book reviews

ELIZABETH ZWICKY,
WITH JEFF BERG, BRANDON CHING,
AND RIK FARROW

DATA CRUNCHING: SOLVE EVERYDAY PROBLEMS USING JAVA, PYTHON, AND MORE

Greg Wilson

Pragmatic Bookshelf, 2005. 188 pp.
ISBN 0-9745140-7-1

Last issue, I reviewed (and loved) *Automating System Administration with Perl*. Let's suppose you want something similar, but you don't like Perl. This is the book for you. What it means by "Java, Python, and more" is basically "Java, Python, a little bit of Ruby, the occasional mention of C." It only brings up Perl to sneer at it.

Data Crunching doesn't cover the same range as *Automating System Administration with Perl*. Obviously, some of this is because it's not about system administration, but the result is that *Data Crunching* focuses on data sitting still, while *Automating System Administration with Perl* also covers notification and data gathering techniques. Thus, *Data Crunching* and *Automating System Administration with Perl* both give you basics of XML, XSLT, and SQL, but *Automating System Administration with Perl* adds DNS, SNMP, SMTP, and log files, while *Data Crunching* provides more coverage of regular expressions and of programming technique.

Data Crunching gives you the basics you need to know in order to beat a data problem to death with programming with an appropriate level of elegance, and when you should choose to use a different solution (either just doing it by hand or doing a proper job of programming).

STAND BACK AND DELIVER

Pollyanna Pixton, Niel Nickolaisen, Todd Little, and Kent McDonald

Addison Wesley, 2009. 152 pp.
ISBN 978-0-321-57288-2

I love management techniques that involve low intervention, so I was predisposed to like this book, and I liked it just as much as I hoped. But as it turns out, my absolute favorite part is not about hands-off, high participation management; it's about convincing people NOT to do things. Lots and lots of books will tell you about ways of convincing people to do things, or helping them choose between options, but very few people will tell you what to do if, as far as you can tell, your entire project is obsessed with choosing between and rank ordering things they just shouldn't be doing at all. It can be agonizingly difficult to convince people that the problem is that they are doing a perfectly competent, well-managed, on-time job—of the Wrong Thing. *Stand Back and Deliver* will not fix all of these situations, but it provides tools for opening up the conversation to the idea of doing some things just well enough, and the idea that you might want to totally rethink the goals of your project.

This is a book for people who already have a grasp of the basics of managing projects and need some new techniques; it's not a full toolbox, it's a set of interesting, specialized tools you may not have seen before. But they're very useful tools that you won't find elsewhere. If you need some new ways of looking at large projects, check it out.

GRACE HOPPER AND THE INVENTION OF THE INFORMATION AGE

Kurt W. Beyer

MIT Press, 2009. 380 pp.
ISBN 978-0-262-01310-9

This is a fascinating biography of Grace Hopper. It's an academic biography, which means that it hews pretty carefully to knowable facts, without dramatizing (in fact, sometimes it seems to be burying them a bit), but the facts are riveting ones if you're at all interested in the history of computing. Sometimes you marvel at how much things have changed, and sometimes you marvel at how little they've changed. The startup experience and the professional group formation experience are essentially unchanged during the time when the computers themselves and the process of programming them have both changed beyond all recognition.

This is mostly about the history of computing, with some discussion of Hopper's position in the history of women in technology. That doesn't appear to be the

author's main interest, but you can't really avoid it when you're talking about the most famous woman in the development of modern computing. There's a very delicate balancing act to be done here. Grace Hopper is sometimes thought of as evidence that women have always been accepted in computing, and sometimes held up as an exceptional person who succeeded despite being female and is not representative. As always, there's truth on both sides. The book does a nice job of trying to provide context for Hopper's achievements.

One warning; the chapter labeled 1 is effectively a prologue. It's more academic and less interesting than the rest of the book, and includes some meta-argumentation about academic biographies that will only be absorbing to those interested in academic biography as a genre rather than in Grace Hopper and the history of computing. I wish I'd started with Chapter 2, which is where the actual story starts.

ALGORITHMS FOR VISUAL DESIGN USING THE PROCESSING LANGUAGE

Kostas Terzidis

Wiley, 2009. 337 pp.
ISBN 978-0-470-37548-8

Theoretically, this book is aimed primarily at people with design and architecture backgrounds who are learning programming from the ground up. This is one of those rare cases where the title does a better job of selecting an audience than the introduction does; don't try this book on somebody who can't already program in some language, because its introduction to programming takes about 30 pages, followed by exercises such as "Write the shortest possible procedural code that can generate the following number pattern using only standard arithmetic operations." You need to be reasonably literate in programming, mathematics, and visual design: not an expert, but comfortable looking at equations, or code samples, or pictures of patterns.

If you're in that audience (or, I suppose, if you are strongly motivated and like a challenge), this book does a nice job of introducing Processing, plus some native Java constructs you might need, and the sorts of algorithms you are likely to want to use to do visual stuff. It covers some stuff that *Processing* (reviewed earlier this year) does not, mostly by going at breakneck speed and including less art.

I found this book useful and enjoyed it, but it has a number of drawbacks. To start with, don't read the introductions to the chapters. Introducing chapters is always tricky, and as an author I sympathize with the struggle to work your way into the meat of the chapter. But if you're looking for advice on writing out files, the following is not going to entice you into the chapter:

> Memory is the mental faculty of retaining and recalling past experience; it is the act of remembering or recollecting.

You'll also probably want another book on Processing, because the author is fond of doing things the hard way while failing to fully explain the easy way. It's nice to know the math behind Bézier curves, but actually implementing them from scratch is not the right answer in a language which has curve primitives built in. Once you understand what's going on, use the primitives. Similarly, I'm glad to know how to do native Java file writes from Processing, but it would have been friendlier and more useful to start by using the handy built-ins. It would also be friendlier to clearly distinguish between Processing constructs and Java constructs, at least so that students can easily figure out what manual they ought to be trying to look things up in, and they will have some idea of what Java constructs will work.

MALWARE FORENSICS: INVESTIGATING AND ANALYZING MALICIOUS CODE

Cameron Malin, Eoghan Casey, and James Aquilina

Syngress, 2008. 592 pp.
ISBN 978-1597492683

REVIEWED BY JEFF BERG

A book running over five hundred pages might seem a bit cumbersome, but *Malware Forensics: Investigating and Analyzing Malicious Code* provides a jump start into malicious code incident response and the issues that encompass it. Even those involved with malicious code research on a daily basis will find this book to be a good tune-up on the methodology, tools, and techniques implemented for effective incident response. Covering everything from retrieving potential artifacts from physical memory and volatile information to minimizing changes to a system, backing up hard drives for further analysis, and the legal ramifications of investigations, *Malware Forensics* discusses all the major topics and then some.

One of the underlying concepts that echoes throughout the book is evidence dynamics—a term that has been coined to reflect influences that will change, relocate, obscure, or damage evidence. This is most easily applied to physical criminal cases in which a

first responder may "disturb the scene" where a victim is found, but translates to malicious code incident response in the way a host or victim is influenced as an investigation begins. Evidence dynamics is an important concept regardless of the specific profession a researcher embarks upon, because it is impossible to know when data collected will have to be used as evidence. The authors do a great job of addressing this topic throughout the entire book.

One excellent feature of this book is the practical case scenarios that are found in each chapter and, in some cases, carried on throughout the book. The scenarios enable the reader to translate the concepts into the real world. The authors also provide helpful analysis tips, such as processing suspect files and conducting analyses in isolated environments, a tip that seems to be common sense to most veteran researchers but might not be apparent to the newbie. Another cool aspect of this book is that the authors constantly suggest tools that could be helpful, along with pointers about how to obtain the tools. However, a good part of the book's value lies not so much in what is included but in what is excluded. The authors defer to other sources on general knowledge-base topics such as network traffic analysis aiding in an investigation, as well as background knowledge of ext2 and ext3 file systems. This is as it should be, but the reader should be aware of the expected prerequisite knowledge and that he or she may need to crack a second book to get up to speed for the provided discussion.

Malware Forensics is broken into ten chapters, some of which cover the same general concepts but concentrate specifically on Windows or Linux. For example, Chapters 1 & 2 cover "Volatile Data Collection & Examination on a Live . . ." Windows or Linux system, respectively. The two chapters explain the methodology, tools, and techniques involved with data collection during the initial portion of a response (e.g., documenting network connections, logged-on users, open ports, process information, etc.), all while keeping evidence dynamics in mind, documenting steps and forensic preservation for deeper analysis. Chapters 4 & 5 discuss the "post-mortem" phase of the analysis—gleaning information from the preserved copies of data and introducing a methodology that is repeatable and used in either a random infection or a test infection, where a system is purposely infected to better understand the piece of code. Chapters 7 & 8 cover "File Identification & Profiling . . ." The authors discuss a methodology, fairly similar across Windows and Linux, for pulling malicious executables or files related to them off the hard drive and "profiling" them—i.e., getting the hash, determining file type, scanning with a malicious code scanner, and running strings for references or keywords to assist in classifying the file and, ultimately, the code. Chapters 9 & 10, "Analysis of a Suspect Program . . . ," tie a lot of the concepts covered throughout the book together into a chapter-long case study of a potential malicious program.

The only two chapters that do not focus on Windows or Linux specifically are 3 and 6. Chapter 3 discusses the methodology, techniques, and different tools necessary for acquiring and analyzing memory for malicious code evidence on both Windows and Linux systems. Chapter 6 is dedicated to the legal aspects of the field, covering issues such as who has jurisdictional authority to conduct an investigation, ensuring that an investigation is performed such that the evidence is admissible, and providing basic guidance for instances where evidence may exist outside jurisdictional authority and how to obtain it. Though the authors stress the need to consult with legal teams, they provide a good base of issues to be aware of.

Malware Forensics will dive as deep as analyzing a suspect program or process in a disassembler to understand what a program is doing, but it doesn't require a reverse engineer's background to gain a good amount of value from reading it. However, the reader should be prepared to spend time with the tools and techniques discussed. If you are anything like me, you'll benefit much more by doing so. And if you are truly worried about the length, practicing with the tools after each chapter will break up any monotony of reading.

SEXY WEB DESIGN: CREATING INTERFACES THAT WORK

Elliot Jay Stocks

SitePoint, 2009. 172 pp.
ISBN 978-0980455236

REVIEWED BY BRANDON CHING

I do not have a single creative bone in my body! OK, that's probably an exaggeration, but when it comes to designing an innovative, attractive, and usable Web site, I definitely could use a helping hand. As a Web developer, I am generally responsible for the data in our sites rather than the look and feel; that's the UI team's domain!

However, not all developers have access to professional UI resources and, depending on the situation, many of us often wear a number of different hats.

As such, *Sexy Web Design* is a book that seems made for folks like me who know a little something about the basics of Web design, but are nowhere near creative experts.

The primary focus of *Sexy Web Design* is on the process and fundamentals of professional design rather than specific HTML and CSS techniques. As such, there is a heavy focus on planning and organizing a Web site and little coverage of actual coding. In fact, there is little to no CSS in the book. You may now be asking yourself, what use is a design book without any code samples? In short, quite a bit.

The author takes you through all phases of planning a site design, from research and customer requirements to site mapping, wire framing, usability, composition, navigation, and his keen emphasis on aesthetics. The text is an ideal companion for designers and developers who are new to interacting directly with customers, as Stocks covers the professional give-and-take process of designer/customer interactions. While coverage of each topic is rather short, it is to the point and seems to cover the requisite ground for a basic introduction.

There are many good visual examples of the principles Stocks is introducing, and he does point the reader to more detailed texts and Web sites for further reading. The text also provides a number of general considerations that people not trained in design could be unfamiliar with, including composition, mood, contrast, volume and depth, typography, and textures.

Overall, the book was very readable and approachable. I would recommend the book to experienced Web developers who don't dabble much in site design or customer interactions (but know the technical details of HTML and CSS) and for those looking for a general introduction to the planning and creative considerations of designing a Web site from the ground up.

I would say that the biggest disappointment of the book was its brevity. Stocks covers a lot of ground in only 172 pages, and with a lot of that real estate used up with visual aids and examples, it doesn't leave much for explanation. However, in combination with other more detailed and technical resources, it does creatively address the often neglected planning, usability, and customer interaction aspects of a bottom-up site design, and for that, I feel that it is a book worthy of consideration.

Web 2.0 Architectures: What Entrepreneurs and Information Architects Need to Know

James Governor, Dion Hinchcliffe, and Duane Nickull

O'Reilly, 2009. 271 pp.
ISBN 9780596514433

REVIEWED BY BRANDON CHING

There are many Web 2.0 books out there these days, but most seem to address the Web 2.0 concept in a business or abstract sense. *Web 2.0 Architectures* by Governor et al. is written for "in the trenches" IT professionals who are charged with the creation and execution of next-generation Web platforms.

Web 2.0 Architectures takes an in-depth look at system architectures from a practical and theoretical perspective relative to Web 2.0 strategies. The book can be seen as being organized into three general areas: introduction and models, architecture patterns, and building for the future.

The first six chapters take you deep into the technical details of Web 2.0 architecture, network, and design principles. The real heavyweight of this section is Chapter 3, where the authors compare and contrast the Web 1.0 and Web 2.0 design patterns of some major companies and concepts in tech, including Akamai/BitTorrent, MP3.com/Napster, and CMSes/wikis. The comparisons are detailed and informative, outlining the major shifts in thinking and design between the approaches. There are plenty of diagrams, flow charts, and network models to emphasize the message. After this section, the reader will have a solid foundation for the more theoretical concepts to come.

Chapter 7, comprising about half the book's page count, is where the authors deliver the main content: detailed coverage of Web 2.0 architecture patterns. The authors discuss twelve patterns in detail, including an explanation of the business problem, solution, behavior, implementation, consequences, and more. Again, there is no shortage of charts and diagrams to help impress upon the reader the lessons being taught, and superb lessons they are indeed.

Finally, Chapter 8 attempts to tie up the loose ends, describing where Web 2.0 is going and identifying offshoots of next-generation platforms such as Advertising 2.0 and Government 2.0. This chapter was not very strong, and I didn't find much useful information that was relevant to the rest of the book.

Overall, *Web 2.0 Architectures* is well worth a read for its comparative analysis of Web 2.0 models and its detailed explanation of Web 2.0 patterns. This book

is ideal for senior-level information architects and technically minded entrepreneurs who are looking to build a Web 2.0 system architecture from the ground up. The language and topics covered are definitely on the technical side, so this book may be out reach for some, but the writing is clear and professional. If you (or your business) are looking to adopt a more advanced and focused architecture but need a bit of conceptual and theoretical guidance, this book is definitely where you should start.

iWork '09: The Missing Manual

Josh Clark

O'Reilly Media/Pogue Press, 2009. 896 pp.
ISBN 978-0-596-15758-6; eBook 978-0-596-80341-4

REVIEWED BY RIK FARROW

I wanted to try an eBook, and I also needed to know more about Keynote, the presentation software that Al Gore made famous. I had used Keynote to make several presentations, and can say that it is *mostly* intuitive and easy to use. But there were certainly some aspects that just had me baffled.

iWork covers Pages, Keynote, and Numbers, the Word, PowerPoint and Excel-like programs from Apple. I focused on the Keynote section, as that is what I was using, and quickly found that I liked reading the book. Clark starts the Keynote section with advice about creating presentations that I actually found useful. Instead of skipping over what would be filler in another book, I read most of this chapter of good advice.

Getting into the nitty-gritty, I was able to quickly find answers to things that you might think would be easy to do, like printing a two-up (two slides per page) version of your slides. I had puzzled over the Print dialogue several times, but Clark explains that using a menu and selecting the Layout tab makes it easy to do this. I had never used animation, and again Clark made this easy to understand.

iWork does have features that overlap between applications, such as fonts and drawing tools, so the Keynote section does not stand alone. But that is not a failure of the book, just me wanting to get answers in a hurry.

I am still not comfortable with eBooks: laptops do not work that well for reading in a comfy chair or in bed, the screen format is not portrait but landscape, the laptop will start to burn your skin, among other interesting failings. Owning a Kindle, which implies that I can no longer lend a book to a friend without lending my entire library, is not the model for me. I did try borrowing the large LCD display known as a TV these days, but again the aspect is landscape, not portrait. I found that the eBook was good as a reference and not so good for sitting down and reading—nowhere near as convenient to use as a plain old paper book.

I can recommend *iWork '09*, the book, as a useful companion to the Apple apps, and well worth the price for the time saved searching for answers to simple questions, as well as for Clark's useful advice.

Data Crunching: Solve Everyday Problems Using Java, Python, and more. Greg Wilson. This book gave me basic knowledge about regular expressions, XML and XSLT. And mainly, moved me from just SELECT in SQL to SELECT .. JOIN etc. Regular Expressions Cookbook. Jan Goyvaerts, Steven Levithan. Thaks to this book I can do much more with regular expressions now still used as cookbook sometimes. Tools. First computer Some homemade 286 based. Favorite editor Visual Studio (Code). Company data powered by. Sergej Brjuchanov. Prague, Czech Republic https://github.com/SergeS. Data Crunching is a method used in information science that makes the preparation of automated processing of large amounts of data and information (Big Data) possible.Â Data crunching is more about correct processing, so that a system can do something with the records and the data format. Data crunching is therefore an upstream process of data analysis. This process, as the data analysis itself, can be iterative when the output of the crunching process includes new data or errors.Â â†‘ Data Crunching: Solve Everyday Problems Using Java, Python, and More media.pragprog.com. Accessed on 03/20/2015. Web Links[edit].