A small sociology of programming languages; beyond Guy Steele.

Camille Akmut

October 22, 2019

Abstract

Extension of a sociology of programming languages started by Guy Steele in the introduction to *Scheme and the Art of Programming* (1989).

”This notation is rather formidable for a human to read”...

Introduction : social computer scientists

A small, but important tradition within computer science exists : they who did not consider their discipline as a mere “technical” subject, matter, enterprise or challenge.

In being and thinking so, they were not alone, not without peers : we can assign to this tradition computer scientists like Edgar Dijkstra, Richard Stallman, Larry Wall or Peter Landin – to name only a few.

Their purported eccentric ways – exceptional only in a field whose dominant model of the scientist favors naively a-political attitudes above all else – turned out to be among the best their discipline ever had — and was ever able to produce.

This was despite great, enduring faults starting with its curriculum and attached sets of implied values : the false, cheap veneer of false or immature sciences. Rejecting politics as exterior to itself, yet full of it wherever the eye reaches.

Guy Steele did not lack a sociological eye either, except for when it came to himself.

—

The separation of ”the technical” and ”the political” – a foundational act of this discipline – serves many purposes, but let us express clearly here its most important function : this simplistic, untenable and unrealistic dichotomy fulfills only the interests of a certain class, or category of people.

They serve to bolster those who appear regularly in our newspapers due to their ever-greater audacity and escalating, shameless buffoonery : the Zuckerberg’s, Sandberg’s, Pichai’s of this world.. Computer scientists and students are their cannon fodder; too often, all too willing.

Unrepentantly, we can repeat here : Watson must have thought not unlike many of them in June of 1934, when he visited Berlin for the second time, and when IBM though its subsidiary IBM Germany (then known as Demahog) had turned greed into virtue : it served everyone, and most zealously so...

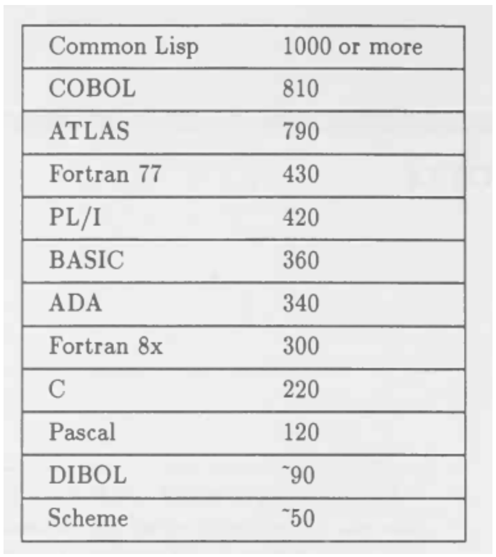
1 A “small” language.

In the introduction to *Scheme and the Art of Programming*, Steele had created a small sociology of programming languages.

His theory : “*Small is (...) powerful, [because] small is easy to understand.*”[1]

Based on this, he created a table of programming language standards ordered by their respective lengths, reproduced down below :

[1]Ibid., “Foreword”, xiv.



Common Lisp	1000 or more
COBOL	810
ATLAS	790
Fortran 77	430
PL/I	420
BASIC	360
ADA	340
Fortran 8x	300
C	220
Pascal	120
DIBOL	~90
Scheme	~50

In this paper, our contribution does not go much beyond updating it and highlighting its importance and greater context, not least for the current era.

Published in the first, 1989 edition of this textbook, it is now extended, exactly 30 years later.

Using language drafts and standards that have appeared since, we came up with the following, updated table :

Java 8	780
Java (2nd ed.)	530
C# (ECMA-334, 1st ed.)	490
Haskell 2010	320
Haskell 98	270
Javascript (3rd ed.)	180
Python 3 (3.5)	150
Python 2 (2.7)	130
Javascript (1st ed.)	110
LISP	$\frac{1}{2}$

But, we can go a bit further by noting that a general trend seems to be that the more corporate a language (Java/Oracle, C#/Microsoft, etc.), the more lengthy its specification. COBOL - the businessman's language - now replaced by Java and Go.

In a previous article we had claimed these languages' popularity was based on corporate power, used to establish them as dominant : advertisements campaigns as heavy as their books, specifications.

—

As to how Guy Steele was able to accomplish the intellectual split required by moving from Scheme to Java (later JavaScript), this conundrum of the mind, a true “jigsaw puzzle”, we would much rather leave to future sociologists, applied and theoretical.

Going from working with hackers of some ideals, to joining the very heart of 1990’s big business : he had learned a lesson then that still escapes us and his successors have perfected now.

References

”Haskell 2010 Language Report”. https://www.haskell.org/definition/haskell2010.pdf
”Haskell 98 Language and Libraries. The Revised Report”. https://www.haskell.org/definition/haskell98-report.pdf
”The Python Language Reference”. https://docs.python.org/2.7/download.html
”The Python Language Reference”. https://docs.python.org/3.5/download.html
”The Java TM Language Specification. Second Edition.” http://www1.cs.columbia.edu/ sedwards/papers/gosling2000java.pdf
”The Java R Language Specification. Java SE 8 Edition.” https://docs.oracle.com/javase/specs/jls/se8/jls8.pdf
ECMA-262, 3rd ed.. ”ECMAScript Language Specification”.
ECMA-262, 1st ed.. ”ECMAScript: A general purpose, cross-platform programming language”. https://www.ecma-international.org/publications/standards/Ecma-262-arch.htm
McCarthy, John. 1959. ”Recursive functions of symbolic expressions and their computation by machine”. In particular, page 13.
Black, Edwin. 2001. *IBM and the Holocaust.*