Book Reviews

MARK LAMOURINE AND RIK FARROW

Bad Choices: How Algorithms Can Help You Think Smarter and Live Happier

Ali Almossawi
Penguin Random House LLC, 2017, 146 pages
ISBN 978-0-7352-2212-0

Reviewed by Mark Lamourine

In the Summer 2020 issue I reviewed Ali Almossawi's first book, in its online and print editions, *Bad Arguments*. That was a book about logic and logical fallacies, a subject that is always timely. It led me to his second book, *Bad Choices*, about algorithms and how we can use them in daily life. This serves two purposes. The superficial goal is to show how algorithmic thinking can make ordinary tasks more efficient and effective. But the real goal, more subtle and subversive, is to show that the concepts of programming and computation aren't as abstract and alien as they seem when presented in the classroom. We use algorithms every day, and, with just a little attention paid, we can see how pretty much all computation matches problem-solving activities from everyday life.

Almossawi starts each chapter with a little anecdote about a character whose name is a bad pun and who has some task to accomplish. The tasks range from sorting socks to finding all the items on a grocery list while visiting the minimum number of different aisles. He presents each vignette with a statement of the objective and two or three methods of trying to accomplish the task, and then the fun begins.

The veneer of a picture book or a children's story slips away pretty quickly. In the first chapter, he introduces the idea of algorithmic complexity based on the relationship between the size of the job and the time needed to complete the task. He doesn't go deep into the math but presents the growth curves and the concepts of polynomial and logarithmic growth. These are sprinkled through the remaining chapters for comparison. Later chapters cover the ideas behind arrays, associative arrays, hash functions, quick sort, and binary trees, among others.

The first seven chapters of *Bad Choices* are online at https://bookofbadchoices.com/. The first page is even read by Almossawi's son, I think. The presentation is very faithful to the book, down to the graphical page-turn transitions. If you're trying to decide if you want to buy a hard copy for someone, it's an excellent facsimile.

The bad choices of the title are the obvious ways that we do small tasks. For the typical size of daily chores, things like bubble sort or exhaustive search of a clothes rack for the right shirt size pose no real problem. Almossawi uses them to present alternatives and introduce in an informal way the most common algorithms used in computer science. It won't make anyone a programmer, and it won't teach a software developer anything they don't already know. But it might help demystify the idea of algorithms for someone who wants to get more comfortable with computation, and it might even help them sort socks or craft a clever tweet.

The Skeptics' Guide to the Universe: How to Know What's Really Real in a World Increasingly Full of Fake

Steven Novella, with Bob Novella, Cara Santa Maria, Jay Novella, and Evan Bernstein
Grand Central Publishing, 2019, 528 pages
ISBN 978-1-5387-6052-9

Reviewed by Mark Lamourine

I think the authors of *The Skeptics' Guide* must see the irony in the fact that their book is a self-help guide, though I don't expect you'll find it in that section of a book store. In a time of industrial-scale misinformation, the skills needed to evaluate what you see and hear and read must be actively taught and learned. Even more important may be the knowledge of how we as humans can be deceived or deliberately deceive ourselves. You have to want to learn and be willing to let go of what you want to believe if you want to grow.

The Skeptics' Guide to the Universe is the collected wisdom of the hosts of a podcast of the same name. They have been working together since 2005. They created and run the Northeast Conference on Science and Skepticism (NECSS). I have been listening to the podcast for several years, and I admit I am a fan.

The core of the book is the idea of *scientific skepticism*, which is not to be confused or dismissed as philosophical skepticism. The latter is the idea that nothing can be known or trusted. Scientific skepticism is an approach to understanding in which one accepts that learning is possible but that it is a matter of refinement. It is the idea that while one can never achieve certainty, it is possible to approach it in a way that allows one to act in the face of incomplete understanding. The ability to give up a cherished idea in the face of evidence is the most important tenet.

Each of the chapters in the book is fairly short, from two to ten pages at most. They are meant as an introduction to a topic and an invitation to learn more. Each chapter is backed by references that the reader can use to go into a topic in more depth.

Curiously, there is almost nothing in the book telling the reader what *to* believe. In the main section of the book, Novella talks about the ways in which we as humans can mistake the world. First he discusses the realm of illusion and the failure of human intuition. Recent research into the malleability of memory and recall, the mind's ability to see patterns where they don't exist, the meanings attributed to dream and near dream experiences all can inform our response to seeing something strange or apparently inexplicable. It can lead us to question our certainty in our memories and experiences, at least enough to withhold judgment without confirmation.

Novella proceeds to talk about how to think about what we perceive and how we interpret it. This process is known as *metacognition*. It's easy to dismiss the idea of metacognition as "navel gazing," but that's actually the point. Those who would dismiss it would cite what they call common sense. This section is a list of the ways in which "common sense" isn't.

This isn't a way in which "people are stupid." The first chapter in this section is on the Dunning-Kruger effect, but if there's any takeaway it's that this applies to everyone, depending on the topic. A well-educated, intelligent person needs to always be on guard because it is easy for anyone to assume that, since they are expert in one field, they are qualified to evaluate and speak about another. The overriding message of this section is that one needs to be constantly aware of the possibility of being mistaken, especially when you are confident that you are not.

Additional chapters in this section cover motivated reasoning, formal logical fallacies, and some of the more common informal fallacies such as appeal to nature, misinterpreting statistics, or believing coincidence is more than coincidental. In each case, the purpose is to help the reader understand the human tendency toward misperception and how to recognize and correct for it.

In the remainder of this section, Novella covers recognizing the characteristics of pseudo-science and understanding a set of lessons from history. Both groupings contain examples of deliberate hoaxes, honest mistakes, and systemic failures.

The final two sections finally begin to talk about what a reader can do to address misinformation in the media and in life. Today we have access to far more information than we can possibly digest individually. We have to learn to evaluate the sources and our own responses to determine how to use what we get to form a view of the world. The point is never to arrive at certainty, but to create a level of understanding and confidence that allows us to act reasonably. This is always a provisional understanding, and it is assumed that we will continue to learn and refine this view, sometimes even rejecting previously held ideas if new data changes our understanding. Those familiar with Bayesian statistics will be familiar with this idea. When our worldview changes, we can change our behavior to match.

Taken as a whole, *The Skeptics' Guide* is a collection of things to note and to keep in mind when taking in the news of the world and trying to make sense of it for everyday life. It holds a number of cautionary tales, but the message is always one of optimism. It *is* possible to learn and to act reasonably in our society, but it takes some care and self-discipline. It's also possible to recognize *don't care* conditions, where you can let your guard down and relax. Not every topic needs skeptical scrutiny.

This book is not going to convert anyone from a closely held ideology. The nature of human identity means that we don't change who we are easily or quickly. For someone who is confused by the current torrent of input and wants some ideas about how to try to process it without becoming cynical or nihilistic, *The Skeptics' Guide to the Universe* is a great start.

Re-Engineering Legacy Software

Chris Birchall
Manning Publications Co., 2016, 214 pages
ISBN 978-1-61729-250-7

Reviewed by Mark Lamourine

In my experience, software developers are prone to producing crap. It's not all our fault. It's a factor of the limitations on money, time, and sometimes attention and patience. Tasks like writing code and running tests repeatedly aren't the most exciting aspects of coding. Regardless of the reason, there's a lot of code out there that meets Birchall's criteria for "legacy software."

The title of the book belies the scope of Birchall's ambitions. His focus is on refactoring the entire process of software development. I've read and reviewed a number of books that go into depth on the facets of modern software engineering processes. There are books about revision control, automated testing and build processes, and agile planning methods. *The Practice of System and Network Administration* [1] and *Refactoring* [2] are classics, but the first is a general-purpose tome defining the ideals of the industry, and the second is a tightly focused exposition of a neglected facet of the software development process. Each has a place, and both can be daunting to someone looking for an overview that touches on all the needed topics but leaves the details and depth for another time. *Re-Engineering Legacy Software* tries to fill that gap.

In the first half of the book, Birchall does concentrate on the code base and he does start with basic refactoring, but he doesn't stop there. In the next two chapters he expands to reworking software architecture and then again to the considerations of a complete rewrite. He doesn't advocate for either method as a means to reach a more maintainable design. His approach is to look at the factors that would influence the decision to implement either an incremental or bulk replacement of an existing code base. He leaves it to readers to evaluate their own situations.

It is in the second half that he moves on to the design of, not the software, but the software development environment and processes. These are aspects that I think are often either neglected or that slavishly adhere to some ideology that may not take into account the specific needs of the team, the customer, or the project. Birchall discusses the common modern techniques of automated testing, continuous development, and delivery, but with a view to adding them to a project where they are not currently in use. He clearly is an advocate of modern practices, and he brings an agile view to implementing them in existing environments. The focus is again on incremental improvement, not on wholesale replacement, though he does discuss times where that might be the best course.

The final section covers project management and software development culture. Here he echoes in brief the messages of *The DevOps Handbook* [3] and *The Phoenix Project* [4]. These are the classic works on modern software development process and culture. Birchall glosses over the types of cultural and personal pressures that can lead to wasting time on precious features, or alternatively, the mistaken avoidance of writing throwaway code to allow for incremental improvement.

Re-Engineering Legacy Software won't replace any of the old favorites on my book shelf. On the other hand, I would recommend it to someone entering software project management cold or approaching a legacy project for the first time. Birchall makes a subject that can be the focus of ideological wars and pet software tools accessible without a lot of the hype and heat that have been present over the last decade or so. The flip side of "fail fast" is "one bite at a time," and Birchall's book is bite-sized.

References

[1] T. Limoncelli, C. J. Hogan, and S. R. Chalup, *The Practice of System and Network Administration,* 3rd edition (Addison-Wesley Professional, 2016).

[2] M. Fowler, *Refactoring,* 2nd edition (Addison-Wesley Professional, 2018).

[3] G. Kim, J. Humble, P. Dubois, J. Willis, and J. Allspaw, *The DevOps Handbook* (IT Revolution Press, 2016).

[4] K. Behr, G. Spafford, and G. Kim, *The Phoenix Project,* 5th Anniversary edition (IT Revolution Press, 2018).

IT Architect Series: Stories from the Field

Matthew Wood, John Yani Arrasjid, and Mark Gabryjelski
IT Architect Resource, LLC, 2020, 270 pages
ISBN: 978-0-9990929-1-0

Reviewed by Rik Farrow

In the preface of this ebook, John Arrasjid writes that the stories are supposed to be both informative and entertaining. I can agree with John's statement, as I learned things from reading about the misfortunes of others, but also found myself often entertained at the same time.

Stories from the Field begins with a long preface, including a classification scheme for categorizing the stories, using terms like Analysis, Communication, Politics, Database, and Risk. As I just read straight through, the categories really didn't make any difference to me, but at least hinted at what I'd soon be reading.

The stories themselves are written by 35 contributors, presumably all IT architects. I wasn't familiar with "IT architect" as a job description, but learned as I read that this person works with a team to design large scale distributed systems for some business purpose. Most of the team works for a company that does installations, often called a partner but what in the past might have been called a VAR. The team includes people who handle the business side of the project, but also technologists like programmers and network engineers working beside the IT architect.

The stories roughly follow a pattern where the project is described, and this is where I learned about the protocols for designing these projects as well as the systems and software used in current IT departments. There are lots of references to VMware products, and the acronyms used for different types of offerings, like virtual desktop infrastructure (VDI), cloud computing hypervisor (vSphere), and the VMware Enterprise hypervisor (ESXi) took some getting used to. Note that this is not a technical book and is not specific to VMware, but VMware products are often involved in the stories.

Each story ends with lessons learned, and after a while I became familiar with the patterns of failure. Most common were failures in communication that led to misunderstandings, but almost as common were mission creep, although sometimes the *creep* was more like a *leap*, as customers would suddenly decide on installing the just-released version of a major release or have purchased different, and usually cheaper, equipment. There are things that an SRE would find more familiar, such as failure to determine all dependencies until a server fails, and turns out to be the keystone in an entire system.

Overall, I enjoyed reading *Stories*, as the stories themselves are short, informative, and generally fun to read. And, honestly, I felt glad that it was somebody else who had to live through the misadventures.

ENIGMA

A USENIX CONFERENCE

SECURITY AND PRIVACY IDEAS THAT MATTER

Enigma centers on a single track of engaging talks covering a wide range of topics in security and privacy. Our goal is to clearly explain emerging threats and defenses in the growing intersection of society and technology, and to foster an intelligent and informed conversation within the community and the world. We view diversity as a key enabler for this goal and actively work to ensure that the Enigma community encourages and welcomes participation from all employment sectors, racial and ethnic backgrounds, nationalities, and genders.

Enigma is committed to fostering an open, collaborative, and respectful environment. Enigma and USENIX are also dedicated to open science and open conversations, and all talk media is available to the public after the conference.

PROGRAM CO-CHAIRS

Lea Kissner
Apple

Daniela Oliveira
University of Florida

The full program and registration will be available in November.

enigma.usenix.org

FEB 1–3, 2021

OAKLAND, CA, USA

usenix
THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

usenix

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Writing for *;login:*

We are looking for people with personal experience and expertise who want to share their knowledge by writing. USENIX supports many conferences and workshops, and articles about topics related to any of these subject areas (system administration, programming, SRE, file systems, storage, networking, distributed systems, operating systems, and security) are welcome. We will also publish opinion articles that are relevant to the computer sciences research community, as well as the system adminstrator and SRE communities.

Writing is not easy for most of us. Having your writing rejected, for any reason, is no fun at all. The way to get your articles published in *;login:*, with the least effort on your part and on the part of the staff of *;login:*, is to submit a proposal to login@usenix.org.

PROPOSALS

In the world of publishing, writing a proposal is nothing new. If you plan on writing a book, you need to write one chapter, a proposed table of contents, and the proposal itself and send the package to a book publisher. Writing the entire book first is asking for rejection, unless you are a well-known, popular writer.

;login: proposals are not like paper submission abstracts. We are not asking you to write a draft of the article as the proposal, but instead to describe the article you wish to write. There are some elements that you will want to include in any proposal:

- What's the topic of the article?
- What type of article is it (case study, tutorial, editorial, article based on published paper, etc.)?
- Who is the intended audience (syadmins, programmers, security wonks, network admins, etc.)?
- Why does this article need to be read?
- What, if any, non-text elements (illustrations, code, diagrams, etc.) will be included?
- What is the approximate length of the article?

Start out by answering each of those six questions. In answering the question about length, the limit for articles is about 3,000 words, and we avoid publishing articles longer than six pages. We suggest that you try to keep your article between two and five pages, as this matches the attention span of many people.

The answer to the question about why the article needs to be read is the place to wax enthusiastic. We do not want marketing, but your most eloquent explanation of why this article is important to the readership of *;login:*, which is also the membership of USENIX.

UNACCEPTABLE ARTICLES

;login: will not publish certain articles. These include but are not limited to:

- Previously published articles. A piece that has appeared on your own Web server but has not been posted to USENET or slashdot is not considered to have been published.
- Marketing pieces of any type. We don't accept articles about products. "Marketing" does not include being enthusiastic about a new tool or software that you can download for free, and you are encouraged to write case studies of hardware or software that you helped install and configure, as long as you are not affiliated with or paid by the company you are writing about.
- Personal attacks

FORMAT

The initial reading of your article will be done by people using UNIX systems. Later phases involve Macs, but please send us text/plain formatted documents for the proposal. Send proposals to login@usenix.org.

The final version can be text/plain, text/html, text/markdown, LaTeX, or Microsoft Word/Libre Office. Illustrations should be PDF or EPS if possible. Raster formats (TIFF, PNG, or JPG) are also acceptable, and should be a minimum of 1,200 pixels wide.

DEADLINES

For our publishing deadlines, including the time you can expect to be asked to read proofs of your article, see the online schedule at www.usenix.org/publications/login/publication_schedule.

COPYRIGHT

You own the copyright to your work and grant USENIX first publication rights. USENIX owns the copyright on the collection that is each issue of *;login:*. You have control over who may reprint your text; financial negotiations are a private matter between you and any reprinter.