The Pragmatic Programmers

TURING 图灵程序设计丛书

Beyond Legacy Code

Nine Practices to Extend the Life (and Value) of Your Software

修改软件的艺术

构建易维护代码的9条最佳实践

[美] David Scott Bernstein 著　李满庆 译

作者30年软件开发经验总结

揭示高质量软件的秘密，阐述真正的敏捷开发之道

中国工信出版集团

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

Beyond Legacy Code

Nine Practices to Extend the Life (and Value) of Your Software

修改软件的艺术

构建易维护代码的9条最佳实践

[美] David Scott Bernstein 著　李满庆 译

人民邮电出版社

北　京

图书在版编目（CIP）数据

修改软件的艺术 : 构建易维护代码的9条最佳实践 / (美) 戴维·斯科特·伯恩斯坦著 ; 李满庆译. -- 北京 : 人民邮电出版社, 2017.10
（图灵程序设计丛书）
ISBN 978-7-115-46776-8

Ⅰ. ①修… Ⅱ. ①戴… ②李… Ⅲ. ①软件开发
Ⅳ. ①TP311.52

中国版本图书馆CIP数据核字(2017)第217905号

内 容 提 要

本书会帮你降低构建与维护软件的成本。如果你是软件开发者，将学到一套实践方法以构建易修改的代码，因为在应用当中代码经常需要修改。对于和软件开发者合作的管理者来说，本书会向你展示为何引入这 9 个基本的实践方法，会使你的团队更加有效地交付软件而不至于让软件演变成遗留代码。

本书适合软件开发人员和 IT 经理阅读。

◆ 著 [美] David Scott Bernstein
译 李满庆
责任编辑 朱 巍
执行编辑 张海艳
责任印制 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
北京鑫正大印刷有限公司印刷

◆ 开本：800×1000 1/16
印张：12
字数：284千字 2017年 10 月第 1 版
印数：1－3 500册 2017年 10 月北京第 1 次印刷
著作权合同登记号 图字：01-2016-9522号

定价：55.00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：京东工商广登字 20170147 号

站在巨人的肩上

Standing on Shoulders of Giants

iTuring.cn

站在巨人的肩上

Standing on Shoulders of Giants

iTuring.cn

版权声明

本书赞誉

本书从全新的视角展现了现代软件开发流程。工程师们会在其中找到解决日常问题的方案，而非工程师们可以对软件开发中所面对的挑战和困难有所认识。

——Stas Zvinyatskovsky，埃森哲公司资深首席软件架构师

David帮我们认清了我们是如何陷入此番境地的。他给出了行之有效的理论和工具，也提出了一些值得深刻思考的问题。对于关心软件开发的人来说，本书是一份厚礼。要善用它。

——Ron Jeffries，RonJeffries.com

如果你想要优化软件交付流程，但是感觉到裹足不前、无能为力，那么这本书正适合你。对于开始频繁迭代交付以及尝试采用敏捷却未见显著效果的人来说，这是本好书。

——Gojko Adzic，Neuri Consulting LLP公司合伙人

本书讨论的内容可以帮我让客户更加满意，也能让他们在需求出现变化时一直开心。

——David Weiser，Moz软件工程师

对于所有的开发者和管理者来说，这都是一本好书，而且适用于各类公司的各类代码。

——Troy Magennis，Focused Objective CEO

David的解释清晰明了，我甚至希望开发团队的管理者以及开发定制软件的公司领导们也能看看这本书。理解这些实践，将能构建出更经济、更易维护和扩展的软件。

——Jim Fiolek，黑骑士金融服务公司软件架构师

书中的各种观点令人欣慰。如果可以让人们都遵从这些原则，我们的生活以及软件开发会变得更加轻松惬意。

——Nick Capito，Unboxed Technology公司软件开发总监

我们努力让每一行代码成为真实产品的一部分。要找出让我们误入歧途的原因。要找到合适的方法，让你的团队在现在以及今后能更有效地开发出客户真正想要的产品。

——Michael Hunter，极客，骇客，首席工程师，架构师

序

遗产

遗产是已经死亡的事物存留下的依旧影响着世界的那部分。

能留下遗产的生命是优秀的，但是软件并非如此。我们用温和的词语“遗留”来形容那些已经失去活力但是依旧运行的代码，让那些过去的决策持续影响着那些深陷其中的人。

软件和硬件要区分对待。我们称硬件为“硬”是因为它是固定的，没有工具是无法调整的。软件的“软”是指它由思想而生，通过代码来表达，加载到硬件中然后行使一些职责。

讽刺的是，代码在编写完成脱离开发者之后变得比硬件还难修改。

开发者通过编程的逻辑表达想法和需求，赋予软件生命。就像是无中生有一样，直到我们意识到所有的这些小心论证都是为了让这个新生命成为我们所期望的那样。

敏捷

面对威胁与机遇的时候，反应迅速的组织被称为“敏捷”组织。一个敏捷的组织会吸取经验教训，不会被那些短期内不会修改的软件束缚手脚，无法行动。

一些思想者，包括我在内，选择“敏捷”这个词来形容我们都希望软件能拥有即刻适应需求变化的能力。这股敏捷的势力在新开发的软件中很强烈，而且不以任何形式消失，因为软件如此重要，它应该在这整个软件生命周期中持久旺盛。

这些思想者向那些“算命”的管理实践提出异议，它们要求软件开发者在一个合作的流程中对未来进行预估，然后根据那些预估的准确与否评判开发者。

这些思想者向那些“提前大规模设计”的开发实践提出异议，它们需要预料到所有可能的情况才能保证后面维护不至于出现“大问题”。

然后这些思想者将他们的建议用“软件开发宣言”的形式发布给开发者和管理者，这宣言在今天比在十多年前发布的时候吸引了更多的注意力。

宣言并不完美。它在两个方面有所欠缺。它的短小让读者以为它仅仅是些一般性观点而不是具体编写软件的特定建议。

同时，宣言标题中的“开发”也让读者认为仅仅是新项目才需要采用，到上线时就不需要了。

奏效

受雇的开发者必须意识到，仅仅言听计从是不够的，他们有义务让交付的软件持续产生价值。

软件的复杂度可能因为不当的处理而加剧，成为程序员的负担，难以逃避。大部分的敏捷方法，尤其是本书，描述了在接受这种现实的组织中工作的方式。

这本书温习了那些实践，它们在最近二十年中被一遍又一遍证实了既有效但又难以实施。

“名义上的敏捷”形容那些采用了在无数的书中描述的实践，却收效甚微的组织。本书将这些实践进行解剖，分析其背后的原因。只有牢牢地把握住敏捷的逻辑，才能判断“名义上的敏捷”的问题所在。

计算成本的不断下降，让真正的高质量软件的价值凸显了出来。哪里商业更迅速，哪里就有财富。在边界逐渐瓦解之时，任何生意都需要优秀的软件才能生存。

奇怪的是，激增的廉价计算机让编程更难了。多数的敏捷开发者，所谓的“多语言程序员”，知道他们的技艺在敏捷组织之外很难得到欣赏。

David Bernstein解释了为什么敏捷方法会起作用。他深入挖掘了自身的经验，用自己的故事展示了这些实践的价值。

《修改软件的艺术》明确了成功使用这些实践并从中获取最大价值的前提。

Ward Cunningham

美国俄勒冈州波特兰市

引　言

本书会帮你降低构建与维护软件的成本。

如果你是软件开发者，将学到一套实践方法以构建易修改的代码，因为代码在应用当中经常需要修改。对于和软件开发者合作的管理者来说，本书会向你展示为何引入这9个基本的实践方法，会使你的团队更加有效地交付软件，而不至于让软件演变成遗留代码。为此，你需要的不仅仅是一份技术性的任务清单，还需要对为什么有这些实践方法以及如何实施这些实践方法有着深刻的理解。

每天，我们都会因为遗留代码而损失时间、金钱和机遇。

不同的人对“遗留代码”有着不同的定义，但是简而言之，遗留代码就是指因为种种原因，格外难以修正、改进以及使用的代码。

这样的代码有很多。实际上我所见过的所有生产环境下的软件几乎都是遗留代码。

软件产业通常轻视可维护性，所以到最后，企业花在维护代码上的成本比一开始编写代码的成本还高。正如我们将在第2章看到的，软件开发的低效仅在美国每年就造成百亿美元级的开销，这可不仅仅是某份报告上的抽象数字。我们每天都受到遗留代码的影响。软件很昂贵，容易出错，而且难以改进。

业内外人士已经开始拉帮结派为某些项目管理方法论争论不休（其中不乏杰出的点子），但是为了更好地做出持久性的变革，我们首先要对软件开发的基础目标达成共识。

本书不仅仅是关于如何构建更好的软件，更是关于如何构建更好的软件产业。书中囊括了我身为专业开发者三十年所学的精华。我从业的头二十年都是在传统的瀑布式开发下度过的，系统分别按照设计、构建、测试的阶段开发。问题是，规划软件开发的方式充满了不可预见的问题，这迫使我们必须对质量和预算做出严重的妥协。

但是在最近的十年间，对我以及其他我认识的开始尝试**极限编程**（Extreme Programming，XP）的软件开发者来说，事情发生了改变。采用这种敏捷开发方法论之后，我们放弃试图在一开始把所有事情想明白，而是循序渐进地来做，每次只设计、构建、测试一小部分软件。

极限编程中的一些实践方法，诸如测试驱动开发和重构，在构建和扩展软件过程中降低风险

和成本方面给我上了重要的一课。这些实践方法的应用呈现给我各种各样的解决软件问题的方法。是否能利用这些实践方法，揭示出构建高质量、高可维护的软件的方法呢？

我的回答是响亮的“能”！

在程序员生涯初期，我被指派去从标准普尔的feed中整理股票数据，并将数据发送到客户的私有数据库中。在此之前，这一过程都是手动完成的，容易出错而且平均每天需要花费十四小时完成。我需要将这一过程自动化，但是对于如何找到最佳的解决方案，一开始我却摸不着头脑。

几周之后的某一天，在已经写了四十多页代码之后，我突然灵光一闪，有了重组数据处理方式的想法。在几个小时之内，我就完成了这个项目并且将代码削减到只剩下五页。那天早上开工时预计要花几个月完成的工作，结果在当天下班前就完成了。从那时起，我有过多次灵光一闪，甄别出藏在问题之下的规律，而这些规律让我能够迅速地构建出高可维护的解决方案。

这仅仅是一个例子，说明同一问题的不同解决方式之间有着巨大的效率差别。我从其他开发者那里听到过许多类似的故事。也许你也有灵光一闪之后，问题化繁为简的故事。

以我的专业经验，效率极高的开发者和一般开发者之间的差别可以非常大。我花了职业生涯中的大部分时间，研究那些罕见的、比一般软件开发者效率高数倍的个例。我了解到，那些人并非生来如此。他们仅仅是形成了一些和我们不同的特质，也许是遵循一些不同寻常的实践方式。所有的这些都是可以后天习得的技能。

作为一个新生的产业，我们依然在摸索着，并且学着去抓住重点。构建软件和构建实体物品不同。也许，软件产业面对的一些挑战乃是植根于对软件开发本质的理解误区。为了理解软件开发，使其可以预估，软件开发被拿来和制造业以及土木工程相比较。尽管软件工程和其他工程领域有着相似点，但是有些基础性的差异对那些不是日常编写软件的人来说不那么显而易见。

软件工程和其他形式的工程不一样，这一点并不为奇。医学和法学并不一样，木工和烘焙并不一样。软件开发与且只与一个东西一样，那就是软件开发。我们需要一些实践方法，使我们能够更高效、更可校验、更容易做出更改。如果我们能做到这些，就可以削减构建软件的短期成本，并且完全消除维护它的那些可怕的长期成本。

为此，我给出来自于极限编程、Scrum和精益等敏捷方法论的9种实践方法。当这9种实践方法不仅被应用而且被彻底理解的时候，它们可以帮助我们避免编写的代码在将来变成遗留代码。

尽管还有很多近乎无法修复或者已经濒临废弃的代码，我们依然可以利用相同的实践方法，慢慢地在已经堆积成山的遗留代码之中找到出路。

这9种实践方法将帮助开发团队构建更好的软件，并且帮助整个产业避免浪费金钱、时间和资源。

我见证过客户如何受益于这些实践方法。那些客户曾构建了一些有史以来最大、最复杂的软件。我知道使用这些实践方法可以产生非凡成效，但是仅仅“使用”它们并不能确保有成效。为

了正确使用它们，我们必须理解这些实践方法背后的道理。

这是一个令人着迷的时代，而我们身在其中。当我们作为先行者在未知的领域里探险时，会有灯光为我们照亮方向。本书中的9种实践方法照亮了我的软件开发之路。我希望它们也成为你的指路明灯。

如何使用本书

本书探讨软件开发，但并不是非得成为软件开发者才能理解本书。

软件的编写过程对于大多数人来说可能是个陌生的概念，但是它却影响着我们所有人。它已经成为了一个如此复杂的活动，以至于开发者经常发现自己在试图给客户甚至经理解释各种概念，可能没有一点参考。本书有助于架起沟通的桥梁，用通俗的语言解释技术概念，帮我们在究竟什么是优秀的软件开发这个问题上达成共识。

尽管让不同类型的读者针对技术实践方法达成共识并非易事，但本书的设计目标是为了帮助5种不同的人群对软件开发拥有同样的理解：

- 软件开发者
- 软件开发和 IT 经理人
- 软件购买者
- 各个行业的产品经理和项目经理
- 其他所有对这个重要技术感兴趣的人

我试图让软件开发对于所有人来说都容易理解，通过讲述故事并利用大量的轶事、类比和隐喻将技术概念形象化。软件开发难以一概而论，所以从我所说的话中找出反例是很容易的，但是通常都会有更深层次的见解尚待发现。

为了让非开发者也能够阅读本书并关注这些实践方法的重要性，本书并没有按照教程方式编写。关于撰写用户故事以及重构等各个方面，已经有许多优秀图书了（见参考文献）。尽管本书也提供了很多实际的建议，但是它最特别也是最有价值的地方是探讨了**为什么**这些技术实践方法会有用。这种方式能让那些非开发者（如管理人员和其他利益相关者）明白开发者在构建软件期间面临的问题和挑战。

第一部分：遗留代码危机

在第一部分，我们将直面软件产业发展的若干重大问题，并且发现由于糟糕的软件开发流程导致每年数十亿美元流失。

大部分维系我们生存的软件易出错、脆弱并且近乎无法扩展，也就是我们所谓的“遗留代

码”。这一部分将探讨我们是如何变成现在这样的，这样的情况又意味着什么。探讨不仅针对软件产业，而且针对所有与其相关的人和产业。

如果你已经对软件产业耳熟能详，甚至已经深感疲惫，将会对这些内容“倍感亲切”。你将会更加深刻地理解为什么在软件构建过程中，事情总无法按照计划执行，以及为什么需要更好的方法。

即使对于业内人士来说，第一部分也可以帮助他们从适当的角度来看待这些巨大挑战。管理人员和开发者都可能会从这些问题中发现全新的视点，而这些问题对我们这个产业来说是每天都在发生的。正如一位经理对我说的那样：“它使我的炮火更加精准。”本书可以帮你说服他人：在解决问题之前，至少我们必须识别出问题所在。

如果你是软件行业的局外人，我敢保证第一部分会让你称奇，甚至震惊。

第二部分：延续软件生命（和价值）的 9 种实践方法

在第二部分中，由于之前已经说明了问题所在，本书剩下的四分之三将从阴暗压抑中走出，介绍一套实践方法，提供真实、可操作的解决方案，首先是一些对管理者特别有用的实践方法。

在第5章和第6章，我将提出一些自己实践后总结的建议，不仅针对如何更好地开始实施复杂软件的开发流程，还针对如何一路把控流程直到完成。这两个实践方法对于软件行业以外的人会非常有用，同时我也提供了适用于任何项目管理环境的建议。实施这些实践方法后，你将能够：

- 更加高效地执行任务
- 节约短期和长期成本
- 构建更高质量的软件
- 增加客户的满意度，带来回头客

接下来是我在职业生涯中发现的非常有效的7个技术性实践方法。这些方法更适合软件开发者。

我见过这些实践方法的成功与失败。有些软件开发团队运用了最好的实践方法，但是技术很差，所以无法获取所期望的价值。成功运用这些实践方法的团队和那些失败的团队之间的差异是，失败的团队没有弄明白这些实践方法为什么如此重要。而这正是本书所强调的。

即便这些是技术性的实践方法，我也想敦促管理者们（任何行业的管理者）以开放的心态了解这些基本的概念。了解你管理的开发者所面对的挑战，跟他们分享这些实际的、可以迅速上手的实践方法，帮助在破碎不堪的流程中苦苦挣扎的团队提高效率与效益。

当你读到这些实践方法的描述时，我希望你在思考如何在项目中实施之前，先想想这些实践方法为什么有其价值，这样会帮助你更加有效地运用这些实践方法。

虽然我建议阅读（并应用）这9个实践方法，但是可以按照任意的顺序去执行。我当然意识到本书的每个读者都有具体的问题和需求，所以我将第二部分组织为9个独立的实践方法。专注于那些对你最有帮助和能快速生效的方法，但是不要止步于此。

我并不想独树一帜，而是想抛砖引玉

如何阅读本书，从中有什么样的收获，全在于你。我努力避免玩弄“敏捷”“Scrum”“极限编程”这样的词汇。我希望本书改变人们对于软件开发这个新兴产业的看法，使其进入主流的视野。我希望在软件开发社区里展开讨论，这个社区常常拉帮结派对细节争论不休但缺乏大局观，我们不应该做井底之蛙，而应该基于一个共同目的分享想法，这个目的就是：尽一切可能构建最优秀的软件。

线上资源

本书的代码示例可以在Pragmatic Programmers网站上的本书主页（http://pragprog.com/book/dblegacy）找到。你也可以在论坛中提问并得到回答，还会发现一个可以提交错误报告的勘误表，等等。

致　　谢

首先要感谢我的妻子 Staci Bernstein给了我编写本书所需的空间以及一切支持，让我可以完成夙愿，还有我们的迷你杜宾犬Nicki，它在我写作、编辑的时候时刻陪伴着我。

其次我要感谢 Ward Cunningham 为本书编写了序，感谢他给予我的所有鼓励。

写一本书是一项重大的任务，而本书依仗许多人的支持，对此我深表感激。感谢我的伙伴、同事，以及客户在我写作此书过程中给予的探讨和支持。

我还要对本书的审阅者深表感谢，他们将自己的时间和经验无私地投入本书，提出宝贵意见，才让本书更加优秀，尤其感谢 Scott Bain，Heidi Helfand，Ron Jeffries，Jim Fiolek，Stas Svinyatskovsky，Ed Kraay，James Couball，Pat Reed，Stephen Vance，Rebecca Wirfs-Brock，Jeff Brice，Jerry Everand，Greg Smith，Ian Gilman，Llewellyn Falco，Fred Daoud，Michael Hunter，Woody Zuill，GojkoAdzic，Troy Magennis，Kevin Gisi，David Weiser，Nick Capito，Sam Who，Michael Hunger，Ken Pugh，Max Guernsey，以及 Chris Sterling。

最后，我要感谢在过去三十年中参加过我课程的数千名软件开发者。我非常感激从你们那里所学到的东西！

目　录