HZ BOOKS 华章科技
Mc Graw Hill Education

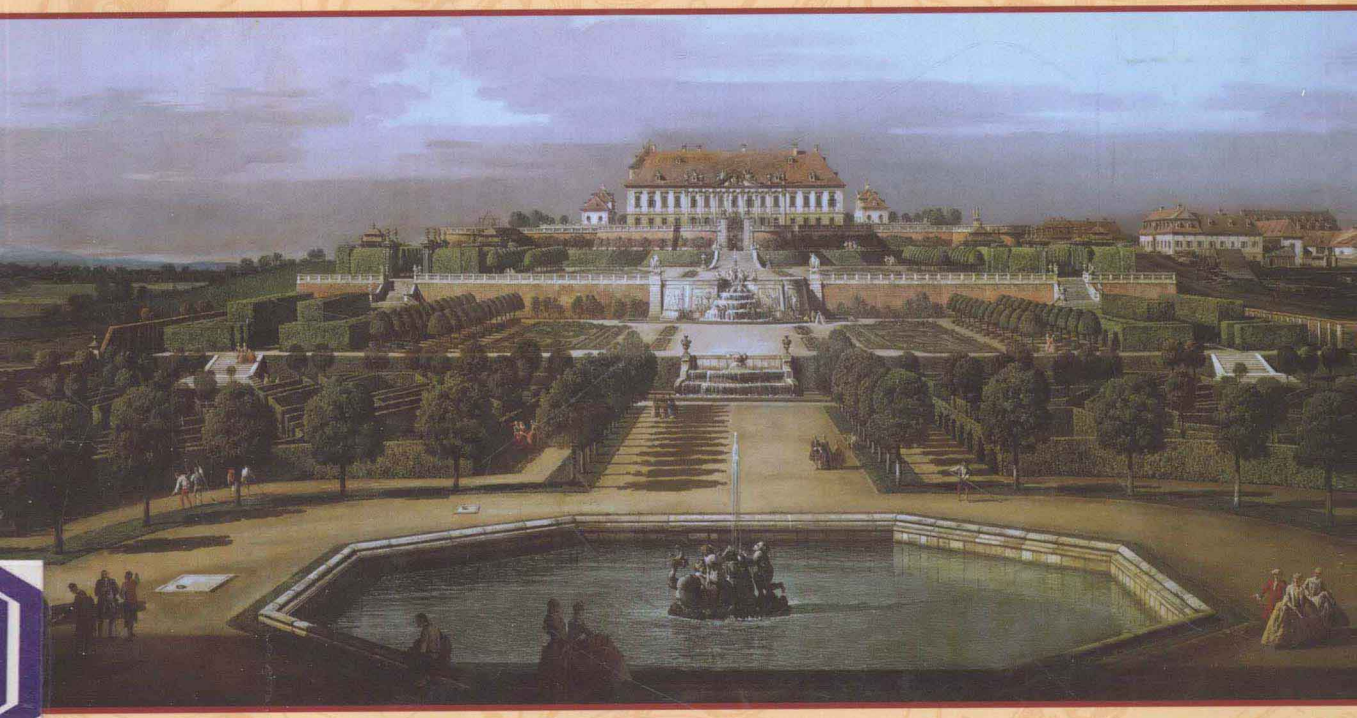
软件工程技术丛书

软件工程最佳实践

Software Engineering Best Practices

Lessons from Successful Projects in the Top Companies

（美）Capers Jones 著　吴舜贤 杨传辉 韩生亮 译



机械工业出版社
China Machine Press

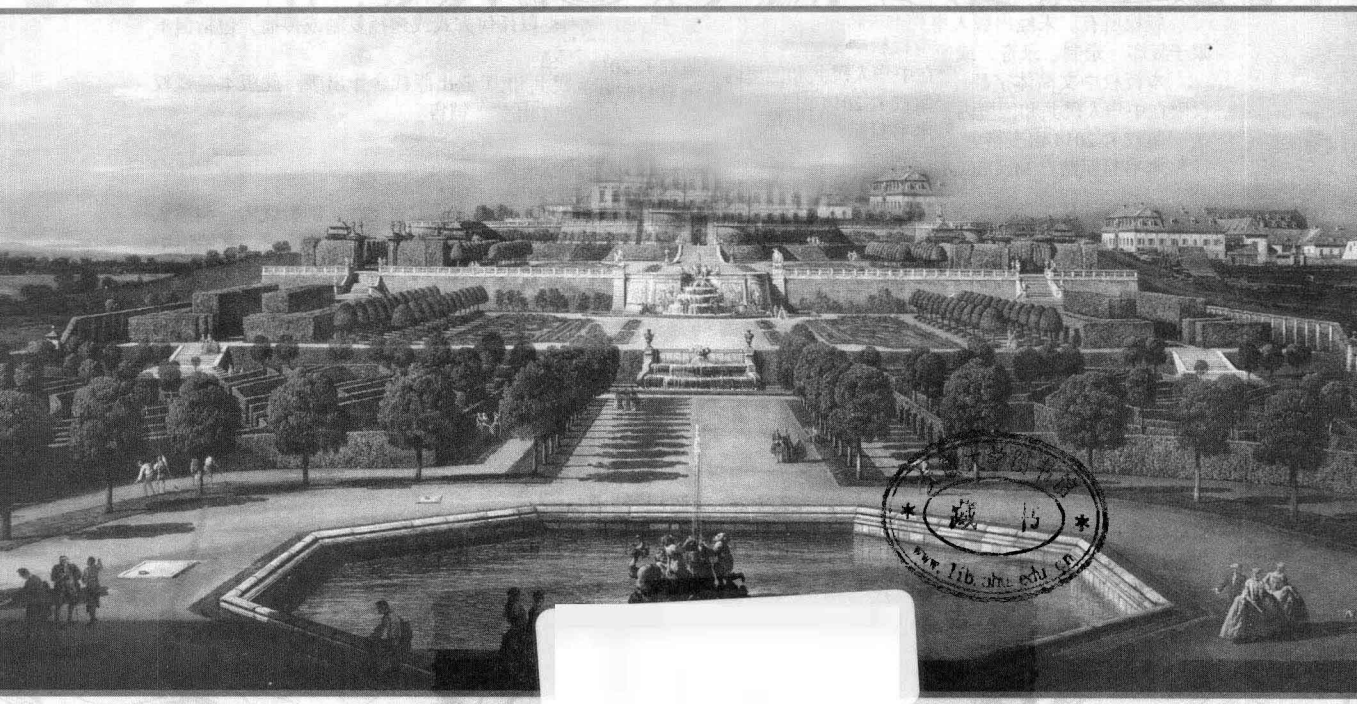
软件工程技术丛书

软件工程最佳实践

Software Engineering Best Practices

Lessons from Successful Projects in the Top Companies

(美) Capers Jones 著　吴舜贤 杨传辉 韩生亮 译



机械工业出版社
China Machine Press

图书在版编目（CIP）数据

软件工程最佳实践 /（美）琼斯（Jones，C.）著；吴舜贤，杨传辉，韩生亮译. —北京：机械工业出版社，2013.11
（软件工程技术丛书）
书名原文：Software Engineering Best Practices: Lessons from Successful Projects in the Top Companies

ISBN 978-7-111-44540-1

I. 软… II. ① 琼… ② 吴… ③ 杨… ④ 韩… III. 软件工程 IV. TP311.5

中国版本图书馆 CIP 数据核字（2013）第 252917 号

本书从软件工程的宏观层面，以专业的视角，摆事实、列数据，对比各种软件工程实践，剖析优劣，洞悉软件工程的是非与成败，揭露各种软件工程实践的伪真理，深刻指出软件项目中存在的各种问题的实质，并给出中肯的改进建议和解决方案。这些最佳实践来自作者所研究的全球超过 600 家知名软件公司和美国 30 余个大型政府机构，可以称得上是软件行业半个世纪以来全球范围内软件工程实践的精华。本书共分 9 章。第 1 章给出软件工程“最佳实践”的定义，第 2 章探讨软件工程领域的 50 条最佳实践，第 3 章展望未来软件开发的状况，第 4 章评估学习新的软件工程信息的 17 个渠道，第 5 章展示许多不同类型组织结构的考察结果，第 6 章讨论涉及项目管理的关键职能，第 8 章探讨编程和代码开发工作以及度量编程效率和编程质量的方法等，第 9 章讨论正式审查、静态分析以及其他 17 种不同形式测试方法的优势和劣势等。

机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码 100037）
责任编辑：谢晓芳
北京市荣盛彩色印刷有限公司印刷
2014 年 1 月第 1 版第 1 次印刷
186mm × 240mm · 31.75 印张
标准书号：ISBN 978-7-111-44540-1
定 价：99.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991 88361066 投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259 读者信箱：hzjsj@hzbook.com

译 者 序

自20世纪40年代中期现代计算机诞生以来，作为现代计算机“灵魂”的计算机软件经历了程序设计、软件设计和软件工程等发展阶段。尤其是自1968年提出“软件工程”概念以来，软件行业经历了风起云涌的飞速发展。各种各样的编程语言、形形色色的开发方法、多如牛毛的软件工具、不计其数的软件应用不断如雨后春笋般涌现。迄今为止，也许还没有哪个工程行业像软件工程领域如此欣欣向荣。这种蓬勃发展的强劲势头的确令人欢欣鼓舞。从表面上看，我们有充足的理由对软件工程的未来充满信心和期待。

对于软件这样一个年近古稀的行业，我们理所当然地认为它应该是非常成熟、可靠的。但事实是这样吗？在软件行业欣欣向荣的背后，我们也看到了令人不安的一面。正如Peter R. Hill（ISBSG CEO）所说的那样，熙熙攘攘的软件产业就像时装行业一样，不断被各种流行时尚裹挟着前行，却始终无法找出解决软件工程中存在的最根本问题的解决办法。屡见不鲜的软件工程项目失败、进度落后、预算超支、软件质量低劣、程序崩溃、数据丢失、客户严重不满甚至法庭上兵戎相见，数十年来似乎从未间断。静下心来，梳理软件工程的纷繁复杂，去芜存菁，细数软件行业的种种实践、方法、工具、语言、组织、专业、度量、质量等方方面面，细细想来，正如本书作者在书中屡屡强调的那样，软件工程还真是一门“手艺”而非一个真正的工程学科。

本书作者Capers Jones是一位睿智的长者。他科学、严谨的研究方法和一丝不苟的态度，使译者对Capers产生了由衷的敬佩！Capers潜心研究软件工程数十年，以深邃的洞察、广博的知识、严谨的思维和严格的论证，洞悉软件工程的是非与成败，揭露各种软件工程实践的伪真理，深刻指出软件项目中存在的各种问题的实质，孜孜不倦地为我们提出各种改进建议和解决方案。Capers的研究，没有空洞的说教，每每以超过15 000个项目的真实状况和历史数据为依据，对比各种方式方法的优劣，让我们能够看清软件行业各种“传闻”的真实面貌。30余年来，作者在他出版的17本书中对各种软件工程实践进行了深刻剖析，对软件行业的种种传闻和说法给出了自己的看法与建议。他的很多观点，多次引起业界热烈讨论，甚至引发了激烈论战。

在这本书里，作者站在软件工程的宏观层面，以专业的视角，摆事实、列数据，对比各种软件工程实践，剖析优劣，并给出中肯的改进建议，所谓“最佳实践”是为如此。这些最佳实践，来自作者所研究的全球超过600家知名软件公司和美国30余个大型政府机构，可以称得上是软件行业半个世纪以来全球范围内软件工程实践的精华。阅读本书，跟随作者将软件工程的各种实践放到显微镜下仔细审视，很多地方常常引起译者的诸多思考和强烈共鸣，不觉为作者的精辟论述拍案叫绝。作者对软件工程的方方面面都进行了探讨，

涵盖面之全、讨论点之细，令译者也无不惊讶。看完本书，译者有一种强烈的冲动，要将这本书和作者对软件工程的深刻见解尽快分享给无数仍然被软件项目中的各种问题所困扰、苦苦挣扎在软件工程实践第一线的广大中国同行们。

如果你是软件公司高管，本书论述的最佳实践将会使你的公司从中受益。如果你是软件项目经理，你将能够更加清楚地看清工作中遇到的各种问题的根本原因。如果你是软件测试人员，本书将使你能够以更高的视角看待软件测试工作，明白软件测试的核心目的不是发现了多少缺陷，而是如何给客户交付高质量的软件。如果你是质量保证人员，你将会更加深刻地明白，我们是来帮助团队提高软件质量而不是来阻碍团队前进的。如果你是软件开发人员，本书将使你跳出编码工作本身，从更高层面深刻认识软件开发所涉及的方方面面，更加明白为什么当前编码工作量不及软件项目全部工作量的40%。

无论你是谁，本书关于软件职业、专家、组织结构及学习渠道的分析，能使你清晰了解你在软件行业的位置，明了你所从事的工作所需要的专业知识以及如何获取这些专业知识。无论如何，站在软件工程的宏观层面，纵览软件工程的前世今生，品析软件行业的是是非非，探讨软件项目的改进建议，预测软件开发的未来发展，能够使软件工程从一门手艺演变为一门真正的工程学科，将对软件行业大有裨益。

这本书是Capers的又一鸿篇巨制，英文版厚达600余页，因而翻译该书绝非一人之功所能轻易为之。本书的成功翻译是团队协作的结晶。吴怡编辑统筹全书翻译，耐心指导、仔细审阅，为翻译的顺利进行提供了很多宝贵的建议和帮助。杨传辉负责本书第1～3章及前言的翻译，韩生亮负责第7～8章的翻译，吴舜贤负责第4～6章、第9章以及前言、致谢等其他部分的翻译。

在本书翻译过程中及完稿之后，很多朋友参与了译稿审校，在这里对他们的帮助深表感谢：王剑华、张晓辉、卢永安、付勇智、叶慧、孟美。很多知名软件人士及朋友对本书某些词汇、句子的翻译进行了深入讨论并给出了翻译建议，在此一并表示感谢：崔启亮、朱少民、王正、徐毅、朱筱丹、刘圣文、赵霞等。最后，感谢家人张少真的全力支持使我能够全身心投入翻译之中。

本书译者均来自软件工程实践第一线，对技术图书的翻译素怀敬畏之心和惶恐之情，唯恐不能精准、地道地表达作者原意以致误导读者，因而在翻译过程中查阅了大量资料，通过微博、邮件等向原书作者Capers Jones先生以及许多知名软件人士广为求教，对书中的很多术语、段落、表达方式反复推敲、再三讨论以确定最终译法，倾尽全力将翻译错误减至最少。但由于时间紧迫加之水平有限，书中错误疏漏之处在所难免，恳请广大读者不吝赐教、批评斧正。

吴舜贤

序

软件工程是一个关于人的行业——需要并使用软件的人、构建软件的人、测试软件的人、管理软件项目的人以及那些支持和维护软件的人。可是，为什么软件工程的重点仍然集中在技术上了呢？

软件开发业务有多少年的历史了？让我们假定它已有55年之久。这意味着软件行业已历经了整整一代人。这些人受到培训，为软件行业奉献了他们毕生的精力，而现在，他们要么已经退休，要么即将退休。在他们的岁月里，他们无数次目睹了“更好方法”的承诺——那些鬼使神差的银弹。现在，软件工程行业已经有了成千上万的新编程语言、形形色色的新方式方法、多如牛毛的新软件工具。有时候，软件行业似乎就像时装行业一样，强劲地被各种各样的流行时尚推动着不断前行。许多从业者崇拜特殊方法、实践或者工具，并成为了虔诚门徒，至少在某一段时间内是这样。但是，当我们收集并分析各种指标数据的时候，我们发现了一个可悲的事实：作为一个行业，我们其实没有取得多大的进步。在按时按预测成本顺利交付高质量软件方面，仍然没有任何重大突破。而怀疑论者也会问：“软件工程是自相矛盾的行业吗？”

我们对技术的执着和眷恋蒙蔽了我们的双眼。我们不希望或者无法看到，我们真正需要的是基础，即良好的工程实践。在这本书里，Capers Jones主张，如果软件行业要想被认可为一个工程学科而不是一门手艺，就必须采用稳固可靠的工程实践方法经济高效地交付可接受的软件产品。技术令人着迷，但当需要出色地完成工作并交付良好质量的软件产品时，它并不是最重要的因素。人们的工作方式、人们做出的抉择以及人们选择去遵守的规范，都比技术选择对软件工程成功的影响大得多。

一定有很多次，Capers会感觉自己就像个先知，孤独地行走在一片荒漠之中。他一直孜孜不倦地为软件行业提供各种指导和教诲，努力推动软件行业成为一个真正的专业和工程学科。他的努力跨越28年之久，成就了16本软件专业著作。很多次我都怀疑Capers晚上是否睡觉。他的书籍草稿总是源源不断地出现在我的收件箱里，无论是白天还是夜晚。当你读到一些东西，然后对自己说“好吧，这很有道理；真的，这显而易见啊”的时候，你会认识到作者已经做了一件了不起的事情。Capers就是这样的一位作家。他所写的东西常常引人入胜、易于理解，而且注重实效。我手头上的他的书籍，经常被我翻阅，还贴满了记事便签。这证实了他所著书籍的实用性。

Capers有一种能力，使得他可以置身事外，认真观察那些仍然长期困扰软件工程行业的各种问题的实质。Capers无所畏惧地批判着他认为对软件行业非常危险的那些实践——在这本书里，他的批评目标是“平均缺陷成本”和“代码行”度量指标。毫无疑问，尽管

他废除这些有害度量措施的理由非常充分，但他的观点仍会引起争论。辩论和讨论将会非常激烈，而这对于软件行业来说仍是期待已久的幸事。Capers 站在专业的角度点燃导火索，引发了一场关于这些度量指标到底是否有害的论战。

在这本书中，作者也将软件质量放到了显微镜下仔细审视。他把软件质量描述为软件工程成功的关键因素——比任何其他因素对软件成本、进度和最终成功都具有更大影响的驱动因素。由于 Capers 对一些常见的软件质量定义提出了挑战，因而同样会充满争论。

纵贯全书，作者始终在鲜明强调软件度量措施和度量指标的重要性。Capers 对软件行业缺乏度量实践和使用他描述为“危险”的那些度量指标进行了批评。鉴于使用度量措施和度量指标很少，软件行业理应受到激烈批评和强烈质疑。正如 Capers 所断言的，当我们无法给出统计学上的量化证据来加以证明时，像“最佳实践”这样的术语令人尴尬。

软件工程已有 55 年的悠久历史了，软件工程应该变得成熟了。在本书中，Capers Jones 对软件工程中的人与管理问题的重点强调，为软件工程实现真正的成熟指出了光明大道，并由此软件行业要成为一个真正工程学科的努力方向。

——Peter R. Hill

国际软件基准组织（ISBSG）有限责任公司 CEO

前　言

本书写作之时，正值全球经济衰退开始。因此，与过去的一些书相比，在应对软件工程的问题上，本书提供了一个新的研究方向。

由于经济衰退，本书有了许多新的素材，这些素材涉及：裁员和机构精简；美国和其他国家之间不断变化的经济平衡；经济衰退时期的软件经济学等。

当2008年金融危机和经济衰退开始时，软件工程并没有立即受到影响，但随着时间推移，风险投资基金开始逐渐耗尽。软件公司其他形式的融资变得困难起来，因此到2009年中期，软件工程类职位开始受到裁员的影响。鉴于在经济下行期间，诸如质量保证和技术写作等专业职位往往在第一轮就被波及，因此这些职位受到的影响甚至更加严重。

经济衰退带来的一个意想不到的副产品是，由于软件工程师供应的大量增加加上福利待遇的降低，因此这使美国成为软件外包的一个候选国家。

截至2009年，美国、印度以及中国之间的成本差距在减少，相对于国际合同，美国国内合同的便利可能也证明这是对美国软件工程界有益的一件事。

随着经济衰退的持续，高成本的软件正在经历空前考验。经济衰退也突出了这样一个事实，即软件一直都是不安全的。由于经济衰退，诸如有价值信息的盗窃、身份盗用甚至监守自盗等网络犯罪现象迅速增加，“网络钓鱼”或试图使用虚假消息以获取个人银行账户访问权限的现象也在大幅上升。

从笔者的角度来看，经济衰退已经凸显了，要使软件工程成为一门可靠的工程学科，而不是一门手艺，需要改进以下四个重要领域：

1. 需要通过提高应用程序的免疫力水平，以及包括编程语言自身更好的安全特性等措施来逐步改善软件的安全性，而访问控制和权限控制一直也是软件工程的薄弱环节。

2. 需要从缺陷预防方法和缺陷去除方法两方面来改善软件质量。近50年来，不佳的软件质量一直破坏着软件经济，而这种情况不能再持续下去了。每一款重要应用软件都需要有效地结合审查、静态分析和测试等方法来改善软件质量，而只靠软件测试这一种方法并不足以获得高品质的软件。

3. 为了更好地理解软件开发和软件维护的真实经济价值，需要对软件的度量方式加以改进。这意味着需要考虑活动级别的软件成本，也意味着要分析传统度量指标（如“平均缺陷成本”和“代码行”等）的缺陷，这些度量指标违反了标准经济学的有关法则。

4. 由于经济衰退，新的软件开发正不断放缓，因此需要更好地理解软件维护和改造的经济价值。维护和改造遗留应用程序的方法变得越来越重要，为了获取“丢失”的软件需求和业务规则而对遗留应用程序进行挖掘的方法也同样重要。

截至2009年，绝大多数企业和软件工程师，对生产力或软件质量都没有一个有效的度量方法，这不符合一个真正的工程学科的标准。有效度量指标的缺乏加上现有危险度量指标的使用意味着，诸如敏捷开发、Rational统一过程（RUP）以及团队软件过程（TSP）等软件方法的有效性评估将更加困难。

虽然在经济增长时期，度量措施以及软件工程方法和实践的有效性评判能力的缺失只是有些不便，但在经济衰退时期，这却是一个严重问题。糟糕的度量方法使得诸如“最佳实践”等短语在描述软件时非常尴尬，因为大量最佳实践的主张并不是基于有效度量指标而获得的可靠度量结果。

本书尝试说明度量指标与度量措施的结合能证明“最佳实践”的有效性，并且希望为软件工程建立一个良好的经济基础。本书所述的“最佳实践”是指那些由量化数据至少提供了一种临时能力以能够判断其有效性的软件实践。

这本书共分9章，每一章都涉及了一套技术和非技术问题。

第1章：软件最佳实践的介绍和定义

由于许多软件应用程序在第一次交付后，可能会持续使用25年甚至更长时间，所以软件工程不能仅仅关注软件开发活动。软件工程需要考虑软件交付多年后的维护和改进工作。软件工程还需要包括为提取或“挖掘”遗留应用程序而使用的有效方法，以恢复丢失的业务规则和软件需求。

负责软件维护的软件工程师数量要比负责新软件开发的工程师多得多。许多软件工程师的任务是维护并非自己开发的应用程序，这些软件可能是用“过时”的编程语言来编码的，并且代码本身既没有任何功能说明也没有有效的注释。

软件工程“最佳实践”不是一个“放之四海而皆准”的技术。最佳实践的评估要求那些实践在拥有100个功能点或者更少功能点的小型应用程序、拥有1000个功能点的中型应用程序以及可能拥有超过10万个功能点的大型系统等规模的软件应用项目中进行评价。

此外，那些对Web应用软件和IT应用软件有效的“最佳实践”并不一定要对那些嵌入式应用、系统软件以及武器系统软件具有同等良好的效果。

由于在应用规模和应用类型上变化巨大，本书对这两个方面的最佳实践均做了评估。

第2章：50个软件最佳实践概述

该章探讨了50个软件工程最佳实践。并非所有的实践都是纯技术性的。例如，在经济衰退时期，如果处理不当，企业裁员将会损坏其多年的经营效益。

该章涉及软件开发的最佳实践、软件维护的最佳实践、项目管理的最佳实践以及社会学的最佳实践，例如，那些涉及裁员的实践办法，这方面经常会处理不当，例如淘汰经理和主管人员过少，但淘汰软件工程师和软件专家过多。

除裁员以外的其他方法，例如，缩短所有员工的工作时间以及减少福利待遇，往往是首选办法，这也相当于减员。

其他最佳实践领域包括安全控制、质量控制、风险分析、软件治理、软件开发、软件维护以及遗留应用改造等。

该章部分内容曾经应用于软件诉讼，这些诉讼中的某些软件项目实践未能符合软件工程最佳实践成为原告提起诉讼索赔的原因之一。

第 3 章：2049 年的软件开发和维护预览

因为软件工程最佳实践是在接近 2009 年时才开始实施的，所以我们很难想象它的变化和改进。第 3 章让我们提前穿越到 2049 年，并探讨那个时候的软件工程将是什么样的。那时的世界可能是这样的，我们所有人都通过社交网络来相互联系，软件工程的工作可以很容易地分配给分别位于许多不同国家的软件工程师。

该章还展望了具体技术的发展，如数据挖掘技术在需求收集方面的作用，以及可获取大量认证的可重用材料。可能也包括可用于重要议题上对信息进行积累和分析的智能工具与搜索机器人。

鉴于技术的高速发展，可以预见的是，到 2049 年，计算设备、网络和通信渠道等硬件将极其成熟。但软件工程往往滞后于硬件的发展。为了跟上硬件和网络的发展速度，软件工程需要在安全性、质量以及可重用性上做出重大改进。

第 4 章：软件人员如何学习新技能

由于经济衰退，纸质图书和软件杂志出版商正面临着严重的打击，其中很多出版商都在裁员。在线出版和电子书籍，如亚马逊的 Kindle 和索尼的 PR-505，正在迅速壮大。就提供者和用户而言，Web 出版物、博客以及 Twitter 也在不断扩张。

第 4 章评估了 17 个学习渠道，即传播和学习新软件工程信息的渠道。在学习效率和成本效益方面，对每个渠道都进行了排名。针对每种渠道的未来也做了长期预测。

已评估的某些学习渠道包括：传统的纸质图书、电子图书、软件期刊、电子期刊和博客、wiki 站点、商业培训、在职培训、学术教育、现场技术大会、网络研讨会以及在线会议等。就成本效益而言，电子媒体已经超越了印刷媒体，而在学习有效性方面也正挑战着传统媒体。

第 4 章还为软件工程师、软件质量保证人员、软件测试人员、软件项目办公室人员以及软件经理推荐了一些课程。虽然当今的教学课程足以满足主流软件工程的需要，但在诸如规模估算、评估、规划、安全、质量控制、维护、革新以及软件工程经济分析等主题上，仍然缺乏可靠的教育。

软件度量指标几乎不被学术界关注，而对诸如平均缺陷成本和代码行等常见指标的缺

陷也很少有批判性的分析。

虽然功能性度量指标在很多大学有所教授，但是很少有人针对行为异常或者违反制造业经济原则的旧度量指标进行经济分析。特别是，固定成本对生产力的影响没有涉及，这也是代码行和平均缺陷成本为什么对经济分析无效的主要原因。

第 5 章：软件团队的组织和专业化

软件工程组织的范围很广，从生产小型应用软件的一个人独立公司，到超过 1000 人的大型组织（可能是雇用超过 5 万软件从业人员的公司的一部分）均包括在内。

除了软件工程师本身，大型软件工程组织雇用了超过 90 种各类专家，例如质量保证专家、技术文档作家、数据库管理员、安全专家、网站管理员以及度量指标专家等。

第 5 章展示了许多不同类型组织结构的考察结果，其中包括结对编程、小型敏捷开发团队、层级式组织、矩阵式组织以及在地理上分散的虚拟组织等。该章同时还说明了组织和管理诸如软件质量保证、测试、技术文档以及项目管理办公室等专家最行之有效的方式。

例如，对于大公司里的大型软件项目，独立的维护团队和独立的测试团队往往比开发团队自身进行软件维护与测试更为有效。软件专家和通才必须共同努力，而组织结构对整体业绩有着很强的影响。

第 6 章：项目管理和软件工程

众所周知，许多软件项目的规模估算是不正确的，因此提交的项目进度日程对开发团队的能力来说可能太短。项目管理中的这些失误可能会影响软件项目，使之要么完全失败，要么出现严重的成本超支和工期延误。

第 6 章涉及了一些关键管理职能，例如项目大小估算、规划、评估、进度跟踪、资源跟踪、基准以及变更管理等，如果处理不好这些管理职能，就会导致软件工程的失败。

对于每个稍大的软件项目，第 6 章建议我们收集那些可以作为项目基线和基准的软件质量与生产率方面的数据。对生产力和质量数据的收集应该是普遍的，而不是罕见的例外情况。

使用严格的度量方法并获取度量结果是职业化的象征。而未能很好地进行度量则说明软件工程还不是一门真正的工程学科。

第 7 章：需求、业务分析、架构及设计

在编写任何代码之前，有必要先理解用户需求。这些需求应映射到有效的软件架构，然后再翻译成详细设计。此外，新的应用程序应该置于整体企业应用项目组合的背景下加以评估。在 20 多种形式的需求方法和 40 多种类型的设计方法中，软件工程师可以有很多

种选择。

该章探讨了处理需求和设计问题使用最广泛的方法，并展示了最适合它们的应用程序类别和类型。敏捷方法、统一建模语言（UML）以及很多其他技术都将在这里加以讨论。

大概在2009年，大型企业的全部投资项目组合可能拥有超过5000个应用程序，总计超过10万个功能点。项目组合可以包括内部应用程序、外包应用程序、商业应用程序以及开源应用程序等。

项目组合还可以包括Web应用程序、信息技术应用软件、嵌入式软件以及系统软件等。由于经济衰退，公司高管们是否了解企业项目组合的规模、内容、价值、安全等级以及质量水平等，变得越来越重要。

过去，难以量化各种应用程序的规模，这已经阻碍了项目组合分析。而今天，在商业应用程序、开源应用程序以及内部应用程序上，新型高速规模估算方法已经开始消除这些历史问题。现在，在短短的几天到几个星期时间里，已经能够估算成千上万个应用程序的规模了。

第8章：编程和代码开发

该章以一个非同寻常的观点来探讨编程和代码开发工作。截至2009年，大约有2500种编程语言及其派生语言。该章提出了“软件工程为什么有如此之多的编程语言”的问题。

第8章还提出了“过多的编程语言对软件工程专业到底是有益还是有害”这样的问题。此外，还讨论了许多应用程序使用2～15种不同编程语言的原因。总体结论是，虽然有些编程语言对软件开发有益，但是2500种语言的存在对软件维护工作来说简直就是一场噩梦。

该章还建议美国国家编程翻译中心记录所有已知语言的语法，并协助将那些用已过时语言编写的应用程序转换成现代编程语言编写的应用程序。

该章还包括，在源代码中发现各种类型的bug信息，以及与功能测试和回归测试等公开活动相比，由软件工程师自己执行的最有效的“个人”缺陷预防和缺陷去除方法。

例如手工检查和单元测试等个人方法经常很难度量。然而，志愿者通过“私人”的缺陷去除活动来发现缺陷并做信息记录，因此这些数据常常可用。

该章还介绍了度量编程效率和编程质量水平的方法。由于挑战传统的在经济上无效的“代码行”（LOC）度量指标，该章内容备受争议。LOC度量指标对高级编程语言极为不利，同时也误导了经济分析。

即便是在2009年，代码行度量指标不能处理需求、设计、界面或文档等工作，而这些非编码活动的费用占了软件开发项目总费用的60%以上。

另一种可供选择的是功能性度量指标，它可以处理所有已知的软件工程活动。然而，软件功能性度量指标的计算速度缓慢并且成本昂贵。大约2009年，开始出现了新型的高速

功能性度量指标，并且这些指标的使用也在不断扩大。

第 9 章：软件质量：软件工程成功的关键

长期以来，在与软件工程相关的各个技术环节中，软件质量一直是最薄弱的环节之一。该章试图涵盖所有影响软件质量的主要因素，其中包括缺陷预防方法和缺陷去除方法。

该章探讨了正式审查、静态分析以及其他 17 种不同形式测试方法的优势和劣势。此外，该章还涉及了各种令人烦恼的度量指标，这些度量指标降低了人们对软件质量的理解。例如，比较流行的“平均缺陷成本”指标实际上对软件质量不利，使漏洞百出的应用软件也能达到最低的平均缺陷成本！此外，软件质量具有的经济价值远远超过了单纯的缺陷去除成本的经济价值，而这种价值不能使用平均缺陷成本指标来体现。

该章的主题是，软件质量是一个在软件成本、进度以及项目成功上比其他指标具有更大影响力的因素。但是，糟糕的度量方法使得难以开展有效的软件工程经济研究。

该章饱受争议的原因是，我们质疑了两种常见的质量定义。质量意味着“符合需求”这一质量定义被质疑的理由是，许多需求是有害或者“有毒的”，并且不应该实现。质量意味着满足一系列以“ility”结尾的特性（比如“可移植性”（portability）），这一质量定义受到挑战的理由是这些特性中的某些既不能预测也无法度量。人们需要一个这样的质量定义，“质量”既可以在应用程序开始运行之前预测，又可以在应用程序完成之后度量。

质量是软件工程成功的关键。但在打开通向专业化的大门之前，我们有必要了解如何度量软件质量以及软件经济性。该章的结论是，无法度量自己成果的活动不是一个真正的工程学科。现在非常需要研究诸如潜在缺陷和缺陷去除效率水平等关键议题。

截至 2009 年，大多数软件项目都包含过多的错误或缺陷，而在软件交付之前去除的缺陷或错误少于全部缺陷的 85%。每一名软件工程师和软件项目经理都应该知道需要什么样的审查、静态分析以及阶段测试结合，以实现缺陷去除效率水平到达 99%。如果没有基于精确度量的缺陷预防和缺陷去除知识，软件工程就是不当用词，而软件开发也只是一门手艺而非一种真正的职业。

软件工程最佳实践的总体目标 本书的灵感之一来自于 1982 年的一本老书，这本书是《The Social Transformation of American Medicine》（《美国医学的社会转型》），作者是普利策奖得主 Paul Starr（保罗·斯塔尔）。

在我读保罗·斯塔尔的书之前，我并不知道，原来 150 年前，医学院的学生通过两年的学习后，就可以被授予医学学位，并且没有任何医院实习或高级住院专科实习的要求。其实，大多数在培训的医生从来没有进入过医院或者医学院。更令人吃惊的是，学生在进入医学院的时候并没有提供任何大学学位或者高中文凭证明。在美国，超过 50% 的医生从来没有上过大学。

保罗·斯塔尔的书详细介绍了美国医学协会的尝试，以改善医生的学术培训、建立职

业过失的准则以淘汰庸医、提高医生的职业地位等。在保罗·斯塔尔的书中，很多经验对软件工程很有价值。

这本关于软件工程最佳实践的书的主要目标是，为促使软件工程建立在软件质量和生产力精确度量这一事实基础之上而提供激励措施。

随着经济衰退的持续，软件行业越来越需要减少软件项目失败、加快软件交付、降低软件维护费用。没有软件工具、开发方法、编程语言以及软件组织结构等的有效性的精确度量，这些要求都无法实现。

精确的度量是获得更好软件质量和安全性的关键。而更好的软件质量和更高的安全性是让软件工程变成一种真正职业的关键，即等同于那些已经取得了成功的旧工程领域。

软件工程的度量结果也会产生更多、更好的软件基准，而这反过来将为已证明是有效的软件工程方法提供有力的证据。本书的整体主题是，作为软件工程成功的先驱条件，软件工程需要更完善的度量措施、更全面的软件基准、更严格的质量控制以及更高的安全性。

致谢

一如既往地首先感谢我的妻子——Eileen，是她的支持使我在过去的28年里完成了16本著作。

还要感谢那些为本书及笔者以前的很多书籍提供真知灼见的很多同行们：Michael Bragen和Doug Brindley，我任职的前一家公司——软件生产力研究所（SPR）的CTO和总裁；Tom Cagley，国际功能点用户组（IFPUG）总裁；Bob Charrette，ITABHI公司总裁；Coverity Systems公司的Ben Chelf；微软公司的Steven Cohen；James Madison大学的Taz Doughtry博士；Chas Douglis，软件生产力研究所前总裁；Larry Driben博士，Pearl Street软件公司总裁；Gary Gack，Process Fusion公司总裁；Jonathan Glazer，PowerBees公司总裁；Scott Goldfarb，质量与生产力管理集团总裁；Steve Heffner，Pennington Systems公司CEO；Peter Hill，国际软件基准组织（ISBSG）CEO；软件工程研究所（SEI）的Watts Humphrey；Ken Hamer-Hodges，Sipantic公司总裁；IBM Rochester研究中心的Steve Kan博士；北德克萨斯大学的Leon Kappleman博士；Ravindra Karanam，Unisys公司软件业务部门的CTO；Dov Levy，Dovél Systems公司总裁；Tom Love博士，Shoulders公司总裁；Steve McConnell，Construx公司总裁；Michael Milutis，信息技术指标与生产力研究所（ITMPI）主管；Peter Mollins，Relativity Technology公司首席营销官；Unisys公司的Prasanna Moses；Walker Royce博士，IBM Rational事业部负责人；Akira Sakakibara博士，IBM东京研究中心的杰出科学家；Tony Salvaggio Computer Aid Inc.（CAI）总裁；Paul Strassmann，信息经济出版社总裁（美国国防部前CIO）；以及Cem Tanyel，Unisys应用软件开发服务部门的副总裁兼总经理。

我们还应当向两位在软件专业化领域做出大量卓越贡献的高管表示特别的赞誉。已故

的 James H. Frame，IBM 圣特雷莎实验室主管及随后的 ITT 康涅狄格州斯特拉特福德编程开发中心副总裁。已故的 Ted Climis，IBM 副总裁，他在 IBM 识别了大量的关键软件角色。上述两位都清楚地认识到软件对于公司商业运营至关重要，需要以最高卓越标准来设计和构建软件产品。

谨以此书献给那些潜身研究并推动软件工程领域不断进步的同行们！其中一些人包括：Allan Albrecht、Barry Boehm、Fred Brooks、Tom DeMarco、Jim Frame（已故）、Peter Hill、Watts Humphrey、Steve Kan、Steve McConnell、Roger Pressman、Tony Salvaggio、Paul Strassmann、Jerry Weinberg 以及 Ed Yourdon。

目　录