San Diego Chapter of the Special Interest Group Ada

John A. (Drew) Hamilton, Jr., Ph.D.
Lieutenant Colonel, Field Artillery
United States Army
Director, Joint Forces Program Office

Code 05J
4301 Pacific Highway
San Diego, CA 92110-3127
(619) 524-7225 voice
(619) 524-7224 fax

SPAWAR

SPAWAR CinC Interoperability Program Office

www.drew-hamilton.com/sigada99

OUTLINE

- **Defense Department programming language objectives**
- **An unofficial view of warfighting software**
- **Future trends in programming languages**

The Here and Now

- We are unlikely to return to the programming language jungle of the 1970s without aggressive mismanagement on the part of DOD.
- Proprietary implementations of compilers can still cause major headaches later.
- “Later” is not on the scopes of many DOD IT organizations.
- Just because you **cannot** buy COTS for many military-specific applications does not mean that COTS is still not the “preferred solution.”

DOD Programming Language Objectives

- **Reliability**

  Software used to kill people and break things MUST be reliable.

- **Maintainability**

  Despite the rapid rate of change in commercial software applications, custom-coded, DOD warfighting software will continue to have long lifecycles.

  Increasingly, weapon upgrades are software based.

- **Reuse**

  Reuse at the function/procedure library level is the realistic target.

- **Portability**

  The DOD will remain a heterogeneous computing environment.

Reliability

- Research in the academic community demonstrates that programming language characteristics can significantly limit the nature and scope of errors.
- Significant risk avoidance can be achieved by using commercially available compilers for which there is an established track record for errors common to a particular language.
- ex. overwriting main memory addresses.
- Standardized compilers, those whose compliance to a standard has actually been validated by testing against a standard suite, pose the least risk in terms of reliability.

Maintainability

- Programming language characteristics, programming style and the problem domain all impact on the future maintainability of the code.
- Compilers which adhere to a standard and which prohibit subsets or supersets of the standard, reduce the learning curve for maintenance programmers.
- Language readability greatly enhances maintainability.
- Strongly typed languages are easier to trace and therefore to maintain.
- Published style guides facilitate understanding the source code by other than the original code developers.

Reuse

- Reuse at the function/procedure library level is the realistic target.
- Commonly used libraries decrease risk since errors may have surfaced in earlier employment of the library function.
- Libraries which only have publicly available APIs are riskier since the complete functionality of a library package may not be verifiable without a complete review of the source code.
- Major non-technical issues associated with reuse.

Portability

- DOD will remain a heterogeneous computing environment.
- Compilers that do not adhere to any recognized standard such as those of the IEEE or the ISO are often tied to a single hardware/OS platform.
- Additionally, commercial compilers often include powerful, but non-standard extensions.

  Early estimates of portability may be achieved by recompiling the source code using another vendor's compiler targeted for the same OS/hardware platform as the first compiler. This will largely identify the use of non-standard constructs.

  Portability between operating systems can be measured early by recompiling the source code with a compiler targeted for another operating system. This will identify the use of OS-specific system calls.

Warfighting Software: An Unofficial View

- **Clearly, this is the software that the Defense Community is most concerned with.**
- **Warfighting software is not COTS.**
- **In my view, any system that can affect battlefield performance is a warfighting system.**
- **It is more than just embedded systems. Information systems (such as the Advanced Field Artillery Tactical Data System) will interact between both embedded systems and warfighters.**

Non-warfighting software

- A BOQ reservation system that is unreliable is inconvenient, but not a war stopper.
- Military requirements for non-warfighting custom software should be minimal.
- Just because a system operates strictly in a CONUS garrison environment does not mean it is not a warfighting system.
- Example: a personnel mobilization system that can erroneously list a reservist as being hospitalized for minor surgery for three continuous years denies a warfighting asset to a theater of operations.

A viable Ada infrastructure is a military necessity

- **“Fifty million lines of Ada warfighting code will become a liability without a robust Ada infrastructure.”**
- **What happens when artillery fire control systems cannot be modified because the software is not maintainable?**
- **What happens when a tank’s targeting system has to be updated due to unexpectedly extreme climactic conditions?**
- **Inability to quickly and adequately maintain combat systems is a potential war stopper.**

Reliability Counts

- **A one degree error at a range of 40 kilometers equals a 700 meter lateral deviation.**
- **The precision engagement imperative of Joint Vision 2010 requires high reliability.**

Lifecycle considerations

- **Military software systems continue to have long lifecycles.**
- **Software maintenance is still the greatest software cost over the software lifecycle.**
- **Ada virtually always wins cost comparisons when maintenance is considered.**

Future Trends

- **Programming at higher levels of abstraction.**
  - **Reuse through common, standard libraries.**
- **Increase in distributed computing.**
- **A greater focus on software maintenance.**
- **Ada as a teaching language.**

Programming at higher levels of abstraction.

- **Language generations: (machine language through 4GLs.)**
- **Non-brittle CASE tools.**
- **Standardized reuse libraries.**
- **Emphasis moving from coding to design.**
  New paradigms, mixed paradigms.

Increased distributed computing support in programming languages

- **Distributed programming support built-in.**
- **Distributed programming constructs at higher levels of abstraction.**
- **Increased emphasis on open systems.**

  Systems that are both reactive and modular.
- **Increased use of middleware in the short term, use of programming languages which do not need middleware in the long term.**

Software Maintenance

- DOD cost estimates for maintenance over the software lifecycle range from 67% to more than 90%.
- Like automobiles, long term utilization increases the overall return on investment.
- Fewer new weapons starts means we will upgrade and modernize the systems we have fielded.
- We can verify the existence of fifty million lines of Ada code in critical warfighting systems.

Final Observations on Trends

- **Common commercial programming languages will evolve to meet military requirements.**
- **Software maintenance requirements will dictate the use of public standard languages.**
- **3GL-style programming languages will look more and more like Ada.**
- **CASE/4GLs will evolve to general-purpose usefulness, but this will take longer than people expect.**

Ada 95 Today

- Ada usage in the DOD impressive, the M1A2 tank, the Aegis system, the F-22 are Ada systems.
- Ada is alive and well in our warfighting systems.
- For the Defense Department this essentially means that the Ada debate is moot.
- Ada will be playing a key role in the Defense Department well into the 21st century.

Ada 95 vs. C++

- This is the wrong question on a variety of levels.
- First, the clear trend in programming languages is towards higher levels of abstraction.
- This trend really works against C and that is one reason why the use of C is declining.
- Higher levels of abstraction supported in C++ are notoriously non-standard. A very interesting illustration of this problem appears in the May, 1997 issue of CrossTalk

COTS is a Lie

- **COTS applications are often brittle, proprietary and incomplete.**
- **We cannot buy weapons systems off the shelf.**
- **Modifying commercial applications through the use of custom code is often the worst of both worlds.**
- **We will not win wars through superior word processing.**

Reliability is Important

- **Commercial software standards are NOT good enough.**
- **A 700 meter range error can easily kill US/Allied soldiers.**
- **Software that works 99% of the time built using “commercial best practices” will not impress a Gold Star Mother.**

Military Conclusions

- **We must not put our servicemen and women into harm's way without providing them the most reliable systems.**
- **We must protect the significant and successful Ada investments the DOD has already made.**
- **When we think war and warfighting software, reliability, maintainability and high assurance must be at the forefront. That is why Ada 95 is the right choice for warfighting software.**

Conclusions

- We are unlikely to see a return of a programming language mandate.
- Programming language selection should be an *engineering* decision.
- When engineering criteria such as reliability, maintainability, reuse and portability are considered, Ada 95 will often emerge as a clear winner.
- Do not underestimate the influence of trade journals and pundits on DoD policy.
- *Publicized* success of Ada in non-government related activities most likely to influence DoD decisionmakers.