TURING 图灵程序设计丛书

MANNING

单元测试的艺术

（第2版）

[以] Roy Osherove 著
金迎 译

The Art of Unit Testing

With Examples in C#
Second Edition

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

单元测试的艺术

（第2版）

[以] Roy Osherove 著
金迎 译

The Art
of Unit Testing
With Examples in C#
Second Edition

人民邮电出版社
北 京

图书在版编目（CIP）数据

单元测试的艺术 : 第2版 / (以) 奥西洛夫(Osherove,R.) 著 ; 金迎译. -- 北京 : 人民邮电出版社, 2014.8
(图灵程序设计丛书)
ISBN 978-7-115-36035-9

Ⅰ. ①单… Ⅱ. ①奥… ②金… Ⅲ. ①软件－测试 Ⅳ. ①TP311.5

中国版本图书馆CIP数据核字(2014)第130882号

内 容 提 要

本书是经典的单元测试学习指南，分四部分全面介绍了单元测试技术。第一部分阐述单元测试基本概念，包括如何使用测试框架。第二部分讨论破除依赖的高级技术：模拟对象、存根和隔离框架，包括重构代码以使用这些技术的模式。第三部分介绍测试代码的组织方式、运行测试和重构测试结构的模式，以及编写测试的最佳实践。第四部分介绍如何在组织内实施变革和修改现有代码。

本书适合所有语言的测试和开发人员，特别是测试主管和项目经理。

◆ 著 [以] Roy Osherove
译 金 迎
责任编辑 李松峰 毛倩倩
执行编辑 程 芃
责任印制 焦志炜

◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路11号
邮编 100164 电子邮件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京鑫正大印刷有限公司印刷

◆ 开本：800×1000 1/16
印张：15.25
字数：380千字 2014年8月第1版
印数：1－3 000册 2014年8月北京第1次印刷
著作权合同登记号 图字：01-2014-1136号

定价：59.00元

读者服务热线：(010)51095186转600 印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京崇工商广字第0021号

版权声明

第2版序

那应该是2009年，我当时正在奥斯陆举行的挪威开发者大会上发言。（啊，6月的奥斯陆！）会议地点是一个超大的运动场。会议组织人员把看台分了区，在各区前面搭台，然后用很厚的黑布围起来，形成8个不同的会话“房间”。记得我刚刚结束发言，好像是关于TDD还是SOLID的，要不就是天文学或者别的什么，突然从旁边的“房间”传来了沙哑响亮的歌声以及吉他演奏。

那个分隔“房间”的帘子让我正好可以从旁边偷看到隔壁搞出这些动静的家伙。那家伙就是Roy Osherove。

认识我的人都知道，像这种在软件技术发言中途放声高歌的事情，如果兴之所至我也是干得出来的。因此当我转头面对听众的时候，心里想的是：这个Roy跟我还真是志趣相投，我一定得多了解了解他。

我的确对他有了更多的了解。实际上，他对我的一本书《程序员的职业素养》（*The Clean Coder*）有很大的贡献，还和我一起教了3天的TDD课。我和Roy相处得非常愉快，并且希望还能有更多时间相处。

我相信你在读Roy的这本书时也会是愉快的，因为这本书非常特别。

你读过James Michener写的小说吗？我没读过，但是听说他的小说都以“原子”开篇。你拿在手里的这本书不是James Michener的小说，但也是以“原子”开篇的——单元测试的“原子”。

当你翻阅开头部分的时候，可不要被误导了，这本书可不仅仅介绍单元测试。这本书是从介绍单元测试开始的，如果你有这方面的经验，可以快速浏览开头几章。随着书中内容的展开，后面各章构建在前面内容基础之上，深度不断增加！的确，当我读到最后一章时（当时还不知道那是最后一章），心想下一章就该讨论世界和平了。我的意思是说，你都解决了在使用陈旧遗留系统的顽固组织中引入单元测试的难题，还有什么对付不了的呢？

这是一本技术书——非常技术，书里有很多代码。这是件好事。但是Roy并不局限于技术话题。他讲述职业生涯中的轶事，讨论设计内涵或集成定义的哲学思想，期间还时不时地抄出吉他放声高歌。他好像很乐意给我们讲述在那遥远的、黑暗的2006年发生的故事，那些他做过的很糟糕的事情。

对了，不要担心书里的代码都是C#的。谁又能分得出C#和Java呢，对吧？再说了，这些都无关紧要。他只是用C#作为一个载体表达他的意图，但是书里的道理同样适用于Java、C、Ruby、Python、PHP，或任何其他的编程语言（也许COBOL除外）。

不管你是单元测试和测试驱动开发的新手，还是已经有了丰富经验的人，都能在这本书里找到适合自己的内容。请准备好欣赏Roy给你演唱的“单元测试的艺术”。

还有，Roy，别忘了给吉他调调音！

——Robert C. Martin（BOB大叔）
CLEANCODER.COM

第1版序

当Roy告诉我他在写一本关于单元测试的书时，我非常高兴。测试迷因（meme）在业界已经兴起多年，但是关于单元测试的资料却相对匮乏。我浏览自己的书架，看到了专门讲测试驱动开发的书，也看到了通用的测试理论书，但是迄今为止还没有一本单元测试的全面参考书——没有书介绍这个主题，并且引导读者从第一步开始直到接触被广泛接受的最佳实践。这个事实令人震惊。单元测试并不是什么新实践，我们怎么会落到现在这种地步呢？

说我们处在一个非常年轻的工业领域几乎是老生常谈了，但是确实是这样。数学家们在不到100年前给我们的工作奠定了基石，但是直到最近60年才有了足够快的硬件能够利用他们的认知。在计算机业，理论和实践间有着最初的差距，而我们才刚开始发现这种差距如何影响着我们的领域。

在早期，计算机时间非常昂贵。程序是分批运行的。程序员们有安排好的时间段，他们必须在卡纸上打洞编程，然后送到计算机房。如果程序不对，就浪费了分配给你的时间段。因此你用铅笔和纸对程序进行手工检查，用大脑检验所有的场景和所有的边界情况。我想那时候自动化单元测试几乎是无法想象的。机器本应用于解决应该解决的问题，为什么要用它来进行测试呢？资源的稀缺限制了我们的思维。

后来，机器速度变快了，我们开始醉心于交互式计算。我们可以随意输入和修改代码。桌面检查代码的做法慢慢消失了，我们不再遵循早年间的一些原则。我们知道编程很难，但是这对我们来说只是意味着要在计算机前花费更多的时间，修改代码行和符号，直到找到那个起作用的神奇咒语。

我们从计算能力的稀缺期直接进入了能力过剩时期，错失了中间地带，但是现在要再次获得它。我们开始重新审视计算机作为开发资源的价值，自动化单元测试则把对计算资源的珍惜和桌面检查的原则结合在了一起。我们可以用开发语言编写自动化测试检查工作——不仅一次，而是任意多次。我认为在软件开发中，几乎没有别的实践像自动化单元测试这么强大。

在2009年写下此文时，我高兴地看到Roy的书付印。这是一本实际指南，既帮助你入门，又是你在进行测试任务时的极佳参考。《单元测试的艺术》不是一本关于理想化场景的书，它讲解如何测试已经存在的代码，如何利用常用的框架，最重要的是，它还告诉你如何编写更容易测试的代码。

《单元测试的艺术》很重要，多年前就应该有这么一本书了，但我们那时还没有准备好。现在我们准备好了。请欣赏。

——Michael Feathers

前　言

我工作过的最失败的一个项目就使用了单元测试。或者说，我是这么认为的。那时我带领着一队程序员开发一个记账应用，采取的是彻底的测试驱动开发方式：编写测试，然后编写代码；看到测试失败，使测试通过，重构代码；然后再开始新一轮过程。

项目前期的几个月非常好，所有的事情都很顺利，我们有测试可以证明代码工作正常。但是随着时间的推移，需求发生了变化。我们被迫修改代码以适应新的需求，但是这样一来，测试失败了，需要修复。产品代码还是可以正常工作的，但是我们编写的测试过于脆弱，代码中任何微小的改变都会导致测试失败，哪怕代码工作正常。修改类或方法中的代码成了一项令人生畏的任务，因为同时还需要修复所有相关的单元测试。

更糟糕的是，因为有些人离开了这个项目，没有人知道他们测试的是什么，也不知道如何维护他们的测试，这些测试就无法使用了。我们给单元测试方法起的名字不够清楚，还有的测试互相依赖。最后的结果是，项目才开始不到6个月，我们就扔掉了大部分的测试。

这个项目是个悲剧，我们让自己编写的测试造成的损失比带来的收益多。从长远来看，维护和理解这些测试花费的时间，超过了它们能为我们节省的时间，因此我们停止使用它们了。我此后又参加过别的项目，这些项目中的单元测试做得好一些，我们使用这些测试获得了很大的成功，节省了大量的调试和集成时间。从那第一个失败的项目之后，我一直在整理单元测试的最佳实践，并在之后的项目中应用。在每个工作过的项目中，我总能找到更多的最佳实践。

理解如何编写单元测试，以及如何使它们可维护、可读和可靠，是这本书的内容，不管你使用的是何种语言或集成开发环境（IDE）。这本书涵盖了编写单元测试的基本知识，然后讲解交互测试基础，介绍了在真实世界中编写、管理和维护单元测试的最佳实践。

致　谢

非常感谢Manning出版社的Michael Stephens和Nermina Miller，他们耐心陪伴我度过了本书漫长写作过程中的每一步。还要感谢Manning出版社参与第2版工作以及幕后的每个人。

感谢Jim Newkirk、Michael Feathers和Gerard Meszaros等人，他们为我提供了灵感和思路，使这本书得以呈现在大家面前。还要特别感谢Bob大叔为第2版作序。

在这本书的写作过程中，很多审阅者在不同的阶段阅读了书稿，我要感谢他们提供了宝贵的意见：Aaron Colcord、Alessandro Campeism、Alessandro Gallo、Bill Sorensen、Bruno Sonnino、Camal Cakar、David Madouros、Dr. Frances Buontempo、Dror Helper、Francesco Goggi、Iván Pazmiño、Jason Hales、João Angelo、Kaleb Pederson、Karl Metivier、Martin Skurla、Martyn Fletcher、Paul Stack、Philip Lee、Pradeep Chellappan、Raphael Faria和Tim Sloan。还要感谢Rickard Nilsson在定稿即将付印前进行的技术审校。

最后，感谢参与Manning出版社早期阅读计划（Early Access Program）的读者在论坛上发表评论。他们帮助塑造了这本书。

关于本书

关于如何学习，我听过的最聪明的说法（忘了是谁说的了）是：要真正学会一样东西，就去教授它。写作这本书的第1版，并在2009年出版，对我来说不亚于一次真实的学习体验。我写这本书的初衷是因为厌烦了总是要回答同样的问题，当然也有别的原因。我希望尝试新的东西，希望进行一个实验，知道自己能从写一本书（任何书）中学到什么。我那时觉得，自己擅长单元测试。问题是，你拥有的经验越丰富，就越觉得自己蠢笨。

第1版中的一些内容，我现在认为有些问题，例如：一个单元是指一个方法。这完全不对。一个单元是一个工作单元，正如我在第2版的第1章中讨论的。它可以小到一个方法，也可以大到好几个类（可能是程序集）……还有其他的内容也进行了修改，接下来我会进行介绍。

第 2 版的新内容

在这一版中，我增加了关于受限和不受限隔离框架的资料，还新增了第6章，介绍好的隔离框架具有什么特征，以及像Typemock这样的框架内部是如何工作的。

我现在已经不使用RhinoMocks了。你最好也不要使用RhinoMocks，这个软件已经停止开发和维护了，至少现在是这样。我使用了NSubstitute作为范例介绍隔离框架基础知识，同时也推荐FakeItEasy。我还是不太喜欢Moq，具体理由会在第6章中介绍。

在讨论如何在组织中实施单元测试的章节中，我介绍了更多的可用技术。

我对书中示例代码的设计进行了很多的修改。总的来说，我不再使用属性设置方法（setter），通常使用构造函数注入。我还在书中加入了一些关于SOLID原则的讨论，但只是点到为止，引起你对这个话题的兴趣。

第7章中，关于构建的部分也包含一些新的内容。自上一版之后，我又学到了很多关于构建自动化和模式的知识。

我建议避免使用setup方法，给出了在测试中实现同样功能的替代方法。同时，我使用了版本更高的NUnit，因此对书中一些较新的NUnit API进行了修改。

第10章中，我更新了关于遗留代码的工具信息。

过去三年中，我在使用.NET的同时，也使用Ruby，对于设计和可测试性有了更多的认识，这些内容反映在第11章中。我也对附录中的工具和框架进行了更新，添加了新工具，删除旧工具。

读者对象

这本书写给任何编写代码并且有兴趣学习单元测试最佳实践的人。书中所有的代码都是在Visual Studio中使用C#编写的，因此.NET开发者会觉得这些代码示例特别有用。但是我教授的内容同样也适用于大部分（如果不是全部）面向对象的静态类型语言（例如VB.NET、Java和C++）。如果你是一位架构师、开发者、团队领导者、（编写代码的）QA工程师或编程新手，这本书会非常适合你。

路线图

如果你从未写过一个单元测试，那最好从头至尾阅读这本书，以掌握整体概念。如果你有单元测试经验，那么可以随意挑选感兴趣的章节阅读。这本书分为4部分。

第一部分可以使你从对编写单元测试一无所知进步到及格水平。第1章和第2章涵盖了基础知识，例如：如何使用一个测试框架（NUnit）；基本的自动化测试属性，例如`[Test]`和`[TestCase]`。这两章还介绍了断言、忽略测试、工作单元测试、单元测试的三种结果类型，以及对应的三种测试类型（值测试、基于状态的测试和交互测试）。

第二部分讨论了破除依赖的高级技术：模拟对象、存根和隔离框架，以及重构代码以使用这些技术的模式。第3章介绍存根的概念，展示了如何手工创建和使用存根。第4章介绍如何使用手工模拟对象进行交互测试。第5章将存根和模拟对象结合，展示了隔离框架如何结合两种想法，并实现存根和模拟对象创建的自动化。第6章深入探讨受限的隔离框架和不受限的隔离框架，以及它们的工作原理。

第三部分介绍了测试代码的组织方式、运行测试和重构测试结构的模式，以及编写测试的最佳实践。第7章讨论了测试层次、如何使用测试架构API，以及如何把测试结合在自动化构建过程中。第8章讨论了创建可维护、可读和可靠的单元测试的最佳实践。

第四部分介绍如何在一个组织内实施变革以及如何修改现有代码。第9章讨论了试图将单元测试引入一个组织中时会遇到的问题以及解决方案，还列出和回答了一些你可能被问到的问题。第10章讨论如何给现有的遗留代码增加单元测试。这一章列出了几种方法，帮助你决定从哪里开始测试，还介绍了一些工具，用以对不可测试的代码进行测试。第11章讨论可测试性设计这个极富争议的话题，以及现有的替代方法。

附录列出了一些工具，在测试中可能会对你有所帮助。

代码约定和下载

你可以从GitHub（https://github.com/royosherove/aout2）、本书网站（www.ArtOfUnitTesting.com）或者出版社网站（www.manning.com/TheArtofUnitTestingSecondEdition）下载本书附带的源代码[①]。代码的根目录以及各章的目录中都有一个Readme.txt文件，提供了安装和运行代码的详细信息。

① 也可以免费注册iTuring.cn，至本书页面下载。——编者注

代码清单和文本中的源代码都使用等宽字体，以区别于普通文本。在代码清单中，**粗体**表示这段代码与前一个例子不同，或者在下一个例子中会被修改。在很多代码清单中，代码包含注释，用于指出关键概念；带编号的注释项会在之后的文本中进行解释。

软件要求

要使用这本书中的代码，你需要Visual Studio C# Express（免费）或者更高级的版本（需要花钱购买）。你还需要NUnit（一个免费的开源框架）以及其他相关工具。书中提到的所有工具，要么是免费和开源的，要么有试用版，你可以在阅读本书时免费使用。

作者在线

购买本书的读者都可以免费访问Manning出版社的一个论坛，对本书进行评论，提出技术问题，寻求作者以及其他用户的帮助。要使用这个论坛，请访问：www.manning.com/TheArtofUnitTestingSecondEdition。这个页面提供的信息包括：注册后如何进入论坛、论坛提供何种帮助以及论坛的行为准则。

Manning向读者承诺提供一个场所，供读者之间以及读者和作者之间进行有意义的对话，但并不承诺作者的参与度。作者对论坛的贡献是自愿的（免费的）。我们建议大家尝试提出一些有挑战性的问题，以保证作者的参与兴趣！

只要这本书还在出版，你都可以从出版社的网站访问作者在线论坛，以及以前的讨论归档。

Roy Osherove的其他项目

Roy也是下列书的作者：

- *Beautiful Builds: Growing Readable, Maintainable Automated Build Processes*，参见http://BeautifulBuilds.com；
- *Notes to a Software Team Leader: Growing Self-Organizing Teams*，参见http://TeamLeadSkills.com。

其他资源如下。

- 和这本书相关的团队领导者博客：http://5whys.com。
- Roy的TDD大师班在线视频：http://TddCourse.Osherove.com。
- 关于单元测试的大量免费视频：http://ArtOfUnitTesting.com和http://Osherove.com/Videos。
- Roy一直在世界各地进行培训和咨询工作。你可以通过http://contact.osherove.com联系他，邀请他去你的公司进行培训。
- 你也可以关注他的推特：@RoyOsherove。

关于封面图片

这本书封面上的人物是“Japonais en costume de cérémonie”，即一位身着礼服的日本男性。这幅画选自James Prichard的*Natural History of Man*一书。这是一本1847年英格兰出版的手工着色的版画书。我们的封面设计师在旧金山的一家古董店里发现了这本书。

Prichard于1813年开始了对世界各地原住民的研究。34年后，他的著作出版时，他已经收集了关于不同人种和民族的大量研究资料，他的工作成为了现代民族学的重要基础。Prichard的这本画册收录了不同人种和部落居民身着本土服装的肖像，这些肖像来自多位艺术家，原作大部分是由艺术家直接观察模特创作的。

和我们使用的其他封面插画一样，Prichard收集的这幅版画唤回了两个世纪前服装和部落习俗的丰富性及多样性。从那以后，人们的穿着习惯发生了变化，当时巨大的地区间差异已经逐渐消失。如今，我们常常难以区分一个大陆和另一个大陆的居民，更不用说不同国家或地区的了。如果要乐观看待这个现象，或许我们是用文化和视觉的多样性换取了个人生活的多样性，或者是换取了更为丰富有趣的知识和技术生活。

Prichard的图片集将多年前地区生活丰富的多样性重新带给我们。我们选择基于这些图片的封面，以庆祝计算机业的创造性、主动性以及趣味性。

目　录

第一部分　入门

第二部分　核心技术

第三部分 测试代码

第四部分　设计和流程