Book Reviews

MARK LAMOURINE AND RIK FARROW

Groovy in Action, 2nd ed.

Dierk König and Paul King
Manning Publications, 2015, 912 pages
ISBN 978-1-93518-244-3

Reviewed by Mark Lamourine

I have been learning Groovy as part of some recent work using Jenkins pipelines for a build-and-test system. The Jenkins pipeline plugin implements a Domain Specific Language (DSL) for job control that extends (and, to some degree, limits) the Groovy language. I started off just reading examples and using search engines to answer my questions. When it became clear to me that this was going to be a regular part of my work, I decided I needed to do some proper reading.

Groovy is a scripting language and execution environment based on Java that first appeared in 2003. The first edition of *Groovy in Action* was released three years later. The second edition covers Groovy 2.4. The current version of Groovy, as of this writing, is 2.5, and a version 3 is in development. The Jenkins pipeline plugin appears to be based on Groovy 2.4.

To be clear, *Groovy in Action* is strictly concerned with the Groovy language and does not treat Jenkins or the pipeline plugin at all. Understanding Groovy is required but not sufficient to write Jenkins pipeline jobs. There's also a lot more to Groovy than Jenkins.

Groovy in Action feels smooth. The progression of topics is clear and logical. König and King know their audience and write well to them. There are no tutorial digressions into theory or foundations. They do make note of things that might be unexpected or unfamiliar, like the concept of optional typing. Groovy is tightly coupled to Java, and the authors assume a level of comfort with Java and with modern programming practice and terminology.

The first section covers the Groovy language proper. I like the grouping of data types into Simple and Collective. Placing a complete chapter on closures before the chapters on control structures and objects breaks the ordering I'd expect, but it makes sense when seen in this context. Closures are no longer an exotic idea, and they are a big part of idiomatic Groovy.

The language section closes with a couple of chapters on dynamic programming and static typing in Groovy. Groovy is an "optionally typed" language: you can specify the types of variables and functions, or you can let the compiler infer them. It is not a dynamically typed language. That is, once you or the compiler have determined the type of a variable, it cannot be changed. Specifying the type allows the compiler to flag mismatches, but you'll get a runtime error if you try to change the type of an inferred object.

In the second section the authors start making some real work possible. The chapters cover how to work with databases, structured data, and web services. Groovy includes something I haven't seen before, called "builders," which are used to create hierarchical data structures without a lot of the boilerplate. Builders can create in-memory trees or structured data like XML or JSON or even HTML. I'm going to be giving this a closer look.

König and King finish up with a section of practicum. This includes chapters on unit-testing, and interacting with the underlying ecosystem. I group the appendices in this as well. These cover installation of Groovy and some great cheat sheets for language constructs. It also includes a small section on the operational steps of the Groovy compiler and a list of compiler annotations, which I find useful and unusual. I like knowing one layer beneath where I'm working.

Groovy in Action, 2nd ed. introduced me to a new way to interact with the Java environment that felt smooth, seamless, and comfortable. Groovy might prove useful for prototyping Java. I don't know if it is a reasonable replacement for Java, but it's certainly easier to use and debug. If you need to work with Jenkins pipelines, I'd consider it invaluable and am using it daily.

Improving Agile Retrospectives: Helping Teams Become More Efficient

Marc Loeffler
Addison-Wesley Professional, 2018, 272 pages
ISBN 978-0-13-467834-4

Reviewed by Mark Lamourine

For most people the Agile retrospective is the least appealing part of the development process. The idea of re-hashing all the things that have gone wrong can lead people to find fault and lay blame. Few people are totally comfortable with having others put their mistakes under a microscope. In *Improving Agile Retrospectives*, Loeffler shows that there is another way (several others, in fact) to draw out experience and make use of it without resorting to finger pointing and personal judgment.

You might not think there's that much to running a meeting that shouldn't last more than an hour, but unlike most meetings, the retrospective has the potential to go very wrong. Loeffler devotes the first half of the book to describing and understanding the purpose of the retrospective and the role of the facilitator. The

retrospective will often require some preparation and thought to create the right atmosphere for candid discussion. Most won't need the full treatment Loeffler offers, but these chapters offer a good toolbox.

The purpose of a retrospective is to understand what happened in the recent past; how reality matched and differed from the plan. It should also be a time to recognize and celebrate what went well. There is always something that either didn't go as planned, or was harder than expected. Chapter 1 sketches a skeleton for the meeting, creating an arc from opening to closing. Loeffler provides an agenda of five phases and a timeline for each. An hour isn't very long, and it's important to keep the process moving.

The next three chapters frame the job of the facilitator. Chapter 2 covers preparation and planning. Chapter 3 walks through the first retrospective, touching on the process of guiding and moderating the meeting and the goals for each phase. Chapter 4, the longest in the book at 35 pages, details the role of the facilitator in the process: how to prompt and guide the discussion while keeping the focus on the team and on constructive participation. A good facilitator is central to the process, but should never become the focus of the meeting. There are goals beyond the production of recommendations and action items. At the end of the ideal meeting, all of the participants should feel that they have been heard and that the results represent the consensus of the team. Items that can't be resolved can be tabled but should not be dismissed. Nothing ever ends up ideal, but the result can still be satisfying.

The real surprise to me was the second half of the book. It turns out that there are quite a number of ways to frame the process of reflection. Retrospectives can have different scopes and goals, and these may require different techniques. A single-team end-of-sprint meeting doesn't compare with a project-management-level post-release review. Some of the metaphors offered here may not work for all audiences. The props needed for the "kitchen retrospective " might not set the right tone for a director-level quarterly meeting but could be just the thing to open a team session with a light, fun tone. It's easy when running a retro every two to three weeks to fall into mental ruts, and a change-up of the format can freshen people's engagement and interest.

Whether you've been asked to moderate a retro meeting or are a participant, *Improving Agile Retrospectives* will give you a better handle on how to participate and get the most from the process. Hopefully, it will encourage both doubters and advocates for the benefits of the retrospective process to participate with an open mind.

The Manga Guide to Cryptography
Masaaki Mitani, Shinichi Sato, Idero Minoki, and Vert Corp.
Omsha Ltd. and No Starch Press, 2018, 234 pages
ISBN: 978-1-59327-7

Reviewed by Mark Lamourine

You might be forgiven for assuming that *TMGTC* is aimed at middle schoolers or teen students. They might be drawn in by the cover and the artwork in the first few pages, but it's not long before the real math and logic come out to play.

There are characters and a story line that interlace with the exposition, but they really just frame the real lessons. If the framing entices a reader to start and continue, then they've done their job, but they don't add much to understanding the content.

The contents are actually a good treatment of the basics of modern encryption. The four sections cover foundational ideas, symmetric and public-key ciphers, and close with practical applications. These last should be familiar to most Internet users today: user identification and authentication, content encryption and validation, and e-commerce.

Another surprise is the breadth and depth of the coverage. The first chapter introduces the mandatory Caesar cipher and transposition ciphers with a page apiece, but doesn't wait there at all. The rest of the section goes into fairly deep exposition of the construction (and deconstruction) of polyalphabetic ciphers like Enigma. The authors close with the introduction of Vernam ciphers and one-time pads. They go on to show how these actually demonstrate the ultimate vulnerabilities of any polyalphabetic cipher system, and set up the next chapter, symmetric key systems.

The symmetric chapter is probably the best one: I've never seen a good explanation of the internals of a block cipher until now. The authors lay out the data flow of DES on a single page and explain triple DES. They show how these are now vulnerable to brute force attacks and have been replaced by AES. While they don't detail the data flow of AES, they've left the reader with an understanding of how block ciphers, as a class, work.

The public-key section is just as complete. Mitani et al. explain in a matter of 75 pages the key exchange problem, trap-door functions, modulo arithmetic, Euler functions, and prime factoring. They finish the chapter by demonstrating a walkthrough of RSA: key generation, encryption, and decryption.

The closing section touches on the ways most people today use encryption even if they don't know it.

This book is one of a series of manga books from Omsha and No Starch. The series includes treatments of physics, relativity, statistics, and calculus. The originals are in Japanese, and No Starch is in the process of bringing them all to English-speaking audiences. Cryptography is the latest release.

The only problem with the translation is that in the opening stories, the authors use several jokes and puns that have to be called out and explained because they only make sense in Japanese. While these stood out, they only happen at the beginning of the book and none of them are critical to understanding.

While *The Manga Guide to Cryptography* is presented as a pop-media book, it would be a disappointment to someone looking for a casual pop treatment of cryptography. There's no code here, either. You won't learn how to use crypto libraries in your app. This book is best suited to the self-learner who wants to be conversant with the underlying ideas and perhaps understand some of what's going on behind-the-scenes every day when they use their web browser.

Millions, Billions, Zillions: Defending Yourself in a World of Too Many Numbers

Brian Kernighan
Princeton University Press, 2018, 174 pages
ISBN 978-0-691-18277-3

Reviewed by Rik Farrow

Kernighan begins his preface by quoting three geniuses, W. E. B. DuBois, John Nash, and Nate Silver, sharing their words about numbers. Of the three, Silver comes closest to the mark when I think about this book: "On average, people should be more skeptical when they see numbers." Kernighan wants us not just to be skeptical, but to use techniques, like estimation, to determine whether the numbers we see in print, from our friends, or in the news are actually correct.

Fortunately, Kernighan doesn't expect the readers of this book to be geniuses. Instead, he starts out by writing that all you need are grade-school arithmetic skills to follow along and learn from this work. I found that was true when I started reading portions of the book to my wife, who had been math-averse ever since some teacher criticized her for not being quick with numbers.

Throughout *Numbers*, my nickname for this book, Kernighan focuses on the use of estimation and rounding. In every example, he starts by using these techniques, allowing him to quickly come up with a decision about some numbers taken from the press: instead of using 330 million for the population of the US, try 300 million as a starting point, for example.

There are 13 chapters in *Numbers*, each focusing on a different aspect of accidental, or occasionally malicious, innumeracy. He covers mistakes involving names for large numbers or units, mis-mixing units, dimensionality, milestones, specious precision, statistics, bias, and arithmetic.

Kernighan begins by explaining how to make large numbers easier to understand. Words like million, billion, and trillion have no intuitive meaning to most people, myself included, and thus we tend to treat them as synonyms for "big," "really big," and "really really big." Kernighan suggests scaling down numbers: for example, if the US budget were $3.9 trillion, each person's share of that would be about $13,000.

He also suggests more use of scientific notation. This is one of the parts of the book I read to my wife as she painted, and she told me she finally understood scientific notation. Kernighan's explanations, and use of examples, really do make this book easy to understand.

Throughout *Numbers*, Kernighan provides shortcuts for quick calculations. Some were ones I already used, while others were new to me, like Little's Law, an easy method for figuring approximately how long it will take for an amount to double with different compound interest rates. Or the relationship of some powers of two to powers of ten that comes in very handy.

The only weak point in the book is the chapter on statistics. *Numbers* isn't a statistics book, and Kernighan does explain with a very clear example the common failing of the usage of the mean when the median would be more appropriate. He also makes many references to Darrell Hull's *How to Lie with Statistics* (1954), a book that he considers worth reading.

I liked reading *Numbers*. Kernighan's style could be said to be didactic, but it's never boring. I recommend *Numbers* to anyone who encounters potentially dubious numbers during their day—that is, everyone.