You Are What You Read

Philippe Kruchten

One of my favorite questions when interviewing software developers is, “What books, journals, or magazines related to our trade have you read lately?” If they’ve read a lot, then we have a springboard for further discussions, drilling down, and probing: “Why this topic, why this author, what’s interesting in this publication, any article in particular that struck you?” Quite often in the past 10 years, however, the answer has been “not much.” A person in a technical field such as ours not reading anything—books or articles—is a bad sign. As I described in this column a few months ago (and some readers wrote to me to confirm my conjecture), the “biological half-life” of our software engineering knowledge is probably around 5 years. So, if concepts, ideas, tools, or techniques in our field have such a short shelf life, we must constantly replenish our brains’ content. We can’t stop learning new things, or we’ll get empty pretty rapidly, as well as useless and obsolete. We must constantly learn new tricks, and reading good, serious material about a topic is still an excellent approach to learning new tricks.

What to Read

I can already hear some of you objecting that this is “so 20th Century,” that all we will ever need is now readily available at the end of a simple Google query, but I maintain my position. A Google query will lead you to a lot of often small, disparate, disconnected pieces of information. Assessing how solid and validated this information is can be difficult, with a few exceptions (such as Wikipedia). You’re left to sort out and synthesize the real knowledge yourself. To get to a deeper understanding about a given topic, you need more than bite-sized chunks of information—you need material that has been carefully thought out and that’s presented with plenty of evidence, examples, illustrations, and so on. A book (be it electronic or paper; the medium isn’t the point here) or a long paper on one specific point will more likely achieve this goal. Ivar Jacobson used to say that any technical book has only 50 really useful pages and that the problem is to find them. He’s probably right, but finding them isn’t the game: the other N – 50 pages play a big role, too, giving us the appropriate context, motivation, discussion, and line of reasoning that the authors used to get to the more crucial pages.

This reading business requires some thought and organization on how to

- select what to read,
- free up time to read,
- get the right reading material at the right time and place, and
- organize the material so that you can later exploit what you’ve read and learned.

I maintain a backlog of material to read in a few different granularities. Subscriptions to journals and magazines feed my backlog with a flow of small and medium-size material, whereas books fill it with more substantial bits. Bookmarks on a Web browser capture the small pieces. Then, for quick access, I tag them by current topic of interest: software architecture, software process, education, and so on.

My friend and former colleague Joe Marasco made it a personal rule to read 12 substantive books per year, or roughly one a month. Moreover, he often made a point, when the book was good, of writing a review and sharing it with his Rational (and
not so rational) colleagues, thus feeding my own backlog. His reviews—especially when we’d read the same book or at least books on similar topics—led to interesting debate, forcing us both to be more reflective and critical about what we’d read. I later found this to be a good practice: taking random notes, scribbling on paper or the book itself. If I must sit down and write a review of the material—even hypothetically—I have to think much more carefully about what I just read or learned. The journey is the destination; I write to better understand what I think.

Retaining What You Read

A much harder problem is capitalizing on what you read, keeping enough reminders around to let you retrieve the knowledge when you need it. Gerald Weinberg once described the “fieldstone” method for writing. Fieldstones are stones that vary in size, color, texture, shape, and density; they’re most often used to build property lines and fences. Building a fieldstone structure requires gathering the right stones in a step-by-step process; you might not use them in the same order you find them. Although I initially accumulated “fieldstones”—which 20 years ago consisted of scraps of paper or notes in the margins—when doing my own writing, I now find the process valuable when reading to simply keep track of what I’ve learned, or of the useful information. I have “repositories” of stones in various formats: little notebooks, folders on my computers and PDA, or email and bookmark folders. I even use specialized tools such as EndNote (www.endnote.com), RefWorks (www.refworks.com), or Zotero (www.zotero.org), or some note-taking or note-grabbing tool such as EverNote (http://evernote.com). Similarly to naming conventions in programming guidelines, you’ll need to define a few rules on how to label your stones, to ease sorting and retrieval when they become numerous and old. For electronic material, I now use something like <Lastname>-<Year>-<Keyword>-<Keyword>-pdf, as in “Boehm 1987 Spiral Model.pdf.”

Because online publications are fluid and disappear, morph, or migrate, I tend to gather the stone itself, creating a PDF on my computer rather than just a bookmark to a page. Retrieving the stones I need from my computer has become much easier with the advent of desktop tools such as Google Desktop, Windows Search, or Spotlight, and my naming conventions help somewhat.

Another practice I developed late (too late) is to keep track of where my major keystones—that is, the good books—are going. We tend to lend friends, colleagues, or grad students only our good and very good books, not the mediocre ones. Consequently, most very good books are missing from the shelf when we need them, sometimes forever. By now, I must have bought four copies of Fred Brooks’ Mythical Man-Month (Addison-Wesley, 1975), and three of Herbert Simon’s Sciences of the Artificial (MIT Press, 3rd ed., 1996). And I wonder who in Vancouver has my Fowler’s UML Distilled (Addison-Wesley, 2nd ed., 1999) and my Strunk & White (Longman, 4th ed., 1999). Now I label the books I lend prominently, and I keep track of the borrowers in a little paper notebook.

A

s you might have guessed, I am a bookworm. I’m also saddened by the recent demise of a few print publications: Byte, ACM Queue, and Dr. Dobbs Journal, to name some I personally read. Although I read more and more electronic material, the print publications are much handier for reading on the bus, the plane, my long ferry rides, or in the dentist’s waiting room.

It seems you’re reading IEEE Software, so you must be on a good track. Especially if someone like me is interviewing you next week! And if you’re looking for a good book to read, check Jurgen Appelo’s list of the 100 best software engineering books at http://tinyurl.com/100sebooks.

“Remember what the dormouse said: Feed your head, Feed your head, Feed your head.”

Reference


Philippe Kruchten is an avid collector of software engineering books, living in rainy Vancouver, British Columbia. Unfortunately, his collection seems to suffer from serious leaks. Send kudos, rants, and flames about this column to kruchten@ieee.org.