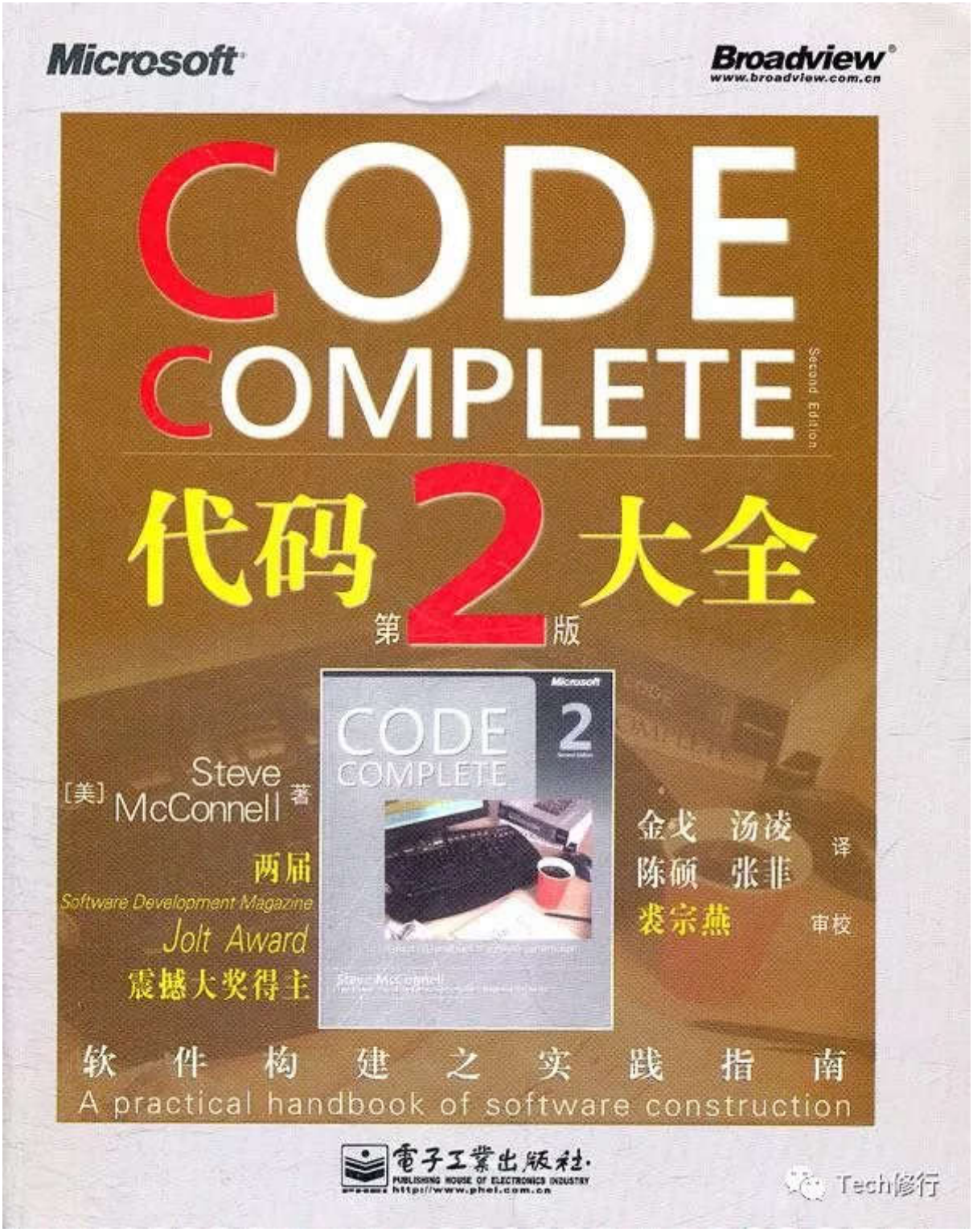
12本程序员必读书籍，从菜鸟到大神

2017-12-25 Tech修行

那些刚开启软件工程生涯的同事和朋友，为了要成为一个更好的开发人员，经常会问应该读哪些书？真的需要读书吗？这是一个很值得探讨的问题，而且也是我在成为软件工程师之时问了很多导师的一个问题。但问题是，很多人推荐的是不同主题的不同书籍。他们推荐的书在他们看来是伟大的，但没有人能告诉我，要想成为一个伟大的工程师，我应该阅读什么，哪些是重要的、是必读的书籍。

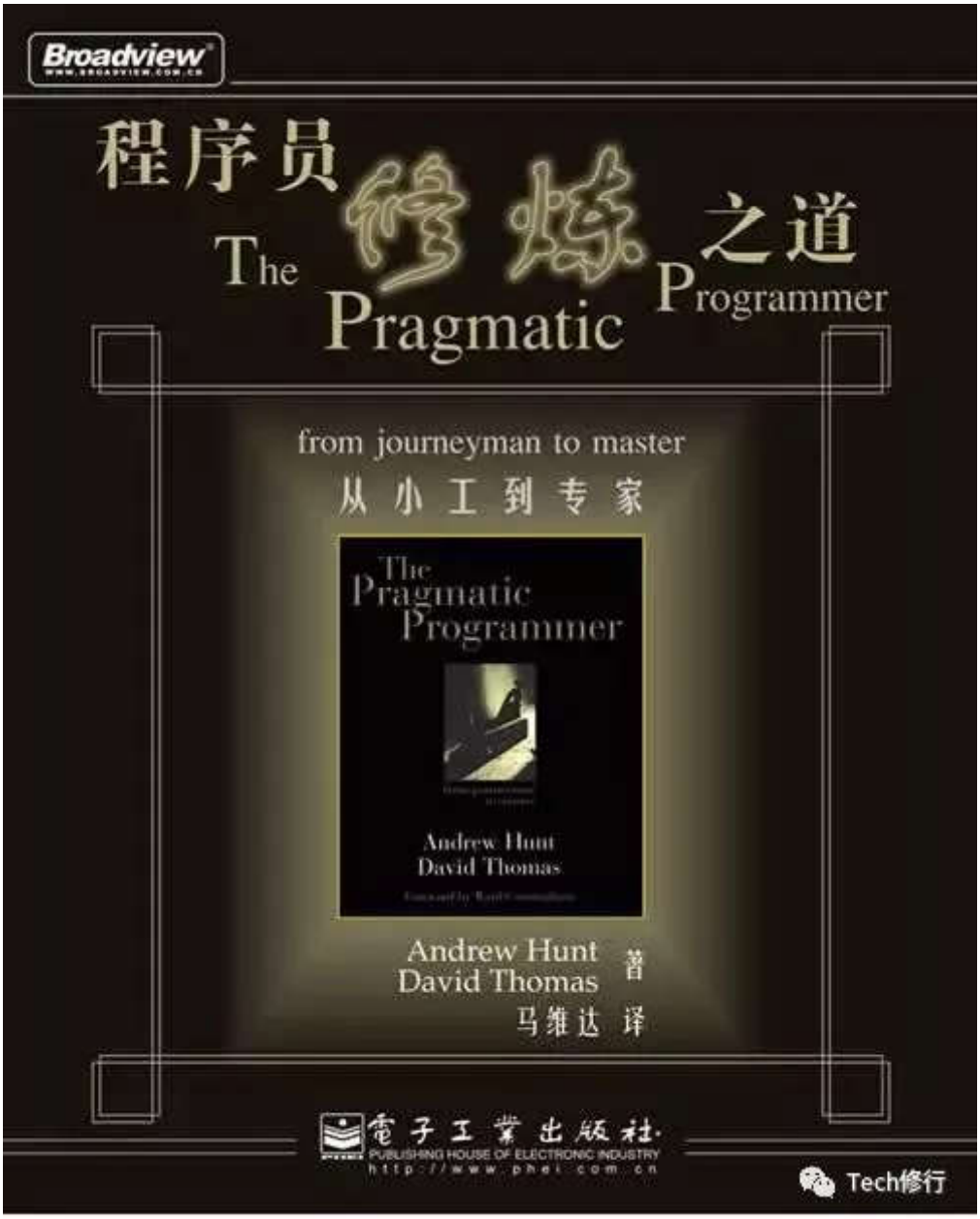
以下是国外大牛整理的一些经典书籍，阅读这些书可以帮助你避免一些常见的陷阱和错误，一些开发人员早期经历的陷阱和犯过的错误。我多么希望在我刚进入软件领域的时候，就有人向我推荐这些书啊，并且我现在依然很庆幸自己发现并反复阅读了这些书！也许你已经在大学读计算机科学或工程课程的时候读到过其中的一些书。也许在那个时候，你觉得它们并不重要，但是我可以用我亲身经历来说明我使用和应用了许多来自于这些书的原则。

1.《Code Complete 2（代码大全 2）》



《Code Complete 2》就是它了！这就是我认为首屈一指值得阅读的书（请恕我妄言），如果你要成为一个优秀的软件工程师的话。它被广泛认为是最好的实用性编程指南之一，Steve McConnell最初的《Code Complete》在过去的10多年时间里，一直在帮助开发人员编写更好的软件。现在，这部经典书籍已全面更新，修改成了前沿的实践方法——以及数以百计的新的代码示例——修订了软件结构的艺术和科学。从研究、学术界和日常商业实践中捕获知识体系，McConnell将最有效的技术和必须知道的原则总结成清晰又务实的指导。无论你的经验水平，开发环境，还有项目规模如何，这本书都可以启迪和激发你的思考，帮助你打造最优质的代码。

2.《Pragmatic Programmer（程序员修炼之道）》

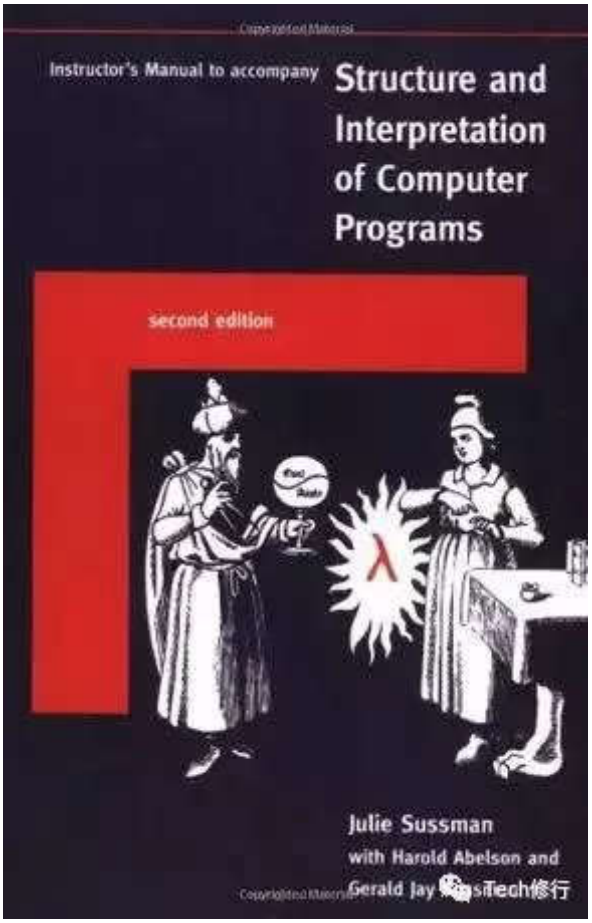


《Pragmatic Programmer》这是我最先阅读的编程书籍之一。是我的一个朋友在我就职第一份专业工作的时候推荐给我的。我很庆幸他这么做了。尽管这本书写于1999年，但是它的概念是我们以一种务实的态度去开发复杂系统的基础。程序员也是工匠，他们被训练使用一组特定的工具（编辑器，对象管理，版本跟踪器）生成某种可在一定环境中（硬件组件上的操作系统）工作的产品（程序）。和任何其他工艺一样，计算机编程也孕育出了智慧，但其中的大多数智慧是不能从大学或认证课程中学到的。大多数程序员只能通过独立的试验，时间一点点的积累，才能掌握这些所谓的技巧。在《Pragmatic Programmer》一书中，Andrew Hunt和David Thomas编纂了很多他们在分别作为软件设计者和代码编写者的职业生涯中发现的真理。

作者的一些实用性建议非常具体，而且显然很易于实施。他们建议读者去学习，例如一个文本编辑器，然后在各种情况下使用它。他们还建议使用版本跟踪软件——即使是对最小型的项目，学习正则表达式语法和文本操作语言。书中还有其他许多也非常有价值的建议。在调试部分，作者指出：“如果你看到蹄印的话，应该考虑马这个范围，而不是斑马。” 也就是说，要怀疑一切，

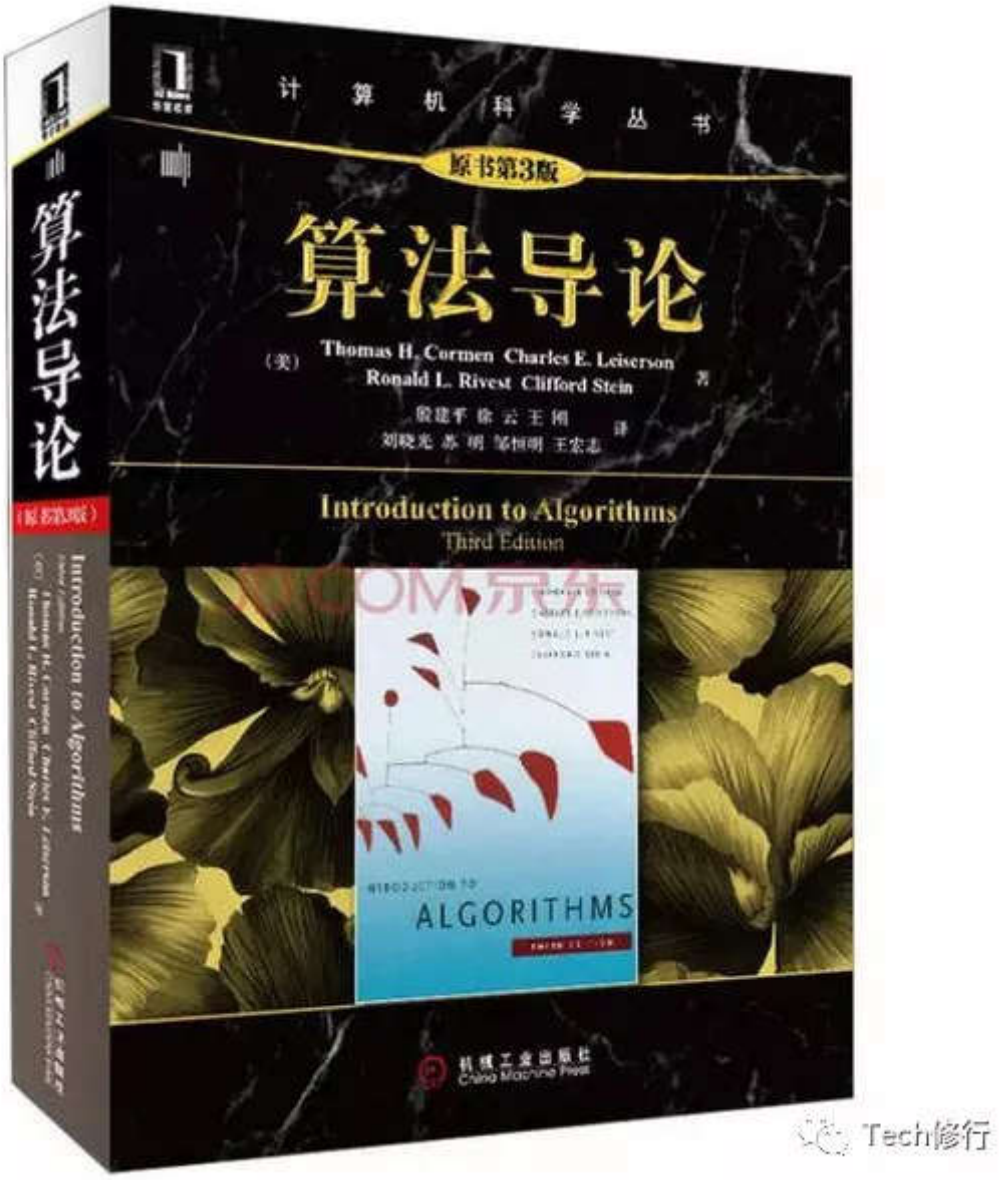
然后从最显眼的地方寻找问题。还有关于预估时间和费用，以及将集成测试纳入到开发进程的建议。《Pragmatic Programmer》让人爱不释手的还有两个原因：它会更清晰地梳理你自己积累的智慧，它还会给你介绍你还没考虑到的工作方法。

3.《Structure and Interpretation of Computer Programs》



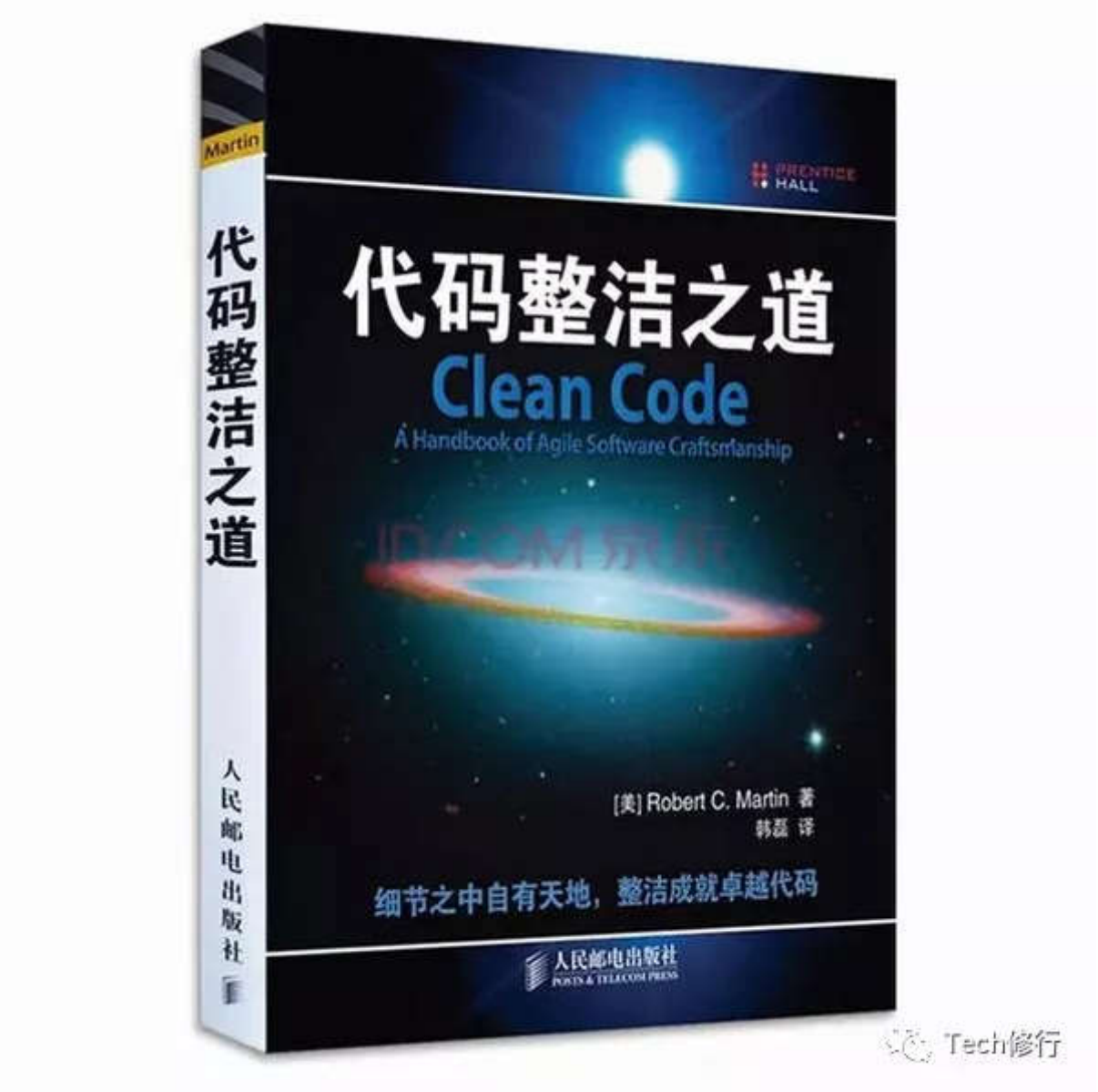
《Structure and Interpretation of Computer Programs》 以一种对解决问题和编程技术分析和严谨的态度，这本书面向于工程。《Structure and Interpretation of Computer Programs》强调通过不同方式来发挥核心作用，以处理计算模型中的时间。其独特的方式使得它非常适合于计算机科学课程，以及编程语言和程序设计的入门。这本书进一步解释了4个最著名的编程语言范式——命令式编程，面向对象编程，基于逻辑编程和应用性编程。

4.《Introduction to Algorithms（算法导论）》



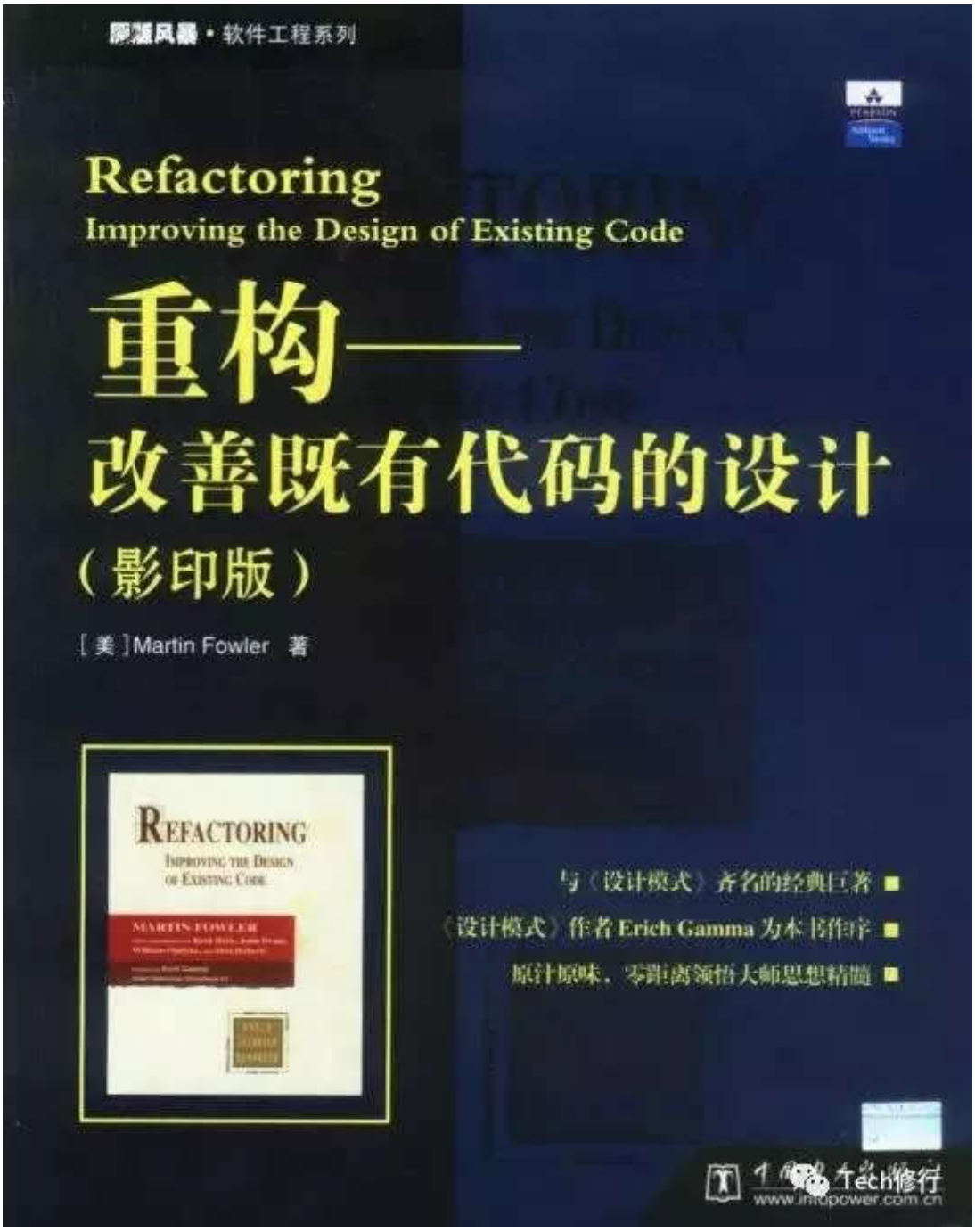
《Introduction to Algorithms》这必须是理解和使用算法（将在软件开发中涉及很多）的最佳书籍。有些关于算法的书虽然严谨，但不全面——还有一些虽然涉及面广，但却缺乏严谨性。《Introduction to Algorithms》独特地结合了严谨性和全面性。这本书涵盖了广泛又深入的算法，使得书中的设计和分析能够面向所有层次的阅读人员。每个章节相对独立，可看作是一个学习单元。算法用英语和可读的伪代码描述。它使用初级基本的解释，并不牺牲覆盖的深度和材料的严谨性。第一版不仅广泛成为了世界各地高校的教材，还成为了专业人士的标准参考书。第二版新增了算法，概率分析，随机算法，线性规划的章节。

5.《Clean Code（代码整洁之道）》



《Clean Code》，作者Robert C. Martin，分为三个部分。第一部分介绍原则、模式和编写干净代码的实践方法。第二部分包括若干个复杂度渐进的研究案例。每个案例研究就是一个清洁代码的练习，也是通过解决代码库中的一些问题让代码变得健全、高效的练习。第三部分是决定性的一个部分：每个单独的章节在创建案例研究的时候包含了一系列启发式的教学法。最后得到了描述我们在编写、阅读和清理代码时的思考方式的知识库。

6.《Refactoring（重构）》



《Refactoring》Martin Fowler写的《Refactoring》主要关于改进现有代码的设计。这是一个改变软件系统而不改变代码的外部行为，却能提高它内部结构的过程。通过重构，你甚至可以将一个糟糕的设计重新制作为一个很好的设计。这本书对重构原则进行了深入探讨，包括在哪里发现重构的机会，以及如何建立所需的测试。另外还有一个目录有40多个已经用细节证明的重构，这些细节包括何时以及为什么要使用重构，逐步说明如何实现重构，并举例说明重构是如何工作的。这本书用Java作为其主要语言而写，但其中的思路适用于任何OO语言。

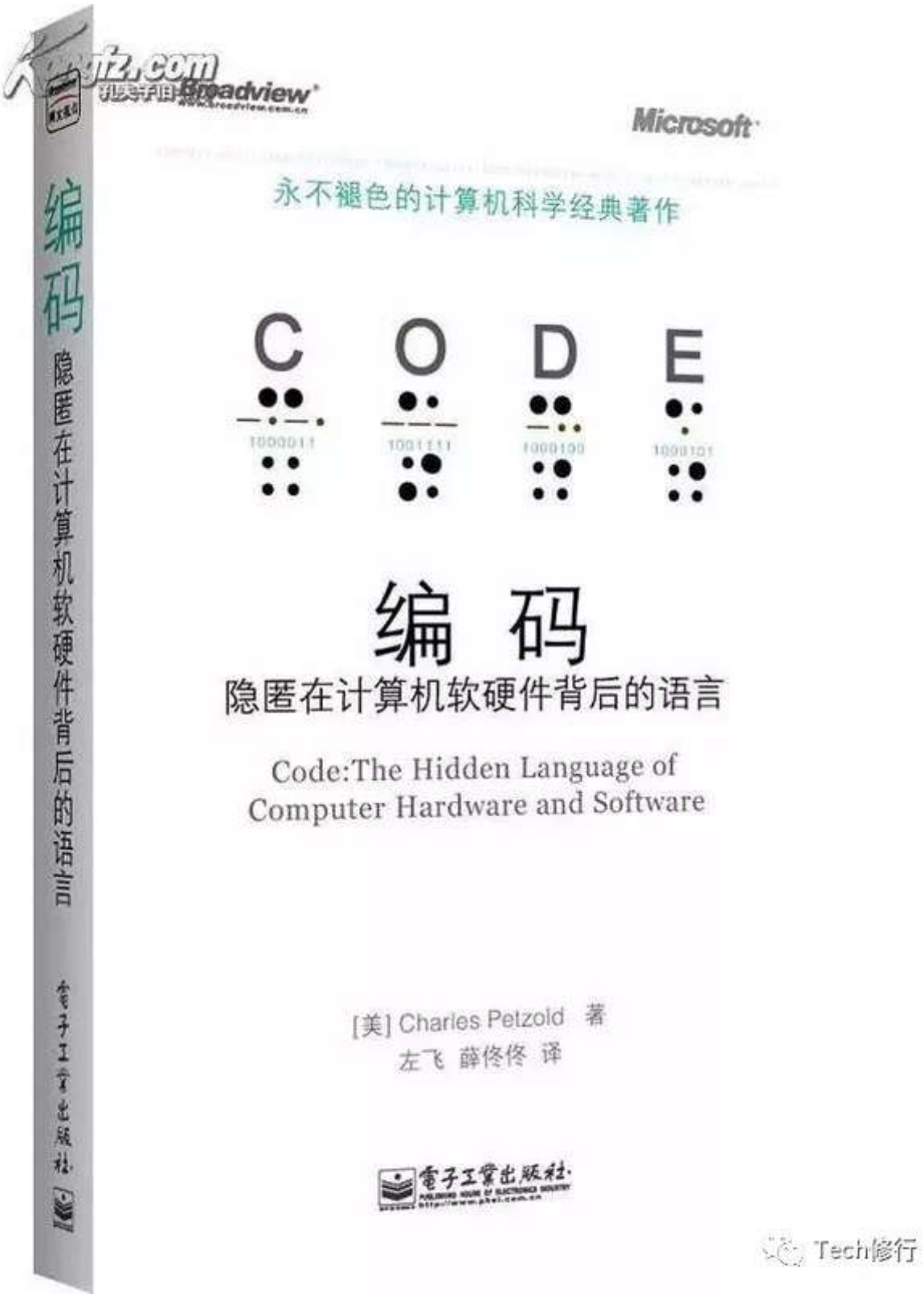
7.《The Art of Computer Programming（计算机程序设计艺术）》

《The Art of Computer Programming》这又是一部经典之作。由著名的计算机科学家教授Donald Knuth编著，并得到行业内众多顶尖程序员的一致好评。甚至连Bill Gates也对这本书赞誉有加：

“如果你认为你是一个真正优秀的程序员……那么就去读[Knuth的]《The Art of Computer Programming》吧……如果你能读懂整本书，那么请一定要给我发简历。”

这本书以基本的编程概念和技术开头，然后聚焦于更具体的信息结构——计算机内的信息表示，数据元素之间的结构关系，以及如何有效地处理这些问题。此外还提供了基本的应用程序给仿真模拟，数值方法，符号计算，软件和系统设计。

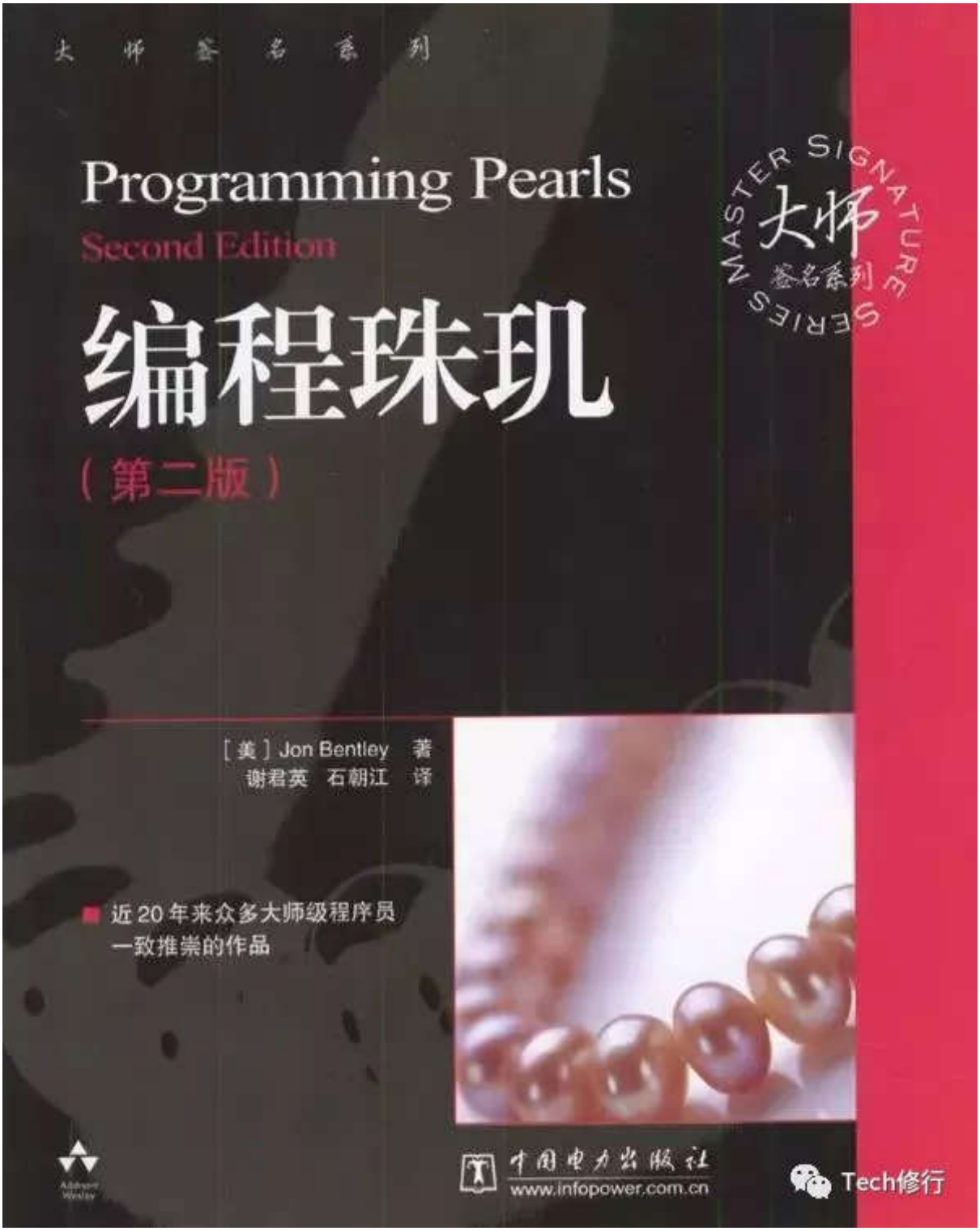
8.《CODE: The Hidden Language of Computer Hardware and Software（编码：隐匿在计算机软硬件背后的语言）》



《CODE: The Hidden Language of Computer Hardware and Software》这本书清理了创建和开发复杂系统的大量“魔法”。现在的抽象是如此之多，以致于一些低层次的细节反而不为开发人员所知。虽然你可能不会发现自己在实践中时刻使用着这本书——但我相信，知道自己正在构建什么以及整个编排的工作原理是什么，总归是一个好主意。当你需要打开“黑匣子”，深入软件或硬件来解决一个讨厌的bug的时候，它就能派上用场了。Charles Petzold写的《CODE: The Hidden Language of Computer Hardware and Software》梳理了许多编程概念——从数字系

统的十进制，八进制，二进制到高级语言。这本书介绍了基于包的通信协议和TCP。许多章节讲解了有关硬件的概念，有五个章节涉及到了软件和教导操作系统，浮点运算和图形用户界面。

9.《Programming Pearls 第二版（编程珠玑）》

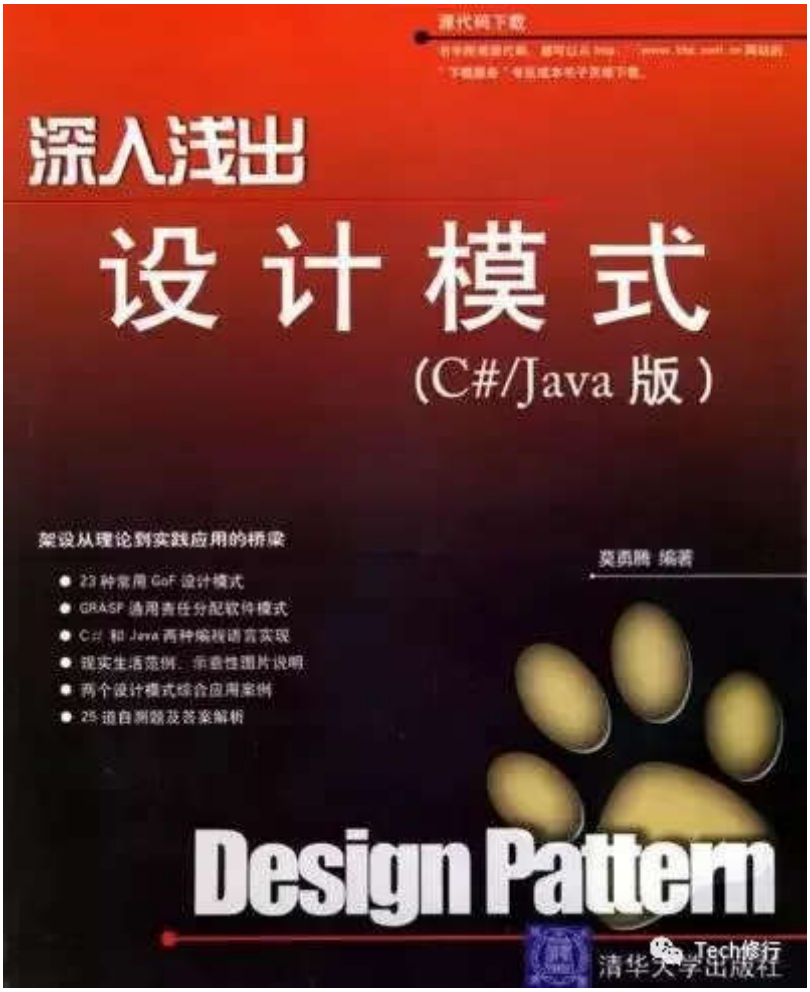


《Programming Pearls 第二版》这本书和这书单上的其他书籍略有不同。我想说这本书可以帮助一个人“像一个程序员一样思考”。《Programming Pearls》是先前发表在“Communications of the ACM（美国计算机学会通讯）”的15篇专栏的一个纲要。这些专栏涵盖了广泛与编程相关的主题：从需求收集到性能优化。重点关注编码技术和算法。

每个专栏被组织为一个章节。章节通常以一个实际问题的情景呈现开头。然后，提出各种解决方案和相应的经验教训。写作风格清晰明快。

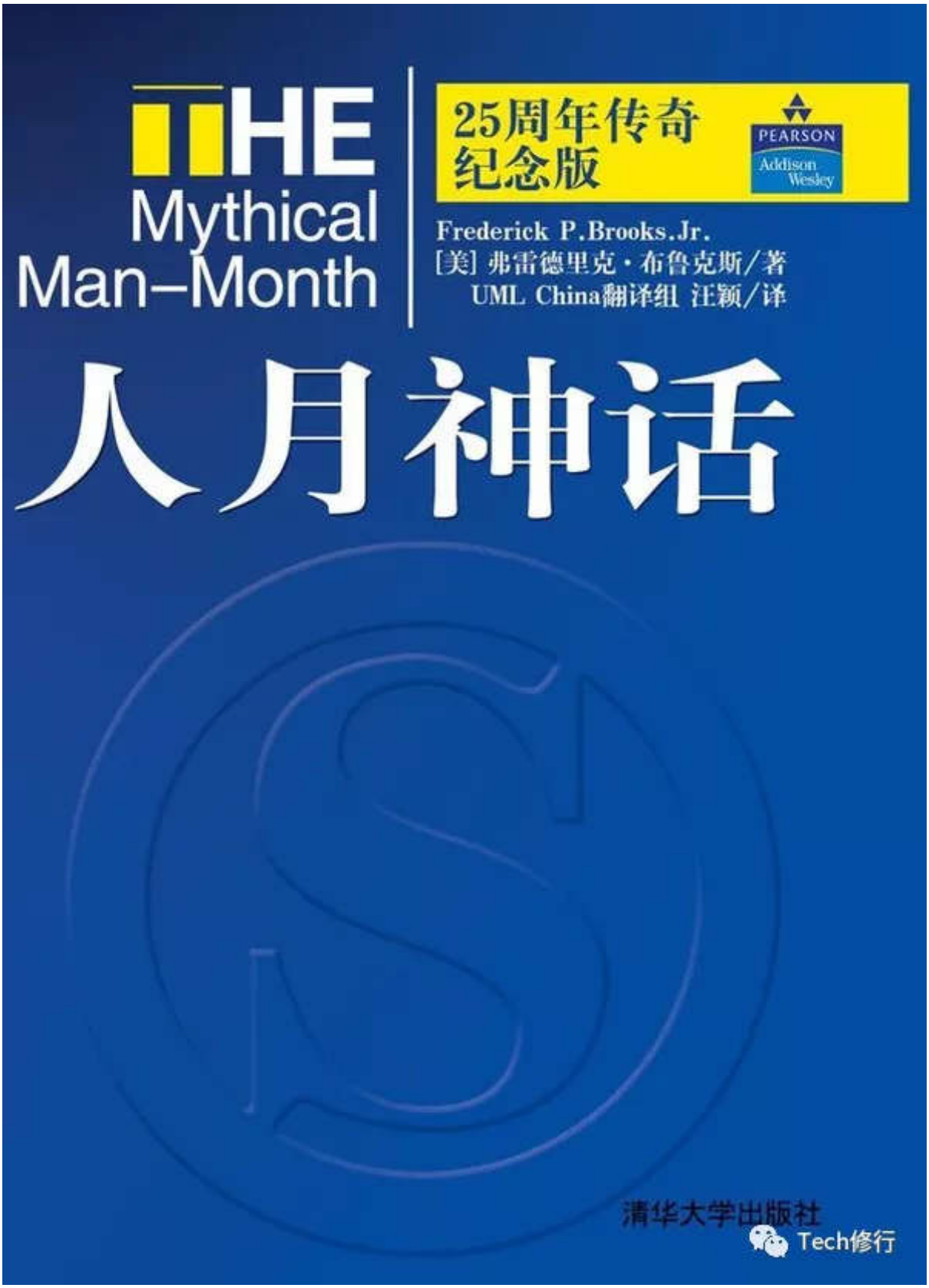
《Programming Pearls》不是教授新编程概念的寻常书籍。虽然它包含了优秀，有时甚至是相当新奇的想法，但这本书的目的不是教你一些新的东西，而是帮助你成为一个更好的问题解决者。

10.《Design Patterns（深入浅出设计模式）》



《Design Patterns》如果你打算成为一名架构师或系统的设计人员，那么你很有可能会被要求阅读这本书。这是一本被誉为有史以来最伟大的有关于软件开发的书，详细讲述了许多不同的设计模式，这么多年来一直在帮助软件工程师避免和处理行业面临的常见问题。遵照这本书的策略可以助你打造更高品质，灵活和可维护的软件。传说中的“四人帮”就是这本书，因为它是由四个著名的作者共同编写的。

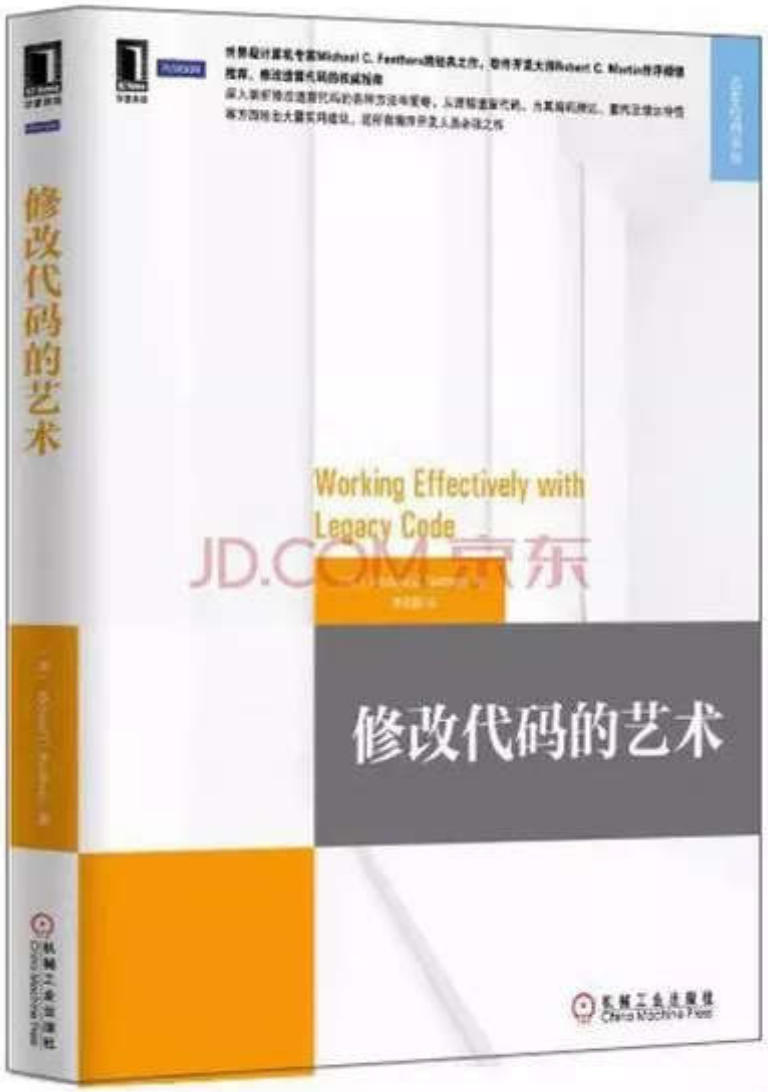
11.《The Mythical Man-Month（人月神话）》



《The Mythical Man-Month》本书是一本经典之作，但最近被修改和更正了。令人惊奇的是这本书仍然与软件产品开发密切相关。如果你从事软件的话，这本书是必读的。这本书最有价值的部分，我相信，是“plan to throw out”原型章节。尽管我们的目标通常是做一个更大，更好，更快，不管它是什么的东西，但是我们总会构建出一些不得不被废弃，需要重做的东西。这种情况我亲身经历过很多很多次。因此，关键是要plan to throw out（计划抛弃），这样你才能适应

接下来的情况。如果你梦想第一个产品就ok，那么你就会有抛弃它们的风险，因为产品的改进和发展是不可避免的。计划抛弃也有助于通过设置合理的里程碑来达到进度目标。

12.《Working Effectively with Legacy Code（代码修改的艺术）》



《Working Effectively with Legacy Code》我之所以喜欢这本书，是因为几乎所有的软件开发人员，在其职业生涯的某个时刻，往往会不得不支持和工作于遗留系统。在这本书中，Michael Feathers提供了从开始到结束的策略，以便于更有效地工作于未经测试的遗留代码库。本书借鉴了Michael为其著名的Object Mentor专题研讨会创作的材料：Michael用于指导的技术，以帮助开发人员，技术管理人员和测试人员掌控遗留系统。