<<代码大全（第2版）>>

图书基本信息

书名：<<代码大全（第2版）>>

13位ISBN编号：9787121022982

10位ISBN编号：7121022982

出版时间：2006-3

出版时间：电子工业出版社

作者：[美] 史蒂夫·迈克康奈尔

页数：914

译者：金戈,汤凌,陈硕,张菲 译,裘宗燕 审校

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们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年的时间甚至更长（Raghavan和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把控更大型的项目，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》（第一版）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50000 个新手进入这一专业领域（BLS 2004, Hecker 2004），但每年却只有35000 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、传统的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。
本书写得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入时的技术，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、VisualBasic，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，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。
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》（第二版）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（ComputerScienceandTechnologyBoard）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（CSTB1990,McConnell1997a）。
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（Weinberg1983，SEN1990）。
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。
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。
AuthorNote 作者注 欢迎您对本书中所探讨的话题进行质询，例如您的勘误报告，或其他相关的内容。请发邮件与我联系，我的邮箱是stevemmc@construx.com，也可以访问我的网站：www.SteveMcConnell.com。

内容概要

代码大全(第2版)是著名Ⅱ畅销书作者、《IEEESoftware》杂志前主编、具有20年编程与项目管理经验的SteveMcConneU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。

作者简介

史蒂夫·迈克康奈尔（Steve
McConnell）被公认为软件开发社区中的首要作者和发言人之一。
他是Construx
Software公司的首席软件工程师。
他所编著的图书包括曾被《软件开发》杂志授予优异产品震撼大奖的《代码大全》和《快速软件开发》，以及《软件项目生存指南》等。

<<代码大全（第2版）>>

书籍目录

章节摘录

书摘　　相对于前面那个呆板的用写作所做的隐喻，一些软件开发人员则认为应当将创造软件想象成类似播种和耕作的情形。
你一次设计系统的一小部分、写出一段代码、做一点测试，并将成果一点点添加到整个系统中。
通过这种小步前进，你可以把每次可能遇到的麻烦减到最小。
　　有时候人们会用很糟的隐喻去描述一种很好的技术，此时需要保全这一技术，并去寻找更好的隐喻。
这个例子里的增量技术是很有价值的，但把它比作播种和耕作却非常糟糕。
　　“每次做一点”这个主意可能在某些方面与农作物生长类似，但把软件开发类比为耕作就很不贴切，也没有太多意义，而且我们很容易用下面即将介绍的更好的隐喻替代它。
人们也很难把耕作这个隐喻引申到“一次做一点事情”之外。
　　如果你认同耕作这种隐喻，就请想象一下图2—2的情况：你会发现自己谈论的是：对系统计划施肥、对细节设计疏果，并通过有效的管理土地来增加代码的产量，最终取得代码大丰收。
你还会说“轮种C++和大麦”，或者让土地闲置一年以增加硬盘里面氮肥的供应量。
　　软件耕作这一隐喻的弱点在于它暗示了人们将无法对开发软件的过程和方式进行任何直接的控制。
你在春天播下代码的种子，然后按照农历节气向土地佬儿许几个愿，你将会在秋天收获到丰盛的代码。
P14-15看《代码大全》闹程序革命！

媒体关注与评论

书评“《代码大全》第1版在我看来堪称软件工程领域的经典之作一而第2版则更棒!”——Ralph Johnson，伊利诺伊州立大学；《设计模式》(Design Patterns)作者之一“无论您是新手还是经验丰富的开发人员，《代码大全》(第2版)都能教会您思考编程的最佳方法。”——Jeffrey Richter(www．wintetlect．com)，((Microsoft NET框架实用编程》(Appfied Microsoft．NET FrameworkProgramming)作者“这本书是讲述软件构建的权威指南一准备孤身前往荒岛的程序员只要带上这本书就足够了。”——Diomidis Spinellis，《代码阅读方法与实践》(Code Reading：The Open Source Perspective)作者“Steve McConnell是一位既在一线实践，又能把其中奥妙讲个明白的少数人之一。”——John Vlissides，IBM研究院；《设计模式》(Design Patterns)作者之一“Steve McConnell比任何人都懂得如何构建软件；我们十分庆幸他能把其所有的深邃见解和实践经验写成这样一本重要而新颖的图书。”——“Visual Basic之父”Alan Cooper，《软件观念革命》(About Face 2．0)作者

编辑推荐

两届震撼大奖得主，数十年软件开发智慧，十二年前的经典，十二年后再铸辉煌！
《代码大全(第2版)》——Amazon全五星一致推荐！
开发者必读著作！
图书馆必备典藏！

名人推荐

“《代码大全》第1版在我看来堪称软件工程领域的经典之作——而第2版则更棒！”——Ralph Johnson，伊利诺伊州立大学；《设计模式》（Design Pattems）作者之一“无论您是新手还是经验丰富的开发人员，《代码大全》（第2版）都能教会您思考编程的最佳方法。”——Jeffrey Richter，《Microsoft.NET框架实用编程》（Applied Microsoft.NET Framework Programming）作者“这本书是讲述软件构建的权威指南——准备孤身前往荒岛的程序员只要带上这本书就足够了。”——Diomidis Spinellis，《代码阅读方法与实践》（Code Reading：The Open Source Perspective）作者“Steve McConnell是一位既在一线实践，又能把其中奥妙讲明白的少数人之一。”——John Vlissides，IBM研究院；《设计模式》（Design Patterns）作者之一”Steve McConnell比任何人都懂得如何构建软件；我们十分庆幸他能把其所有的深邃见解和实践经验写成这样一本重要而新颖的图书。”——“Visual Basic之父”Alan Cooper，《软件观念革命》（About Face 2.0）作者

版权说明