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

Gerald M. Weinberg's *The Psychology of Computer Programming: Silver Anniversary Edition* is this month's Passages selection. Weinberg died last August, but this book alone makes him immortal in the world of software engineering. The *Psychology* is one of my favorite software engineering classics, and one of the books that originally motivated this column. Tim Budd gave me my first copy, when I was a young and ignorant professor at Oregon State, when he retired (the opposite of the usual retirement gifting procedure) and I read it with great pleasure and attention for the first time, not long after, and began to form the idea of Passages not long after that. Why did I not write about it before this?

The question forces a confession: I've been saving this one. My major worry, when I started these columns, was that at some point I would run out of interesting, actually suitable, books and start stretching the definition to the breaking point, including too many out-of-left field reviews. I wanted some surprises (I don't feel bad about shoving Hugh Kenner or Confucius into the limelight, such as it is, of *Software Engineering Notes*) but I don't want to always be unpredictable; that's just another kind of predictability, and lowers the useful entropy of the column. *This* is a suitable book, with no doubt, and so long as I have not written on it there is one more good column left to write. So, then, have we reached desperation? No, I simply decided there were too many good books in the queue to deny myself the pleasure of writing about this one any longer.

Where to begin? First, let's take a quotation that is dear to my own heart:

"Now, what cognitive dissonance has to do with our programming conflict should be vividly clear. A programmer who truly sees his program as an extension of his own ego is not going to be trying to find all the errors in that program. On the contrary, he is going to be trying to prove that the program is correct—even if this means the oversight of errors which are monstrous to another eye. All programmers are familiar with the symptoms of this dissonance resolution—-in others, of course."

Is this still relevant? Well, you do it, I do it; birds and bees would do it if they coded. It is perhaps the great threat to testing your own code: you don't want it to be wrong. There are modern mitigations that aim to encourage what Weinberg calls "egoless" programming: open source, issue trackers, pull requests, continuous integration, and, most importantly, code reviews, all, arguably, partly have the goal of rewarding egoless behavior, or at least making egocentric coding riskier. Egoless programming, however, is not the subject of a vast treatise, which it easily could have been; in different hands, almost every idea in Chapter 4 (*The Programming Group*) might have been the subject of a small and good, or large and tedious, book. But Weinberg says enough to get you thinking, and moves on.

Perhaps you are thinking "I am sure Weinberg meant well, but a book called *The Psychology of Computer Programming* written in 1971 is likely to be full of psychological experiments that don't replicate, and even worse, mere anecdotes from the early days of the field. I don’t want to swallow whole a bunch of ill-supported fictions, however charmingly presented.” The modern replication crisis holds no threats to Weinberg’s book, which is notably clear about being provisional, exploratory, and thought-provoking, aiming to start a field, not solve it. Yes, Weinberg is full of opinions, and supports many of them with (semi-)experiments, or with anecdata. The support is more illustrative than bludgeoning, however; you will not be bullied by Weinberg, but can disagree while learning. You must think of this as being like reading a better-organized “Montaigne on code” than reading a hectoring article in *Psychology Today* telling you what not to do with your life, and why the p-values for studies on 45 undergrads prove it beyond a doubt. Weinberg uses psychological notions, like cognitive dissonance, but does not rely on their exact conformance to reality. There are technical terms here, but much more “common sense” than dubious science.

I am biased, of course; perhaps no other book so closely follows the concerns of this column. Weinberg’s first chapter suggests, audaciously, that programs should be read, frequently, by humans, and even treated as a literature. Weinberg’s bibliographical notes are miniature Passages columns, e.g.:

“Festinger, L. A., A Theory of Cognitive Dissonance, Evanston, Ill., Row, Peterson, 1957. Festinger's work on cognitive dissonance grew out of an earlier study of what happens when a group which prophesied the end of the world saw the day arrive and the world go on. (When Prophecy Fails). Obviously, dissonance theory has a lot to say to people who work in an environment where prophecy fails each and every day—especially the prophecy that the program is sure to work this time, now that the last bug has been removed.”

In fact, one route to becoming a well, if idiosyncratically (the best kind of syncratic!), educated thinker about the deep issues of computer programming and software engineering would simply be to read each chapter of *Psychology*’s bibliography, and pick and read the three most interesting sounding books or papers. You might end up reading William James, Frank Lloyd Wright, R. A. Fisher, Betty Friedan(!), Polya, and a big bunch of PL/1 manuals, and being all the better for it.

As great as the bibliographies are, the most notable repeated structural element of *Psychology* is the set of QUESTIONS at the end of every chapter. These are divided into questions for managers and questions for programmers. Those who are both (25% of computer science professors, for example) are free to answer both sets. The questions are always interesting, and highlight the “let’s think together” exploratory nature of the book. The chapter on reading programs concludes:

“2. If you are a higher-level manager, are your first-line managers capable of reading programs written by their programmers? Are you sure? Ask the programmers themselves, then answer this question again. Then find out if the first-line managers actually do read programs, even if they are capable. Our surveys indicate that nine-tenths do not, for one reason or another. Do you think it is possible for a first-line manager to know how good his programmers are or how well they are doing without occasionally reading their programs? *For Programmers* 1. When was the last time you read somebody else's program? Why has it been so long? When was the last time somebody else read one of your programs and discussed it with you? Was it your manager?..."

I could simply end this review by noting a few more of Weinberg's most interesting ideas, most amusing stories, most insightful questions, or most provocative (even wrong, but exciting) claims, but perhaps it would be better to say where this book fits in the world of books that Passages explores. Every book can be described, in part, by explaining which other books it is most like, and how much it is like them. Some books are described by the greatness of magnitude of the distance that a good metric would measure: nothing else, that does not imitate it intentionally, is all that much like *Moby Dick*, *Ulysses*, Proust's *Remembrance*, *There Are Doors*, or *The Intrepid Gnomes.* Some books are best described in the opposite way: if you have read one mystery novel about a cat, you have sorta read them all. Where does Weinberg fit? I would say that, to restrict ourselves to previous Passages selections, Weinberg is significantly different from, but located "in between" a set of other key books in the field: *Peopleware*, *The Mythical Man-Month*, *The Sciences of the Artificial*, *The Pragmatic Programmer*, and *The Elements of Programming Style*. In each case, *d*(book X, *Psychology*) is quite large, but these are the nearest neighbors. That is a great place to be, the meeting of many fresh and life-giving waters.

One final warning is in order. Perhaps you are a manager. This book is certainly *for* managers, as the devotion of half of the probing questions that end each chapter to questions exclusively for managers proves. But Weinberg is hard on managers, e.g.:

"If any one thing proves that psychological research has been ignored by working managers, it's the continuing use of half-partitions to divide workspaces into cubicles. I might have thought that DeMarco and Lister's *Peopleware* would set that issue to rest once and for all. That is, I might have thought so if I hadn't learned about the managerial psychology that puts social status above productivity."

Programmers come in for their share of criticism here (who do you think has the ego to frustrate?) but most stories with a villain in this book offer up a clueless or malevolent manager as antagonist. There's nothing unfair about it, and I think Weinberg admires good managers deeply; but I'm just not sure he really believes they exist.