A bigger sociology of programming languages.

Camille Akmut
October 27, 2019

Abstract
On “anti-intellectual” programming languages, and Larry Wall.
1 Two “anti-intellectual” languages: Java and Go

In a recent conversation, Go had been described to us as an “anti-intellectual” language. This is perhaps the best description this particular programming language has been given.\(^1\)

While official descriptions\(^2\) are mostly limited to the letter, this one is most true to the spirit of this language.

Our interlocutor had meant that given the same function to write, two programmers were likely to come up – in this programming language – with similar if not identical outcomes.

Programmers tended toward creativity and a wish for self-expression, he argued; this one limited them, by limiting their possibilities.

But, he assigned virtues to it; while we can add here that this serves the other purpose of making programmers easily replaceable — like screws to screwdrivers (or as cogs in a wheel).

This too was certainly no random feature of it, considering the context of its creation and design (within Google).

I proposed that Java had fulfilled the same function in previous decades. He disagreed, having in mind its current incarnation with its inclusion of various new elements like idioms imported from functional programming, among other things.

But, certainly it would have been true for this language as it had been for a long time, up to a certain version, and year (he himself did not recognize it when compared with earlier versions e.g. 5).\(^3\)

(All the while, I kept thinking about Perl as being an anti-thesis to this, at the other end of this sociology of programming languages.)

2 Mind cubicles

To the identical, repetitive structures of cubicles and open office designs implemented by technology companies, that represent the infrastructure of their world, are superimposed rigid mental structures: that of the programming languages in which programmers (are made to) think.

In these “anti-intellectual” languages, thus, employees are turned themselves into functions of identical, reproducible outputs.

The industrial revolution, and the capitalism of Manchester had brutalized bodies, but had comparatively left minds alone.

The executives of the information age on the other hand have extended the shackles of the mind much, much further than any of their predecessors could have ever dreamed to do, or been able to do.

(“Go here, go there, do this, do that” they mostly limited themselves to say still, then.)

---

\(^1\)It is worthwhile to note that this irreverent description had come from someone who was no outsider, but on the contrary occupied a leading position at a technology company.

\(^2\)Found in abundance in the de facto official manual for this programming language, by Donovan/Kernighan.

\(^3\)The mere presence or addition of new features in a language does not automatically lead to their adoption or use in the corporate place or in a regulated environment.
These new systems of control – affecting the mind more so than the body – we call here “mind cubicles”.

Bent backs come before bent minds. This is the approximate law of all modern information societies, in their worst phases.4

The machineries seen in the factories of Metropolis drive the protagonist, who is unable to keep up with their infernal rhythms and demands, mad (the same who later reconciles workers and factory owner?).

Previously mostly exterior structures have now truly and finally been made to become interior ones: invisible, silent, hard to resist, protest.

Some even love them —

As did house slaves with their masters, who, in renditions by Malcolm X, said “us” when they only meant “them”, because in their heads both had become the same.

Such was the degree of their domination.

Needless to weigh in the mind what these lines may have for relevance when it comes to an entire class of modern technology workers; but, some of them have descended far below in their delusions: with one arm outstretched and a fist closed they mimic the fights of black people, and with the other, open, they collect their checks...

4If one could only skip the 21st, 22nd and 23rd centuries to reach that ideal one imagined by certain science fiction television writers. But, we must live in the first of them; the worst, not the worst of them most likely.
3 Larry Wall’s sociology

Previously, we had highlighted the importance of Jean Cavailles: “the pairing of an old, humanist education with a knowledge of science – an explosive combination”, we had about written.

Computer scientists often look for their “heroes” – a terminology favored by this peculiar culture, favorable to hagiographies – in the wrong places.

A search for antecedents, an infinite endeavor and activity, would halt at Wall as a conservative, but temporary choice for our nebulous figure of the “social computer scientist”.

An altogether exceptional though not unique figure in computer science, many may prefer to limit his contribution and role to that of a mere programmer: his identity so far removed from theirs, arguably a threat to theirs, to them.

But, not only did he move into their art and science coming from a previously seldom seen background (linguistics / religious studies), this transition did not occur at the price of renunciation, of doing away with it.

On the contrary, he injected elements of his own culture into theirs, with much courage for he did not have many peers to look back to, and continues to do so. The outside became the inside, here too.

Elements of a sociology of programming languages are not found in Steele’s work only. Larry Wall has offered his own interpretation, and in some ways had gone further (having come after, too).

We cite here from an interview in which he retraced the sociological-historical characteristic or traits of a variety of languages.

This came as an answer to the question “What are the five languages everyone should know?”, but could not occur before a rebuttal. To others, that question might have seemed natural, an answer readily available.

Oh boy. That’s a really tough question. It’s kind of like asking “What are the five countries you should know about?” if you’re not interested in geology or, uh geography or politics. And, the answer varies depends on what your actual interests are. Or: “What are the five companies you should know?” And, the answer changes over time too. Back when I was getting started lo these many decades ago, the answers would have been: FORTRAN, COBOL, BASIC, LISP and maybe APL. And, those were very formative languages back then. People learned a lot from them.

But, these days it might be more important for you to know uh... JAVASCRIPT! Even if the only reason you know that is...

\[\text{Oh boy. That’s a really tough question. It’s kind of like asking “What are the five countries you should know about?” if you’re not interested in geology or, uh geography or politics. And, the answer varies depends on what your actual interests are. Or: “What are the five companies you should know?” And, the answer changes over time too. Back when I was getting started lo these many decades ago, the answers would have been: FORTRAN, COBOL, BASIC, LISP and maybe APL. And, those were very formative languages back then. People learned a lot from them. But, these days it might be more important for you to know uh... JAVASCRIPT! Even if the only reason you know that is...}\]

\[\text{Oh boy. That’s a really tough question. It’s kind of like asking “What are the five countries you should know about?” if you’re not interested in geology or, uh geography or politics. And, the answer varies depends on what your actual interests are. Or: “What are the five companies you should know?” And, the answer changes over time too. Back when I was getting started lo these many decades ago, the answers would have been: FORTRAN, COBOL, BASIC, LISP and maybe APL. And, those were very formative languages back then. People learned a lot from them. But, these days it might be more important for you to know uh... JAVASCRIPT! Even if the only reason you know that is...}\]
to know whether or not click the "Enable Javascript" in your browser (...)

the elephant in the room is sort of JAVA : you can’t really make a list of modern languages without talking about it. Java is sort of the COBOL of the 21st century. It’s heavy-weight, verbose, and everyone loves to hate it – though not everyone will admit that. But, managers kind of like because it looks like you’re getting a lot done. You know, if a 100 lines of Java code accomplish a task, then it looks like you’ve written a 100 lines even though in a different language it might only take 5 lines7. (...) But, because it is considered an industrial language and programmers are sort of interchangeable parts, managers like it for that. And, for that reason a lot of Java jobs have been outsourced from the United States. (...)

Going in a different direction, coming from academia, we have a language like HASKELL, which we call a functional programming language. That means function in a mathematical sense, not because other languages are dysfunctional... (...) Haskell is one of those languages that mathematician-type minded people love. It’s sort of a language for geniuses, by geniuses... (...) and you know you have to hire some really smart people to program in it. Haskell is sort of a modern kind of LISP, in a sense.8 (...)

What else? Well, we can’t leave off modern languages without talking about C. The C language is actually about 40 years old, but people have tried to replace it with other languages that are like it and have by and large not succeeded, because C is a very minimalistic language (...) it let’s you do very fine-grained stuff, very efficiently, but it’s a lot of hard work. But, once you’ve done that work you can run it pretty much everywhere.

So, almost all of the other languages that you see – Java, Perl, whatever – actually if you look underneath, they’re actually implemented in C, or in a closely related language. (...)

And, finally (...) one of the scripting languages (...) PYTHON, RUBY but of course I’m prejudiced in favor of PERL, because I think it has the liveliest community (...)

More is found in Programming Perl, with the third edition holding that “the so-called "fourth generation languages" make it easy to do some things, but nearly impossible to do others [while] so-called "industrial-strength" [ones] make it equally difficult to do almost everything.”...
References


— Schwartz, Randal and Wall, Larry. 1991. *Programming Perl*. (Followed by later editions and more co-authors.)

Description : buffoonery, from buffoon or jester : a nobleman-appointed servaent used for the distraction of their court. Buffoons danced and sang; on all feet, to all songs, and for all audiences... The Middle ages continue to serve as a mirror to our average (moyen) days... Project : parse Le Goff,Bloch, Febvre and Duby books and replace "noble" with "bourgeois" or "capitalist" alternatively, "fief" with "factory" and "start-up (building)" instead...

— Lang, Fritz. 1927. *Metropolis*.
Description : the modern-day equivalent of this strange, aborted film is found in season 3 of *Mr. Robot*. The workers, seen rioting just a moment ago, are shown shaking hands with their capitalist the next. Meanwhile shutting down banks appeared as too extreme in the end for these fashionable anarchists... (A 5th season would have shown them as proper, settled, unrepentant bourgeois Libertarians (complete with Netflix subscription, Amazon deliveries, Tesla cars and Nest thermostats... The whole cycle.).)