

<http://yourdon.com/personal/blog/2006/08/18/the-greatest-software-ever-written/> <acesso em 09/09/2006>

The Greatest Software Ever Written - E.Yourdon

The journalist Charlie Babcock has written a fascinating article with his “top 10” list (which actually contains twelve items) of what he considers the greatest software products/systems/programs ever written. (..)

You should definitely read his entire article, but if you’re in a hurry, here’s the list:

1. Unix
2. IBM’s System-R database product, forerunner of relational databases
3. Gene-sequencing software at the Institute for Genomic Research
4. IBM’s System 360 operating system
5. The Java language
6. The Mosaic Web browser
7. American Airlines’ Sabre reservation system
8. The Macintosh operating system
9. Excel, as a robust, industrial-strength implementation of Visicalc
10. The Apollo spacecraft guidance system
11. Google’s search ranking software
12. The Morris worm

I might quibble with a few items on this list, but Babcock has put a lot of thought into his choices (as you’ll be able to see by reading his article), and has presumably had his initial thoughts confirmed and refined by the group mentioned above. I might have an epiphany about something they’ve all overlooked after I mull things over for a few days, but for now, I’m happy to take Babcock’s list as a darn good starting point, if not the final word on the subject.

Still, I’d like to know why things like e-mail and word-processing didn’t show up on the list. I guess there’s no single example of an exemplary e-mail program or word processor that changed the world, but as a *class* of software, their impact on society has been enormous. If I had a little more time, I would take advantage of some links at the end of the article to participate in a poll, offer my own candidates for “best software ever,” and see a list of software products that almost made it. Obviously, you can do the same.

But there’s something far more important I think we should be doing instead of quibbling over the past. Let’s accept Babcock’s list without any further quibbling, and then ask ourselves: what does it tell us about the next generation of “greatest software” that we might look forward to in the next 40 years? Keep in mind that Google’s page-ranking software was introduced only about 5 years ago (again, let’s not quibble about the details), Java was introduced a little over 10 years ago, and Mosaic appeared sometime in the early 90s. But the Mac operating system first appeared some 20 years ago, in 1984; Unix appeared in the early 1970s, and Sabre, Apollo, and IBM’s 360 software date back 40 years, to the 1960s.

So we shouldn’t necessarily expect the next item on Babcock’s list — which might appear as #13, or might shove one of the current top-12 entries out of its spot — within the next year or two. It might take five years, it could easily take a decade, and perhaps even a full generation of hard work and inspiration by software engineers just entering the work force today. But what will it be? Will it be something entirely serendipitous, something we haven’t even imagined in today’s computing environment? Or will it be something we *know* we want, but just haven’t figured out to do — e.g., a truly awesome AI system (Babcock discusses a couple that almost made his list, but weren’t quite impressive enough), or a truly accurate long-term weather forecasting system, or a robust, industrial strength real-time any-language-to-any-language translation program? Does the existing list of “greatest software” offer us any clues? I don’t know, at this point, but I’m certainly going to think about it...

Equally important: how should we expect the next several instances of “greatest software” to come into existence? I was fascinated by a comment in Babcock’s description of the #1 entry on his list, Unix: “Bell Labs often gets credit for creating the Unix operating system, but Bell never funded its development. *In fact, the labs’ management knew nothing about it* (emphasis added).” This is apparently how a lot of Google’s new products come into existence — subsidized, essentially, by the engineers being allowed to spend 20% of their time pursuing their own R&D initiatives — and it’s obviously in stark contrast to the kind of top-down, hierarchical development efforts typified by the American Airlines Sabre system, or the Apollo spacecraft guidance system. (..)

There’s no doubt that software has transformed our world during the past 40 years, and I’m optimistic enough to think that if we don’t blow ourselves up with nuclear weapons, or destroy our environment with global warming, we will have an opportunity to continue transforming our world for another 40 years, as today’s generation of computer science students (and millions of others who have figured out how to do great software-related things on their own) emerge from universities and enter the workforce. Heck, some of them are already doing great things with their computers in high school. But it would be nice if we could give them a “wish list” of things we’d love to have them tackle for us. And in the meantime, it would be a good idea to be prepared to be “blindsided” by some awesome new software that we haven’t even thought about yet.