# All I Want for Christmas Is ...

I am writing this column in September, but since you will be reading this during the holiday season; I feel I need write about what I want for Christmas!

First of all, I need a standardized language to program in (notice "need" not "want"—there is a big difference). I have been programming since the late 1960s. Remember the first attempt to unify the myriad of programming languages, PL/I? Back in the 1950s and 1960s, there were two distinct classes of programmers—business and scientific. The scientific programmers had started by using assembly, but most transitioned to Fortran. The business programmers, on the other hand, were almost universally moving from assembly to COBOL. IBM, in bringing out its OS/360 architecture wanted to have a new, unified language that would offer a single programming language for all users. In 1966, the same year that OS/360 was released, the first PL/I compiler was also released. While the language is still used today, it is certainly a niche language for lots of reasons. The language contained elements of both Fortran and COBOL. The Fortran programmers noticed the COBOL features, and considered it a business language. The COBOL programmers noticed the Fortran features, and declared it unsuitable for business programming. The language contained lots of seldom-used features, making the overall language very large. And, in the beginning, it was not known for producing highly optimized object code. All of these issues (and many more) prevented PL/I from ever becoming a unifying language.

Over the years, I have certainly seen other language unification attempts. I was (and still am) an Ada proponent. It was initially offered as a real time and embedded system language, but the current version of the language is object-oriented and general purpose. It has features for both the business and scientific camps. I still teach and use Ada, and still feel that for high-precision or safety-critical systems, it is the best language we have. Alas, for many reasons (some technical, some political) it is now a niche language, also.

We have lots of languages to choose from now—Java, C++, Ruby, Python, Perl, etc. Some are good for large-scale systems, some for scripting; some are more suited for hacking. None have really unified the programming community. I am also quick to point out that a language is just a language—design and requirements doom large software projects much faster than poor language selection. But still, why do I have to go through the same arguments and discussion of what language should be used for every project I consult on?

So, if you can not give me a single programming language, well ...

Secondly, I want a standardized operating system. I "grew up" on UNIX, with occasional journeys on Multics and CTSS, and some GECOS. I also spent some time with CP/M, MS/DOS, VMS, Commodore OS, transitioned to Windows 2.0, 3.0, and beyond. In addition, I have moved through the Mac OS X zoo (Cheetah, Puma, Jaguar, Panther, Tiger, Leopard, Snow Leopard, Lion, and now Mountain Lion). And let us not forget the many, many flavors of UNIX/Linux (Red Hat, SuSe, FreeBSD, etc).

Each of the major operating systems in use today has some really cool features. And there is certainly no serious or significant movement to merge the operating systems, so I will still have to pick and choose which OS to run depending upon what my OS needs are.

But what about the specific needs of the DoD? Oh yeah—we totally forgot about those who need a Real Time Operating System (RTOS). In which case, none of the above are sufficient, and you have to choose from LynxOS, OSE, QNX, RTLinux, VxWorks, Windows CE, etc.

If I can not have a standardized programming language or a standardized operating system, then ...

The third item I would like for Christmas would be a single design methodology. I have been through flowcharts, Program Design Language, Structured System Design, Systems Analysis and Systems Design, Hierarchical Input Process Output charts, Data Flow Diagrams, Control Flow Diagrams—just to name a few. Rather than elaborate with more acronym soup, let us just shorten this paragraph. I have the CMMI®. And, of course, I have various agile methodologies to use, too. And I have UML. One is a methodology or a touch-stone for measuring my maturity, one is a type of methodology, and one is a design language.

While I find UML a wonderful tool for some aspects of design, it is not the notational tool for multiple languages that I had hoped for years ago. And, as for the CMMI and Agile methodologies—let us face it—the much maligned waterfall model is STILL used as the basis for a huge amount of the software development throughout the DoD, the U.S., and the world.

And yet we survive. We somehow manage to get high-quality and mission-critical software delivered to the people who need it—sometimes on time, sometimes within budget, and sometimes with high quality.

It is the end of 2012. I do not have a standard language. I cannot standardize the operating systems. And my design modeling language cannot yet get me all the way from initial design to full code. And the mission-critical code still needs to be delivered on time, within budget.

Should make 2013 an interesting year.

**David A. Cook, Ph.D.**
**Stephen F. Austin State University**
**cookda@sfasu.edu**