HZ BOOKS
华章教育

计 算 机 科 学 丛 书

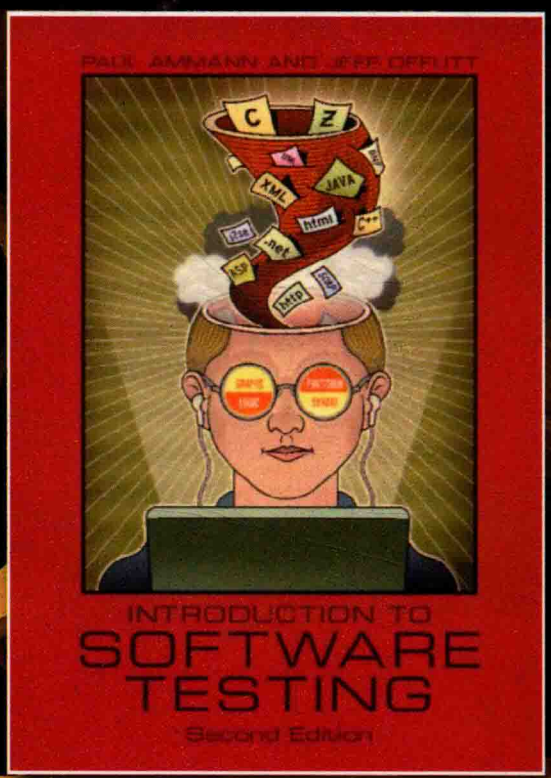
CAMBRIDGE

原书第2版

软件测试基础

[美] 保罗·阿曼（Paul Ammann） 杰夫·奥法特（Jeff Offutt）著
李楠 译

Introduction to Software Testing
Second Edition



机械工业出版社
China Machine Press

计 算 机 科 学 丛 书

原书第2版

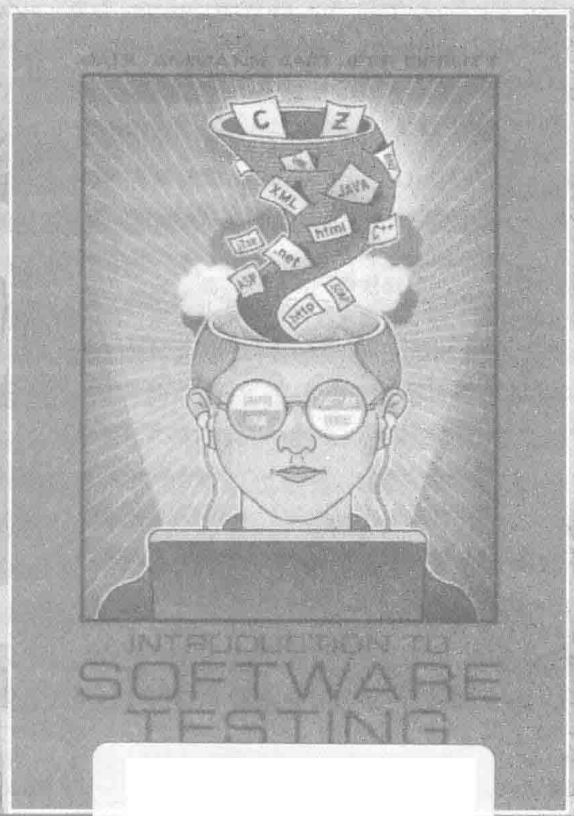
软件测试基础

[美] 保罗·阿曼（Paul Ammann） 杰夫·奥法特（Jeff Offutt）著

李楠 译

Introduction to Software Testing

Second Edition



机械工业出版社
China Machine Press

图书在版编目（CIP）数据

软件测试基础（原书第 2 版）/（美）保罗·阿曼（Paul Ammann），（美）杰夫·奥法特（Jeff Offutt）著；李楠译．—北京：机械工业出版社，2018.10
（计算机科学丛书）
书名原文：Introduction to Software Testing, Second Edition

ISBN 978-7-111-61129-5

I. 软… II. ①保… ②杰… ③李… III. 软件－测试 IV. TP311.55

中国版本图书馆 CIP 数据核字（2018）第 234093 号

本书版权登记号：图字 01-2018-0352

本书采用了一种创新性的方法来解释软件测试：软件测试被定义为一个将具有通用目的且精确的准则应用于软件结构或模型的过程。本书覆盖了软件测试的新发展，包括测试现代软件类型（比如面向对象、网络应用程序和嵌入式软件）的技术。第 2 版极大地扩展了基础知识，详尽地讨论了测试自动化框架，还增加了新的例子以及大量的练习。

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码：100037）
责任编辑：赵　静　　责任校对：殷　虹
印　　刷：北京文昌阁彩色印刷有限责任公司　　版　　次：2018 年 11 月第 1 版第 1 次印刷
开　　本：185mm×260mm　1/16　　印　　张：18.5
书　　号：ISBN 978-7-111-61129-5　　定　　价：79.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991　88361066　　投稿热线：（010）88379604
购书热线：（010）68326294　88379649　68995259　　读者信箱：hzjsj@hzbook.com

文艺复兴以来，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的优势，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭示了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短的现状下，美国等发达国家在其计算机科学发展的几十年间积淀和发展的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起到积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章公司较早意识到“出版要为教育服务”。自 1998 年开始，我们就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与 Pearson、McGraw-Hill、Elsevier、MIT、John Wiley & Sons、Cengage 等世界著名出版公司建立了良好的合作关系，从它们现有的数百种教材中甄选出 Andrew S. Tanenbaum、Bjarne Stroustrup、Brian W. Kernighan、Dennis Ritchie、Jim Gray、Afred V. Aho、John E. Hopcroft、Jeffrey D. Ullman、Abraham Silberschatz、William Stallings、Donald E. Knuth、John L. Hennessy、Larry L. Peterson 等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及珍藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力相助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专门为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近 500 个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证。随着计算机科学与技术专业学科建设的不断完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都将步入一个新的阶段，我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

华章网站：www.hzbook.com
电子邮件：hzjsj@hzbook.com
联系电话：（010）88379604
联系地址：北京市西城区百万庄南街 1 号
邮政编码：100037

华章科技图书出版中心

译 者 序

Introduction to Software Testing, Second Edition

伴随着软件的大规模应用以及与人们日常生活的紧密结合，软件已经改变了人们的行为方式。同时，软件质量也在某种程度上决定着人们的生活质量，所以软件质量变得尤其重要。软件测试是保证软件质量最有效也最常用的手段。与早期相比，软件测试活动已经处于软件开发的中心地位。在测试驱动开发（test-driven development）中，开发者需要写测试用例来帮助设计和完成代码。而在目前流行的敏捷开发（agile development）中，在执行每个开发故事（story）之前，开发者都必须与测试者一起开会，并且在代码实现之前就需要定义如何测试软件。基于这样的背景以及测试自动化的潮流，软件测试领域的两位权威专家 Paul Ammann 和 Jeff Offutt 推出了目前全球颇具影响力的软件测试教科书——《软件测试基础》，即本书。

本书的特点非常鲜明：紧密地围绕模型驱动测试设计来展开。模型驱动测试设计指的是在自动化测试中建立模型来产生测试需求，生成测试用例覆盖需求，执行并评估测试用例的过程。模型驱动设计中最重要也最难的部分是测试用例设计。测试用例的有效性和成本直接决定了测试的成功与否，这部分也是软件测试最主要的研究方向。生成测试用例的技术有数百种甚至更多，但 Paul 和 Jeff 在本书中创造性地将几乎所有技术归纳为对四种基本模型的应用：图、逻辑表达式、语法和输入域。第 2 版不仅保留了这些模型，还对第一部分软件基础进行了扩展：明确地提出了模型驱动测试设计，并且加入了测试自动化框架和测试驱动程序开发的内容。

本书自 2008 年第 1 版发行以来，已经用于美国、欧洲、中国数所大学的软件测试课程当中，是当今软件测试领域最受欢迎也最具影响力的教科书。作为一本教科书，本书包括了软件测试领域过去 40 年研究的精华，系统地介绍了软件测试的基础理论，比如覆盖理论，包含理论、RIPR 模型等。在讲述理论的同时，本书还重点考虑了与工业界的结合。在包含通用的测试过程和测试计划之外，本书主要强调的是在自动化测试的每个阶段，实践者应该选择在正确的抽象层次上和模型中应用有效合理的测试技术。注意，本书没有描述如何使用具体的测试工具，因为如果那样的话，当工具被淘汰时，书中的内容就会过时。通过讲述理论以及描述与实际结合的样例，本书试图激发实践者的智慧和潜力来在实践中应用这些理论，并提出更有效和有效率的方法。每章末尾包括参考文献注解，这些注解详尽地解释了所引用的每个概念和技术的出处，并对不同的技术进行比较。这对软件测试方向的博士生和研究人员有很大的帮助。事实上，本书在学术论文中的引用数已经超过了 1 300 次。

译者观察到，目前国内有很多关于测试工具操作以及测试过程和方法实践的书籍，但是缺乏系统讲述软件测试理论的书籍。本书正好填补了这方面的空白。从译者参加的多个国内外软件测试研究和实际项目来看，本书作为参考资料经常能够发挥巨大的作用。有人认为现在很多 IT 公司并没有应用太多书中的技术，所以在实践中本书作用不大。但译者认为：1）本书不仅覆盖早期理论研究的沉淀，还囊括了一些最新研究的成果，这些成果的价值将会在未来得以体现，所以本书不仅应用于当代，更是面向未来；2）大部分 IT 公司的要求是

在保证一定质量的前提下，对产品进行快速迭代。所以目前看来有些复杂的技术，由于成本原因并没有得到应用（同时也给予了实践者改进它们的机会）。但是，对其他行业软件（如航天航空和高铁上的安全关键软件）来说，使用复杂技术对其进行严格的测试是必要的，同时也是具有一定成本边际效应的（因为需求相对固定且潜在的人民生命财产损失巨大）。综上所述，译者认为本书为各个行业的软件测试都提供了理论和实践上的全方位支持。

译者在过去近一年中，尽自己的最大努力，对每个概念、术语、定义的翻译都进行了认真细致的斟酌。译者也参考了第 1 版的中文翻译，尽可能多地保留了之前的术语译法。当然，有些概念和术语都比较新，目前在国内还没有统一的翻译标准。欢迎各位专家、同仁、读者提出宝贵意见，以期尽快对某些术语的翻译达成统一意见。在翻译的过程中，译者就一些疑问向作者 Paul 和 Jeff 进行了请教。对某些不易理解的部分，译者加了注释；对错误的部分，译者进行了改正。所以，从理论上来讲，本书比英语原版要更加完善。当然任何书籍的翻译都很难做到完美，在此希望读者提出指正意见和建议。

感谢原书作者 Paul Ammann 和 Jeff Offutt 对此次翻译的支持。还要感谢华章公司的编辑朱捷和赵静，他们对此次翻译工作提出了很多宝贵的建议并做了大量的校订工作。最后，感谢太太郭昀在翻译过程中为家庭的付出以及对我工作的理解。当然也要感谢我的父母、妹妹、岳父岳母在精神上给予我的鼓励。

自本书的第 1 版发行以来，软件测试领域已经发生了太多变化。高水平的测试现在在工业界已经变得非常普遍。测试自动化已经无处不在，工业界绝大多数领域都默认必须使用测试自动化。敏捷过程和测试驱动开发变得广为人知而且许多人都在使用。更多的学校在本科和研究生阶段开设了软件测试的课程。ACM 关于软件工程的课程大纲在很多地方都包括了软件测试，并且把它设置为强烈推荐的课程 [Ardis et al.，2015]。

第 2 版包括了新的特点和内容，同时保留了第 1 版中深受数百位教师喜欢的结构、理念和在线资源。

第 2 版的新内容

当拿到一本书的新版本时，任何教师要做的第一件事就是研究所讲的课程中需要做哪些改动。因为我们已经经历过很多次这样的情况了，所以我们想让读者很容易地明白第 2 版的改动。

第 1 版	第 2 版	主题
第一部分　软件测试基础		
第 1 章	第 1 章	为什么测试软件（动机）
	第 2 章	模型驱动测试设计（抽象）
	第 3 章	测试自动化（JUnit）
	第 4 章	测试优先（TDD）
	第 5 章	基于准则的测试设计（准则）
第二部分　覆盖准则		
第 2 章	第 7 章	图覆盖
第 3 章	第 8 章	逻辑覆盖
第 4 章	第 9 章	基于语法的测试
第 5 章	第 6 章	输入空间划分
第三部分　实践中的测试		
第 6 章	第 10 章	管理测试过程
	第 11 章	编写测试计划
	第 12 章	测试实现
	第 13 章	软件演化中的回归测试
	第 14 章	编写有效的测试预言
第 7 章	N/A	技术
第 8 章	N/A	工具
第 9 章	N/A	挑战

第 2 版最明显、最大的改动是将第 1 版中导言性质的第 1 章扩展成了 5 个不同的章节。这个重大的扩展使本书变得更加完善。新版的第一部分是由我们的课程讲义发展而来的。第 1 版发行之后，我们开始不断地向测试课程中添加更多的基础内容。这些新的想法最终组织成了这 5 个新的章节。新版第 1 章用到了很多第 1 版第 1 章的内容，包括动机和基本概念。第 1 章结束的时候包括了一段来自 2002 年的 RTI 报告，这篇报告讨论的是在开发晚期才进

行测试所造成的巨大成本。每个软件测试研究的项目提案都会引用这个重要的调查研究结果。在完成第 1 版后，我们意识到这本书的关键创新点在于将测试设计成抽象的活动，独立于用来生成测试用例的软件工件（artifact）。这个观点暗示软件设计已经变成一个和以往不同的过程。这样的想法引出了第 2 章，这一章讲述如何将测试准则和实践相结合。在我们做咨询的过程中，我们已经帮助软件公司包含这一模型以修正其测试过程。

第 1 版中有个遗憾是没有提及 JUnit 或其他的测试自动化架构。2016 年，JUnit 已经在工业界广泛使用，而且通常用在 CS1 和 CS2 的课堂上给作业自动打分。第 3 章改正了这个疏忽。在这一章里，我们叙述了测试自动化的概况，说明了实施测试自动化的难点，也明确地教授了 JUnit。虽然本书所讲的内容在很大程度上并不依附于某个具体技术，但在全书的例子和练习中使用统一的测试架构是方便读者理解的。在课堂上，我们通常要求测试必须自动化，也经常要求学生在作业中尝试别的“ *-Unit”单元测试架构，比如 HttpUnit。我们认为在拥有自动化的测试用例之前，测试机构还不具备成功应用测试准则的能力。

很自然地，我们在第 4 章讲到了测试驱动开发（TDD）。虽然 TDD 和本书的其余部分不太一样，但这对测试教育者和研究人员来说却是一个令人激动的主题。原因是 TDD 把测试提前且放到了软件开发的中心位置，测试变成了需求。在第一部分的最后，第 5 章用抽象的方式介绍了测试准则的概念。软心豆粒糖（jelly bean）的例子（尤其是在课堂上讲述这个例子的时候，我们的学生都很喜欢）和其他概念（如包含关系）依然在第 2 版中保留了下来。

第二部分是本书的核心，但在第 2 版中改动最少。2014 年的一天，Jeff 问了 Paul 一个简单的问题：“第二部分四个章节的顺序为什么是现在这样？”答案是惊愕地沉默，因为我们意识到我们从未想过它们应该出现的顺序。事实上，无可争议地处于软件测试中心地位的 RIPR 模型已经给出了一个逻辑顺序。具体来说，输入空间划分不需要可达性、影响或传播（这些概念在第 2 章介绍）。图覆盖准则只需要测试执行“经过”待测软件的一些部分，这就是可达性而没有影响和传播。逻辑覆盖准则需要到达而执行谓词，还要以一种特别的方式使用它进而改变它的结果。这就是说，这个谓词被影响了。最后，语法覆盖不仅要求到达程序的某个地方，同时“变异”的程序状态必须和原程序不同，而且这个不同之处必须在程序执行之后观察到。这就是说，程序状态的变化要传播出来。第 2 版依据 RIPR 模型按顺序列出来这四个概念，而它们所对应章节的要求在递进地增强。从实用的角度来说，我们只是将前一版的第 5 章（现为第 6 章）移到了图覆盖章节（现为第 7 章）之前。

结构上的另一个主要改动是第 2 版**没有**再包含第 1 版中的第 7 章到第 9 章。第 1 版中的这三章已经过时，相比本书的其他部分，这三章用到的频率较少，所以我们决定在重写这部分之前先发表现有的章节。我们计划在第 3 版中更好地描述这三章。

我们还做出了数百处更加细微的改动。最近的研究发现，测试能够成功不仅需要一个错误值传播到输出结果，而且要求自动化测试预言检查合适的输出结果。这就是说，测试预言必须揭示软件失败。因此，新的 RIPR 模型取代了旧的 RIP 模型。本书在一些地方有拓展性或深度的讨论，超出了对概念理解的基本要求。第 2 版现在包括了“深入讨论”。这个附加的讨论部分可能会激发一些学生的兴趣或让他们产生有见解的想法。而对其他学生来说，则不必阅读这些略有难度的讨论。

新的第 6 章现在包含了一个完整的例子。这个例子展示了如何由广泛使用的 Java 类库的接口推出输入域模型（6.4 节）。我们的学生发现这个例子能够帮助他们理解怎样使用输入

空间划分技术。第 1 版有一节是关于如何使用代数来表示图的。虽然我们中的一个人认为这部分内容有趣，但是我们都认同很难找到使用这个技术的动机而且这个技术在实践中很少使用。此外，这个技术也有一些小的缺陷。因此，我们在第 2 版中没有再包含这部分。新的第 8 章（逻辑覆盖）有一个重大的结构改动。DNF 准则（之前在 3.6 节）被合理地放到了本章的前面部分。第 8 章现在以 8.1 节的语义逻辑准则（ACC 和 ICC）开始，然后进入 8.2 节的语法逻辑准则（DNF）。语法逻辑准则也做了改动。我们去掉了 UTPC 准则而加上了 MUTP 和 MNFP 准则。这两个准则和 CUTPNFP 一起组成了 MUMCUT 准则。

整本书（特别是第二部分）中，我们改进了例子，简化了定义，还包括了更多的习题。当第 1 版问世的时候，只有一个包括部分习题答案的手册，而这个手册前前后后花了五年的时间才完成。我们从中得到了一个教训，因此定下一个规矩（而且会一直坚持下去）：不会在没有添加答案的情况下增加习题。读者可以把这个规矩视为对习题的测试。我们很高兴地宣布本书第 2 版的网站**上线**之际就配有完整的习题答案。

第 2 版还改正了很多第 1 版的错误，部分来源于第 1 版网站的错误列表，剩下的发现于我们编写第 2 版的时候。第 2 版的索引也变得更好了。写第 1 版时，我们只用了大约一天完成了索引。这次我们一边写书一边制作索引，而且在成书的最后时刻专门花时间来完善。给未来的作者一个建议，做索引是一项艰巨的任务，不要轻易地把这项工作交给作者之外的人。

第 2 版和第 1 版的相同之处

第 1 版的成功之处在第 2 版中都得到了保留。总体来说，我们以四种结构来归纳测试准则的视角依然是第 2 版的核心组织原则。第 2 版还采用了工程师的视角。我们假设读者是工程师，目的是想用最低的代价写出高质量的软件。本书的所有概念都有理论支撑，但是以实践的方法来展示。这就是说，本书将理论实践化。理论部分从研究文献的角度来说是可靠的，而我们展示了如何将理论应用到实际。

本书还可以当作教材来用。本书有清晰的解释、简单但生动的例子，还有很多适用于课内或课外的练习题。每一章的最后还有参考文献的注解，这可以帮助刚进入研究领域的学生学习软件测试中涉及的更深的思想。本书的网站（https://cs.gmu.edu/~offutt/softwaretest/）[⊖]有很多内容丰富的材料，包括习题答案手册、本书中所有程序样例的列表、高质量的 PowerPoint 讲义，以及帮助学生理解图覆盖、逻辑覆盖和变异分析的软件。我们还用一些视频来做讲解，希望将来能做更多这样的视频。习题答案手册有两种形式。一种是学生答案手册，面向所有人，但是只有一半习题的答案。另一种是教师答案手册，包括所有习题的答案，但是我们只提供给教授软件测试课程的教师。

如何在不同课程中使用本书

我们使用模块化的方法来组织本书的章节。虽然我们以一定的顺序来展示这些章节，但大部分章节都是相互独立的。章节之间没有什么关系，教师几乎可以以任意顺序来使用它们。在我们学校，主要的目标课程是一门大四的课程（SWE 437）和一门研究生第一年的课

⊖ 关于本书教辅资源，只有使用本书作为教材的教师才可以申请，需要的教师可向剑桥大学出版社北京代表处申请，电子邮件 solutions@cambridge.org。——编辑注

程（SWE 637）。有兴趣的读者可以在网上搜索我们的课程（“mason swe 437”或“mason swe 637”）来查看课程安排以及我们是如何使用本书的。这两门课都是必修课。SWE 437在应用计算机科学专业软件工程方向是必修课。SWE 637是软件工程专业硕士的必修课[⊖]。第1和3章可以从两个方面应用在早期的课程（如CS2）中。这样，首先学生在早期就可以充分认识到软件质量的重要性，其次让学生开始学习测试自动化（我们在乔治梅森大学使用JUnit架构）。大二的测试课程可以覆盖第一部分的全部、第二部分的第6章和第三部分的全部或部分。对大二的学生来说，第二部分的其他章节可能有些超出他们的需要，但输入空间划分是高级结构化测试中的入门技术，非常容易理解。在北美计算机科学课程中，通常大三学生要上的一门课是软件工程。本书的第一部分对于这样的课程将会是非常合适的。2016年，我们新开了一门软件测试的研究生高级课程。这门课程包含了最新的知识和目前的研究成果。这门课将会用到第三部分的一些内容、我们正在编写的第四部分内容，还有一些节选的研究论文。

如何讲授软件测试

本书的两位作者在过去的十年中都在学习如何教学。21世纪初，我们授课的模式依然是传统的。我们在课堂上的大部分时间里做演讲，将大量的PowerPoint讲义组织得井井有条，要求学生独立完成课后作业，并且组织一些有挑战性和高强度的考试。第1版的PowerPoint讲义和习题均为此模式所设计。

然而，我们的教学一直在发展中。我们用每周的小测验代替了期中考试。每次测验占用课堂的前15分钟。这样的测验占了整个学期成绩相当大的比例，缓解了期中考试的压力，使学生每周都能跟上进度而不是在考前死记硬背。这样也能帮助我们在学期的早期发现哪些学生毫不费力，哪些学生则比较吃力。

我们学到了一种称为“翻转课堂”的教学模式。我们试验性地开始应用此模式。我们提前录制授课内容，要求学生在课前观看，然后在课堂上完成“课后”作业，而我们则在旁边提供及时的帮助。我们发现这个方法对讲授涉及复杂数学的内容特别有帮助，同时对学习吃力的学生尤其有效果。当教育研究实际验证了传统讲课方式的弊端时，我们开始渐渐地放弃了教师直接演讲两个小时的方式，即所谓的“讲台上的圣人”模式。现在我们经常只讲10～20分钟，给出当堂的练习[⊖]，然后由学生立即试着去解决问题或给出答案。我们承认这对我们来说很难，因为我们喜欢演讲。我们使用的另一种方法是在课堂上不直接讲解例子，而是引出一个例子，让学生以小组的方式来完成下一步，然后分享结果。有时候我们的解决方案更好，有时候学生的更好，而其他的时候我们各自的方案难分高下，各有特点，从而激发了课堂上的讨论。

毫无疑问，这种新的教学方式花费时间而且不能和我们所有的PowerPoint讲义相适应。我们相信虽然我们覆盖的讲义更少，但是启发更多，这个看法和我们的学生在期末考试中的表现是一致的。

⊖ 我们的硕士项目本质上是面向实际应用的，而非面向研究的。大部分学生已经在软件行业具有5～10年的工作经验。SWE 637这门课导致了这本书的诞生，因为我们意识到之前所用的Beizer的经典教科书[Beizer, 1990]已经不再版了。

⊖ 这些当堂的练习还没有成为本书网站的一个正式部分。但我们经常会使用书中的常规练习。有兴趣的读者可以用搜索引擎在我们课程的网页上找到最新的版本。

大部分的课堂练习都是分小组来完成的。我们也鼓励学生合作完成课外的作业。这不仅是因为有证据显示学生一起合作（“同事间学习”）时可以学到更多的东西，而且他们乐在其中，这也符合工业界的实际情况。非常少的软件工程师是独自工作的。

当然，你们可以以自己认为合适的方式来使用此书。我们提供的这些看法只是适合我们的情况。简单总结一下我们现在的讲课哲学：*少讲话，多解惑*。

致谢

我们很高兴在此感谢众多为此书做出贡献的人们，我们会将他们的名字一一列出。首先从乔治梅森大学的学生开始，他们为第 2 版的早期草稿提供了非常棒的反馈。他们是：Firass Almiski, Natalia Anpilova, Khalid Bargqdle, Mathew Fadoul, Mark Feghali, Angelica Garcia, Mahmoud Hammad, Husam Hilal, Carolyn Koerner, Han-Tsung Liu, Charon Lu, Brian Mitchell, Tuan Nguyen, Bill Shelton, Dzung Tran, Dzung Tray, Sam Tryon, Jing Wu, Zhonghua Xi, Chris Yeung。

我们特别感谢已经使用了第 2 版初始章节的同事们，他们提供了很有价值的反馈意见，对最终的完备版本起了极大的作用。他们是：Moataz Ahmed, King Fahd University of Petroleum & Minerals; Jeff Carver, University of Alabama; Richard Carver, George Mason University; Jens Hannemann, Kentucky State University; Jane Hayes, University of Kentucky; Kathleen Keogh, Federation University Australia; Robyn Lutz, Iowa State University; Upson Praphamontripong, George Mason University; Alper Sen, Bogazici University; Marjan Sirjani, Reykjavik University; Mary Lou Soffa, University of Virginia; Katie Stolee, North Carolina State University; Xiaohong Wang, Salisbury University。

其他的一些人对第 1 版提供了绝好的反馈意见：Andy Brooks, Mark Hampton, Jian Zhou, Jeff (Yu) Lei, 以及六位出版社联系的匿名评审。以下各位改正或编写了练习答案：Sana'a Alshdefat, Yasmine Badr, Jim Bowring, Steven Dastvan, Justin Donnelly, Martin Gebert, 顾晶晶, Jane Hayes, Rama Kesavan, Ignacio Martín, Marcel Medina-Mora, Xin Meng, Beth Paredes, Matt Rutherford, Farida Sbry, Aya Salah, Hooman Safaee, Preetham Vemasani, Greg Williams。下面的乔治梅森大学的学生发现并且通常顺带改正了第 1 版中的错误：Arif Al-Mashhadani, Yousuf Ashparie, Parag Bhagwat, Firdu Bati, Andrew Hollingsworth, Gary Kaminski, Rama Kesavan, Steve Kinder, John Krause, Jae Hyuk Kwak, 李楠, Mohita Mathur, Maricel Medina Mora, Upsorn Praphamontripong, Rowland Pitts, Mark Pumphrey, Mark Shapiro, Bill Shelton, David Sracic, Jose Torres, Preetham Vemasani, 汪爽, Lance Witkowski, Leonard S. Woody III, Yanyan Zhu。下面来自别的组织和机构的人发现并通常顺带改正了第 1 版中的错误：Sana'a Alshdefat, Alexandre Bartel, Don Braffitt, Andrew Brooks, Josh Dehlinger, Gordon Fraser, Rob Fredericks, Weiyi Li, Hassan Mirian, Alan Moraes, Miika Nurminen, Thomas Reinbacher, Hooman Fafat Safaee, Hossein Saiedian, Aya Salah, Markku Sakkinen。北京大学的郁莲将第 1 版翻译成简体中文。

我们也想感谢以下已经对第 1 版做出明确贡献进而对第 2 版做出贡献的人们：Aynur Abdurazik, Muhammad Abdulla, Roger Alexander, Lionel Briand, Renee Bryce, George P. Burdell, Guillermo Calderon-Meza, Jyothi Chinman, Yuquin Ding, Blaine Donley, Patrick Emery, Brian Geary, Hassan Gomaa, Mats Grindal, Becky Hartley, Jane Hayes, Mark Hinkle,

Justin Hollingsworth, Hong Huang, Gary Kaminski, John King, Yuelan li, Ling Liu, Xiaojuan Liu, Chris Magrin, Darko Marinov, Robert Nilsson, Andrew J. Offutt, Buzz Pioso, Jyothi Reddy, Arthur Reyes, Raimi Rufai, Bo Sanden, Jeremy Schneider, Bill Shelton, Michael Shin, Frank Shukis, Greg Williams, Quansheng Xiao, Tao Xie, Wuzhi Xu, Linzhen Xue。

当编写第 2 版的时候，乔治梅森大学的研究生助教给了我们关于早期草稿出色的反馈：邓琳，顾晶晶，李楠，Upsorn Praphamontripong。特别强调的是，李楠和邓琳实现了软件覆盖率的工具，而且在本书网站上不断地演化和维护了此工具。

我们还要感谢编辑 Lauren Cowles 提供了坚定不移的支持，并且设定了截止期限来推进本书的项目。我们也要感谢以前的编辑 Heather Bergmann 为这个长期项目所提供的强有力的支持。

最后，如果没有家庭的支持，我们当然无法完成任何的事情。谢谢 Becky、Jian、Steffi、Matt、Joyce 和 Andrew 能让我们的工作和生活平衡。

就像所有的程序都有故障一样，所有的书也都有错误。本书也不例外。同理，就像程序员对软件故障负责一样，我们也对本书的错误负责。特别强调的是，软件测试领域是浩瀚的，其中涉及的技术是复杂的，参考文献注解只反映了我们对测试领域的认知。我们对漏掉的文献表示歉意，同时欢迎指出相关文献的出处。

目　录

Introduction to Software Testing, Second Edition

第一部分

Introduction to Software Testing, Second Edition

软件测试基础

第 1 章

Introduction to Software Testing, Second Edition

为什么测试软件

对于测试者来说，真正需要考虑的是如何设计测试用例，而不是测试本身。

本书的目标就是教授软件工程师如何测试。作为本书的读者，你可能是一名程序员，需要了解如何去写单元测试；你也可能是一名测试工程师，大部分的时候站在用户的角度来检查需求是否满足；你还可能是一名经理，主管程序开发和测试；你还可能是介于这些之间的一个角色，不管你的工作是什么，测试的知识都是非常有用的。我们已经步入 21 世纪的第二个十年，对于整个软件行业而言，软件质量已经成为头等大事；对于所有软件工程师来说，软件测试的知识已经变得不可或缺。

今天，系统软件定义着我们人类文明在使用这些系统时的一些行为。这些系统包括网络路由器、金融计算引擎、交换网络、互联网、电网和交通运输系统，还有关键的通信、命令和控制服务。在过去 20 多年的时间里，软件行业已经发展得更加强大，更具有竞争性，并拥有更多的用户。无论是对一些高新的嵌入式应用（如飞机、飞船和空中交通控制系统），还是对一些平常的设备（如手表、烤炉、汽车、DVD 机、车库开门装置、手机和遥控器）来说，软件都是这些系统的核心部件。现代化的家居拥有数百个处理器，一辆新车可能有一千多个处理器。所有这些处理器都在运行着软件，而有些理想主义的消费者可能会以为它们从来不会出错！虽然很多因素影响着软件工程的可靠性，比如仔细的设计和合理的项目管理，但测试是工业界在开发中用来评估软件的最主要方式。最近敏捷过程的广泛应用给了测试很大的压力，单元测试被给予了非常大的关注度，而测试驱动开发的应用使测试成为功能需求中的关键所在。毫无疑问，工业界已经深深地处在一个变革之中：如何认识测试对于软件产品成功的作用。

幸运的是，对于很多类别的软件应用程序来说，我们用一些基本的软件测试概念就可以设计出测试用例。本书的目的之一就是介绍这些概念，使学生和工程师能够轻易地把它们应用在任何测试环境中。

和其他软件测试书籍相比，本书有一些不同。最重要的区别就是如何看待测试技术。Beizer 在他里程碑式的图书《软件测试技术》里面讲过，测试其实很简单，测试者需要做的就是“找到一个图，然后覆盖它。”感谢 Beizer 的远见卓识，虽然软件测试文献中大量的技术第一眼看上去不太一样，但是我们越发清楚地认识到它们其实有很多相同之处。通常来说，测试技术应用在特定软件工件（artifact）上，比如需求文档或是代码；测试技术也应用在开发周期中的某个特定阶段，比如需求分析或是编程实现。非常遗憾的是，这样的表象掩盖了技术之间的相似性。

本书用了两个有新意但又简单的方法来阐述这些测试技术之间的相似性。首先，我们展示如果使用经典工程学的方法，那么测试会变得更加有效果同时效率也会提高。相对于在软件工件比如源代码或需求上直接设计和生成测试用例，我们的方法是建立抽象模型，在模型上设计抽象的测试用例，然后实现抽象的测试用例使之实例化，同时满足抽象的测试设计。