HZ BOOKS
华章科技
PEARSON

4位世界顶级软件开发专家、敏捷导师兼Jolt大奖获得者数十年工作经验结晶，敏捷软件开发领域公认的经典著作

围绕意图导向编程、分离构造和使用、测试先行、Shalloway原则、面向接口设计、测试驱动开发、避免过度设计、持续集成、共性和可变性分析、重构等核心技术主题给出了大量最佳实践，字字珠玑

华章程序员书库

Essential Skills for the Agile Developer
A Guide to Better Programming and Design

敏捷技能修炼

敏捷软件开发与设计的最佳实践

（美） Alan Shalloway　Scott Bain　Ken Pugh　Amir Kolsky　著
郑立 邹骏 黄灵　译

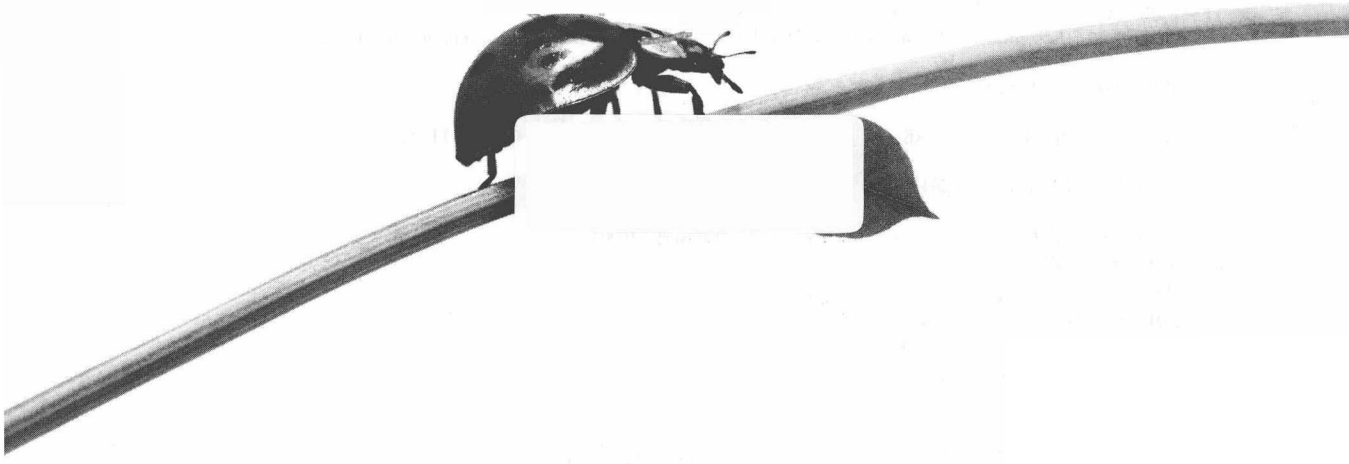
机械工业出版社
China Machine Press

华章程序员书库

Essential Skills for the Agile Developer
A Guide to Better Programming and Design

敏捷技能修炼
敏捷软件开发与设计的最佳实践

（美） Alan Shalloway　Scott Bain　Ken Pugh　Amir Kolsky　著
郑立　邹骏　黄灵　译



机械工业出版社
China Machine Press

本书的4位作者都是世界顶级的软件开发专家和敏捷导师，都有数十年的软件行业从业经验，其中3位曾荣获Jolt大奖。本书是敏捷软件开发领域公认的经典著作，权威性毋庸置疑。

书中内容围绕“敏捷式编程”这一主题展开，对每一位敏捷软件开发人员都应该掌握的核心技能和技术进行了深入阐述，总结出了大量最佳实践，提供了一整套最精炼的技术集合，可以帮助他们在开发中变得游刃有余，极大地提高开发效率和软件质量。

全书共分四个部分：第一部分（1～7章），阐述了在软件开发过程中能起到“四两拨千斤”作用的几种思想方法（“小舵板”），如意图导向编程、分离构造和使用、测试先行和Shalloway原则等，并总结了业界常用的几种实践，包括如何封装、面向接口的设计和验收测试驱动等；第二部分（8～9章），对过度设计和持续集成这两个问题进行了深入的探讨，并给出了最佳实践；第三部分（10～13章），作者分享了很多只有在他们的教学现场才能获得的经验，这些经验是优秀架构师应该具备的，具体包括共性和可变性分析、以开放关闭原则为目标的重构、需求与功能接口、何时以及如何使用继承等重要内容；第四部分是附录，介绍了统一建模语言、提高代码质量的原则，以及如何封装原始数据类型等。

本书版权登记号：图字：01-2011-7732

图书在版编目（CIP）数据

敏捷技能修炼：敏捷软件开发与设计的最佳实践/（美）沙洛维（Shalloway，A.）等著；郑立，邹骏，黄灵译. —北京：机械工业出版社，2012.8

书名原文：Essential Skills for the Agile Developer：A Guide to Better Programming and Design

ISBN 978-7-111-39527-0

Ⅰ. 敏… Ⅱ. ①沙… ②郑… ③邹… ④黄… Ⅲ. 软件开发 Ⅳ. TP311.52

中国版本图书馆CIP数据核字（2012）第194447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：盛思源

北京市荣盛彩色印刷有限公司印刷

2012年9月第1版第1次印刷

186mm×240mm · 12.75印张

标准书号：ISBN 978-7-111-39527-0

定价：59.00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

推 荐 序

软件行业一直有这样的传说，认为同等经验的两个不同程序员，在效率和质量上可能会有10倍的差距。不同的说法很多，具体的倍数差距也各不一样，但都证明了一点，不同人的编程生产力确实存在着巨大的差距。而且，一般来说，那些在单位时间内能够写出更多代码的程序员，他们的代码质量相对更低且更容易出错。我们不由得要问，大家都是写代码的，造成如此巨大差距的原因何在？

俗话说“失之毫厘，谬以千里”，对于程序员这个职业来说，若失去良好的编程习惯，那么谬的就是效率、成效和质量。养成良好的编程习惯无比重要，而其中最重要的就是一定要有设计意识，好的程序员不仅仅只是意识低下的键盘、鼠标输入器，而是要将优秀的设计思想和对实现对象的理解融入其中，从而开发出高质量、简洁的代码。

本书的几位作者在书中和大家分享了他们高效软件开发的经验，给大家提供了4个撬动敏捷软件开发的“小舵板”：意图导向编程、分离构造与使用、测试先行，以及Shalloway原则。无须从一开始就考虑全盘实践所有的敏捷编码原则，只要从这些小细节入手，就能够逐渐地产生大变化。阿基米德曾说过，“给我一个支点，我就能撬动地球。”而这些小细节，就是敏捷程序员撬动高质量软件开发的支点。

不管读者是已经有多年的编程经验还是刚刚入行开始写代码，都可以从此书中有所收获。亡羊补牢为时未晚，现在开始永远都不迟。本书的内容也非常耐读易懂，4位作者就像是循循善诱的老师，带着大家一步一步地体会其中的实践过程。他们总是从一个具体的问题出发，讨论有哪些方法可以实现，使用“小舵板”的方式又有什么样的好处，而使用对比的方式则有助于大家加深对这些编码原则的理解。

文中绝大部分内容都是开发人员凭一己之力即可以做到的，只有ATDD和CI需要有其他同事和相应工具的支持才行，这意味着本书的确就是一部开发人员个人的敏捷实践指南。而根据“一万小时天才理论”的说法，任何人都得经过一万小时（即10年每天3小时）才能达到世界级水准。如果我说这本书能够帮助大家事半功倍可能略嫌夸张，但遵循作者们的建议，至少也能够少走弯路，按正常的速度成为高手。

近些年来，我一直在从事敏捷教练和顾问的工作，协助一些组织、部门、团队和个人使用敏捷方法和实践。在与软件开发中不同角色的交流合作中，在和不同的软件开发思想、方法论

的实践者和积极推动者的交流中，我发现，IT 行业其实相当“喜新厌旧”，各种新概念、新方法论、新思想层出不穷，而在这其中唯一不变的就是我们对高质量代码的追求。而这本书的内容，即便到了将来，敏捷可能不再热门的时候，也依然适用，因为它讲的就是如何写出高质量代码的技巧和习惯。

书写高质量代码，是每一个软件人的责任。如今有很多团队，他们耗费大量的时间纠缠于所谓的“遗留系统”，为其进行后续开发和维护。这一切都是代码质量低下惹的祸，不管是从创建代码的第一天起，还是在修改代码的每一次，软件缺陷就已经悄悄地扎下了根。其实代码就像是土壤，盐碱地里种不出好粮食，不勤耕细作同样也种不出好粮食，而这一切就需要我们能够了解土壤自身的情况以及耕种所需要顾及的各个方面。代码也一样，你熟悉你的代码吗？面对满目疮痍的遗留代码，你会拿着化肥、杀虫剂、催产素等一大堆化学制品往上面泼，只为了能够继续生产出“可运行”的产品功能吗？着手开发代码之前，你清楚它的脾性吗？知道种植庄稼需要什么样的环境和呵护吗？你会不断地甚至优先地进行检查吗？你会设立可视化手段持续地监控其健康指标吗？你会选择好的种子和作物吗？

当然，书写高质量代码也不仅仅是个人层面的事情，它也关系到组织和企业软件研发的延续性。铁打的营盘流水的兵，人员流动对企业来说是常事，而代码则屹立不动、不断延续，直到某一天产品也寿终正寝的时刻便化为灰烬。阅读和修改晦涩难懂的代码，总是比阅读、修改简单易懂的代码来得困难，耗时也更多，出错的概率也更高，这一切也都意味着更高的时间和资金成本。当然，除了可读性，代码的设计和结构也很重要，影响范围和效果也更大。

遇到丑陋的代码，我们的第一反应可能是躲避，也可能是“那就改掉呗”。然而，“冰冻三尺非一日之寒”，恶劣代码的形成也不是一朝一夕的，要扭转局面需要比制造局面更坚强的毅力。而在这一切背后，是人的习惯和思维定势，需要创建出一种鼓励产出优良代码、鼓励形成良好习惯的环境，从而培育出更良好的代码，维持可持续的研发速度，不为缺陷修复所累。每一位程序员都应该阅读此书，学习“小舵板”，持之以恒地磨练。研发经理也应该阅读此书，以此为标准，创造出所需的环境并鼓励团队和个人加以实践。

“己所不欲，勿施于人”，没有人愿意维护难以理解、难以修改、难以增加功能的代码，那么也应该所有人都努力督促自己不写出这样的代码。就让我们从现在开始，努力地写高质量代码吧！

徐毅

诺基亚　敏捷及精益教练

Agile China 2012 大会执行副主席

Scrum Gathering Shanghai 2011 及 2012 核心组织者

译 者 序

一直记得一句话：一个好的程序员，每天至少写一行代码。

经常看到一句话：程序写得好不好，就看架构写得好不好。

现在想到一句话：是否敏捷程序员，就看有没有看过本书。

翻译这本书之前，我们每人每年平均能看十来本书，以为翻译也是个轻松活，应该可以轻松搞定。结果翻译这本书时，我们三个人花了将近 9 个月的时间。回顾整个过程，不是这本书涉及的技术难倒了我们，而是担心没有理解原作者的意图，生怕给读者传达了错误的信息。好书就应该体现出它应有的价值。我们在文字表达上尽量避免使用拗口的方式，力争用朴素的语言给大家打开这扇学习之门，唯一遗憾的是我们使用的语言还不够诙谐和幽默，所以请你在看本书的时候，尽量选择能够放松的环境，来点背景音乐可能会更好。

译完本书最大的感悟是，“世界上不存在这样一种方法：只要套用，就可以写出完美的软件，无论使用的是哪种设计模式；但是却存在一种开发方式，可以帮助我们一步步构造出需要的软件和架构——这就是敏捷软件开发模式。”

本书不是一本介绍编程基础的书，作者没有花任何精力去介绍如何使用一门编程语言，也没有在此介绍任何工具。自始至终，作者围绕一个主题——“敏捷式编程”。书中引用了作者工作过程中的大量实际例子作为参考，认真解释了每一步分析过程，如果你正处于追求设计能力提高的阶段，或者你正为如何减轻以后重构的负担，又或者碰到了一些一直解决不了的设计问题，那么你将很可能在本书中得到一些启发。

本书共分为四个部分。

第一部分重点介绍了“小舵板”理论，顾名思义这是一种四两拨千斤的思想方法。如何使用我们每个人心中的小舵板是每个架构师应该考虑的问题。这一部分介绍了多种小舵板：意图导向编程、分离构造和使用以及测试先行。此外还介绍了业界常用的几种最佳实践，包括如何封装、面向接口的设计和验收测试驱动等。

意图导向编程并不是一个新概念，如果你看过 Net Objective 产品开发系列丛书，应该对它或多或少有所了解。本书将以意图导向编程开始，让你了解到如何从上往下，逐步将使用的需求转换为被封装功能或者意图。就是那么简单的层层分解和封装，加上我们基本的设计方法即可实现意图导向编程。通过意图导向编程，我们可以使代码的内聚性和可读性得到提升，并易

于重构和测试。

和意图导向编程有相同优点的是测试先行，这两种方法都是对功能实现的过程。分离构造和使用部分着重于封装如何去构造。再加上 Shalloway——去冗余理论，结合这四种方式，将给我们未来代码的设计和实现带来意想不到的清晰感和发挥空间。你将不会再为重构难而犯愁了。

本书第二部分对过度设计问题和持续集成这两个最佳实践做出了诠释，相信可以帮助那些正处于设计阶段感到困难的架构师一点启示。持续集成应该是老话题了，不过作为敏捷的最佳实践之一，有些经验性的东西，作者很愿意与大家分享。

在本书第三部分中，作者凭借多年的教学经验，分享了很多只有在他们的教学现场才能得到的经验。这些内容你或许听说过，只有少部分人能够真正理解这些精髓。但这些经验却是优秀架构师应该具备的。这部分重点介绍了设计中隐藏的一些问题，以及我们该如何去解决。

例如，对象共性和可变性分析帮助我们定义架构的基础，译者也曾在介绍如何分析用户故事时用到这一理念，颇为受益。开放关闭原则告诉我们任何系统都可能被更改，因此我们的设计要时刻为改变做好准备。遵循开放关闭原则是我们可复用设计的基石。这个部分将用一个实例介绍开放关闭原则，为将来如何做抽象层、如何定义接口带来很多启示。

第四部分是附录，介绍了统一建模语言、提高代码质量的原则，以及如何封装原始数据类型等。

本书除了作者的一些经验总结外，还有很多与作者相交甚深的知名人士的经验，包括《设计模式》、《多范式设计》、《重构》等书的作者，也有来自客户的经验，当然还有作者所在公司内部之间的相互沟通和学习。因此，译者在此不仅要介绍作者所传授的知识，那种获取知识的途径也是难能可贵的学习财富。

作者写了很多，译者在此总结却很少，因为除此以外，本书还有很多对高含金量技术的介绍，书中有很多隐藏的宝藏等待你亲自去发掘。相信本书一定会给你的工作和学习带来较大的收获。

参与本书翻译的主要人员是郑立、邹骏和黄灵。熬过将近两百多个日夜的激烈讨论，终于完成了本书的翻译。虽然比起作者，我们的付出是微小的，但是这个过程我们收获了很多，不仅仅是知识、友情、历练以及成就感，这些收获在我们自己的人生道路上是无价的。

最后，十分感谢在翻译过程中给予大力帮助的亲戚和朋友，包括：郭继菁对一些重要章节的友情协助；郑立贤惠的妻子和可爱的女儿的精神鼓舞；邹骏妻儿的默默奉献；黄灵家人的温馨陪伴。正是这些亲人和朋友的贡献，给敏捷注入了生命力，让思想随着这本书可以不断地延续和扩展。

译者

2012 年 7 月

丛书[⊖]前言

如果你和我一样，常常认为前言没什么实质性内容，从而跳过这个丛书前言直接看下文的话，你就会错过不少有价值的东西。

我们一起来思考一个许多人都知道但很少思索过的故事。这个故事阐明了软件行业的困境。这也是我们为什么要写 Net Objectives 公司产品开发系列丛书和本书的背景原因。

我从 1970 年起就一直从事软件开发。直到 40 年后的今天我还是对软件开发有始终如一的新奇感。在这 40 年里，我一直致力于如何将事情做得更好，但有些事也会让人觉得确实能力有限。但是我很喜欢这一行。

在职业生涯中，我也会对其他行业产生兴趣，尤其是工程业和建筑业。直至今日，工程业和建筑业也曾经遭遇过一些重大的失误：像比萨斜塔、塔科马海峡大桥、哈勃望远镜。建筑工程师开始也不太了解力是如何相互作用的。他们通过一次次尝试和失败，从中逐渐学习而慢慢有了提高。这个过程整整用了数个世纪，之后他们才彻底弄清楚应该怎样科学地实施工程和修建建筑物。

时至今日，所有造桥的人都遵循着这个久经考验的建筑学实践（包括应力、抗压等因素）。然而，软件开发人员每天都由着自己的性子来写代码，很少，或者说根本没有同事会去抱怨他们。而这种现象并不只是出现在开发人员身上：管理者经常要求别人如何工作，连他们自己都明知只会起到反效果。这就让我们不得不深思，为什么要这么工作呢？

以上只是故事的一部分。而实际上，剩下的大半部分才关系到我们为什么要将这套丛书起名叫做 Net Objectives 产品开发系列。Net Objectives，顾名思义，本系列所有图书的作者不是 Net Objectives 公司的员工，就是与我们志同道合的人。那为什么叫产品开发呢？这是因为在编写软件时，应该牢记的是：软件开发实质上就是产品开发。

就其本身而言，软件几乎没有什么内在价值。它的价值只有在交付产品和服务后才能体现出来。也就是说，要把软件开发视为产品开发的一部分——我们用它去发现和创造能迎合客户需要的产品，与此同时帮助他们提升公司的战略目标。

⊖ 本书原版是“The Net Objectives Lean-Agile Series”（Net Objectives 精益 - 敏捷系列）丛书中的一本。——编辑注

Mary 和 Tom Poppendieck 在他们的大作《Implementing Lean Software Development：From Concept to Cash》（Addison-Wesley，2006）中提到：

> 在开发过程中，那些包含软件的产品、活动和流程才是产品真正需要开发的地方。软件开发只是整体产品开发流程中的一个子集。所以，从某种意义上来讲，我们可以把软件开发称为产品开发的一个子集。因此，如果我们想要了解精益软件开发，就应该好好探索优秀产品的构成要素有哪些。

换而言之，软件本身并不重要。它的价值在于对业务、顾客、用户的贡献，那才是最重要的。在开发软件时，必须时刻留意我们的工作给客户带来了什么样的价值。在某种程度上，大家都明白这点。但往往来自组织架构的屏障，阻碍着我们协同工作，令我们无法集中精力去创造价值。

在跨组织的环境下实现有效的产品开发最好的方法（多半也是唯一的方法），就是把那些既关系到我们的工作，又关系到从事工作之人的原则和实践妥善地结合在一起。这要求开发团队、管理层，甚至执行董事来共同推动，完成这些工作。这也就是编写 Net Objectives 产品开发系列丛书的初衷所在。

长期以来，软件行业经历着犹如永无止境的钟摆运动般的变革，从没有流程到太多流程再回到没有流程；从企业级的重量级控制方法，到培养专业的团队让他们注重一线项目的开发。发展到现在，管理层和员工紧密合作，齐心协力将贯穿企业的产品商业价值最大化的时机已经到来。相信精益原则可以指导我们做到这些。

精益原则告诉我们，要提高工作速度和质量（从而降低成本），就应该坚持不懈地改善工作机制和企业运作方式。这要求做到以下几点：

- 业务上，选择能获得最大回报价值的软件开发领域。
- 团队上，有自己的工作机制，并不断改进。
- 管理上，培训并帮助团队进行工作。
- 质量上，有正确的质量评估和褒奖体系。

目前，要在软件开发行业完全实现以上几点似乎还任重而道远，但未来肯定是往这个方向发展。虽然很困难，但是精益原则将帮助我们实现前三点，而且随着对编程和设计专业化的理解更加成熟，还会帮助我们改进第四点，所以这也没我们想的那么可怕。

随着对精益、敏捷、设计模式、测试驱动开发价值的不断发掘，我们在规范、思维方式、技能各方面都在不断地改进现有的分析和编码方法，软件开发终有一天也会从纯粹的手动式作

坊发展成为真正的专业化。目前我们已经具备了这样的能力，现在需要的只是树立信念，用新的姿态去面对。

“Net Objectives 精益 - 敏捷系列”丛书旨在发扬这种信念。我们的目标是帮助管理层和员工共同协作，实现“整体优化”：

- **整个组织** 将企业、团队和个人整合起来，使他们的合作完美无间。
- **整个产品** 不仅仅是开发，还包括维护和集成。
- **整个周期** 不仅仅是现在，还有将来。希望我们的努力能带来持续不断的投资回报。

本书在本系列丛书中的作用

曾经有一段时期，敏捷方法不再包含技术性的实践。值得庆幸的是，它们又回来了。Scrum 爱好者意识到对其自身敏捷性的体现而言，这些技术性实践是非常必要的。我们曾经提到，极限编程（eXtreme Programming，XP）有一条主线贯穿于它的所有实践当中，而在那时候，看板（Kanban）和极限编程就已经广泛地结合在一起使用了。

本书是为了对那些刚开始采用精益、看板、Scrum 或者敏捷的团队，提供一个权宜之计。抛开采取哪种开发方法不谈，团队在不同时期写出来的代码风格也将不同。这是一个自然的演化过程。多年来，那些认真接受我们培训的人，大多数已经掌握了他们所需要掌握的一切，这令我们很受鼓舞。不管采用什么方法去开发，他们只需要稍做调整，或者得到少许关键的领悟，都将极大地提高开发效率。

为什么本书只是个“权宜之计”呢？因为这只是实现目标的一种方式而已。它为开发人员提供了一套最精炼的技术集合，帮助他们在递增式开发中变得游刃有余。一旦开发人员完全掌握了这些技能，他们就能判断下一步需要干什么，或者下一步需要哪些新的能力。本书就像为开发人员提供了一个启蒙导师，让他们做好准备，在接下来的旅程中充满乐趣。

一个时代的结束，意味着新时代的开始

我认为软件行业目前正处于它发展的关键时刻。一方面软件行业的不断扩张使之成为我们日常生活中不可或缺的一部分，另一方面软件开发团体正面临一些岌岌可危的问题。原始代码犹如过期食品，越来越难以清理和维护，员工超负荷的工作也盼不到尽头。虽然敏捷方法已经令许多团队取得了巨大进步，但这还远远不够。尽管如此，我们有信心找到答案。我们相信在精益原则的指导和敏捷实践的相互结合下，将创造出真正的软件专业化。

自从本系列第一本书面世以后，我发现软件行业已经有了相当大的改观。尤其是看板的出现，改变了许多团队和组织的工作方式。这令我信心十足。

希望本系列丛书对你有指导意义。

Alan Shalloway

Net Objectives 公司[⊖]CEO

⊖ 该公司的宣言是：为实现企业和团队敏捷而奋斗。

前　言

尽管本书是一本技术性的书籍，但里面涉及的很多想法都是我们在 Net Objectives 公司的敏捷开发培训课程中迸发出来的。过去，在给学生传授如何采用 Scrum 或者精益方法的时候，经常有人会问我："我们应该怎样做，才有能力一步步地构造我们的软件呢?"答案对我而言显而易见。他们其实真正想问的问题是："我们怎么才能用最好的方式，学习如何一步步地构造我们的软件?"关于这个问题，有下面三种方法：

- **读书**　我相信任何一个读过并且读懂了《Design Patterns Explained：A New Perspective on Object-Oriented Design》和《Emergent Design：The Evolutionary Nature of Professional Software Development》的人，都知道如何一步步地写软件。
- **参加培训**　这个方法要更好一点。如果能够把 Net Objectives 公司的课程（设计模式（Design Patterns）和浮现式设计（Emergent Design））结合在一起上，那就无法超越了。
- **学习一些"小舵板"（trim tab）**　软件开发中的"小舵板"使得一步步地构造软件更加有效。

第一种方法需要投入很多时间，第二种方法则花费很高。相比之下，第三种方法所需要的投入则小得多。遗憾的是，这些"小舵板"不是简简单单就可以讲清楚的。

"什么是小舵板（trim tab ㊀）呢?"它们是飞机和船只上的零件，用以减少控制飞机副翼或者船舵所需要的能量。但是这里"小舵板"的意思则是来源于 Bucky Fuller 曾经提到过的东西。

> 在思考一个小小的人类个体能够做什么的时候，一些东西曾经给过我很深的冲击。
>
> 想象一下"玛丽皇后"号——当整艘船远去，它的舵浮现眼前。你会发现，在舵的边缘，有一个小零件，叫做"小舵板"。它是一个微缩版的船舵。只需移动这块小小的舵板，产生一个轻微的压力，就可以拉动整个船舵转动，毫不费力。我认为小小

㊀ 或译为配平调整片。——译者注

的个体也可以成为“小舵板”。一个人在社会中的行为，就好比一艘船的舵板，细微的行动也可以驱动社会这艘大船向前行驶。

所以，请叫我“小舵板”。

换言之，这里要学习的“小舵板”，就是指可以让我们在较小投入下获得最深刻理解的措施和真知灼见。在设计模式课程中，我们识别出三个基本的“小舵板”。实践了这三点的学生，他们都看到了自己在设计和编程能力上的巨大提高。“这三点是什么?”当然就是本书要讲的内容：

- 意图导向编程（Programming by Intention）。
- 把使用从构造中分离。
- 编码前考虑可测试性。

这三点非常简单，实践它们也几乎不会增加额外的时间。实际上，它们都是与封装相关的。第一点和第三点封装的是行为的具体实现，而第二点则显然着重于封装如何构造。这个主旨非常重要，因为对实现进行封装是一种抽象过程。它是众多实现方式中的一种——在未来，可能还会有其他的方式。我认为，把新代码集成到原有系统时遇到严重问题的主要原因，就是因为忘记了这一点。

要推荐的第四个“小舵板”是，遵循 Shalloway 原则。这一点需要多花点时间，但总是很有用处。

本书就是“小舵板”的汇集。这些“小舵板”都是 Net Objectives 公司的讲师和教练发现的敏捷软件开发人员可以遵从的基本要素，告诉我们如何以有效的形式来编写高质量的代码。你可以在闲暇的时间里，按任何的章节顺序来阅读本书。也就是说，这些章节顺序如此编排仅仅是为了协助我们理顺思绪而已。

致　谢

来自 Alan Shalloway

非常感谢 Buckminster Fuller 对本书的撰写给予了诸多帮助。这里不得不先介绍一下这位朋友：认识 Bucky 的朋友都知道，他是个非常亲切的人。很遗憾，我从来没有见过他，否则他一定会成为我的挚交好友。Bucky 因发明短程线网壳结构（Geodesic Dome）和术语“地球飞船”（Spaceship Earth）而成名。他还创造了“协同论”——系统间转换的理论，我们 Net Objectives 公司本质上也就是做这个的。当然，最重要的是，他提出的术语“小舵板”（在前言中提到的）是本书灵感的源泉。

同时，他也一直是我的心灵导师，令我不断追寻更好的点子。下面这段话摘自我最喜爱的 Bucky 理论：

> 我很热衷于研究人类的奇思妙想和灵光乍现。设想如果你身处一艘失事的船上，并且所有的小船都开走了，这时如果飘过一块钢琴三角盖，它足够使你漂浮在海面上，而且你幸运地保住了性命。但这并不是说，最佳的救生工具要设计成钢琴三角盖那个样子。我发现，在面临问题的时候人们往往只局限于用过去经验总结的几个方法，就像在经历了昨天的遭遇后就固守于只用钢琴顶板做救生工具一样，这就缺乏变通。

基于以上这些原因，我对他表示十分感谢。事实上，让我真正意识到应当向 Bucky 先生表示诚挚感谢，是因为从 1983 年起（没有料到的是，他却在这一年离世了），他就一直是我生命中的灵感导师。有人睿智，有人高尚，而 Bucky 是两者兼备，德艺双馨。无论你是否了解这位伟大的人物，我都建议你去 Buckminster Fuller 协会（http：//www. bfi. org）了解有关他的更多内容。

同样致谢

本书中描述的针对某些技能的观点，都是我们认真思考和研究的总结。每一个敏捷开发人

员都应该掌握。然而，本书并非是我们依靠一己之力写成的，在此我们要由衷地感谢以下这些人：

Christopher Alexander，架构大师，《The Timeless Way of Building》的作者。虽然他并不是软件专业的技术人员，但 Alexander 的强大思想几乎渗透到我们工作的所有方面，尤其是“因地制宜设计”（design by context）的概念。

《Design Patterns：Elements of Reusable Object-Oriented Software》的作者 Erich Gamma、Richard Helm、Ralph Johnson，以及 John Vlissides。虽然我们希望能把他们所做的工作向前更推进一步，但他们表达的智慧仍是如今指导我们思索的根源。

James Coplien 的论文“Multi-Paradigm Design”已经成为一本关于共性和可变性分析（Commonality-Variability Analysis）的教科书。此书现在又帮助我们理解如何使用模式和对象来解决我们面临的问题领域。本书中所传授的很多技巧在 Jim 的书中得到了强有力的实证。

《Refactoring》和《UML Distilled》的作者 Martin Fowler。他还写过许多其他在这方面既有思想又非常有帮助的书。Martin 绝对是开发人员的良师益友。

Ward Cunningham，极限编程的创始人之一，他也是开发人员参与每日测试实践的发起人之一。他的这些核心理念给我们带来了数不胜数的好东西。还有，十分感谢 Ward 发明了维基（wiki）。

Robert C. Martin，《Agile Software Development》及其他许多书籍和文献的作者。“Bob 大叔”教会我们如何将各种关键性的编码技巧结合起来使用，使软件代码变得易读、可扩展、易维护和优雅。

除了这些作者和思想领袖之外，我们还要感谢数万名学员和来咨询的客户。正是他们的不断贡献让我们了解到什么才是真正的好软件，以及如何才能开发出这些软件。有句谚语叫：教学相长，我们对此深信不疑。比起 10 年前 Net Objectives 公司刚成立的时候，如今我们更能体会到这一点。我们的客户给了我们不计其数的机会来拓展我们的思维，验证我们的想法，并在实际的工作中提供至关重要的反馈。

如果没有客户就不会有 Net Objectives 公司。在此对我们的客户表示深切的热爱和真挚的感谢。

目　录

第二部分 基本态度