book reviews

ELIZABETH ZWICKY, WITH BRANDON CHING AND SAM STOVER

HACKERS: HEROES OF THE COMPUTER REVOLUTION

Steven Levy

O'Reilly, 2010. 487 pp.
ISBN 978-1-449-38839-3

I am profoundly conflicted about this book. It is a (mostly) sympathetic portrait of people who did interesting and often under-appreciated stuff in the early days of the PC and just before and after. It gives a clear picture of their value to the world and their values, which are not always mainstream, and it presents them as lovable if quirky. In the process, it often accepts their worldview uncritically. This is a community I am a part of, at least on the edges, and I like to see it appreciated. I do not think that uncritically accepting it is healthy. There is a fine line between celebrating difficult people and making it seem like you have to be anti-social to do any important work in computing, and I don't think this book always manages to stay on the good side of that line.

It is extraordinarily difficult to write books about real people, particularly big groups of real living people. Being true and honest and kind and inclusive without muddying up the story beyond all comprehensibility is an immense balancing act, and it's a balancing act this book manages really pretty well. The result does end up omitting a fair amount of important stuff that is present in hacker culture. This is a book about monogamous straight white guys, for instance, and while the culture itself is overwhelmingly white guys, it's not nearly so overwhelmingly straight or monogamous. That could be an accident of the people chosen, or it could be an artifact of trying to be polite, by some standards of politeness, particularly that current at the time the first edition was written. The book also ignores the fraught territory of deciding where the line is between merely being eccentric and actually having a condition that might be detrimental to your daily life to the extent where medical assistance could be very helpful. Again, this is difficult, private stuff, but, you know, much of one chapter is devoted to a multi-person quest for a single programmer to "get laid," so it's not as if the book confines itself to discussing public matters.

I didn't read it when it first came out, because it was a book that many people I knew loved and that I was pretty sure I'd hate for fundamentally feminist reasons. I didn't need to be fighting with my friends over it, but early in my career my feelings about being a woman in computing were pretty raw, for very good reason. And, indeed, even all these decades later, I gritted my teeth through most of the early chapters. It's not that women are never mentioned. No, that would actually be an improvement. Instead, the possession of a girlfriend or wife is used as a kind of indicator of level of socialization; if you have a woman, you are socially competent. If you don't, you aren't. (This is not, in my experience, accurate, in either direction.) And one of the interviewees presents the concept that women are so underrepresented as hackers that there must be a genetic cause, which is exactly as stupid as saying that Macintosh viruses are so rare that there must be a hardware cause. (In case you're wondering, Macintosh viruses are rare mostly for cultural/economic reasons—PC viruses pay better—and to some extent for software reasons.) Culture and genetics interact in complicated ways, but culture is vastly influential, as important to human behavior as software is to computer behavior.

But despite my teeth-gritting, and my very complex feelings about the book, I think it is a good book—as honest as it can probably get about what its chosen tiny part of computing history is about. As a slice of time, and as an introduction to the idea that really weird people can, in fact, end up reshaping the world, it's a gripping book.

If you want to know about computing history, do add in other books that cover more threads in the

story; this one leaves UNIX off at Multics and picks it back up again roughly at Linux, which is a whole other tapestry taking place overlapping with the events described here. Peter Salus's *A Quarter Century of UNIX* is a very, very different book from *Hackers*, but it fills in some of that missing territory.

TEXTMATE: POWER EDITING FOR THE MAC

James Edward Gray II

Pragmatic Bookshelf, 2007. 182 pp.
ISBN 978-0-9787392-3-2

OK, enough with the sociological stuff. TextMate is a Macintosh editor in the spirit of Emacs, which is to say it is powerful, flexible, extensible, and programmer-friendly. (Also, it has a lot of Emacs key-bindings.) It's easy to overlook, but capable of many beautiful feats. Our Macs at work come with it installed, and although I'm no TextMate wizard, I use it, enjoy it, and have converted some of my colleagues. Some of them have been converted by cheap tricks (the impenetrable thicket of machine-generated JSON which we pasted into TextMate and had it auto-format as XML; auto-format is hardly innovative, but it saved our day) and some by actual elegance. One colleague was watching me write HTML and insert multiple links; the first one I didn't have the URL for, and TextMate did a nice job of setting things up for me, but nothing impressive. For the second one, I copied the URL from my browser first, and when I told TextMate to insert a link, it did the whole thing, fetching it from the paste buffer, looking up its title and inserting that as the link text, and preselecting it in case I wanted other linked text. At that point he said, "What is that, and where do I get it?"

TextMate has a number of tricks which are either automatic or easy to find, but fully getting your head around controlling, as the author phrases it, the team of magic ninjas it provides is a slow process which this book is very helpful with. If you find yourself using TextMate, you'll find the book helpful, and you will almost certainly want a copy if you're trying to extend your use of it.

THE JAZZ PROCESS: COLLABORATION, INNOVATION, AND AGILITY

Adrian Cho

Addison Wesley, 2010. 279 pp.
ISBN 978-0-321-63645-4

In general, the metaphors people use for thinking about business are based on sports or on war. These have several risks. For instance, sports metaphors translate poorly between cultures or to non-enthusiasts. Often a sports-unenthused audience may have only the vaguest idea what the speaker means. War metaphors often result in people being metaphorically exhorted to kill the customer or die in the attempt, which is not apt to result in a productive relationship.

I was therefore happy to find a book that used a metaphor which is inherently based in collaboration and pleasing the audience, rather than attacking it. (And amused to discover that the author likes to translate his insights into sports and war metaphors, just in case you needed something more familiar and macho. The temptation to stray back into social commentary is very high here.) Aside from appreciating the base metaphor, I also like the content a lot. It includes a number of insights that are common (hire good performers, enable them to lead as needed, watch the health of the team) and some that are not. The discussion of feedback loops and "hunting" is particularly rare. Even rarer, it discusses the good and bad points of its advice, instead of merely advocating One True Way.

This is a good, general theoretical overview of how to manage a team. It advocates a collaborative hacker-friendly management style that values excellence and transparency. It is, however, relatively abstract. It's a good conceptual basis, and usefully provides the metaphors and explanations you'll need to communicate this kind of management style to other managers. But it won't give you the nitty-gritty of how to implement it.

PERSUASIVE TECHNOLOGY: USING COMPUTERS TO CHANGE WHAT WE THINK AND DO

B.J. Fogg

Morgan Kaufmann, 2003. 265 pp.
ISBN 978-2-55860-643-2

When you think about it, most of us do, in fact, use computers as persuasive systems. System administrators try to convince users to do all sorts

of things, from setting good passwords to submitting useful trouble tickets. Programmers want people to use their programs, register them, and file bug reports when they break, each of which requires persuasion. We use online dieting or exercise tools to try to persuade ourselves. And the book gives the example of an oscilloscope that increased its user satisfaction ratings significantly by rewording the error messages, which were brusque and annoying.

This book also covers a bunch of territory that may not be immediately obvious when you think about persuasion. For instance, it gives an interesting breakdown of roles that a computer takes in interactions, and ways that software behavior can work for and against those. (Social behavior from the computer is great! Except when you're trying to use it as a tool, at which point it should not be waving and offering helpful advice.) It also talks a lot about trust, which is necessary for persuasion but of course desirable for all sorts of other reasons.

There's a good bit of interesting information here, but at an academic book price and with a relatively slow start as it puts together an academic framework. If you're looking for specific instructions, this book isn't going to satisfy you (but then again, you don't have any other options; this is as techy as books about persuasion get).

NETWORK FLOW ANALYSIS

Michael W. Lucas

No Starch Press, 2010. 224 pp.
ISBN 978-1593272036

REVIEWED BY SAM STOVER

Having read and loved *Absolute OpenBSD* by this same author, I was really looking forward to *Network Flow Analysis*. Definitely not a disappointment, this book was exactly what I was hoping for. Combining a great writing style with lots of technical info, this book provides a learning experience that's both fun and interesting. Not too many technical books can claim that.

Chapter 1 describes flow fundamentals, and even if you know IP inside and out, it's probably not a bad idea to read it. One of the main tenets of "flows" is that each flow only describes unidirectional traffic, so for a single TCP session, you actually have two flows: from client to server and from server to client. Once you adjust to this way of thinking of flows, you'll understand how the flow analysis tools view the traffic, which is pretty important to have under your belt before heading off to the rest of the book. In addition, an overview of NetFlow history and versions provides some background of the how, why, and what netflow analysis can provide.

Chapter 2 talks about collectors and sensors, which are the two main components for gathering your NetFlow data. Hardware and software methods are discussed, and installation of the flow-tools software is shown. Once you have your collectors and sensors configured, you're ready for Chapter 3, which walks you through viewing flows. In any kind of enterprise environment you'll quickly see that, as with any other (firehose) data source, you'll want to be able to filter out the cruft to see what's really important. Chapter 4 describes filtering "primitives" and constructing useful methods for parsing your flow data. Chapter 5 deals with reporting and follow-up analysis. The default report is a great start, but this chapter shows how to modify it to suit your needs, as well as providing suggestions on some useful reports that you can use as a starting point for different types of network traffic: IP Address Reports, Traffic Size Reports, and BGP Reports, to name a few.

At this point, assuming you've been installing and configuring per the book examples, you have at a minimum a functional netflow analysis platform. Chapter 6 takes it a step further by introducing the Cflow.pm Perl module. Evidently, due to earlier flow analysis tool development, there is some confusion between Cflow.pm and cflowd (an obsolete flow analysis tool; this chapter will make sure that you get everything installed correctly. Once you do, you can start leveraging this Perl module to write your own analysis code. Not only that, but "the most user-friendly flow-reporting" tool, FlowScan, depends on it. This chapter will show you how to set up FlowScan and use it to start generating Web-based, graphical reports. Since FlowScan is somewhat non-customizable, you'll probably want to install the FlowView suite, and Chapter 7 will guide you all the way. You can think of FlowScan as being good for "normal" people and FlowView for network administrators. 'Nuff said.

Chapter 8 will gently help you set up gnuplot so that you can quickly create "impressive graphs of your network data" without wanting to kill yourself. What more could you ask for in a chapter?

The final chapter addresses NetFlow v9, which the current version of flow-tools does not accept.

Instead, the emerging successor to flow-tools, flowd, is introduced and explained. The author even says that he had considered using flowd as the core of the book, but it's still a little too new. As we've seen in the previous chapters, there is a lot of information you can get out of your network using flow-tools, but if you're interested in coding up your own reports and you need to handle NetFlow v9 traffic, then flowd is for you.

This is one of the best books I've read in a long time. It does precisely what it sets out to do: teach you about flow analysis and how to use the tools to set it up yourself. Lucas has a great sense of humor that isn't overwhelming and keeps the tone of the book moving along. If you are looking to learn about flow analysis or implement something in your network, you simply cannot go wrong with this book. If you don't care about network flow analysis, well, you should, so go get this book anyway. You won't be disappointed.

HIGH PERFORMANCE JAVASCRIPT

Nicholas C. Zakas

O'Reilly Media. 209 pp.
ISBN 9780596802790

REVIEWED BY BRANDON CHING

In my experience, most developers have only cursory JavaScript skills. Sure, most can write form validation and do some neat tricks with Jquery, but there seem to be very few developers who have actually mastered JavaScript. *High Performance JavaScript* is one of the books that can help you do so.

Don't let its relatively thin size fool you; Nicholas Zakas (and five other contributing authors) packed loads of insightful and valuable information into the pages of this very readable book. It covers not only the obvious performance topics of loading and execution but also DOM scripting, AJAX performance, algorithm and flow control, and even a chapter on regular-expression performance (which is really quite good). The chapter on DOM scripting really stood out. Extensive in its coverage and explanation, it is loaded with valuable nuggets of information.

While these types of topics can be dense even for experienced developers, the author's writing style is clear, consistent, and to the point. I often found myself at the end of a paragraph asking a question, only to have it clearly answered in the next paragraph. There are hundreds of code samples, performance metrics, and charts that demonstrate the more general point the author is trying to make at a clear, practical level, with concise explanations of why he is recommending something. This makes the book much more approachable to a wider audience of developers.

High Performance JavaScript is definitely intended for experienced Web developers. However, given the approachability of the writing and the continuity of the information, I think that even junior and mid-level developers would find immense value in having this book within easy reach. While it may be a bit of overkill, I can also feel confident in recommending this book to Web UI designers who might be responsible for the JavaScript-driven UI aspects of their sites. After just the first chapter, I was amazed at all the things I had learned, and I think many other developers would feel the same after reading this book. Very highly recommended.

This 25th anniversary edition of Steven Levy's classic book traces the exploits of the computer revolution's original hackers -- those brilliant and eccentric nerds from the late 1950s through the early '80s who took risks, bent the rules, and pushed the world in a radical new direction. With updated material from noteworthy hackers such as Bill Gates, Mark Zuckerberg, Richard Stallman, and Steve Wozniak, Hackers is a fascinating story that begins in early computer research labs and leads to the first home computers. Levy profiles the imaginative brainiacs who found clever and u