Microsoft®

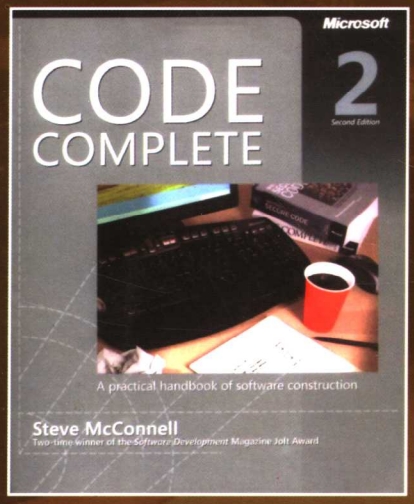
Broadview®
WWW.BROADVIEW.COM.CN

CODE COMPLETE

Second Edition

代码2大全

第2版



[美] Steve McConnell 著

两届 *Software Development Magazine* *Jolt Award* 震撼大奖得主

金戈 汤凌 陈硕 张菲 译

裘宗燕 审校

软件构建之实践指南

A practical handbook of software construction

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

代 码 大 全

—第 2 版—

CODE COMPLETE

Second Edition

[美] Steve McConnell 著

金戈 汤凌 陈硕 张菲 译

裘宗燕 审校

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

内 容 简 介

代码大全(第2版)是著名IT畅销书作者、《*IEEE Software*》杂志前主编、具有20年编程与项目管理经验的Steve McConnell十余年前的经典著作的全新演绎：第2版做了全面的更新，增加了很多与时俱进的内容，包括对新语言、新的开发过程与方法论的讨论，等等。这是一本百科全书式的软件构建手册，涵盖了软件构建活动的方方面面，尤其强调提高软件质量的种种实践方法。

作者特别注重源代码的可读性，详细讨论了类和函数命名、变量命名、数据类型和控制结构、代码布局等编程的最基本要素，也讨论了防御式编程、表驱动法、协同构建、开发者测试、性能优化等有效开发实践，这些都服务于软件的首要技术使命：管理复杂度。为了培养程序员编写高质量代码的习惯，书中展示了大量高质量代码示例(以及用作对比的低质量代码)，提高软件质量是降低开发成本的重要途径。除此之外，本书归纳总结了来自专家的经验、业界研究以及学术成果，列举了大量软件开发领域的真实案例与统计数据，提高本书的说服力。

本书中所论述的技术不仅填补了初级与高级编程实践之间的空白，而且也为程序员们提供了一个有关软件开发技术的信息来源。本书对经验丰富的程序员、技术带头人、自学的程序员及没有太多编程经验的学生都是大有裨益的。可以说，只要您具有一定的编程基础，想成为一名优秀的程序员，阅读本书都不会让您失望。

版权贸易合同登记号：图字：01-2005-0909

图书在版编目（CIP）数据

代码大全：第2版 /（美）迈克康奈尔（McConnell,S.）著；金戈等译. —北京：电子工业出版社，2006.3
书名原文：Code Complete，Second Edition
ISBN 7-121-02298-2

Ⅰ.代... Ⅱ.①迈...②金... Ⅲ.软件开发—手册 Ⅳ.TP311.52-62

中国版本图书馆CIP数据核字（2006）第011427号

责任编辑：周 筠 陈元玉
印 刷：北京智力达印刷有限公司
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：60.25 字数：1 000千字
印 次：2006年4月第2次印刷
印 数：10 001～20 000册
定 价：98.00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。

对《代码大全》的更多赞誉

“《代码大全》是有关编程风格和软件构建的绝好指导书。”

▶ **Martin Fowler**,《重构》

“Steve McConnell 的《代码大全》……为程序员提供了通向智慧的捷径……他的书读起来饶有趣味，要知道他可是有切实的亲身经验的。”

▶ **Jon Bentley**,《编程珠玑（第二版）》

“这无疑是我所看过的软件构建方面最好的书籍。每个开发人员都应该有一本，并且每年都从头到尾读一遍。九年来我每年都读这本书，仍能从中有新的收获。”

▶ **John Robbins**,《Microsoft .NET 和 Windows 应用程序调试》

“当今的软件必须是健壮、有弹性的，而安全的代码始于规范的构建。第 1 版出版后的十年里，没有出现比《代码大全》更权威的书。”

▶ **Michael Howard**,《编写安全的代码》

“《代码大全》广泛剖析编程工艺的各种实战话题。McConnell 的著作涵盖软件架构、编码标准、测试、集成以及软件工艺的本质等内容。”

▶ **Grady Booch**,《*Object Solutions*》

“对软件开发者而言，终极的百科全书就是 Steve McConnell 的《代码大全》。这本长达 850 页厚的书确如其副标题所说，是一本实用手册。它旨在缩短‘业界大师与教授’（例如 Yourdon 和 Pressman）的知识与一般商业实践之间的距离，帮助读者用较短的时间、碰到较少的麻烦去编写更好的程序……每个开发者都应该拥有这本书，其风格和内容是切实可用的。”

▶ **Chris Loosley**,《*High-Performance Client/Server*》

“Steve McConnell 的创新书籍《代码大全》是详述软件开发方面最易懂的一本书……”

▶ **Erik Bethke**,《*Game Development and Production*》

“《代码大全》是关于设计与生产优秀软件的实用信息与建议的宝藏。”

▶ **John Dempster**,《*The Laboratory Computer: A Practical Guide for Physiologists and Neuroscientists*》

“如果你有意改进编程技术，就该有一本 Steve McConnell 的《代码大全》。”

▶ **Jean J. Labrosse**,《*Embedded Systems Building Blocks: Complete and Ready-To-Use Modules in C*》

“Steve McConnell 写出了一本独立于特定计算机环境的软件开发方面最好的书籍。”

▶ **Kenneth Rosen**,《*Unix: The Complete Reference*》

“每个时代你都会遇到一本书，提供你获得经验的捷径，节省数年走弯路的时间……千言万语都无法说明这本书有多好。标题《代码大全》尚不足以表达出该作品的全部智慧与内涵。”

▶ **Jeff Duntemann**,《*PC Techniques*》

“Microsoft 出版社出版了我认为是软件构建方面很好的书，每个软件开发人员的书架上都该有这本书。”

▶ **Warren Keuffel**,《*Software Development*》

“每个程序员都该读读这本杰出的书籍。”

▶ **T.L. (Frank) Pappas**,《*Computer*》

“假如你期望成为专业程序员，这将是你投资 35 美元能得到的最好回报。不要只是看看这个书评，赶快冲出去买一本回来！McConnell 声称此书意在拉近业界大师的知识与一般商业实践之间的距离……令人称奇的是他做到了。”

▶ **Richard Mateosian**,《*IEEE Micro*》

“应当让在软件开发领域中的每个人都来读读《代码大全》。”

▶ **Tommy Usher**,《*C User's Journal*》

“我不遗余力地为 Steve McConnell 的《代码大全》拍手叫好……这本书取代了 API 参考手册，成为伴我干活的最亲密的书。

▶ **Jim Kyle**,《*Windows Tech Journal*》

“这本编纂精良的巨著有望成为软件实现的实践方面最好的专著。”

▶ **Tommy Usher**,《*Embedded Systems Programming*》

“这是我所读过的软件工程方面最好的书籍。”

▶ **Edward Kenworth**,《*.Exe Magazine*》

“该书必将成为一部经典的、所有开发人员及其管理者必备的读物。”

▶ **Peter Wright**,《*Program Now*》

译 序

这本书讲什么

《代码大全》这本书的原名叫《*Code Complete*》，那么 code complete 在这里是何含义呢？首先，它不代表现代集成开发环境（IDE）中的代码自动补全功能，本书也不打算向您讲解 Eclipse 或 Visual Studio 2005 中的代码自动补全功能是如何实现的☺。其次，code complete 也不是真正的软件源代码“大全”的意思☹，这本书既没有列出连接各种数据库的代码、也没有列出网页中常用的各种 JavaScript 代码。书中的代码示例恐怕也不能直接 copy&paste 代码到您自己的项目中。

那么 code complete 到底是什么意思？中译本为什么又要取名为“代码大全”呢？虽然从网上讨论的情况看，各位网友对书名含义的理解有出入，但是译者有充分的理由相信，code complete 是“编码完成”的意思，是一个软件项目开发过程中的重要里程碑（milestone）。软件项目进行到这里，表明已经完成了所有的编码工作，即将开始系统测试。

这本书讲的正是为了到达“编码完成”这一重要里程碑所必需的软件构建技术，确切地说，就是如何编写高质量的代码。作者认为，应该首先为人编写代码，其次才是为机器（第 34.3 节）；代码主要是供人阅读的。遍布全书的提高代码质量的实实在在的技术和诀窍，是本书最有价值的部分。事实上，我们认为第 6、7、10 至 19 章这 300 多页的内容是本书的精华内容，在其他书里恐怕很难找到如此详尽的对变量、语句、子程序等编程基本要素的讨论。

十多年前，本书第 1 版以《代码大全》为名翻译出版，在过去的 10 余年中，这本书影响了整整一代程序员，“代码大全”四个字已成为一个响当当的名字。鉴于此，本书第 2 版决定保留这个无伤大雅的“错误”，沿用“代码大全”作为书名，也借此向原书第 1 版各位译者、修订者们的辛勤劳动表示我们的敬意。无论如何，对 code complete 的理解不会影响对整本书的理解。

本书除了讲如何构建高质量的软件，还讲如何成为一名优秀的程序员（第 33 章“个人性格”、第 4.3 节“你在技术浪潮中的位置”、第 34.4 节“深入一门语言去编程”）。

这本书适合谁看，该怎么看

任何想写出好程序的人，或者想带领一群程序员写出好软件的人，都不应该错过这本好书。作者在前言中指明了本书的读者群（包括经验丰富的程序员、技术带头人、自学的程序员、学生等），请您参阅。

这是一本800多页的大部头，从头到尾阅读要花不少时间，谁都希望能尽快找到对自己有用的内容。译者大致针对不同的读者群提一点阅读建议，仅供参考。

- 初级程序员，请先看第18章“表驱动法”：将复杂的逻辑判断转换为查表，从而简化代码的编写与维护。另外，本章中的一个示例说明了，面向对象设计并不只要因为它是“面向对象”，就一定会好于其他的设计。
- 高级程序员，请先看第4章“关键的‘构建’决策”，本章关注的焦点是程序员和技术带头人个人必须（直接或间接）负责的项目准备工作。
- 项目经理，请先看第33章“个人性格”，程序设计是一项纯粹的脑力劳动，本章对挑选和培养优秀程序员提出了建议。事实证明，相对于聪明程度（智商），个人性格（情商）对于造就出程序员高手更具有决定性的意义。
- 低年级学生，请先看第11章“变量名的力量”。这本书用了整整一章（30多页）的篇幅来讲解“为变量命名”这一编程中最常见的活动，这里提供的建议在别的书里是很难见到的。
- 高年级学生，请先看第8章“防御式编程”，本章讲述如何面对严酷的充斥非法数据的真实世界，在遇到“绝不会发生”的事件和其他程序员犯下的错误时如何保护自己。对于那些正在从学术环境转向专业开发环境的学生来说，这是必备的一课。
- 制定编码标准的人，请先看第32章“自说明代码”，本章中有一段关于注释的精彩对话，它可能会改变您在制定编码标准时对注释的要求。
- 自学编程的人，请先看第7章“高质量的子程序”，本章详细讨论了子程序的命名和参数选择等问题，其中对子程序最佳长度的讨论颇有借鉴意义。
- 喜欢参与网上争论的人，请先看第13.3节“全局数据”和第17.3节“goto语句”，听听学术界在这些问题上的争论也挺有意思。

当然，这整本书都非常值得一读，准确地说，值得反复阅读。书中不仅有实实在在的数据和论述，也有一些有趣的比喻，作者偶尔还开开玩笑，读起来一点也不枯燥。

另外需要说明的一点是，书中出现的诸如“(Yourdon 1986b)”表示的是参考文献，可以从第863页起的参考文献列表中查到文献的原名和出处，例如，(Yourdon 1986b)代表的

是 Edward Yourdon 写的《*Nations at Risk*》一书。如果只出现“(2000)”字样，那么请您从上下文中推断出作者姓名。

配套网站

这本书英文版的配套网站是 http://www.cc2e.com，书中左侧出现的类似 cc2e.com/1234 的标志的含义请参阅前言中的说明。本书中文版的配套网站是 http://www.cc2e.com.cn，凡是书中出现的 cc2e.com/1234 均可对应访问 cc2e.com.cn/1234。

本书已经根据原书截至 2006 年 2 月初的勘误表进行了修订，译者发现的原书疏漏也已用译注标明。就像写程序做不到 bug free 一样，翻译书难免也会有错，如果您在阅读中发现任何疑问，欢迎来本书配套网站与译者交流。这个网站还提供最新的勘误表和其他一些信息（例如我们把 routine 翻译为“子程序”的理由、对书中观点的讨论、书评等）。

致 谢

本书的翻译工作由 4 名译者共同完成，各人负责的章节如下：金戈翻译前言和第 1、2、5~9 章，汤凌翻译第 10~26 章，陈硕翻译第 3、4、27~30 章，张菲翻译第 31~35 章。北京大学的裘宗燕教授审校了全部书稿，对译稿做了大量的修订并提出相当多的指导意见。在此我们对裘老师表示衷心的感谢，他的辛勤劳动使本书的翻译质量上了一个大台阶。全书最后由陈硕统稿。译者汤凌特别要感谢同事雷程炜工程师，他为汤凌复查了大部分初译版本文字。另外，本书部分翻译工作基于杨哈达和郑毅帆的初译稿，在此也一并致谢。

译者特别感谢本书编辑团队中负责全书统筹工作的陈元玉女士和负责本书配套网站建设的余广先生以及编辑团队的其他人员。

最后，祝读者能借助本书提高自己的编程功力，成为优秀的软件开发人才。谢谢！

译 者

2006 年 2 月初

谨以此书献给我的妻子 Ashlie。尽管她并没有干过什么计算机编程，但做了数不清的事情来使我的生活丰富多彩。

一切皆有可能

—出版人感言—

2003 年夏天，博文视点刚成立不久，一次，我和孟岩在 msn 上聊天，孟岩说："周老师，有本绝好的书刚刚出了第 2 版，不知现在版权还在不在？"我向来深信孟大侠的眼光，让他赶紧告诉我，原来是《代码大全》。孟岩还告诉我，该书第 1 版是十多年前问世的，很多人都在找这本书，未果。

我当即请电子社版权部的同事向微软出版社洽询《代码大全》（第 2 版）的翻译版权事宜，但版权部多次积极联络，对方的回复总是"在查询中"。后来和孟岩谈及，我们俩感到，很可能这本书的翻译版权已花落他家，多半没戏了。但既然对方没肯定说版权已经授予国内其他出版社，那么始终笃信"一切皆有可能"的我，就不会放弃申请。

2004 年的某个阶段，微软出版社因某种原因，暂停与中国出版社的版权贸易。向来办事效率极高的电子社版权部经多方打听，了解到微软出版社与国内出版社暂停合作的原因，提出了电子社和微软出版社率先启动合作的具体方案，并请博文视点参与同微软出版社的合作谈判，我两次在北京出席与微软出版社代表的面谈。每次面谈，我都要求博文的外版编辑在申请合作的书目上，首先列出《代码大全》（第 2 版），而对方也总是不能给予正式的回复。隐隐中，感觉这本书的翻译版权似乎离博文视点有些遥远……

然而，电子社是幸运的，博文视点是幸运的，我是幸运的——由于电子社版权部办事效率极高，赢得了微软出版社的信任，2004 年 12 月，对方发来了授权文件，其中就有《代码大全》（第 2 版）。得知这个消息，博文视点的外版编辑方舟有些不敢相信，还问我，会不会是微软出版社弄错了。方舟是个怀疑派，所以当初我要求他把这本书列入申请名单时，他略微嘟囔了几句，大意是说我们这些瞎猫想逮活耗子，云云。

接下来寻找译者也不易，所幸我没看错我的朋友金戈，他在翻译过程中，几次遇到困难：翻译团队中途换人，由他领衔主持的国家级项目，时间要求也非常紧，但他咬牙坚持下来了。也非常感谢裘宗燕老师一向对我工作的支持，裘老师答应担任这本书的审校工作，让我感到幸运而踏实。

这本书也凝聚了我的同事陈元玉无数个日夜的心血，她以高度负责的态度赢得了

全体译者的赞誉。方舟编辑，也从怀疑变为狂喜，在他眼里，这本书是“绝色佳人”，因此，美术基础不错的他亲自为这本书设计了封面。博文的市场经理佘广是网页设计爱好者，和编辑张昊一起为《代码大全》（第 2 版）精心打造了中文版配套网站（http://www.cc2e.com.cn）。

当年慧眼指路的孟岩，已经是《程序员》杂志的技术主编，他在 2006 年第 3 期《程序员》杂志上为这本书组织了 15 个版面的专题报道，这是空前的，也是“绝色佳人”才有的待遇。

出版人的快乐，莫不来自于与好书结缘。回顾一年多为这本书付出的辛劳，过往的一切都显得那么美好。我自己，更是感到特别的幸运——我能和这样一群优秀的伙伴在一个团队里工作，能在我的职业生涯里和这样一本“绝色佳人”级别的好书结缘，惟有感恩！

希望这本书，能带给读者真正的帮助，也恳请读者朋友随时指出我们应该改进的地方。

博文视点，愿与所有向上的心合作，共同成长！

周 筠

2006 年 3 月于武汉

Preface
前 言

普通的软件工程实践与最优秀的软件实践差距巨大——多半比其他工程学科中的这种差距都要大。因此，传播优秀实践经验的工具是十分重要的。

— Fred Brooks

我写这本书的首要目的，就是希望缩小本行业中一般商业实践与大师级人物及专家们之间的知识差距。许多强大的编程技术在被编程领域的大众接触之前，都已在学术论文和期刊里尘封了多年。

虽然近年来前卫的软件开发实践迅速发展，但普通的实践手段并没有太大变化。很多程序的开发仍然是漏洞百出、迟于交付并且超出预算，还有很多根本就无法满足用户的需求。软件业界以及学术界的研究人员已经发现了不少行之有效的实践经验，足以解决自 20 世纪 70 年代以来编程领域中日益蔓延的大多数问题。可是这些实践经验很少在高度专业化的技术期刊之外对外发表，所以时至今日大多数编程的机构和组织还没能用上这些技术。有研究表明，一项研发成果从其诞生之日起，到进入商业实践阶段，通常要经历 5 到 15 年甚至更长的时间（Raghavan and Chand 1989；Rogers 1995；Parnas 1999）。这本手册就是想缩短这一漫长的过程，让那些关键性的研发成果现在就能为更多编程人员所用。

Who Should Read This Book
谁应当阅读本书

本书中所汇集的研究成果和编程经验，将帮助你创建更高质量的软件，使你能更快速地进行开发，遇到的问题更少。本书将帮你弄明白过去为什么会遇到那些问题，并告诉你如何在将来避免它们。这里所描述的编程实践将帮助你掌控更大型的项目，还能在项目的需求发生变动时帮助你成功地维护并修改已经开发出来的软件。

Experienced Programmers
经验丰富的程序员

对于经验丰富的程序员而言，本书正是他们想要的一本翔实、易用的软件开发指南。本书关注的是“构建（construction）”，即整个软件生命周期中最为人熟知的部分；本书把强大的软件开发技术写得让自学的程序员和参加过正规训练的程序员都能读懂。

Technical Leads
技术领导

许多技术领导（或者说是技术带头人）都曾在他们的团队中使用《代码大全》（第1版）培训经验不足的程序员。当然，本书也可以用来填补你自己的知识缺陷。如果你是一位经验丰富的程序员，你不一定会同意我给出的所有结论（如果不是这样，我倒会觉得奇怪）。但如果你阅读本书并思索其中的每一个问题之后，那么几乎不会有人再能提出什么你未曾思考过的软件构建方面的问题了。

Self-Taught Programmers
自学的程序员

如果你没有受过太多的正规训练，本书正是你的良伴。每年约有 50 000 个新手进入这一专业领域（BLS 2004, Hecker 2004），但每年却只有 35 000 个人获得与软件相关的学位（NCES 2002）。从这些数据中我们可以很快得出一个结论——很多程序员并没有接受过软件开发方面的正规教育。在许多新兴的专业人员社群中都可以看到自学的编程人员——工程师、会计师、科学家、教师以及小公司的老板们；编程序是他们工作的一部分，但他们并不一定把自己看作是程序员。无论你在编程方面受过何种程度的教育，本手册都会让你能对各种行之有效的编程实践有深入的了解。

Students
学生

与有经验但缺乏正规培训的程序员对应的，是那些刚刚毕业的大学生。新近毕业的学生大多拥有丰富的理论知识，但却缺乏创建产品级的程序（production programs）的实践技术。关于编写优秀代码的实践知识，就像部落里祭祀仪式上的舞蹈一样，只能慢慢地从软件架构师、项目负责人、分析师以及更有经验的程序员那里传承下来。更多的时候，这些知识就是程序员个人反复的尝试和犯错后的结晶。本书则是这些缓慢、传统智慧传承方式的一种替代方案，它汇集了以往只能从他人经验中猎取和收集的大量实用的经验技巧和有效的开发策略。对于那些正在从学术环境转向专业环境的学生来说，这是一本必备的读物。

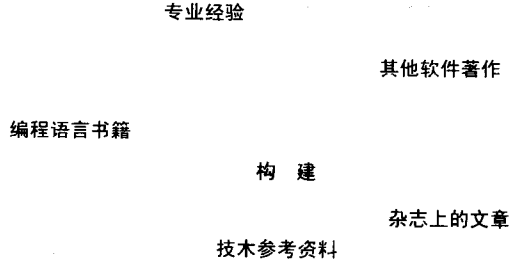
Where Else Can You Find This Information
还能从何处找到这些信息

本书综合整理了来自四面八方的多种软件构建技术。这些技术是软件构建领域长年累月积聚下来的智慧财富，它们不仅分散，而且其中大部分素材常年散落于纸面之外（Hildebrand 1989，McConnell 1997a）。其实，内行的程序员们所用的那些强大有效的编程技术并不神秘。但是这些内行人士面对手头日复一日紧张冲刺的项目，几乎没有谁花些时间和大家分享他们所学到的知识和技能。因此，程序员们可能很难找到

很好的关于编程的信息来源。

而本书所描述的技术则填补了入门图书和高级编程图书之间的空白。当你读过了《Java 编程入门》、《高级 Java 编程》和《高高级 Java 编程》之后，如果你还想学更多的编程知识，那还能读点什么呢？你可以阅读 Intel 或 Motorola 的硬件参考手册，阅读 Microsoft Windows 或 Linux 操作系统的函数手册，甚至是去阅读讲另外一门编程语言的书籍——你确实无法在一个缺乏这种详细参考资料的环境中使用语言或者程序。但本书是为数不多的探究编程本质的书籍之一。无论你在何种环境下、用何种语言编写程序，书中某些最有益处的编程技术都能派上用场。其他的书一般都忽略了这些实践知识，而这也正是本书专注于这些知识的原因。

本书中的信息是从许多来源中提炼出来的，如下图所示。想完全获得在本书中看到的这些信息的另外途径只有一条，那就是通读堆积如山的书籍和成百上千本技术期刊，还得再加上大量的实际经验。即便你把这些事情都做到了，本书仍然会对你很有益处，因为它把所有这些资料都集于一处，便于查阅。



Key Benefits of This Handbook
阅读本书的收益

无论你是何种背景，本书都能助你在更短的时间内写出更棒的程序，还不会那么头疼。

全面的软件构建参考　本书讨论了软件构建活动的方方面面，比如说软件的质量，还有编程的思维方式。它还会深入阐述构建活动中的重要细节，如创建一个类的步骤，使用数据和控制结构时的各种事项，还有调试、重构、代码调优的技术与策略等。你无须逐页通读所有主题。本书可以让你很容易就能找到感兴趣的特定话题。

随时备用的核对表　本书包括了大量的核对表（checklist），你可以用它来评估软件架构、设计方法、类和子程序的质量、变量命名、控制结构、代码格式、测试用例，等等。

与时俱进的信息 本书介绍了一些当今最为时兴的技术，其中有许多还未被广泛采用。正因为本书撷取了实践与研究两者的精髓，它所介绍的这些技术将经久不衰，受用多年。

以更广的视角检视软件开发 本书将给你一个机会，让你凌驾于日复一日、忙于救火的混乱场面之上，看看到底什么是可行的，而什么又是不可行的。实践中的程序员们很少有时间去阅读数以百计的书籍与期刊，而本手册萃取了其中的精华。本书所汇集的理论研究与实践经验将活跃你的思维，激励你对自己项目的思考，使你的行动更有策略，避免反复陷入完全一样的战斗。

绝不注水 有些软件书籍，其中精髓部分的净重也就 1 克，却注入了重达 10 克的水分。本书则会公平地探讨每项技术的优劣。关于你自己项目的特定需求，你了解得要比任何人都清楚。因此本书仅是给你公正客观的信息，让你能够具体情况具体分析，做出正确的决策。

有关概念适用于大多数常见的语言 本书中介绍的技术能让你可以更好地利用你的编程语言，无论是 C++、C#、Java、Visual Basic，还是其他类似语言。

丰富的代码示例 本书中收集了近 500 个用于展现优、劣代码之差异的示例。之所以给出这么多示例也是出于个人的偏好。因为从示例中我最能学到东西，我想其他程序员也该可以通过这种方式学得更好吧。

这些示例是用了多种不同的语言所写成的，因为学习并掌握不止一门语言通常是专业程序员职业生涯中的分水岭。一旦一名程序员意识到编程原则是超越特定语言语法的东西时，通往能够实质地改善编程质量并提高工作效率的知识的大门也就向他敞开了。

为了避免以多种语言写成的例子成为读者的负担，我会尽量避免使用各语言中那些深奥的特性——除非当时就是需要探讨它。为了弄懂一个代码片段要表达的问题，你无须完全理解所有的细枝末节。如果你集中关注示例所展示的问题，那么无论它是用什么语言写成的，你都能读懂。为让你更容易理解这些示例，我还给其中的关键部分加了注解。

引用其他信息来源 本书汇集了为数众多关于软件构建方面的可用信息，但这并不算完。在本书所有的章节中，“更多资源”一节都会介绍其他一些书籍和文章，你希望进一步深入了解感兴趣的话题时可以阅读它们。

cc2e.com/1234　**配套网站**　在本书的配套网站 cc2e.com 上会提供更新的核对表、参考书目、杂志文章、网页链接等内容。要访问《代码大全》（第 2 版）中的相关信息，请如本段文字左侧所示，在浏览器中输入“cc2e.com/”，后跟一个四位阿拉伯数字即可。这样的网址参考链接在本书中会有很多。

Why This Handbook Was Written
为什么要写这本手册

在软件工程界，人们都清楚地认识到，应该把软件开发中行之有效的实践知识归纳、编撰成一本开发手册。计算机科学与技术委员会（Computer Science and Technology Board）的一份报告指出，要想大幅提高软件开发的质量和工作效率，需要把已知的行之有效的软件开发实践知识归纳、统一并广为传播（CSTB 1990, McConnell 1997a）。该委员会还指出，传播这些知识的策略应建立在软件工程手册这个概念的基础之上。

The Topic of Construction Has Been Neglected
软件构建的话题常被忽视

曾几何时，软件开发和编写代码被认为是同一件事情。但随着软件开发周期中的各个活动被人们逐渐认识，该领域中一些最棒的头头脑脑们就开始花更多时间去分析和争论诸如项目管理方法、需求、设计、测试等问题了。在这场学习研究新兴领域的浪潮中，代码构建这个与软件开发骨肉相连的环节反而被忽视了。

关于软件构建的讨论之所以步履蹒跚，也是因为有人认为，如果将构建活动视作软件开发中的一项特定**活动**，就暗示着也必须把它视作其中的一个特定**阶段**。然而实际上，软件开发中的各项**活动**和各个**阶段**无须以特定的关系一一对应起来；而且无论其他的软件活动是分阶段（phase）进行、还是迭代式（interation）进行，或者以某种其他方式进行，都不妨碍我们探讨“构建活动”。

Construction Is Important
构建活动是重要的

构建活动被学者和作者所忽略的另一个原因是源于一个错误的观念，他们认为与其他软件开发活动相比，构建是一个相对机械化的过程，并没有太多可改进的机会。然而事实并非如此。

“代码构建”一般占据了小型项目 65%的工作量，而在中型项目上也达到了 50%。同时，“构建”也要为小型项目中 75%的错误负责，在中到大型项目上这一比例为 50%

到75%。任何一个要为50%到75%的错误负责的活动环节显然都是应该加以改善的。（第27章中对这些统计数据有更多详细的探讨。）

也有一些评论家指出，虽然构建阶段发生的错误在所有错误中占有很大的比例，但修正这些错误的代价往往比“修正那些由于需求和架构所导致的错误”要低很多，这也就暗示着构建活动因此不那么重要。诚然，修正由构建活动所导致的错误的代价比较低这一说法是正确的，但它也引起了误导——因为如果不修正这些错误，代价反而会高得令人难以置信。研究人员发现，软件中一些代价最为昂贵的错误，其罪魁祸首常常是一些小范围的代码错误，其代价甚至可以飙至上亿美元的程度（Weinberg 1983，SEN 1990）。可以用较低代价修正的错误，并不意味着这些错误的修正不重要。

人们忽视构建活动的另一种原因则颇具讽刺意味——就因为它是软件开发中唯一一项肯定能完成的活动。对于需求，人们可以自以为是而不去潜心分析；对于架构，人们可以偷工减料而不去精心设计；对于测试，人们可以短斤少两甚至跳过不做，而不去整体计划并付诸实施。但只要写出来的是程序，总归要进行构建活动，这也说明，只要改进软件构建这一环节，就一定对软件开发实践有好处。

No Comparable Book Is Available
没有可媲美的同类书籍

既然看到构建活动有着如此清晰的重要性，我曾相信，当我构思此书时已有人写过关于有效的软件构建实践的书籍了。对这样一本介绍如何有效地进行编程的书籍的需求是显而易见的，但是我却只找到很少几本关于软件构建这一题材的书，而且那些书也仅是涉及到这个话题的一部分罢了。有些书写于15年前，还是和一些深奥的语言——如ALGOL、PL/I、Ratfor以及Smalltalk等——紧密相关的。有些则是出自并不实际编写产品代码的教授之手。教授们写出来的技术内容对于学生们的项目而言还行得通，但他们通常不知道如何在完整规模的开发环境中施展这些技术；还有些书是为了宣传作者最新钟情的某种方法论，却忽略了那些被时间反复证明是行之有效的成熟实践技术的巨大宝库。

当艺术评论家聚在一起的时候，他们谈论的都是关于版式、结构以及意蕴之类的话题；而当真正的艺术家聚在一起的时候，他们谈论的则是到哪儿才能买到更便宜的松节油。
—*Pablo Picasso*（毕加索）

简而言之，我没有找到哪怕是一本试图归纳总结来自专家经验、业界研究以及学术成果的实践编程技术的书籍。关于这个话题的讨论要能和现今的编程语言、面向对象编程技术以及前沿的开发实践紧密结合。很明显需要有人写出一本这样的书出来，而他必须了解当今的理论发展水平，同时也编写过足够多的能反映实践状况的产品级代码。我把本书构思成关于代码构建活动的完整探讨——一个程序员给其他程序员写的书。