Some Observations, Predictions, Prejudices, and Impressions about Programmers

Le plus ça change, le plus ça ressemble même chose. First it was Assembly versus Fortran, then Fortran versus Pascal; now it's Ada versus Fortran. The history of computing is strewn with the litter of the battlefield of linguistic sophistication. Mr. Drissel, in the January, 1982 "Open Channel" represents the quintessence of the war: old, friendly, low-level, and flexible versus new, high-level, rigid, and somehow clinically harsh and unfriendly. The war goes on at the lofiest theoretical level: "The notion that the benefits of Ada will repay the costs already incurred is preposterous," as well as in the trenches: "Ada preserves two manites which are perpetual snags but the working programmer: "I = 1" for replacement. This requires two upper case characters..." (Well, not on my terminal as I prepare this.) Drissel also writes: "In my experience, program maintenance costs are affected only by the number of people assigned to maintenance organizations. It is certainly true that no one is assigned to program maintenance.

then maintenance costs are zero, at least implying a linear relationship.

But the heart of the issue is not how many characters you have to type to set a variable to a new value (a linguistic issue, applicable to all projects individually). Nor is it an issue of how many support people are required to maintain an IBM 370 implementation of APSE (an environment issue, presumably distributed across many projects). And it is definitely not an issue of what a truly competent programmer needs in order to produce good quality software in a relaxed, supportive environment, because such people will produce good code if they have to translate it to ones and zeroes by hand. (I have known such people.)

No, instead, what we must address is just what it takes in both a language and a programming environment to make it easy for an average programmer to produce software which is "high quality" in the traditional measures of quality: efficient in time and space and easy to maintain. Perhaps Mr. Drissel's maintenance programmers will "still rewrite working code to speed it up or clean it up," but it seems to me that if a language really addresses the needs of its consumers (the users and the maintainers of programs written in the language who must use it whether they like it or not), then

(1) the code won't need to be cleaned up;
(2) the compiler will do almost as well as the programmer in execution speed; and
(3) the maintenance organization will have plenty enough work to do enhancing the program so that no one will be interested in doing the above.

It has been my experience that maintenance programmers rewrite working code only if they find it so painful to understand that they can't stand the thought of meddling with it. Most languages unfortunately make it easy to write such code. The excessive internal complexity of a program is both the major reason for slow execution and the major source of bugs. Controlling that complexity is a legitimate area for linguistic improvement.

Does anyone seriously suggest that existing programs are easy to modify for use in applications other than their original ones? Does Mr. Drissel really think that his associates—or any programmer, including yours truly—really would be wasting $10,000 on a tool that significantly improved the maintainability of the code written? If he does, I suggest that Mr. Drissel check the wind direction before he spends his budget.

Don't misunderstand. I am not at all sure that Ada solves anybody's problems, and it surely will cause headaches in changeover. But a programmer only has to learn Ada and the use of APSE once, whereas he currently has to struggle every time he is presented with a new program to maintain. No matter how obscure Ada may seem, if it really helps alleviate this struggle almost any amount of effort (and money) spent will, in the long term, be repaid.

I can readily understand the frustration evident in Mr. Drissel's letter: "The next generation of language theorists will declare Ada objectionable and obsolete before it is in place." But progress does go on. No end of trouble was created by switching from vacuum tubes to transistors and subsequently to integrated circuits, but does anyone really want to go back? After all, Ada at least has some solid rationale behind most of it, something lacking in most general-purpose languages to date.

In the long run, it will be the working programmer who decides the future of Ada and Fortran, just as the future of Basic was decided over other contemporary variations like Cal. If it works, it will stick around; if it doesn't, it will go away, a concept this country was founded on. For my money, the possible gains are worth the risk of the investment, and, as Mr. Drissel so aptly observed, "Ada is coming anyway."

Martin Morris
GTE Automatic Electric Labs
Phoenix, Arizona

Erratum

Some time ago (never mind which issue) we misquoted the late W. C. Fields concerning the charming city of Philadelphia. To set the record straight:

W. C. Fields wrote "On the whole, I'd rather be in Philadelphia" in answer to a 1929 Vanity Fair magazine questionnaire asking celebrities to compose their own epitaphs. But his gravestone in L. A.'s Forest Lawn—he died Christmas Day 1946—simply records his name and life dates.

—Herb Caen,
San Francisco Chronicle
April 3, 1981

Our apologies to the citizens of Philadelphia for any inconveniences, etc.

J. H.