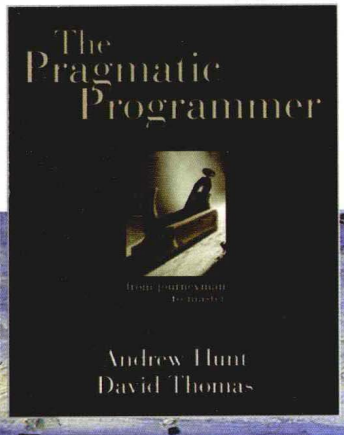
Broadview®
www.broadview.com.cn
PEARSON

传世经典书丛
Eternal Classics

程序员修炼之道
——从小工到专家



[美] Andrew Hunt
David Thomas 著
马维达 译

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

程序员修炼之道——从小工到专家

The Pragmatic Programmer : From Journeyman to Master

[美] Andrew Hunt, David Thomas 著

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

内 容 简 介

《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。

版权贸易合同登记号：图字：01-2003-4993

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，2011.1
（传世经典书丛）
书名原文：The Pragmatic Programmer: From Journeyman to Master
ISBN 978-7-121-12336-8

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

中国版本图书馆 CIP 数据核字(2010)第 224029 号

责任编辑：周　筠
文字编辑：许　艳
印　　刷：
装　　订：北京市铁成印刷厂
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
开　　本：787×980　1/16　印张：19.5　字数：300 千字
印　　次：2011 年 1 月第 1 次印刷
定　　价：55.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

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。

服务热线：（010）88258888。

悦读上品 得乎益友

孔子云："取乎其上，得乎其中；取乎其中，得乎其下；取乎其下，则无所得矣"。

对于读书求知而言，这句古训教我们去读好书，最好是好书中的上品——经典书。其中，科技人员要读的技术书，因为直接关乎客观是非与生产效率，阅读选材本更应慎重。然而，随着技术图书品种的日益丰富，发现经典书越来越难，尤其对于涉世尚浅的新读者，更为不易，而他们又往往是最需要阅读、提升的重要群体。

所谓经典书，或说上品，是指选材精良、内容精练、讲述生动、外延丰盈、表现手法体贴入微的读品，它们会成为读者的知识和经验库中的重要组成部分，并且拥有从不断重读中汲取养分的空间。因此，选择阅读上品的问题便成了有效阅读的首要问题。当然，这不只是效率问题，上品促成的既是对某一种技术、思想的真正理解和掌握，同时又是一种感悟或享受，是一种愉悦。

与技术本身类似，经典 IT 技术书多来自国外。深厚的积累、良好的写作氛围，使一批大师为全球技术学习者留下了璀璨的智慧瑰宝。就在那个年代即将远去之时，无须回眸，也能感受到这一部部厚重而深邃的经典著作，在造福无数读者后从未蒙尘的熠熠光辉。而这些凝结众多当今国内技术中坚美妙记忆与绝佳体验的技术图书，虽然尚在国外图书市场上大放异彩，却已逐渐淡出国人的视线。最为遗憾的是，迟迟未有可以填补空缺的新书问世。而无可替代，不正是经典书被奉为圭臬的原因？

为了不让国内读者，尤其是即将步入技术生涯的新一代读者，就此错失这些滋养过先行者们的好书，以出版 IT 精品图书，满足技术人群需求为己任的我们，愿意承担这一使命。本次机遇惠顾了我们，让我们有机会携手权威的 Pearson 公司，精心推出"传世经典书丛"。

在我们眼中，"传世经典"的价值首先在于——既适合喜爱科技图书的读者，也符合专家们挑剔的标准。幸运的是，我们的确找到了这些堪称上品的佳作。丛书带给我们的幸运颇多，细数一下吧。

得以引荐大师著作

有恐思虑不周，我们大量参考了国外权威机构和网站的评选结果，并得到了 Pearson 的专业支持，又进

一步对符合标准之图书的国内外口碑与销售情况进行细致分析，也听取了国内技术专家的宝贵建议，才有幸选出对国内读者最富有技术养分的大师上品。

向深邃的技术内涵致敬

中外技术环境存在差异，很多享誉国外的好书未必适用于国内读者；且技术与应用瞬息万变，很容易让人心生迷惘或疲于奔命。本丛书的图书遴选，注重打好思考方法与技术理念的根基，旨在帮助读者修炼内功，提升境界，将技术真正融入个人知识体系，从而可以一通百通，从容面对随时涌现的技术变化。

翻译与评注的双项选择

引进优秀外版著作，将其翻译为中文供国内读者阅读，较为有效与常见。但另有一些外语水平较高、喜好阅读原版的读者，苦于对技术理解不足，不能充分体会原文表述的精妙，需要有人指导与点拨。而一批本土技术精英经过长期经典熏陶及实践锤炼，已足以胜任这一工作。有鉴于此，本丛书在翻译版的同时推出融合英文原著与中文点评、注释的评注版，供不同志趣的读者自由选择。

承蒙国内一流译(注)者的扶持

优秀的英文原著最终转化为真正的上品，尚需跨越翻译鸿沟，外版图书的翻译质量一直屡遭国内读者诟病。评注版的增值与含金量，同样依赖于评注者的高卓才具。好在，本丛书得到了久经考验的权威译(注)者的认可和支持，首肯我们选用其佳作，或亲自参与评注工作。正是他们的参与保证了经典的品质，既再次为我们的选材把关，更提供了一流的中文表述。

期望带给读者良好的阅读体验

一本好书带给人的愉悦不止于知识收获，良好的阅读感受同样不可缺少，且对学业不无助益。为让读者收获与上品相称的体验，我们在图书装帧设计与选材用料上同样不敢轻率，惟愿送到读者手中的除了珠玑章句，还有舒适与熨帖的视觉感受。

所有参与丛书出版的人员，尽管能力有限，却无不心怀严谨之心与完美愿望。如果读者朋友能从潜心阅读这些上品中偶有获益，不啻为对我们工作的最佳褒奖。若有阅读感悟，敬请拨冗告知，以鼓励我们继续在这一道路上贡献绵薄之力。如有不周之处，也请不吝指教。

电子工业出版社博文视点

二〇一〇年十二月

领悟程序员的哲学

在大学的时候，编程是我的兴趣，也是当时我给自己定位的职业方向。

当我在图书馆看到这本《程序员修炼之道》的时候，直觉告诉我应该看看这本书，或许对我的成长有帮助。读完之后更加肯定了自己的直觉是对的。当时我虽然没有实际项目的开发经验，不能一时领悟其意，但我明白，这本书中总结的原则和方法对我来说是极为宝贵的，于是买了一本放在床头。

参加工作后，随着编程经验的积累，我越来越能体会到这本书中的观点。每次重读书中的章节，我都会有新的收获；再结合自己的每次经历，都能与之共鸣——这是对我影响最深的一本书，也是我向朋友和同事推荐次数最多的一本书。有趣的是，书前 Kevin Ruland 的评论说：这是我唯一不会出借的一本书。**究竟是一本什么样的书会让大师如此爱不释手？**

这本书所涉及的内容很广，涵盖了程序员成长过程中和软件开发过程中要注意的地方。从程序员的个体哲学到编码过程中的各个环节，再到团队的项目管理；从程序员要如何扩充知识，如何思考问题，如何利用有效的工具打造个人的工作环境，到项目启动之前如何建立一些基本准则，如何分析、设计、编写、测试、重构，如何实现自动化，甚至是项目团队中提高实效的原则。书中的内容全都来自经验的总结，倡导编程中正确的观念和良好的习惯，而这正是优秀的程序员必须拥有的良好素质。

书中讲述的原则源于实践，高于实践，它们蕴涵着前辈们的智慧。随着知识的扩展、编程体验的增加，对这本书中的内容的理解也会愈加深刻。反过来，对前辈菁华的吸收，有助于我们提高编程水平，开发出更好的产品。

我深信这不是一本只要读一遍的书。这些原则看似简单，但细细品味一番，却是大哲大道，环环相扣，要理解透彻并不容易。例如，提示 44 告诉我们“不要靠巧合编程”，这道理看起来好像很简单，但我发现实际工作中还是很容易就犯这个错的。细想一下 Bug 列表中的问题，其中

大多数问题不正是由于作了不正确的假设，或者是想当然造成的吗？要是一开始就有了深思熟虑，经过了合理的设计，完整有效地进行了测试，应该大部分都可以避免吧。而思考、设计、测试又紧扣书中其他章节。

曾经和朋友讨论关于员工培训的事。**如果给程序员做培训，我首选的材料就是这本《程序员修炼之道》**。

▶LAMP 程序员 赵钟秋（belltoy）
http://blog.belltoy.net/

再次阅读，感受颇多

记得四年前刚开始工作时从公司拿到的第一本书，就是这本《程序员修炼之道》（英文版），作为新入职员工 study group 的学习材料，当时在 senior engineer 带领下和其他同事一起学习了这本书。虽然之前就听说这是一本好书，当时看的时候也只是觉得讲的都有道理，但这些是很自然的啊，干嘛花这么大的篇幅说来说去？所以只是囫囵吞枣地翻过也就扔在一边了。

之后也看过很多类似的书籍，**《程序员修炼之道》也一直是公司新人的必备学习材料**，而我却一直没再重拾这本书仔细读一遍，直到最近周筠老师发给我中文电子版，才又从书架上翻出当年的英文版，对照着中文电子版仔细读了一遍。

此次重读，感受颇多，也颇能理解为何公司一直选用此书作为新人教材。

这本书里虽只包含了很多看似粗浅朴素的道理，实则是若干经验的心血总结。比如谁都知道不要对自己家的破窗户置之不理，可实际中听到太多的妥协：这个代码已经这样了，只能继续在上面贴上丑陋的 workaround，这其实是一种对责任的推卸和对未来的不负责。当然现实是不完美的，有时救火队员也不得不放下破窗户而迁就其他，但作为一个 pragmatic 程序员，保持追求完美的心态还是很有必要的，正因为这个心态，我们才会去追求代码的优美、设计、实现的正交、DRY（Don't Repeat Yourself）原则……

关于 DRY，我想说，不但 don't repeat yourself，也 don't repeat others，我们看到太多重复造轮子的故事，正如书中提到“鞋匠的孩子没鞋穿”，作为一个 pragmatic 程序员，合理地使用工具、库，以及自己积累的开发的轮子，会让自己的 productivity 不断提升。这让我想起公司里一个让人崇拜的“牛人”，大家一直想把产品进程内 cache 做成多进程共享，正在大家讨论该怎么做的时候，“牛人”用短短几天时间已经完成了，众人再次对他又崇拜了一把。“牛人”其实是备有很多现成代码的，完成这个功能只是把之前积累的封装良好的模块重用就可以了。

书中推崇的另外一个方法：曳光弹。自己之前用 prototype，一直犹豫于代码是否需要重用。其实原则上 prototype 的代码应该是抛弃型的，但有时候前期做的一些工作是为了确定方案、构建框架，而这些也是作为后期工作的基础。事实上，在项目前期值得仔细考虑的究竟是采用 prototype 还是曳光弹，取决于它们的适用场景（对于产品开发，曳光弹的应用场景可能相对会更多一些）。

当然，对于书中提到的对知识资产的管理（知识投资）、沟通和交流的重要性等，我想这就不单单对于程序员适用了，任何一个要想有所作为的人，这些方面的重要性都毋庸多说了。而对于自动化和文本处理等方面的经验，也是很多书中都提到的经验之谈（《UNIX 编程艺术》、《卓有成效的程序员》等）。

最后，说一下这本书的译者马维达，我最早是在学校时读过他翻译的 ACE 文档及相关资料，收益颇多，ACE 可谓网络编程技术的集大成者，而这本《程序员修炼之道》则可谓编程的集大成者，从项目管理、软件架构和设计、代码编写和测试，各方面在此书中都有精到的阐述。此书的翻译质量应该说比较准确，基本真实地表达了原书的意思，但正因直译，有些语句可能在理解上会有一些难度，比如 P146,“只要对于那些被耦合在一起的模块而言，这是众所周知的和可以接受的，你的设计就没有问题。”不过细读这本书，这些有所晦涩的内容还是能理解的。当然，译者还是可以适当加些“译注”，让读者更容易理解，内容更顺畅的，比如书中直接用古鲁来翻译 Guru，如果加上解释可能会更好；又比如 Law of Demeter，原书没有解释得太清楚，如果多加些解释可能会更便于理解。

感谢周筠老师让我有机会重温这本优秀的书籍，为了完成作业，也为了让自己的认识提升。

▶ 趋势科技 stuff engineer 邹飞

http://blog.csdn.net/zoufeiyy/

一切阅读都是误读

一切阅读都是误读
—— 安伯托·艾柯

上次读这本书已经是五年前的事了，中文版刚出版我就买了一本。那时候，我的工作相对比较清闲，有大量的时间阅读。恰巧我在负责公司的校园招聘及新员工培训，非常需要一些不错的教材，更早的时候听说过这本书的英文版，但是没能一读，中文版自是不能放过。另外，那年我在写书，记录一些程序员生涯中的心得，对经验的总结都颇有兴趣。

爱不释手，是我第一次读完后的心境。完整经历了人生中第一个成功的大的软件项目后，我有许多感慨。知道了不少东西怎样做对，怎样做不对，但是要一条条写下来，却不知道怎么总结。这本书说出了许多我想说的，但却不知道该怎么说的道理。

接下来的日子，我在公司做过好几次技术培训，课题都是以这本书中的某个或某几个观点，再结合自己的经历展开的。**对于信任我的同学，我总是将它作为第一本列在给他们开的书单中。**

后来，国内又引进了几本类似的好书。比如《代码大全》、《UNIX 编程艺术》。古人云，读书有三上，马上、枕上、厕上。我还真把书买了好几本，分别置于床头、办公桌上，方便睡前、如厕时阅读；手机里放入电子版，上下班路上，偶尔翻阅。这些书的确是值得逐章挑选出来，反复精读的。《程序员修炼之道》却于几年前推荐给新入职的同事，从我的视野里消失了。

这几天，同事把书还了我，加上周筠老师发给我电子版，我又重读了一遍。**原以为那些嚼烂了的东西，不会再有新味道，但是我错了。**

不同的人从不同的角度用不同的方式，阐述相同的道理。其中细微的差异，是需要读者有了许多许多的经历后，才能体会的。比如，在《程序员修炼之道》中花了六页分析 DRY - Don't Repeat

Yourself 原则；而在《UNIX 编程艺术》中把它称作 SPOT - Single Point of Truth，大约用了一页半的篇幅。他们真是想表达完全一致的理念吗？我看未必。所以，作为读者，同样会有许许多多的想法。随着编程经历越来越多，思考次数的增加，重新和这些前辈的思想相印证，也是一件乐事。

我们以为理解了作者，其实是误解。但我们将再一次理解编程。

▶ **网易互动娱乐有限公司 杭州研究中心总监 云风**

http://blog.codingnow.com/

程序员升级必备

学过高中物理的人，应该会记得，原子中的电子获得能量之后，会发生能级跃迁，到达更高的能量状态。其实任何工种都是一样的，要跳出自己的水平，到达更高的级别，不是件容易的事。这个跳跃过程总需要一些东西的辅助。诚然，如果要成为一个好人，那么只要做好在幼儿园中学到的一切就足够。**如果要成为一个好程序员，其实所需要的道理也多不了多少，只不过，当水平不够的时候，永远不能认识到那些朴素道理的重要**。而当水平达到的时候，这些道理自然会明白。所以一本帮助程序员进阶的书，很容易落到低手觉得是废话，高手也觉得是废话的悲惨境地。

好几年前，有人向我推荐过这本《程序员修炼之道》，甚至专门买了一本送到我家。而当年的我，不知道是由于无知、自负、浮躁，或是其他，只草草翻了一下，就下了个“烂书”的定义，扔在书架一角。后来有朋友在我书架上发现，如获至宝，说已经买不到了。我当然乐得送了人情。在我心目中，最好的入门书永远是《代码大全》，那也是对我影响最深的一部书。

几年后，再来谈这本书，发现很多人的评价比我高得多，自知不妙，赶紧找来重读，才知道错过了什么。在一个滥俗的译名之下，在一个看起来不知所云的目录之后，在一些读起来拗口的句子之中，隐藏的竟然是相当伟大的思想——朴素而真挚，简单而有效。这时候我突然明白，**这是一本不逊于《代码大全》的伟大著作**，后者一直被我誉为“新手圣经”。

经验这个东西，往往并不能告诉我们什么一定对，但是可以告诉我们什么一定不对。这本书完全是经验凝成，没有大道理，没有新观念。这些朴素的道理就是创造一个合格软件和作一个好程序员所必须了解的。比如“提示 44 不要靠巧合编程”，这句话表达的意思是“不要预设立场”。听起来简单，但是只要随手翻翻你最新写过的一段程序，通常都会发现代码中做了大量的“假设”。书中用一道习题，假设了用户使用命令行环境，假设用户懂英语……都可能导致问题。怕了吧？幸好还有“提示 30 你不可能写出完美的软件”，这可不是帮你开脱责任，而是在讲如何控制需

求，这正是能顺利完成一个项目的根本前提，可惜事实上往往到了项目失败的时候，人们才想起来需求出了问题。

这本书涉猎的范围相当广，如何设计架构，如何思考问题，如何测试，如何编码，如何处理文档……如果细心琢磨，构建软件的所有主干和细微枝节都有所涉及。和很多人的看法不同，我不认为这是一本可以轻松读完的书。一方面，这本书涉及的内容太多，虽然已经尽量讲述，但所有话题都可以继续引申出无限的内容；如果用心，还可以配合附录中所提到的各种论文和资源继续学习。习题也要仔细思考。这绝不是一本小说。另一方面，作者用了大量的隐喻，导致读起来有一定难度。开始我认为是翻译质量有问题，不过慢慢发现美国的读者读起来也未必容易。原因还是涉及的范围过大。我特意模仿这种风格写了本文的第一段，虽然是中文，可读起来也不容易。

可能以上的两点会阻挡一部分人阅读这本书，因我也是曾经受阻的人之一。不过，好书并不会随着时间的推移和平台的变化而消亡，好书只会成为经典。无论是《人月神话》，还是《代码大全》，都在时间的长河中沉淀下来，传颂至今。这本书，虽然也只有 10 年历史，不过现在再来翻看，不仅毫不落伍，甚至感觉穿透了时间，**看到了这些年中不少自己犯过的错误，我相信这也是一本能经得起时间沉淀的书，只不过需要多点耐心**。因此，我郑重地写下这篇书评，希望能读到这本书的人再多一点耐心，越过语言的障碍，直入本质，直至跃向更高级别。这个希望，不仅是对新手所说，其实也包括我自己。如本书开头所说：**注重实效的程序员应该不断学习**。我们都应该不断地学习下去。

▶ 银杏科技创始人 霍炬
http://blog.devep.net/virushuo/

程序员心底的小声音

编程大约有三个境界，新手、高手和高不成低不就的中手。这三个境界，大致和王国维先生划定的做学问的三个境界一一对应。一般来说，如果不经过几十万行的代码的锤炼（衣带渐宽终不悔，为伊消得人憔悴），或者长期在一个高手团队里面打磨切磋，那么无论怎么样的理论熟悉，打字熟练，考试全A，编程起来，都应该算是中手。一个中手如果机缘很好，得到高人亲自指点，则能很快成长为高手；如果没有这样的机缘，那就要在“众里寻她千百度”这个层次苦苦地求索锤炼很久，才能“蓦然回首”。

读书是一种很好弥补没有高手在场的方法，因为书是最好的老师。可现实是，高手写给中手的书很少。在任何行业，适合新手入门的书很多，适合中手的书就很少。原因有两个，一来高手极少愿意耐心指点成长秘诀，即使写了，也是蜻蜓点水，因为这些经验啊结论啊，都被他们本身提炼成了珠玑，他们觉得最重要的也就是那么寥寥几句，也没有太多的废话好写。而读者如果没有类似的经历，看到这些珠玑，也只是觉得把玩颇为有趣而已，极少能有同感。鲜有高手，能把技术书写成散文集，如 Brooks 一样，在《人月神话》中把经验教训和经历背景等一一道来，并且从这些经历中抽出一般性的知识。因此，高手的风格一般是浮光掠影地概括自己领会的几个原则和教训。这些寥寥数语的珠玑，对其他高手来说一看就懂，但是对于中手来说就很难理解了。所以很多高手写出来的给中手看的书就曲高和寡。二来，中手其实水平差异巨大，偏好也各不一样，有的或许根本认识不到自己应该走的成长轨迹，有的认为这些书籍是片面知识，所以把不喜欢的书都扔给到垃圾堆了，光捡自己喜欢的书看；有的未必看得上高手的经验，认为高手说的那些自己也早已领悟到。因此，也不喜欢购买这些书籍。由于这两个原因，造成了高手提携中手的书在市场上很少见到。

不过这样的书倒不是没有，比方说在编程领域，我至少可以推荐这四本书——《Pragmatic Programmer》、《The Art of UNIX Programming》、《Elements of Programming Style》和《The Productive

Programmer》，它们都是高手所写，属于高手指导中手的典范。第二本和第三本我以前介绍过，第四本余晟同学的书评也比我写的好几百倍，所以我就以《Pragmatic Programmer》为例说说这个问题吧。

我们前面说了，对于中手，特别是在“寻她千百度”这个层次的中手来说，或许本身已经捡到了一些珠玑，或许对于像《Pragmatic Programmer》里面说的那些 Tip，有的是深有同感的。比如 DRY（Don't Repeat Yourself 不要重复你自己），基本上大家都知道，可是在实际中（至少我自己）还是不停地一次一次地犯错误，做事情也不符合 DRY 原则（一次一次犯错误本身也是一个 DRY 错误，因为 DRY 原则要求你对每种错误只能犯一次）。我们读的时候深有同感，可写代码的时候却忘到 Java 国去了，这还真不是个案，是非常普遍的现象。

能不能让正确的原则指导正确的行动本身，其实就是区分是否是高手的一个显著标志。试想，两个都了解 KISS 原则的程序员在一起写代码，高手的代码必然会自然流露出 KISS 的优雅，而中手或许需要旁人的提醒和多次重构，才能达到理想的状态。出现这个问题的原因很明显——中手没有完全内化 KISS 原则，因此尚且不能“运用自如”。内化是一个非常复杂的认知过程，本身涉及大脑中 mind set 和 paradigm 的切换，所以必然不是一个简单的隔夜就能完成的过程，这也就是为啥能够“消得人憔悴”，但是切换一旦完成，实践中就会自然流露出这种新的认识，也就是到了一个新的境界，发现灯火阑珊处。

那么原则和知识的内化这个过程如何加速呢？也就是说，怎么才能较快地到达高手境界呢？可以肯定地说，光靠对自己说我“下次一定按照这个原则这样做”是不行的。认知科学认为，频繁的高强度的外部刺激和自主的有意识的反复提醒是加速内化的两个重要方法。第一个方法需要外部环境的支撑。试想，如果一个程序员不是天天和复杂的文本处理打交道，他必然没有足够的外部刺激来熟悉和内化正则表达式；如果一个程序员不是天天和极度复杂的大项目打交道，即使用全自动编译环境和自动单元测试，也显得无甚必要。因此，除非你正好掉进了一个天天有高强度训练的环境，否则全靠第一点是不可能的。尤其是自学一门语言和一门技术的程序员，往往在没有高强度训练之前就拿着这些技能投入工作了，因此想成为某方面的高手，只能采取第二条路，就是有意识地强化实践和反复提醒。

《圣经》里有一个故事，说一个人在沙漠里，信心丧失的时候，突然听到“A Still Small Voice”（平静的小声音），即上帝的启示。这个平静的小声音把他从绝望中拉了回来。其实对这个人来

说，他本身的实践能力在“平静的小声音”出现前后并没有多大的改变，唯一的不同就是他知道该怎么做了。

内化一个人知识或认识的时候所循的路径也是一样的。我们常常会“忘了”应该怎么正确地做一件事情（这个地方的“忘了”，指我们之前从书中或其他渠道读到看到了正确的原则或方法，但是在那一刻脑子里根本没考虑这个原则或方法，因为这个原则或方法根本没有亲自实践过，所以根本不是自己的一部分，不属于自己）。在这个时候，如果突然有一个平静的小声音跳出来，说，“嘿，你是不是该遵循这个原则，用这个方法？”无须说，我们对问题的思考就能顿时全面起来，也会更加深刻理解原先读到看到的不属于自己的原则和方法。当然，我们更加感兴趣的是，如何能够在身边没有高手和上帝发出这样的平静的小声音的时候，自己发出这样的小声音？

怎么靠自己呢，记得鲁迅小朋友破坏公物在课桌上刻的“早”么？是的，我们须要抽象出一些简单的词句和规则，靠记忆和不断地提醒，小规模地内化这些小声音，让这些简单的小声音能够时刻从大脑里跳到耳边，提醒自己。具体来说，在阅读上面几本书，尤其是阅读《Pragmatic Programmer》的时候，如果仅仅以普通的浏览的方式阅读，就会很简单地陷入“啊，这个我知道了，啊，那个我了解了，嗯，这个以后要注意”的套路中。这样的阅读方式，只会强化原有的自己已经知道的部分，而不大可能把“以后要注意”这部分全部内化。所以，自负的读者读完之后必然觉得“哈哈，高手不过如此，大部分我也知道嘛”，而不是“是的，我还有不少要注意”。这两种态度，就把高手和易于满足的中手永恒地隔开了。我觉得，**想要内化这些小声音，还是要靠实践，如果不实践，即使你把这些小声音写在 100 块钱的高档笔记本上也没有用**。我个人觉得，理想的阅读状态应该是先大致理解和记住里面的 Tip，然后每周争取实践 2~3 个 Tip。其实这样做完一圈也就是半年，在这一圈后就会记住所有的 Tip 的内容，这时候，小声音就成了自己的一部分了。然后在剩下的几年里，只要时时有这些小声音跳出来，告诉你，“要自动频繁地测试”，或者“别手动做繁琐的工作”，你会很快被强迫转换到高效而优雅的工作状态中来。到了那个时候，这些小声音就再也不会跳出来了，因为你早就自然地遵守这些小声音的要求了。

《Pragmatic Programmer》和《The Elements of Programming Style》书里面的 Tip 都不是来自上帝的话语，但都是值得随身带着的小声音。其实只要处理过实际问题，编过几万行程序，大多程序员都会有或深刻或浅显的对各个 Tip 的感悟，而且我相信或许有程序员对有些 Tip 的认识能比原书的作者还要深刻，这是很正常的。事实上，每一个 Tip 只是一句话而已，对这一句话的理

解层次，则完全不是这一句话能够覆盖的。比如说，一天写了两个 Hello Word 的程序员能领悟到 DRY，一位刚刚重构扔掉几千行重复代码的程序员也能领悟到 DRY，而这两个 DRY 所在的认识层面，必然是不一样的。再好比说我在“编程珠玑番外篇”这个系列里面写的有些文字，看上去很有道理，但笔者本人对这些文字的认识可能比我的读者要浅。即使有些牛人觉得上面这几本书的作者在某些原则上的认识不够深刻，或者觉得作者只是在罗列一些小碎片，但只要读这些书，特别是《Pragmatic Programmer》这本书的那些小 Tip，依然是有益的，因为他或许能触发你高于作者的思考，然后在你的脑中形成更加圆润的珠玑。而对于像我这样属于中手下游平时又没有大项目训练的人，《Pragmatic Programmer》这本书，和其他几本书一起，实在是很好的“小声音汇编”。

▶ Washington University Ph.D. candidate 徐宥

http://blog.youxu.info/