We've Come a Long Way From Machine Code to Current Programming Languages

As I reviewed this month's CrossTalk theme articles on programming languages, I found it interesting to look back at the evolution of programming during the last 30 years. You can probably guess my age when I admit that the first programs I wrote in college were in machine code.

Lt. Col. Thomas M. Schorsch and Dr. David A. Cook begin this issue with their article *Evolutionary Trends of Programming Languages*. The authors discuss the typical generations of programming languages defining first, second, and third generations, then admit that there is probably no general agreement on what constitutes fourth, fifth, and future generations of languages. (Of course, I actually related to writing programs in a first-generation or machine-code language. One assignment I remember required us to actually write a fully functional program with a limitation of 100 bytes of storage.) Schorsch and Cook go on to describe several general evolutionary trends that have influenced programming languages, as well as some specific recent advances. After discussing many different languages and how they came about, the authors conclude that throughout this total evolution the basic role of a programming languages does not change. This role is to allow the developer to easily express abstract ideas in a language that a machine can execute.

Dennis Ludwig's article, *Language Considerations*, proposes some pertinent ideas for dealing with the question: What programming language should I use for my new project? He includes some real-world examples of how this decision has been made in the past. He then suggests some decision-making parameters that could be formalized into a decision table that could form the basis of a decision-making process.

In *SEPR and Programming Language Selection*, author Richard Riehle laments the misunderstanding and misuse of the 1996 memo from Assistant Secretary of Defense Emmett Paige. He contends that many readers mistakenly assumed the memo's intent was a license to abandon Ada rather than advice to include language selection as part of a rational evaluation step. He discusses some criteria used to evaluate the selection of a language for a particular purpose or project. Riehle contends that the strengths and weaknesses of the more popular languages should be well understood so that the decision whether to choose them or to reject them is based upon consideration of sufficient specific criteria reflecting the project's full life-cycle needs.

We are fortunate in this issue to also get an international perspective on the growth and challenges of the global software market. François Coallier, chairman of Sub-Committee 7 (ISO/IEC JTC 1/SC7), in his article *International Standardization in Software and Systems Engineering* provides an introduction to international standardization in information technology. This article provides status and describes the current activities in international software and systems engineering standardization. Coallier explains why all of these are important for professionals and organizations in the software arena.

Following this ISO tutorial is an application example from another international author, Dr. Paolo Donzelli of Italy. In *An Enterprise Modeling Framework for Complex Software Systems*, Dr. Donzelli describes a goal-oriented, agent-based Enterprise Modeling Framework where advanced requirements engineering techniques are combined with software quality modeling approaches. This provides an environment within which stakeholders and analysts can easily cooperate to discover, verify, and validate the requirements for a new software system.

Lastly, CrossTalk's Associate Publisher Elizabeth Starrett reports on ways to expand your people skills in *Highpoints From the Amplifying Your Effectiveness Conference*. This conference uses out-of-the-box techniques to teach personal skills to complement technical skills for a total package that can help you improve your organization, project, or process.

I hope that this issue will prove useful in your efforts to understand the challenges associated with the myriad of programming languages available, and how to make some practical choices for your particular needs.

H. Bruce Allgood
Deputy Director, Computer Resources Support Improvement Program