Should Programming Languages Hang On English?:

An argument for Culturally Sustaining Programming

Ryan Enser enser@buffalo.edu

Background

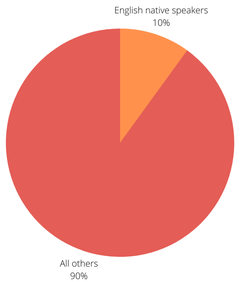
The most popular programming languages used around the world are all constructed from the natural language of English, but English is spoken natively by only about 400,000,000 people, less than 10% of the global population (Lingoda, 2021; TIOBE Software BV, 2021). English is even the standard language of supporting materials and tools for those learning and working in programming languages further exacerbating the English language barrier to accessing programming languages (Ruby & David, 2016).

Even though English dominates computer programming languages, it is not because its characteristics make it better suited for constructing them. In fact, natural languages are not better or worse than any other in their characteristics or their usefulness as communicative tools. As the field of linguistics developed over the last 100 years it has abandoned its raciolinguistic prejudice that deemed some languages, especially European languages, superior to others (Dixon, 2016; Flores & Rosa, 2015). This breakthrough produced the current view within linguistics that all languages are similar in the value of their characteristics and in their utility for communication.

Exclusion

Because English is as useful as other languages for communicative purposes, its exclusive role as the source language for nearly all programming languages can best be explained by historical and sociolinguistic factors. Historically, software engineering, which programs computers with programming languages, largely began and continues to be conducted in English-speaking parts of the world. To help illustrate this, North America only contains about 6.5% of the global population, but commands 40% of the share of the information technology market (U.S. Department of Commerce, 2021). The historical narrative of how software engineering advanced in English-speaking parts of the world is a useful model for how software engineering could be developed in other linguistic contexts, but it should not be used to limit software engineering to English-speaking parts of the world.

Sociolinguistically, the English language has been used as a tool for empowering some and disenfranchising others, so it is not a stretch to propose that this same phenomenon is happening with programming languages as well. Nelson Flores has written extensively on raciolinguistic ideologies that exclude minoritized Americans from educational opportunities based on racial and linguistic variation (Flores, 2020; Flores & Rosa, 2015). How much more could raciolinguistic ideologies be responsible for the exclusion of international participants in software engineering based on even greater racial and linguistic variation? Within schools, efforts to decenter the dominant English variety in the United States have been met with fierce resistance from educators, as well (Metz, 2017). How much more resistance could there be from programmers to the effort of decentering English from its dominant position in programming languages?

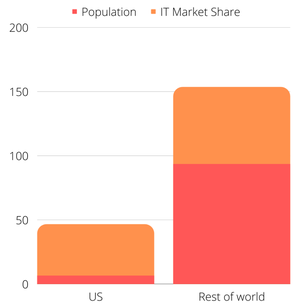


Comparison of English speakers and all other language speakers

Culturally Sustaining Programming

To empower native language programmers to develop their own programming languages, culturally sustaining pedagogies, an approach from the field of education principally developed by Django Paris (2017), could be adapted for software engineering as culturally sustaining programming. Culturally sustaining pedagogies flows from the intersection of education and cultural pluralism and can be adapted as culturally sustaining programming at the intersection of software development and cultural pluralism.

In a similar way to how culturally sustaining pedagogies views the plurality of cultures and languages of students as an asset, culturally sustaining programming views the plurality of cultures and languages of programmers as an asset, as well. This additive model empowers native language software engineers to derive programming languages from their own native languages.



Example of English-speaking country's (United States) disproportionate share of the information technology market

Affordances

Culturally sustaining programming affords many benefits for languages and cultures traditionally excluded from programming. Prospective programmers would be able to bypass the high barrier of learning English to access programming languages, which would surely open the field of programming to many more aspirants. These native language programmers would then develop new native language software engineering industries that would economically and materially benefit their linguistic group. Additionally, programming languages derived from natural languages other than English would become a linguistic and cultural asset of the linguistic group. For example, an Arabic-derived programming language and the products created by that language would surely strengthen the linguistic and cultural identity of Arab speakers. Surely there are even more culturally sustaining programming affordances beyond these stated here, as well.

Conclusion

The linguistic hegemony of English as the sole source language for all popular programming languages has historically and sociolinguistically excluded those whose native languages are not English from software engineering. The approach of culturally sustaining programming provides many affordances for prospective native language programmers and the linguistic populations that they represent.

References

1. Dixon, R.M. (2016). *Are some languages better than others?* Oxford University Press.
2. Flores, N. (2020). From academic language to language architecture: Challenging raciolinguistic ideologies in research and practice. Theory into practice, 59(1), 22-31.
3. Flores, N., & Rosa, J. (2015). Undoing Appropriateness: Raciolinguistic Ideologies and Language Diversity in Education. Harvard educational review, 85(2), 149-171.
4. Lingoda. (2021). What are the main English speaking countries? Retrieved November 29, 2021 from https://www.lingoda.com/en/content/english-speaking-countries/
5. Metz, M. (2017). Addressing English teachers' concerns about decentering Standard English. English Teaching: Practice & Critique, 16, 00-00.
6. Paris, D., & Alim, H. S. (2017). Culturally sustaining pedagogies: Teaching and learning for justice in a changing world. Teachers College Press.
7. Ruby, I., & David, S. (2016). Natural-Language Neutrality in Programming Languages: Bridging the Knowledge Divide in Software Engineering. In P. Zaphiris & A. Ioannou, Learning and Collaboration Technologies Cham.
8. TIOBE Software BV. (2021). TIOBE index for November 2021. Retrieved November 29, 2021 from https://www.tiobe.com/tiobe-index/
9. U.S. Department of Commerce. (2021). Select USA: Software and Information Technology Spotlight.

University at Buffalo The State University of New York
Department of Learning and Instruction
Graduate School of Education
ed.buffalo.edu