Steve Francia
3 Nov 2025

Programming-Languages Leadership Engineering-Culture Management Economy

✨ Series Overview

This is the first entry in a series on the true cost of a programming language. When you are finished here, read Part 2: **The 9 Factors of a Language's True Cost** ↗.

The Leadership Blindspot: How Identity Drives Multi-Million Dollar Technical Debt

A programming language is the single most expensive choice a company makes, yet we treat it like a technical debate. After watching this mistake bankrupt dozens of companies and hurt hundreds more, I've learned the uncomfortable truth: these decisions are rarely about technology. They're about identity, emotion, and ego, and they're destroying your velocity and budget in ways you can't see until it's too late.

Early in my career, I worked at Takkle, a promising social network. A sudden departure vaulted me from lead engineer to VP of Engineering, leading a team of 12. While we were delivering on all our goals, I was in my early 20s and lacked experience, a risk our board wanted to fix. They pressured our CEO to recruit a CTO with more experience. I looked forward to learning from him; he was a well known figure in the Perl community and arrived with a stack of O'Reilly "camel" books.

One of his first acts was to pronounce our language, PHP, the wrong choice. He decreed a switch to Perl. This decree happened after what felt to me like a sham analysis comparing PHP and Perl.

Our velocity collapsed. Our team had to not only learn a new language but rebuild from scratch, delaying our product by nine months. Our monthly burn rate jumped from $200K to $500K as we more than doubled our size to make up for the lost velocity while building the new Perl based system, which halved our runway.

Our CTO did deliver, at least on some of his promises. We built a beautiful system, one I was truly proud of. But it was too late. By the time we finally launched, the market opportunity had vanished proud of. But it was too late. By the time we finally launched, the market opportunity had vanished.

Facebook had now expanded beyond colleges, and we were at the end of our monitary runway. The increased spend had shortened our runway by half, and we didn't have enough momentum with the new site to reach the milestones required to raise more money.

I've often wondered: What if we had just stuck with PHP? We had a fine system and real momentum. We would have launched much earlier at a fraction of the cost. PHP was good enough for Facebook; why not us?

But the question that haunted me was: **Why did such an experienced leader make such an terrible mistake**?

- Switching to Perl would unlock the architecture we needed
- Rebuilding from scratch would accelerate hiring and quality

- Velocity collapsed as the team relearned and rebuilt everything
- Burn rate jumped from $200K to $500K per month

The Pattern Repeats

As my career progressed I saw this same pattern over and over. As Languages Product Lead at Google, my group included C++, Java, Go, and Python. At MongoDB, I managed teams writing in 13 different languages. In both places I saw brilliant engineers arguing past each other, armed with conflicting data, all of it true, but none of it complete. At Google Cloud, I saw these same challenges across our customers.

Fast forward two decades from Takkle, and I had déjà vu. I watched as a VP of Engineering presented to leadership why his team needed to build their next system in Rust. The presentation eerily paralleled that Takkle experience. In that old presentation, nearly every reason the CTO gave for Perl was truer of PHP at the time. Now, every single reason given for choosing Rust in this presentation, Go was objectively better at. As an example: they cited "easy build and deploy" as a

Rust advantage. It’s true that this is a strength of Rust, but Go’s nearly instant cross compilation and single static binary is even stronger than Rust in this specic critera with Rust’s very long build times.

It’s not that I think they should have chosen Go, Go would have been the wrong choice for their situation, and I believe Rust was the right choice. But what struck me was how broken their reasoning was. If they were making a logical argument, surely they would have considered Go and in doing so with their presented criteria they would have realized Go was a better option and, at the very least, refined their critera.

I pulled the VP aside after the meeting. “Walk me through how you evaluated other language candidates,” I said. His face went blank. “We… didn’t really look at any others,” he admitted. “Everyone’s talking about Rust.” There it was: a 50 million dollar decision made on hype, about to be green lit.

For me this was the moment of epiphany, finally an answer to the question for the beginning of my career. The presentation didn’t share an analysis, they hadn’t done one; it was a justification for a choice already made. This was a decision based purely on hype, emotion, and identity.

Every Technical Debate Is Really Two Conversations

In every language discussion, two conversations are happening simultaneously.

: “Rust has memory safety without garbage collection.” “Go has faster compile times and easier deployment.” “Python has the richest ML ecosystem.”

: “I am a Rust programmer.” “I want to become a Rust programmer.” “I cannot imagine being someone who doesn’t choose Rust.”

If you just read that and thought “well, my last language choice was different, I was being rational,” your invisible conversation is running right now, defending itself while you read this sentence.

your invisible conversation is running right now, defending itself while you read this sentence.

My CTO at Takkle was having the invisible conversation. Every point in his visible Perl analysis was technically true, but it felt like a sham because it was only there to cover the much deeper invisible conversation. He wasn't evaluating languages. He was protecting an identity he'd spent a decade building. What our company paid $300K per month extra for wasn't better architecture or faster hiring. We paid for the opportunity for him to be a Perl CTO instead of a PHP CTO. That was the real transaction. The rebuild was just the payment plan.

That VP's Rust presentation listed "easy build and deploy" as an advantage, technically true, but Go is objectively better on that specific criterion. If they were truly having the visible conversation, they would have caught that. They would have at least considered Go in their analysis.

They hadn't. Because they weren't having that conversation at all. And they were about to spend $50 million on the invisible one.

The Neuroscience of Why You Can't See Your Own Bias

In one of the most **fascinating studies** done in the past 20 years, researchers set out to understand why people cling to beliefs even when confronted with overwhelming contradictory evidence. What **they discovered** fundamentally changed our understanding of human decision making.

The researchers recruited participants and first identified which beliefs were central to each person's identity—their core political views, their fundamental values, the beliefs that defined who they were. Then, while participants lay in an fMRI scanner, the researchers presented them with carefully constructed challenges to these identity based beliefs, alongside challenges to beliefs the participants held but weren't central to their sense of self.

The brain scans revealed something remarkable: these two types of challenges activated completely different neural pathways.

When a **peripheral belief** was challenged, something the person believed but wasn't core to their identity, the brain's reasoning centers activated normally. Participants could consider the evidence, weigh the arguments, and sometimes even change their minds.

But when an **identity based belief** was challenged, the brain responded as if under physical attack. The amygdala, your threat detection system, the same system that fires when you encounter a predator or a physical danger, activates immediately. The insular cortex, which processes emotional pain and disgust, lit up with activity. Most tellingly, the brain's Default Mode Network, the system that maintains your sense of self and personal narrative, went into defensive mode, working to protect the existing identity rather than evaluate the new information.

In other words, your brain wasn't weighing evidence. It was defending itself from an existential threat.

> The researchers' conclusion was stark:

Your brain can't objectively evaluate challenges to identity based beliefs because doing so requires temporarily dismantling the neural architecture that defines who you are. It's not a matter of being more rational or trying harder. The mechanism that would allow you to see the bias clearly is the same mechanism the bias has compromised.

Think about what this means in practice. Every time an engineer evaluates a language that isn't "theirs," their brain is literally working against them. They're not just analyzing technical trade offs, they're contemplating a version of themselves that doesn't exist yet, that feels threatening to the version that does. The Python developer reads case studies about Go's performance and their amygdala quietly marks each one as a threat to be neutralized. The Rust advocate looks at identical problems and their Default Mode Network constructs narratives about why "only" Rust can solve them.

We're not lying. We genuinely believe our reasoning is sound. That's what makes identity based thinking so expensive, and so invisible.

We've Built Our Industry Around the Wrong Conversation

We call ourselves Pythonistas, Gophers, Rustaceans, we wear these labels like badges, sometimes we even wear literal badges (t-shirts, stickers, etc). There's a reason so many of our surnames come from people's crafts: Potter, Smith, Brewer. What we do becomes who we are. What looks like decorative labels are really decision making frameworks that operate beneath conscious thought.

We've built our entire industry around the **visible conversation**. We train engineers to debate technical merits. We create decision frameworks based on feature matrices. We think if we just gather enough benchmarks and case studies, we'll make the right choice.

But the **invisible conversation** is much stronger. It's why my CTO chose Perl. It's why that VP chose Rust. And it's operating in your next language decision right now, invisible and unexamined.

The moment you hire a Rust developer to evaluate languages, you've already chosen Rust. You've just added a $2 million feasibility study to make the predetermined decision feel rational.

The Real Cost

The question isn't whether this bias exists, the science is conclusive. The real question is: can you afford to let it make your decisions?

Because the invisible conversation has a price tag. Industry research suggests that technology stack decisions account for 40-60% of total development costs over a product's lifecycle.
Research by Stripe ↗ found that developers spend 42% of their time on technical debt. Wh

Research by Stripe ↗ found that developers spend 42% of their time on technical debt. When you let identity drive that decision you're mortgaging your velocity, your budget, and your runway to pay for someone's sense of self.

The visible conversation is about technology. The invisible conversation is about identity. And the invisible conversation always wins.

A New Framework: Language as Economic Decision

So how do we win, when the invisible conversation is constantly working against us? Change the conversation entirely.

Instead of asking "which language is best?" we need to ask "what is this language going to cost us?" Not just in salaries, but in velocity, in technical debt, in hiring difficulty, in operational complexity, in every dimension that actually determines whether you survive.

Reframe it from a technical debate to an economic one. And unlike identity, economics can be measured, compared, and decided without anyone's ego being threatened.

Choosing a programming language is the single most expensive economic decision your company will make. It will define your culture, constrain your budget, determine your hiring pipeline, set your operational costs, and ultimately dictate whether you can move fast enough to win your market.

We need a framework that makes the invisible costs visible. One that lets us have the economic conversation instead of the identity conversation. One that works whether you're choosing your first language or evaluating a migration.

Our industry has never really had that framework... **Until now.** ↗

Up Next

The next post introduces **The 9 Factors of a Language's True Cost ↗**. A comprehensive framework for evaluating language decisions based on economics, not identity.

You'll learn how to quantify the hidden costs, predict long term impact, and make defensible decisions that your team can align behind, regardless of their language preferences.

You may also like

See All Programming-Languages

11 Nov 2025

A comprehensive framework for evaluating the total, long-term economic impact of programming language choices across nine critical cost dimensions.

12 Sep 2025

Benjamin Franklin discovered gradient descent 250 years before we had the mathematics to describe it.

28 Apr 2011

If you don't know Rands (real name Michael Lopp), you should. His blog is full of excellent ...

Steve Francia

For 3 decades, I've helped transform breakthrough ideas into trusted platforms now used by more than 50% of the world's developers. I've scaled this impact through leadership at Google, MongoDB, Docker, and Two Sigma as Managing Director, VP, and Product Leader shaping the tools that power modern software development.

My approach is simple: listen deeply, build elegantly, and maintain a healthy disregard for the impossible.

This philosophy guided me through architecting MongoDB's user experience that sparked mass adoption, scaling Go from 400K to 4.5 million developers at Google, pioneering container standards at Docker, and directing AI platform innovation at Two Sigma.

While technical problems are exciting to solve, I've learned that it's the human stack: the teams, cultures, and communities we build that truly determines lasting impact. Here, I share lessons on building products and teams that not only work, but endure and inspire.