Java's Cover

paulgraham.com · Paul Graham · 2001-04 · source

|
April 2001

This essay developed out of conversations I've had with
several other programmers about why Java smelled suspicious. It's not
a critique of Java! It is a case study of hacker's radar.

Over time, hackers develop a nose for good (and bad) technology.
I thought it might be interesting to try and write down what
made Java seem suspect to me.

Some people who've read this think it's an interesting attempt to write about
something that hasn't been written about before. Others say I
will get in trouble for appearing to be writing about
things I don't understand. So, just in
case it does any good, let me clarify that I'm not writing here
about Java (which I have never used) but about hacker's radar
(which I have thought about a lot).

The aphorism "you can't tell a book by its cover" originated in
the times when books were sold in plain cardboard covers, to be
bound by each purchaser according to his own taste. In those days,
you couldn't tell a book by its cover. But publishing has advanced
since then: present-day publishers work hard to make the cover
something you can tell a book by.

I spend a lot of time in bookshops and I feel as if I have by now
learned to understand everything publishers mean to tell me about
a book, and perhaps a bit more. The time I haven't spent in
bookshops I've spent mostly in front of computers, and I feel as
if I've learned, to some degree, to judge technology by its cover

as well. It may be just luck, but I've saved myself from a few technologies that turned out to be real stinkers.

So far, Java seems like a stinker to me. I've never written a Java program, never more than glanced over reference books about it, but I have a hunch that it won't be a very successful language. I may turn out to be mistaken; making predictions about technology is a dangerous business. But for what it's worth, as a sort of time capsule, here's why I don't like the look of Java:

1. It has been so energetically hyped. Real standards don't have to be promoted. No one had to promote C, or Unix, or HTML. A real standard tends to be already established by the time most people hear about it. On the hacker radar screen, Perl is as big as Java, or bigger, just on the strength of its own merits.

2. It's aimed low. In the original Java white paper, Gosling explicitly says Java was designed not to be too difficult for programmers used to C. It was designed to be another C++: C plus a few ideas taken from more advanced languages. Like the creators of sitcoms or junk food or package tours, Java's designers were consciously designing a product for people not as smart as them. Historically, languages designed for other people to use have been bad: Cobol, PL/I, Pascal, Ada, C++. The good languages have been those that were designed for their own creators: C, Perl, Smalltalk, Lisp.

3. It has ulterior motives. Someone once said that the world would be a better place if people only wrote books because they had something to say, rather than because they wanted to write a book. Likewise, the reason we hear about Java all the time is not because it has something to say about programming languages. We hear about Java as part of a plan by Sun to undermine Microsoft.

4. No one loves it. C, Perl, Python, Smalltalk, and Lisp programmers love their languages. I've never heard anyone say that they loved Java.

5. People are forced to use it. A lot of the people I know using Java are using it because they feel they have to. Either it's

something they felt they had to do to get funded, or something they thought customers would want, or something they were told to do by management. These are smart people; if the technology was good, they'd have used it voluntarily.

6. It has too many cooks. The best programming languages have been developed by small groups. Java seems to be run by a committee. If it turns out to be a good language, it will be the first time in history that a committee has designed a good language.

7. It's bureaucratic. From what little I know about Java, there seem to be a lot of protocols for doing things. Really good languages aren't like that. They let you do what you want and get out of the way.

8. It's pseudo-hip. Sun now pretends that Java is a grassroots, open-source language effort like Perl or Python. This one just happens to be controlled by a giant company. So the language is likely to have the same drab clunkiness as anything else that comes out of a big company.

9. It's designed for large organizations. Large organizations have different aims from hackers. They want languages that are (believed to be) suitable for use by large teams of mediocre programmers--languages with features that, like the speed limiters in U-Haul trucks, prevent fools from doing too much damage. Hackers don't like a language that talks down to them. Hackers just want power. Historically, languages designed for large organizations (PL/I, Ada) have lost, while hacker languages (C, Perl) have won. The reason: today's teenage hacker is tomorrow's CTO.

10. The wrong people like it. The programmers I admire most are not, on the whole, captivated by Java. Who does like Java? Suits, who don't know one language from another, but know that they keep hearing about Java in the press; programmers at big companies, who are amazed to find that there is something even better than C++; and plug-and-chug undergrads, who are ready to like anything that might get them a job (will this be on the test?). These people's opinions change with every wind.

11. Its daddy is in a pinch. Sun's business model is being undermined on two fronts. Cheap Intel processors, of the same type used in desktop machines, are now more than fast enough for servers. And FreeBSD seems to be at least as good an OS for servers as Solaris. Sun's advertising implies that you need Sun servers for industrial strength applications. If this were true, Yahoo would be first in line to buy Suns; but when I worked there, the servers were all Intel boxes running FreeBSD. This bodes ill for Sun's future. If Sun runs into trouble, they could drag Java down with them.

12. The DoD likes it. The Defense Department is encouraging developers to use Java. This seems to me the most damning sign of all. The Defense Department does a fine (though expensive) job of defending the country, but they love plans and procedures and protocols. Their culture is the opposite of hacker culture; on questions of software they will tend to bet wrong. The last time the DoD really liked a programming language, it was Ada.

Bear in mind, this is not a critique of Java, but a critique of its cover. I don't know Java well enough to like it or dislike it. This is just an explanation of why I don't find that I'm eager to learn it.

It may seem cavalier to dismiss a language before you've even tried writing programs in it. But this is something all programmers have to do. There are too many technologies out there to learn them all. You have to learn to judge by outward signs which will be worth your time. I have likewise cavalierly dismissed Cobol, Ada, Visual Basic, the IBM AS400, VRML, ISO 9000, the SET protocol, VMS, Novell Netware, and CORBA, among others. They just smelled wrong.

It could be that in Java's case I'm mistaken. It could be that a language promoted by one big company to undermine another, designed by a committee for a "mainstream" audience, hyped to the skies, and beloved of the DoD, happens nonetheless to be a clean, beautiful, powerful language that I would love programming in. It could be, but it seems very unlikely.

|