The Programming Language Wars

Questions and Responsibilities for the Programming Language Community

Andreas Stefik
University of Nevada, Las Vegas

Stefan Hanenberg
University of Duisburg-Essen
In the last 50 years, science has expanded at a daunting pace, in part due to computing technologies.

In modern society, physicists, biologists, psychologists, and many others use computation as a part of their daily work.

Programming languages are at the core of computation: they let us tell the computer what we would like it to do.
As computer scientists, or especially language designers, we have significant influence in how programming languages are created, thereby impacting how computation is performed.

Stewardship, by definition, involves a number of key observations:

- When we create new languages, it causes communities to shrink/grow.
- When we make alterations, it may have significant cost to academia/industry (e.g., changing textbooks, software).
- When we alter language products, or create new ones, it may cause duplication of effort (e.g., reinventing a new hash table).
The term the “programming language wars” describes the chaotic reality of programming language design/use in the real world.

We think it involves at least three core ideas:

- **Language Divergence** – many languages exist and their designs are often incompatible.
- **Language Impact** – scientific experiments are rarely conducted to evaluate the impact of programming languages on people or society.
- **Language Communities** – How language designs impact various communities (e.g., physicists, chemists, psychologists) is poorly understood.
Humans Have Natural Conflicts of Interest in Language Design

- Before we begin, let us remember a few important points about people that use languages

- We can define at least these properties:
  - **Owners**: people that own a programming language (e.g., Oracle, Microsoft)
  - **Followers**: followers are those that use a language without ownership
  - **Believers**: those that claim a particular language design is good or bad, with or without evidence
  - **Dependents**: Those that rely on the use of a programming language (e.g., PHP and Facebook)
  - **Volunteers**: Those that use a language product with no obvious conflict of interest

- We need *objective and data-driven* ways to evaluate the impact of language designs that are external from conflicts of interest
What might the Outcomes of the Programming Language Wars be?

- We often hear quite a few arguments about the language wars. We have given the two most popular names:
  - **One Language to Rule Them All**: The belief that any solution to the language wars requires all possible tasks under all possible conditions to be solved best with one language
  - **Unique Snowflakes**: The belief that all developers are so unique that they require their own programming language

- Both of these views seem pretty unlikely, but discussing possible outcomes may provide insight into this problem
It seems unlikely we will ever get to the point where one programming language will fit the needs for all possible developers under all possible conditions, because:

- Humans vary (although we poorly understand how)
- Problem domains vary (although we poorly understand why)
- Even good languages (whatever that means) may not be adopted

But it does bring up a few questions …
We might want to know …

Which domain specific languages benefit humans?

What language designs help the largest user base possible under the greatest number of circumstances?

Answers to these questions should be based on solid, replicable, scientific evidence.

But how much evidence have we been gathering?
However, Language Designers often make claims that Divergence is Positive

- For example, consider the “idea” of domain specific languages
- We can hypothetically come up with situations where a domain specific language “might be” helpful
  - Is ANTLR easier to use for parsing than Java?
  - Is R easier to use for psychologists conducting statistics than using a statistical package in Python?
  - Is scripting with Lua in World of Warcraft easier for young players compared to writing scripts in a C++ style?
- We can dream up answers to these questions, but if **rigorous scientific measurements were made**, what would the **size of the effect** be and **which direction would it go** (positive or negative)?
Are we all just *Unique Snowflakes*?

- Alternatively, it could be the case that the language wars is just an expression of our individuality as humans. If true, several claims must be verified through rigorous evidence gathering, namely:
  - Standard deviations in studies on programmers must be so high that no statistical impacts can be determined using standard procedures used in modern science (e.g., the 6-sigma standard in physics, \( p < .05 \) in psychology)
  - Having a unique language for all developers would have to *improve* productivity for an individual or organization overall

- Is this view supported by evidence from the literature? With some confidence, we can say that *the answer is no.*
The Unique Snowflakes view is Refuted by the Evidence

- Consider a few points now known from randomized controlled trials with human developers:
  - On average, static typing improves developer productivity under a wide variety of conditions.
  - Threads, compared to software transactional memory, leads to approximately 8-fold more bugs amongst those in the sample.
  - Some languages (e.g., Perl, Java) have symbols so poorly chosen that a randomly generated language is just as easy for a novice to initially use.

- If the Unique Snowflakes view is correct, these results provide meaningful counterexamples from carefully gathered evidence.
Example 1: Static Typing* affords Greater Human Productivity

*In this context, we are referring to static typing where the annotations are made explicit.

Figure 3: Boxplot for raw development time measurements

Figure 7. Boxplot for second round (no repeated measurement of same tasks)

ICSE 2014

OOPSLA 2012
Example 2: Language Representation Impacts People*

Token Accuracy Maps tell us which tokens cause problems

* This does not just mean novices: Programmers’ Build Errors: A Case Study (at Google) Hyunmin Seo, Caitlin Sadowski, Sebastian Elbaum, Edward Aftandilian, and Robert Bowdidge, Hong Kong University of Science and Technology, China; Google, USA; University of Nebraska-Lincoln, USA, ICSE 2014
In Academia, we have little systematic data collection even on what languages are used.

Our initial step isn’t perfect, but we have started analyzing this:
- We looked at 39 universities in the mid-west
- Of these, we saw the languages C, C++, Alice, VB, Python, Java, C#, COBOL, ADA were used in first courses
- So far as we could tell from discussing the choice with scholars, no reliable scientific evidence was being used for making the decision

Lesson: Scholars seem to just do whatever they want and, sometimes, claim the choice was the right one
The Impact of the Language Wars on Society is Poorly Studied

What is the total cost throughout history of developing all computer programming languages, supporting libraries, and tools?

Of the time used in developing programming languages, what percentage was spent reinventing the same solution?

If language designers created languages with an increasingly larger evidence base, what would the impact be over the next century?

What is the total historical productivity lost by switching/updating programming languages in various settings (e.g., academic, industrial, military)?
Language Designers *Generally do not* Study Language Impact Systematically

- As most of us know, designers very much so do study the mathematical properties of programming languages (e.g., type soundness, algorithmic correctness, efficiency)

- For understanding the impact of language designs on people or society, however:
  - Mathematical proofs tell us little about the impact of a feature on an individual or group (e.g., 1st year novices, children, 4th year professionals at Google)
  - Mathematical proofs are obviously the gold standard, but *only in situations where they can provide meaningful information*

- We find a significant lack of evidence about societal/human impact in the scholarly literature (examples to follow)
We often code scholarly papers from the literature, documenting:
- Did an author have evidence about language design?
- Did it follow standard scientific conventions (e.g., it had a control group)?
- Workshops have little evidence. Primary conferences also have little
- To-date, we have coded approximately 2,200 papers in the language conferences
Evidence Gathering *Needs* to be *Part of Our Scholarly Culture*

*The DSM is a standard manual for understanding mental disorders.*

How can we increase the programming language design community’s evidence standards?

Can we create a catalog, using the DSM5* as an exemplar, for reliable tests on language impact?

* The DSM is a standard manual for understanding mental disorders.
Computer Scientists Have Responsibilities

The language wars is one of the most problematic issues in our discipline and we need to take action.

Let’s start with responsibilities of scholars and language communities.
Let’s take a detour into Aspect-Oriented Programming

The first paper on AOP came out in 1997

Results from this idea *seem* genuinely appealing (at least to me)
  - We really can think of examples where cross-cutting concerns could be a problem (e.g., logging)
  - It came from a background in language design, meta-object protocols, compiler construction, etc.

But was there evidence that cross-cutting concerns were a problem or that the solution was actually helpful?
In the case of AOP, it took more than a decade before an independent randomized controlled trial was run.

Overall, *measurements* of the impact of AOP appeared to be limited.

Independent verification of claims need to come much sooner.

Figure 3. Box plot for measured results
If it Ain’t Broke, Don’t Fix it.

Java

- Consider JDK 5
  - Largely about generics
  - This release also added new ways to iterate over a collection (with no supporting evidence)

- What about JDK 8’s new features?
  - Do any studies exist?
  - What is the impact on pros?
  - What is the impact on students?
  - How much will it cost to change/purchase all the new textbooks?

Boo

- The Boo manifesto says:
  “Sometimes it is appropriate to give up the safety net provided by static typing. Maybe you just want to explore an API without worrying too much about method signatures or maybe you’re creating code that talks to external components such as COM objects. Either way, the choice is yours not mine.”

- The problem, though, is that this is refuted by the evidence

- But is this author really doing anything different than what scholars at our prestigious conferences do (not use evidence for human factors decisions)?
Impose Science on the Chaos of Organic Evolution*

Reducing replication of effort, and cost could substantially benefit society.

*A phrase by Sovereign from the video game Mass Effect.
Disseminate *Verifiable* claims about Language Design

- While conducting empirical studies is important, unless developers in industry know about the results, little will change.

- Since major corporations own some of the more popular language products, just telling other scholars will have marginal impact.

- We think the following activities *may* be beneficial:
  - Do more outreach with industry partners, discussing the results of empirical studies *before* they implement new versions.
  - Do more outreach with developers, so they can better differentiate between data-oriented investigations and marketing hype.
  - Do more outreach at conferences and with students on what we actually know about the human or social impact of language design.
The Software Industry has Responsibilities

- We need more information about how programming languages are used in practice.
- One way we do this now is the TIOBE index.
  - Unfortunately, this index is methodologically weak.
  - It’s unclear how accurate or unbiased it really is.
- Since running experiments is costly, such information may help us prioritize.
Language Changes Should Have Evidence: *Consider the C++ 11 standard*

What was the measurable impact of each decision? How much of an impact? Under what circumstances? What is the impact on different kinds of people (e.g., first year students, second year, 6th year developers)? *Why is little/no evidence being gathered on clearly important considerations before putting these things into a standard accepted world-wide?* Is industry to blame or can we get it to help?
Educational Institutions have Responsibilities

- Educational institutions have a broad impact on how language technologies will ultimately be used.
- These institutions have at least two major responsibilities:
  - Analyze the impact of language designs on everyone, including students
  - Teach empirical methods to programming language students
- They also need to engage less in marketing and focus on evidence.
A claim empirical study designers often get is as follows:

“Hey, I like this study, but you tested here with group X and I’m convinced, without evidence, that group Y would give different results.”

With students, we need many more studies, in part because we have evidence that programming languages are incredibly difficult for this group to use.

We could realistically test anyone on this spectrum:

---

Lesson: In science, generalizability is measured through replication over time. It cannot be guessed, or assumed, by authors or reviewers.
Which of these claims are based on evidence?
Students Might Benefit from Changes to Principles of Programming Courses

- Principle of programming courses are often a survey of programming languages.
- Scholars often choose one, or several, languages (e.g., a functional one, an OOP one) and go over their approach.
- Rarely do scholars teach how differences in language design impact people or communities.
- Students need to know evidence-based arguments about such issues if they are to distinguish facts from marketing.
We need to think more deeply about the programming language wars, before we leave a mess for the next generation to clean up.

- The programming language wars is a major social ill in our discipline:
  - We duplicate effort massively
  - We reinvent the wheel constantly
  - We deploy features world-wide with nigh a shred of evidence of its impact on almost anyone at almost any level in almost any organization
  - Some parts of what we create are probably helping, while other parts may do harm, but we have no idea which or why

- We need to take action on the programming language wars, especially through a significant re-thinking of our foundation of evidence, through outreach to a variety of communities, and through substantive changes to our educational programs
Throughout history, every mystery, ever solved, has turned out to be … not magic.

- Tim Minchin, Storm

I imagine the problems we have discussed in our paper, and in this talk, seem daunting to many.

Ultimately, making any progress at all on this problem will take time and the efforts of a broad community.

Thanks for listening.

If you are interested in language designs based on evidence, check out the Quorum project:

www.quorumlanguage.com