Book Reviews

By Joel Spolsky
Wednesday, March 13, 2002
Printer Friendly Version

> *"Pretty close to the perfect short list for any programmer"* — Jan Derk

You can learn a lot about somebody by the books they've read. And I've always thought that if you read all the same books I read, you'll come to think like me, too.

So here it is -- Joel's Programmer's Bookshelf. This is the short list of all the books that I honestly think that every working programmer needs to read, with my own book hidden in there in case you didn't notice because I get about two bucks if you buy it.

Painless Software Management

Peopleware: Productive Projects and Teams
Tom Demarco and Timothy R. Lister
As summer interns at Microsoft, my friends and I used to take "field trips" to the company supply room to stock up on school supplies. Among the floppy disks, mouse pads, and post-it notes was a stack of small paperback books, so I took one home to read. The book was Peopleware, by Tom DeMarco and Timothy Lister. This book was one of the most influential books I've ever read. The best way to describe it would be as an Anti-Dilbert Manifesto. Ever wonder why everybody at Microsoft gets their own office, with walls and a door that shuts? It's in there. Why do managers give so much leeway to their teams to get things done? That's in there too. Why are there so many jelled SWAT teams at Microsoft that are remarkably productive? Mainly because Bill Gates has built a company full of managers who read Peopleware. I can't recommend this book highly enough. It is the one thing every software manager needs to read... not just once, but once a year.

The Mythical Man-Month
Frederick P. Brooks
Certainly one of the classics of software project management, this book first appeared a quarter of a century ago, when Fred Brooks tried to run one of the first very large scale software engineering projects (the OS/360 operating system at IBM) and became the first person to describe how radically different software is from other types of engineering. This book is most famous for discovering the principle that adding more programmers to an already-late project makes it later, but that's only the tip of the iceberg. Understanding this book is a prerequisite for thinking correctly about managing software teams.

Rapid Development: Taming Wild Software Schedules
Steve C. McConnell
Steve McConnell captures a lot of the development management ideas that Microsoft figured out in their first decade or so of developing software on a large scale. You'll see a lot of overlap between the ideas in this book and my wholly unoriginal Joel Test (surprise, surprise), although the emphasis here is on getting control of the scheduling process.

Code Craftsmanship

Code Complete: A Practical Handbook of Software Construction
Steve C. McConnell
The encyclopedia of good programming practice, Code Complete focuses on individual craftsmanship -- all the things that add up to what we instinctively call "writing clean code." This is the kind of book that has 50 pages just talking about code layout and whitespace.

The Pragmatic Programmer: From Journeyman to Master
Andrew Hunt and David Thomas
This is a great book for programmers who have learned the mechanics of programming, maybe in college, but don't quite feel secure deciding what to do. It's like the difference between drafting and architecture. What you learned in that class in college was drafting, and you can draw beautifully, but if you still feel like you wouldn't quite know where to begin if someone told you to write a P2P music-swapping network all by yourself, this is the book for you.

Philosophy of Programming

Microserfs
Douglas Coupland
Here's an important thing to understand about working at Microsoft right out of college. You are young. You are in a new city. You don't know anybody. There's nothing to do, and you're a computer geek, and the fun toys are at work, so chances are, after getting your take-out dinner at the Taco Time driveup counter, you'll just be bored so you'll go back to your plush office with a view of mountains and 100 foot evergreens and code. For many of these young programmers life outside of work is pretty lonely and empty, which works great for Microsoft, because you put all your energy into the really fun part of the day, developing cool software.

Nothing quite captures the feeling of being a young programmer at a big software company as well as Microserfs. Douglas Coupland's portrayal of life at Microsoft in the early 90s was so stunningly on-target it floored me -- but then he went further and provided a moral and ethical understanding of what was going on that hadn't quite occurred to anybody. Nobody understands the emptiness, the banal loneliness, and the quest for personal connection of modern age North America like Coupland.

Zen and the Art of Motorcycle Maintenance
Robert M. Pirsig
Some people's attitude towards programming is that it's a nifty way to pay the bills. For others, that's not enough... our work is a significant a part of our lives, and we need a philosophical understanding to make sense of it. This book goes a long way towards relating engineering and philosophy.

Godel, Escher, Bach: An Eternal Golden Braid
Douglas R. Hofstadter
I read this before starting college, and decided that I wanted to major in "Godel Escher Bach." My primary criterion for choosing courses was to cover topics that were raised in this magnificant, panoramic, and brilliantly interesting book: AI, cognitive science, computer science, philosophy, psychology, music, and art are woven together magically.

A Pattern Language: Towns, Buildings, Construction
Christopher Alexander, Sara Ishikawa, Murray Silverstein, et al.
Yeah, OK, it's a book about architecture. You know, buildings and stuff. I don't think there's a single mention of computers in the whole book. I bought it because I'm interested in architecture. Then I noticed something: almost everything in the book can be applied to the work we do as software designers. For example, the splash screen in CityDesk is based on the highly influential pattern of Zen View. Dave Winer's Radio Userland appeals to people because it follows the pattern of Windows Overlooking Life. Software that understands the Hierarchy of Space pattern is easier to comprehend.

A similar and somewhat short-lived movement was fashionable in programming a few years ago; I think the patterns movement in programming never quite took off because it was an attempt to copy the form of this book rather than the wisdom of this book.

User Interface Design

User Interface Design for Programmers
Joel Spolsky
I might as well plug my own book, right?

UI for Programmers is my attempt to teach what I consider the top level, most important principles of UI design that every programmer needs to know. The most common reaction I've heard from readers is "after reading your book, I found three things I just HAVE to change in my program."

Don't Make Me Think
Steve Krug
Don't Make Me Think is an excellent and entertaining book on UI design for the web. Finally, a book that tries to understand the *principles* of good UI design, not just the mundane rules (like "don't change the colors of links"). Steve Krug's primary thesis is that the less you make people think, the easier your site will be to use. "What a waste it is to lose one's mind. Or not to have a mind is being very wasteful. How true that is."

About Face: The Essentials of User Interface Design
Alan Cooper
A classic of UI design, this is a great bible of GUI design from the inventor of Visual Basic.

Design of Everyday Things

Donald A. Norman
Donald Norman's classic The Design of Everyday Things (also published under the name "The Psychology of Everyday Things") is one of the best books on "UI design", even though it talks more about doors and and refrigerators than computers. This was a groundbreaking work for its theory of *affordances* which I talk about it chapter 4 of my UI book, which remains one of the most influential ideas in good design of everyday objects.

Designing Web Usability
Jakob Nielsen
I know, I know, poor Jakob gets a bad rap for his bizarre pronouncements like "ClearType ... can save users $2,000 per year" and "Micropayments are the answer." Um, right. Still, among all the silly math, Nielsen comes up with a stunning number of *exactly right* notions that you just need to know. If you're doing any kind of web design, you need to know the principles in this book. If you're doing non-web design, think of this is an excellent case study in usability engineering.

Capitalism for Programmers

A Random Walk Down Wall Street
Burton Gordon Malkiel
If you spend enough time in this industry it's almost impossible to avoid suddenly finding yourself with a big pile of money that you are going to have to manage somehow. And if you don't want to screw it up, you need to know a few things.

Oh, but it all seems so *complicated*, you say. How can you ever outsmart the wily foxes on Wall Street who are out to rip off the common folk? It seems like doing even reasonably well at investing should take constant research, studying, working, reading, learning. All those Annual Reports. And you'll have to subscribe to all kinds of dense boring newspapers with columns and columns of tiny print.

What if I told you that you could read *one book* and know everything you are going to need to know about managing your investments? And I mean, everything. Well, it's true. And this is the book. If you can't be bothered to read *anything else* about investing, read this one book.

Striking Out On Your Own

Growing a Business
Paul Hawken
People regularly email me and say, "gosh, I love your theory about starting a company the Ben and Jerry's way, but, how do I get started?" This is the book you want to read. It's a little bit lite 'n' fluffy but it does give you the philosophy of growing a company organically.

Graphic Design

The Non-Designer's Design Book
Robin Williams
Wow! Everybody has to do some graphic design, and not every software team has the luxury of professional designers. This excellent, thin book will give you a grasp of the principles behind page layout, fonts, etc. The good news is, you can read it in the bath before the water gets cold, and the next day, your dialog boxes and powerpoints and web pages will start looking better.

Making a Difference

Influence: The Psychology of Persuasion
Robert B. Cialdini
Another book worth reading and re-reading is Robert B. Cialdini's classic Influence. When charitable organizations send you a request for a donation, they almost always include a "gift" in the envelope. Sticky labels with your address on them. Or a couple of blank greeting cards. The reason they're giving you the gift is because of the social principle of *reciprocity*; now you will feel obliged to give something back. You've probably heard the expression "hurry, supplies are limited!" so many times in television advertisements that it hardly registers any more. But it's there because of the principle of *scarcity*; your natural assumption that something that is scarce is worth more money. These tricks, among others, are used by salespeople, marketers, and advertisers to influence people to behave in a certain way. Cialdini's excellent book discusses the psychological theories behind the science and practice of influencing the behavior of other people. Read it before they do!

Helplessness: On Depression, Development, and Death
Martin E. P. Seligman

A few months ago when we released CityDesk, I got an email from a customer complaining that he was used to doing Alt+F, Alt+S to save files. Unfortunately due to a tiny, unnoticed bug, that keyboard shortcut saved the file *and then closed it*, irritatingly. I had never noticed because I'm in the habit of doing Alt+F,S to save files, not Alt+F,**Alt**+S -- a tiny difference -- and Alt+F,S worked fine.

Once you get into the habit of doing Alt+F,Alt+S to save, it becomes so automatic you don't *think* of it as Alt+F,Alt+S. You think of it as *save*. And when you push the "save" button in your brain and the file you were working on goes away, it makes you feel like you're not in control of your environment. It's a small thing, but about the fourth time that it happens, you're going to be seriously unhappy. That's why I spent several hours tracking down this bug and fixing it. In a bizarre application of Murphy's Law, this fix led to a cascade of events that caused us to waste something like a week, but that's neither here nor there. It was worth the time spent. This is what it means to be concerned about usability. If you still think that something as small as how long you hold down the Alt key when you active a menu command doesn't matter, well, your software is going to make people unhappy. These tiny inconsistencies are what makes Swing applications so unbearably annoying to use, and in my opinion it's why there are virtually no commercially successful Java GUI applications.

I say time and time again that the reason good UI design matters is that it makes people happy. I mean that literally. If your UI design is good, the people who use your software will be *happier*. If it's bad, they will be *unhappy*.

What does this have to do with a book on depression? Well, it turns out that people literally become clinically depressed when they feel like they can't control their lives and their environment. And Seligman, a pioneer in the field, has found that one of the most effective known non-drug therapies for depression is encouraging people to take small steps to exert control over their environments.

I Wanna Be A Programmer

A couple of books for my non-programmer readers who want to learn about programming.

Code
Charles Petzold
This book is explicitly NOT for programmers, it's for all those non-programmers who either want to become programmers, or want to understand what programmers do, or just want to explore the weird world of bits and bytes. Start by reading this book. If you find it fascinating and think you want to become a programmer, read (and work through) The C Programming Language next.

The C Programming Language
Brian W. Kernighan and Dennis M. Ritchie
One of the most misguided ideas in programming pedagogy is the idea that you have to seduce people into programming by starting with simple, fun, graphical stuff. Some people think that the best way to learn programming is to start with HTML, maybe, and then learn how to cut and paste some javascripts, and then move on. Another misconception is that you should start with a trendy, marketable programming skill like Java or Web Database Programming.

Well, those people are wrong.

For various reasons too complicated to go into here, I believe that you have to start programming at a level that is as close to the machine as reasonable. I think that this book, universally known as K&R, is THE book anyone who wants to be a programmer must learn first. Pick it up and work through it in detail. If you love every minute of it, you can be a programmer. If you find this old-school text programming stuff boring, or the pointer stuff drives you crazy, trust me, you're not going to like programming very much. If you need to be seduced into programming or if you don't have the patience to figure out what all those crazy asterisks mean, you're going to be happier doing something else. Really. But if you can make it through this book by yourself, you've got what it takes to be a top gun programmer, and you've got a terrific foundation for everything else you're going to learn.

Some discussion from readers of their favorite books. Anything else that should be here? Post a note to the discussion group!