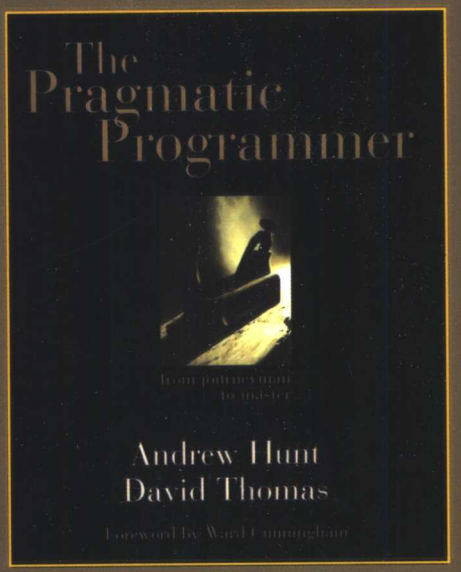
Broadview
WWW.BROADVIEW.COM.CN

程序员修炼之道

The Pragmatic Programmer

from journeyman to master

从小工到专家



[美] Andrew Hunt
David Thomas 著
马维达 译

PEARSON Addison Wesley
電子工業出版社
PUBLISHING HOUSE OF ELECTRONIC INDUSTRY
http://www.phei.com.cn

程序员修炼之道

—从小工到专家—

The Pragmatic Programmer

[美] Andrew Hunt　David Thomas　著

马维达　译

電子工業出版社
Publishing House of Electronics Industry
北京 • BEIJING

内 容 提 要

《程序员修炼之道》由一系列独立的部分组成，涵盖的主题从个人责任、职业发展，直到用于使代码保持灵活、并且易于改编和复用的各种架构技术，利用许多富有娱乐性的奇闻轶事、有思想性的例子以及有趣的类比，全面阐释了软件开发的许多不同方面的最佳实践和重大陷阱。无论你是初学者，是有经验的程序员，还是软件项目经理，本书都适合你阅读。

图书在版编目（CIP）数据

程序员修炼之道：从小工到专家 /（美）亨特（Hunt, A.），（美）托马斯（Thomas, D.）著；马维达译. —北京：电子工业出版社，2004.3
书名原文：The Pragmatic Programmer: From Journeyman to Master
ISBN 7-5053-9719-2

Ⅰ. 程… Ⅱ. ①亨… ②托… ③马… Ⅲ. 程序设计—方法 Ⅳ. TP311.11

中国版本图书馆 CIP 数据核字（2004）第 015597 号

责任编辑：周 筠 方 舟
责任校对：张兴田
印 刷：北京增富印刷有限公司
出版发行：电子工业出版社
北京市海淀区万寿路 173 信箱 邮编 100036
经 销：各地新华书店
开 本：787×980 1/16 印张：22.5 字数：300 千字
印 次：2004 年 5 月第 2 次印刷
印 数：3 000 册 定价：48.00 元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系。联系电话：(010) 68279077。质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

译 序

本书原名“The Pragmatic Programmer”，也就是“**注重实效的程序员**”。正如书名所示，本书将围绕“注重实效”讲述关于编程的各种话题：个人责任、曳光弹开发、调试策略、元程序设计、按合约设计（Design By Contract）、重构、无情的测试，等等。看到本书的目录，你也许会奇怪，300 多页的篇幅，怎么能涵盖如此多内容？但本书的两位作者 Andy Hunt 和 Dave Thomas 的确做到了，他们知道抵达编程的各种维度的途径，并找到了一种言简意赅的方式讲述这些途径；与此同时，在书中还提供了大量资源，可以帮助你找到各种更深入讨论这些话题的读物。本书的各个小节既独立又相关，你可以从头开始阅读，也可以随手翻开任何一页开始阅读——Dave Thomas 就将本书视为一本“洗手间读物”。如果你是编程初学者，你可以从本书中了解到各种编程技术和方法，根据书中的指引拓展你的编程生涯；如果你是富有经验的程序员，同样可以从本书中获益：如果一本书能够全面、明晰地总结你从实践中获得的各种认识、总结你从其他书里散乱地读到的技术和方法，这本书就一定不是无益的。

除了是程序员，Andy Hunt 还是一位木匠和音乐家，而 Dave Thomas 则喜欢驾驶单引擎飞机。尽管作者未曾明言，在本书的许多地方，你都将看到与这样的背景相关的叙述。我想，对于两位作者而言，编程就和木匠活、和音乐创作、或是驾驶飞机一样，既需要禀赋，更需要坚持不懈的学习和训练——这也正是书中所说的，编程是一种技艺，一种需要用心学习的技艺。也许，只有在长久的学习之后，我们才会开始明白书中提到的“hacker”的真正含义：“Someone who loves to program and enjoys being clever about it”（摘自《自由软件杂志》）。

我仍然要感谢侯捷先生和周筠老师，他们像以前一样，为了行业的发展扶掖后进，竭尽心力。谢谢你们的支持和帮助。倘若我未能始终如一，请你们原宥。感谢本书的

编辑方舟先生，他是一个诚恳、好学的年轻人，从不因我的苛刻批评而存有怨言。他的热情、他的年轻，常常让我想起自己那些古怪的、正渐渐没入记忆深处的青春时光。

这是一本“注重实效”的书，其实也可以说，是一本“实用主义”的书。但正因为这样，两位作者在书序的最后给家人的谢辞或许就更加意味深长：

谢谢你们让我们梦想。

马维达 于贵阳

E-mail: weida@flyingdonkey.com

网站：http://www.flyingdonkey.com

前　言

作为评阅者，我得到了提早阅读你拿在手上的这本书的机会。即使当时还只是草稿，它就已是一本很好的书。Dave Thomas 和 Andy Hunt 有话要说，并且知道怎样去说。我见过他们所做的事情，知道他们所说的将是有效的。我请求让我来撰写这篇前言，以便有机会向你解释其中的原因。

简而言之，本书将告诉你怎样以一种你能够遵循的方式去编程。也许你不认为这是一件困难的事情，但事情却并非如此。为什么？原因之一是，并非所有的编程书籍都是由程序员撰写的。其中有许多是由语言设计者、或是与他们有合作关系的报刊记者编撰而成，意在推销他们的作品。那些书告诉你怎样通过某种编程语言进行表达——这当然很重要，但却只是程序员所做的事情的一小部分。

除了通过编程语言进行表达，程序员还要做些什么？嗯，这是一个更深入的问题。大多数程序员在解释他们所做的事情这个问题上都会有困难。编程是一项充满了各种细节的工作，追踪这些细节需要专注。时间流逝、代码出现，你查看它们，那里全是些语句。如果你不仔细思考，你也许会以为编程不过就是敲入某种编程语言的语句。你当然错了，但找遍书店的编程专柜，你却还是讲不出所以然。

在《程序员修炼之道》一书中，Dave 和 Andy 将告诉我们怎样以一种我们能够遵循的方式编程。他们何以能这样聪明？他们不也是和其他程序员一样，专注于各种细节而已吗？答案是他们在做某件事情时，会把注意力投注在他们在做的事情上——然后他们会试着把它做得更好。

设想你在参加一个会议。或许你在想，这个会议没完没了，你还不如去写程序。而 Dave 和 Andy 会想，他们为什么在开会，他们想知道是否可以通过另外的方式取代会议，并决定是否可使某样事情自动化，以使开会的工作推后。然后他们就会这样去

做。

这就是 Dave 和 Andy 思考的方式。开会并非是某种使他们远离编程的事情。开会就是编程，并且是能够加以改善的编程。我之所以知道他们以这样的方式思考，是因为这是书中的第二条提示：思考你的工作。

那么再设想一下，他们这样思考了几年。很快他们就会拥有一堆解决方案。现在设想他们在工作中使用这些解决方案，又是几年；他们还放弃了其中太过困难、或者不能总是产生结果的解决方案。噢，这样的途径几乎定义了“*pragmatic*”（注重实效）的含义。现在设想他们又用了一两年来写下他们的解决方案。你也许会想，这些信息可真是金*矿*。你想对了。

两位作者告诉我们他们是怎样编程的，并且是以一种我们能够遵循的方式来告诉我们的。但这一陈述的后半部分的含义也许要多于你所想到的。让我来解释一下。

作者一直在小心避免提出软件开发理论。这是一件幸运的事情，因为如果他们那样做了，他们就不得不为了捍卫他们的理论而对各章进行“调整”。这样的“调整”是，比如说，物理科学中的传统，在这些学科中，理论不是最终成为定律，就是被静静地丢弃。而另一方面，编程所具有的法则（如果有）却非常少。所以围绕想要成为法则的东西形成的编程建议在纸面上也许显得很好，而在实践中却无法让人满意。这也是那么多方法学书籍误入歧途之处。

我研究这一问题已有十多年，并发现一种叫做模式语言（pattern language）的方法最有前途。简而言之，模式就是解决方案，而模式语言就是相互支援的若干解决方案的系统。围绕着对这些系统的探求，已经形成了一整个社群。

本书不只是一堆提示。它是一种“披着羊皮”的模式语言。我这样说，是因为每一条提示都汲取自经验、作为具体建议讲授、并与其他提示关联而形成系统。是这些特征使我们能够学习并遵循模式语言。在本书中它们以同样的方式发挥着作用。

你可以遵循本书的建议，因为它们是具体的。你不会发现含混不清的抽象。Dave 和 Andy 直接为你而写，就好像每一条提示都是能给你的编程生涯供给能量的重大策略。他们让提示保持简单，他们讲故事，他们使用轻松的笔触，他们接着还给出了各

种问题的解答，这些问题将在你进行尝试时出现。

不仅如此。在你阅读了十或十五条提示之后，你将开始看到工作的另外一个维度。我们有时称之为“*QWAN*”，也即“*quality without a name*”（无名的品质）。本书的哲学将渗入你的意识，并与你自己的哲学交融在一起。它不鼓吹，它只是讲述什么可行。但在讲述中却又有更多的东西到临。这正是本书美之所在：它体现它的哲学，以如此谦逊的方式。

这就是它：一本易于阅读——也易于应用——的关于整个编程实践的书。我一直在不断讲述它为何有效，而你关心的也许只是它的确有效。它的确有效，你会看到的。

——*Ward Cunningham*

序

本书将帮助你成为更好的程序员

不论你是单独的开发者，是大型项目团队中的一员，还是同时与许多客户共事的顾问，这都没有关系。本书将帮助你，作为一个个体，更好地完成工作。本书不是理论书籍——我们专注于实践性的话题，专注于让你利用你的经验做出更有见识的决策。*pragmatic* 一词来自拉丁语的 *pragmaticus*——“精于事务”——后者又源自希腊语的 $\pi\rho\alpha\tau\tau\varepsilon\iota\upsilon$，意为“to do”。这是一本关于“doing”的书。

编程是一种技艺。用最简单的话表述，编程可归结为让计算机做你（或你的用户）想要它做的事情。作为程序员，你既是倾听者，又是顾问；既是解释者，又是发号施令者。你设法捕捉难以捉摸的需求，并找到表达它们的方式，让一台纯粹的机器能够合理地处理它们。你设法为你的工作建立文档，以使他人能够理解它；你还设法使你的工作工程化，以使他人能够以它为基础进行构建。还有，你设法在项目时钟无休止的“嘀嗒”声的催迫下完成所有这些工作。你每天都在创造小小的奇迹。

编程是艰难的工作。

有许多人声称要给你帮助。工具供应商吹嘘它们的产品所展现出的奇迹。方法学古鲁（guru）允诺说他们的技术保证有效。每个人都声称他们的编程语言是最好的，而每一种操作系统都是对所有可以想象得到的问题的解答。

当然，所有这些都不是真的。并不存在容易的答案。也不存在最佳解决方案这样一种东西，无论它是工具，是语言，还是操作系统。能够存在的只是在某些特定情形下更为适宜的系统。

这正是注重实效（pragmatism）登场的地方。你不应该局限于任何特定的技术，而是应该拥有足够广博的背景和经验基础，以让你能在特定情况下选择好的解决方案。

你的背景源自对计算机科学的基本原理的理解，而你的经验来自广泛的实际项目。理论与实践的结合使你强大起来。

你调整你的方法，以适应当前情形与环境。你判断对项目有影响的所有因素的相对重要性，并利用你的经验制定适宜的解决方案。你随着工作的进展持续不断地进行这样的活动。**注重实效的程序员**不仅要完成工作，而且要完成得漂亮。

谁应该阅读本书

本书的目标读者是想要变得更为有效、更多产的程序员。或许你觉得灰心，因为你好像没有在实现自己的潜能。或许你看见同事似乎在使用一些工具，使他们自己比你更多产。也许你现在的工作使用的是较老的技术，而你却想要知道怎样把较新的思想应用于你正在做的工作。

我们不会假装自己拥有所有的（或者即使是大部分）答案，我们的思想也并非适用于所有情况。我们所能说的只是，如果你遵循我们的方法，你将迅速地获取经验，你的生产力将会提高，并且你还将更好地理解整个开发过程。你将能编写更好的软件。

注重实效的程序员有哪些特征

每一个开发者都是独特的，有着个人的力量和弱点、偏好和嫌恶。随着时间的过去，每一个开发者都会营造出他或她自己的个人环境。这个环境会有力地反映这个程序员的个性，就像他或她的业余爱好、衣着或是发型一样。但是，如果你是一个**注重实效的程序员**，你就会具有下列特征中的许多特征：

- **早期的采纳者/快速的改编者。**你具有技术和技巧上的直觉，你喜爱试验各种事物。给你一样新东西，你很快就能把握它，并把它与你的知识的其余部分结合在一起。你的自信出自经验。

- **好奇**。你喜欢提问。那很漂亮——你是怎么做的？你用那个库时有问题吗？我听说的这个 BeOS 是什么？符号链接是怎样实现的？你是收集小知识的林鼠（pack rat），每一条小知识都可能会影响今后几年里的某项决策。
- **批判的思考者**。你不会不首先抓住事实而照搬别人的说法。当同事说“因为就该那么做”或者供应商允诺为你的全部问题提供解决方案时，你就会嗅到挑战的气息。
- **有现实感**。你会设法理解你面临的每个问题的内在本质。这样的现实主义给了你良好的感知能力：事情有多困难，需要多长时间？让你自己了解某个过程会有困难，或是要用一点时间才能完成，能够给予你坚持不懈的毅力。
- **多才多艺**。你尽力熟悉广泛的技术和环境，并且努力工作，以与各种新发展并肩前行。尽管你现在的工作也许只要求你成为某方面的专才，你却总是能够转向新的领域和新的挑战。

我们把最基本的特征留到了最后。所有**注重实效的程序员**都具有这些特征。它们基本得足以用提示的方式来陈述：

提示 1

Care About Your Craft
关心你的技艺

我们觉得，除非你在乎能否漂亮地开发出软件，否则其他事情都是没有意义的。

提示 2

Think! About Your Work
思考！你的工作

为了让你成为**注重实效的程序员**，我们向你发出挑战：在你做某件事情的时候思

考你在做什么。这不是对当前实践的一次性审计——它是对你每一天、在每一次开发上所做出的每一项决策的批判评估。不要依靠自动驾驶仪。不间断地思考，实时地批判你的工作。老 IBM 公司的箴言，THINK!，是**注重实效的程序员**的曼特罗（mantra，印度教或佛教的颂歌、咒语——译注）。

如果这在你听来是困难的工作，那么你就正在展现出有现实感的特征。这将占据你的一些宝贵时间——很可能是已经处在极大压力之下的时间。酬劳则是更为活跃地参与你喜爱的工作、感觉到自己在掌握范围日增的各种主题以及因感受到持续的进步而欢愉。从长远来说，你在时间上的投入将会随着你和你的团队变得更为高效、编写出更易于维护的代码以及开会时间的减少而得到回报。

注重实效的个体，大型的团队

有人觉得在大型团队或复杂项目中没有个性的位置。“软件构造是工程学科。”他们说：“如果个别的团队成员自行其是，团队就会崩溃。”

我们不同意这种看法。

软件的构造应该是工程学科。但是，这并不排斥个人的技艺。想一想中世纪在欧洲建造的大教堂，每一座都需要数千人年的努力，跨越许多个十年。学到的教训被传递给下一批建造者，后者又通过他们的造诣去提高结构工程的水平。但木匠、石匠、雕刻工和玻璃工都是手艺人，他们解释各种工程需求，以制造超越了建筑的纯粹机械方面的一个整体。他们相信，他们个人的贡献支撑了整个项目：

> 我们，采集的只是石头，却必须时刻展望未来的大教堂。
>
> **——采石工人的信条**

在一个项目的总体结构中，总有个性和技艺的位置。就软件工程目前的状态而言，事情就更是如此。一百年之后，我们的工程看起来或许已很古老，就像是中世纪的大教堂建造者所使用的技术在今天的土木工程师看来很古老一样，但我们的技艺却仍将

受到尊重。

它是一个持续的过程

> 一位参观英格兰伊顿公学的游客问那里的园丁，他是怎样让草坪变得如此完美的。“那很容易，”园丁回答说，“你只要每天早晨拂去露水，每隔一天刈一次草，每个星期碾压一次就行了。”
>
> “就是这些吗？”游客问。
>
> “就是这些，”园丁回答说，“这样做上 500 年，你也将拥有一片漂亮的草坪。”

了不起的草坪需要每天给予一点关心，了不起的程序员也是这样。管理顾问们喜欢在谈话中扔出“*kaizen*”这个词。“Kaizen”是一个日语术语（译注：kaizen，日文“改善(かいぜん)”），表达的是持续地做出许多小改进的概念。它被认为是日本制造业在生产率与质量方面取得长足进步的主要原因之一，并且在世界各地得到了广泛的效仿。Kaizen 也适用于个人。每天为提炼你所拥有的技能而工作，为把新的工具增加到你的技能列表中而工作。与伊顿的草坪不同，你在几天之中就将开始看到结果。几年之后，你将会惊奇你的经验得到了怎样的发展，你的技能得到了怎样的提升。

本书的组织方式

本书由一系列小节组成。每一节都是独立的，并且讨论一个特定的话题。你会发现大量交叉引用，帮助把各个话题置入相关语境（context）中。请随意以任何次序阅读各节——这不是一本需要你从头到尾顺次阅读的书。

有时你会遇到标有提示 nn（比如第 xix 页上的提示 1：“关心你的技艺”）的。除了强调文本中的要点以外，我们还觉得提示有其自身的生命——我们每天都和它们生活在一起。在本书末尾你可以找到全部提示的一览表。

附录 A 包含了一组资源：本书的参考文献、Web 资源的 URL 列表、以及我们推荐的期刊、书籍和专业组织的列表。贯穿全书你将会发现对参考文献和 URL 列表（比如[KP99]和[URL18]）的分别引用。

我们还在适当的地方包括了一些练习和挑战。练习通常有相对直接的答案，而挑战则更为开放。为了让你对我们的想法有所了解，我们还在附录 B 中包括了我们对练习的解答，但这些练习很少只有一个正确的解决方案。挑战可以用做小组讨论的基础，或是高级编程课程中的论文作业。

名称的内涵

> “我使用词语时，” Humpty Dumpty 用一种轻蔑的语调说，“我要它是什么意思它就是什么意思——不多也不少。”
>
> ——Lewis Carroll, *Through the Looking-Glass*

在全书的各个地方，你会遇到各种各样的行话（jargon）——它们或者是非常纯正的英语，被故意误用来表示某些技术事物；或者是人为捏造的可怕词语，由对语言怀有嫉妒之心的计算机科学家赋予了各种含义。我们第一次使用某个这样的行话词语时，会试着定义它，或至少是给出关于其含义的提示。但是，我们确信某些行话已经掉进了故纸堆，而另一些，比如对象和关系数据库，十分常用，加上定义反而让人厌烦。如果你确实遇到了一个你未曾见过的术语，请不要轻易跳过它。花点时间查一查，或是在网上，或是在某本计算机科学课本中。而且，如果有机会，就给我们发个邮件，抱怨一下，让我们在下一版中加上定义。

说过所有这些之后，我们决定报复一下计算机科学家们。有时候，有一些行话词语能够完好地表达各种概念，我们却决定忽略它们。为什么？因为已有的行话通常都被限定在特定的问题领域中，或是特定的开发阶段。而这本书的基本哲学之一就是我们推荐的大多数技术都是普遍适用的：例如，模块性适用于代码、设计、文档以及团队组织。当我们想要在更宽泛的语境中使用传统的行话词语时，它就会造成混淆——

我们似乎无法克服原来的术语所附带的包袱。在发生这样的事情时，我们就会发明我们自己的术语，并以此为语言的衰败做出贡献。

源码与其他资源

书中所示的大部分代码都摘录自可从我们的网站上下载的可编译源文件：

```
www.pragmaticprogrammer.com
```

你也能在那里找到我们认为有用的资源链接，以及本书的更新及关于其他**注重实效的程序员**的开发活动的新闻。

给我们发送反馈

我们将重视你的来信。意见、建议、文本中的错误、或是例子中的问题都很欢迎。我们的邮件地址是：

```
ppbook@pragmaticprogrammer.com
```

致谢

当我们开始撰写本书时，我们并不知道它最终会成为这样一个协作的成果。

Addison-Wesley 一直都很卓越，从选题到待印制的拷贝，他们领着一对幼稚的黑客走过了整个书籍制作过程。十分感谢 John Wait 和 Meera Ravindiran 最初的支持；感谢 Mike Hendrickson，我们热心的编辑（和小气的封面设计者！）、还有 Lorraine Ferrier 以及 John Fuller 在制作上提供的帮助；感谢不屈不挠的 Julie Debaggis 把我们大家团结在一起。

然后是评阅者：Greg Andress、Mark Cheers、Chris Cleeland、Alistair Cockburn、Ward Cunningham，Martin Fowler、Thanh T. Giang、Robert L. Glass、Scott

Henninger、Michael Hunter、Brian Kirby、John Lakos、Pete McBreen、Carey P. Morris、Jared Richardson、Kevin Ruland、Eric Starr、Eric Vought 和 Chris Van Wyk。没有他们细心的评阅和宝贵的洞见，这本书不会像现在这样易读、准确，并且会加长一倍。谢谢你们大家的时间和智慧。

几年来，我们与许多锐意进取的客户工作在一起，取得并提炼了我们在此写下的经验。最近，我们有幸与 Peter Gehrke 一道参与了若干大型项目的开发，非常感谢他对我们的技术的支持和热情。

本书通过 Linux 下的 Bash 和 zsh shell，使用 LaTex、pic、Perl、dvips、ghostview、ispell、GNU make、CVS、Emacs、XEmacs、EGCS、GCC、Java、iContract 和 SmallEiffel 制作。令人惊讶的是，所有这些奇妙的软件都可以自由获取。我们应该向世界各地的**注重实效的程序员**大声说："谢谢你们"，他们为我们大家奉献了上述的以及其他的作品。我们尤其要感谢 Reto Kramer 在 iContract 方面给我们的帮助。

最后，但绝非最不重要的是，我们对我们的家人亏欠甚多。她们不仅要忍受深夜的键盘敲击声、巨额的电话账单以及我们长期心不在焉的状态，她们还一次又一次很有风度地阅读了我们所写下的东西。谢谢你们让我们梦想。

Andy Hunt

Dave Thomas

目 录