开发 (执行)	108
12.1.2	任务2: 定义总体的功能需求	81	14.1	步骤1: 开发测试脚本	108
12.1.3	任务3: 确定手动/自动测试 的种类	82	14.1.1	任务1: 开发自动/手动图形 用户界面测试和功能测试 脚本	108
12.1.4	任务4: 确定测试结束标准	82	14.1.2	任务2: 开发自动/手动的 阶段性系统测试的脚本	109
12.1.5	任务5: 制定回归测试策略	83	14.2	步骤2: 测试开发的评审和批准	109
12.1.6	任务6: 定义测试交付物	84	14.2.1	任务1: 评审的日程安排/ 准备	109
12.1.7	任务7: 组建测试团队	85	14.2.2	任务2: 获得批准	109
12.1.8	任务8: 建立测试环境	86	第 15 章	通过可追溯性实现测试覆盖	111
12.1.9	任务9: 定义依赖关系	86	15.1	用例和可追溯性	112
12.1.10	任务10: 创建测试进度表	87	15.2	小结	113
12.1.11	任务11: 选择测试工具	89	第 16 章	测试执行/评价 (执行/检查)	114
12.1.12	任务12: 建立缺陷报告/跟 踪规程	89	16.1	步骤1: 组织测试内容并进行测试	114
12.1.13	任务13: 建立变更请求规程	90	16.1.1	任务1: 回归测试上次螺旋 过程中的缺陷	114
12.1.14	任务14: 建立版本控制规程	91	16.1.2	任务2: 执行新螺旋测试中 的手动/自动测试	115
12.1.15	任务15: 定义配置构建规程	91	16.1.3	任务3: 记录螺旋测试中发 现的缺陷	115
12.1.16	任务16: 定义项目问题解决 规程	91	16.2	步骤2: 测试评价	115
12.1.17	任务17: 建立报告规程	92	16.3	步骤3: 发布中期报告	116
12.1.18	任务18: 定义批准规程	92	16.3.1	任务1: 重新安排测试进度表	116
12.2	步骤2: 定义度量目标	92	16.3.2	任务2: 确定需求变更	116
12.2.1	任务1: 定义度量标准	93	第 17 章	准备下次螺旋测试 (改进)	118
12.2.2	任务2: 定义度量要点	93	17.1	步骤1: 更新测试内容	119
12.3	步骤3: 评审与批准测试计划	95	17.1.1	任务1: 更新功能/图形用 户界面测试内容	119
12.3.1	任务1: 评审的日程安排/ 执行	95	17.1.2	任务2: 更新阶段性系统测	
12.3.2	任务2: 获得批准	96			
第 13 章	测试用例设计 (执行)	97			
13.1	步骤1: 设计功能测试	97			
13.1.1	任务1: 完善功能测试需求	97			
13.1.2	任务2: 建立功能/测试矩阵	102			
13.2	步骤2: 设计图形用户界面测试	103			
13.2.1	任务1: 确定应用程序图形 用户界面组件	103			
13.2.2	任务2: 定义图形用户界面 测试	104			

试内容.....	119	18.3.2 任务2: 获得批准	135
17.1.3 任务3: 更新验收测试内容	119	18.4 步骤4: 执行系统测试	135
17.2 步骤2: 重新评价测试队伍、规程、 环境	119	18.4.1 任务1: 对系统测试中的修正 进行回归测试	135
17.2.1 任务1: 评价测试队伍	119	18.4.2 任务2: 执行新的系统测试	135
17.2.2 任务2: 评审测试控制规程	120	18.4.3 任务3: 记录系统测试中发 现的缺陷	136
17.2.3 任务3: 更新测试环境	120		
17.3 步骤3: 发布中期测试报告	121		
第 18 章 进行系统测试	123	第 19 章 进行验收测试	137
18.1 步骤1: 完成系统测试计划	123	19.1 步骤1: 完成验收测试计划	138
18.1.1 任务1: 确定系统测试的 类型	123	19.1.1 任务1: 确定验收测试的 类型	138
18.1.2 任务2: 确定系统测试的日 程安排	124	19.1.2 任务2: 确定验收测试的日 程安排	138
18.1.3 任务3: 组建系统测试团队	126	19.1.3 任务3: 组建验收测试团队	138
18.1.4 任务4: 建立系统测试环境	126	19.1.4 任务4: 建立验收测试环境	139
18.1.5 任务5: 安装系统测试工具	127	19.1.5 任务5: 安装验收测试工具	139
18.2 步骤2: 完成系统测试用例	127	19.2 步骤2: 完成验收测试用例	139
18.2.1 任务1: 设计/脚本化性能 测试	127	19.2.1 任务1: 系统级别测试用例 的子集	139
18.2.2 任务2: 设计/脚本化安全 性测试	129	19.2.2 任务2: 设计/脚本化附加 的验收测试	140
18.2.3 任务3: 设计/脚本化容量 测试	130	19.3 步骤3: 验收测试计划的评审和 批准	140
18.2.4 任务4: 设计/脚本化压力 测试	130	19.3.1 任务1: 评审的日程安排/ 执行	140
18.2.5 任务5: 设计/脚本化兼容 性测试	131	19.3.2 任务2: 获得批准	140
18.2.6 任务6: 设计/脚本化转换 测试	131	19.4 步骤4: 执行验收测试	140
18.2.7 任务7: 设计/脚本化易用 性测试	131	19.4.1 任务1: 对验收测试中的修正 进行回归测试	140
18.2.8 任务8: 设计/脚本化文档 测试	132	19.4.2 任务2: 执行新的验收测试	141
18.2.9 任务9: 设计/脚本化备份 测试	133	19.4.3 任务3: 记录验收测试中发现 的缺陷	141
18.2.10 任务10: 设计/脚本化恢 复性测试	133		
18.2.11 任务11: 设计/脚本化安装 测试	133	第 20 章 总结/报告螺旋测试结果	142
18.2.12 任务12: 设计/脚本化其他 类型的系统测试	134	20.1 步骤1: 执行数据精简	142
18.3 步骤3: 系统测试的评审和批准	135	20.1.1 任务1: 确保所有的测试均 已执行/解决	142
18.3.1 任务1: 评审的日程安排/ 执行	135	20.1.2 任务2: 通过测试编号整理 测试缺陷	143
		20.1.3 任务3: 将剩余的缺陷写入 一个矩阵	143
		20.2 步骤2: 准备最终的测试报告	143
		20.2.1 任务1: 准备项目概述	143
		20.2.2 任务2: 总结测试活动	143
		20.2.3 任务3: 分析/创建度量图	144

20.2.4 任务4: 总结测试成果/建议	148
20.3 步骤3: 最终测试报告的评审和批准	149
20.3.1 任务1: 评审的日程安排/执行	149
20.3.2 任务2: 获得批准	150
20.3.3 任务3: 发布最终的测试报告	150

第四部分 测试项目管理

第 21 章 项目管理概述	152
21.1 定义目标	152
21.2 定义项目范围	152
21.3 识别关键活动	153
21.4 正确地估算	153
21.5 设计	153
21.6 人员管理	153
21.6.1 领导力	153
21.6.2 沟通	154
21.6.3 解决问题	154
21.6.4 持续监控	154
21.6.5 变更管理	155
第 22 章 测试项目管理	156
22.1 理解需求	156
22.2 测试规划	157
22.3 测试执行	157
22.4 识别及改进过程	157
22.5 测试项目经理的基本特质	158
22.5.1 需求分析	158
22.5.2 差距分析	158
22.5.3 开发测试用例时要进行横向思考	158
22.5.4 避免重复	158
22.5.5 测试数据生成	158
22.5.6 确认测试环境	159
22.5.7 带着破坏目的去测试	159
22.5.8 分析测试结果	159
22.5.9 毫不犹豫地接受别人的帮助	159
22.5.10 出现问题时将它们转移	159
22.5.11 增强沟通	159
22.5.12 不断更新自身的业务知识	159
22.5.13 学习新的测试技术和工具	160
22.5.14 提交质量	160
22.5.15 过程改进	160

22.5.16 创建知识库	160
22.5.17 重获成功	160
第 23 章 测试评估	161
23.1 测试评估中的关键活动	163
23.1.1 测试范围文档	163
23.1.2 测试策略	163
23.1.3 测试条件	163
23.1.4 测试用例	164
23.1.5 测试脚本	164
23.1.6 执行/运行计划	164
23.2 影响测试评估的因素	164
23.3 测试计划评估	164
23.4 测试执行与控制工作量	165
23.5 测试结果分析	165
23.6 工作量估算——项目模型	166
第 24 章 缺陷监控及管理过程	169
24.1 缺陷报告	170
24.2 缺陷会议	170
24.3 缺陷分类	170
24.4 缺陷的优先级	171
24.5 缺陷的种类	171
第 25 章 测试与开发方法的整合	173
25.1 步骤1: 组建测试团队	173
25.2 步骤2: 确定将要整合的测试步骤及任务	174
25.3 步骤3: 自定义测试步骤及任务	174
25.4 步骤4: 选择整合要点	174
25.5 步骤5: 修改开发方法	174
25.6 步骤6: 合并缺陷记录	174
25.7 步骤7: 对测试方法的使用进行培训	175
第 26 章 本土/离岸模型	176
26.1 步骤1: 分析	176
26.2 步骤2: 确定经济上的得失	176
26.3 步骤3: 确定选择标准	177
26.4 项目管理和监控	177
26.5 外包方法	177
26.5.1 本土活动	178
26.5.2 离岸活动	178
26.6 实现本土/离岸模型	179
26.6.1 知识转移	179
26.6.2 详细设计	179
26.6.3 基于里程碑的转移过程	179

26.6.4 稳定状态	179	30.3.2 任务2: 定义选择标准	208
26.6.5 应用管理	179	30.3.3 任务3: 确定候选工具	208
26.6.6 关系模型	180	30.3.4 任务4: 进行候选工具评审	208
26.7 本土/离岸方法的收益	181	30.3.5 任务5: 为候选工具打分	208
26.8 本土/离岸模型的未来	183	30.3.6 任务6: 选择工具	208
第五部分 现代软件测试工具			
第 27 章 软件测试简史	186	30.4 步骤3b: 正式采购模式下的选择活动	209
27.1 自动测试工具的发展	188	30.4.1 任务1: 制定采购计划	209
27.1.1 静态录制/回放工具 (不附带脚本语言)	189	30.4.2 任务2: 创建技术需求文档	209
27.1.2 静态录制/回放工具 (具有脚本语言)	189	30.4.3 任务3: 评审需求	209
27.1.3 可变的录制/回放工具	189	30.4.4 任务4: 生成请求建议	209
27.2 软件测试和开发的历史比较	191	30.4.5 任务5: 简化建议	209
第 28 章 软件测试趋势	193	30.4.6 任务6: 进行技术评估	209
28.1 自动录制/回放测试工具	193	30.4.7 任务7: 选择工具源	209
28.2 测试用例建立工具	193	30.5 步骤4: 采购测试工具	210
28.3 领先的自动测试工具	194	30.6 步骤5: 制定评价计划	210
28.4 领先的测试用例生成工具	195	30.7 步骤6: 制定工具经理计划	210
28.5 必要条件和充分条件	195	30.8 步骤7: 创建培训计划	210
28.6 测试数据/测试用例的生成	195	30.9 步骤8: 接收工具	210
28.6.1 生产数据抽样	196	30.10 步骤9: 执行验收测试	211
28.6.2 从零开始	196	30.11 步骤10: 召开推介会议	211
28.6.3 数据播种	196	30.12 步骤11: 实施修改	211
28.6.4 根据数据库生成数据	197	30.13 步骤12: 培训工具的用户	211
28.6.5 根据需求生成测试数据/测试用例	197	30.14 步骤13: 在操作环境中使用工具	211
第 29 章 测试工具的分类	199	30.15 步骤14: 撰写评价报告	212
29.1 测试工具选择检查表	199	30.16 步骤15: 确定目标是否实现	212
29.2 厂商工具描述	199	附录 A 螺旋测试法	213
29.3 需要使用测试自动化的情况	205	附录 B 软件质量保证计划	221
29.4 不需要使用测试自动化的情况	205	附录 C 需求规约	222
第 30 章 自动测试工具的评价方法	207	附录 D 变更请求表	224
30.1 步骤1: 定义你的测试需求	207	附录 E 测试模板	225
30.2 步骤2: 设定工具的目标	207	附录 F 检查表	251
30.3 步骤3a: 非正式采购模式下的选择活动	207	附录 G 软件测试技术	281
30.3.1 任务1: 制定采购计划	207	参考文献	325
		术语表	331
		索引	334

软件质量透视

新世纪是以信息技术世界的一个重大挫折开始的，20世纪90年代的信息技术泡沫已经破灭了。2000年问题刚刚消失，IT产业又发生了严重的停滞，数以百计的新公司消失了，不是关门大吉就是被人兼并，行业中出现了合并浪潮，在全球范围内多家跨国公司开始了重大的结构重组。20世纪的最后3年是一个兼并的时期，这个时期比较好的变化是人们认识到了软件质量的重要性。

- 网络商务的风险使得软件的完整性、健壮性和可靠性成为必需，否则黑客可能会卷走数百万美元。因此，所有的公司都开始把它们的软件质量保证部门摆上重要位置，而在此之前质量保证部门是经常被人忽视的。
- 萨班斯-奥克斯利法案，是对应于安然、世界通信和其他类似的惨败应运而生的法案，在美国极大地提高了QA和测试的重要性，并且公司领导也开始对他们的QA团队给予了同样的重视，在此之前QA团队是另外一个被忽视的环节。

软件质量是所有人都想要的。经理们知道他们想要高质量，软件开发人员知道他们想要生产出一个高质量的产品，用户也坚持软件应当能始终如一地工作并且十分可靠。

美国软件质量协会（ASQ）和质量保证协会（QAI）定义了许多由各种组织采用的不同质量保证实践。许多组织组成质量保证小组来对其软件应用程序加以改进和评估。然而对质量保证而言，没有普遍接受的实践，因此在不同的组织中质量保证小组可能完成不同的职责，并使用不同的过程执行他们的计划。在有些组织中，软件测试是质量保证小组的责任；而在另外一些组织中，软件测试则是开发小组或者其他独立组织的职责。

许多软件质量小组做出与测试计划相似的软件质量保证计划。一个软件质量保证计划可能包括更多的活动，而不会局限于测试计划中所包括的那些活动。尽管质量保证计划包括了整个质量策略，但测试计划仍是质量保证计划的一部分，而质量保证计划是质量控制工具之一。

本部分的目标如下：

- 定义质量及其成本。
- 区分质量预防和质量检测。
- 区分验证和确认。
- 简要描述质量保证的组成。
- 简要描述通用的测试技术。
- 描述持续改进过程在获取质量的过程中是如何实现的。

1.1 什么是质量

在韦氏字典中，质量是这样定义的：“质量是事物的一种本质特征，是与生俱来的，它有别于事物的其他特征，能够表明事物的优秀程度或级别。”如果查阅一下有关计算机的文献，你就会发现有两种关于质量的定义基本上可以接受。第一种将质量定义为“满足需求”，在这种定义之下，要有一个高质量的产品，需求必须是可度量的，这样才能知道产品的需求是否被满足了。也就是说，质量处于一种非此即彼的状态，就一个产品而言，要么是有质量的，要么就是没质量的。需求可以非常复杂，也可以非常简单，但只要这些需求是可度量的，就可以确定产品是否达到了质量要求。这是生产者的质量观，即用质量来表示产品是否满足了需求或达到了规约。从本质上讲，最终应该满足规约。

我们通常使用的是质量的另一种定义，它是从客户角度出发的。按照这种定义，客户把质量定义为产品或服务是否满足了客户的需求，换一种说法就是“适用”。通常在客户的“需求规约”（见附录C）里会描述产品的用途。需求规约是最重要的文档，质量体系的问题都要围绕它来解决。另外，在客户的需求规约中还要描述质量的属性，例如对易用性、可移植性和可复用性的描述。易用性是指用户和软件应用程序交互时的容易程度；可移植性是指软件系统在不同的硬件平台上执行时的兼容性；可复用性是指把一个软件系统中的组件复用在另一个软件系统中。

每个人都应该对质量负责。然而，下面列举的许多人所持有的混淆不清的看法，实际上妨碍了对质量的追求。

- 质量需要承诺，尤其是来自最高管理层的承诺。为了实现高质量，要求管理层与员工紧密合作。
- 许多人认为无缺陷的产品和服务是不可能的，产品有一定的缺陷是正常的，是可以接受的。
- 人们经常把质量与成本联系在一起，即高质量就意味着高成本，这样在设计质量和构造质量上就出现了混淆。
- 质量要求需求规约足够详细，这样生产出的产品才能对照规约从数量上进行检测。但是许多组织都没有能力或不愿意花很多精力去写足够详细的规约。
- 技术人员常常认为标准会抑制他们的创造性，因此不愿意遵从标准的规定。但是，为了保证产品质量，他们必须遵守具有明确定义的标准和规程。

1.2 预防与检测

质量是不能通过评估已经生产出来的产品来达到的，因此，目标应该是首先去预防产品缺

陷的产生，并且使产品可以通过质量保证量度进行评估。质量保证量度包括运用软件开发标准将开发过程结构化，并在开发过程中运用方法、技术和工具。软件中没有检测出来的缺陷可能会带来几百万元的损失，因此有必要发展独立的测试，并且应该由公司来执行，而不是由软件系统的开发人员来执行。

就质量管理程序而言，不仅有必要进行产品评估，过程评估也同样非常必要，例如编码标准的文档化，标准、方法和工具的规定和使用，数据备份的规程，测试方法，变更管理，缺陷记录及修正等。

质量管理可以降低产品的成本，因为越早发现缺陷并加以修正，就越能减少在最终带来的成本。随着自动测试工具的出现，虽然最初要做一些投资，但从长远来看，将会获得高质量的产品，并能降低维护成本。

有效质量管理的全部成本是由以下4部分成本的总和构成的：预防、检测、内部故障和外部故障。预防成本是由最初为防止缺陷出现而采取的一系列行动产生的。检测成本是由测量、评估、审计带来的，其目的是使产品或服务符合标准和规范。内部故障成本是用来解决在发货前发现的缺陷时发生的成本。外部故障成本则是用来解决产品发布以后才发现的缺陷时产生的成本。后者可能是破坏性的，因为这样会损坏公司的声誉，并对将来销售不利。

6

要想最大限度地获得回报，就要采取预防措施。重视预防成本就能降低产品到客户那里时的缺陷数，从而提高产品的质量，降低生产和维护的成本。

1.3 验证与确认

验证贯穿在整个开发生命周期中，用来评价产品是否满足了在前面一些已经正确完成的活动中定义的需求，而确认发生在生命周期的末尾，用来检查系统是否满足了客户的需求。已经证明，如果产品满足了用户的期望值，就能确保可执行系统如规约描述的那样运行。与验证相比，测试产品的创建与确认的关系更加紧密。从传统意义上讲，软件测试一直被认为是一个确认过程，也就是说，被当作生命周期的一个阶段。在程序编写完成后，对系统进行确认或测试，以确定其功能性和可操作性。

如果把验证整合到测试过程中，测试将贯穿产品开发的整个生命周期。实践证明，为了获得最好的结果，在测试过程中可以把验证与确认结合在一起。在软件开发生命周期中，验证包括评审、分析和测试的系统化过程，从软件需求阶段一直到代码编写阶段。验证保证了软件开发与维护的质量。同时，验证使得开发过程变得更加有组织、更加系统化，从而使开发出的应用程序很容易被第三方理解和评估。

验证是在大约20多年前为满足航空工业系统软件极高的可靠性要求而出现的，因为程序中的一个错误就可能导致整个任务失败，造成巨大的时间和经济损失，甚至威胁到生命安全。验证的概念包括两个基本标准：软件必须能适当地并正确地执行预期的功能，并且，软件一定不能自行执行某个功能或某些功能的组合，那样会降低整个系统的性能。验证的总目标就是要保证在软件生命周期中开发出的每个软件产品都能满足软件需求文档中描述的客户需求和目标。

验证还在软件文档各章节相关部分的需求规约之间建立了联系。努力做好全面的验证工作能够确保对规约中提到的所有软件性能和质量需求进行充分的测试，并且在发生变更后这些测试结果可以复现。验证是一个“持续改进过程”，没有明确的终点。为了维护配置和操作完整性，应该在系统的整个生命周期中都使用验证。

7

验证确保软件能实现预期的功能及所需的特征（如可移植性），增加了使软件只包含个别错误的可能性（即最终的产品中的错误数是可接受的）。验证为紧密监控软件开发项目提供了一个