开发人员专业技术丛书

HZ BOOKS 华章科技

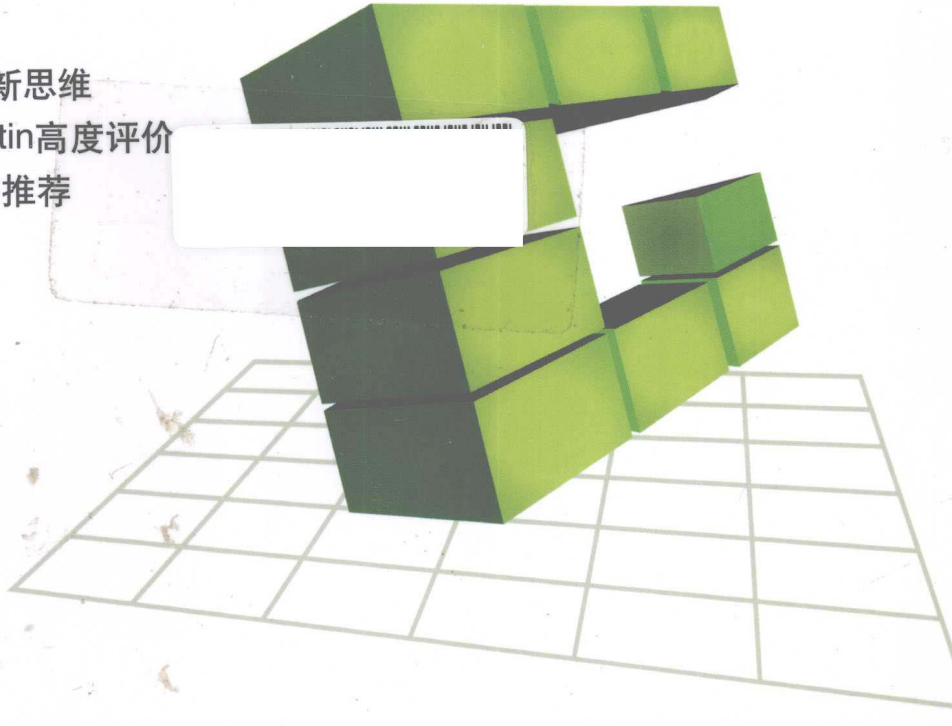
PEARSON

测试驱动的面向对象软件开发

Growing Object-Oriented Software, Guided by Tests

（美） Steve Freeman
Nat Pryce 著 王海鹏 等译

- 旧代码带来的新思维
- Robert C. Martin高度评价
- Kent Berk作序推荐



机械工业出版社
China Machine Press

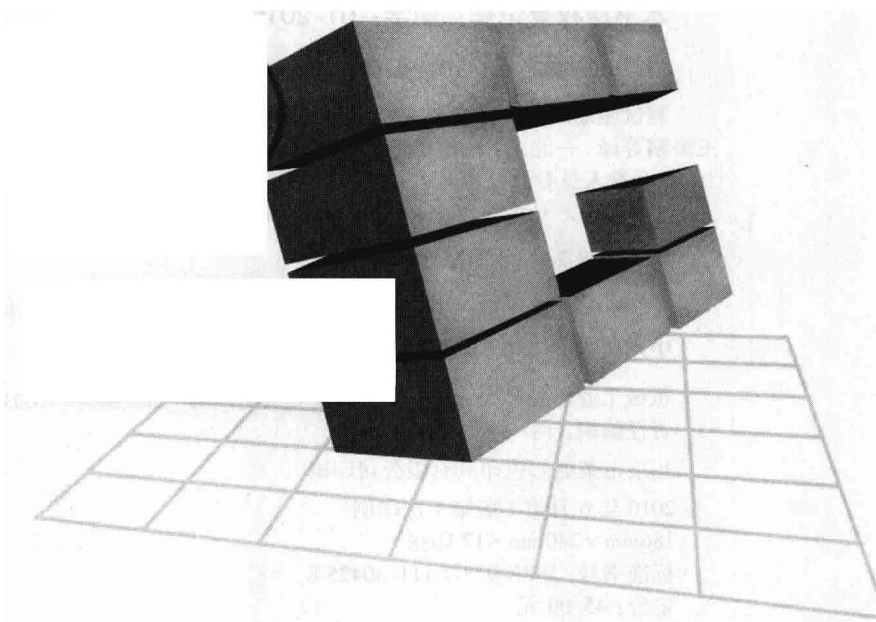
PEARSON Education

开发人员专业技术丛书

测试驱动的面向对象软件开发

Growing Object-Oriented Software, Guided by Tests

（美） Steve Freeman
Nat Pryce 著 王海鹏 等译



机械工业出版社
China Machine Press

本书采用通俗易懂的比喻，众所周知的编程语言，短小精悍的工作实例，深入浅出的分析处理——仿佛在和几位世界级的编程高手一边喝茶，一边聊天，循序渐进地让读者在不知不觉中进入编程的最高境界。即使是刚刚入门的初学者，也会从中找到读书的乐趣，因为可以从一开始就找到开启面向对象开发大门的钥匙；随着经验的积累，编程水平的提高，再来看这本书，用不同的视角重新审视程序，又会体会到更深层的编程哲学。

本书是编程爱好者的启蒙指南，更是系统分析人员、测试人员、程序设计人员、软件开发人员以及面向对象程序研究人员等专业人士革新编程思想的必备手册。

本书版权登记号：图字：01-2010-1665

图书在版编目(CIP)数据

测试驱动的面向对象软件开发/(美)弗里曼(Freeman,S.),(美)普雷斯(Pryce,N.)著;王海鹏等译.—北京：机械工业出版社，2010.5

(开发人员专业技术丛书)

书名原文：Growing Object-Oriented Software, Guided by Tests

ISBN 978-7-111-30425-8

Ⅰ.测…　Ⅱ.①弗…②普…③王…　Ⅲ.面向对象语言－软件开发　Ⅳ.TP311.52

中国版本图书馆 CIP 数据核字（2010）第 069903 号

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：白　宇

北京市荣盛彩色印刷有限公司印刷

2010 年 6 月第 1 版第 1 次印刷

186mm×240mm · 17 印张

标准书号：ISBN 978-7-111-30425-8

定价：45.00 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

购书热线：(010) 68326294；88379649；68995259

投稿热线：(010) 88379604

读者信箱：hzjsj@hzbook.com

对本书的赞誉

“本书的作者领导了编程手艺的一场革命，他们控制了软件培育的环境。他们的培养皿是模拟对象，他们的显微镜是单元测试。本书可以告诉您这些工具如何在工作中引入可重复性，让所有科学家都羡慕。”

——Ward Cunningham

“终于有一本书用丰富的代码揭示 TDD 和 OOD 之间的共生关系。作者是测试驱动开发的先驱，在本书中介绍了原则、实践、探索法，以及他们数十年职业经验中的（最好的）轶事。每个软件手艺人都希望钻研这些工作实例的章节，研究这些高级的测试和设计原则。本书值得保存。”

——Robert C. Martin

“人们常常深入地讨论设计，但不提经验。人们常常强调测试，但只是局限于狭隘的品质定义，关注是否存在缺陷。这些方面都有价值，但每一方面都只是一个巴掌。Steve 和 Nat 将两个巴掌放在了一起，这值得鼓掌（也是我最诚恳的评价）。他们以清晰、理智和幽默的方式，说明了一种对设计、测试、代码、对象、实践和过程的观点，这种观点是有说服力的、可以实践的，充满了深刻的见解。”

——Kevlin Henney，《Pattern-Oriented Software Architecture》和《97 Things Every Programmer Should Know》的合著者

“Steve 和 Nat 写了一本好书，与其他人分享了他们的软件手艺。这本书不只应该读，还应该研究。花足够的时间和精力来研究本书的人，将获得超级开发技能。”

——David Vydra，发行人，testdriven. com

“本书提出了测试驱动开发的一种独特见解。它描述了 TDD 的另一种成熟形式——这种形式 21 世纪初在伦敦迅速发展，其特点是完全关注用户场景，突出强调对象间的消息机制。如果您想成为当前 TDD 领域的一名专家，就需要理解本书中的思想。”

——Michael Feathers

“在本书中，您会学到节奏、思考的细微差异和有效的编程实践，目的是培育经过测试的、设计良好的、大师级的面向对象应用程序。”

——Rebecca Wirfs-Brock

译 者 序

这是一本关于利用模拟对象进行测试的书。类因职责而存在，对象根据契约与它的协作者进行交互。用模拟对象进行测试的要点就是检查对象之间的契约。这些契约要明确无误，没有二义性，所以用模拟对象进行测试对类的设计有着很高的要求。

这是一本关于测试与面向对象设计如何相互影响的书。易测试的设计是好设计，难测试的设计是不好的设计，测试成本高的设计是昂贵的设计，无法测试的设计是不可行的设计。当我们用心聆听测试代码的反馈，它就会告诉你产品代码的功能正确性、运行效率、可靠性、API 的易用性、可维护性、可移植性和可伸缩性。

这是一本关于增量式软件开发的书。我们在写程序的过程中学习，而不是学习如何写程序。软件开发者与领域专家的合作是一个学习的过程。软件一开始可能只有一个特征、一些模糊的概念，然后逐渐发展出许许多多更清晰的特征和概念，并且在使用过程中会不断发展。在我看来，增量式的开发是唯一的软件开发方式。

这本书的目的在于告诉我们如何进行增量式的软件开发。爱因斯坦说，例子不是一种教学方式，而是唯一的教学方式。书中介绍的拍卖狙击者的例子，提供了软件开发者所需要注意的细节，一个特征接一个特征地教读者利用面向对象的思想，增量式地开发灵活的、面向对象的程序。

过程为我们带来乐趣，成就只是副产品。开发的过程是否有趣？是否像打怪升级的游戏一样？每一个已实现的特征就是被你杀死的怪，你不希望被杀死的怪经常复活，向你寻仇。不好的程序是地狱，因为那里面经常复活的怪太多。这本书教我们写清晰的特征列表，然后用验收测试来确保这些特征的实现。

本书给我最大的感悟是：如果软件是孩子，程序员就是家长，需要用心血去培育。如果软件是诗歌，程序员就是炼字师，可以用编程语言来准确反映自己的思想。值得去写的程序，就值得写好。我们应该做得更好，也可以做得更好。

“知之不如好之，好之不如乐之。”对于软件开发，作者有一种乐此不疲的态度。编程是一门艺术，在追求艺术的道路上，没有止境。

本书改变了我对代码（尤其是对测试代码）的看法。像所有好书一样，它可以改变人们对事物的看法。开卷有益，因此，我郑重地向大家推荐它。

除封面署名外，参加本书翻译工作的人员还有：王海燕、李国安、周建鸣、范俊、张海洲、谢伟奇、林冀、钱立强、甘莉萍。

王海鹏

庚寅年初春于上海

序

随着软件发布周期越来越短，一个困境出现了：如何在更少的时间里发布更多的软件，并持续地发布下去？我们需要新的观点来解决这个困境。只有技术的转变是不够的。

本书展示了这样一个新观点——假如软件不是“做”出来的，不像我们做纸飞机那样，叠好后就飞出去，情况会怎样呢？假如我们像对待有价值、有产出的农作物那样（培育、剪草、收割、施肥和灌溉）对待软件，情况会怎样呢？几十年或几个世纪以来，传统的农夫知道如何让农作物高产。如果我们以同样的方式对待程序，软件开发会有什么不同呢？

我印象最深的是，本书同时提供了这种观点转变的哲学思想和实现机制。本书的作者是善于写代码的实践者，他们也善于教别人写代码。您从中可以学到如何保持高产，以及如何重新审视您的程序。

这里展示的测试驱动开发的方式不同于我所使用的方式。虽然我还不能清晰表达其中的区别，但我已从作者清楚、自信的技巧介绍中受益。方言的多样化已经为我进一步优化开发提供了思考的新源泉。本书展示了一个有条理的、一致的开发体系，其中的不同技巧相互支持。

我建议您阅读本书，通过书中的例子，了解作者如何思考编程，如何实践编程。这种体验将丰富您的软件开发方式，有助于您编程，而且同样重要的是，有助于您以不同的观点来看待您的程序。

Kent Beck

前　言

这本书是讲什么的

这是一本实践指南，介绍了我们发现的编写面向对象软件的最好方式：测试驱动开发（test-driven development，TDD）。它描述了我们遵循的过程、追求的设计原则，以及使用的工具。它以我们数十年的经验为基础。在这些年里，我们与世界上一些最好的程序员共事，向他们学习。

本书讨论了一些问题与困惑，它们是我们在一个个项目中所见到的。如何将测试驱动开发应用到软件项目中？我从哪里开始？为什么我应该既编写单元测试，又编写用户场景测试？测试“驱动”开发是什么意思？如何测试某个难弄的特征？

本书同时也着重讨论了设计，以及设计方式对 TDD 方式所产生的影响。如果说我们从中学到了什么，那就是测试驱动开发作为一个整体时，效果最好。我们曾看到过一些团队虽然采用了一些基本实践（编写并运行测试），但他们仍不得要领，因为他们并没有采用背后更深层次的过程。

为什么“培育”面向对象软件

本书使用“培育”（Growing）这个词是因为它让人们感觉到开发是增量式的。我们在任何时候都有可以工作的软件，确保代码总是尽可能地具备良好的结构，并经过全面的测试。任何事情都不比交付可以工作的系统更有效。正如 John Gall 在［Gall03］中所写的，“能工作的复杂系统总是从能工作的简单系统演进而来的。”

“培育”也暗示了一个在好的软件中发现的生物学特点，即在每一层结构中一致的感觉。它将我们的方式与面向对象结合起来，正好与 Alan Kay[⊖]的观点一致，即对象像生物细胞一样彼此发送消息。

为什么要以测试为“指导”

我们先写测试，因为我们发现这有助于写出更好的代码。先写一个测试迫使我们澄清自己的意图，只有无二义地描述了应该做什么以后，我们才会开始下一步的工作。先写测试的过程帮助我们发现设计是否太僵硬或没有关注要点。然后，当我们希望继续下一步工作并修复设计缺陷时，测试可以提供回归覆盖测试构成的安全网。

本书使用“指导”（Guided）这个词，是因为此项技巧仍然需要使用者的技能和经验。我们发

⊖ Alan Kay 是 Smalltalk 的创始人之一，发明了“面向对象”这一术语。

现测试驱动开发是一种有效的设计支持工具——只要我们学会如何增量式地开发并“聆听测试”。像其他正规的设计活动一样，TDD 需要理解和持续的努力才有效。

我们曾发现一些团队同时编写测试和代码（甚至有一些团队先写测试），他们的代码仍然一团糟，测试只是增加了维护的成本。他们已经起步，但还没有明白这种技巧是要让测试来指导开发。利用测试的内容，将关注点放在取得进展上，并利用测试的反馈来提升系统的品质。

什么是模拟对象

编写本书最初的动机是完整解释使用模拟对象（Mock Object）的技术[⊖]，我们看到这种技术常被人误解。随着写作的深入才意识到，我们社区对模拟对象的发现和使用实际上体现了我们写软件的方式。这是大局观的一部分。

本书将利用 jMock 库来展示模拟对象是如何工作的。更具体地说，我们将展示模拟对象应该用于 TDD 过程的什么位置，以及它在面向对象开发的环境中具有怎样的意义。

这本书为谁而写

这本书是为“有一定知识的读者”而写的。它的目标读者是具有专业经验的开发者，他们至少曾看过测试驱动开发。在编写时，我们假设是向一些以前未接触过这些技术的同事进行解释。

为了留出篇幅以介绍更深入的内容，我们假定读者具备有关基本概念和工具的一些知识。别的一些书对 TDD 进行了很好的介绍。

这是一本 Java 书吗

本书从头到尾使用 Java 语言，因为它非常普及，我们希望读者至少能理解这些例子。也就是说，本书实际上是在介绍任何面向对象环境都适用的一组技巧。

如果您目前不用 Java，在许多其他语言中也有与我们使用的测试和模拟库（JUnit 和 jMock）等价的类库。这些语言包括 C#、Ruby、Python、Smalltalk、Objective-C，以及（令人印象深刻的）C ++。甚至还有更少见的语言的版本，如 Scala。在 Java 中，也有其他的测试框架和模拟框架。

为什么您要听信我们

本书汇集了我们几十年的经验，包括近十年的测试驱动开发。在这段时间里，我们在各种项目中使用 TDD。这些项目包括：面向消息的大型企业集成系统（具有交互式 Web 前端和多处理器计算网格后端）、微型嵌入式系统（必须运行在几十 KB 的内存中）、用作关键业务系统广告的免费游戏、后端中间件和网络服务（支持高度交互的图形桌面应用）。而且，我们为世界各地的会议和公司讲授这些内容。

我们也从同事的经验中获益，他们来自伦敦的 TDD 社区。我们花了不少工作时间和业余时间让思想接受挑战和磨炼。我们很感谢能有机会与这样活跃的（善于争论的）同事一起工作。

⊖ 模拟对象是一些替代的实现，目的是测试对象如何与相关的对象进行交互。

本书有些什么内容

这本书有五个部分：

第一部分“简介”，是在软件开发项目背景下，对测试驱动开发、模拟对象和面向对象设计的高层次的介绍。同时也介绍了本书其他部分中用到的一些测试框架。即使您已经熟悉 TDD，仍然建议您通读第 1 章和第 2 章，因为它们描述我们进行软件开发的方法。如果您熟悉 JUnit 和 jMock，您也许愿意跳过简介的其他部分。

第二部分“测试驱动开发过程”，描述了 TDD 过程，展示了如何开始开发，并让开发进行下去。我们深入探讨了测试驱动开发和面向对象编程之间的关系，说明了这两种技术的原理是如何相互支持的。最后，我们讨论了如何处理外部代码。这一部分介绍了概念，下一部分将这些概念投入实战。

第三部分“工作的例子”，是一个扩展的例子，让您初步体验一下以测试驱动的方式来开发面向对象应用。在整个过程中，我们讨论了一些折中和做决定的动机。这个例子相当长，因为我们希望说明 TDD 的某些特征是怎样随着代码规模的扩大变得更为重要的。

第四部分“可持续的测试驱动开发”，描述了一些保持系统可维护的实践。我们现在非常注意保持代码整洁和富有表现力，因为这些年来，我们已经了解了代码变差的代价。这部分描述我们采用的一些实践，并解释了为什么这么做。

第五部分“高级主题”，探讨了 TDD 更难的一些方面：复杂的测试数据、持久性和并发性。我们展示了处理这些问题的方法，并讨论了这对代码的设计和测试所产生的影响。

最后，附录包含了关于 jMock 和 Hamcrest 的一些支持材料。

本书不包含什么内容

这是一本技术书籍。我们不讨论使项目成功的其他主题，如团队组织、需求管理和产品设计。采用增量式的、测试驱动的方法来开发显然与项目运作的方式有着密切的关系。TDD 支持一些新的活动，例如频繁交付；它也会被一些组织环境破坏，例如早期设计冻结或利益相关人之间缺乏沟通。同样，有许多其他书籍讨论这些主题。

作者简介

Steve Freeman

Steve Freeman 是一名独立咨询师，擅长领域是敏捷软件开发（http://www.m3p.co.uk）。他与 Nat Pryce 一同赢得了 2006 年敏捷联盟的 Gordon Pask 奖。他是伦敦极限星期二俱乐部（London Extreme Tuesday Club）的创建成员，也是第一任伦敦 XP 日（London XP Day）的主席，还经常在国际会议上担任组织者和演讲者。Steve 曾在各种类型的组织中工作过，从为 IBM 开发完整零售版软件，到为大的研究实验室开发原型。Steve 拥有剑桥大学的哲学博士学位，并拥有统计和音乐学位。Steve 居住在英国伦敦。

Nat Pryce

在帝国理工学院取得博士学位之后，Nat Pryce 加入了 .com 风潮，刚好看到泡沫破灭。从那时起，他就在许多不同的行业（包括体育新闻报道、市场营销传播、零售、电信和金融业）中当程序员、架构师、培训师和咨询师。他也曾参加学术研究项目，偶尔在大学任教。作为 XP 的早期采用者，他编写了一些支持 TDD 的开放源码库，对一些库作出了贡献，他也是伦敦 XP 日（London XP Day）的发起组织者之一。他也经常在国际会议上发表演讲。Nat 居住在英国伦敦。

致　谢

作者要感谢在本书编写过程中每一个提供了支持和反馈的人。Kent Beck 和 Greg Doench 在第一时间将这本书的编写工作委托给我们，Dmitry Kirsanov 和 Alina Kirsanova（以极大的耐心）对本书进行修润并使其付梓。

许多人曾帮助过我们，和我们交流在阅读和评审草稿时遇到的困难，或提供支持和鼓励，他们是：Romilly Cocking、Jamie Dobson、Michael Feathers、Martin Fowler、Naresh Jain、Pete Keller、Tim Mackinnon、Duncan McGregor、Ivan Moore、Farshad Nayeri、Isaiah Perumalla、David Peterson、Nick Pomfret、J. B. Rainsberger、James Richardson、Lauren Schmitt、Douglas Squirrel、The Silicon Valley Patterns Group、Vladimir Trofimov、Daniel Wellman 和 Matt Wynne 。

谢谢 Dave Denton、Jonathan “Buck” Rogers 和 Jim Kuo 的建模工作。

如果没有极限星期二俱乐部（XTC），这本书和书中描述的技巧都不会存在。XTC 是伦敦定期的非正式聚会，让对敏捷、极限编程和测试驱动开发有兴趣的人能聚在一起。非常感谢所有和我们分享经验、技巧、教训与解决办法的人。

目　录

第三部分 工作的例子

第五部分　高级主题

第一部分

简　介

测试驱动开发（TDD）是一个貌似简单的想法：在写代码之前先为代码写测试。这里用“貌似简单”，是因为它把测试所扮演的角色转移到了开发过程中，对业界关于测试目的的假定提出了挑战。测试不再是为用户消除缺陷，而是帮助团队理解用户需要的特征，并可靠地、可预测地交付这些特征。如果遵循了它的结论，TDD 将彻底改变软件开发的方式。根据我们的经验，它将极大地改变我们建造的系统的品质，特别是系统的可靠性和响应新需求时的灵活性。

测试驱动开发广泛地应用于“敏捷”软件开发方法之中。它是极限编程（XP）［Beck99］的核心实践，是 Crystal Clear［Cockburn04］的推荐实践，经常在 Scrum 项目［Schwaber01］中使用。我们在所有的敏捷项目中使用 TDD，并在非敏捷项目中也见到有人使用 TDD。我们甚至发现它有助于在纯研究项目中取得进展，这种项目的动机是探索思想而不是交付特征。