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

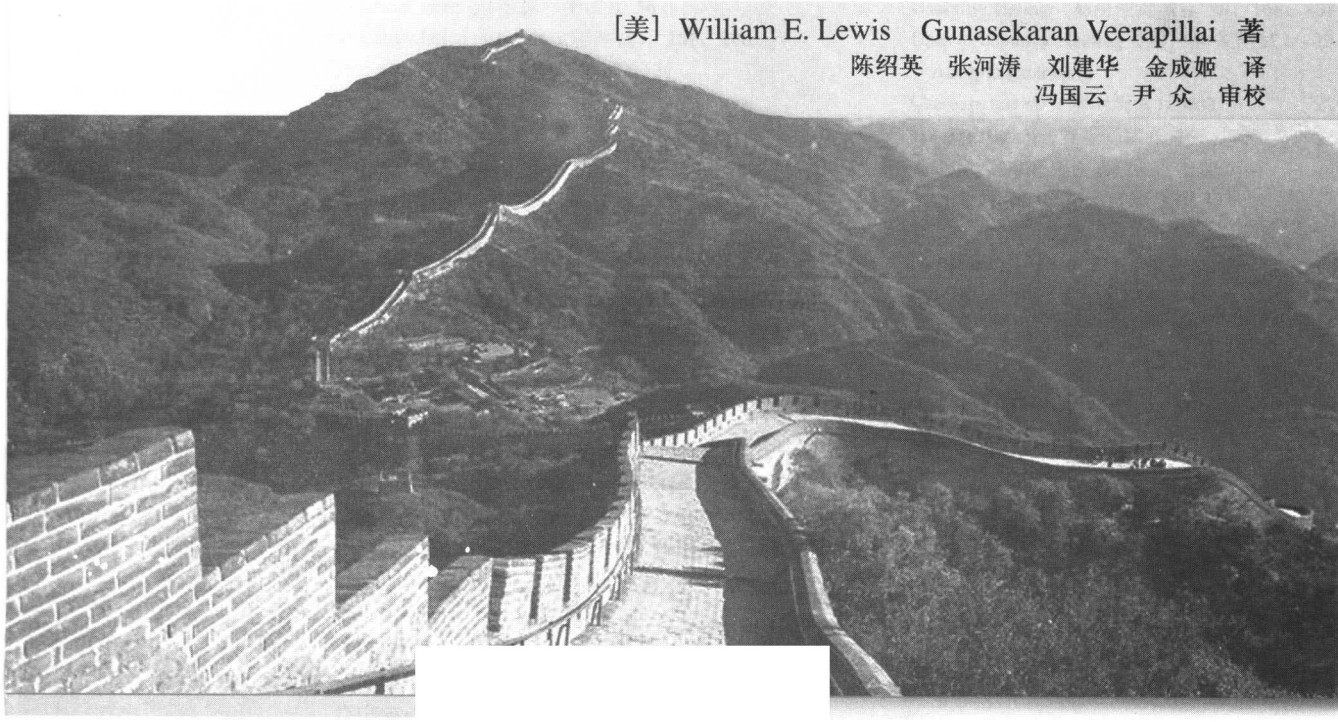
人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵计算机科学丛书

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

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方法提供有力的参考。

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

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。

第1章 质量保证框架

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 6
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。

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。在软件开发的生命周期中，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 7
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。

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