Alex Groce (agroce@gmail.com), Northern Arizona University

Peter Seibel's *Coders at Work: Reflections on the Craft of Programming* is 600-odd pages of extraordinary software engineering wisdom and insight. It is also self-contradictory; these pages often disagree with each other, and sometimes actually argue with each other. As Donald Knuth says, near the end of this book, "the key is usually to have a stereo view instead of a one-dimensional view."

This book, inspired by the famous Paris Review *Writers at Work* series, consists of interviews with 15 luminaries of software creativity: Jamie Zawinski, Brad Fitzpatrick, Douglas Crockford, Brendan Eich, Joshua Bloch, Joe Armstrong, Simon Peyton Jones, Peter Norvig, Guy Steele, Dan Ingalls, L. Peter Deutsch, Ken Thompson, Fran Allen, Bernie Cosell, and Donald Knuth. One of these subjects appeared in the previous *Passages* selection consisting of a series of interviews, Federico Biancuzzi and Shane Warden's *Masterminds of Programming: Conversations with the Creators of Major Programming Languages*. This isn't a problem; Simon Peyton Jones is well worth listening to twice.

For comparison, the first (of many) volumes of the Paris Review series interviewed: E. M. Forster, Frank O'Connor, Dorothy Parker, Nelson Algren, Francois Mauriac, James Thurber, William Faulkner, Robert Penn Warren, Georges Simenon, Francoise Sagan, William Styron, Thornton Wilder, Truman Capote, Angus Wilson, Joyce Cary, and Alberto Moravia. I think we have to give the nod to Malcolm Cowley's collection, here, though I believe the Nobel and Turing counts are equal, and I don't know if anyone really reads Algren or Cary any more (more's the shame in Cary's case; his "First Trilogy" is wonderful).

Each interview is different, and the emphasis slowly shifts from modern JavaScript and web issues to topics dating back to the very beginning of computer science. However, Seibel makes it easy to contrast the ideas of the interviewees by asking certain questions (Seibel calls them his "standard questions") of every, or almost every, subject.

These standard questions include: "How did you learn to program?", "Do you still enjoy programming?", "What's your preferred debugging tool?", "Do you use assertions?", "Do you use formal proofs?", "How do you go about reading a piece of software you didn't write?", "Top-down or bottom-up?", "How do you hire people?", "In your teams, does one person own a piece of code, or does everyone share ownership?", and perhaps most interestingly for the big picture, "Do you think of yourself as a scientist or an engineer or an artist or a craftsman or something else?" Seibel also asks most interviewees if they have ever used Knuth's literate programming approach.

The repeated mention of literate programming, and the overall link to the Paris Review series, connects this book to a fundamental theme of *Passages*: code is like writing, and is in fact written as much for other humans to read as for computers to execute. Many of the writers, I mean coders, Seibel talks to discuss this fundamental point, and many have strong opinions

about how to make code that is a pleasure to read.

The answers to the full set of common questions, as you would expect from the list of coders, vary widely. There are some interesting commonalities: nobody really proves code correct; many people fear and loathe C++ or C, or both, though Thompson offers up a gruff argument in refutation to the C-related complaints.

I don't intend to summarize the things that are said in this book; there are too many of them, and most importantly, the meat of the book really lies in the un-summarizable personal stories of how each of these luminaries became involved in computing, and some of what each one of them actually did. The list of questions above suggests the fundamental questions of software addressed here. The book covers larger issues of project management, program structure, and approaches to refactoring, as well, and it's safe to say that almost no important issue of software engineering is entirely neglected. This book is a treasure trove that should be on every software engineer's shelf. Seibel asks some of his subjects: "Do you still program *for fun*?" and the answers suggest one final important common opinion of these diverse folks: we're lucky to be doing this!