Book Reviews

MARK LAMOURINE AND RIK FARROW

UNIX and Linux System Administration Handbook, 5th Ed.
Evi Nemeth, Garth Snyder, Trent R. Hein, Ben Whaley, and Dan Mackin
Pearson Education, 2018, 1180 pages
ISBN 978-0134277554

Reviewed by Mark Lamourine

There are few books that I would recommend to every working sysadmin at every level of ability and at any point in their career. This is one. I'm going to refer to it merely as [Nemeth5] to avoid writing the whole title repeatedly. This is also a deliberate tribute to Evi Nemeth, one of the original authors and a pioneer of learning and teaching system administration as an art and profession.

I've owned all five editions of [Nemeth] as soon as they were released, and I've learned or re-learned something from each of them. [Nemeth4] was the last one that Evi worked on. Evi was crew on the *Niña*, a 50-foot sailing yacht that went missing in the Tasman Sea in 2013. In [Nemeth5], the remaining co-authors have maintained the range and quality of the previous editions.

From the first edition, [Nemeth] has been a *handbook*. Although it's big and has fine paper pages, it is meant to be kept close and thumbed through often. A handbook doesn't have the narrative of a tutorial or the depth of a topical reference manual.

When the first edition appeared, the World Wide Web didn't exist. Today you can find everything in [Nemeth5] through a search engine. The paper book has one often overlooked advantage: compactness. By this I mean that all of the of the searching and sorting and question refinement has been done for you, the reader. All you need to do is flip to the table of contents or the index to find what you need.

Each edition has been based on a set of four or five currently popular vendors or distributions. For the 5th edition, the authors chose FreeBSD and three flavors of Linux: Debian, Ubuntu, and CentOS. This selection is broader than it seems because these generalized distributions are commonly used as a base for more targeted flavors. Users of these derivative distributions will still find a lot of value here. The authors make note of another 10 distributions, including their strengths and their relationships to the selected core set.

Nemeth et al. have never been shy about expressing an opinion, and you'll find a lot of it here still, though usually couched in wry humor. For example, a paragraph comparing boot time init systems is entitled "inits judged and assigned their proper punishments." The authors address all of the options that a reader is likely to encounter. Their goal is to assist the reader, but they don't feel the need to appear impartial in their evaluation of the tools they're describing.

I can't possibly enumerate all of the sections and topics the authors cram into this two-inch-thick tome. Instead, I'm just going to note a few of the things that caught my attention as new or interesting as I leafed through.

The first item that I came across was a scheduling-tool alternative to cron. I've worked with systemd since it was introduced in Fedora, but I've never seen systemd timers before. It may or may not replace cron, but it certainly presents an alternative, offers much finer control, and allows explicit sequencing capabilities with other systemd controlled events.

The section on scripting I would recommend to beginning system administrators even over most books on the topic. The authors give very good advice on style and approach. They describe and provide examples for all of the most critical language features and a number that are more obscure, useful, and commonly overlooked. The chapter concludes with brief introductions to both Python and Ruby. While it may be true that one can be a good system administrator without programming, I would claim that anyone would find the job easier with some skill in scripting.

When the 4th edition was released in 2010, cloud computing was in its infancy. Modern containers were introduced with Docker in 2013. [Nemeth5] includes a complete chapter on commercial cloud service concepts, providers, and a few examples. It includes chapters on virtualization and containers. None of these are deep, but they are broad and touch on all the important ideas. The writing is clear, without any of the hyperbole or misplaced enthusiasm that is common in dedicated books. Like most chapters, these end with a list of external references and suggested reading.

I don't think I can express just how encyclopedic this book is. It has one of the best technical introductions I've seen on DNS and DNSSEC; a fairly complete examination of SMTP interactions; reasonable default configurations of Sendmail, Exim, and Postfix; and good comparisons of recent CM tools such as Ansible and Salt. Puppet and Chef get mentioned, but they're not the cool kids now, apparently. See: Opinions. The book closes with a chapter on datacenter management that includes the merits of various DC layouts, with example floor plans. I've left out nearly half of the topics in the book and haven't touched on the obscure-but-valuable little knowledge tidbits sprinkled in every section.

Throughout, [Nemeth5] is readable and accessible. It's perfect for either thumbing through or finding just the start you need on any topic. It is an ongoing tribute to Evi and her lifetime of work.

Accelerate: Building and Scaling High Performing Technology Organizations
Nicole Forsgren, PhD, Jez Humble, and Gene Kim
IT Revolution Press, 2018, 256 pages
ISBN 978-1-942788-33-1

Reviewed by Mark Lamourine

Finally, someone applies science to the Agile/DevOps practice.

For years the Agile/DevOps movement has had only hype, surmise, and anecdote to support a counterintuitive idea: that giving individuals more agency in their work with less managerial gatekeeping (along with the tools to detect and respond rapidly to problems) leads to faster, better, more reliable software and services. That's not to say that there was no evidence. The anecdotes are many, and, when done well, numerous informal case studies have made people confident that there's something to the ideas. Agile practice is derived from the Toyota Production System (TPS), first formally described in 1988 [1]. The advantages of TPS are backed by strong commercial and academic research, but TPS is designed for manufacturing production, and it is not a given that it would translate trivially to software development. Some additional confirmation is needed.

In *Accelerate*, Forsgren et al. have applied modern sociological methods, first to define and then to measure the effectiveness of Agile practices in software development and service delivery. You won't learn how to run a scrum stand-up or use a Kanban board (unless you follow the references in the bibliography). What you will find is the first real demonstration that Agile practices, writ large, are effective, and specifically which aspects have the most demonstrable benefit. They also show proper recognition that there is more work to be done to design and implement good practices and to keep learning how to measure and evaluate them.

Forsgren et al. provide the three elements you expect in a peer reviewed paper, but in a narrative form that non-academic readers will fine comfortable. Don't let the form put you off. If you can read a good technical reference, you can follow their exposition and arguments. If you have read any RFCs, this will be a breeze.

The reason for the somewhat different presentation from most other books in this arena is that *Accelerate* is based on the same data set that the authors used for two peer reviewed papers [2, 3] in 2016, and continue to use for more recent papers. You can find references to the ongoing work on their website [4].

They begin by defining the question under study: What is meant by "Agile practice"? What is meant by "effectiveness" and how do you measure it? The first third of the book defines what her team will try to measure and what they will not.

In the middle section, they lay out their findings so far. They start by describing their data collection methods and briefly justify the use of sociological models before proceeding to explain the data and what it means.

The final section is the most technical. Here the authors explain the methodology and models that they chose to use when gathering the data. They justify the selection and design of the models, questions and data analysis in terms that will be familiar to anyone in the social sciences. They go further, to explain briefly the more technical terminology and offer references for those who want to learn more about the details.

There's nothing earth-shattering in the results. This is primarily because that's not what they were trying to do. The metrics they chose for software delivery quality are deliberately constrained: software delivery rate, delivery lead time, software failure rate, and time to fix. Then they surveyed a broad range of people and companies using varying degrees of Agile methods in their software development. They hoped to test for correlations between those who use Agile methods and those who achieve high marks in their quality metrics.

In general, they find that "higher quality" as defined by their metrics are associated with groups and companies that conform to Agile tenets. A couple of things that raised an eyebrow: the best performers commit directly to a master SCM branch and have no or very short term development branches. The change in lead times in the highest performers were measured in hours. I'm not so much skeptical of these findings as I am wondering whether the strict definitions required for a good metric are out of line with my colloquial understandings of these terms.

What most worries me is the possibility that the metric definitions are in some way forming a logical circle with the Agile methods supposedly under test. The descriptions of the metrics align fairly strongly with my understanding of the purpose and goals behind Agile philosophy and methods design. Is it possible that what is being measured is the effectiveness of the methods used to achieve the stated behavioral goals, without ever evaluating whether those behaviors actually improve the software being delivered? I'm inclined to view Agile methods as indeed effective and beneficial. I'm not sure how to show something stronger than "They do what they claim, and aim, to do."

In every case, Forsgren et al. are careful to qualify any claims. There are muddy and unanswered questions. There are anomalies in the data that need explanation and resolution. Mostly, we need more data and a longer history to work with. Over time the metrics will, I hope, be broadened and refined. A larger longitudinal data set will make trends more evident and the conclusions more sound. This is what distinguishes research from advocacy.

This isn't a book for the beginning coder. It's not even for most people who are already faithfully attempting to use Agile methods (or not). *Accelerate* is for the doubters who need evidence to show that Agile methods aren't merely a buzzword fad, and for developers and managers who might need some talking points when trying to pitch or improve the software development practices where they work.

There's still a lot of work to be done to find the best ways to manage software development and delivery, but we have a foundation on which to build.

References
[1] T. Ohno, *Toyota Production System: Beyond Large-Scale Production,* 1st Edition (Productivity Inc., 1988).

[2] N. Forsgren, A. Durcikova, P. F. Clay, and X. Wang, "The Integrated User Satisfaction Model: Assessing Information Quality and System Quality as Second-Order Constructs in System Administration," *Communications of the Association for Information Systems* 38 (2016), pp. 803–839.

[3] N. Forsgren and J. Humble, "DevOps: Profiles in ITSM Performance and Contributing Factors," in *Proceedings of Western Decision Sciences Institute (WSDI) 2016*.

[4] https://devops-research.com/research.html.

The Computer Book: From the Abacus to Artificial Intelligence, 250 Milestones in the History of Computer Science (Sterling Milestones)
Simson L. Garfinkel and Rachel H. Grunspan
Sterling, 2018, 528 pages
ISBN 978-1454926214

Reviewed by Rik Farrow

I got to see a proof of this book, coming out November 2018, and have to say I was pleasantly surprised. Not that I didn't expect Simson, both a friend and someone who has written many books, to succeed at this task. But because in reading the book, I kept saying to myself, "Damn, that's how these events fit together."

The authors chose 250 milestones, ranging from a clay tablet abacus to artificial general intelligence—not that we are anywhere near that. The format of the book consists of a page of text on the left with a color photo or illustration on the right. The photos always add some real context to the page of history, besides often being beautiful (the Babbage replica) or just plain interesting. The writing is concise, as it must be to fit on a single page, but always held my attention and was easy to read. Each page ends with cross-references to other pages.

One of the benefits of reading this book is that I learned about the origin of many of the technologies and the terms we use. While some were obscure, like the meaning of "RS" in RS-232, others were real eye-openers.

While this book won't help you with programming or sysadmin, it is a lot of fun to read. And I guarantee that you will find lots of surprises, whether you read it cover-to-cover or just pick it up and flip through the pages at random. And finally, for a book full of photos, the price is very reasonable.