I Have a Feeling We're Not in Emerald City Anymore

Author: *Henry G. Baker, http://home.pipeline.com/~hbaker1/home.html; hbaker1@pipeline.com*

Twenty years ago, the US DoD asked "What color is my parachute [programmed in]?" The answer was 'Green,' and the rest is history, or so we have been told. But there is more to this story—much, much more. By piecing together information from unclassified documents, we are now able—for the first time—to bring you the real story of the Ada Project, whose secrecy, scope and cost rival those of the Manhattan Project.

In the early 1970's, the US Department of Defense was facing a software crisis of staggering proportions. Software was becoming an increasingly important component of defense systems, and the percentage of software costs to hardware costs was rising precipitously. Something had to be done to bring these costs back into balance.

Our leaders understood well the issues:

> ...our warfighting strategy sustains and builds on...the application of information technology to gain great military leverage to continue to give us [an] unfair competitive advantage" [Perry96]

Yet, the US was facing a 'Software Gap' with the Soviets. According to [Davis78], the Soviets had "one of the most potent mathematical communities in the world":

> To an unusual extent, software productivity is a research and development activity, and thus it benefits from a relative Soviet strength. ... *Thus software would appear to have some relative advantages over hardware, even within the scope of Soviet R&D.* ...One of the reasons mathematics has done so well in the Soviet Union is that it is relatively insensitive to the constraints [of high quality materials]... *Software may have a similar advantage as long as it can operate within local hardware limitations.* [Davis78; emphasis supplied]

Although Soviet computer hardware was inferior, particularly in memory technology (typical Soviet computers had only a few hundred Kbytes of core and a few tens of Mbytes of disk), Soviet software ingenuity might make their hardware disadvantage moot:

> The availability of ES hardware has resulted in something of a ... software explosion... [Davis78]

Even more troubling were reports of incredible software productivity from US projects using languages like Lisp, APL, Prolog, Smalltalk and Forth. Entire robotic planning systems with natural language and graphics interfaces were programmed by just one or two people in Lisp. Reams of PL/I code were replaced by a single line of APL. Prolog obsoleted a generation of Cobol programmers. Smalltalk and Logo were taught to children, and raised the spectre of a 12-year-old outproducing a dozen beltway bandits. Perhaps most troubling of all was Forth, with its ability to very quickly program substantial real-time systems that took insignificant amounts of memory.

This software "productivity explosion" threatened the very axiom of American military might—that more is better, and a whole lot more is a whole lot better. Software productivity was expanding faster than software demand, and the trends predicted that by 1997 entire avionics systems would be programmed by a single hacker in sandals, love beads and a pony tail. It looked like a handful of Soviet super-programmers using these powerful new languages might wipe out any advantage from American hardware prowess.

The American military could not afford to ignore this Soviet software threat. Although the massive post-Sputnik spending to upgrade American education had produced some early success, SAT scores soon peaked and began a sickening decline. Joe Geek might not be able to compete mano-á-mano with Ivan Geek.

US defense contractors were also hopping mad. If the trend of the increasing percentage of software to hardware continued, and the productivity of software people exploded, total system spending would fall precipitously. In order to restore a proper balance, something had to be done to dramatically increase the cost of computer hardware.

After a number of top-secret meetings at the highest levels,[1] the "Ada Project" was conceived. The Ada

[1] Rumors persist that the famous gap in the Nixon tapes covered up

Project was to be a disinformation campaign of unprecedented scope and duration, involving all of the branches of the military,[2] the President, the Queen of England,[3] the Congress, and some academics. Its goal was to divert Soviet attention from truly productive computer languages like Lisp, and convince them that only a bloated, grossly inefficient, high order compiled language along the lines of PL/I could be reasonably utilized in the deployment of military embedded systems. The use of a standardized, inefficient language would provide a one-two punch: it would render super-programmers useless, and it would increase the demands on hardware by more than two orders of magnitude.

The Ada Project was inspired by the unexpected success of the IBM System/360 architecture behind the Iron Curtain. The Ada Project's wizards[4] reasoned that if the Soviets could be lured into copying the 360 architecture, they could also be lured into copying the Ada language, and if this language were fiendishly designed to make real-time systems essentially impossible to program, then the Soviet military machine would grind to a halt.

Although Ada would also severely impact American software productivity, it was felt that—just as cancer-fighting chemotherapy nearly kills healthy tissue while it kills tumors—the healthier US economy would be better able to bear the severe burden of an unproductive software industry than the Soviet economy could. Thus, while American geeks were inferior to Soviet geeks, our Elbonian hordes could beat their Mongolian hordes.

However, convincing crack Soviet programmers—e.g., Ershov—to believe in this disinformation campaign would be difficult:

> There was also a strong bias on the part of Soviet programmers who favored the 'efficiency' of machine or assembly language programming. *Clearly some of this bias rose from real considerations...* [Davis78; emphasis supplied]

The Ada Project's strategy for this problem was code-named 'Dogfood.' Just as dog food is purchased not by the dog himself, but by his owner, Soviet programmers have little control over the computers that they use. Project Ada thus set up an elaborate top-down bureaucracy in the finest central-planning tradition that would have a special appeal for the Communist Party apparatchiks, and the Soviet programmers would then be forced to go along.

Even knowledge of the Ada Project's name required the highest clearances and a need-to-know. The code name itself was an inside joke: Ada Augusta, Countess of Lovelace, was a famous armchair programmer/system architect who never in her entire life had gotten a single program to compile, link, or run.[5] Ada's code name was finally declassified—extensive research had shown that no one ever got the inside joke.[6]

The Ada Project was designed from the beginning as a international NATO project. Without the complicity of other countries, it would have had much less credibility with Ivan. Furthermore, the American military was not sure that American ingenuity could accomplish such a fiendishly difficult task without help from abroad. Prior to Ada, no one had ever attempted to design a computer language whose primary goal was dysfunctionality. However, the timely appearance of inscrutible documents from the European Algol-68 project provided hope and guidance.

The Ada Project's plan was absolutely brilliant. Because high order languages already in use by the various services and countries were known to work moderately well, an excuse had to be found to not use them. The cover story was that interservice rivalries could be minimized by not using these existing languages. Neither could existing "proprietary" languages like PL/I or Mesa be used; the cover story was that single companies would unreasonably benefit.

The C language was an exceptionally difficult case. Although the C language itself had lots of delicious ambiguities to exploit, it had some major problems. C was small and fast; it had small and fast compilers; and it had been utilized by two people to build an entire operating system that ran on small computers. This was precisely the sort of capability that would give Ivan an advantage, so C had to be buried.

> ...[The Ada wizards] took advantage of this connection between DARPA and Bell Labs to request their

such a discussion. "We got some agreement that [low] cost and [high] reliability was a problem, but the question was—what could be done... We go the problem passed to ... the Secretary of Defense, which is raising it to the "highest level" in the DoD..." [Whitaker93].

[2] DARPA jeopardized the cover story by providing only lukewarm support.

[3] [Whitaker93] discloses that the Queen's email name is '`EIIR`', but gives no domain name. The domain may be `royal.gov.uk` ("British Royalty Hold Court on the Internet," *L.A. Times*, March 8, 1997).

[4] The Ada Project was conceived at Kirtland AFB, NM, near Roswell.

[5] An analogous joke would be giving an Air Force plane the name "Kiwi," after a flightless bird.

[6] 'Project Ada' was not the first name suggested. 'Project Potemkin' was rejected when it was realized that Soviets might recognize this old Russian ruse.

> cooperation. When Bell Labs were invited to evaluate C against the DoD requirements, they said that there was no chance of C meeting the requirements of readability, safety, etc., for which we were striving, and that *it should not even be on the list of evaluated languages*. [Whitaker93; emphasis supplied]

Furthermore, combinations of languages had to be examined in order to make sure that no synergies precluded complete dysfunctionality.

> All candidate languages were evaluated by more than one contractor, and each contractor evaluated several languages, thereby providing a technical crosscheck on the individual evaluations. ...for each language requirement, the contractor was to determine the degree of compliance of each of the candidate languages, to comment on the feasibility of modifying the language to bring it into compliance, and *to identify features in excess of the requirements*. [Whitaker93; emphasis supplied]

In order to be successful as a disinformation campaign, the Project had to be 'public' in such a way that innocuous internal documents would be readily accessible to the Soviets to convince them of the authenticity of the project. The newly operational ARPANET fulfilled this requirement with the help of nodes conveniently located in neutral countries:

> The project was extraordinarily well documented... The requirements were circulated externally, *certainly far more so than has any other language effort, before or since*... The language comparisons and contract evaluations were published in excruciating detail and *are available to the public*... ...there were numerous Language Study Notes written and *made available to a large community over the ARPANET*... All issues submitted have been addressed, *and results were available on the ARPANET*... [Whitaker93; emphasis supplied]

Before putting the actual Ada language design out for bid, preliminary requirements documents were developed—STRAWMAN, TINMAN, LIONKING, etc.—all a bit dotty with just a little todo. However, in finest waterfall tradition, the final requirements were written *after* the winning language had already been defined. These STEELMAN requirements [STEELMAN78] were a masterful example of disinformation double-speak:

> *[The language] should emphasize program readability*. [Translation: It should be nearly impossible to write a program that will compile or execute.]
>
> *There shall be no language restrictions that are not enforceable by translators*. [Translation: No reasonable program will ever compile.]
>
> *[The language] shall attempt to avoid features whose semantics depend upon characteristics of the object machine*... [Translation: Issues critical to embedded systems like time and memory cost must be avoided.]
>
> *The language shall be completely and unambiguously defined*. [Translation: By using English as the definition, we can put off 'completely and unambiguously' defining until enough experience has been accumulated to make the decision that maximally reduces productivity.]
>
> *Every source program shall ... have a representation that uses only the following 55 character subset of ... ASCII*... [Translation: Ada must stay within the character set limitations of Soviet I/O devices.]
>
> *Separately translated units may be assembled into operational systems*. [Translation: Each system is to be shredded into small, separately programmed units on a need-to-know basis, so that no one person can comprehend it.]

Four bidders developed languages: Red, Green, Blue and Yellow. Although all four languages met the dysfunctionality requirements, the Green language—an export of Bull—was chosen after evaluators were told that they were voting on a new color for their uniforms. Green's dysfunctionality for embedded systems was outstanding:

- no interrupts, prioritized or not
- a synchronization primitive that nobody had ever heard of and was grossly expensive to implement
- no bit twiddling (no cyclic redundancy check or encryption codes)
- no ability to manage storage
- no guarantee of aborting a task
- no ability to manage scheduling
- no ability to interface to hardware

Ada initially planted a subconscious Y2K time-bomb: "YEAR: INTEGER **range** 0..2000;" [Ada79, p.3-12], but then got Third Reich ambitions: "YEAR: INTEGER **range** 0..4000;" [Ada83, p.3-34], before settling on a Y2.1K time-bomb: "**subtype** YEAR_NUMBER **is** INTEGER **range** 1901..2099;" [Ada83, p.9-11]. The prospect of every piece of embedded military software in the world simultaneously signalling a constraint error at the end of a century was too delicious for the wizards of Ada to pass up, and fixing these time-bombs would guarantee full employment for the defense contractors they would retire to.

The Ada Project had to publicly appear decisive in order to maintain its credibility: "...all other implementations of new high order programming languages for R&D programs were halted." [Whitaker93].[7]

The time to produce working, validated compilers exceeded all hope. Many years passed before the first validated compilers appeared, and another two years before they could be run at a customer site without crashing. To establish credibility with the Soviets, the U.S. Navy spent $40M developing a single Ada compiler. This at a time when they could have had four different compilers from four different competing commercial Ada vendors for a total of $10M, or even purchased all four independent Ada companies for less than $40M.

The dysfunctionality of Ada exceeded all expectations. A company called Rational was formed to build a programming environment for Ada, which itself was programmed in Ada. The Rational machine could never manage its own storage and had to be rebooted every few hours. Another Ada company's compiler was written in Ada, with multiple tasks for multiple compilations. Neither could it manage its own storage, and also had to be constantly rebooted. If applications with minimal real-time response requirements could not be programmed in Ada, what hope would Ivan have with real-time avionics systems?

The Ada compiler validation suite was cleverly designed to test only exceptional cases, not common cases. Its primary purpose was to detect undocumented enhancements, in case such an enhancement might prove useful and functional. Validation thus proved only that the compiler had no ambitions outside Ada; it said nothing about the ability to recognize and compile legal Ada programs.

The Ada validation suite also cleverly guaranteed that no parser-generating tools could be used for the Ada language by constraining the nature of syntax error messages, and by requiring that all such errors be found in a single compilation. The DIANA intermediate data structure was also a stroke of genius, because any compiler that used it used 10-100 times the memory of a C compiler running on the same host.

Because previous computer languages had evolved to become more productive in response to user feedback, a mechanism had to be developed to make sure that similar improvements could not happen to Ada. In order to keep Ada dysfunctional, very clever people ("Dystinguished Reviewers", or "Drs.") were appointed to dispatch user questions according to a formal protocol:

- Challenge the questioner's programmerhood. Tell him that no reasonable person would ever do this, and he should spend 3 more months in a straightjacket at the Software Engineering Institute.
- Bury him in legalities. Explain how the features of the language had been carefully designed to fit together in a certain way, and he couldn't begin to understand the wisdom of these decisions.
- Change the validation suite. Make sure that these kinds of programs won't even compile in the future.
- If worse comes to worst, "elucidate" the already "unambiguous" definition in a way that guarantees that no one will ever want to bring up a similar question again. The new interpretation will require all Ada systems to be revised in incompatible ways that also reduce performance by another 3X.

In order that the Ada Project's cover not be blown, students were kept away from actual Ada implementations. It would have been most embarrassing if a student were to compare Ada to Scheme, or even to C—it might lead to an "Emperor's New Clothes" situation. The teaching of Ada was restricted to courses like 'software engineering,' where only 'architecture' was argued and diagrams were drawn; programs were never allowed to actually be compiled or run.

'Software Reuse' was another goal of the Ada Project. To further reduce the cost of programming, programmers were encouraged not to throw defective code away, but to recycle it. At first, recycling bins of different colors were set up. Later, when things got GUI, these bins were replaced by 'trash icons' of different colors in the corner of the screen. Old versions and little bits of code that were cut but never pasted were forwarded to the SIMTEL

[7] In order to allow real systems to continue to be developed, the Ada Project secretly granted a large number of "waivers."

recycling center. The SIMTEL recycling center's motto was: "we utilize everything but the SQL."

We now know that the Ada Project was very successful. Ivan accepted the Ada wizards' humbuggering at face value. At the time of the Fall of Communism, a number of Soviet Ada projects were under way, and afterwards, at least one Soviet Ada compiler was offered for commercial sale over the Internet.

Now that the Wicked Witch of the East is dead, the wizards have finally allowed Ada to evolve into Ada9X, which fixed some of Ada's more egregious dysfunctions. However, even today the brilliance of Ada's original conception still shines brightly through.

That the Ada Project was able to keep its secret for 20 years is a tribute to the dedication and resourcefulness of the wizards of Ada. It wasn't easy being Green—those associated with the Ada Project withstood great criticism and still managed to keep a straight face. The Ada Project cost billions and billions in direct and indirect costs, but who can argue with the result? All of us owe a great debt of gratitude to those in the Ada Project who helped keep America free. We agree with Churchill: "Never in the field of human conflict was so much owed by so many to so few." To memorialize those who fought this valiant fight, we would like to dedicate The Cubicle of the Unknown Programmer.

Ada, we salute you!

If I Only Had Ada

(Sung to the tune "If I only had a Brain" from the movie "The Wizard of Oz.")

I could discriminate records,
deriving by the hoards, constraining the data.
Your packages, I'd be using,
while my names I'd be losing,
If I only had Ada.

I could loop away the days,
Suspending with delays, accepting every port.
And my coffee, I'd be perking,
While my tasks were busy working,
If I only had abort.

I could NEW to good effect,
without dealloc'd unchecked, sizing in ecstasy.
And my nerves would not be jangling,
when my pointers were left adangling,
If I only had GC.

I would meet every deadline, for each missile and each mine,
By land and by the sea.
And my storage would be pooling,
While my tasks were busy dueling,
If I only had GC.

References

[Ada79] "Preliminary Ada Reference Manual." *ACM Sigplan Not.* **14**, 6 (June 1979), Part A.

[Ada83] *Reference Manual for the Ada (R) Programming Language*. ANSI/MIL-STD-1815A-1983, 1983.

[Davis78] Davis, N.C. (US CIA), Goodman, S.E. "The Soviet Bloc's Unified System of Computers." *ACM Computing Surveys* **10**, 2 (June 1978), 93-122.

[Perry96] Perry, William. "Bueche Prize Acceptance Address." National Academy of Engineering, Wash., DC, 1996.

[STEELMAN78] *DoD Requirements for High Order Computer Programming Languages*. June 1978.

[Whitaker93] Whitaker, W.A., Col. USAF. "Ada—The Project: The DoD High Order Language Working Group." *ACM Sigplan Not.* **28**, 3 (March 1993), 299-331.

Henry Baker has been diddling bits for 35 years, with time off for good behavior at MIT and Symbolics. In his spare time, he collects garbage and tilts at windbags. *This column appeared in ACM Sigplan Notices 32,4 (Apr 1997), 22-26.*