Ada: Right problem, wrong answer?

To the editor:

The Soft News story about Ada catching on in the commercial market ("Ada Catches on in the Commercial Market," November, p. 81) is depressing. Will we never learn?

Grady Booch is correct: Complexity is the problem. But complexity is not overcome by adopting a new language or a new technology. It is overcome by exploiting the pattern in the problem. If you can find and exploit the pattern in a complex problem, you don't need Ada. If you can't find the pattern, Ada won't help.

My company has developed many large (200,000-750,000 lines of code) and complex information systems entirely in Cobol. We have developed reusable software in Cobol, too. We achieve productivity of 50 to 100 lines of code per person per day, we complete the projects on schedule, we satisfy our clients, and we make a profit.

The reusable software exploits the pattern that exists in large transaction-processing systems. To adopt an analogy from the construction industry, you cannot build an office building the same way you build a split-level house. You have to look for common service functions (bathrooms, elevators, etc.) and handle them uniformly on every floor. You may partition and decorate each floor uniquely, but the overall structure follows a uniform pattern.

My company develops systems the same way. We realize most of the productivity gains from using standard frameworks of reusable software. It directly provides about a third of the code in a typical 500,000-line system. We achieve further gains by using a standard control structure in the application (sort of a standard floor plan).

This approach is clearly consistent with the package philosophy inherent in Ada. However, you don't have to use Ada to achieve its benefits. Indeed, how will Ada prevent the wrong partitioning of a complex system? How will it assure the right one?

The solution to the design problem comes not from adopting a new language, but from adopting a systems engineering perspective in software design. If Ada is successful, it will be because it forces us to assign one or two good people the task of software design.

We have failed in the past precisely because we turned programmers loose to pursue functional objectives without an overall software architecture. If Ada's complexity prevents us from doing that in the future, we will have stumbled on the right answer for the wrong reasons. There is an engineering problem to be solved here, not a language problem.

Milton S. Hess
Vice President
Federal Systems Group
American Management Systems, Inc.
Arlington, Va.

The author responds:

Hess is, of course, correct in his view that there is much more to good software engineering than just the language in which the eventual code is written. Nevertheless, those with experience in Ada—at least the sample that I interviewed—are uniformly enthusiastic about the added value of the new language. The DoD is mandating that language for its contractors, but no one is forcing it on the commercial community. The marketplace will decide.

Ware Myers
Contributing Editor

Tool standards

To the editor:

Reading Robert Postons standards column ("Getting the Most out of New Tools," January, p. 65), I could not keep myself from nodding with approval at its many good points. That is, until I read this sentence: "You should not have to change the way you do things to use the tool."

That, I'm afraid, may be a real problem. I suspect that we are all too often inclined to disregard as irrelevant many things of value simply because using them would require that we change how we do things.

Instead, we spend our energy on flatterings things: ways and means that insiduously tell us, "Look what great skill you have." We are often like the 1910 horse-carriage driver more interested in buying a new whip than learning how to drive them new horseless carriages.

This said, I must state that I stand 100 percent behind you on the matter of standards, but with a twist: Virtually every company defines its own software standard. Suppliers are hard-pressed to provide tools that conform to such a zoo. Corporate management should be aware of the high cost of imposing industry-wide standards used only by a few programmers rather than industry-wide standards with many tool sources.

Yet they have a valid excuse: Industry-wide standards are still very limited in scope. Let's hope this excuse will quickly evaporate.

Michel Virard
President, Syslog, Inc.
Montreal

Fair review?

To the editor:

In the review of Menu Manager (New Product Reviews, January, p. 72) by Sorel Reisman, it is obvious that Reisman has never had to maintain a user environment for a large application. In this type of environment, the user is just interested in invoking an application and either doesn't know or doesn't care about how things are organized.

Having been involved with maintaining PC-based systems in an end-user environment with several hundred PCs, I believe that the concept of a product like Menu Manager is very attractive. I suggest that some editorial control be exercised to prevent a reviewer's personal prejudice from influencing a review as badly as Reisman's did in this case.

Richard H. Miller
Head, Systems Support
Baylor College of Medicine
Houston

A reviewer obviously has an opinion about the products reviewed. While there should be no conflict of interest (such as working for a competitor or for the company that created the product), there should be definite opinions. Readers are encouraged to draw their own conclusions from the facts and attitudes given by the reviewer. — Editor

IEEE Software welcomes your letters. Send them to Letters Editor, IEEE Software, 10662 Los Vaqueros Cir., Los Alamitos, CA 90720. All submissions are subject to editing for style, space, and clarity considerations.