

## LETTER TO THE EDITOR

Dear CrossTalk Editor,

The February issue of CrossTalk printed an article I wrote on choosing a software language. At the beginning of the article, I give some examples of how language choices are being made. In the original submission, one of the examples was the bookstore method, in which the number of books in the bookstore were counted and that was deemed to be the best language. However, this was edited out of the final article. Perhaps the editor didn't believe it, or maybe it was cut to save some space.

At the 2003 Software Technology Conference in April, one panel for a military program presented a brief on their language choice for their program. One of the slides they presented was a study of the number of books published on the various languages. To understand why this is a bad metric, I invite the reader to tour my home library. I want to show you books on three of the over 10 languages represented.

First, I have three books on Java. One is obsolete, because the language has changed since I bought it. The one I want you to look at closely is by Deitel and Deitel; it is a well written college textbook targeting the first software language crowd. These days when I hear Simon and Garfunkel singing about the words of the prophets being written on the subway walls, I think I see them scribbling the word Java. There will be many more additions to this language in the next couple of years, like the addition of generics. I think I will be buying more books on Java.

My older C++ books say that C++ is a superset of C and so there is already a large base of programmers using it. This is a blatant lie. It wasn't true then and it is not true now. One book is specific to Borland C++. Two books are specific to Visual C++, but they are obsolete as that platform has changed since I got them. My favorite is like three books in one because it covers three types of C++: switch (pick your version) {case version\_one: do it this way; break; case version\_two; do it that way; break; case version\_three; do it like so; break;}. Old C++ books are not useful because they don't have information on exceptions or namespaces or other additions to the language.

I have two stacks of magazines that are devoted to this language, and they are full of articles that tell me how to use it safely by not using some of the features, or how to be careful using other features. An interesting note about these magazines is that a lot of the code in them is actually Delphi, a language that was advertised as C++ without the problems. C++ claims to be a friend to object-oriented programming, but the friend function violates the paradigm; unless you believe the book that says it doesn't. These books claim that C++ is a strongly typed language, but I disagree. Besides my books and magazines, I have a notebook full of coding standards that tell how to make this language readable and safe to use, but they are full of contradictions.

I feel that these books and the C++ language have deceived me. My theory is that this language is still being used because of some weird Abilene paradox phenomenon. In the Abilene paradox, a group of people went to

town to eat and when they got back, they discovered that none of them really wanted to go, but they all went because they thought the others wanted to go. In more than one discussion with fellow engineers we all agree that C++ is a poor language, but we all use it because everyone else is using it. Engineers have to learn to read this language. For example, of about 20 books in my library on neural networks, 10 of them have example code in C++, even though Ada tasks would be more suited for building neuro nets. I don't plan to buy any more books on C++.

So, let's look at my Ada books. I have about 25 of them. Two of them I don't like. Five of them are now available on the Internet for free. There are three on the Internet that I don't have in hard copy, and I'm not counting them. Seven of them are Ada95, and the rest are Ada83 books. The old Ada books are still useful because of the backwards compatibility of the language and the fact that many of them are also about software engineering principles as well as the language. The one on distributed real-time systems is still a good book on distributed real-time systems, even though Ada95 fixed many of the issues that were raised.

These books compliment each other. Some authors are better at explaining access types, others at explaining tasking, and still others at explaining scope rules. These books are honest; they don't contradict each other. Unfortunately, our universities are slower to change and adapt to new technology than our militaries, so not many schools teach this language.

At the head of my linked list of want-to-have books is "Real-Time Systems and Programming Languages: Ada95, Real-Time Java and Real-Time Posix," by Alan Burns and Andrew J. Wellings. Should I file this with my Ada books or my Java books?

It is not the number of books that are published that count; it is their honesty, consistency, and the fact that they are endurable that counts. This is a much harder metric to measure. Last week in the bookstore, the number of books on C# and Java were each more numerous than the number of books on C++. Still, neither of these languages is as powerful or as stable as Ada. They are just more popular. It pleased me to see that there were three or four vendors at the STC from which I could buy Ada compilers and programming environments.

Dennis Ludwig  
Simulation and Analysis Facility  
Wright-Patterson Air Force Base

---

CrossTalk invites readers to submit their thoughts, comments, and ideas on its themes and articles as a "Letter to the Editor." Simply e-mail letters to <crosstalk.staff@hill.af.mil>.