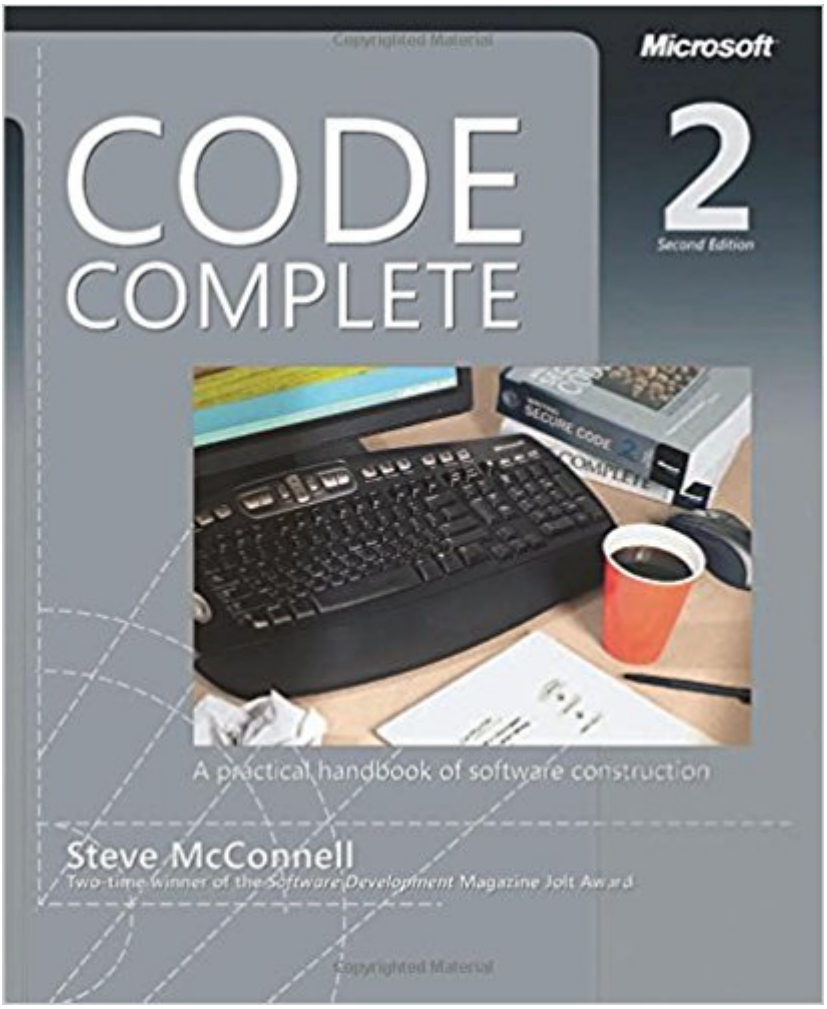
Code Complete: A Practical Handbook Of Software Construction, Second Edition



Synopsis

Widely considered one of the best practical guides to programming, Steve McConnellÃ¢â ¬â„¢s original CODE COMPLETE has been helping developers write better software for more than a decade. Now this classic book has been fully updated and revised with leading-edge practicesÃ¢â ¬â •and hundreds of new code samplesÃ¢â ¬â •illustrating the art and science of software construction. Capturing the body of knowledge available from research, academia, and everyday commercial practice, McConnell synthesizes the most effective techniques and must-know principles into clear, pragmatic guidance. No matter what your experience level, development environment, or project size, this book will inform and stimulate your thinkingÃ¢â ¬â •and help you build the highest quality code. Discover the timeless techniques and strategies that help you: Design for minimum complexity and maximum creativity Reap the benefits of collaborative development Apply defensive programming techniques to reduce and flush out errors Exploit opportunities to refactorÃ¢â ¬â •or evolveÃ¢â ¬â •code, and do it safely Use construction practices that are right-weight for your project Debug problems quickly and effectively Resolve critical construction issues early and correctly Build quality into the beginning, middle, and end of your project

Book Information

Paperback: 960 pages

Publisher: Microsoft Press; 2nd edition (June 19, 2004)

Language: English

ISBN-10: 0735619670

ISBN-13: 978-0735619678

Product Dimensions: 7.3 x 2.1 x 8.9 inches

Shipping Weight: 3.3 pounds (View shipping rates and policies)

Average Customer Review: 4.6 out of 5 stars 463 customer reviews

Best Sellers Rank: #5,175 in Books (See Top 100 in Books) #1 inÃ Â Books > Computers & Technology > Networking & Cloud Computing > Data in the Enterprise > Client-Server Systems #3 inÃ Â Books > Textbooks > Computer Science > Software Design & Engineering #5 inÃ Â Books > Computers & Technology > Business Technology > Software > Enterprise Applications

Customer Reviews

Steve McConnell is recognized as one of the premier authors and voices in the development community. He is Chief Software Engineer of Construx Software and was the lead developer of

Construx Estimate and of SPC Estimate Professional, winner of Software Development magazine's Productivity Award. He is the author of several books, including Code Complete and Rapid Development, both honored with Software Development magazine's Jolt Award.

The tragedy for books that become classics is that there are many more people who have heard of them (or perhaps also bought them) than people who have read them. In this case, the fact that Steve McConnell's "Code Complete" is approximately 900 pages long doesn't help, either. Even so, this is a book that not only deserves to be read, but also rewards multiple readings.The Good: McConnell deserves credit for writing the first (and only?) readable encyclopedia of best practices on software quality, covering topics such as how to build classes, use data and control structures, debug, refactor, and code-tune. Yes, it would be nice if the book was updated to include substantive material on languages like Ruby or Python (cf. p. 65, Python "also contains some support for creating larger programs") but, in the words of Gertrude Stein, "Not everything can be about everything" -- though Code Complete does come pretty close. This book contains an astonishing number of practical points on a variety of topics. Here is a quasi-random selection: a) don't use booleans as status variables (chs. 5, 12), b) when you feel the need to override a function and have it do nothing, don't; refactor instead (ch. 6), c) when choosing variable names, avoid homonyms (ch. 11), d) if you decide to use a goto, indenting your code properly will be difficult or impossible (ch. 17), e) trying to improve software quality by increasing the amount of testing is like trying to lose weight by weighing yourself more often (ch. 22), f) make your code so good that you don't need comments, and then comment it to make it even better (ch. 32), and finally the oft-repeated g) you should try to program into your language, not in it (ch. 34). McConnell also sprinkles the text with classic words of wisdom, e.g. "The competent programmer is fully aware of the strictly limited size of his own skull" (Edsger Dijkstra), "Never debug standing up" (Gerald Weinberg), "Copy and paste is a design error" (David Parnas), "Any fool can defend his or her mistakes -- and most fools do." (Dale Carnegie). It is important to point out that even though this volume is encyclopedia-like, it does have both a sense of humor (e.g. "the encryption algorithm is so convoluted that it seems like it's been used on itself") and a clear authorial voice (e.g. "Though sometimes tempting, that's dumb."). Another example of the latter: in ch. 33, after quoting Edward Yourdon at length, McConnell adds "This lusty tribute to programming machismo is pure B.S. and an almost certain recipe for failure".The Bad: overall the writing is very good, but the occasional infelicity reminds us that McConnell is human (e.g. p. 369 "A loop-with-exit loop is a loop in which", p. 809 "A program contains all the routines in a program."). In a technical book of this breadth, minor mistakes are

bound to creep in. For example, in ch. 10 McConnell mentions the different possible levels of a variable's scope in C++, and then adds that in Java and C# one can also use namespaces, thus effectively ignoring the existence of the namespace concept in C++ (which is baffling, given that he then discusses precisely that topic in ch. 11). Another example, this one more serious, is McConnell's recommendation that you should use a pointer - not a reference - if you want to pass by reference in C++ (ch. 13), something which is contrary to C++ best practices (see e.g. Sutter & Alexandrescu, "C++ Coding Standards", Item 25). A less technical point: in ch.2 McConnell criticizes Frederick Brooks for writing (in 1975): "Plan to throw one away; you will, anyhow". I found this to be bizarre, given that in the 1995 edition of "The Mythical Man-Month" Brooks states in no uncertain terms that he has changed his mind on this: "This I now perceive to be wrong" (p. 265). Given that Code Complete 2 was published nearly 10 years later (in 2004), criticizing Brooks for his publicly repudiated former opinion seems improper. On a different note, although some of the on-line accompanying material is fascinating (e.g. the links to the original Dijkstra and Lawrence articles in ch. 17) many of the links are just electronic versions of McConnell's checklists or bibliographies, while some are simply disappointing. To name only a couple of these, as of this writing the link on p. 856 on the economics of XP is a dead link, while the one on p. 76 is downright embarrassing (it links to a google search for "emergent design"). Finally, even though the book has a dedicated website, no list of errata is provided there. If you dig deeper, you can find one on the O'Reilly website, but that is woefully inadequate, e.g. it contains no information on separate printings.The most common criticism one hears about this book is that any decent software developer should already know the material covered in it. Ironically enough, this is true. To quote Dr. Johnson: "People need to be reminded more often than they need to be instructed".Alex Gezerlis

I don't think yet another long review is required for this book because I consider it among the standard books that a real-world software developer should have read. To put it another way, any software developer who already internalized all of the wisdom and techniques in that book can be considered a pretty good developer and / or software team manager.But be careful, this book is not a how-to book and does not focus on any specific technology. You can be an ASP.NET developer working on MS Windows platform or a Python developer working on a GNU/Linux platform. If you are involved with a project that includes more than a few thousands line of code and you work as a part of a team then you owe yourself to have the knowledge described in the book.I think I'll create a small file that includes all the checklists that are given at the end of the chapters as well as the further reading lists and send a copy of that every junior software developer I work with.

I consider this to be an encyclopedia to starting out coding. The other Microsoft book, Software Requirements is a more in depth look at coding. This book outlines the steps for constructing and maintaining code, but is not really geared towards quality issues and management, which I consider more important and a prerequisite to these topics. It is a complete reference for anyone needing an introduction.

The most revealing comment in the book is that when Steve McConnell sat down to write it, his initial impression was that it would be approximately 250 pages, yet when he set out to plan it in detail, he realised it would be 800 pages. As a result, the book consists of one third genuinely insightful material. Everything is repeated at least once, adding another third to the length. And there is another third of the book that can be considered "filler" - topics covered very lightly, so that the book will live up to the title "Complete". Specifically, Chapters 1-5, half of chapter 6, and chapters 27-33 could easily have been omitted. The result would have been something of power of Scott Myers Effective C++, a book that is frequently referenced and clearly admired by Steve McConnell. Rather than read these superfluous chapters, the reader would be advised to head straight for the books and articles given as further reading in these sections. As a general rule, pages which contain code are the ones worth reading.However, I think that the 250 pages of real content is exceptional and all serious programmers should read this book at some point, particularly if self-taught or have never worked on a large project before. The book collects a huge body of programming lore, that is often rediscovered or passed on as superstition between programmers. The focus, always, is on readability and robustness, which are easily forgotten in the rush to code. Overall, the advice in this book is clear, practical and good. A particular highlight is the gathering of research on where time is wasted in coding and how it can be avoided. Inevitably it is most valuable for those relatively new to programming. But even experienced programmers need reminding about many of the ideas every now and then.I don’t agree with everything McConnell suggested. The book is happy to recommend the long parameter lists and verbose variable names typical in Windows and Java programming. There is also a clear bias towards the Microsoft environment and Visual Basic in particular (comprising about a quarter of examples) . The somewhat rigid and overwrought UML style of structuring object-oriented code is advocated over more flexible layered approaches. Nevertheless, I think that there is nothing that is clearly bad, and if followed rigorously would improve the quality of almost any project.At 800 pages, it is a long read and arduous at times, but in the end well worth it and the best parts are wonderful.

Code Complete: A Practical Handbook of Software Construction, Second Edition Software Engineering: The Current Practice (Chapman & Hall/CRC Innovations in Software Engineering and Software Development Series) Construction Contract Dispute and Claim Handbook, Introduction, and Division 01: A Primer on the Nature of Construction Contract Disputes for Attorneys, ... (Construction Contract Dispute Handbook) Agile Project Management: Agile Revolution, Beyond Software Limits: A Practical Guide to Implementing Agile Outside Software Development (Agile Business Leadership, Book 4) Dwelling Construction Under the 2007 California Building Code, Revised Edition (International Code Council Series) McGraw-Hill's National Electrical Code 2017 Handbook, 29th Edition (Mcgraw Hill's National Electrical Code Handbook) McGraw-Hill's National Electrical Code (NEC) 2017 Handbook, 29th Edition (Mcgraw Hill's National Electrical Code Handbook) 2016 National Construction Estimator (National Construction Estimator) (National Construction Estimator (W/CD)) 2012 International Plumbing Code (Includes International Private Sewage Disposal Code) (International Code Council Series) Building Code Basics: Commercial; Based on the International Building Code (International Code Council Series) National Electrical Code 2014 Handbook (National Electrical Code Handbook) National Electrical Code 2008 Handbook (National Electrical Code Handbook) National Electrical Code 2002 Handbook (National Electrical Code Handbook) McGraw-HillÃ¢â ¬â„¢s National Electrical Safety Code 2017 Handbook (Mcgraw Hill's National Electrical Safety Code Handbook) McGraw-Hill's National Electrical Code 2011 Handbook (McGraw-Hill's National Electrical Code Handbook) Illustrated 2009 Building Code Handbook (Illustrated Building Code Handbook) Code Check Complete 2nd Edition: An Illustrated Guide to the Building, Plumbing, Mechanical, and Electrical Codes (Code Check Complete: An Illustrated Guide to Building,) Clean Code: A Handbook of Agile Software Craftsmanship The Mining Construction Handbook: Your Complete Guide to Minecraft Construction The Software Requirements Memory Jogger: A Pocket Guide to Help Software And Business Teams Develop And Manage Requirements (Memory Jogger)