Tip 5

Invest in Your Intelligence

When choosing what to focus on, it can be tempting to simply look at the technologies that yield the most jobs and focus on those. Java is big. .NET is big. Learning Java has a simple, transitive effect: if I know Java, I can apply for, and possibly get, a job writing Java code.

Using this logic, it would be foolish to choose to invest in a niche technology, especially if you had no plans to try to exploit that niche.

TIOBE Software uses Internet search engines to indicate the relative popularity of programming languages, based on people talking about those languages on the Internet.[3] According to TIOBE's website, "The ratings are based on the worldwide availability of skilled engineers, courses, and third-party vendors." It's definitely not a scientifically provable measure of popularity, but it's a pretty good indicator.

At the time of writing, the most popular language is Java, followed by C. C# is in a respectable sixth place but with a slight upward trajectory. SAP's ABAP is in seventeenth place and is moving slowly downward. Ruby, which is my personal favorite programming language—the one I do pretty much all of my *serious* work in and the one for which I co-organize an international conference every year—is in eleventh place. But at the time the first edition of this book was published, it wasn't even in the top twenty. It was below ABAP!

Was I crazy to use Ruby or just stupid? I must be one of the two, right?

In his essay "Great Hackers,"[4] Paul Graham annoyed the industry with the assertion that Java programmers aren't as smart as Python programmers. He made a lot of stupid Java programmers mad (did I say that?), causing a lot of them to write counterarguments on their websites. The violent reaction indicates that he touched a nerve. I was in the audience when his essay was first presented, in the form of a speech. For me, it sparked a flashback.

3. http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html
4. http://paulgraham.com/gh.html

Prepared exclusively for Casimir Creative

I was on a recruiting trip in India weeding through hundreds of candidates for only tens of jobs, and the interview team was exhausting itself and running out of time because of a poor interview-to-hire hit rate. Heads hurting and eyes red, we held a late-night meeting to discuss a strategic change in the way we would go through the candidates. We had to either optimize the process so we could interview more people or somehow interview *better* people (or both). With what little was left of my voice after twelve straight hours of trying to drag answers out of dumbstruck programmers, I argued for adding Smalltalk to the list of keywords our headhunters were using to search their résumé database. "But, nobody knows Smalltalk in India," cried the human resources director. That was my point. Nobody knew it, and programming in Smalltalk was a fundamentally different experience than programming in Java. The varying experience would give candidates a different level of expectations, and the dynamic nature of the Smalltalk environment would reshape the way a Java programmer would approach a problem. My hope was that these factors would encourage a level of technical maturity that I hadn't been seeing from the candidates I'd met so far.

The addition of Smalltalk to the requirements list yielded a candidate pool that was tiny in contrast to our previous list. These people were diamonds in the rough. They really understood object-oriented programming. They were aware that Java isn't the idealistic panacea it's sometimes made out to be. Many of them *loved* to program! *Where have you been for the past two weeks?* we thought.

Unfortunately, our ability to attract these developers for the salaries we were able to pay was limited. They were calling the shots, and most of them chose to stay where they were or to keep looking for a new job. Though we failed to recruit many of them, we learned a valuable recruiting lesson: we were more likely to extend offers to candidates with diverse (and even unorthodox) experience than to those whose experiences were homogenous. My explanation is that either good people seek out diversity, because they love to learn new things, *or* being forced into alien experiences and environments created more mature, well-rounded software developers. I suspect it's a little of both, but regardless of *why* it works, we learned that it works. I still use this technique when looking for developers.

So, other than trying to show up on *my* radar screen when I'm looking to hire someone, why else would you want to invest in fringe

Prepared exclusively for Casimir Creative

technologies that you may rarely or never have an opportunity to actually get paid to use?

For me, as a hiring manager, the first reason is that it shows that you're interested. If I *know* you learned something for the sake of self-development and (better) pure fun, I know you are excited and motivated about your profession. It drives me crazy to ask people whether they've seen or used certain not-quite-mainstream technologies only to hear, "I haven't been given the opportunity to work on that" in return. *Given* the opportunity? Neither was I! I *took* the opportunity to learn.

> *I haven't been given the opportunity…?*
>
> Seize the opportunity!

More important than portraying the perception of being suitably motivated and engaged by your field is that exposure to these fringe technologies and methodologies actually makes you deeper, better, smarter, and more creative.

If that's not good enough reason, then you're probably in the wrong profession.

Act on It!

1. Learn a new programming language. But, don't go from Java to C# or from C to C++. Learn a new language that makes you think in a new way. If you're a Java or C# programmer, try learning a language like Smalltalk or Ruby that doesn't employ strong, static typing. Or, if you've been doing object-oriented programming for a long time, try a functional language like Haskell or Scheme. You don't *have* to become an expert. Work through enough code that you truly feel the difference in the new programming environment. If it doesn't feel strange enough, either you've picked the wrong language or you're applying your old way of thinking to the new language. Go out of your way to learn the idioms of the new language. Ask old-timers to review your code and make suggestions that would make it more idiomatically correct.

Prepared exclusively for Casimir Creative