Book Reviews

ELIZABETH ZWICKY, MARK LAMOURINE, AND RIK FARROW

Beyond Software Architecture: Creating and Sustaining Winning Solutions
Luke Hohmann
Addison Wesley, 2003, 302 pp.
ISBN 978-0-201-77594-5
Reviewed by Elizabeth Zwicky

When you go to build real systems, you will discover that the architecture is driven by a giant pile of considerations that are not theoretically part of software architecture. ("Why don't you log it in this log file?" I asked the other day. "Ah. Well. That log file is written by this other part of the software, and this part of the software only writes this log file. They have different lead programmers, see. And there's no way to log across the boundary without rewriting both components.") This is a book about the space where software architecture meets the need to actually make money and satisfy the people who build it.

Most of the book is about the relationship between what you sell to customers (the marketing architecture, or "marketecture," as the author calls it) and what you build (the technical architecture, or "tarchitecture," which sounds sticky and unpleasant to me, but makes sense as an abbreviation). It suggests ways in which they should interact, while being clear on the differences between them.

If you are making the leap to designing whole systems, particularly commercial software (which is the focus here, although the author does consider internal systems in passing), this is a nice balanced look at the problem, which does not present either the customer or the marketing department as an enemy. It encourages sensible, honest, and human behavior without pandering to the whims of programmers. For instance, it suggests that you should not try to make water flow uphill or ignore the personal needs of your team to feel like they have important, relevant tasks, but it also suggests avoiding getting sucked into new technologies because they'll look good on resumes or marketing collateral.

The section on security is OK but a bit perfunctory and not fully integrated. It does not have the same feel of hard-learned lessons that most of the rest book does, but it does at least hit the high points (you can't add security at the end of a project; do not write your own encryption; be sure you know what you're trying to protect against and protect sufficiently for that rather than trying to protect everything everywhere all the time against everybody).

Bad Data Handbook
Q. Ethan McCallum
O'Reilly Media, 2013, 256 pp.
ISBN 978-1-449-32188-8
Reviewed by Elizabeth Zwicky

Continuing in this month's theme of ruthless realism, here's an entire book on analyzing the data you can actually get, which is never quite the data you wanted. It's a collection of essays, not a cohesive whole, but it's full of interesting and useful insights. They range from generalizations (why and how you should cross-check data) to very specific techniques (how to process text of unknown origin and dubious character set in Python).

Anybody who is new to data analysis can use this book. Most new analysts have a wholly unfounded faith in the data, which leads them to produce charming flights of fancy that fall apart when you look at them at all closely (hey, according to this the sum of the parts is greater than the whole!). When they lose this faith, they are not sure what to substitute for it, and poke gingerly at the data like a small child presented with a new food. This book suggests concrete steps to take to verify and sanitize data, as well as pointing out the importance of understanding exactly where your data comes from and how.

As always, some parts are weaker than others. As somebody who works with human-generated data a lot, I found the section on "liars" less useful. It's good to understand that the people who use systems often have complicated meta-goals and almost always behave in ways you don't expect (the example the author uses is a great one, and I don't want to give it away); the users understand how the system works, mostly, but they don't really get all the implications of the difference between automated and human systems. Their attempts to optimize are damaging to the system as a whole, but it's not exactly because they intend to lie, and the author fails to draw out the implications for human-based systems. There are always people who are trying to cheat the system, but there are also always people who are simply at odds with your design goals, intentionally or unintentionally. New analysts, in general, assume that you can divide all the data points into "good" and "bad," and anything you weren't expecting is somebody being Wrong on the Internet. In fact there are at least four relevant categories—"good," "bad," "bug in the processing system," and "not what we were hoping the user would do."

Data Insights: New Ways to Visualize and Make Sense of Data

Hunter Whitney
Morgan Kaufman, 2013, 296 pp.
ISBN 978-0-12-387793-2
Reviewed by Elizabeth Zwicky

This is a very pretty book, with nice examples of visualizations and with generalizations and advice that are both engaging and accurate. It is suitable for smart people who aren't clear on where to aim for when going from a pile of numbers to something meaningful, or a manager who wants or needs to understand what comes beyond Excel's defaults for pie charts. It is not, however, at all specific on how to get to where you are aiming at. If you want step by step advice, or actual techniques at all, you'll need to look elsewhere, so absolute beginners and experts alike are probably going to be frustrated.

I didn't disagree with the content; I think the author's approach is right, but I felt it kept getting close to being useful, coherent, and organized, and then getting distracted by another pretty visualization and wandering off again. Plus, QR codes as ways to provide URLs with examples sound like a great idea, but they're eye-catching without being illuminating to the reader. And, in my case, the first two I tried don't work on the device I had at hand; one is a Flash animation and the other loads but for some unclear reason doesn't do a lot. This is an object lesson in something about visualization, but it's not clear that it's what the author had in mind. (And I worry about the level of impermanency implied in using goo.gl-shortened links to a magazine Web site; six months from now, are these QR codes going to lead anywhere at all?)

Finally, there's an illustration labeled "The Esperanto of visualization." I think that's meant to be a compliment, but without any further explanation of the illustration, I can't be totally sure, since the illustration conveys no data at all to me. It could be meant as a scathing condemnation of both Esperanto and the visualization. This was an unusually clear example, but I often found myself unsure of exactly what message I was supposed to take away and what I was supposed to do about it. Ultimately, I ended up treating it like a fashion magazine; look at the pretty pictures, maybe some of the captions about what they are.

Generation Blend: Managing Across the Technology Age Gap

Rob Salkowitz
Wiley, 2008, 243 pp.
ISBN 978-0-470-19396-9
Reviewed by Elizabeth Zwicky

Sadly, I got through the entire book without being convinced that in fact there is a significant difference between grouping people by generation and grouping them by astrological sign. Supposedly, Millenials (the youngest generation at work) like to mix life and work by having lives at work, and Boomers (the oldest) and Generation X (mine, which is apparently why I'm so cynical) don't—Boomers take work home, and Generation X works to live. I fail to see how this goes with the popular statistics about lost work time for the Super Bowl and Thanksgiving shopping, and while anecdote is not data, I promise you I get mail (to my personal address) from the personal address of my Boomer colleague during work hours. I also read it. Apparently it's not just the kids these days.

If you believe in firm distinctions between the generations, and dislike one of them, this is a useful introduction to how all of them are useful in the workplace. Otherwise, the most useful part of the book is a chapter on how to teach computers to older people who don't have much experience with them.

Digital Capture After Dark

Amanda Quintenz-Fiedler and Philipp Scholz Ritterman
Rocky Nook, 2013, 190 pp.
ISBN 978-1-9333952-66-6
Reviewed by Elizabeth Zwicky

Digital cameras these days will do all kinds of fascinating things with almost no human help. So, of course, people want to do the things they're bad at. Photographing in the dark is one of those things—your camera will be great, aside from the fact that it won't focus or correctly calculate exposure for you, so pretty much, you're on your own. Once it gets dark your easy-going camera that shows you something like you see and can let you review it on the spot becomes a lying tyrannical magic box which must be kept still and shielded from stray light but then will show you things you can't see.

The authors go over the gear you need (a tripod, a remote shutter release, extra batteries, a flashlight, extra batteries for the flashlight) and then get into the techniques. Most of these are photographic techniques, but some of them are more practical. On the photographic side, you can't trust the camera to meter and you can't trust the preview or the viewfinder to show you what you're really getting, so you have to learn to figure out what you're probably getting and how to raise your odds of getting what you want. On the practical side, it's cold and dark at night. Digital cameras run on electricity. Batteries hate the cold. How do you maximize your ability to take photographs? How do you minimize the risk to your camera from coming back inside to the damp warmth?

Most popular kinds of night photography are discussed, including special lighting techniques like light painting and use of light bursts to combine motion and still in one picture. There isn't any discussion of lightning (oddly, because there's a lightning photo), but lightning is basically just a big fast light you don't control.

The book succeeded in making me feel that photographing in the dark was exciting and within my grasp, and its advice is consistent with the little night photography experience I've had. I'm looking forward to more experimentation.

Hacker's Delight, 2nd Ed.

Henry S. Warren, Jr.
Addison-Wesley, 2013, 494 pp.
ISBN 978-0-321-84268-8
Reviewed by Mark Lamourine

http://www.hackersdelight.org/

Hacker's Delight is a rare find: a clear, well-written book about a fundamental element of programming. I would put it on the same level as Stevens' TCP/IP Illustrated. Most people will never need to code binary logic and arithmetic operations at the level of individual RISC instructions, but knowing what's going on down there can only help.

Warren opens Hacker's Delight by describing the sandbox in which he will play: a basically unrestricted number of general purpose 32-bit registers and a "basic" and an "extended" RISC instruction set. All of the operations are simple integer arithmetic bit shifts and binary comparisons on registers or constants. Warren also limits branches and memory load and store instructions because they are expensive. The goal is to see what can be done within these constraints. The remainder of the book demonstrates that the answer is "a lot."

The first seven chapters cover a wide range of simple but sometimes obscure bit and byte operations. Where a reader might ask, "But why would I want to do that?" Warren provides a brief answer or a reference to more detail.

The chapters are broken into subsections of about one to three pages which describe and then explain a particular operation. The algorithms are presented in computer algebra (presented in Chapter 1) or in ISO C99. At least one algorithm is offered in Python, showing that the applications are not limited to machine-level code. Where a formal derivation or proof of the algorithm is needed it is presented in more traditional logical or mathematical form.

As Warren progresses, he moves from simple operations to more complex tasks. He shows how to implement such things as integer multiplication and long division, CRCs, and even how to implement floating-point arithmetic.

Warren briefly mentions issues that someone might find when trying to implement some operations in other languages, such as Java or on variations in real-world processors which may differ slightly from his abstract instruction set. As a high-level programmer, I would have liked to see some discussion of the applicability of these algorithms in modern scripting languages. While the logic remains correct regardless, I'm curious if the implementation of the scripting languages will carry the compact logical operations down into the resulting machine code.

There is also an associated Web site: http://www.hackersdelight.org. This refreshingly unadorned set links to copies of the code samples, errata, and additional resources.

Hacker's Delight reminded me that there is a case to be made for clever logical minimalism in specific cases, and this can have its own beauty and clarity. That said, I expect that it will be of limited direct use to the vast majority of the computing community. But utility isn't everything. Hacker's Delight will be a pleasure to anyone who started working with computers out of curiosity about how stuff works and an appreciation of the aesthetics of fundamental logic.

Dart in Action

Chris Buckett
Manning Publications Co., 2013, 398 pp.
ISBN 978-1-617290-86-2
Reviewed by Mark Lamourine

There have been a number of attempts to improve upon or replace JavaScript as the client-side programming language for the Web. Google's Dart programming language is one of these.

Dart is a young language. Chris Buckett introduces Dart and the entire development ecosystem: an IDE, testing frameworks, client- and server-side coding and deployment. The first section is a fairly traditional whirlwind setup and "Hello World" treatment.

In the second section Buckett does an excellent job of detailing the syntax of Dart. Especially important are the sections on two language features which appear to be unique to Dart. The first is the optional type system. In an attempt to address the concerns of both the fans of strongly typed languages (for compile time type checking and diagnostics) and those who prefer the flexibility of weakly typed languages, Dart allows the coder to choose whether or not to provide data type information.

The second feature, isolates, is a mechanism to provide concurrency in a single-threaded environment. Since Dart must be translated to JavaScript, and JavaScript is single-threaded, then Dart must be too. Isolates allow concurrency by creating functions with distinct execution contexts, and limiting interaction between them. Buckett provides a series of examples for using isolates to manage concurrent queries on a Web service written in Dart.

In the final two main sections of the book, Buckett addresses the requirements and capabilities of Dart on the Web browser and the Web server. He gives special attention to client-server communications and data storage. At the time the book was written

Dart did not yet have a browser GUI library, but the Dart Web site indicates that the current release does include one.

Buckett concludes with a pair of appendices which together provide a language reference to match the tutorial of the second main section of the book.

I like the general style of the Manning "In Action" series. At the end of each tutorial section there is a highlighted paragraph listing the significant points from the preceding section. Buckett's inclusion of the reference appendices at the end of the book means that it will continue to have use after the tutorials are finished.

Dart projects can be deployed either by translating to JavaScript (much like CoffeeScript is deployed) or by using the DartVM. Currently, the only Web browser to offer a DartVM is a specially built Chrome browser provided with the Dart SDK. Indications are 18 months after Dart's announcement no other Web browser producer has plans to include a DartVM, and even Google has no plans to include it in the main-line Chromium release.

If you're working on a project which already uses Dart or if you are interested in alternatives to writing Web applications in JavaScript, "Dart in Action" is a good place to start.

Distributed Network Data

Alasdair Allan and Kipp Bradford
O'Reilly Media, Inc., 2013, 155 pp.
ISBN 978-1-449-36026-9
Reviewed by Mark Lamourine

The subtitle of Distributed Network Data is "From Hardware to Data to Visualization." Allan and Bradford really do take the reader of this slim book from a hardware parts list to graphing the collected data.

To fully appreciate this book the reader really should commit to acquiring the parts and following along. This is a book for the adventurous beginning Maker. It is a little helpful (but not necessary) to have some familiarity with a soldering iron, a text editor, C, and a search engine. It's really important not to be intimidated by three- and four-letter acronyms. The authors explain the ones that matter and refer the reader to other sources when necessary.

Allan and Bradford have structured the book so that each chapter presents a nicely self-contained task. Each chapter builds on the one before. They begin with an explanation of the goal and any needed theory and then dive into a guided tutorial, ending with a demonstration of some new capability.

In the first few chapters the reader will get comfortable with Arduino, breadboarding, and some fairly easy circuits. The second section brings in the "network" from the title with XBee network devices. The final sections introduce three different pieces of software for visualizing both the plan for the project and the resulting data.

I haven't understood until now how a Mac OS user feels when presented with examples and screenshots from Windows or Linux. The examples and code in Distributed Network Data are all created on Mac OS. Two of the three visualization platforms, Processing and Fritzing, are open source, free to download, and available for all three platforms. LabView is a commercial product, but there is a free version tailored for use with Arduino.

Allan and Bradford close the book with a chapter giving references to the original sites for each of the software packages and for additional reading.

At the end of the process the reader will have a working wireless sensor system collecting data and the ability to plot the data. More importantly, the reader will have the confidence to try different configurations and pointers to additional resources, including a community of Arduino Makers, to tap for more ideas and explorations.

Absolute OpenBSD, 2nd Edition

Mike Lucas
No Starch Press, 2013, 491 pp.
ISBN 978-1-59327-476-4
Reviewed by Rik Farrow

Mike Lucas produces books that are clear and easy to read, and this book is no exception. He provides information that is specific to installing and maintaining the most recent version of OpenBSD at the time (5.3), and with a level of detail that is refreshing. Today, most of us just "ask the Web" when we want a quick answer. Each chapter in this book is more like a tutorial that goes beyond just telling you what to do, but why you should be doing it, or not doing it.

You won't find information about configuring Apache 2.2 in this book—in fact, Mike recommends that you use nginx instead, but has nothing to say about configuring nginx. This is a book about the OpenBSD system, not applications. Mike does have a thorough chapter about how to find applications, use the package system, or use ports if you cannot use the package system, but that's it. And as it should be.

You will find detailed information about configuring OpenBSD, from the options in login classes, to the purposes of the various files found in the /etc directory. There are two chapters about TCP/IP, one on theory, the other on the practical aspects, including IPv6. Mike also wrote two excellent chapters on PF, the OpenBSD firewall, which in itself is a good reason to use OpenBSD. If you want to try an operating system focused on security, and also want detailed instructions, this is the book for you.