TURING 图灵程序设计丛书

CRC Press
Taylor & Francis Group

Software Testing
A Craftsman's Approach
Third Edition

软件测试（第3版）

[美] Paul C. Jorgensen 著
李海峰 马琳 译

- 国外众多大学采用的优秀教材
- 理论与实践的完美结合
- 涵盖软件标准和开发方法的最新进展

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

Software Testing
A Craftsman's Approach
Third Edition

软件测试（第3版）

[美] Paul C. Jorgensen 著
李海峰 马琳 译

人民邮电出版社
北京

图书在版编目（CIP）数据

软件测试 : 第3版 / (美) 乔根森(Jorgensen, P. C.) 著 ; 李海峰, 马琳译. -- 北京 : 人民邮电出版社, 2011.3
(图灵程序设计丛书)
书名原文: Software Testing:A Craftsman' s Approach, Third Edition
ISBN 978-7-115-24799-5

Ⅰ. ①软… Ⅱ. ①乔… ②李… ③马… Ⅲ. ①软件—测试 Ⅳ. ①TP311.5

中国版本图书馆CIP数据核字(2011)第009935号

内 容 提 要

本书是经典的软件测试教材。书中对基础知识、方法提供了系统的综合阐述，既涉及基于模型的开发又介绍测试驱动的开发，做到了理论与实践的完美结合，反映了软件标准和开发的最新进展和变化。

本书适合作为高等院校计算机学院及软件学院相关专业软件测试课程的教材，也是软件测试领域技术人员的理想参考书。

图灵程序设计丛书

软件测试（第3版）

◆ 著 [美] Paul C. Jorgensen
译 李海峰 马 琳
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

◆ 开本：787×1092 1/16
印张：21.5
字数：536千字 2011年3月第1版
印数：1－3 000册 2011年3月北京第1次印刷

著作权合同登记号 图字：01-2008-2206号

ISBN 978-7-115-24799-5

定价：59.00元

读者服务热线：(010)51095186 印装质量热线：(010)67129223
反盗版热线：(010)67171154

译 者 序

很长时间以来，我们一直关注着软件测试技术的发展。工作在计算机科学与技术领域中，如何确保软件产品在实际工作中“不出错”，是我们时刻都要面对的一个非常现实的问题。目前大量关于软件测试方面的专著和教材已经面市，从中可以看到许多不同的测试原理与测试方法。如何科学地选择和运用这些测试原理与技术，在很大程度上取决于个人的经验。同时，对诸如如何评价软件测试的结果，测试后的软件是否还有残留缺陷，残留的缺陷对软件有何影响等问题，目前还没有一个统一的评估标准。因此，我们认为，软件开发人员应该是软件测试的行家，软件测试人员也应该是软件开发的高手。在大力培养软件开发人才的同时大力培养软件测试人才，并特别重视其对科学的思维方式的培养，对软件行业的发展至关重要。

原书是作者在总结长期从事软件设计、开发与测试工作经验的基础上，融合近年在软件测试教学工作中的心得体会而写成的。本书是第 3 版，较前两版最主要的改动是增加了在更高层次上对软件测试整体策略的全面讨论。本书的重要特点是：对软件测试理论与技术的介绍层次分明、全面精当；以若干实例为线索展开内容，循序渐进，便于读者掌握；在很多章节的最后，还提供深入的对比和讨论，总结了在软件测试中普遍存在的实际问题，精辟深刻；此外，原书在语言上不拘一格，多有诙谐之处，这也是我们翻译过程中所要面对的一项挑战。

在翻译过程中，我们在忠实于原文的同时，针对软件测试工作所涉及的各种理论与技术进行了整理和推敲，对基础理论和技术概念尽可能采用相关学科的主流说法，对新技术和新名词尽可能使用业界当前流行的说法，以期使全书更易于理解、更标准和更科学；对作者举例时讲述的故事，则力求通俗流畅，保持原书的韵味。

本书由李海峰翻译第 1~10 章和第 21~23 章，马琳翻译第 11~20 章、第 24 章和第 25 章，李海峰对全书进行了审校和润色。在翻译工作中我们借鉴了本书前两版和其他同类著作，在此我们向其作译者表示深深的谢意。对人民邮电出版社编辑的艰苦工作和大力支持表示深深的感谢。

好作品必然是经过了反复修改才日臻完善的。由于时间有限，加上译者的知识水平和实际工作经验有限，不当之处在所难免，恳请读者和同行批评指正，提出宝贵意见。

李海峰　马　琳

2010 年 10 月于哈工大

第 3 版前言

本书第 2 版问世已经有 5 年了，由于需求的增长和技术的推动，软件测试已经开始了新的复兴。最重要的变化是敏捷软件开发思想被广泛接受并逐渐成为主流。各种各样的敏捷软件开发方法对软件测试都产生了重要的影响。与此同时，用于软件开发和软件测试的基于模型的方法也赢得了众多的追随者。本书第 3 版的第六部分将分析 21 世纪出现的一些新兴软件测试技术。前五个部分基本上没有做大的改动，只是纠正了一些错误。

多年以来，很多热心读者给予了我很大的帮助，指出了书中存在的问题，包括排版错误和其他一些较严重的问题。这里要特别感谢明尼苏达州的 Neil Bitzenhofer、北京的韩柯、伊利诺伊州的 Jacob Minidor 和俄亥俄州的 Jim Davenport。此外，在过去的 15 年里，我带的研究生们也提出了许多有益的建议。还要特别感谢我的两位同事 Roger Ferguson 博士和 Christian Trefftz 博士的巨大帮助。

本书已经在几十个国家被采用为软件测试课程的教材。为了满足教师和学生两方面的需求，我增加了很多习题。

5 年以来，我本人也变了许多。我结识了一些提顿族（美国达科他的印第安人）朋友，他们对本书第 23 章的内容有一些有趣的影响。这里我只说一句：我们同在一个世界！(Hecatuyelo!)

Paul C. Jorgensen

2007 年 12 月于密歇根州罗克福德市

第 2 版前言

我为本书第 1 版写前言是在 7 年以前。这 7 年中新象迭出，因此编写第 2 版势在必行。其中最重要的变化是 UML（统一建模语言）已经成为面向对象软件规范和设计的一种标准，所以第 2 版的主要变化是增加了第五部分（共 5 章），专门来讨论面向对象软件的测试，其中绝大部分内容都以 UML 为基础。

第二个主要变化是把第 1 版中所有用 Pascal 语言编写的例子都替换成了独立于语言的伪代码，并且做了详细描述，可以用 CRC 出版社 Web 站点（www.crcpress.com）提供的 Visual Basic 可执行模块来支持。还增加了一些用于说明面向对象软件测试的例子，此外还有其他许多修改之处，最重要的是更新了对等价类测试的介绍，加入了一个贯穿全书的示例，更详细地介绍了集成测试。

我很高兴地看到本书第 1 版成为了由 ACM 和 IEEE 计算机学会（www.swebok.org）联合编写的“软件工程知识体系”软件测试方面试用标准的主要参考文献之一。同时这也使得努力纠正第 1 版中存在的错误成为我的一块心病。韩国一位读者给我寄来了一份第 1 版的错误清单，共有 38 处错误，听我讲授软件测试课程的研究生也找出了其他一些错误。这和测试过程非常相似：我改正了所有已知的错误，所以本书的编辑就说没必要再去找新的错误了。如果读者发现了任何问题，请及时通知我，纠正它们是我的责任。我的电子邮件地址是 jorgensp@gvsu.edu。

我要感谢 CRC 出版社的 Jerry Papke 和 Helena Redshaw 耐心细致的工作，还要感谢我的朋友和同事 Roger Ferguson 教授，感谢他对第五部分新内容所提供的长期帮助，特别是帮助我写成那个面向对象日历的示例。从某种意义上讲，Roger 实际上是第 16 章至第 20 章好几稿的测试者。

Paul C. Jorgensen

2002 年 5 月于密歇根州罗克福德市

第 1 版前言

我们聚集在会议室门口，通过门上的小窗户向里面张望。会议室里，一位新来的软件设计师在会议桌上摊开了源程序清单，手拿一个长柄水晶放大镜正在仔仔细细地检查着源代码，并时不时地在源程序清单上画红圈圈。后来，同事问这个设计师当时在做什么，他冷冷地回答："在找程序中的 bug。"这是发生在 20 世纪 80 年代中期的一件真事，那时候人们对水晶的魔力寄予厚望。

从某种意义上讲，本书的目的就是要向读者提供一套更好的"水晶放大镜"。正如本书英文书名所示，我认为软件（和系统）测试是一种技艺，而且我自己已经在一定程度上掌握了这种技艺。在我 20 多年的电话交换机系统研发经历中，大约有三分之一的时间是花在测试上的，包括定义测试方法和标准，安排国际长途电话收费系统的系统测试，描述并帮助构建两个测试工具（现在我们称之为 CASE 工具），还有就是完成一大堆平淡无奇的手工测试工作。过去 7 年里，我一直在大学里为研究生讲授软件工程课程，学术研究主要集中在规格说明和软件测试两方面。我认同牛津学派（Oxford Method）所信奉的：知识，只有在你给别人讲授过之后，你自己才能真正领会。听我的软件测试课程的学生都是当地公司的全职员工。所以请相信我，我讲的都是真正有用的东西。本书就是以我的课程讲义和学生作业为基础总结出来的。

我自认为是一名软件工程师，但是当我将软件领域相关知识的精度和深度同其他传统工程领域进行比较时，我又深感不安。Myers 的著作《软件测试的艺术》（*The Art of Software Testing*）刚出版时，我和一个同事正要到意大利去完成一个项目。在去机场的路上，我们走进麻省理工学院书店恰好买到了这本书。在过去的 15 年中，我相信软件测试已经从一门艺术发展成一种技艺。起初我想给本书起名为《软件测试的技艺》（*The Craft of Software Testing*），但就在本书即将完成的时候，另一本同名的书出版了。不过这正好印证了我将软件测试视为一种技艺的观点。然而在软件测试能够成为一门科学之前，还有很长的一段路要走。

掌握任何一种技艺都需要理解工具和材料的性能与局限性。一个好木匠有各种各样的工具，选哪种工具最合适取决于要制作的东西和所使用的木材。在软件开发生命周期的传统瀑布模型的各个阶段，测试是最适合做精确分析的。将软件测试提升为一种技艺，要求测试技艺师了解基本的工具。为此，本书第 3 章和第 4 章将介绍在其他各章中涉及的数学基础知识。

数学是一种描述工具，有助于人们更好地理解被测试的软件。但仅有精确的描述还不够，还必须拥有精良的技术和判断力来选择恰当的测试方法，并付诸实施。这些就是本书第二部分和第三部分的目标，这两个部分将讨论基本的功能测试和结构测试技术。这些技术将被应用到第 2 章介绍的各个例子上，而这些实例将贯穿全书。第四部分将这些技术用于集成测试和系统测试，以及面向对象测试。在这些层次上，更重要的是明确测试些什么，而不是怎样测试，因此讨论的重点是需求规格说明。第四部分将研究对软件控制系统中人机交互的测试，并简要讨

论客户/服务器系统。

具有讽刺意味的是，介绍测试的书也是会有错误的。尽管编审人员已经付出了艰辛的劳动，但是书中肯定还有错误，对此我应负全责。

1977年我参加了一个由Edward Miller组织的测试研讨会，后来Edward Miller成为了软件测试界的领军人物。在那次研讨会上，Miller不遗余力地向我们证明测试工作不一定是一件烦琐的苦差事，而是软件开发中富有创造性和趣味性工作的一部分。本书的目的就是要使读者成为测试的行家，体味一个真正的技师在出色地完成一项工作之后油然而生的自豪感和愉悦感。

Paul C. Jorgensen

1995年1月于密歇根州罗克福德市

目　录

第一部分　数学基础

第二部分 功能测试

第三部分 结构测试

第五部分 面向对象测试

第六部分 新兴测试技术

第一部分

数学基础

本部分内容

第1章

测 试 概 述

为什么要测试？最主要的目的有两个：一是对质量或可接受性作出评判，二是发现存在的问题。之所以要测试，是因为人经常会出错，特别是在软件领域和采用软件控制的系统中，这个问题尤为突出。本章的目标是给出软件测试的全貌，而本书的其他各章都将在这个知识框架下展开。

1.1 基本概念

在许多测试方面的文献中，名词术语的使用都比较混乱（有时不统一），究其原因，可能是因为测试技术在近几十年中不断地演化进步，而且文献作者所处领域不同也有差异。全书所采用的术语都取自美国 IEEE 计算机学会颁布的技术标准。我们首先研究几个有用的术语。

- **错误**（error）：人会做错事。错误的同义词是过失（mistake）。编程时出的错称为“bug”。错误很容易传递和放大，比如需求分析方面的错误在系统设计时有可能会被放大，而且在编码时还会被进一步放大。
- **故障**（fault）：故障是错误的后果。更确切地说，故障是错误的具体表现形式，比如文字叙述、数据流图、层次结构图、源代码等。与把编程错误称为 bug 类似，故障的同义词是缺陷（defect）。故障可能难以捕获。比如，设计人员犯下一个遗漏错误，所导致的故障可能只是在表现上丢掉了一些应有的内容。这里也可以把故障进一步细分为过失故障和遗漏故障。如果在表象中添加了不正确的信息，这是过失故障；而未输入正确的信息，则是遗漏故障。在这两类故障中，遗漏故障更难检测和纠正。
- **失效**（failure）：发生故障会导致失效。失效具有两个很微妙的特征：（1）失效只出现在程序的可执行表现形式中，通常是源代码，确切地说是加载后的目标代码；（2）这样定义的失效只和过失故障有关。那么如何处理遗漏故障所对应的失效呢？进一步说，对于不轻易发生的故障，或者长期不发生的故障，情况又会怎样呢？米开朗基罗（Michelangelo）病毒就是这种故障的一个例子，它只有在 3 月 6 日（米开朗基罗生日）才执行。采用代码评审能够通过查找故障来避免失效。实际上，好的代码评审同样能检查出遗漏故障来。
- **事故**（incident）：失效发生时，用户（或客户和测试人员）可能察觉得到，也可能察觉不到。事故是与失效相关联的症状，它警示用户有失效发生。
- **测试**（test）：测试显然要考虑到错误、故障、失效和事故等诸多问题。测试是利用测试用例来操作软件的活动。测试有两个明确目标：找出失效问题和证实软件执行的正确性。

- **测试用例**（test case）：每个测试用例都有一个用例标识，并与程序的行为密切相关。每个测试用例还包括若干输入和期望输出。

图 1-1 给出了一种测试的生命周期模型。从中可以看出，在软件开发阶段有三个地方可能会产生错误，由此引发的故障将传递到后续开发过程中。一位资深的测试专家将此生命周期模型总结为：前三个阶段是“注入 bug”阶段，测试阶段是“发现 bug”阶段，后面的三个阶段是“清除 bug”阶段（Poston, 1990）。“故障解决”实际上也是一处可能产生错误（以及新故障）的地方。如果修复操作导致原先正确的软件出现异常行为，这样的修复就是不完善的。本书后面讨论回归测试时还要进一步研究这个问题。

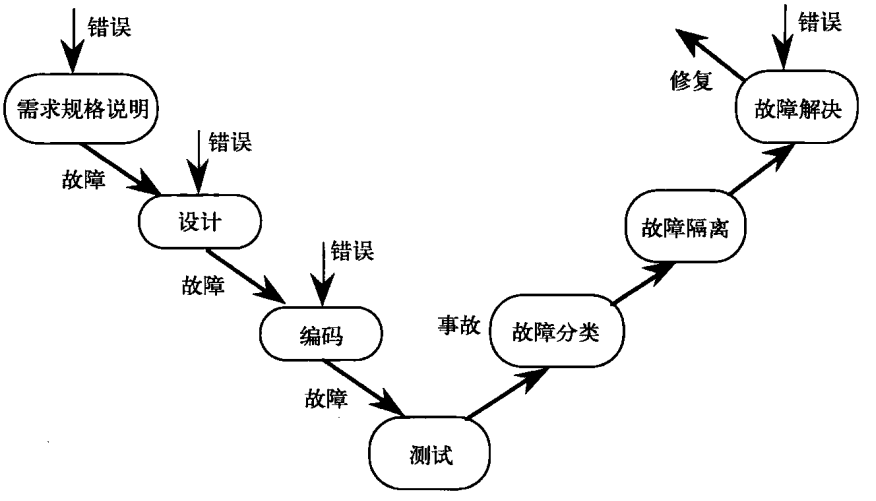

图 1-1 测试的生命周期

从这一系列术语可以看出，测试用例在测试中占核心地位。测试的过程还可以进一步细分为若干独立的步骤：测试计划制订、测试用例开发、测试用例运行以及测试结果评估等。本书的重点是研究如何构造有用的测试用例集合。

1.2 测试用例

软件测试的本质是为被测试对象建立一个测试用例的集合。在进一步研究之前，首先要弄清楚测试用例中应该包含哪些必要的信息。

- **输入**，实际上有两类，即前置条件（在测试用例运行之前已经存在的环境因素）和用某些测试方法所构建的实际输入。
- **期望输出**，也有两类，即后置条件和程序的实际输出。

测试用例的输出部分常常会被忽视。这是很不应该的，因为输出通常是测试用例最难的部分。举个例子，假设你要测试一个软件，已知美国联邦航空管理局（FAA）的航线限制和飞行当天的气象数据，这个软件要确定飞机的最佳航线。然而你又怎么知道这个最佳航线到底是什么呢？这个问题可能会有各种各样的答案。从学术角度来看，任何问题都一定会有一个确切的答案。从行业角度来看，可以采用“参考测试”（reference testing）的办法，由专家用户来参与

系统的测试，作出评价，比如，给定一组测试用例输入，由这些专家来评价系统运行后的实际输出是否可以接受。

测试活动包括建立必要的前置条件，给出测试用例输入，观察输出结果，将实际输出与期望输出进行比较，然后在保证预期后置条件成立的情况下，判断测试能否通过。

在一个设计完善的测试用例中，还应该包括一些用来支持测试管理的其他信息（如图 1-2 所示）。测试用例应该有一个用例标识（ID）和一个存在的理由（主要目的是进行需求跟踪）。记录测试用例的执行过程也是很有用的，需要记录该测试用例的运行时间和执行人，每次执行是通过还是失败，以及所测试软件的版本号等信息。由此可以看出，测试用例显然是非常有价值的，至少和源代码一样珍贵。所以，需要对测试用例进行开发、评审、使用、管理和保存。

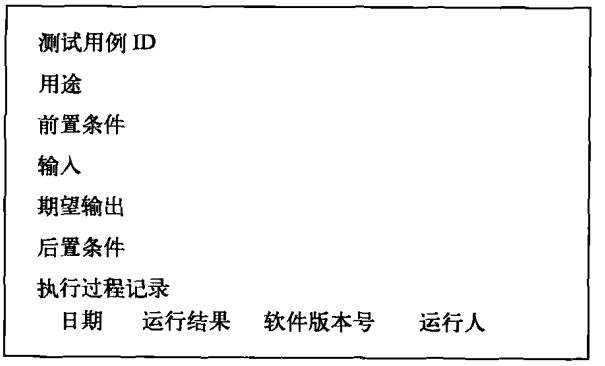

图 1-2　典型测试用例所包含的信息

1.3　通过维恩图来考察测试

从本质上讲，测试关心的是程序的行为，而程序的行为同软件（或系统）开发人员所常用的结构视图并没有直接关系。

结构视图与行为视图的最明显差别在于：前者侧重于“程序是什么”，而后者则关注“程序干什么”。然而不断困扰测试人员的一个问题是：基础性的文档通常都是由开发人员编写的，并且是为开发人员服务的，所以文档就很自然地强调程序的结构信息而不是行为信息。本节将开发一种简单的维恩图，借以明确一些测试中很微妙的问题。

我们先来看看程序的行为空间（注意，此处我们研究的是测试的本质），对于给定的程序及其规格说明，考察其规格说明所规定的行为集合 S 和编程实现的行为集合 P。图 1-3 给出了程序行为空间与 S 以及 P 之间的关系。在程序的所有可能行为中，规定的行为都在圆圈 S 内，所有实际实现的行为都在圆圈 P 中（此处要留心 P 同全空间 U 之间存在的差异）。利用这个维恩图，可以清楚地看到测试人员所遇到的问题。如果某些规定行为未经编程实现，情况会怎样呢？用前面提到的术语来讲，这些是遗漏故障。类似地，如果编程实现的某些行为不是规格说明所

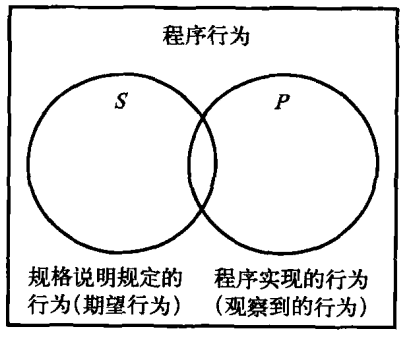

图 1-3　程序的规定行为与实现行为