《程序员的思维修炼》

图书基本信息

书名：《程序员的思维修炼》

13位ISBN编号：9787115242334

10位ISBN编号：711524233X

出版时间：2010-12-10

出版社：人民邮电出版社

作者：Andy Hunt

页数：213

译者：崔康

《程序员的思维修炼》

前言

这是一本教你如何对大脑“编程”的书！运用一门程序设计语言编程对大多数普通程序员来说是“小菜一碟”，那么如何更上层楼成为一名专家级的软件开发者呢？本书给出了答案——优秀的学习能力和思考能力。作者从软件开发领域的角度，阐述了每一名程序员提升“内力”所需要的各种软性知识：从新手到专家的5个层次、人类大脑的运行机制、直觉和理性的利与弊、学习方法和实践经验的重要性、控制注意力的技巧，等等，可谓是一本程序员“素质教育”的微型百科全书。我非常支持一个白话版的“素质”定义：除了书本知识、硬性记忆以外的东西，扪心自问，包括我自己在内的大多数程序员除了固化的编程知识以外，又有多少“素质”拿得出手呢？IT领域知识更新换代之快需要我们不停地往前奔跑，当我们痛苦地追逐时尚的新鲜玩意时，更需放慢脚步，冷静地修炼自己的“内功”，以不变应万变，才能立于不败之地。如果你想改变现状，本书可以作为一个良好的起点。作者对各种软性技能都做了深入研究，并结合自己的经验总结成你可以借鉴的知识点，让你无需阅读各个领域（认知科学、神经学、行为理论）的专著，就能够汲取适合自己的精华。 在翻译本书时，我切实地感受到，虽然它文字不多、篇幅不大，但却内容丰富、引经据典，可见作者知识的渊博和写作的认真。我建议读者在阅读本书时，不要急于求成，要仔细地阅读各个章节，结合自己的日常经验体会文字背后的含义。对每一节中的“实践单元”，要立刻应用到日常工作中，观察和比较实践的前后效果，找出适合自己的行动指南！千里之行始于足下。请翻开本书的下一页，或许可以改变你的一生。

《程序员的思维修炼》

内容概要

本书解释了为什么软件开发是一种精神活动，思考如何解决问题，并就开发人员如何能更好地开发软件进行了评论。书中不仅给出了一些理论上的答案，同时提供了大量实践技术和窍门。
本书供各层次软件开发人员阅读。

《程序员的思维修炼》

作者简介

Andy Hunt 敏捷开发权威人士，敏捷宣言首倡者之一，著名IT图书出版公司Pragmatic Programmers创始人。除本书外，他还与人合著有多本获奖图书，深受读者欢迎，包括《高效程序员的45个习惯——敏捷开发修炼之道》、《程序员修炼之道——从小工到专家》等。

《程序员的思维修炼》

书籍目录

《程序员的思维修炼》

章节摘录

插图：

媒体关注与评论

“我一直在寻觅能够帮助我提高学习能力的书，到目前我还没有发现可与本书媲美的。Andy提供了最好的方法，帮助你成为专家级学者，指导你通过快速易行的思考方式提高工作效率。” ——Oscar Del Ben，软件开发人员“我把这本书推荐给了所有愿意听我唠叨的人。虽然各种关于科学学习的著作中都提到过一些思考和学习的方法，但本书的宝贵之处在于全面、精炼，并且更适合软件开发人员。” ——Paul V. Gestwicki博士，鲍尔州立大学本科部主任、教授“如果你是一名程序员，并且还没读过这本书，请马上就去买一本来仔细研读。就说你呢，还犹豫啥，我说的可是马上就去，晚了的话你可能就被人领先啦！” ——Gregory Long，《洛杉矶技术评论》

《程序员的思维修炼》

编辑推荐

《程序员的思维修炼:开发认知潜能的九堂课》：做一名程序员，就意味着要不断地学习，不但要学习新技术，还要学习怎么解决应用领域的问题，要了解用户社区的奇思妙想，要适应同事的怪癖，等等。在《程序员的思维修炼:开发认知潜能的九堂课》中。作者将带领大家共同经历一次有关认知科学、神经学、学习和行为理论的旅程，探索人类大脑令人惊奇的工作机制，并研究如何克服这一系统局限来提高自己的学习和思考技能。书中主要内容包括：成为一名专家级程序员的关键要素，大脑运行机制简介，如何正确使用和调试大脑。改进学习能力的具体技巧，如何通过自我引导积累经验。控制注意力的方法。为了让读者加深印象。作者还特别设立了一个“ 实践单元 ”，其中包括具体的练习和实验，旨在让读者真正掌握所学内容。生命中没有什么是一成不变的。人们需要改变自己的习惯和方法。不论你是程序员、软件公司管理者、技术奇人还是思想家。或者你只是想让自己的大脑更聪明一点儿，所有尝试改变自己的人，请把《程序员的思维修炼:开发认知潜能的九堂课》当作改变的开始……

《程序员的思维修炼》

精彩短评

《程序员的思维修炼》

精彩书评

《程序员的思维修炼》

章节试读

1、《程序员的思维修炼》的笔记-第4页

正能量：随着不断的成长和适应，人民需要改变自己的习惯和方法。生命中没有什么是永恒不变的，只有死鱼才会随波逐流，尝试改变自己。

2、《程序员的思维修炼》的笔记-第1页

http://www.5wpc.info/it/living/habit/2011/02/10/PragmaticThinkingAndLearning/

3、《程序员的思维修炼》的笔记-第14页

混了 11 年 IT，随着懂得东西越来越多，总结的也越多越多，反倒感觉到了自己的渺小和无知。就像奋斗了多年终于登上了一座高山，远远望去，却发现原来自己曾经居住过的村子，只是山下星星点点的众多小村落中的一个；而自己沾沾自喜的技能，每个村子都有水平相当的同行。
相对于整个世界来说，其实自己懂得的很少，而自己不懂的那么多；甚至，自己多年实践、思考而领悟到的东西，只不过是这个世界上早就存在的简单、朴素的客观真理。很多事情，虽然心里明白一两句话就能说的清楚道理，但能做到却很不容易。

这种渺小和无知感，让我恐慌：我是专家？还是新手？

我拿这个问题问 Dennis，他就推荐了《程序员的思维修炼》，说能解答我的问题。

到目前为止，只看完了其中有一章讨论了 Deryfus 模型的内容。在这一章中，作者讨论了从初学者到高级初学者，再到胜任者、精通者，最后到专家，一个人内在的能力和心态是如何“进化”的。简单总结下是 2 条：
1.根据具体场景分析应该做出什么反应，以及如何通过创造性的思维达成目标的能力；
2.能够通过听取、寻求他人意见，并通过观察和吸收掌握他人能力的能力。

对于“专家”，作者的观点类似于“以无法为有法，以无限为有限”，认为所谓的专家即通过直觉来做出反应的人——不是凭直觉把车开到沟里，而是凭直觉能理解什么叫“盐少许，中火翻炒几下”并最终能做出美味佳肴。而“专家通常很难把这种直觉和经验做出恰如其分的解释，因为他们的很多行为是如此的熟练以至于变成无意识的了。”

说的直白一些，就是所谓的专家，是在通过长期大量反复的实践、总结和思考/冥想以后，对某个专业领域事情的处理已经变得像是无需经过思考，而直接给出正确的解决方案——即使每次的事情都略有不同，甚至同一件事情在处理的过程中仍然在发生变化，他们都能从容不迫的随时根据“情景”的变化进行处理，并且整个过程如行云流水般，丝毫看不到也感觉不到他们有紧张感、焦虑或任何情绪的变化。

而新手，则是完全相反的：毫无经验，必须依靠指令和手册才能工作。

虽然计算机仍然是一门科学，而科学就应该是量化的、可反复准确重现的，似乎不应该允许这种“专家”的存在，也不应该允许靠“直觉”这种东西来处理问题。但我们生活在一个飞速变化的时代，除了流水线上的工人外，大多数工作本身都缺少精确重复的特性，总有些这样那样的不同，而各种细节变化的累加，足以使一件事情变成了另外一件事情。同时，做出决策所需的时间，与决策所起到的作用成反比。因为一切变化的如此之快，如果无法快速进行应对，可能等决策出来的时候面对的又是一

个新的问题了。这就需要用到专家们丰富的经验和直觉了。

同理，其他的任何一个领域，都存在类似的情况，当遇到一个新的路口，需要快速决定往哪边走时，就是专家们 show time。这或许就是为什么软件研发从 UP 时代向 Agile 时代转变，从重型流程转向轻量级的流程+重视“ 人 ”的作用和价值。

有兴趣的也可以看看 彼得.德鲁克 在《卓有成效的管理者》中对“ 知识型员工 ”的论述。

高手在民间，每个人都是自己领域的专家。

（未完待续）

4、《程序员的思维修炼》的笔记-第4页

切忌随波逐流
Only dead fish go with the flow.

不要盲目的听从任何建议，包括我的建议。你可以用开放的思维来阅读本书，尝试执行一些建议，再判断哪些对你有用。

5、《程序员的思维修炼》的笔记-第12页

事件理论可以被测量，这类理论可以被验证或证明。构建理论是无形的抽象，无法被证明。

6、《程序员的思维修炼》的笔记-第15页

由定义可知，新手在该技能领域经验很少或者根本没有经验。这里提到的经验，指的是通过实施这项技术促进了思维的改变。举个反例，可能一个开发人员声称拥有十年的经验，但实际上只是一年的经验重复了九次，那么这就不算是经验。

7、《程序员的思维修炼》的笔记-第5页

就

8、《程序员的思维修炼》的笔记-第2页

当前最重要的两项技能就是：
沟通能力
学习和思考能力敏捷方法实质是提高沟通能力。本书着眼于提高学习和思考能力，所以用了三章解析人脑的工作原理，一遍充分开发人脑潜能

9、《程序员的思维修炼》的笔记-第14页

专家们认识世界的方式，解决问题的方法，运用的思维模型等和普通人显著不同。

10、《程序员的思维修炼》的笔记-第12页

实际工作中，专家级职员并不总认为是专家，也没有拿到相应的薪水。
不是所有专家级职员都想成为管理者。

11、《程序员的思维修炼》的笔记-第11页

德雷福斯模型是所谓的构建理论。理论分为两种：事件理论和构建理论。这两种理论都用于解释我们观察到的现象。

事件理论可以被测量，这类理论可以被验证或证明。你能够判定某个事件理论的准确性。

构建理论是无形的抽象，无法被证明。构建理论是通过它的用处来衡量的。你无法判断某个构建理论准确与否。它是客观存在和抽象的结合体。就像苹果是存在，苹果是事物，存在即抽象。
在分析问题前，要弄清楚面对的是怎样的理论语境。

12、《程序员的思维修炼》的笔记-第1页

神经可塑性也意味着你能够学习的最大容量或者你可以获得的技能数量不是固定的。

13、《程序员的思维修炼》的笔记-第7页

应用你自己的实践经验、理解情境和利用直觉

14、《程序员的思维修炼》的笔记-第2页

软件产品处于应用、用户、规则和硬件（也就是平台）的合力之下。这些因素总在不断变化，迫使软件产品也随之改变。在我看来当前最重要的两项技能就是：#沟通能力 #学习和思考能力。

15、《程序员的思维修炼》的笔记-第1页

“ 非理性 ”没什么不对，思维过程是非理性或者不可重复的并不意味着它是不科学的、不负责任的、不合适的。

16、《程序员的思维修炼》的笔记-第7页

技能获取领域的德雷福斯模型（Dreyfus model）是研究如何超越新手层次，如何不断精通技术的有效方法。我们将会探讨德雷福斯模型，并特别关注成为一名专家的关键要素：应用你自己的实践经验、理解情境和利用直觉。

17、《程序员的思维修炼》的笔记-第1页

失败分两种。有一种失败对我们有益，可以从中学到东西。但是另一种对我们无益。第二种失败没有产生任何知识；它要么一开始阻止我们学习，要么中途毁了我们的学习。

18、《程序员的思维修炼》的笔记-第21页

误解德雷福斯模型会埋没专家的专业技能。事实上，专家的名声和业绩很容易遭到破坏。最后你只是在强迫他们遵循规则。

在德雷福斯的一项研究中，研究人员就是这样做的。他们邀请经验丰富的飞行员做实验，请他们给新手制定一套规则，要求代表他们的最佳实践做法。他们照做了，新手基于这些规则的确能够提高自己

的业绩。

然后，研究人员要求专家遵循自己制定的规则。

结果专家的表现明显不如以往。

这对软件的开发也会产生影响。考虑一下，任何对开发指定严格规则的方法或企业文化，会对团队里的专家产生什么影响呢？这将拖累其业绩表现下降到新手的水平。公司失去了他们所擅长的所有竞争优势。

但是，整个行业一直在试图通过这种方式“ 毁灭 ”专家。你可能会说，我们正试图训练赛马。但这不是获得良好的投资回报的办法，你需要让赛马自己去跑。

直觉是专家的工具，但公司往往轻视它，因为他们错误地认为，直觉“ 不科学 ”或者“ 不可重复 ”。因此，我们往往本末倒置，不倾听高薪酬高昂的专家们的意见。

相反，我们也往往喜欢使用新手，把他们扔在发展水平等级的最底层，让他们觉得未来遥不可及。在这种情况下你可能会说，我们正在试图赛羊。同样，这不是一个使用新手的有效方法。他们需要“ 被驾驭 ”，也就是说，明确方向，快速成功，等等。

但是，来自企业的压力从两个方面阻碍了我们。被误导了的政策公平思维要求我们同等对待所有开发人员，不论能力大小。这伤害了新手和专家（因为忽视了这样一个事实：根据不同的研究成果，开发人员之间存在20:1~40:1的生产力差异）。

新手使用规则，专家使用直觉。

19、《程序员的思维修炼》的笔记-第12页

下面列举了一些人们观察到的现象，适用于护理和软件开发，也可能适用于其他行业。

实际工作中，专家级职员并不总被认为是专家，也没有拿到相称的薪水。

不是所有专家级职员都想成为管理者。

职员的能力存在巨大的差异。

管理者的能力存在巨大的差异。

任何团队的成员在技术水平上可能各不相同，无法看作一个同质的可替代资源集合。

20、《程序员的思维修炼》的笔记-第14页

专家通常很难把他们的行为恰如其分地解释清楚，他们的很多行为是如此地熟练以至于已经变得无意识了。他们的大量经验都是通过大脑的非语言，无意识区域存储的，这让我们难以观察，而专家则难以表述。

而如果专家被强制遵从那些规则操作，他们的工作就会变得效率低下。
「经验战胜精确」的观点不敢苟同，为了写书把严谨的德国人黑的够呛。

21、《程序员的思维修炼》的笔记-第7页

成为一名专家的关键要素：应用你自己的实践经验，理解情景，利用直觉。

22、《程序员的思维修炼》的笔记-第1页

积极的情感对学习和创造性思维非常关键。处于高兴的状态可以扩展你的思维过程，激活更多的大脑物质。

23、《程序员的思维修炼》的笔记-第14页

例如就专家级大厨来说，他们徜徉于面粉和香料的缭绕之中，不必关心越堆越高的脏盘子（这些都留给实习生清洗），大厨只要努力琢磨、清楚表达如何做好这道菜。“来一点这个，那个少点——不要太多，然后开始烹饪直到完成。”

厨师长克劳德这样说不是故意卖关子，他知道“烹饪直到完成”的含义。他知道“刚好够”和“太多”之间的细微区别依赖于湿度、肉的来源以及蔬菜的新鲜程度。

专家通常很难把他们的行为恰如其分地解释清楚，他们的很多行为是如此地熟练以至于已经变成无意识的了。他们的大量经验都是通过大脑的非语言、无意识区域存储的，这让我们难以观察，而专家则难以表述。

当专家在做事时，我们其他人觉得十分神奇。神秘的魔法看起来似乎无处可寻，当我们甚至还不完全认识问题的时候，专家就已经凭借一种不可思议的能力知道了正确的答案。

当然，这不是魔法，只是他们认识世界的方式、解决问题的方法、运用的思维模型等都和普通人显著不同。

而一个新厨师在辛苦工作一天回到家里后，可能不会关心湿度和原料方面的细微差别。他只想知道食谱中到底需要放入多少藏红花（不仅仅只考虑藏红花特别昂贵这个因素）。

他想知道的是，如果已知肉的重量，如何精确设定烤肉箱的定时器时间，等等。这并不是说他迂腐或者愚蠢，只是他需要明确的、与情境无关的指令，便于参照执行。而如果专家被强制遵从那些规则操作，他们的工作就会变得效率低下。

24、《程序员的思维修炼》的笔记-第15页

早在20世纪70年代，德雷福斯兄弟（休伯特和斯图尔特）就开始研究人类如何获取和掌握技能。

阶段1 新手
新手非常在乎他们能否成功。新手不是特别想要学习，他们只是想实现一个立竿见影的目标。他们不知道如何应付错误，所以出错的时候，他们非常容易慌乱。

新手需要指令清单
Novices need recipes.

阶段2：高级新手
高级新手能够开始多多少少地摆脱固定的规则。他们可以独自尝试任务，但仍难以解决问题。
他们想要快速获取信息，不想在此刻寻根究底，或者重新温习一遍基础知识。

高级新手不想要全局思维
Advanced beginners don't want the big picture.

看不到这种联系，因为你层次还不够，只处于较低的技能水平。

阶段3：胜任者

与更高水平者追随下意识反应不同，胜任者会探寻和解决问题，他们的工作更多是基于谨慎的计划和过去的经验。如果没有更多的经验，在解决问题时，他们将难以确定关注哪些细节。

胜任者能够解决问题。
Competents can troubleshoot.

处于这一水平的人通常被认为“有主动性”和“足智多谋”。他们往往在团队中发挥领导作用（无论是否有正式的头衔）。他们是团队里的好人，既可以指导新手，也不会经常骚扰专家。

阶段4：精通者

精通水平的从业者需要全局思维。他们将围绕这个技术，寻找并想了解更大的概念框架。对于过于简单化的信息，他们会非常沮丧。

精通者能够自我纠正。
Proficient practitioners can self-correct.

在德雷福斯模型中，处于精通水平的从业人员有一项重大突破：他们能够纠正以往不好的工作表现。他们会反思以前是如何做的，并修改其做法，期望下一次表现得更好。

同时，他们会学习他人的经验。作为精通者，他能够阅读案例研究，倾听有关失败项目的流言蜚语。观察别人怎么做，从故事中认真学习，即使他没有亲自参与。

伴随向他人学习的能力而来的，是理解和运用格言经验之谈（maxim）的能力，这些经验之谈犹如谚语或格言，是可以应用于当前情境的基本原理。

精通者可以充分利用思考和反馈，这些都是敏捷开发的核心。相对早期阶段，这是一次巨大的飞跃。处于精通阶段的人更像是初级专家，而不是高级胜任者。

阶段5：专家

专家是各个领域知识和信息的主要来源。他们总是不断的寻找更好的方法和方式去做事。他们有丰富的经验，可以在恰当的情境中选取和应用这些经验。他们著书、写文章、做巡回演讲。他们是当代的巫师。

专家凭直觉工作
Experts work from intuition.

根据统计，专家的数量很少，大概占总人数的1%~5%。

专家知道哪些是无关紧要的细节，哪些是非常重要的细节。也许不是有意识的，但是专家知道应该关

注哪些细节，可以放心地忽略哪些细节。专家非常擅长有针对性的特征匹配。

25、《程序员的思维修炼》的笔记-第5页

一切都是互相关联的
Everything is interconnected.

在《第五项修炼》一书中，Peter Senge推广了系统思维（systems thinking）这个词语，描述了另外一种观察世界的方法。在系统思维中，人们试图将一个事物看作几个系统的连接点，而不是一个独立的个体。

始终关注情境。

26、《程序员的思维修炼》的笔记-第4页

不要盲目听从任何建议，包括我的建议。你可以用开放性的思想来阅读本书，尝试执行一些建议，再判断哪些对你有用。

Only dead fish go with the flow.

27、《程序员的思维修炼》的笔记-第11页

制造问题的思维方式无法用来解决问题。----阿尔伯特 爱因斯坦

28、《程序员的思维修炼》的笔记-第15页

29、《程序员的思维修炼》的笔记-第22页

当你在某领域不是很擅长时，你更可能认为自己是这方面的专家。
缺少准确的自我评估被称为二阶不胜任（second-order incompetence），也就是说，不知道自己不知道。（或者可以翻译成双重无能？）
这种情况在软件开发领域是个大问题，因为很多程序员和经理都意识不到有更好的方法和实践存在。我已经见过很多年轻的程序员（1～5年工作经验）从来没有做过一个成功的项目。他们已经彻底缴械投降，认为平常的项目就应该是痛苦和失败的。
达尔文说过：“ 无知往往来自于自信而不是知识。 ”

30、《程序员的思维修炼》的笔记-第1页

第二章：http://www.douban.com/note/220405130/
第三章：http://www.douban.com/note/220917408/
第四章：http://www.douban.com/note/224574702/
第五章：http://www.douban.com/note/230713218/

31、《程序员的思维修炼》的笔记-第2页

《程序员的思维修炼》

在我看来当前最重要的两项技能就是：
1、沟通能力
2、学习和思考能力

32、《程序员的思维修炼》的笔记-第1页

1. 德雷福斯技能获取模型的5个阶段：

* 新手：新手需要指令清单；
* 高级新手：高级新手不想要全局思维；
* 胜任者：胜任者能够解决问题 ；
* 精通者：精通者能够自我纠正；
* 专家：专家凭直觉工作。

1. 最重要的两项技能：

* 沟通能力；
* 学习和思考能力。

1. 你如何称呼以为专家级软件开发人员呢？巫师。（有点意思）

1. 当你在某个领域不是很熟悉时，你更可能认为自己是这方面的专家。

1. 达尔文说过：无知往往来自于自信而不是知识。

1. 大多数人都是高级新手。

1. 你想成为专家吗？大约需要投入十年的努力，不论哪个领域。当然你需要积极的实践：

* 需要一个明确定义的任务；

* 任务需要有适当难度——有挑战性但是可行；
* 任务环境可以有大量反馈，以便于你采取行动；
* 提供重复犯错和错误纠正的机会。

1. 大脑双CPU模式：

运用类比，我们可以这样说：大脑是双CPU，单主机总线设计。两个CPU提供了L型和R型处理模式。R型对直觉、问题解决和创造性非常重要；L型让你细致工作并实现目标。

大脑不是软件，软件不会老化。而大脑必须刷新，必须使用，否则就会失去记忆。

积极的情感对于学习和创造性思维非常关键，处于“高兴”的状态可以扩展你的思维过程，激活更多的大脑物质。

1. 大部分问题都有多个解决方案或者正确答案，唯一正确的答案可能只在小学算数里面才有。

1. 认知偏见：

曝光效应：我们往往因为非常熟悉某些事物而对它有所偏爱，比如不再好用甚至会出错的工具、技术或者方法。
霍桑效应：人们在知道自己正被审视时，往往会改变自己的行为。
“很少”不意味着“没有”：极其不可能的巧合事件其实每天都在发生。

1. 预期影响现实。

1. 技术本身并不重要，持续学习才是最重要的。

1. 尝试设立一些明确的小任务作为行动计划的一部分。我要按照目标指定一些小的任务来创建行动计划以实现该目标。

1. 像管理你的金融投资一样管理你的知识投资。

知识投资和金融投资的一个主要区别是所有知识投资都有些价值。即使你从来不会在工作中使用某项技术，它也会影响你思考和解决问题的方式。因此，你学习的任何东西都有价值，只是有可能不是直接的、有物质回报的或和当前工作相关的价值。也许它会有助于开发R型思维或者改善R型到L型的切

换。

1. 做笔记非常重要，即使你从来不阅读；

写文档的过程比文档本身重要；
学习某项事物最简单和有效的办法是尝试教别人；
利用wiki来管理知识。

1. 正确对待实践中的失败：“ 我不知道 ”是一个好答案，但不要就此止步。

1. 如果你的周围全是高技能的人，你就会增长自己的技能水平。一部分原因是来自于对他们实践和方法的观察和运用，还有一部分是来自于对自己大脑的调节，使其在更高水平上工作。

1. 要总是保持一个新手的头脑。你需要像小孩一样拥有无穷的好奇心，充满问题和惊讶。可能这种新编程语言真的很酷。或者另一种更新的语言是这样。或许我可以从这门新的面向对象的操作系统中学到知识，即使我从未准备用它。

33、《程序员的思维修炼》的笔记-第10页

德雷福斯模型是个好东西，能让我看清目前能力的现状。看第一遍的时候我还以为自己不是精通者至少也是介于胜任者和精通者之间。但反复映证之后还是有点难堪地承认：我只是处于胜任者罢了。但至少有个进步的方向了，这就有收获了啊~处于精通阶段的人更像是初级专家，而不是高级胜任者。

34、《程序员的思维修炼》的笔记-第1页

阿

35、《程序员的思维修炼》的笔记-第15页

这里提到的经验，指的是通过实施这项技术促进了思维的改变。
工作经验不只是量的积累

36、《程序员的思维修炼》的笔记-第1页

《程序员的思维修炼》

【第1章 绪论】
程序设计其实就是解决问题，它需要发明、创造和灵感。不论你从事什么职业，可能都需要创造性地去解决问题。然而，对于程序员来说，既要受到数字计算机系统的严格约束，又要展开丰富而灵活的人类思考，这就会展示二者的强大力量，又会深深地暴露二者的缺陷。

考虑到社会中各个相关团体的复杂交互影响和社会的持续变化，在我看来当前最重要的两项技能就是：
沟通能力
学习和思考能力

【2.1 新手与专家】
清晰表述专业技能十分困难。
It's hard to articulate expertise.
新手和专家有着根本区别，他们看待世界的方式不同，反应也不同。

【2.2 德雷福斯模型的5个阶段】
德雷福斯模型针对每项技能。
Dreyfus is applicable per skill.

【2.3 现实中的德雷福斯模型：赛马和赛羊】
诀窍2：新手使用规则，专家使用直觉。
从新手到专家的过程涉及的不仅仅是规则和直觉。在你提升技能水平的过程中，有许多方面会发生改变。最重要的三个变化如下。
从依赖规则向依赖直觉转变。

观念的改变，问题已不再是一个相关度等同的所有单元的集合体，而是一个完整和独特的整体，其中只有某些单元是相关的。

最后，从问题的旁观者转变为问题涉及的系统本身的一部分。

这是从新手到专家的转变，脱离独立和绝对化的规则，进入直觉的境界并（记得系统思考吗？）最终成为系统本身的一部分（参见图2-3）。

大多数人都是高级新手。
Most people are advanced beginners.

专家并不总是最好的老师。教学是一门技能，你在某个领域是专家，这并不能保证你可以把它教给别人。

【2.4 有效地使用德雷福斯模型】
而且需要辛勤工作——只是在某领域工作十年是不够的。你需要实践。根据著名认知科学家Dr. K. Anderson Ericsson的说法，积极的实践需要四个条件。
需要一个明确定义的任务。
任务需要有适当难度——有挑战性但可行。
任务环境可以提供大量反馈，以便于你采取行动。
提供重复犯错和纠正错误的机会。
稳步做这种实践十年，你就会达到目标。正
一旦你成了某个领域的专家，在别的领域成为专家就会变得更容易。至少你已经有了现成的获取知识

的技能和模型构建的能力。
诀窍4：通过观察和模仿来学习。
小号手Clark Terry曾经告诉学生们学习音乐的秘密是经历三个阶段：
模仿
吸收
创新
也就是说，首先模仿现有的做法，然后慢慢地吸收内在的知识和经验，最终将超越模仿阶段并能自主创新。这和被称为Shu Ha Ri的武术训练周期有异曲同工之妙。
在Shu阶段，学生模仿老师教授的技术，原模原样。在Ha阶段，学生必须思考其中的意义和目的，以达到更深的理解。Ri意味着超越，不再是一名学生，已经具有了自己的创新。

【2.5 警惕工具陷阱】
诀窍6：如果你需要创造力、直觉或者独创能力，避免使用形式方法。
不要屈服于工具或者模型的虚假权威。没有什么可以替代思考。

【2.7 日常的德雷福斯模型】
诀窍7：学习如何学习的技能。
要获取专业技能，需要做到如下几项：
培养更多的直觉。
认识到情境和观察情境模式的重要性。
更好地利用我们自己的经验。

【3.2 随时（24×7）记录想法】
诀窍8：捕获所有的想法以从中获益更多。
如果你不记录这些伟大的想法，你就不会意识到拥有过它们。
每个人——不论教育背景、经济状况如何，不论日常工作是什么，不论年龄大小——都有好想法。但是在这么多拥有好想法的人里面，只有少数人在努力跟踪它们。而其中，又只有更少数人会努力付诸行动。随后，仅有少之又少的人有能力将好想法成功实现。要想达到图3-3中金字塔的最顶层，必须跟踪好想法，这是最基本的要求。

【3.3 L型和R型的特征】
L型处理特点
L型处理令人感到舒适、熟悉而轻松。L型提供以下9种能力。
语言能力：使用词语来命名、描述和定义。
分析能力：有理有节分析事情。
符号能力：用符号表示事物。
抽象能力：抽取小部分信息（本质），并用其表示事物整体。
时间能力：遵时循序。
推理能力：基于理智和事实得到结论。
数字能力：使用数字计数。
逻辑能力：基于逻辑（定理、明确的论点）得出结论。
线性思维能力：按照关联、依序推演问题和思考，经常会得出收敛性结论。

R型是非语言的，它可以获取语言但是不能创建语言。它喜欢综合学习：集合事物形成整体。它总是如实地反应事物，从这一点来说，它非常具体实在，至少目前是。它使用类比来评价事物之间的关系。它喜欢听好听的，而且不愿意为守时而费心。它不受理性的约束，因为它不需要基于原因或者已知事实来处理输入——因而，它完全愿意暂时不作任何判断。

R型绝对是注重整体的，总是希望一次就能看到事物整体，感知整体的模式和结构。它具有空间性，

喜欢弄清楚事物之间的空间关系，部分如何形成整体。最重要的是，它是直觉性的、跳越性的思维，通常基于不完整的模式、直觉、感觉或者视觉影像来做判断。

“ 非理性 ”没有什么不对，思维过程是非理性或者不可重复的并不意味着它是不科学的、不负责任的、不合适的。

我们需要更多地使用R型，因为R型能够提供直觉，这是成为一名专家所迫切需要的。没有它，我们就不能成为专家。德雷福斯模型强调专家对隐性知识的依赖，这也属于R型的范畴。专家依赖观察和区分模式，这里也有模式匹配。

R型的类比和整体思考方式对软件架构和设计非常有价值，好的设计就是由这些组成的。

综合是一项非常强大的学习技术，以至于麻省理工学院媒体实验室的尼葛洛庞蒂在Don't Dissect the Frog, Build It [Neg94]中建议，真正想要了解一只青蛙，传统的解剖不是办法，更好的方式是构造一只青蛙。

诀窍9：综合学习与分析学习并重。

【3.4 R型的崛起】
诀窍10：争取好的设计，它真的很有效。

【3.5 R型看森林，L型看树木】
这项实验说明了一个事实：如果你想发现全局、整体的模式，你需要R型；如果你需要分析部分和细节，你需要L型。对于我们大多数人来说，这种层次的专长就是这样区别的。R型看森林，L型看树木。

【3.6 DIY脑部手术和神经可塑性】
诀窍11：重新连线大脑，坚信这一点并不断实践。

【第4章 利用右脑】
人应该努力学习洞察和培养自己内心深处的灵光一现，这远远胜于外面流光溢彩的整个世界。然而，人总会下意识地抛弃自己特有的想法，仅仅因为那是他自己的想法。
——拉尔夫·瓦尔多·爱默生（1803—1882），
美国散文家、思想家、诗人

我们容易忽略不寻常的或者感觉不舒服的想法，而这恰恰是很糟糕的事情。你丢弃的可能是一生中最有价值的想法。因此，你需要重视头脑中的所有想法。

【4.1 启动感观输入】
诀窍12：增加感观体验以促进大脑的使用。

【5.3 了解个性倾向】
百分之七十五的人偏于外向型，剩下百分之二十五的人则希望单独呆着。

感觉（S）与直觉（N）　你如何获取信息？在所有人格特质中，这条轴线可能最容易产生误传和误解。感觉型的人强调可行性和事实，完全基于当时的细节。直觉型的人非常富有想象力，喜欢比喻，创新力强，能够看到多种可能性——生活总是在下一个拐角等着我们。直

百分之七十五的人是感觉型的。

思考（T）与情感（F）　你如何做决定？思考型的人基于规则。情感型的人除了考虑适当的规则之外，还会评估个人和情感的影响。

判断（J）与知觉（P）　你的决定是封闭的还是开放的？即，你是快速做出判断还是持续感知？如果你非常喜欢早下定论，你就是J型。J型直到做出结论才会感觉舒服。P型则是会在做出决定后感到不安。两种类型在人口中大约各占一半。

性格类型的研究在考虑到人们之间的关系时最为有趣。强N型和强S型在一起工作时会产生摩擦。强J型和强P型或许就不应该一起来敲定一份时间表。事实如此。

人交往时请记住一个重要的背景信息：
别人的性格缺陷肯定与你不同。
诀窍22：尊重不同人的不同性格。
当你想与人争辩时，请想一想这点。

【5.5 现在我不知道该思考什么】
我们通过逻辑来证明，通过直觉去发现。
——庞加莱

【6.1 学习是什么……不是什么】
教育（Education）来自于拉丁文educare，字面意思是“被引出”，即引导出某样东西。我发现一件非常有趣的事情，当我们想到教育时，通常并不考虑它这个词源的含义——从学习者那里引导出一些东西。

【6.2 瞄准SMART目标】
使用SMART方法实现你的目标。
在这里，SMART代表具体的、可度量的、可实现的、相关的和时间可控的（Specific, Measurable, Achievable, Relevant, and Time-boxed）。对于任何目标（减肥、炒老板鱿鱼、征服世界等），你都需要制定一个计划，定出一系列帮助你实现目标的任务（objective）。每一个任务都应该具有SMART特性。

目标任务使你更靠近目标。
Objectives move you to your goal.

我们往往对于这两个词目标（goal）和目标任务（objective）的意思有一些模糊。明确地说：目标是一种理想状态，通常是短期的，是你努力要达到的状态。目标任务是一种帮你接近目标的事物。

相关的
这个目标真的与你有关吗——对你重要吗？你对此有热情吗？是在你控制之下的事情吗？
如果不是，这个目标就是不相关的。
目标需要相关，需要在你掌控之中。

时间可控的
这可能是目标最重要的一个特性。这意味着你需要设定一个最后期限。没有期限，目标会逐步衰退，永远被每天更紧急的事情所排挤。这样它永远都不会实现。
再强调一遍，稳扎稳打。采取循序渐进、比较细小的里程碑。当实现它们后，你会更有动力去实现下一个里程碑。

诀窍25：建立SMART任务实现你的目标。

【6.3 建立一个务实的投资计划】
知识投资和金融投资的一个主要区别是所有知识投资都有些价值。即使你从来不会在工作中使用某项技术，它也会影响你思考和解决问题的方式。因此，你学习的任何东西都有价值，只是有可能不是直接的、有物质回报的或和当前工作相关的价值。也许它会有助于开发R型思维或者改善R型到L型的切换。

【6.5 一起工作，一起学习】
成人教育的关键
成人学习者和儿童或大学生不同。马尔科姆·诺尔斯（Malcolm Knowles）在The Adult Learner: a Neglected Species [Kno90]一书中指出了成人的学习特点和学习环境。
如果学习能够满足成年人的兴趣和需求，他们就会主动学习。
学习的对象应该是与现实生活相关，而不是孤立的个体。
学习者主要使用经验分析法。
成年人需要自我引导，老师应该帮助他们互相交流。
老师必须允许风格、时间、地点和节奏的差异。

有多种选择来设立学习小组，既可以是非正式的也可以是正式的。对于非正式的，可以是大家共同选定阅读一本书，然后轮流让成员在wiki或者邮件列表里总结每章内容，或者聚在一起边吃午饭边讨论。

对于正式的，你可以采取下面几项谨慎的步骤。
　寻求建议：看看大家的想法。获取足够的提议，每个提议都要有拥护者。寻求广泛的主题：技术方面的、软技能方面的，包括还没有使用的技术或者希望使用的技术。
　选择一项提议和一个负责人：需要有人领导这个学习小组就某个专题进行学习。他们不需要擅长这个主题，但是必须对这个主题和学习充满热情。
　买书：公司为所有参与者买书。大多数出版商（包括Pragmatic Bookshelf）提供团购折扣，所以请务必注意。
　安排午餐会议：公司提供午餐或者大家自带午餐。应该用自己个人的时间来完成阅读，不过要安排午饭会议，准备一顿九十分钟的超长午餐。在会议上，安排前半个小时吃饭、社交和非正式的交谈。然后，正式开会。请一个人总结大家读完的一章。按不同主题或章节轮流总结，不要总是一个人。然后开始讨论这章：提出问题，提供意见。要想寻求灵感，你可以参考每章结尾的问题、任何明确的学习指导问题或者像本书里的实践单元。

诀窍28：组织学习小组学习和辅导。

尽量保持每一个小组不超过八到十个人。如果团队很大，可以将其分成多个更小的组织来讨论。
除了对学习本身有惊人的帮助，这还是增强团队凝聚力的一个好办法。大家一起学习，也可以互相学习，而且学得更有效。

【6.6 使用增强的学习法】
主动阅读和总结书面材料的更好方式
使用思维导图探索和发现模式和关系
以教代学

【6.7 使用SQ3R法主动阅读】
此时此刻，你正在阅读本书。一生中，你所读的书可能比听的讲座多很多。但是相比于任何由经验式的学习方法，阅读是一种效率最低的学习方法。

使阅读更有效的办法是更主动一点，而不是随便捡起一本书来开始埋头苦读。广为使用的好方法为数不少，我们来仔细研究其中一个，但是与其功效类似的方法还有很多。

这项学习一本书或其他印刷品的方法称为SQ3R，是该方法具体步骤的首字母缩写。
调查（Survey）：扫描目录和每章总结，得出总体看法。
问题（Question）：记录所有问题。
阅读（Read）：阅读全部内容。
复述（Recite）：总结，做笔记，用自己的话来描述。
回顾（Review）：重读，扩展笔记，与同事讨论。

你需要的不仅仅是笔记，你需要思维导图。

【6.8 使用思维导图】
现代思维导图是一种二维的、有机的整体大纲。建立思维导图的规则是松散的，但是大致步骤如下。
准备一张很大的无格白纸。
在纸中间写上标题，用圆圈框起来。
对于每一个主要的子标题，从圆圈引出线，添加标题。
重复执行其他层次的节点。
对于其他的单独事实或者想法，从合适的标题引出线，写上标题。

当你打算添加一条新信息、一个新想法或者领悟到思维导图时，你要面对这样一个问题：这属于哪一块？你必须评估想法之间的关系，不仅仅是想法本身，这是一项非常具有启迪作用的活动。

虽然很多不错的公司都制作了思维导图软件，但我认为软件工具只是更有利于协作和文档——而不是头脑风暴、学习和探索性思维。对于这些活动，我建议手动绘制思维导图。

无论是笔记还是思维导图，手写是关键。例如，听讲座时做笔记真的能帮助我记忆讲座内容——即使我再也没有看过这些笔记。

【7.1 为了学习而玩耍】
头脑风暴：儿童、计算机及充满活力的创意》（Mindstorms: Children, Computers, and Powerful Ideas）[Pap93]命名。

【7.2 利用现有知识】
波利亚写过一本非常具有影响力的书，详细介绍了解决问题的若干经典技术，并描述了具体步骤（How to Solve It: A New Aspect of Mathematical Method [PC85]，参见下面的概要）。
波利亚的解题方法

解决问题时，先提问自己。
未知量是什么？
已知量是什么？
条件是什么？
然后制定一个计划，执行之，回顾结果。波利亚建议的一些技巧（如下所示）听起来非常熟悉。
努力回想拥有相同或类似未知量的常见问题。
画一张图。
解决一个相关的或者更简单的问题，放宽限制，或者使用已知量的子集。
所有已知量和条件都用上了吗？如果没有，为什么？
尝试重新叙述这个问题。
尝试从未知量推到已知量。

波利亚的一个关键建议是寻找以前类似的解决方案：如果你解决不了这个问题，你知道如何解决类似的问题吗？也许相似点是完全一致的（比如“这就像我上周看到的bug”），或者是一种隐喻关系（比如“数据库的工作情况就像是一滩水”）。

诀窍34：从相似点中学习，从差异中忘却。

【7.3 正确对待实践中的失败】
探索就是在陌生的环境中“玩”。你需要自由地探索才能学习。但是，这种探索应该相对没有风险，因为你肯定不想因担心害怕而止住探索的脚步。你需要探索，即使你不知道走向何处。同样，你需要自由地创造——不介意自己的创造没有成果。最后，你需要在日常实践中应用你学到的东西。一种高效有益的学习环境应该允许你安全地做三件事情：探索、创造和应用。
诀窍35：在你的环境中安全地探索、创造和应用。

【8.1 提高注意力】
冥想的训练可以提高人的注意力。

如果你想在一天中更有效地支配你的“注意力资源”，那么就需要学习一些基本的冥想技巧。

寻找一个安静的地方，摆脱干扰或中断。这个可能是最难的一步。
舒适、清醒地坐着，挺直背。让你的身体放松下来，就像一个玩具娃娃。花点时间感受体内的任何紧张情绪，将其释放。
闭上眼睛，将注意力集中在呼吸——空气进入和离开你身体的这一点上。
注意呼吸节奏，吸气的长短和质量，吸气后屏气的短暂间歇，呼气的质量，呼气后屏气的短暂间歇。不要试图去改变它，只是感受。
将思维集中于呼吸。不要说话。不要描述你的呼吸或其他任何想法。不要与自己交谈。这是另一个困难的部分。
你可能会发现自己在思考一些问题或在与自己交谈。每当你注意力游荡开去，就要抛弃这些想法，轻轻将注意力回到呼吸上。
即使你的思维经常游荡，这个练习能使你发现自己的注意力在游荡，并且每次都能使自己回来，这对你是很有帮助的。

经过一段时间的练习后，你可以主动尝试控制自己的呼吸。分段呼吸的方法是，将呼吸看作空气经过三个独立的仓库：
腹部
胸部和胸腔
胸部的最上部和锁骨（但不包括喉咙）

【8.3 管理知识】
一旦你读过一次，记得去什么地方找到对应的细节就足够了。

爱因斯坦也深谙此道。据称他曾被问到一英里有多少英尺，他回答说，他不会在头脑中填满能轻易找到的东西。这就是参考书籍的用处，这是一种有效使用资源的方式。

将wiki作为基于文本的思维导图来使用。
Use a wiki as a text-based mind map.

可登录http://en.wikipedia. org/wiki/Personal_Wiki 查看最新的列表。
诀窍41：使用wiki来管理信息和知识。

有了wiki，当你有一个随意的想法后，可以把它写下来放在你的主页上，因为这时你不知道还能对它做些什么。一段时间以后，你有了第二个相关的想法，而现在你可以将这两个想法放在一起，存储在新的一页中。现在突然更多的相关想法出现了，因为你有一个地方来存放它，而你的思维也会非常乐意帮忙。
一旦你有了地方来存放某类想法，你就会得到更多这类想法。无论是wiki还是在纸上写的日志，也无论是便签还是鞋盒，对于特定主题领域或项目的相关想法，有一个地方来存放它们就是外部信息系统的主要优点。

将一些笔记从原来的形式抄写到wiki中（或整理到同一个wiki上），这有助于大脑吸收这些资料。就如同抄写会议或课堂上的笔记，这样做提供了第二次深入接触材料的机会，并能使大脑神经更强烈地感受这些信息。你越是接触它，越可能会发现原来你没有注意到的材料间的关系和模型。再次，你可以对一些较有意思的信息重构思维导图，以获得更深入的理解，并将其写回wiki。
你会更积极地寻找模式。

【9.3 超越专家】
在你变成专家之后，你最想追求的事情是……新手的思维。

新手的大脑有很多可能性，但是专家心里只有很少。
——铃木俊隆禅师

对于专家来说最致命的弱点是像专家一样行动。一旦你相信自己的专业水平，你就会对其他的可能性视而不见。你停止了好奇心。你可能开始抵制所属领域的改变，担心在你花费了很多努力才得以精通的主题上失去权威。你自己的判断和看法不再支持你，而是囚禁你。

要总是保持一个新手的头脑。你需要像小孩一样拥有无穷的好奇心，充满问题和惊讶。

处理学习方面的事情，不要先入为主，不要存在事先的判断或者固定的看法。要像小孩子一样看待事物的真实面貌。

37、《程序员的思维修炼》的笔记-第1页

程序设计其实就是解决问题，它需要发明、创造和灵感。不论你从事什么职业，可能都需要创造性地解决问题。然而，对于程序员来说，既要收到数字计算机系统的严格约束，又要展开丰富而灵活的人类思考，这就会展示二者的强大力量，又会深深地暴露二者的缺陷。

38、《程序员的思维修炼》的笔记-第15页

阶段1：新手，需要指令清单
阶段2：高级新手，缺乏全局思维
阶段3：胜任者，能够解决问题
阶段4：精通者，能够自我纠正
阶段5：专家，凭直觉工作，规则会断送专家

39、《程序员的思维修炼》的笔记-第3页

难题：团队内部和团队间的交流，甚至更困难的问题是完全陈旧的思想。没有任何项目是孤岛，软件不可能孤立地创建或者运行。
最重要的两个技能：

沟通能力。 这个能力，也许可以像销售人员学习。。
学习和思考能力。 这个嘛。。应该就是开发人员了。。

书里提到了敏捷方法。。以前不是太在意，现在开始有点在意了。。。

有这么一句话：很多敏捷思想和实践都是与良好的认知习惯相整合的。
改变。。。X。。这个词。。。和中国的《易经》里讲得一样，和《鬼谷子》《孙子兵法》里的思想。。一样。。。东西方哲学没想到又交轨了。。和PUA里的一些想法也一样。。这样。。正义与邪恶第一次完美的。。交峰啦。。 完全一SB样。。不过，世界可能就这样，万物背后的道理，难道真得相通吗？ 为什么我还没看出来，你看出来了吗。说说看。。
为什么会改变？力？

40、《程序员的思维修炼》的笔记-第11页

德雷福斯技能获取模型：从新手到专家的五个阶段。
雷福斯模型研究人类如何获取和掌握技能。

阶段：新手--高级新手--胜任者--精通者--专家

新手：照既存指令工作和学习的人

高级新手：开始摆脱固定规则，可以独自尝试任务，但仍然难以决绝难题

胜任者：能建立问题域的概念模型，独立解决问题，如果没有更多经验，在解决问题时，他们将难以确定关注哪些细节。

精通者：全局思维，寻找并想了解更大的概念框架。能够阅读案例研究，从中学习他人经验。拥有足够的经验，他们知道下一步会发生什么。

专家：知道哪些是无关紧要的细节，哪些是非常重要的细节，专家知道应该关注哪些细节，可以放心地忽略哪些细节。

41、《程序员的思维修炼》的笔记-第12页

提到了dreyfus模型，又提到了事件理论与构建理论，这些我统统不知道。
我感觉有戏。

42、《程序员的思维修炼》的笔记-第7页

直觉是专家的基本特征，事实上难以驾驭。

43、《程序员的思维修炼》的笔记-第20页

敏捷开发就是在一个高度协作的环境中，不断地使用反馈进行自我调节和完善。但是基于以往表现进行自我纠正，只在较高的技能水平上才能实现。

44、《程序员的思维修炼》的笔记-第3页

软件并不是在集成开发环境（IDE）或其他工具上设计出来的，它是在我们的大脑中想象和创造出

来的。思想和概念是需要在团队（也包括付钱让我们开发软件的人）中分享和交流的。 我们已经在改进基础技术--程序设计语言，工具，方法上花费了很多时间，当然这也是十分必要的，但现在是我们更进一步的时候了。

PS: 大脑->vim->IDE
实践性体系越来越强了。。。从以前的小工到专家的,shell游戏，自动化工具，到这里的大脑。。思想。。 跟着专家走。。没错。。哈哈。

45、《程序员的思维修炼》的笔记-第1页

德雷福斯(Dreyfus)技能获取模型：
阶段1:新手 --需要规则，需要指令清单
阶段2:高级新手 --高级新手不想要全局思维
阶段3:胜任者 --会探寻和解决问题
阶段4:精通者 --全局思维，自我纠正
阶段5:专家 --凭直觉工作

新手使用规则，专家使用直觉

高级新手的人群所点比例最大

二阶不胜任：不知道自己不知道多少

通过观察和模仿来学习。

如果你需要创造力，直觉或独创能力，避免使用形式方法。

脑模式：
R(富模式)型对直觉、问题解决和创造性非常重要，L(线性模式)型让你细致工作并实现目标。

24X7记录想法，捕获所有的想法以从中获益更多

L型处理特点：
提供9种能力：语言，分析，符号，抽象，推理，数字，逻辑，线性思维 能力

R型处理特点：
非线性的，非语言，非理性，综合，空间性，具体，直觉，分析，全面...

综合学习与分析学习并重.

重新连线大脑，坚信这一点并不断实践。

增加感观体验以促进
写文章，写博客，晨间写作

建立SMART任务实现你的目标：
具体的，可度量的，可实现的，相关的，时间可控的

管理你的知识投资（对主动学习的投资做好计划）
1，制定具体计划
2，多样性（多领域学习）
3，主动的，而不是被动的投资
4，定期投资（成本平均法）

使用原生学习模式：（视觉型，听觉型，动觉型）

使用SQ3R法主动阅读：（阅读是一种效率最低的学习方法）
调查(survey):扫描目录和每章总结，得出总体看法
问题(question)：记录所有问题
阅读(read)：阅读全部内容
复述(recite)：总结，做笔记，用自己的话来描述
回顾(review)：重读，扩展笔记，与同事讨论。

使用思维导图

通过SQ3R法使用思维导图
使用思维导图理清思路

在玩中学习，思维导图你玩的越多，效果就越好..

在探索中学习

最后期限会使大脑恐惧（当面对时间压力时，人没有创造力）
面对压力时，需要放松
允许失败，你会走向成功（建立一种失败代价接近零的环境）

很多感知是基于预测的,预测则基于情境和过去的经验。

让大脑为成功形成惯例。

冥想可以提高注意力

管理知识：
开发外部信息处理系统：使用wiki
避免同时处理多个任务，避免分心

有效改变：

制定计划
“ 不作为 ” 是敌人，而 “ 错误 ” 不是
给新习惯适应的时间
信念是真实的
采取步步为营的细小步骤

46、《程序员的思维修炼》的笔记-第3页

实用主义pragmatism的本质就是做对你有用的事情。

47、《程序员的思维修炼》的笔记-第22页

当你在某领域不是很擅长时，你更可能认为自己是这方面的专家。

在文章 “ Unskilled and Unaware of It: How Difficulties in Recognizing One's Own Incompetence Lead to Inflated Self-Assessments ” 中，心理学家Kruger和Dunning讲述了一个自以为是的小偷，他在光天化日之下抢劫银行。他不相信自己这么快就被捕了，因为他以为在脸上涂满柠檬汁，摄像头就监视不到他。

“ 柠檬汁人 ” 从来没有怀疑他自己的假设。缺少准确的自我评估被称为二阶不胜任（second-order incompetence），也就是说，不知道自己不知道。

这种情况在软件开发领域是个大问题，因为很多程序员和经理都意识不到有更好的方法和实践存在。我已经见过很多年轻的程序员（1~5年经验）从来没有做过一个成功的项目。他们已经彻底缴械投降了，认为平常的项目就应该是痛苦和失败的。

达尔文说过：“ 无知往往来自于自信而不是知识。 ”

反过来似乎也是对的。一旦你真的成为了一名专家，你会痛苦地意识到你知道的是多么少。

48、《程序员的思维修炼》的笔记-第5页

诀窍一 始终关注情境

49、《程序员的思维修炼》的笔记-第8页

不要刻意地拼命实践，过犹不及。我们将研究如何利用反馈，乐趣和失败来创造更有效的学习环境，关注设定最后期限的危害，并体会如何通过自我引导积累经验。

我们还将探讨为何需要分散一些注意力，以便更好地聚焦于思维浸泡之中，并以更积极的方式管理你的知识。

《程序员的思维修炼》

版权说明