Why Programming Languages Matter

Lt. Col. John A. Hamilton Jr.
U.S. Military Academy

The Department of Defense (DoD) policy that requires the exclusive use of Ada in DoD weapons systems has been rescinded. However, because of its robustness and valuable strengths, such as strong typing, ease of learning, and object-oriented design, Ada remains the language of choice for systems where human life is at stake.

The Ada initiative was undertaken to meet the special software needs of the DoD. Defense software systems have high assurance and high reliability requirements and are characterized by long procurement times and long lifecycles. None of the recent policy changes have changed DoD software characteristics.

Ada has unquestionably been a technical success. The successful revision, Ada 95, is an International Organization for Standardization standard distributed, object-oriented programming language designed for use in high assurance applications.

A major reason for the development of Ada was the proliferation of high-order programming languages in use by the DoD in the mid-1970s. It is commonly estimated that more than 450 different high-order languages were in use in 1975. As shown in Figure 1, the number of high-order languages in use in the DoD has significantly declined.

More important, more than 90 percent of both weapons systems and automated information systems (AISs) are implemented in one of seven languages. More than a third of our weapons systems are implemented in Ada as shown in Figure 2. The existence of at least 50 million lines of Ada code in operational war-fighting software has been verified.

Twenty-two percent of all DoD AISs are coded in Ada as shown in Figure 3. It should be noted that almost 60 percent of all DoD AISs are still coded in COBOL. This is significant because this illustrates the long lifecycles of DoD software.

Ada is the most widely used programming language in the weapons systems world and second only to COBOL in the AIS world. Given the long lifecycles of DoD systems, it is clear that systems coded in Ada will play a key war-fighting role well into the next century. For this reason, a lot of the debates on Ada are moot. Regardless of what happens in the short term, we will go to war with weapons systems written in Ada well into the next century.

Despite the successful use of Ada in many major projects, the use of Ada has remained somewhat controversial. The technical merits of Ada are not in serious doubt, rather, most concerns and complaints and general nay saying has come from managers, not engineers.

Nonetheless, the practice of software engineering has matured since Ada was first fielded and in that context, retired Lt. Gen. Emmett Paige Jr., assistant secretary of defense for command, control, computers, and intelligence (ASD/C3I), directed a study by the Computer Science and Telecommunications Board of the National Research Council (NRC) entitled *Ada and Beyond, Software Policies for the Department of Defense*.

The NRC Report

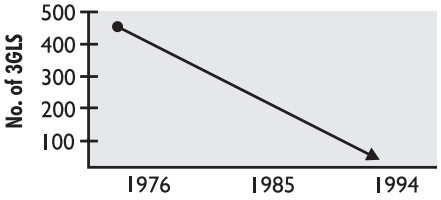
If you have not read the NCR report, you should do so. You can link to an online version via the AdaIC Web page, http://www.sw-eng.falls-church.va.us. The NRC study group was chaired by Barry Boehm who is the internationally recognized expert on software metrics. The solid explanation of Ada's reliability and suitability for warfighting software must not be ignored.

It is unclear what the final impact of the NRC report will be. The study findings may be summarized as follows:

- Ada gives DoD a competitive advantage in warfighting software applications.
- The current Ada requirement is overly broad in application.
- The current Ada requirement overemphasizes programming language considerations.
- Ada requirement and waiver process is unevenly implemented.
- For Ada to remain the strongest programming language for war-fighting software, DoD must provide technology and infrastructure support.
- Incomplete software metrics data weakens software decision making.

For high-assurance software, Ada is still often the best solution. In this, the NRC independently validated what the

Figure 1. *Trend in DoD high-order language use 1975-1994.*



Ada community has been saying for years. However, software engineering is not a one-size-fits-all proposition. Based on these findings, the NRC made the following recommendations:

- Require Ada for DoD war-fighting software.
- Drop Ada requirement for other DoD software.
- Invest $15 million yearly for Ada infrastructure - or drop Ada requirement.
- Program language selection should be part of a rational software engineering process.

The ASD(C3I) decided to accept all of the NRC recommendations save one, the mandate for war-fighting software. It would be extremely difficult to precisely define what constitutes war-fighing software and what does not. It would be interesting to see which program managers chose to classify their programs as "non-war fighting."

However, the best summary of the NRC report comes from then Lt. Gen. Paige who wrote,

> The study is a good one and I am prepared to accept and implement all of their recommendations with one exception. I believe DoD should no longer require Ada for any of its systems but continue to support it as the preferred language, particularly for our weapon systems and C4ISR systems. By doing this we will take the lone contentious point of resentment out of the DoD software process, but with the other recommendations such as the requirement for a software engineering plan, I believe we will get the desired results without mandating any particular programming language.

I respectfully applaud Paige's bold decision to end the Ada mandate and implement the remaining NRC recommendations. Paige's April 29, 1997 directive is available on the Ada Information Clearinghouse Web site. Ada should be used where it makes engineering sense. In military applications, reliability makes engineering sense. For military systems with 20-year plus lifecycles, a maintainable language makes engineering sense. Ada will win where engineering factors are part of the decision process.

Reliability Counts

Make no mistake, war is about killing people and breaking things. Military weapons systems are designed to be lethal and must be reliably controlled. Unreliable military software is frightening. We often hear of Ada's strong typing. An implicit-type conversion that results in a one-degree rounding error will, at a range of 40 kilometers, put ordinance 700 meters off target. In a close combat situation, a 700-meter error can result in friendly casualties. Reliability is important. That this software was procured using "best commercial practices" and was determined to be "good enough" for military use is likely to be of small comfort to a gold star mother. Military software is a life or death proposition. People who do not understand this should get out of the business.

An excellent series of articles entitled "The Debugging Scandal," appeared in the April, 1997 issue of *Communications of the ACM* [2]. In his critique of one of the articles, "My Hairiest Bug War Stories," Dr. John McCormick makes the following observations:

> In [the] data, the use of Ada would have identified 35 percent of ALL the faults reported in the usenet bug reports. Ada would have identified 88 percent of those that were the "direct responsibility" of the programmer (e.g. array bounds violations, scalar range violations, uninitialized variables, etc.). A 35 percent reduction is a significant improvement and worth serious consideration when selecting a programming language. The 88 percent reduction is probably an appropriate measure of Ada's impact on classroom assignments, which are done in a more controlled environment than real-world software.

Programming languages matter. Compilers that do strong checking produce fewer run-time errors. There are entire classes of errors that some programming languages will not compile.

Ada in Education

Contrary to popular mythology, Ada is a remarkably easy language to learn. In the spring of 1996, a controlled experiment conducted at the U.S. Military Academy. An introductory course was conducted with two instructors each teaching two sections using Ada and two sections using Pascal. Recall that Pascal was a programming language specifically designed for instructional purposes. The experiment demonstrated that students can go further and faster in Ada than in Pascal. This was a particularly significant finding given the controlled environment and rigorous standards of instruction at the Academy.

This is an important consideration for the education and training communities since it is generally agreed that Pascal is coming to the end of its usefulness. The unsuitability of C and C++ for educational purposes is well docu-

Figure 2. *Breakout of programming language employment in DoD weapons systems.*

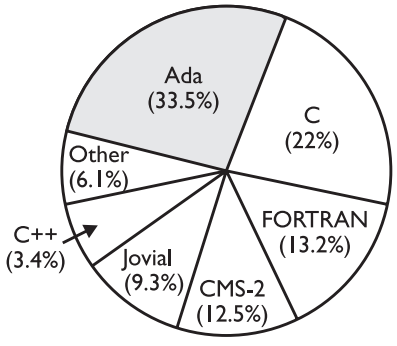
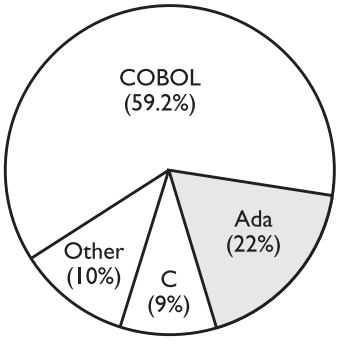


Figure 3. *Breakout of programming language employment in DoD automated information systems.*



mented. The confusing syntax is detailed in [3] and the lack of standardization of C++ is discussed in [1].

Ada's suitability for education should be of particular interest to those who complain that there are "no Ada programmers." One wonders what became of the programmers who wrote those 50 million lines of Ada currently in the inventory. It is well known that there is currently a serious shortage of skilled software personnel. Any hard skill computer scientist can learn Ada quickly. It is well established that Ada enforces good software engineering practices. This is not only important in education and training but also important for operational development. The choice of a programming language matters.

The Future of Ada

Dropping the mandate does *not* mean dropping Ada from defense programs. Inside the beltway, there is no stopping a catchy analogy. Ada is often compared to Betamax videocassette recorders (VCRs), the analogy being that there is no question of the technical superiority of the Beta format over VHS, yet Beta format VCRs failed to capture the consumer market. Like many analogies, this one does not go far enough. In fact, Beta format VCRs continue to be used by video professionals specifically because of the technical superiority of the Beta format. Just because Blockbuster doesn't stock Beta format videotapes doesn't impact on the engineering decisions made by video professionals.

The technical merit of Ada is not an issue. Ada is often attacked with a so-called "business" argument. This business argument is disturbing because military software and commercial software are simply not the same. Consider a program to compute field artillery gunnery solutions. One could certainly write such a program using a commercial spreadsheet. Unfortunately, even mature spreadsheets have undocumented numeric errors that arise from time to time. Although this can (and has) seriously affected a business, it is unacceptable to have this occur when firing live rounds.

It almost certainly does not make business sense to use all the extra safety features provided by an Ada compiler for a commercial product that will be on the shelf less than 18 months. The business case that applies to military systems is the lifecycle argument. Although sometimes difficult to rigorously quantify, the software maintenance advantages of Ada are not seriously disputed. With software maintenance costs running 70 percent to 90 percent of software lifecycle costs, it is unclear how Ada can be said to lose the "business case." Programming languages matter when considering software maintenance.

Military software is not the same as consumer software. You cannot buy "off-the-shelf" fire control systems. Ada was designed to build software to military specifications. Where reliability counts and where software engineering considerations factor into project management, Ada will continue to thrive. We will not win wars through superior word processing.

Conclusion

Language selection should be part of a rational software engineering process. This process cannot merely manage away the very real technical issues any nontrivial software project faces. Ada cannot win the technical battle in an environment where technical issues are often poorly understood. Programming languages matter and can make a difference in reliability, software design, maintenance, and education. The Department of Defense has a vital interest in standard, reliable software. As noted by the National Research Council [4], that requirement is often best met through the use of Ada. ◆

About the Author

Lt. Col. John A. (Drew) Hamilton Jr., U.S. Army, is currently assigned to the U.S. Military Academy Department of Electrical Engineering and Computer Science. He most recently served on special assignment as chief of the Ada Joint Program Office in the Defense Systems Information Agency. Previously, he served as chief of the Officer Training Division at the Computer Science School at Fort Gordon, Ga. He has a bachelor's degree in journalism from Texas Tech University, where he was commissioned in Field Artillery, a master's degree in systems management from the University of Southern California, and an master's degree in computer science from Vanderbilt University. He also holds a doctorate in computer science at Texas A&M University. He cowrote *Distributed Simulation*, published by CRC Press this year, with Maj. D. A. Nash and Udo W. Pooch. Hamilton is a distinguished graduate of the Naval War College.

Electrical Engineering & Computer Science
U.S. Military Academy
West Point, NY 10996
Email: dj7560@eecs1.eecs.usma.edu

References

1. Ben-Ari, M. and K. Henney, "A Critique of the Advanced Placement C++ Subset," *Special Interest Group on Computer Science Education Bulletin*, Vol. 29, No. 2, September 1991, pp. 7-10.
2. Eisenstadt, M., "My Hairiest Bug War Stories," *Communications of the ACM*, Vol. 40, No. 4, April 1997, pp. 30-37.
3. Mody, R.P., "C in Education and Software Engineering," *Special Interest Group on Computer Science Education Bulletin*, Vol. 23, No. 3, September 1991, pp. 45-56.
4. Computer Science and Telecommunications Board, National Research Council, *Ada and Beyond, Software Policies for the Department of Defense*, National Academy Press, Washington, D.C., 1997.