There's Still Some Life Left in Ada

Alexander Wolfe, Science Writer

Ada remains the Rodney Dangerfield of computer programming languages, getting little respect despite a solid technical rationale for its existence. Originally pressed into service by the U.S. Department of Defense in the late 1970s, these days Ada is just considered a remnant of bloated military engineering practices.

In reality, Ada aids software reliability, though that benefit admittedly comes at the expense of some possibly unpleasant software-engineering work in the form of added design time and documentation overhead. Nevertheless, in a world where software developers are increasingly forced to deal with time-to-market pressures, which lead to bugs, you can't get away from the fact that Ada enforces compile-time consistency checks. Indeed, the range and overflow checks applied by Ada compilers typically help catch more bugs earlier in the software development life cycle, where they are easier and less costly to fix and certainly less threatening.

To Ada's advocates, the net result is that, all things being equal, Ada projects, though requiring a bit more effort, result in a higher quality of code than those written in other languages. Perhaps more important, Ada programs are maintainable by developers other than the ones who created them. This is because they're amply commented.

Ada supporters would claim that such quality is evident in many of the real-world software efforts in which Ada has seen service. In the world of high finance, Ada solutions have been applied at the Swiss Postbank electronic funds transfer system and at Banque Nationale de Paris.

Realistically, however, most Ada projects of note have been developed in the defense sector, where the perceived added costs of conforming to the better software design practices enforced by Ada can be more readily absorbed. For example, Ball Aerospace applied Ada during the development of code for the so-called "black program" portion of a classified satellite built for the U.S. Department of Defense. And Sikorsky Aircraft tapped Ada to program the control display unit for the Black Hawk helicopter.

Perhaps the best-known Ada project isn't even military, though it's in the allied field of aerospace. That

When it comes to

SURVIVAL OF THE FITTEST,

ADA AIN'T NO DINOSAUR

effort is AIMS (airplane information management system), used in the Boeing 777. The main contractor was Honeywell, which

used the Ada compiler supplied by Phoenix, Arizona-based DDC-I Inc.

The most exciting new Ada project is Boeing's upcoming 7E7 Dreamliner. Wind River and AdaCore software will be used to develop the 7E7's Common Core System, which is the electronics and software backbone of the airplane's computers, networks, and interfacing electronics. The 7E7 Common Core System consists of 80 to 100 applications running simultaneously. These applications will control many of the airplane's avionics and utilities functions.

Truthfully, however, the fond hopes of Ada advocates that the language would be widely deployed outside of such specialized projects haven't been realized.

Over the past decade, the twin roadblocks derailing the adoption of Ada have been the rise of Java and C++. Neither the complexity of C++ (a flaw Ada has often been charged with) nor the limitations of Java have kept those two languages from gaining industry mindshare and support.

And there's more bad news. According to a 2003 report out of Carnegie Mellon University's Software Engineering Institute, "Ada is perceived as a dying language by most software engineers."¹

The report goes on to note the following: "Ada is used primarily within the DoD community. Beyond that community, it has been adopted by some software developers for the civilian market, especially where there is potential defense market cross-over or where there are similar requirements, such as in commercial aviation, process control, and medical instrumentation. However, this commercial use is a small fraction of the total commercial software market."

(On the plus side, market researcher Venture Development Corp. estimates that sales of Ada software tools—including integrated development environments, compilers, and debuggers—totaled \$47.7 million in 2001.

Sales are forecast to grow at an annual rate of 0.8 percent annually through 2006.² VDC estimated that some 8,000 to 10,000 developers are engaged in writing Ada firmware for embedded applications.)

Despite the seemingly bad news, that doesn't mean there's no place for Ada. Indeed, Ada in many ways has been the language that can't be killed off. It has survived through its Ada83 and Ada95 incarnations. Now, it may be about to take on a new role: Ada is being pushed as the answer to the software quality and security problems, which have been high on the agenda everywhere since 9/11.

Indeed, according to longtime Ada guru Tucker Taft, who serves as the president of SofCheck Inc.: "There is growing realization that something must be done to improve the quality and security of software, but the costs associated with exhaustive testing approaches are economically prohibitive for all but the most critical systems."³

If anybody has a shot at reviving Ada by repositioning it as the way to realize better quality software, it's Taft. He has a long history as an Ada guru, including a stint as chief scientist at Intermetrics in the 1980s and work on the development of the AIE (Ada Integrated Environment) for the Air Force. In the early 1990s, he led the Ada9X design team, which put together the specs for Ada95 and got it approved as the first ISO standardized object-oriented programming language.

**Those who believe Ada is
finished would do well
to remember that it's been
counted out before.**

Now, as founder of SofCheck, Taft is positioning his company's Inspector product as a high-reliability automated error detector, which relies on static analysis to catch software defects before code goes out the door, without the need for lengthy test plans that theoretically should be part of every software validation effort. (In practice, of course, many of us have worked in situations where such test plans were ignored or where testing was done by a seat-of-the-pants approach because of time constraints and intense pressure to ship the code.)

Taft recently argued his position in a paper he presented in June at the Ada Europe 2004 conference in

Palma de Mallorca, Spain. With perhaps some hyperbole, SofCheck argues that the cost of undetected bugs in shipping software products is \$60 billion annually and that automated analysis could greatly reduce this number.

Earlier this year, Taft described the utility of the static analysis process in his talk: "Static analysis has the potential to automate the process of line-by-line, full path inspection of source code for defects, while also providing component-by-component characterization of the software system in terms of inputs, outputs, and effects, thereby fostering better understanding of legacy software and enabling more informed software evolution and reuse strategies."

Taft's technology turns up defects by differential diagnosis—that is, by assessing the surrounding conditions under which bugs occur. Specifically, the Inspector tool scans source code and puts together a count of quantities such as stack depth, loop counts, dynamic storage allocation, lock frequency, and duration.

To date, SofCheck has worked with the likes of Raytheon, Northrop Grumman, and United Technologies. Whether its Inspector will gain adherents outside of Ada's traditional stomping grounds in the defense sector is anybody's guess.

Of course, any language is in trouble if its future success hinges on the adoption of a single product. But those who believe Ada is finished would do well to remember that it's been counted out before.

Moving forward, several reasons make it likely that, at least for the next several years, Ada will continue to thrive at least in a narrow niche. Most importantly, there's a small but entrenched group of Ada supporters in place in the embedded- and defense-software communities and in academia.

In the former, along with Taft's SofCheck, companies such as Aonix, Green Hills, I-Logix, AdaCore Technologies, RR Software, and others continue to create and market new Ada tools.

For example, in July, San Diego-based Aonix released the beta version of its IDE (integrated development environment) for the Ada language. Dubbed AonixADT, the IDE is based on the increasingly popular Eclipse open source toolkit framework. Initially, the IDE will run under Windows, but Solaris and Linux versions are on the way as well.

Some of the old-line Ada vendors are wisely hedging their bets by broadening their tools to handle multiple languages. For example, the new GNAT Pro 5.02a1 IDE from AdaCore Technologies has enhanced capabilities for multilanguage programming for Ada, C, and C++.

The IDE runs on a wide range of operating systems and can be fitted with numerous back ends for different target platforms.

Another successful tack would be to embrace Linux. That's exactly what I-Logix has done with its Rhapsody in Ada UML (Unified Modeling Language) object-oriented modeling tool. Earlier this year, the company released ports that run under both Red Hat and MontaVista Linux.

Perhaps the best hope for an Ada resurgence lies in academia. Michael Feldman, chairman of the ACM SIGAda Education Working Group and professor of computer science at George Washington University in Washington, D.C., has compiled a list of schools offering Ada as a "foundation language"—that is, in early programming courses for engineering and computer-science majors. His list turned up 117 colleges.⁴

Not surprisingly, the dedicated group of ACM members comprising SIGAda remain an important part of the Ada movement. They will meet for SIGAda 2004, November 14-18, at the DoubleTree Hotel in Atlanta. Further information can be found at: <http://www.acm.org/sigs/sigada/conf/sigada2004/>.

In a keynote that will provide more insight into the subjects we've touched on here, Pam Thompson, director of software engineering for Lockheed Martin Aerospace, will deliver a talk entitled, "Can Ada Stand Up to the Challenges of C/C++ and Java?" In her abstract, she notes that Lockheed Martin Aeronautics has several large software systems implemented using Ada, which will likely require maintenance for the next 10 years. Concerns include compiler viability, vendor interest in writing Ada applications, an aging Ada programming community, and software development environment upgrades that omit Ada compatibility requirements.

"Emerging technologies like model-driven architecture and object-oriented design replacing structured applications have the ability to change significantly the role of programming languages like Ada in the future," her abstract notes.

With such software imperatives ahead of us, perhaps Ada might find more than just one useful niche with developers. ☐

REFERENCES

1. Smith, J. What about Ada? The state of the technology in 2003. Technical Note-021 (July 2003), Carnegie Mellon University Software Engineering Institute; www.sei.cmu.edu/publications/documents/03.reports/03tn021.html.

AdaCore
Used with Boeing's upcoming 7E7 Dreamliners.
<http://www.adaic.org/news/act-7E7.html>

AonixADT
Beta version of Aonix's IDE for the Ada language, based on the Eclipse open-source toolkit framework.
http://www.aonix.com/pr_07.26.04.html

GNAT Pro 5.02a1
From AdaCore Technologies. Provides enhanced capabilities for multilanguage programming for Ada, C, and C++.
http://www.gnat.info/product_updates.php#GPS%202.0.0

Rhapsody in Ada
From I-Logix. A visual application development program for Ada programmers based on UML (Unified Modeling Language).
http://www.ilogix.com/newsroom/newsroom_detail.cfm?pressrelease=2002_02_19_055040_36053pr.cfm

2. Lanfear, C. The embedded software strategic market intelligence program 2001-2002: Ada in embedded systems. Venture Development Corp. (March 2002); <http://www.vdc-corp.com/embedded/white/02/02embeddedada.pdf>.
3. Taft, S. T. Fixing software before it breaks: Using static analysis to help solve the software quality quagmire. Ninth International Conference on Reliable Software Technologies Ada-Europe 2004; <http://dmi.uib.es/~AE2004/AE2004speakers.html#tuc>.
4. Feldman, M. Ada as a foundation programming language. SIGAda (Spring 2002); <http://www.seas.gwu.edu/~mfeldman/ada-foundation.html>.

LOVE IT, HATE IT? LET US KNOW

feedback@acmqueue.com or www.acmqueue.com/forums

ALEXANDER WOLFE received his electrical engineering degree from Cooper Union in New York City. A science writer based in Forest Hills, New York, he has contributed to *IEEE Spectrum*, *EE Times*, *Embedded Systems Programming*, and *Byte.com*.

© 2004 ACM 1542-7730/04/1000 \$5.00