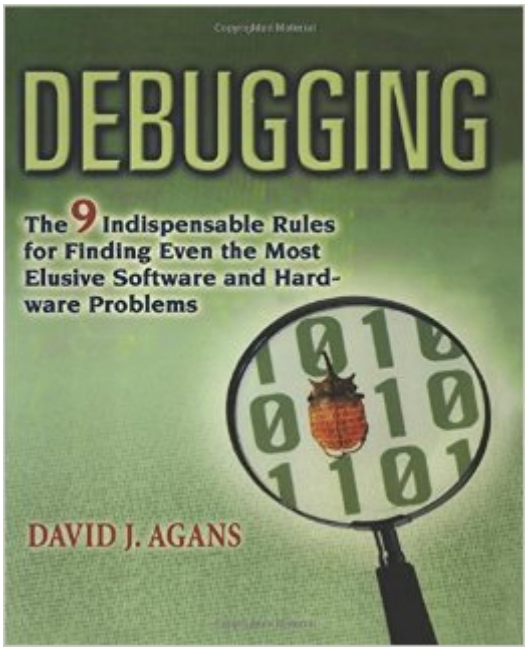
Debugging: The 9 Indispensable Rules For Finding Even The Most Elusive Software And Hardware Problems



Synopsis

The rules of battle for tracking down -- and eliminating -- hardware and software bugs. When the pressure is on to root out an elusive software or hardware glitch, what’s needed is a cool head courtesy of a set of rules guaranteed to work on any system, in any circumstance. Written in a frank but engaging style, Debugging provides simple, foolproof principles guaranteed to help find any bug quickly. This book makes those shelves of application-specific debugging books (on C++, Perl, Java, etc.) obsolete. It changes the way readers think about debugging, making those pesky problems suddenly much easier to find and fix. Illustrating the rules with real-life bug-detection war stories, the book shows readers how to: * Understand the system: how perceiving the ""roadmap"" can hasten your journey * Quit thinking and look: when hands-on investigation can’t be avoided * Isolate critical factors: why changing one element at a time can be an essential tool * Keep an audit trail: how keeping a record of the debugging process can win the day"

Book Information

Paperback: 192 pages

Publisher: AMACOM (November 5, 2006)

Language: English

ISBN-10: 0814474578

ISBN-13: 978-0814474570

Product Dimensions: 7.3 x 0.5 x 9.1 inches

Shipping Weight: 14.1 ounces (View shipping rates and policies)

Average Customer Review: 4.7 out of 5 starsÂ Â See all reviewsÂ (46 customer reviews)

Best Sellers Rank: #203,426 in Books (See Top 100 in Books) #8 inÂ Books > Computers & Technology > Programming > Languages & Tools > Debugging #153 inÂ Books > Engineering & Transportation > Engineering > Industrial, Manufacturing & Operational Systems > Robotics & Automation #316 inÂ Books > Computers & Technology > Business Technology > Software > Enterprise Applications

Customer Reviews

This book is very useful for beginners and intermediate programmers. "Debugging" is full of practical advice on debugging in general. It is not tied to any particular programming language.The book describes 9 main debugging "rules", and many smaller "sub-rules". The rules (such as "Make it fail" or "Quit thinking and look") and sub-rules (such as "Start from a known state" or "Build instrumentation in") are derived from common sense and years of experience. Many people know

most of the rules, but perhaps do not systematically follow them. "Debugging" clarifies and makes a systematic review of the debugging practices, with examples taken from real life, simplified to remove the jargon.The book is quite funny and makes enjoyable reading. I am looking forward to more: perhaps we can see more stories in the next edition, or in a companion volume, or on the debugging rules web site.

If, after twelve years or so of work, you find yourself bringing junior team members along and filling the role of mentor or lead, you are expected to impart your accumulated experience and wisdom onto the newcomers, so that the hard-won lessons need not become a re-nvention of the wheel. The most common problem in doing this is succinctly and effectively communicating the messages. This book does exactly that, for debugging problems.It’s worth noting that the author is very careful to distinguish between troubleshooting and debugging. Getting to the root cause of a problem is debugging; fixing the painful symptom(s) is troubleshooting. The analogy used in the book is this: instead of wiping off the oil spots so you don’t slip and fall, secure the machine with four bolts instead of two. Keeping the floor clean is troubleshooting. Securing the machine so it doesn’t vibrate, loosen the fittings, and leak oil in the first place, is debugging.Debugging is hard. It requires discipline, attention to detail, and patience. The toughest leap of faith for anyone trying to adhere to the system might be in slowing down and proceeding in a meticulous, measured fashion; however, it is almost always exactly what’s required to zero in on a problem’s root cause, and accomplish the job.Mr. Agans liberally fills the book with real-life war stories of how each rules is applied (or not applied); in some instances he can detail a good story illustrating every single rule. The rules cover exactly enough ground, and are reinforced well by examples. There’s absolutely no filler in this book, despite any incomprehsible claim to the contrary. In fact, one might well wish for more examples in order to help make the necessary analogies to someone else when teaching from this book.Here’s a good example. It’s my own war story showing how I used the rules, and how I didn’t..... I was trying to figure out why a particular application wouldn’t export output to Excel. It was an out-of-the-box feature from the vendor, but the application is extremely complex to configure and I’d misconfigured something somewhere (it has to be me, no one else was brave enough to touch this application).I had been wrestling with it, trying this configuration file and that one, all to no avail. I finally broke down and emailed the vendor, who put e in touch with a support specialist.At this point, I had been changing too many things at once (rule #5 violated), not keeping a good audit trail because I was moving too fast (rule #6 violated), and clearly not looking (rule #3 busted). The only rule I’d tried was #8 "Get a Fresh View". The tide turned.We started from square one, and rule #1

"Understand the System" came into play. We walked over all the possibilities, listing where exporting to Excel lays in the application. This also helped us to narrow down our search (rule #4, "Divide and Conquer"). Next, we started changing one thing at a time (rule #5) while keeping a good audit trail of what worked or didn't (rule #6). At one point, we'd revisited a section of code that sparked something in my memory. We'd downloaded a fix and forgot to fully test it. It was integral to exporting to Excel, so I backed out the fix, and export to Excel ow worked. It turns out the fix solved one problem but broke exporting to Excel. We put the fix back in, saw the problem (rule #2 "Make it Fail"), and backed out the fix. Now we'd proven that we'd found the real culprit. The vendor investigated and shipped us a new patch that worked *and* didn't break anything else.If I had to teach all that to someone, I'd end up making a far less effective book than this one. Mr. Agans has done all the hard work here, and now I can give this book to junior team members, quiz them after reading it, and relate our own issues with respect to a common approach and means of describing the situation. Without a doubt, a classic book and one that every single professional problem-solver should own, in any field that has a diagnostic component to it.-Fred

David Agans does a great job of explaining how to approach debugging as a science rather than an art. If you're a novice programmer, the information here will prove invaluable; discovering how to debug effectively on your own can take many years.Experienced programmers may consider most of the rules to be obvious; however, those same programmers might be surprised to find how many of these obvious rules they neglect to follow. I've been debugging for more than 20 years, and still learnt some useful new tricks.Peppered throughout the text are a large number of war stories from the author's own experience with embedded systems. As well as illustrating how to (and more commonly, how not to) approach a particular problem, these are all well written and often entertaining.Some of my favourites: how wearing the wrong shirt to work caused a new video compression chip to crash; teenagers coming home from school subtly altering the behaviour of a video conferencing system; a vacuum cleaner that made the house lights flash on and off; a noisy read/write line that led a junior engineer to mistakenly redesign an entire co-processor memory circuit; and the self-test feature on an old Pong video game.Although most examples are hardware related, the approach described can be applied to almost any problem; indeed, several of the examples used have nothing to do with computing.This is not a large book, but it's well laid out, easy to follow, and doesn't talk down to the reader. It's also packed with enough meat to satisfy the hungriest of programmers. Highly recommended.

I worked with Dave Agans for over 10 years and I can tell you first hand the man knows what he's talking about. From developing hand-held controllers in the late eighties to single-board OS/2-based videoconferencing products to software collaboration tools, we have debugged problems of every ilk. Whether the problem was an FPGA bug, a faulty component in a board, a race condition in a device driver or a dangling pointer in a DLL, Dave always approached the problem with his same set of debugging rules, and they never let him down. Read this book. It's engaging and fun to read. But more importantly it will make you a better debugger, whether you're debugging hardware, software or your lawnmower.

Debugging: The 9 Indispensable Rules for Finding Even the Most Elusive Software and Hardware Problems Software Engineering Classics: Software Project Survival Guide/ Debugging the Development Process/ Dynamics of Software Development (Programming/General) Raspberry Pi Cookbook: Software and Hardware Problems and Solutions The Hidden Lives of Owls: The Science and Spirit of Nature's Most Elusive Birds Great Soul of Siberia: Passion, Obsession, and One Man's Quest for the World's Most Elusive Tiger Zodiac: The Shocking True Story of the Hunt for the Nation's Most Elusive Serial Killer Rules of The Trade: Indispensable Insights for Online Profits Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers Computer Organization and Design, Fifth Edition: The Hardware/Software Interface (The Morgan Kaufmann Series in Computer Architecture and Design) Computer Organization and Design: The Hardware/Software Interface (The Morgan Kaufmann Series in Computer Architecture and Design) Getting Started with 3D Printing: A Hands-on Guide to the Hardware, Software, and Services Behind the New Manufacturing Revolution Code: The Hidden Language of Computer Hardware and Software Linux Enterprise Cluster: Build a Highly Available Cluster with Commodity Hardware and Free Software Make: FPGAs: Turning Software into Hardware with Eight Fun and Easy DIY Projects Applying Design for Six Sigma to Software and Hardware Systems (paperback) Code: The Hidden Language of Computer Hardware and Software (Developer Best Practices) Complete CompTIA A+ Guide to IT Hardware and Software (7th Edition) Computer Networking from LANs to WANs: Hardware, Software and Security (Networking) Application Debugging: An MVS Abend Handbook for Cobol, Assembly, PL/I, and Fortran Programmers (Prentice-Hall Software Series) Surreptitious Software: Obfuscation, Watermarking, and Tamperproofing for Software Protection: Obfuscation, Watermarking, and Tamperproofing for Software Protection