

# CODE COMPLETE

Second Edition

# 代码 2 大全

第 2 版 精华本

精选全书  
要点 KEY POINTS  
完整收录  
核对表 CHECKLIST  
开发必备  
参考手册

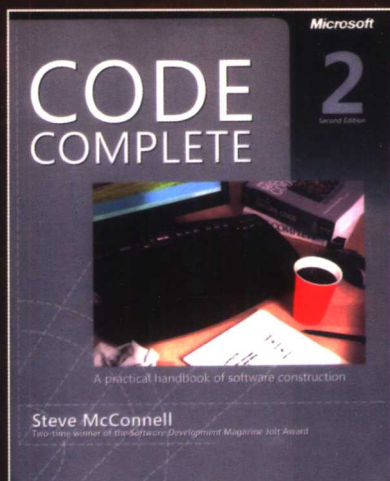
[美] Steve McConnell 著

两届

Software Development Magazine

Jolt Award

震撼大奖得主



金戈 汤凌  
陈硕 张菲  
裘宗燕

译

审校

软件构建之实践指南  
A practical handbook of software construction



电子工业出版社  
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY  
<http://www.phei.com.cn>

# 代 码 大 全

(第 2 版) 精华本

---

## 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

代码大全（第2版）精华本

---

# Microsoft Windows Internals

Fourth Edition:

Microsoft Windows Server 2003, Windows XP, and Windows 2000

[美] Mark E. Russinovich      著  
David A. Solomon              著  
潘爱民                              译

电子工业出版社

Publishing House of Electronics Industry

北京 · BEIJING

## 内 容 简 介

本书精选了《代码大全（第2版）》中的精华内容，包括各章“要点（Key Points）”以及“核对表（CHECKLIST）”的全部内容，便于读者在工作学习中随时查阅，极具参考价值。另外，本书还附有《深入解析 Windows 操作系统，第4版——Microsoft Windows Server 2003/Windows XP/Windows 2000 技术内幕》第14章的内容，供广大读者试读。本书适合计算机相关专业学生和教师、软件开发人员、IT 专业人员以及计算专业知识爱好者阅读和参考。

Copyright © 2006 by Microsoft Corporation. All rights reserved.

Original English language edition ©2004 by Microsoft by Steve C. McConnell.

All rights reserved.

Chinese Simplified Language Edition published by Publishing House of Electronics Industry.

Simplified Chinese edition published by arrangement with the original publisher, Microsoft Corporation, Redmond, Washington, U.S.A.

本书中文简体版专有版权由 Microsoft Corporation 授予电子工业出版社，未经许可，不得以任何方式复制或抄袭本书的任何部分。

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：精华本 /（美）迈克康奈尔（McConnell,S.）著；金戈等译.

—北京：电子工业出版社，2007.8

书名原文：Code Complete, Second Edition

ISBN 978-7-121-04618-6

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（2007）第 092414 号

责任编辑：陈元玉

印 刷：北京机工印刷厂

装 订：北京中新伟业印刷有限公司

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

开 本：720\*1000 1/16 印张：9.5 字数：120 千字

印 次：2007 年 8 月第 1 次印刷

定 价：15.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlt@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

## 《代码大全（第2版）》目录一览<sup>1</sup>

第1章	欢迎进入软件构建的世界.....	3
第2章	用隐喻来更充分地理解软件开发.....	9
第3章	三思而后行：前期准备.....	23
第4章	关键的“构建”决策.....	61
第5章	软件构建中的设计.....	73
第6章	可以工作的类.....	125
第7章	高质量的子程序.....	161
第8章	防御式编程.....	187
第9章	伪代码编程过程.....	215
第10章	使用变量的一般事项.....	237
第11章	变量名的力量.....	259
第12章	基本数据类型.....	291
第13章	不常见的数据类型.....	319
第14章	组织直线型代码.....	347
第15章	使用条件语句.....	355
第16章	控制循环.....	367
第17章	不常见的控制结构.....	391
第18章	表驱动法.....	411
第19章	一般控制问题.....	431
第20章	软件质量概述.....	463
第21章	协同构建.....	479
第22章	开发者测试.....	499
第23章	调 试.....	535
第24章	重 构.....	563
第25章	代码调整策略.....	587
第26章	代码调整技术.....	609
第27章	程序规模对构建的影响.....	649
第28章	管理构建.....	661

---

<sup>1</sup> 此目录页码为《代码大全（第2版）》页码。

第 29 章	集 成.....	689
第 30 章	编程工具.....	709
第 31 章	布局与风格.....	729
第 32 章	自说明代码.....	777
第 33 章	个人性格.....	819
第 34 章	软件工艺的话题.....	837
第 35 章	何处有更多信息.....	855

## 《代码大全（第2版）》核对表目录

核对表：需求.....	4
核对表：架构.....	7
核对表：前期准备.....	9
核对表：主要的构建实践.....	11
核对表：软件构造中的设计.....	14
核对表：类的质量.....	17
核对表：高质量的子程序.....	24
核对表：防御式编程.....	27
核对表：伪代码编程过程.....	30
核对表：使用数据的一般事项.....	33
核对表：变量命名.....	35
核对表：基本数据类型.....	39
核对表：使用不常见数据类型的注意事项.....	42
核对表：组织直线型代码.....	44
核对表：使用条件语句.....	46
核对表：循环.....	48
核对表：不常见的控制结构.....	50
核对表：表驱动法.....	52
核对表：控制结构相关事宜.....	54
核对表：质量保证计划.....	56
核对表：有效的结对编程.....	58
核对表：有效的详查.....	59
核对表：测试用例.....	62
核对表：关于调试的建议.....	65
核对表：重构的理由.....	69
核对表：重构总结.....	70
核对表：安全的重构.....	73
核对表：代码调整策略.....	75
核对表：代码调整方法.....	78

核对表：配置管理.....	83
核对表：集成.....	85
核对表：编程工具.....	88
核对表：布局.....	90
核对表：自说明代码.....	93
检查表：好的注释技术.....	95



## 《深入解析Windows操作系统，第4版》目录一览<sup>1</sup>

第1章	概念和工具.....	1
第2章	系统结构.....	35
第3章	系统机制.....	85
第4章	管理机制.....	183
第5章	启动和停机.....	251
第6章	进程、线程和作业.....	289
第7章	内存管理.....	375
第8章	安全性.....	485
第9章	I/O 系统.....	537
第10章	存储管理.....	615
第11章	缓存管理器.....	655
第12章	文件系统.....	689
第13章	网络.....	787
第14章	崩溃转储分析.....	845

---

<sup>1</sup> 此页码为《深入解析 Windows 操作系统，第4版》所对应的页码。

# 第1章 欢迎进入软件构建的世界

cc2e.com/0178 内容

- 1.1 什么是软件构建：第 3 页
- 1.2 软件构建为何如此重要：第 6 页
- 1.3 如何阅读本书：第 8 页

## 相关章节

- 谁应当阅读本书：前言
- 阅读本书的收益：前言
- 为什么要写这本手册：前言

你一定知道“构建（construction）”一词在软件开发领域之外的含义。“构建”就是“建筑工人（construction workers）”在建设一栋房屋、一所学校、乃至一座摩天大楼时所做的工作。在你年轻时，可能也曾用“硬纸板（construction paper）”构建过什么东西。按照一般的用法，“构建”是指建设的过程。构建过程可能包含有计划、设计、检查工作的一些方面，但在多数时候，“构建”就是指创建事物过程中动手的那些部分。

## Key Points 要点

- 软件构建是软件开发的核​​心活动；构建活动是每个项目中唯一一项必不可少的工作。
- 软件构建的主要活动包括：详细设计、编码、调试、集成、开发者测试（developer testing）（包括单元测试和集成测试）。
- 构建也常被称作“编码”和“编程”。
- 构建活动的质量对软件的质量有着实质性的影响。

代码大全（第2版）精华本

- 最后，你对“如何进行构建”的理解程度，决定了你这名程序员优秀程度——这就是本书其余部分的主题了。

## 第2章 用隐喻来更充分地理解软件开发

cc2e.com/0278 内容

- 2.1 隐喻的重要性：第 9 页
- 2.2 如何使用软件隐喻：第 11 页
- 2.3 常见的软件隐喻：第 13 页

### 相关章节

- 设计中的试探法：第 5.1 节中的“设计是一个启发式过程”

计算机科学领域中有着所有学科中最为丰富多彩的语言。你走进一间安全严密、温度精确控制在 20 °C 的房间，并在里面发现了病毒 (virus)、特洛伊木马 (Trojan horse)、蠕虫 (worm)、臭虫 (bug)、逻辑炸弹 (bomb)、崩溃 (crash)、论坛口水战 (flame)、双绞线转换头 (twisted sex changer)、还有致命错误 (fatal error) ……在其他领域中，你能得到这些吗？

这些形象的隐喻（比喻）描述了软件领域中各种特定的现象和事物。像这样生动活泼的隐喻还能够描述更加广泛的现象。借助这些隐喻，我们能更深刻地理解软件开发的过程。

本书其他章节的内容并不直接依赖在这一章中讨论的隐喻。因此，如果想直接学习实践方面的知识，你可以跳过本章不读；而如果你想更清楚地理解软件开发的过程，请读读这一章吧。

### Key Points

#### 要点

- 隐喻是启示而不是算法。因此它们往往有一点随意 (sloppy)。
- 隐喻把软件开发过程与其他你熟悉的活动联系在一起，帮助

代码大全 (第2版) 精华本

你更好地理解。

- 有些隐喻比其他一些隐喻更贴切。
- 通过把软件的构建过程比作是房屋的建设过程，我们可以发现，仔细的准备是必要的，而大型项目和小型项目之间也是有差异的。
- 通过把软件开发中的实践比作是智慧工具箱中的工具，我们又发现，每位程序员都有许多工具，但并不存在任何一个能适用于所有工作的工具，因地制宜地选择正确工具是成为能有效编程的程序员的关键。
- 不同的隐喻彼此并不排斥，应当使用对你最有益处的某种隐喻组合。

## 第3章 三思而后行：前期准备

cc2e.com/0309 内容

- 3.1 前期准备的重要性：第 24 页
- 3.2 辨明你所从事的软件的类型：第 31 页
- 3.3 问题定义的先决条件：第 36 页
- 3.4 需求的先决条件：第 38 页
- 3.5 架构的先决条件：第 43 页
- 3.6 花费在前期准备上的时间长度：第 55 页

### 相关章节

- 关键的“构建”决策：第 4 章
- 项目规模对“构建”及前期准备的影响：第 27 章
- 质量目标与构建活动之间的关系：第 20 章
- 管理构建：第 28 章
- 设计：第 5 章

在开始建造房子之前，施工人员会审视蓝图（包含所有细节

代码大全（第2版）精华本

信息的设计详图)，查看是否获得了全部（建筑）许可证，并测量房屋的地基。施工人员建造摩天大楼用一种方法，建造普通住宅用另一种方法，建造犬舍用第三种方法。无论何种项目，都会对“准备工作”进行剪裁，使之符合项目的特定需要；在构建活动开始之前，准备工作要做周全。

本章描述软件构建必须做的准备工作。就像修建建筑物一样，项目的成败很大程度上在构建活动开始之前就已经注定了。如果地基没打好，或者计划不充分，那么你在构建期间能做的无非是尽量让损害最小罢了。

工匠的谚语“瞄两次，切一次”（Measure twice, cut once/三思而后行）与软件开发中的构建部分有密切联系，构建活动差不多占整个项目成本的65%。最糟糕的软件项目最终会进行两三次（甚至更多）构建。将项目中最昂贵的部分执行两遍，这无论在软件行业还是在其他行业都是愚蠢的主意。

虽然本章是为成功的软件构建打地基，但是并没有直接讨论构建活动。如果你觉得自己是食肉动物，或者已经精通软件工程师的生命周期，那么请径直翻到第5章“软件构建中的设计”这块肥肉。如果你不打算动“为构建活动做前期准备”这个念头，那么请复习第3.2节“辨明你所从事的软件的类型”，认识一下如何将前期准备应用到你所处的情形。然后再关注一下第3.1节中的数据，这些数据描述了不做前期准备将会付出的代价。

cc2e.com /0323

## Checklist: Requirements

### 核对表：需求

这张需求核对表包含了一系列的问题——问问自己项目的需求工作做得如何。本书并不会告诉你如何做出好的需求分析，所以列表里面也不会有这样的问题。在开始构建之前，用这份列表做一次“心智健全”检查，看看你的地基到底有多坚固——用“需求里氏震级”来衡量。

并不是核对表中所有的问题都适用于你的项目。如果你做

的是一个非正式项目，那么你会发现有些东西根本就不需要考虑。你还会发现一些问题你需要考虑，但不需要做出正式的回答。如果你在做一个大型的、正式的项目，你也许就要逐条考虑了。

### 针对功能需求

- 是否详细定义了系统的全部输入，包括其来源、精度、取值范围、出现频率等？
- 是否详细定义了系统的全部输出，包括目的地、精度、取值范围、出现频率、格式等？
- 是否详细定义了所有输出格式（Web 页面、报表，等等）？
- 是否详细定义了所有硬件及软件的外部接口？
- 是否详细定义了全部外部通信接口，包括握手协议、纠错协议、通信协议等？
- 是否列出了用户想要做的全部事情？
- 是否详细定义了每个任务所用的数据，以及每个任务得到的数据？

### 针对非功能需求（质量需求）

- 是否为全部必要的操作，从用户的视角，详细描述了期望响应时间？
- 是否详细描述了其他与计时有关的考虑，例如处理时间、数据传输率、系统吞吐量？
- 是否详细定义了安全级别？
- 是否详细定义了可靠性，包括软件失灵的后果、发生故障时需要保护的至关重要的信息、错误检测与恢复的策略等？
- 是否详细定义了机器内存和剩余磁盘空间的最小值？
- 是否详细定义了系统的可维护性，包括适应特定功能的

变更、操作环境的变更、与其他软件的接口的变更能力？

- ❑ 是否包含对“成功”的定义？“失败”的定义呢？

### 需求的质量

- ❑ 需求是用用户的语言书写的吗？用户也这么认为吗？
- ❑ 每条需求都不与其他需求冲突吗？
- ❑ 是否详细定义了相互竞争的特性之间的权衡——例如，健壮性与正确性之间的权衡？
- ❑ 是否避免在需求中规定设计（方案）？
- ❑ 需求是否在详细程度上保持相当一致的水平？有些需求应该更详细地描述吗？有些需求应该更粗略地描述吗？
- ❑ 需求是否足够清晰，即使转交给一个独立的小组去构建，他们也能理解吗？开发者也这么想吗？
- ❑ 每个条款都与待解决的问题及其解决方案相关吗？能从每个条款上溯到它在问题域中对应的根源吗？
- ❑ 是否每条需求都是可测试的？是否可能进行独立的测试，以检验满不满足各项需求？
- ❑ 是否详细描述了所有可能的对需求的改动，包括各项改动的可能性？

### 需求的完备性

- ❑ 对于在开始开发之前无法获得的信息，是否详细描述了信息不完全的区域？
- ❑ 需求的完备度是否能达到这种程度：如果产品满足所有需求，那么它就是可接受的？
- ❑ 你对全部需求都感到很舒服吗？你是否已经去掉了那些不可能实现的需求——那些只是为了安抚客户和老板的东西？

cc2e.com /0337

## Checklist: Architecture

### 核对表：架构

以下是一份问题列表，优秀的架构应该关注这些问题。这张核对表的意图并非用做一份有关如何做架构的完全指南，而是作为一种实用的评估手段，用来评估软件食物链到了程序员这一头还有多少营养成分。这张核对表可用做你自己的核对表的出发点。就像“需求”的核对表一样，如果你从事的是非正式项目，那么你会发现其中某些条款甚至都不不用去想。如果你从事的是更大型的项目，那么大多数条款都会是很有用的。

#### 针对各架构主题

- 程序的整体组织结构是否清晰？是否包含一个良好的架构全局观（及其理由）？
- 是否明确定义了主要的构造块（包括每个构造块的职责范围及与其他构造块的接口）？
- 是否明显涵盖了“需求”中列出的所有功能（每个功能对应的构造块不太多也不太少）？
- 是否描述并论证了那些最关键的类？
- 是否描述并论证了数据设计？
- 是否详细定义了数据库的组织结构和内容？
- 是否指出了所用关键的业务规则，并描述其对系统的影响？
- 是否描述了用户界面设计的策略？
- 是否将用户界面模块化，使界面的变更不会影响程序其余部分？
- 是否描述并论证了处理 I/O 的策略？
- 是否估算了稀缺资源（如线程、数据库连接、句柄、网络带宽等）的使用量，是否描述并论证了资源管理的策略？



- 是否描述了架构的安全需求？
- 架构是否为每个类、每个子系统、或每个功能域（functionality area）提出空间与时间预算？
- 架构是否描述了如何达到可伸缩性？
- 架构是否关注互操作性？
- 是否描述了国际化/本地化的策略？
- 是否提供了一套内聚的错误处理策略？
- 是否规定了容错的办法（如果需要）？
- 是否证实了系统各个部分的技术可行性？
- 是否详细描述了过度工程(overengineering)的方法？
- 是否包含了必要的“买 vs. 造”的决策？
- 架构是否描述了如何加工被复用的代码，使之符合其他架构目标？
- 是否将架构设计得能够适应很可能出现的变更？

### 架构的总体质量

- 架构是否解决了全部需求？
- 有没有哪个部分是“过度架构/overarchitected”或“欠架构/underarchitected”？是否明确宣布了在这方面的预期指标？
- 整个架构是否在概念上协调一致？
- 顶层设计是否独立于用作实现它的机器和语言？
- 是否说明了所有主要的决策的动机？
- 你，作为一名实现该系统的程序员，是否对这个架构感觉良好？