软件开发方法学精选系列

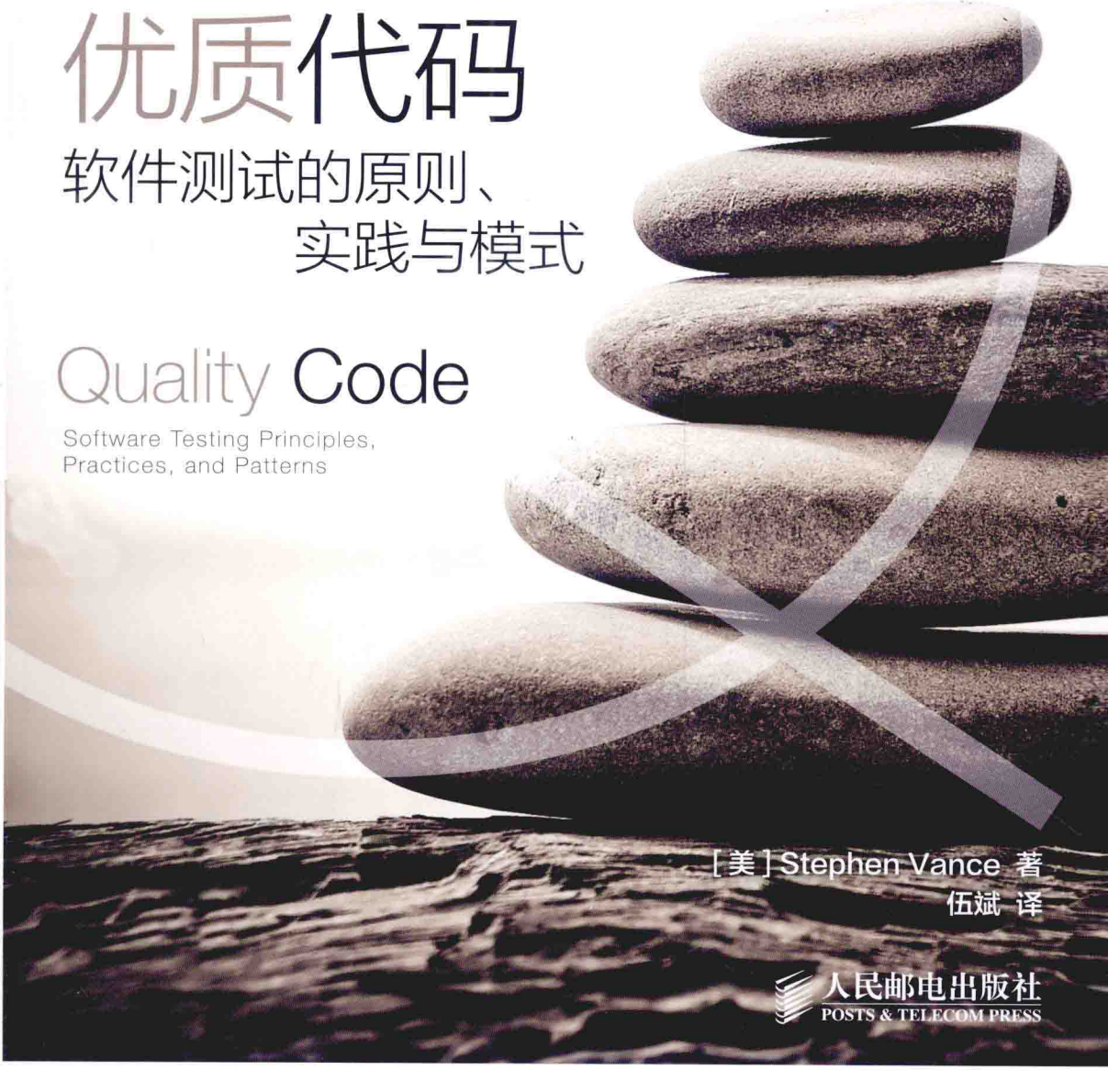
PEARSON

优质代码

软件测试的原则、实践与模式

Quality Code

Software Testing Principles, Practices, and Patterns



[美] Stephen Vance 著

伍斌 译

人民邮电出版社
POSTS & TELECOM PRESS

软件开发方法学精选系列

Quality Code

Software Testing Principles, Practices, and Patterns

优质代码

软件测试的原则、实践与模式

[美] Stephen Vance 著
伍斌 译

人民邮电出版社
北京

图书在版编目（CIP）数据

优质代码 : 软件测试的原则、实践与模式 / (美)万斯 (Vance, S.) 著 ; 伍斌译. — 北京 : 人民邮电出版社, 2015.1
(软件开发方法学精选系列)
书名原文: Quality code:software testing principles, practices, and patterns
ISBN 978-7-115-37558-2

Ⅰ. ①优… Ⅱ. ①万… ②伍… Ⅲ. ①软件—测试 Ⅳ. ①TP311.5

中国版本图书馆CIP数据核字(2014)第273844号

内 容 提 要

本书讲述如何对所有的软件进行轻松的例行测试，书中为读者提供一些工具——一些实现模式，这些工具几乎可以测试任何代码。本书分为三个部分：第一部分讨论了测试的一些原则和实践，包括首次优质、代码意图、测试攻略和测试与设计之间的关系等；第二部分讨论了有关测试实践方面的一些模式，包括测试构造器和 getter/setter、处理字符串、封装与覆写、调整代码可见性、测试单例模式、验证错误条件，以及利用各种接缝和测试多线程等；第三部分展示了两个实例的编程过程，其中一个是用测试驱动开发方法编写新的 Java 应用程序 WebRetriever，另一个是为一个未写测试的 JavaScript 开源项目 jQuery Timepicker Addon 添加测试代码。

本书适合对测试驱动开发有初步了解或实践并想提升测试代码编写技能的程序员和自动化测试工程师阅读，也适合想通过本书在 GitHub 上的微量提交的代码来学习用测试驱动开发方法编写 Java 新项目和用测试来驯服 JavaScript 遗留代码的详细过程的任何读者阅读。

◆ 著 [美] Stephen Vance
译 伍 斌
责任编辑 杨海玲
责任印制 张佳莹 彭志环

◆ 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
北京铭成印刷有限公司印刷

◆ 开本：720×960 1/16
印张：13.5
字数：241 千字 2015 年 1 月第 1 版
印数：1-4 000 册 2015 年 1 月北京第 1 次印刷
著作权合同登记号 图字：01-2014-1729 号

定价：49.00 元

读者服务热线：(010)81055410 印装质量热线：(010)81055316
反盗版热线：(010)81055315

版权声明

译者序

2014年仲夏前的一个下午，当我正在为自己的处女作《驯服烂代码》的收官忙得焦头烂额时，接到了策划编辑杨海玲老师发出的翻译本书的邀请。由于之前翻译过海玲老师策划的《测试驱动数据库开发》，了解译书的辛苦意味着什么。看了本书的页数，我就知道，要翻译它需要我在写书的同时，额外再花3个月每天用五六个小时来翻译。我想时间真的不够用，要是这书不合我的口味，就找个理由谢绝吧。于是就在亚马逊网站上找到了本书的目录和书评。

本书在亚马逊网站的评价是4颗星，一个不错的分数。再看给最高星级的评价说本书中的例子很清晰。嗯，我喜欢讲述清晰的作品。

再看看目录吧。这一看不要紧，本来是要找不翻译本书的理由，结果反倒找到了下面这些对我胃口的关键词：软件匠艺、首次优质、代码意图、单例模式测试、错误条件验证、接缝的使用、多线程测试等。这些关键词成了我要译本书的理由。

回想我自己写的书，在讨论测试技术实现的广度和深度方面与本书相比真是望尘莫及。比如前面提到的单例模式测试、错误条件验证和多线程测试，都是我基本从未考虑过的。由此可见，本书作者在测试驱动开发领域的实践方面功力深厚。

我和本书作者都认同编程是门手艺。而手艺人所最看重的，不是最后精彩绝伦的成果，而是成就这个结果的精雕细琢的过程。因为“授人以鱼，不如授之以渔”，所以我在写自己的《驯服烂代码》时，就刻意通过详述若干个带有GitHub微量代码提交的实例的编写和思考过程，来在编程操练中悟道。可喜的是，本书作者也看到了这一点，在本书最后两章中，提供了两个带有GitHub微量代码提交并能涵盖前面所讨论的主要测试原则的编程实例，来供读者体会作者用测试驱动开发方法开发Java新项目和用测试来驯服已有JavaScript烂代码的详细编写过程。这对读者正是“授之以渔”。这意味着，在阅读本书前面讨论原则和模式的内容时，遇到不懂的地方也不要紧，在重现最后两章的带有详尽提交注释的微量代码提交的实例的编写过程中，就能体悟到前面所讨论的原则，并能从其中悟出更多的“道”。

重点突出的测试实践原则，富有特色的测试实践模式和微量提交的编程实例，令我对本书爱不释手，最终选择翻译本书，并在本书的启发下完成了我自己的拙作《驯服烂代码》。

伍 斌

匠艺程序员，公益编程操练社区“bjdp.org 北京设计模式学习组”创办者

前　言

在过去的几十年里，精益（Lean）生产方式已经革命性地彻底改变了制造业。像全面质量管理（TQM）、准时生产（Just-In-Time）、约束理论（Theory of Constraints）和丰田生产方式（Toyota Production System）这样的框架已经大幅地改善了汽车质量和制造质量的总体状况，并造就了一个充满活力的竞争局面。各种敏捷软件开发方法都将精益生产的原则引入到软件产品开发的知识体系中，但这些原则需要改造，以适应软件开发这个不同于机械制造的领域。

像那些为了提升客户满意度和减少产品整个生命周期的维护成本，而将质量内建到产品中的想法，已经造就了测试驱动开发（TDD）和其他测试先行（test-first）及尽早测试（test-early）的方法的诞生。无论遵循哪种方法，都需要了解可测试的软件看起来是什么样子的，并掌握广泛的技术来成功地实现测试。笔者已经发现，在各种测试的调理方案中，上述原则和实践现状之间所存在的差距，往往是一个未被引起重视的失败根源。“使用测试驱动开发”这句话说起来容易，但是当面对一个项目时，许多开发人员都不知从何下手。

当向人们展示如何运用测试驱动开发或者尽早测试开发时，经常发现那些，其中的一个障碍就是编写测试的技术性细节。如果以数学函数的方式来执行一个方法，单纯地将输入转换成所期望的输出，那是没有问题的。但是很多软件都有副作用、行为特征或上下文约束，这些都使得对其进行测试并不是那么容易的。

本书源自下面这样经常性的需求，即要为开发人员展示如何针对那些曾经让他们挠头的具体场景进行测试。当这种需求来临时，我们总是会坐下来，为那些惹麻烦的代码编写一个单元测试，使开发人员能够拥有一个新的工具。

本书内容

本书讲述了如何对所有的软件进行轻松的例行测试。主要侧重于单元测试，但是其中的许多技术也同样适用于较高层次的测试（higher-level test）。本书将为读者提供一些工具——一些实现模式（implementation pattern），这些工具几乎可以测试任何代码，用它们还能够识别出代码什么时候需要改成可测试的。

本书目标

本书将帮助读者：

- 了解如何对所有的代码进行轻松的单元测试；
- 提高软件设计的可测试性；
- 针对代码找到适用的测试替代方案；
- 以和应用程序的生长相适应的方式来编写测试。

为了达到这些目标，本书将提供：

- 20 多个测试代码的具体技术和许多例子；
- 在各种场景下使用正确的测试技术的指南；
- 一种能帮助读者更加专注地进行单元测试的方法。

本书读者

本书主要针对那些希望提升自己在代码层面的测试技能以增强代码质量的、专业的软件开发人员和在测试领域的软件开发人员。本书尤其对那些测试驱动开发（TDD）和尽早测试的实践者们特别有用，可以帮助他们从一开始就确保其代码的正确性。本书中的许多技术也同样适用于集成测试和系统测试。

读者背景

本书假定读者具有以下特点。

- 熟练掌握面向对象的编程语言，能够阅读和理解 Java、C++和其他语言的样例代码，并能将从中学到的知识运用到自己所运用的编程语言中。
- 熟悉代码层面测试的概念和像 JUnit 这样的 xUnit 测试框架的工具。
- 了解或可以查阅到有关设计模式和重构的信息。

本书章节

第一部分（第 1～5 章）涵盖了能成功指导测试的原则和实践。第 1 章将本书所讨论的方法放到工程的背景下，讨论了工程、匠艺（craftsmanship）和首次优质[①]

① 首次优质，是指第一次就把产品质量做好的理念，以消除后来检查产品质量缺陷的成本和时间。——译者注

（first-time quality）的一般概念，以及一些针对软件的具体问题。第 2 章探讨了意图（intent）的作用。第 3 章描述了一种能帮助读者更加专注地工作的测试方法。第 4 章讨论了设计和可测试性之间的相互作用，其中包括了一些能够帮助拓展测试工作的想法。第 5 章介绍了一些能够用来指导做出测试决策的测试原则。

第二部分（第 6～13 章）详细介绍了测试的实现模式。首先，第 6 章介绍了 bootstrapping 测试，并讨论了相关技术的基本类别，而第 7～12 章则对第 6 章介绍的内容进行了详细的阐述。同时在第 9 章对意图这个概念进行了更加深入的研究。第 13 章则通过引入一些能确定性地重现竞态条件[①]（race condition）的技术，在技术上更深入地探究了许多人认为不可能办到的事情。

第三部分（第 14～15 章）详细描述了如何把本书前面所讨论的原则和技术，运用到两个真实工作中的例子上。第 14 章探讨了使用测试驱动开发来从头创建一个 Java 应用，展示了如何开始以及如何将上述技术运用到一个严格定义类型的语言上。第 15 章选择了一个未写测试代码的开源 JavaScript jQuery 插件，来为其添加测试代码，展示了驯服用动态语言编写的遗留代码的方法。在描述过程中，这两个例子都包含了详细的 GitHub 的代码提交历史信息，以供参考。

同道前辈

人类所有的进步都是建立在他人先前努力的基础之上的。本书所讨论的内容正是在过去 15 年计算机发展的影响下成长起来的。下面的列表并不详尽，希望列表中有遗漏或宣传得不够到位的地方不会冒犯这些前辈。

- 敏捷宣言（Agile Manifesto）的推动者们和签署者们。
- 早期敏捷开发方法的先驱们，如极限编程（eXtreme Programming）的先驱 Kent Beck。
- 提出设计模式的“四巨头”[②]（Gang of Four）和提出重构的 Martin Fowler。
- 软件匠艺运动和人称“Bob 大叔”的 Robert C. Martin。

① 竞态条件，指一种电子或软件系统的行为，该行为的输出依赖于其他不可控事件的发生顺序或发生时间（引自 wikipedia.org）。——译者注

② 四巨头，指《设计模式》（Design Patterns）一书的四位合著者 Erich Gamma、Richard Helm、Ralph Johnson 和 John Vlissides。——译者注

- 最近 Michael Feathers 和 Gerard Meszaros 在软件测试领域所做出的开创性的工作。

很有幸曾与上述杰出前辈的一些原团队的同事们一起共事过。

致　谢

所有的作者都会说，如果没有生活伴侣的支持，他们是不可能把书写完的。对于这一点，直到自己亲身经历写书的过程后，才真正地体会到其中的滋味。如果没有妻子珍妮的持续支持和鼓励，没有她对写书所费时间的耐心，没有她每日所说的“我来刷盘子，你去写书吧。”这样的话语，真的没法完成这本书的写作。

感谢 Greg Doench 对于头次写书的作者所表现出的耐心，以及他在本书的编辑过程中所提供的经验丰富的指导。Zee Spencer、Eric E. Meyer 和 Jerry Eshbaugh 针对本书的审阅反馈帮助改进了本书的内容，并使其更加专注。希望本书没有辜负他们所提供的反馈。

开源软件项目 jQuery Timepicker Addon 插件是本书第 15 章中使用的样例的主体。该插件作者 Trent Richardson 一直都对把这个开源项目的代码纳入测试的保护之下这件事抱有极大的热心和支持。他接受了迄今为止的所有 pull requests[①]。在撰写本书之时，该项目的第一个带有测试的版本上线了。

多年来，笔者曾辅导和管理过多个团队。一个人只有在去教会别人的过程中，才能让自己真正明白一些东西。感谢这些团队让笔者得以成长。

有了《Dr. Dobbs' Journal》和《Software Development》杂志的多位作者的激发，笔者才产生了有关尽早测试方法一些早期的想法。Eric E. Meyer、Jean-Pierre Lejacq 和 Ken Faw 展示了有关 TDD 的一些训练有素的方法。笔者的机会，许多都是 Christopher Beale 提供的，或者和他一起创造的。我们的事业交织在一起，这包括我们和 Joe Dionise 在一起所做的工作。许多早期的设计和架构经验，都是在他们的指导下形成的。

最后，两位教授在不知不觉中对笔者产生了很大的影响，这种影响在本书中达到

① pull request 指的是程序员在 GitHub 修改代码后所发出的用来通知其他程序员有关该代码变动的请求。GitHub 是一种基于 Web 的使用代码版本控制系统 Git 的软件开发项目托管服务，用来管理源代码的版本变更。——译者注

了极致。20 世纪 90 年代初，曾在密歇根大学任教的 Lee 教授，向笔者展示了计算机科学可以不仅仅是一种爱好。在硕士课程的第一堂课中，奥克兰大学的 Janusz Laski 教授介绍了形式化验证（formal verification）和静态与动态分析方法，对理解和宣传那些支持软件开发过程的工具很有帮助。

目　录

第一部分　测试的原则和实践

第二部分 测试与可测试性模式

第三部分 实例

第一部分　测试的原则和实践

测试，尤其是用代码编写的自动化测试，贯穿软件工程的始终。无论是通过测试驱动开发、持续集成、行为驱动开发、持续交付、验收测试驱动开发、实例化需求（specification by example）、集成测试、系统测试，还是单元测试，每一个参与基于软件的产品开发的人，都会有机会编写自动化测试。敏捷、精益（lean）[①]和软件匠艺运动都赞成使用高水平的测试，来支持快速开发和高品质的产品。

软件工程领域的思想领袖们，把基于代码的测试作为专业开发人员的保留节目来加以推动。而人称“Bob 大叔的”Martin[CC08，CC11]则把它称为软件匠艺。Gerard Meszaros[xTP]整合了与其相关的词汇。Steve Freeman 和 Nat Pryce[GOOS]描述了如何使用健康的测试剂量来培育软件产品的开发。Michael Feathers[WEwLC]为大家展示了如何通过测试来拯救旧代码，他甚至把遗留代码（legacy code）定义为没有对其编写测试的代码。Kent Beck[XPE]等人阐释了如何通过使用包括测试驱动开发方法在内的手段，来将编程发挥到极限。

上述杰出人物中的每一位，都描述了在软件开发中使用测试的一个重要部分。他们所撰写的每一本技术书籍，都使用了大量的示例来帮助大家领会他们各具特色的教导。

然而，当笔者用测试的调理方案辅导了多个团队之后，却一次次地发现阻碍采用测试的障碍，既不是对测试流程缺乏了解，也不是对相关概念的误解，或者词汇的缺失，更不是对测试实践的价值的怀疑。虽然所有这些障碍都因时因人而异，但是一个最常见的、一直没有很好地加以面对的障碍，就是没有充分地理解测试代码的机制和编写可测试代码的机制。

虽然本书专注于有关编程测试和可测试性方面的多种机制——各种实现模式，但

① 本书所讨论的精益是指从精益制造（Lean Manufacturing）运动派生而来的精益生产（Lean Production）方式，这有别于精益创业（Lean Startup）。但如果精益创业就是公司的经营方针的话，当从精益创业的试验中体悟到其中的运作之道后，就会需要本章所讨论的这些适用于可持续产品化的软件的质量观点。