;login:
THE MAGAZINE OF USENIX & SAGE
June 2002 volume 27 • number 3

inside:
BOOK REVIEWS

USENIX & SAGE
The Advanced Computing Systems Association &
The System Administrators Guild

the bookworm

by Peter H. Salus

Peter H. Salus is a member of the ACM, the Early English Text Society, and the Trollope Society, and is a life member of the American Oriental Society. He is Chief Knowledge Officer at Matrix.net. He owns neither a dog nor a cat.

peter@matrix.net

BOOKS REVIEWED IN THIS COLUMN

IP ROUTING
RAVI MALHOTRA
Sebastopol, CA: O'Reilly &Associates, 2002. Pp. 219. ISBN 0-596-00275-0.

THE PROCMAIL COMPANION
MARTIN MCCARTHY
Edinburgh, Scotland: Addison-Wesley, 2002. Pp. 235. ISBN 0-201-73790-6.

J2ME IN A NUTSHELL
KIM TOPLEY
Sebastopol, CA:O'Reilly & Associates, 2002. Pp. 450. ISBN 0-596-00253-X.

THE J2EE TUTORIAL
STEPHANIE BODOFF, ET AL.
Boston, MA: Addison-Wesley, 2002. Pp. 491 + CD-ROM. ISBN 0-201-79168-4.

JAVA IN A NUTSHELL, 4TH ED.
DAVID FLANAGAN
Sebastopol, CA: O'Reilly & Associates, 2002. Pp. 969. ISBN 0-596-00283-1.

EMBEDDED LINUX
CRAIG HOLLABAUGH
Boston, MA: Addison-Wesley, 2002. Pp. 419. ISBN 0-672-32226-9.

LINUX ADMINISTRATION HANDBOOK
EVI NEMETH, ET AL.
Upper Saddle River, NJ: Prentice Hall, 2002. Pp. 889. ISBN 0-13-008466-2.

VMWARE
BRIAN WARD
San Francisco, CA: No Starch Press, 2002. Pp. 249. ISBN 1-886411-72-7.

INTRODUCTION TO PROGRAMMING IN EMACS LISP, 2ND ED
ROBERT J. CHASSELL
Boston, MA: FSF, 2001. Pp. 292. ISBN 1882114-43-4

I've got a heap of books I want to talk about this month. And a poster, too, for those of you with space on a wall between "User Friendly" and "Dilbert" clippings.

Getting Around

There are currently over 175 million host machines on the Internet (up from 213 in August 1981). Getting mail, VoIP, streaming video, http[s] packets, ftp and scp packets, AUDIO, and other stuff from box to box is not trivial. I have relied on folks like Huitema and Perlman to elucidate the complexities of routing in the past. Malhotra's small (barely 200 pages) book will now live next to my desk. Though ostensibly about Cisco routing, Malhotra's expositions of RIP, IGRP, EIGRP, RIP-2, OSPF, and BGP-4 are good enough for those working with Juniper, Nortel, etc., hardware. Definitely a must have!

Incidentally, if you're interested at all in Internet history, Peacock (*www.peacockmaps.com*) has a really fine "First Maps of the Internet" poster for $29.95. Just right for covering the hole in the plaster you tossed that CPU through . . .

/dev/null

I get a lot of email. Much of it offers me $$$ or XXX or and opportunity to buy Viagra or enlarge my breast size. For several years, procmail has been my friend: I'm now down to about 100 messages a day; some of them actually meaningful. Marty McCarthy has written a fine book on setting up and using procmail. (I have to confess that I read and commented on an early manuscript and wrote the Foreword to McCarthy's book. I have no financial interest in it.)

Cups of Java

J2ME is the Java 2 Micro-Edition: it is designed for resource-limited devices like cell phones or pagers. It is really cleverly designed. Topley has done the excellent job that I've come to expect from the Nutshell handbooks and references.

J2EE is the Java 2 Enterprise Edition. This tutorial is full of real examples and thoroughly useful pointers on just how J2EE can be used in your enterprise. The CD contains three (!) J2EE tutorials, J2SE and J2EE software development kits, a sample Java BluePrints, and the Forte for Java plug-in. Whew!

For over five years, Flanagan has occupied a prominent place on my bookcase. The 4th edition of his "Desktop Quick Reference" has gained a lot of weight: just over doubling the 438 pages of the 1996 version. Hardly any flab, though.

A Pair of Penguins

It's no secret that I'm a Linux user and a Linux enthusiast. Over the past two years or so, more developers have turned to Linux to provide solutions for embedded systems. Hollabaugh's presentation is more than merely adequate, but I have to admit that I still prefer John Lombardo's presentation (published by New Riders last year).

And here's the book for every penguin-user! Nemeth's UNIX handbook has spun through several editions since 1989. Each one more impressive than the previous. Now here's the Linux version, written by Nemeth and her colleagues, Garth Snyder and Trent Hein. It covers Red Hat, SuSE and Debian. It is well-written. If you use Linux or if someone on your site uses Linux, this book is indispensible. Thanks, Evi. Thanks, Garth. Thanks, Trent.

book reviews

Virtual Machines

A virtual machine enables the operator to pretend to use one OS while running on top of another. VMWare does this for a variety of Windows platforms, Linux and FreeBSD. Ward's presentation is good, though I have a minor problem: my notion of how to "get the most out of Windows" is to just run something else.

Lisping

It has been over a decade between editions of Bob Chassell's Emacs Lisp book. The new edition contains a really fine tutorial as well as the new features included in GNU Emacs v21. It's definitely worthwhile. Bob, another fine piece of work.

IP SANS: A GUIDE TO ISCSI, IFCP, AND FCIP PROTOCOLS FOR STORAGE AREA NETWORKS

TOM CLARK

Boston: Addison-Wesley, 2002
ISBN: 0-201-75277-8

Reviewed by Steve Reames
reames@diskdrive.com

Storage Area Networks (SANs), and storage networking in general, are becoming increasingly important components in corporate data centers. Traditionally dominated by Fibre Channel, a recent plethora of IP-based standards are competing to either augment or replace the entrenched standard. Clark has done an excellent job of putting together the important aspects of these new protocols, and comparing and contrasting them in a fair and even-handed way. If you have anything to do with storage networks, you need to read this book.

IP SANs is just over 280 pages long and packed with illustrations. It would be easy to jump into the technical details of the standards, but instead Clark takes his time and covers the background material that will be needed later. The first few chapters cover shared storage, Fibre Channel, SCSI, and TCP/IP. The level of coverage is introductory, and the knowledgeable reader can skim through these chapters. But network experts learning about storage or storage gurus who need to learn about networking will find one or more of these chapters worth careful reading.

Chapter 8 is the heart of the book, where iSCSI (Internet SCSI), iFCP (Internet Fibre Channel Protocol), and FCIP (Fibre Channel over Internet Protocol) are discussed and contrasted. Clark's extensive Fibre Channel background really shines here as he is able to examine how these new protocols interact with existing Fibre Channel systems. You can spend weeks combing the standards (I've done it!) trying to learn what Clark clearly explains in a few simple pages.

It turns out that these protocols do not really stand on their own, and the next three chapters are devoted to iSNS (Internet Storage Name Server), security, and QoS (Quality of Service). The iSNS protocol provides naming and discovery services that are inherent to Fibre Channel, but require an additional server for IP-based storage protocols. The chapter on security discusses the issue from two perspectives. First, security in Fibre Channel SANs is discussed. This gives the reader the needed perspective for the second section, which covers the same issues for IP-based SANs.

A short chapter on Infiniband gives some insights about how this interface may or may not become part of SANs in the future. The chapter on SAN applications is thorough and detailed, and probably provides the most extensive collection of real-world scenarios yet compiled. Even if you have a Fibre Channel SAN and intend to stay with it, you need to read this chapter to understand how the industry is developing. A final chapter of conclusions examines the potential of IP SANs, and outlines the things that need to happen for IP SANs to succeed.

No book is perfect. In the explanation of RAID (Figure 3-1 in the book), a "0 + 1" RAID is shown as a mirror of two striped arrays (RAID 0, then 1). Almost no one implements it this way; instead, they stripe together a set of mirrored drives (RAID 1, then 0). On page 174 the lower drawing should read "Tunnel Mode Security Association" as opposed to "Transport Mode." To spend more than a couple of sentences on the book's flaws would be an injustice to the tremendous collection of knowledge contained within. This book is destined to stand for many years as the reference on IP SANs.

SOFTWARE FOR YOUR HEAD: CORE PROTOCOLS FOR CREATING AND MAINTAINING SHARED VISION

JIM AND MICHELE MCCARTHY

Boston: Addison-Wesley, 2001. Pp. 464.
ISBN: 0-201-60456-6

Reviewed by Steve Johnson
yacc@yaccman.com

This is one of the more unusual books on software engineering available today. Some of the ideas it proposes are brilliant, some are weird, and some are both.

In an earlier book (*Dynamics of Software Development*), Jim McCarthy states that "Software is the process of turning ideas into bits." He goes on to make a convincing case that the big problem in software engineering is not the individual programmers' abilities to turn their individual ideas into bits but, rather, "aligning all the ideas in the various programmers' heads." If the ideas are aligned, any problems one person may have realizing these ideas are quickly caught and remedied. If the ideas are not aligned, problems will be frequent, contentious, and difficult to find and eliminate.

Software for Your Head is an attempt to provide some algorithms or patterns a

group can follow that will lead to this alignment of ideas. The underlying premise is that most groups are working at a fraction of their possible potential, and by improving the group functioning you can get large productivity gains and increase everyone's job satisfaction. People who have worked in both high- and low-functioning groups will have no trouble with this concept. The key to the "software problem" is not reading another book on multiple inheritance, or even using the latest requirements language, but getting the group aligned and functioning at a high level.

By this time, many readers are probably stifling yawns. There are lots of "teamwork" workshops that teach people their Meyers-Briggs scores and purport to teach "trust" by hanging on ropes. The authors are scornful of such remedies, which typically do not hold individuals accountable for their actions in the group and do not articulate the desired group behavior clearly.

One concept in particular struck me. The authors believe that every group functions at the level of the least engaged person. In many meetings, some people want to be there and so have a vested interest in seeing the issues resolved; others are just warming chairs. These disengaged people may see things that they could contribute but have the attitude, "Why should I speak up? It will just prolong the meeting . . ."

The authors see this attitude as a loss of integrity. If people do not function at the highest level, they are weakening themselves and the group. And, more to the point, they hold the group (and themselves) back. The authors believe in making high-quality group participation a conscious goal. It is not necessary to be fully engaged all the time. But it is required to be honest with yourself and your coworkers about whether you are "in" or "out" at any time.

The authors also believe that when someone is "out," it is often because they are feeling some emotion (work-related or not) that is interfering with their full engagement. So one way to allow more people to be "in" more of the time is to allow emotions, even negative ones, to be acknowledged within the group. They are not suggesting that software teams become encounter groups or therapy groups – far from it. Just that it should be OK for someone whose daughter is in the hospital to acknowledge it in the group, and through this honesty help the group to become stronger.

Another idea put forth in the book is that the "product" of management is, in fact, the group behavior. So if management is unhappy with the group, it needs to look at its own actions and attitudes for the root cause of the problem. This is the kind of obvious statement that never seems to be obvious to the managers themselves.

The description of group maladaptive patterns, and how to break out of them, accounts for some of the most brilliant and insightful writing. The authors are frequently quite vivid in their language. For example, they discuss the phrase "He gets an A for effort." In other words, they point out, he failed, and he wasted a great deal of time and energy failing. They prefer the motto "Fail Cheap."

I found the authors to be very effective in describing the problems that hold groups back and articulating how an effective group would function. I have more difficulty with the remedies they suggest. These are overly structured, even draconian, for my taste. They may well be effective, but I believe other techniques can also be effective and ultimately less invasive.

There are many examples in our society of cooperation at a much higher level than most software teams achieve. Sports teams are an obvious example. A less obvious example can be found by going to an opera (turning ideas into notes), where hundreds of people execute virtuoso feats with split-second accuracy over a span of several hours, with rarely any major errors. Most software teams would literally think this is impossible.

If, as I believe, cooperation is built into our nervous systems (like language and the ability to reason), creating an effective group should be as easy as creating a safe place for our "cooperation genes" to express themselves. And, sometimes, it is this easy.

Of course, musicians are trained from early on to communicate with other musicians, the conductor, and the audience. Athletes in team sports get similar training. Programmers, on the other hand, are still taught as if success depends solely on one's ability to understand multiple inheritance, data structures, and complexity theory. Not only do students not learn to cooperate, but their professors – role models – function in universities that often prize individual achievement over cooperation as well, reinforcing this isolationism.

Is it surprising, then, that many software people find working in a group boring, difficult, and frustrating? And their managers, promoted because they were the best coders, are not typically trained to break this cycle. In fact, the good programmers with integrity often turn down or resign from management jobs, recognizing that they are contributing to the problem rather than the solution. With this background, perhaps some draconian techniques *are* necessary, and may well be effective.

I invite you to read the book and make up your own mind. It will make you think and may challenge your assumptions; even the ideas that you might not accept are interesting and strangely compelling. And they grow on you.