Technical Report ITL-94-8
September 1994

US Army Corps
of Engineers
Waterways Experiment
Station

Transition to Ada

by William A. Ward, Jr.,
University of South Alabama

Approved For Public Release; Distribution Is Unlimited

Prepared for U.S. Army Environmental Center

Technical Report ITL-94-8
September 1994

Transition to Ada

by William A. Ward, Jr.

Faculty Court West 20
School of Computer and Information Sciences
University of South Alabama
Mobile, AL 36688

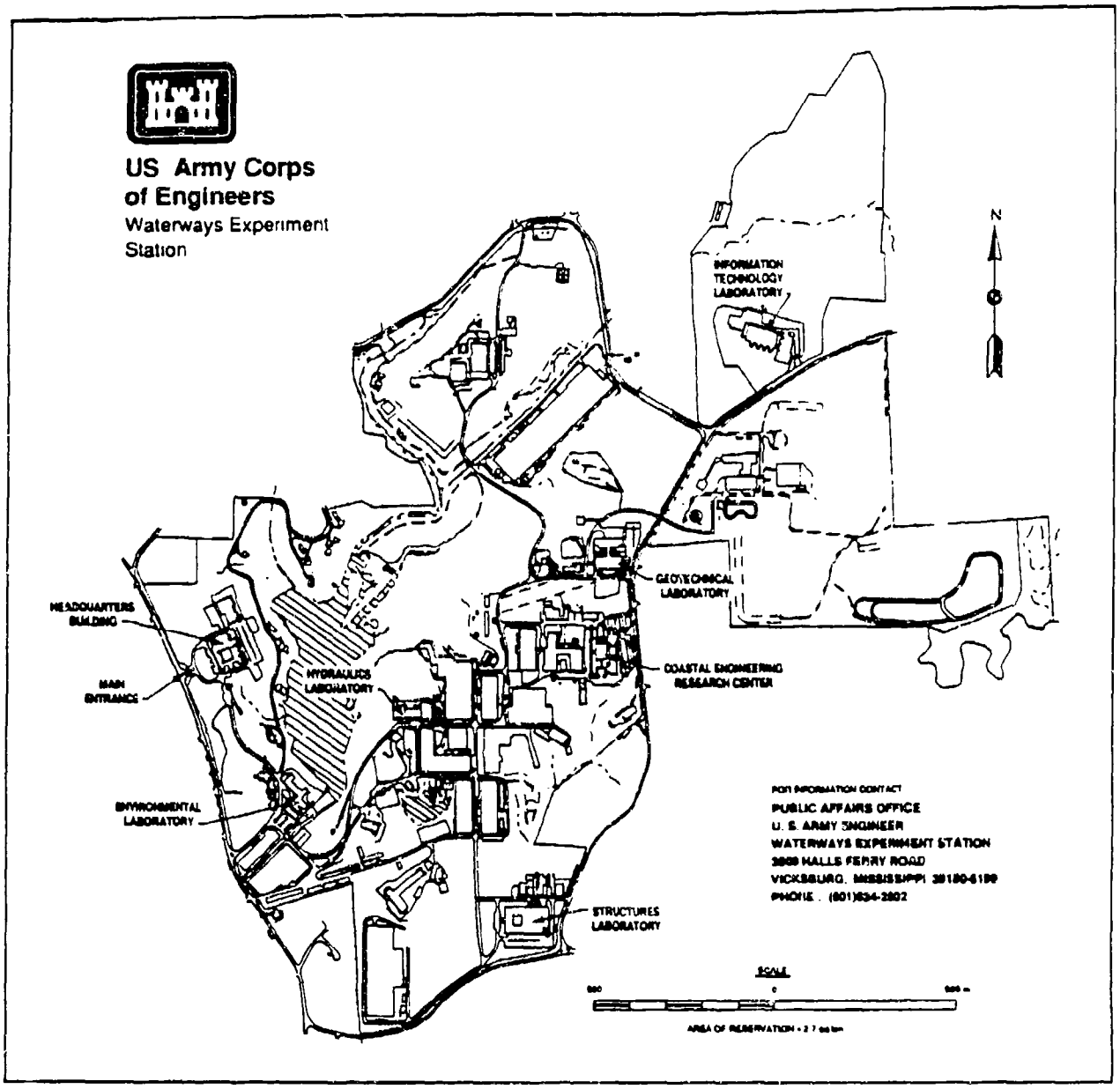
Final report

Approved for public release; distribution is unlimited

Prepared for U.S. Army Environmental Center
Building E4435, Edgewood Area
Aberdeen Proving Ground, MS 21010

Under Contract No. DACA39-93-K-0016

Monitored by U.S. Army Engineer Waterways Experiment Station
3909 Halls Ferry Road, Vicksburg, MS 39180-6199



Waterways Experiment Station Cataloging-in-Publication Data

Ward, William A.

Transition to Ada / by William A. Ward, Jr. ; prepared for U.S. Army Environmental Center ; monitored by U.S. Army Engineer Waterways Experiment Station.

104 p. : ill. ; 28 cm. — (Technical report ; ITL-94-8)

Includes bibliographic references.

1. United States. Army. Corps of Engineers. — Automation. 2. Ada (Computer program language) 3. Government information — Automation. 4. United States. Dept. of Defense — Automation. I. United States. Army. Corps of Engineers. II. U.S. Army Engineer Waterways Experiment Station. III. Information Technology Laboratory (US Army Corps of Engineers, Waterways Experiment Station) IV. U.S. Army Environmental Center. V. Title. VI. Series: Technical report (U.S. Army Engineer Waterways Experiment Station) ; ITL-94-8.

TA7 W34 no.ITL-94-8

Contents

Preface

This report is published in the interest of scientific and technical information exchange; the ideas and findings contained herein should not be construed as an official position of the U.S. Army Corps of Engineers. Use of any trademarks in this report is not intended in any way to infringe on the rights of the trademark holder.

While much work has gone into the development of this report, it should be emphasized that it is introductory, not comprehensive, in nature; doubtless there are questions related to Ada which have gone unanswered. Nevertheless, after reading this document managers should be aware of the broad issues associated with the transition and of the resources available when further investigation is necessary.

Much of the information presented here is drawn from a study conducted by Dr. Orville E. Wheeler, Memphis State University, which specifically addressed the transition of PC-targeted systems development from a C, C-Scape, dbVista environment to an environment based on Ada (Wheeler 1992). Ms. Lynn Mikulich, Construction Engineering Research Laboratory (CERL), contributed to the description of the Rational Environment, and Mr. Ralph Kahn, Oracle Corporation, to the discussion on Ada/Oracle interfaces. Dr. Windell F. Ingram reviewed this report and made many helpful comments. The author gratefully acknowledges the contributions of all these individuals. Where noted, information has been drawn from the Ada Information Clearinghouse (AdaIC).

This report was prepared under the auspices of the Executive Software Subcommittee established by the Field Information Management User's Group (FIMUG) of the US Army Corps of Engineers (USACE). FIMUG is a field advisory body to the Director of Information Management. The Executive Software Subcommittee was assigned the task of preparing a report giving background on and discussing issues relevant to the mandated transition to the Ada programming language. This report completes that task.

The production of this report was sponsored by the U.S. Army Environmental Center (AEC) and funded through the U.S. Army Engineer Waterways Experiment Station (WES) Information Technology Laboratory (ITL) under Contract

No. DACW39-93-K-0016 from March 3, 1993 to December 31, 1993.

Mr. Mark N. Bovelsky was Chief of the Information Management Branch, AEC, and also Chairman of the Executive Software Subcommittee of the FIMUG during the preparation of this report. The contract was monitored by Dr. Windell F. Ingram, Chief, Computer Sciences Division, ITL. Dr. N. Radhakrishnan was Director, ITL, Dr. Robert W. Whalin was Director of WES, and COL Bruce K. Howard, EN, was Commander.

1 Executive Summary

This report addresses issues relevant to the transition to the use of Ada from a Corps of Engineers perspective. The direct discussion of these issues is preceded by background material on Ada itself. First, the Department of Defense (DoD) software crisis that led to the development of Ada is described and the causes underlying it are presented. Potential solutions to the crisis are discussed, including programmer productivity aids, structured programming, standardization efforts, computer-aided software engineering (CASE) tools, and research centers. Next, a brief history of Ada is presented to show how it fits into the Government's approach to meeting the crisis. This includes a description of the systematic design and review process to which preliminary versions of Ada were subjected, followed by a discussion of the guidelines which apply to the use of Ada, including the Congressional mandate to use Ada and the pertinent DoD and Army regulations.

The second major section of this report discusses the Corps transition to Ada. This transition will involve not only a change in the programming language used by the Corps, but also a change in development philosophy; software engineering principles must be incorporated into the development process for the transition to be successful. The various issues to be addressed by the Corps in order to accomplish this are then presented. The first of these is the selection of a compiler for PC platforms. This issue is discussed using results drawn from a Corps-sponsored study which reviewed various Ada compilers and development tools. The importance of formal training for programming staff is stressed. A four-week sequence of courses is proposed which covers object-oriented design, Ada coding, and object-oriented analysis. Other aspects of training, including attendance at technical conferences and acquisition of relevant literature, are also discussed. Because of their importance to the Corps during the Ada transition, two special topics are then addressed: the Rational Environment (a state-of-the-art Ada programming support environment) and use of Ada with the Oracle RDBMS. CASE tools will have to be acquired to support this effort; examples of such tools, including code production environments, screen generators, database managers, analysis and design aids, and metrics collectors, are described to illustrate the range of capabilities available.

The report concludes with recommendations concerning practical steps Corps development sites can take to ensure a successful transition to the use of Ada. The most immediate of these are acquisition of Ada compilers and programming support environments, initiation of a training sequence, selection of Ada experts at each site, and making a decision regarding acquisition of the Rational Environment. An important long-range recommendation would be for the various development sites to successfully participate in the Software Engineering Institute's (SEI) Software Process Assessment. As a final note, readers should view the mandate to use Ada as an opportunity for the Corps to assume a leadership role within the Army in the area of software development. The Corps of *Engineers* has a long history of *engineering* many things well, and there is no reason why the Corps should not *engineer* software well, too.

2 Introduction to Ada

Past Problems

Government activities depend increasingly, if not totally, on information technology for their successful completion. Because declining appropriations are forcing reductions in manpower while workloads continue to increase, this dependence will accelerate as functional organizations automate more and more tasks. This will in turn increase the workload of information management (IM) as they aid these organizations in this automation process. Unfortunately, IM is not immune from current budgetary pressures; as a result, it must automate its own software development activities.

This fiscal difficulty, although serious enough in and of itself, is not the only problem faced by software developers. There is another which has plagued IM for many years and which manifests itself in software systems which are delivered years late, which are delivered without all of the required capabilities, or which are not delivered at all. Furthermore, many such systems, even those which meet their functional specifications, perform at unacceptable levels because they are designed for and installed on computer systems of insufficient power. To determine the extent of this problem, the US General Accounting Office conducted a study of nine software development contracts totaling $6.8 million (U.S. General Accounting Office 1979, p. 11). The results were not encouraging:

- $3.2 million was spent on software delivered but never successfully used.
- $1.95 million was spent on software paid for but not delivered.
- $1.3 million was spent on software used but extensively reworked or later abandoned.
- $198,000 was spent on software that could be used after changes.

- $119,000 was spent on software that could be used as delivered.

Furthermore, the average delay on these contracts was an additional 75 percent of the original time estimate. Unfortunately, there is little evidence that the situation has improved since this study was performed.

There are several reasons for this system development crisis. First, many project managers are optimistic when specifying the scope of a project because they want to solve as many of their IM problems as possible. Second, reaching consensus on automation needs is time-consuming; the current method involves isolating key personnel at a remote site for one or more months to analyze needs and produce requirements. This is disruptive to work at the home sites and there is still no guarantee that significant changes in these requirements will not be necessary. Third, contractors have little incentive to limit project scope, because the more objectives there are, the more money there is to be made. Fourth, even if Government project managers are realistic in their estimates of what can reasonably be done in a given period of time, they generally do not have the technical expertise or the time to ascertain if the contractor is following accepted software development practices, much less using appropriate modern software engineering methodology. Furthermore, contractor personnel who actually design and implement the system rarely have sufficient software engineering experience because software engineering is not viewed as an important part of the computer science curriculum at many universities (Frakes, Fox, and Nejmeh 1991, p. 3).

Potential Solutions

The computing community has long recognized the need to facilitate the timely production of large software systems. Many of these have focused on improving the productivity of individual programmers, including the invention of assemblers (to replace the use of machine language), compilers for third generation high-level languages (to replace the use of assemblers), fourth-generation language processors (to replace use of compilers), full-screen language-specific editors (to replace use of keypunches and line editors), the use of source code version control tools such as *make* and *sccs* (to reduce the complexity of the edit-compile-test cycle), and the use of structured programming (to reduce the complexity of the software itself).

The Government has also initiated a number of efforts to resolve the software development crisis. Perhaps the most successful of these are the standardization activities, including the establishment and ongoing revision of the Federal Information Processing Standards (FIPS) (NIST 1991) by the National Institute for Standards and Technology (NIST, formerly the National Bureau of Standards); these FIPS address issues ranging from hardware data formats to programming language features. More recent efforts at standardization include the required use of Ada for all new DoD software projects and the establishment of standard documentation formats (DoD-STD-2167/2167A). Other organizations are also active in this process, including the Institute for Electrical and Electronic

Engineers (IEEE), which has promulgated standards for software engineering and software quality assurance (IEEE 1983, IEEE 1984, IEEE 1986), as well as the American National Standards Institute (ANSI), and the International Standards Organization (ISO), which have produced standards for various programming languages. Finally, there are de facto standards which exist by virtue of their widespread use; examples include operating systems (DOS, MVS, and UNIX) window environments (Microsoft Windows and the X Window System), and printer control languages (HPPCL and PostScript).

Unfortunately, use of programmer productivity aids is tactical in nature because it addresses only the implementation phase of the software life cycle. The standardization efforts are an example of a more global approach, but enforcing use of such standards by contractors is sometimes difficult for non-technical personnel, and in any case such efforts do not go far enough toward addressing the real issues. What is needed is a more strategic approach; experts in both the academic and commercial worlds have recognized this and a number of possible approaches have been proposed, including groupware (GW), computer-aided software engineering (CASE) tools, object-oriented design (OOD), object-oriented programming (OOP), and rapid prototyping.

DoD has recognized the potential of such methods and has established several centers to encourage further software engineering research and to transfer this technology to DoD projects. These centers include the SEI, at Carnegie-Mellon University in Pittsburgh, the US Army Software Engineering Directorate at Fort Monmouth, the US Army Institute for Research in Management Information, Communications, and Computer Science (AIRMICS) at the Georgia Institute of Technology, and the US Air Force Software Technology Support Center (STSC) at Hill Air Force Base. Other DoD programs that address software engineering issues include the Software Technology for Adaptable, Reliable Systems (STARS) Joint Program Office, partially sponsored by the Defense Advanced Researched Projects Agency (DARPA), and the Data and Analysis Center for Software, which provides the DoD with data, information, products, and services in order to facilitate technology transition.

A History of Ada

Why has Ada been selected by DoD as the required high-order language? How does it fit into the solution approach described above? To answer these questions, some background on the history of Ada is required. A study performed during 1973 and 1974 revealed that the Department of Defense was spending $3 billion per year on software, that this cost was steadily rising, and that most of the cost was consumed not by development of new systems, but by maintenance of old systems. The reasons for this were apparent; DoD software projects were so specialized that the contractor who originally developed the system was almost invariably the only one who could maintain it, thus limiting competitiveness and eliminating any incentive to cut costs (Cohen 1986, p. 2).

A second problem was the use of obsolete programming languages. Because these languages were developed in the 1960s, they do not provide features to support modern software development concepts, particularly top-down design, structured programming, data abstraction, module reuse, and program validation. Furthermore, they do not allow interface checking between separately compiled program modules, and assembly language must often be used for device-level I/O to compensate for the language's deficiencies in that area (Cohen 1986, p. 2).

This situation was further aggravated by frequent contractor use of project-specific programming languages or platform-specific versions of "standard" languages. This discouraged development of software tools because their one-time use could not justify the initial cost of their production. Even when some simple tools, such as compilers, linkers, editors, and configuration managers, were developed, they were typically not used on multiple projects (Cohen 1986, p. 2).

The findings of the 1973-1974 study led to the establishment of the High Order Language Working Group (HOLWG), which was charged with identifying a standard language for use throughout DoD, particularly for embedded systems. The HOLWG included members from the Office of the Secretary of Defense, all three branches of the military, the DARPA, the Defense Communications Agency, and the National Security Agency. The initial requirements for this language, tentatively named DoD-1, were released in April 1975 and circulated within the Government and to selected outside experts in the field of programming languages. This requirements document, termed "Strawman," was subsequently revised to produce "Woodenman" in August 1975 and "Tinman" in January 1976; each revision incorporated recommendations provided by reviewers from government, industry, and academia. After further study, the HOLWG announced in January 1977 that no existing language satisfied the Tinman requirements. The Tinman requirements were rewritten in the form of an actual language specification, dubbed "Ironman," and an RFP for a language design was released in April 1977. Following an incremental review spanning two years in which the number of designs was narrowed from seventeen to four to two, the design proposed by CII-Honeywell Bull was accepted. The formal language specification was ultimately published in 1983 as ANSI/MIL-STD-1815A-1983, and as a FIPS (NIST 1985).

Since the standardization of the Ada language specification in 1983, DoD has steadily moved towards adopting Ada as *the* standard language for systems development. Initially, requirements for the use of Ada applied only to embedded weapons systems. Even then, it was sometimes possible to obtain waivers to these requirements. However, concerns about runaway software development costs, coupled with the recognition that use of Ada throughout DoD would result in major cost savings, prompted more stringent regulations. On 2 April 1987, DoD issued Directive 3405.1 which stated,

> The Ada programming language shall be the single, common, computer programming language for Defense computer resources used in intelligence systems, for the command and control of military

> forces, or as an integral part of a weapons system. Programming languages other than Ada that were authorized and being used in full-scale development may continue to be used through deployment and for software maintenance, but not for major software upgrades. Ada shall be used for all other applications, except when the use of another approved higher order language is more cost-effective over the application's life-cycle, in keeping with the long-range goal of establishing Ada as the primary DoD higher order language (HOL) (U.S. Department of Defense 1987).

In this context, a major software upgrade is defined to be "redesign or addition of more than one-third of the software." The complete text of this directive is given in Appendix A.

HQDA emphasized the Army's adherence to this policy in LTR 25-90-1, issued on 16 July 1990, which specifically addressed "Implementation of the Ada Programming Language." This letter stated,

> The Ada programming language as defined in ANSI/MIL-STD-1815A-1983 is the single, common, high order computer programming language for all computer resources used in the Army unless another language is mandated by a higher level directive. Existing software need not be rewritten in Ada solely for the purpose of converting to Ada. All systems, however, will transition to Ada when the next hardware/software upgrade requires modification of more than one-third of the existing code over the system life cycle, unless a waiver is obtained (U.S. Department of the Army 1990).

Waivers are not necessary for use of (1) off-the-shelf software requiring no Government maintenance or modification, (2) SQL as an interface to a DBMS, (3) non-SQL-compliant 4GLs to produce prototypes or systems with a useful life of less than 3 years, and (4) machine or assembly language in performance critical situations when the ratio of non-Ada to Ada source code does not exceed 15% and the non-Ada code is not more than 10,000 lines. All other situations require waivers. Requests for waivers must be accompanied by a thorough cost and technical analysis which clearly demonstrates the cost effectiveness of the proposed language. The complete text of this letter is given in Appendix B.

Finally, on 5 November 1990, the President signed the FY 1991 DoD appropriations bill (Public Law 101-511) Section 8092 of this law, popularly known as the "Ada Mandate," states,

> Notwithstanding any other provisions of law, after June 1, 1991, where cost effective, all Department of Defense software shall be written in the programming language Ada, in the absence of special exemption by an official designated by the Secretary of Defense." (U.S. Congress 1990)

Supporting information on this law is given in Appendix C

The Superiority of Ada

Responses to these policy statements from the software development community have often been quite skeptical, and many still question Ada's purported cost benefits. From a scientific perspective, one of the strongest challenges has come from adherents of object-oriented design (OOD) and object-oriented programming (OOP). This relatively recent technology emphasizes the correspondence between objects and operations in the real world and data types and operations in a software system. It promises order-of-magnitude improvements in software development productivity. Although object-oriented principles and methodologies are generally language independent, several languages have been designed to provide object-oriented features. Smalltalk, at the same time a programming environment and a programming language, is preeminent in this respect. It claims to be a pure object-oriented language, but it typically makes heavier demands on system resources than general-purpose procedural languages, and few large software systems have been implemented in it. However, these particular criticisms are not valid for Ada's strongest challenger, C++. This language was designed to directly support object-oriented techniques and at the same time be completely upward compatible with C. In an attempt to obtain a waiver to the use of Ada, the Air Force conducted a study to determine the relative cost effectiveness of Ada and C++ (U.S. Department of the Air Force 1991). (An overview of the Air Force report is given in Appendix D.) Surprisingly, the study determined the opposite of what many anticipated; Ada, not C++, was found to be superior.

This study consisted of four substudies which compared the two languages from various perspectives. The first substudy, performed by the Institute for Defense Analyses (IDA), addressed tools, environments, and training. Their report concluded that there are more US vendors of Ada compilers than C++ compilers (28 vs. 18), that Ada compilers are subjected to a relatively rigorous validation process whereas C++ compilers cannot be validated because no C++ standard even exists, that Ada has cross-compilation systems and code generators while C++ does not, and that 223 universities and 13 DoD installations teach Ada compared to 4 and 0, respectively, for C++. The second substudy was conducted by the SEI and included a quantitative comparison of the two languages based on six categories: capability, efficiency, availability/reliability, maintainability/extensibility, life cycle cost, and risk. Ada was clearly superior by a score of 78.8 to 63.9 (on a 100-point scale). The third substudy, completed by CTA, concluded, "Ada projects have reported 15% higher productivity with increased quality and double the average size. Normalizing these data to comparable size projects would result in an expected Ada productivity advantage of about 35%." Specifically, their data indicated that C++ suffered from error rates three times greater than Ada (as measured at the software formal qualification test). The final study, performed by TRW, established 18 criteria to judge the life cycle cost effectiveness of the two languages. A panel of experts was then used to establish the scores and weights for each of these criteria; the score for

Ada was 23% higher for MIS systems and 24% higher for C^3 systems. The study's overall conclusions are given below.

> All four substudies which specifically compared Ada and C++ (IDA, SEI, CTA, TRW) report a significant near term Ada advantage over C++ for all categories of systems. This advantage could be eroded as C++ and its supporting environments mature over the next few years. On the other hand, as aggressive overseas Ada initiatives stimulate even wider domestic Ada interest, as Ada tools/environments further mature, and when the Ada update (Ada 9X) is complete, the balance could tip further in Ada's favor.

> Adding to the case for Ada is that the Ada scoring so well herein is Ada's 1983 version, MIL-STD-1815A. Just as C++ incorporates into C certain software engineering concepts already in Ada (e.g., modularity, strong typing, specification of interfaces), so an Ada update now underway will bring into Ada selected features now included in C++. This update, known as the Ada 9X Project, is targeted for completion in 1993 (Ada 9X Project Office 1991). The product of extensive community involvement (including the C^3 and MIS communities), Ada 9X will bring to Ada such improvements as decimal arithmetic, international character sets, improved input/output, support for calls between Ada and other languages, further representation specifications, and inheritance/polymorphism (popular features of C++). At the same time, Ada 9X has been designed so that neither existing Ada benefits nor performance will be lost. For example, Ada 9X inheritance will be controlled so as not to reduce life cycle supportability.

> In summary, Ada is the most cost effective programming language for DoD applications. Specifically, it is not possible to make a credible case for the existence of classes of "more cost effective" C++ systems compared to Ada. Business cost effectiveness data collected for this study are typified by the TRW conclusion that Ada provides development cost advantages on the order of 35% and maintenance cost advantages on the order of 70%. In terms of future life cycle costs, it will be some time before data exists which could justify a cost savings for C++. Today, there is limited life cycle data available for Ada and almost none for C++.

> For the foreseeable future, then, there are more than enough reasons for the DoD to stick firmly with Ada for all high order language (3GL and 3-1/2 GL) development and for exclusive use as a target language of 4GL application generators in the large class of applications for which 3GL code must supplement generated code (U.S. Department of the Air Force 1991).

These conclusions carry particular weight for the Corps when one notes that the study focused on information and C^3 systems (not embedded weapons systems), and that those who initiated the study were biased toward C++ and against Ada.

3 The Corps Transition to Ada

Scope of the Transition

Without the Ada mandate, the Corps would have three alternatives in the area of programming language selection: to stay with whatever is currently in use, to adopt a more modern language other than Ada, or to adopt Ada. The first approach has been tried for over a quarter century and has proven ineffective. The most promising choice for the second alternative, C++, is still immature, lacking a standard and associated development tools. Ada, the third choice, has proven its effectiveness in numerous large projects, has a standard which has been in place almost ten years and which is rigorously monitored, and has a wide variety of tools which are available to assist programmers and analysts. Even if the Government had not required the transition to Ada, there would still be many compelling reasons to do so, and no compelling reasons not to do so. Pursuing this third choice will, however, have a price. The immediate costs will be associated with the purchase of Ada compilers and the training of programmers. Learning the Ada language will be relatively straightforward for those developers who are already experienced with Pascal and, to a lesser extent, C. Those who have programmed exclusively in FORTRAN or Cobol will typically require a longer training period.

However, if the transition to Ada involved only a change in programming language, it would indeed be relatively painless. Unfortunately, this is not the case. Additionally, to maximize the benefits of the change, recognized software engineering principles must be incorporated throughout the software life cycle. More specifically, the underlying methodology used by project leaders to design large software systems must be changed; the leading contender for this methodology is the object-oriented approach already mentioned. Secondly, computer-aided software engineering (CASE) tools must be acquired to automate the development process; these range from language-sensitive editors which help minimize syntax errors to high-level design tools which facilitate application of an object-oriented methodology. Finally, application programmers and analysts must be equipped with a development platform which has greater functionality than the current DOS-based environment. The memory limitations inherent in

DOS, the absence of protected mode execution, its lack of virtual memory, its inability to support multiple users, the absence of true multitasking (even with Windows), and its inability to support multiple users are deficiencies which limit the productivity of applications developers. That Microsoft recognizes these shortcomings is evidenced by their soon-to-be-released NT operating system. Other candidates include IBM's OS/2, and of course, the more mature and nonproprietary UNIX operating system. However, these three additional issues would need to be resolved regardless of the language selected; furthermore, they must be resolved to make optimum use of Ada in the development of large software systems.

Compiler Selection

Many computer manufacturers provide compilers for their own systems while other software companies have produced cross compilers primarily for embedded systems. From the Corps' perspective, however, there are currently three major independent Ada compiler vendors: Alsys, Telesoft, and Verdix. Of these three, Alsys is probably the vendor of choice for the Corps' Control Data (CD) and PC platforms, although the Verdix VADS environment is probably better for RISC workstations (SRA 1992). Alsys offers compilers for a broad range of hardware platforms, including IBM-compatible PCs running DOS and UNIX, various Motorola 68000-based computers (Apple Macintoshes, Apollo Domain workstations, HP 9000/300s, and Sun-3s), most RISC-based workstations and servers (DECstations, IBM RS/6000s, MIPS, and Sun SPARC), DEC VAX/VMS workstations and minicomputers, and IBM 370 series mainframes. The platforms of particular interest to the Corps are 80X86 running DOS, to be discussed below, and the Control Data 4000 series. Control Data has licensed the Alsys Ada compiler for MIPS-based systems and has placed it on the Corps of Engineers Automation Plan (CEAP) contract. This product consists of a compilation system, which includes the Ada compiler, global optimizer, linker, library manager, run-time executive, standard Ada packages, and an ISO-compliant math library, as well as a tool set, which includes a source-level symbolic debugger, recompiler to automate the system rebuilds, cross reference generator, source code reformatter with user-controllable options to enforce particular coding standards, syntax checker, source generator to produce source code from compilation units, name expander to convert identifiers visible through use clauses into fully qualified names, and profiler to determine where execution time is spent. Pertinent contract information is given in Table 1.

Many Corps programmers will initially use 80X86/DOS platforms for their development work. Although there are over 270 validated Ada compilers available today, only three run under MS-DOS: FirstAda from Alsys, OpenAda from Meridian Software Systems (recently purchased by Verdix), and Janus/Ada from R&R Software. As part of a Corps-sponsored study (Wheeler 1992), these three compilers were compared using cost, documentation, adherence to standard, resources required, support environment availability, vendor history and future, upward migration, and performance as evaluation criteria. The evaluations were based on published reports in trade journals, personal communication with

Table 1
Ada Compilers on the CEAP Contract

CEAP Computer System	Contract Line Item Number	Initial License Cost	Monthly Maint. Charge
CD 4330	1075TJ	$15,000	$250
CD 4680	1075TK	$20,000	$335

vendor representatives, documentation supplied with each product, actual benchmarks performed for this study, as well as personal experience with each of the compilers. Each of the three compilers was assigned a score from 1 to 10 for each criterion and weights indicating the relative importance of each criterion were then used to produce a weighted average. A complete description of the evaluation, including rationale for the scores presented here, is provided in the study's final report (Wheeler 1992). The results displayed in the Table 2 indicate the superiority of Alsys FirstAda (A) over its Meridian (M) and R&R (R) competitors. Its documentation and performance were clearly better than the other two while recent price reductions from Alsys make cost differences negligible (Alsys' volume price is $973.50 each, including media, documentation, and one year of maintenance).

Table 2
PC Ada Compiler Comparison[1]

Evaluation Criteria	Raw Scores			Weight Factor	Weighted Scores		
	A	M	R		A	M	R
Cost	8	9	9	5	40	45	45
Documentation	9	6	2	10	90	60	20
Standard	9	8	6	8	72	64	48
Resources	6	8	9	2	12	16	18
Environment	8	6	7	8	64	48	56
Vendor	9	7	7	7	63	49	49
Migration	9	6	3	5	45	30	15
Performance	9	6	2	10	90	60	20
Aggregate					**476**	**372**	**271**
Percent					**86.5**	**67.6**	**49.3**

[1] From (Wheeler 1992)

As a final comparison, the well-known Whetstone benchmark (Curnow and Wichman 1976) was used to compare the performance of Alsys FirstAda, Microsoft FORTRAN, Borland C, and Borland Turbo Pascal. The tests were performed on a 16 MHz Zenith 386. Using a full math library with run-time checking disabled, the results, as measured in thousands of Whetstone instructions per second, were 767, 650, 572, and 111. Although this comparison is obviously not exhaustive, it does indicate that Ada compilers, at least as exemplified by FirstAda, are competitive with other available language compilers. Again, further information on this benchmark may be found in the WES report. (Wheeler 1992)

Personnel Training

As noted earlier, because a successful transition to Ada will involve more than just changing compilers, so will training programmers to use Ada involve more than teaching Ada syntax and semantics. The required changes in thinking and working patterns cannot be induced in software developers without formal training. This approach will emphasize to the students the importance management attaches to the Ada transition, much more so than if they are handed a book and told to pick up the technology on their own. This training should develop in personnel a new perspective on the design of large software systems and encourage the incorporation of modern software engineering practices into their development projects. Based on successful experience at CERL, THAMA, and WES, it is recommended that this training include an overview of the software development crisis and the history of Ada (two days), object-oriented design with Ada (one week), coding programs in Ada (two weeks), and object-oriented requirements analysis (one week). Modern software engineering principles should be incorporated into every course. No more than twenty students should be in any class and only personnel having prior experience with another procedural programming language should attend. At least one or two weeks should elapse between courses to prevent students from becoming overwhelmed with new information. Managers should attend the design and analysis courses if at all possible. System development work in Ada should follow soon after the course work to provide immediate positive reinforcement of the concepts learned during the training. When a large project is envisioned which involves development teams at different sites simultaneously making the transition to Ada, a common training program is advised to provide a common technical vocabulary, design methodology, and development philosophy.

The purpose of the initial two-day overview session should be to motivate the development staff for participation in the remaining courses. This introduction should cover the DoD software development crisis and the various attempts within the software engineering community to address it. The history of Ada should be given to present the language as an essential part of the solution to this problem. Ada should be compared to other programming languages, noting its similarities, its unique capabilities, and the rigorous standardization process to which Ada compilers are subjected. Software engineering should be introduced not as the application of an array of CASE tools, but as the application of good engineering management principles to the development of software, i.e., understanding the problem, planning the solution, constructing the design, executing the design, and communicating the solution. The need to change from "hand crafted" software to "engineered" software should be stressed, and the improvements in understandability, reliability, maintainability, and efficiency which will result from this change should be noted. Object-oriented techniques should be introduced as a tool to effect this transition. Case studies should be provided to illustrate the successful use of Ada in the development of large software systems. Students should be impressed with the notion that Ada's overall design and many of its specific features were intended to support software engineering principles and that the transition to Ada should be viewed as a commitment to adopt those principles.

The remaining courses are intended to provide the necessary knowledge about the language, software engineering, object technology, and the development environment to allow programmers to begin using Ada. The object-oriented design (OOD) course should cover the following topics: motivation for using object classes to model the problem space; comparison of functional and object class problem decomposition; introduction to object-oriented design methodology; material necessary to introduce the notion of separate compilation, including Ada program units, program structure, and the Ada program library; the black-box principle for information hiding and data abstraction, including encapsulation, packages, and private types; various Ada types, including integer, floating point, enumeration, record, array, and access types; generics and predefined units. Case studies should be provided to illustrate the design process and how to critique a design. Exercises should be assigned to provide experience in the use of Ada as a program design language (Texel 1992 Object-Oriented Design with Ada). The Ada coding course should cover the following topics: overview of the syntax of various Ada statements; Ada identifiers; subprograms, input/output, and exceptions; more in-depth coverage of Ada types than was provided in the previous course including when to use distinct types, subtypes, and predefined types; Ada control structures: loops, if statements, case statements; additional information on generics; tasking. Class time should be about half lecture and half hands-on work on coding exercises (Texel 1992 Ada Coding). The object-oriented requirements analysis (OORA) course should cover the following topics: introduction to OORA: comparison with other requirements analysis techniques; role of object classes, attributes, and relationships in OORA; use of state and process models; transition to the design stage. Practical application issues should be addressed through case studies and extensive exercises (Texel 1992 Object-Oriented Requirements Analysis). The following topics from software engineering should also be covered at some point in the course sequence: aims of software engineering; models of software development; requirements definition and evolution; design processes and strategies; software components and reusability; programming for reliability, including exception handling and defensive programming; software testing; programmer productivity, quality control, and software metrics.

After these courses are completed, steps should be taken to maintain staff expertise. Programmers should be encouraged to obtain membership in the ACM, ACM SIGAda, and the IEEE Computer Society. If possible, the organization should subscribe to relevant publications from these organizations, as well as any others which may be helpful. A recommended list would include *Ada Letters*, *CrossTalk*, *Communications of the ACM*, and *IEEE Computer*. If possible, *ACM Computing Surveys*, *ACM Transactions on Information Systems*, *ACM Transactions on Software Engineering and Methodology*, *IEEE Software*, *IEEE Transactions on Sofiware Engineering*, and *Software: Practice and Experience* should be added as well. This library should also include texts and conference proceedings covering Ada, object-oriented technology, and software engineering; the bibliography at the end of this report should serve as a starting point for this effort.

A few individuals should be selected to become the organization's Ada experts. They should be the recipients of further training on a regular basis, stay aware of the activities of the Ada Joint Program Office, and should also attend selected conferences and seminars, e.g., the TRI-Ada Symposium, the Washington Ada Symposium, and the Software Technology Conference. Refer to Appendix G: Ada Events Calendar for a more complete list of such events. Furthermore, they should become familiar with Ada-related resources available over the Internet, such as those at the SEI, the Software Technology Support Center, and the SIMTEL20 database; these include technical reports as well as repositories of potentially reusable Ada software components. The catalog by Nyberg (Nyberg 1991) will be of particular assistance in this respect. In addition to their regularly assigned duties, these individuals would be responsible for keeping abreast of advances in Ada development and software engineering practice, acting as consultants for the rest of the development staff, providing introductory assistance to new employees, and teaching short courses as necessary.

The Rational Environment

Every software system is implemented in three environments, the decision environment, the design environment, and the coding environment. The decision environment is that in which the system's requirements are specified; here decisions are made regarding how much of a process should be automated, what inputs must be supplied to the system, how those inputs are to be obtained, what outputs the system will produce, how those outputs are to be presented, and how users will interact with the system. Generally, input into this decision-making process is provided by functional managers and, occasionally, by prospective end-users and system developers. The output from the decision environment provides strategic guidance to the developers of the system. The subsystems are assigned to development teams who may actually be independent contractors. These teams are responsible for taking the broad guidelines supplied to them and incrementally refining the design until an actual working system is obtained. The organizational and computational context in which this is done is termed the development environment; it is broken into two subenvironments, the design environment and the coding environment. In the former, technical managers make decisions regarding breakdown of subsystems into packages and modules, and the information flow between them, while in the latter, programmers translate the design into actual program text, which is then compiled, debugged, and tested.

There are numerous potential problems in this process. Because the magnitude of these problems is so large and because they are so widespread, each is worthy of, and is being addressed by, research devoted to that particular field. In each case experts are attempting to apply increased levels of automation to bring this software development crisis under control; within the first environment, the current approach involves the use of the IDEF methodology (D. Appleton Company 1992) for business process modeling and computer-supported collaborative work (CSCW) techniques such as electronic meeting systems (Johansen 1988) for reaching group consensus; within the second, CASE tools, such as code

generators, are proposed; within the third, programmer productivity enhancements, such as language sensitive editors, have been applied.

However, even if satisfactory solutions are found to the problems within all of these environments, the maximum benefit will not be obtained unless the three environments are integrated. The issue to be resolved is one of communication; each environment involves a different group of people, each depending on feedback from the others to accomplish their objectives. Seamless interfaces between these environments must be developed to facilitate smooth transfer of knowledge and specifications. THAMA is currently sponsoring research at WES to remedy this situation. This work involves development of prototype interfaces between existing solutions (IDEF, CSCW, and CASE) to make progress in a timely manner. At the same time preliminary investigations are under way to explore the feasibility of a single environment-spanning software system to provide such an integrated environment.

The system which holds the most promise for progress in this area is the Rational Environment™, which currently addresses the coding environment only. Site visits to Rational user sites indicate that it is far superior to all other currently available systems for implementing large software systems in Ada. Because the Rational provides a programmable interface which will allow design tools to communicate with it, it is hoped that it will be possible to extend its area of applicability to the first two environments. Because it is so effective in enhancing programmer productivity, it is worth elaborating on its capabilities.

The Rational Environment (or simply, "the Rational") is an integrated, interactive software engineering environment for development, testing, maintenance, and control of large Ada projects. It replaces the usual collection of edit, make, and version control utilities with a completely integrated system for program development. For performance reasons, the compute intensive portion of this environment executes on Rational's own proprietary hardware, the R1000 processor. This environment server is accessed over an Ethernet through a software interface available on Sun SPARC and IBM RS/6000 workstations running the X Window System, as well as PCs running Microsoft Windows. One such R1000 processor will support anywhere from five to fifteen developers, depending on the mix of editing and compilation. An influx of large compilation jobs degrades the system's performance, but the intelligence built into the system reduces the likelihood of such a situation.

Use of the Rational on a large project begins with use of the Rational Design Facility (RDF). It supports the requirements analysis and design phases of the software life cycle. Ada is used in this context as a program design language (PDL) for requirements capture, software design, and MIL-STD-2167A compliant document generation. The RDF also supports integration with third-party CASE and desk top publishing software packages, including Cadre Teamwork™ and Interleaf TPS™.

During the code development phase, the Rational provides an intuitive window-based interface which allows users to browse Ada systems according to

syntactic structure or semantic dependencies. After identifying the portion of the code which is of interest, programmers use Rational's Configuration Management and Version Control (CMVC) feature to check out individual procedures or entire packages for development or modification. This allows one programmer exclusive access to some portion of the software and thus prevents developers from "stepping on one another's toes." Rational's Ada-sensitive editor is then used to enter program statements. Ada packages, procedures, functions, loops, and other program structures are automatically closed with the appropriate end statements, thus minimizing the possibility of syntax errors. This editor also allows customization and enforcement of site-specific coding standards.

Periodically, the current version of the program must be tested. Ada is a strongly typed language, and it checks subprogram calls with subprogram definitions to insure argument compatibility. Furthermore, compilation units (e.g., subprograms and packages) may import other packages in order to use their type and variable declarations, and such imported references must match the original declarations. These relationships between compilation units are called dependencies and a compilation unit which accesses or uses the resources of another is said to be dependent on that unit. Obviously, the dependent unit must be compiled after the unit on which it depends. In large Ada software systems it is common for what seems to be a minor modification to require recompilation of many units because of the chain of dependencies. If such systems are sufficiently large, the time required for recompilation becomes the bottleneck in the development process. The Rational Environment avoids this problem through intelligent dependency checking and incremental compilation. The former feature allows the Rational to avoid recompilation of dependent units if a modification has no impact (e.g., a comment was added or changed). After this impact analysis, only those *statements* which are dependent are recompiled. (This statement by statement compilation is possible because Ada programs are stored using the Descriptive Intermediate Attributed Notation for Ada (DIANA) (Goos et al. 1983). Ada objects (e.g., statements) are implemented as DIANA structures that represent syntactic structure and include semantic information and executable code. This is much different from the conventional approach of source code, object code, and executable images that are stored in separate files.)

When a version of the software is required for the actual target platform, Rational's Target Build Facility allows convenient transfer of source code to and compilation on a particular host. This process is facilitated by the Rational Compilation Integrator, which permits any third-party Ada compiler to be integrated with and managed by the Rational Environment. This tool allows developers to build software for multiple platforms (mainframe, minicomputer, workstation, personal computer) at the same time. With the Compilation Integrator, only those portions of the program (individual unit, unit closure, subsystem, or, if required, entire system) are (re)compiled. Obviously this is more efficient than compiling the entire system prior to every test.

Periodically, project managers and team leaders require information about the structure and status of the system under development. Rational Insight™ performs this function by gathering information about Ada units and subsystems

from the integrated information repository and then displaying it in a graphical fashion. The resulting diagrams show dependencies among the various program structures and aid in understanding the relationships among the software components. During system reengineering, this is an invaluable capability.

The SEI performed an evaluation of the Rational Environment for DoD (Feiler, Dart, and Downey 1988). This study assessed its functionality in the areas of system installation, system administration, detailed design and coding functions, testing and debugging, compiler quality, configuration management and project management functions. The results of this study were very positive. THAMA, WES, and CERL have jointly conducted their own evaluation of the Rational Environment, visiting Rational's Washington office for two technical briefings and interviewing Rational users at four sites in northern Virginia. Impressions from these visits were very favorable. Users who have conducted their own studies have emphasized that no other development environment approaches the Rational in functionality. These users have particularly high praise for Rational's customer assistance, stressing Rational's commitment to the success of the customer's project. There are, however, a few disadvantages to use of the Rational. It is quite expensive; an entry level system, including hardware and software, costs about $250,000. Furthermore, a significant amount of training is required to take full advantage of this sophisticated environment's capabilities. In spite of this required investment in hardware, software, and personnel, the Rational Environment is capable of increasing productivity, lowering labor costs, and adjusting to changes in requirements or target platforms. For the foreseeable future, it appears to be the best way to effectively develop large systems in Ada.

Ada/Oracle Interfaces

The Congressional mandate to use Ada has posed some significant technical problems for those involved in MIS system development. Most MIS systems must utilize commercially available graphical user interfaces (GUIs) and/or relational database management systems (RDBMSs); unfortunately, the interfaces between Ada and these packages have traditionally been rather poor. This problem is compounded by the lack of a single standard in each of these areas. For example, a program written for the MOTIF GUI is not portable to the Open Look GUI, much less to Microsoft Windows, and an application which accesses the Xdb RDBMS generally requires modification in order to run with the Oracle RDBMS. The NIST has recognized these problems and responded with an RDBMS standard, FIPSPUB 127-1 (NIST 1990 February), and a GUI standard, FIPSPUB 158 (NIST 1990 May). Adherence to these standards by both software vendors and developers will make development of portable applications in Ada much easier.

The reaction of many software DBMS vendors, including Oracle, to the adoption of Ada and the development of standards has been to provide bindings between Ada and their products. There are currently four ways to construct an Ada/RDBMS interface: (1) proprietary application programmer interface, (2)

FIPS 127-1 compliant embedded SQL precompiler, (3) FIPS 127-1 compliant module language compiler, and (4) SQL-Ada Module Description Language (SAMeDL) compiler. Because the first method is neither standard nor portable, it does not satisfy the Government's needs. The last technique is based on research at Carnegie Mellon University. It is still in an embryonic stage with no applicable standards and no available commercial implementations. Neither of these techniques will be discussed here.

The second method is currently the most widely used method of binding Ada programs to SQL-based RDBMSs. From a technical perspective, such a FIPS-compliant Ada/SQL binding is merely a way of allowing SQL statements to be embedded in an Ada program so that the program may exchange information with a FIPS-compliant RDBMS. Prior to actual compilation, such a program is first processed by a precompiler which translates the SQL statements into Ada statements. Oracle's implementation of this method, Pro*Ada, was introduced in 1989 and has been sold to over 300 sites around the world. It is available on a variety of platforms, including the Corps' Control Data CEAP systems as well as other major UNIX platforms (e.g., HP, IBM, SCO, and Sun). Furthermore, Pro*Ada has the distinction of being the first Ada/SQL embedded SQL binding to pass the NIST test for ANSI compliance. It has since passed that test on many different platforms, both UNIX and non-UNIX. Although the use of Pro*Ada adds an additional step to the process of program testing, it actually improves programmer productivity and system performance. Moreover, it allows dynamic construction of SQL statements at runtime. The nature of these benefits will be discussed in the following paragraph.

Improved programmer productivity is a result of Pro*Ada's extensive syntactic and semantic checking. All SQL statements are validated during precompilation so that errors may be detected and corrected before the potentially time consuming compilation and binding of the Ada program. During the semantic checking phase, Pro*Ada queries the database to first determine if the table used in the SQL statement actually exists, and then to see if the table has a column whose name matches the one mentioned in the statement. These features save developers from performing many useless compilations. Increased system performance results from use of Pro*Ada's array interface (an extension to the FIPS standard). Use of this interface allows insertion or retrieval of batches of records with a single database call. For example, an EXEC SQL FETCH may fetch up to 32K records; this can reduce network traffic because one fetch of 32K records requires fewer packets than 32K fetches of single records. Finally, Pro*Ada allows dynamic construction of SQL statements at execution time. This powerful capability allows construction of an SQL statement within a program variable and submission of the statement to the RDBMS while the program is running (a feature similar in spirit to FORTRAN's object-time FORMAT).

The third method is the newest approved standard for binding Ada programs to SQL-based RDBMSs; it differs from the previous approach in that the Ada program and the SQL procedures (modules) are stored in separate files. The SQL file is translated by the SQL*Module compiler into Ada, then compiled, and placed in an Ada library. The Ada program is then also compiled and external

references to the SQL modules are satisfied from this library. Oracle's implementation of this technique, SQL*Module, was announced in June 1992 and should be available in the first half of 1993. Prior to its release it will meet the NIST test for FIPS-compliant module language compilers.

The first benefit of using this third technique is the clean separation of Ada and SQL, thus improving maintainability and allowing developers to specialize in Ada or SQL without having to learn both. It also allows developers to use Ada-specific tools and SQL-specific tools where they are appropriate. Additional productivity is obtained through the use of stored procedures. Developers may store their SQL*Module procedures in a database and access them from an Ada program, thus promoting module reuse.

Finally, another technique for improving productivity is the use of fourth generation languages (4GLs). This involves specification of an application in a high-level design language (the 4GL), and then translation of this program to Ada. Many such code generators are available; however, the vast majority of them generate package specifications, type declarations, and procedure calls, but then provide only procedure stubs. An exception to this is the Oracle product GenerAda, a prototype of which was recently demonstrated at TRI-Ada '92. It creates a complete application in Ada which is compilable and executable. Early in 1993, WES will be working with Oracle, under the sponsorship of THAMA, to test and evaluate this product using a real application.

Code Production Environments

Even if the Corps decides to obtain one or more Rational systems, they will not be capable of supporting a large number of developers. Cost-effective tools must be found to support the remaining developers. For this purpose the Corps requires a comprehensive tool set containing high-level analysis and design aids, a code production environment, a database interface, a test generation facility, software metrics collectors, and project management tools. These should be seamlessly integrated with an intuitive interface and be available on a variety of platforms, including PCs, RISC workstations, and CD 4000 series servers.

Unfortunately, such an environment does not exist. As a first step in that direction, the Government has defined an Ada Programming Support Environment (APSE) to assist commercial vendors in producing comprehensive software engineering environments. The lowest defined level is called a Minimal APSE (MAPSE) and consists of an editor, compiler, library management system, linker, and run time environment. Except for the latter two items, which are provided by DOS, this MAPSE tool set is provided with the Alsys FirstAda compiler via a menu-driven interface called Adam. Adam is actually built on top of the Alsys AdaWorld command line environment and translates menu selections to AdaWorld commands. In addition to the MAPSE set, other features accessible through Adam include a verifier (fast syntax checker), source code level debugger, reformatter, cross referencer, make facility, and a line counting metric collector. Users may substitute their own editor for the Adam default, and access

to the operating system and the AdaWorld command line environment is provided. Novice and expert modes of operation are available, which display few or many menu options, respectively. The Adam interface takes up a significant amount of memory. Binding a moderately sized program will occasionally fail due to lack of memory for symbol tables; this may be remedied by switching to the command line environment, Adaworld. In spite of this minor problem, experience with this product indicates that it is a useful, effective, and smoothly integrated system for Ada code production (Wheeler 1992).

The Ada Workstation Environment (AWE), marketed by AETECH, is intended to provide a full set of Ada programming tools. Like Adam, it is built on top of the Alsys AdaWorld command line environment running under DOS. Its features include an editor, library manager, macro generation facility, compiler-binder control facility, templates for types and program units, and a menu for support tools. These support tools, which must be purchased separately, include an on-line Ada reference manual, an on-line computer-assisted instruction system, a hypertext Ada reference manual, an ASCII table, a border graphics drawing tool, tool for conversion of numbers to different bases, and tools to perform top-down structured design, object-oriented design, generation of source code from design, and generation of procedure body from procedure specification. The basic functions work very well, and the template facility, not available in Adam, is particularly useful in making skeletons of program units. Most common tasks have a convenient, single key operation; The built-in editor has a native set of key functions, but may be reconfigured to emulate other editors. The primary alternative to AWE is Adam, and Adam's biggest advantage is that it is bundled with the Alsys compiler. However, even without the additional tools, AWE may offer enough additional functionality to be considered as an alternative, but a decision regarding its adoption should be delayed until a decision is made on whether DOS or UNIX is to be the ultimate development environment.

Screen Generators

Screen Machine, marketed by Objective Interface Systems, is an interactive package that provides a means of graphically creating panels on a screen, editing these panels, and placing fields in them using the PanEdit interface. Panel information is saved in a database and the source code to create these panels (two complete Ada packages) is generated by the program GenCode. To complete the interface, the programmer writes the application software to manipulate these screens and process the input data and then modifies one of the package bodies to handle specific input selections.

The major advantage of Screen Machine is the ease of producing the actual panel creation source code. The user has only to enter the size, location, color, number of fields, and any included title to create a panel, and this is done through an interactive screen. Fields can easily be added to this panel to form a database entry screen or just an information panel. The user is able to see the created panel and edit it without having to write or compile any code. After the user is

satisfied with the appearance of a panel, two approaches are possible: first, the Ada program may access it as needed from a disk file, or second, the Ada source code to create the panel may be generated and then incorporated into a program. The former method minimizes program size, while the latter maximizes program execution speed.

A major disadvantage of the Screen Machine is its documentation. The version evaluated for the Corps was documented in a single three ring binder with chapters dedicated to PanEdit, GenCode, and the package libraries. The text was readable but vague, and lacked examples. Furthermore, the index has not been updated to reflect changes in the text (Wheeler 1992). Thus, the Screen Machine is a good tool for learning the process of screen development without having to actually program a screen from scratch. It enables the user to see the screen as it is being created and allows one to make changes without repetitively recompiling and relinking. The version described here seemed to lack the functionality required for more elaborate I/O, although many developers at the Rational user sites visited by the Corps were using it effectively and recommended it highly. This apparent difference in capability may, however, be due to the Corps's evaluation of Screen Machine on a PC running DOS, while the Rational sites were using it on RISC-based workstations running UNIX and the X Window System.

The Textual User Interface (TUI) is a screen development tool written entirely in Ada and marketed by AdaSoft, Inc. TUI is comprised of three tools: AdaWindows, which contains procedures for window generation and control, AdaMenus, which deals with menus of various types, and AdaForms, which deals with forms for input and output. Documentation for each package is provided separately and is of high quality. A facility is provided for maintaining default values between sessions and for maintaining a history of all data entered. Some screen builders are, in effect, low level libraries of functions, while others, such as Screen Machine provide a much higher level of abstraction. TUI is a compromise lying between these two extremes. It is composed of procedures and functions, but at an intermediate level with much of the detail still hidden from the user. TUI includes a number of useful features; for example, when an editor is needed in a field, it is present by default and does not have to be explicitly invoked, used, and then exited. AdaForms handles I/O for all standard Ada types except records and arrays. strings are the only array type allowed. In addition, it provides a List_Class, for selection from a displayed list, and a Text_Class, for I/O of multiple lines of text. Ada type checking is performed on input data to a field before it is accepted. The TUI is a well designed, logically structured, flexible, well documented set of components for screen building in Ada, and is available for both DOS and UNIX platforms (Wheeler 1992).

Database Managers

Interfaces between Ada and DBMSs will be important to the Corps for the development of management information systems. Possible interfaces to Oracle have already been discussed; three additional possibilities are considered here. The first is AdaSAGE, a public domain database generation package written in Ada that produces a complete system, including interface screens. The second alternative would make use of two products marketed by AdaSoft: AdaManager, which creates and manipulates databases, and AdaQuest, which serves as an interactive interface to AdaManager. The third possibility involves construction of an Ada "wrapper" around an existing DBMS (dbVista III) using the Ada interface pragma.

AdaSAGE is a public domain package developed and maintained by the Idaho National Engineering Laboratory (INEL); it is designed to facilitate rapid development of Ada systems. A descendant of the SAGE system developed in FORTRAN by INEL about 10 years ago, AdaSAGE is the result of work on the Marine Corps Combat Readiness Evaluation System (MCCRES). INEL produced a prototype of MCCRES in Ada for the Marine Corps in Alsys Ada and at the same time converted SAGE to Ada. Primarily suited to MIS applications, its capabilities include command line and embedded ANSI-compliant SQL, graphics, communications, formatted windows, on-line help, sorting, editing, and more. AdaSAGE applications can be run in the stand-alone mode or in a multiuser environment. One of the most powerful features of AdaSAGE is the Generic RApid Prototyping Language (GRAPL). This interpretive language allows complete prototyping of a database application that can be executed interpretatively without actual compilation. After the developer is satisfied with the application, it is relatively easy to convert it to Ada. Using AdaSAGE, it is possible to design and implement an application in a minimum of time that performs well and is easy to modify.

All four Services have reported significant success in developing applications with AdaSAGE; their experience has shown AdaSAGE to be equally applicable to both small and large projects. The Government considers it to be an extremely valuable tool and continues to fund its maintenance and enhancement. If the Corps decides to use AdaSAGE, the software itself is free, but Corps personnel should attend formal AdaSAGE training provided by INEL, and one individual at each development site should join the AdaSAGE User's Group ($1,600/year). As a member, this person will serve as the organization's point of contact for AdaSAGE and will be provided with the following services: current versions of software libraries and documentation, technical support via a telephone hot line, access to electronic bulletin board services, newsletter subscription, and invitation to the annual meeting held in Idaho Falls.

The second approach is the use of AdaManager/Adaquest. AdaManager produces a relational database that is dynamically structured rather than built on a static schema. It allows the logical structure of the data to be changed, and additional columns to be added without unloading and reloading the data. It is possible to insert, delete, update, fetch, select, and view rows as well as join and save

tables, and re-order rows during retrieval. The base types used are those of Ada except that strings are the only structured types allowed. AdaQuest is an interactive interface to the AdaManager database system. It is a stand-alone tool containing 32 commands (procedures) and provides either menu or command line access to all of the functions of AdaManager. It is particularly valuable for constructing databases. AdaQuest can be used to define databases, tables, types, and columns, as well as performing administrative tasks. Although it is easy to use, it does require a knowledge of AdaManager. Documentation is complete, logically arranged, and clearly written; its high quality is consistent with that supplied with other AdaSoft products (Wheeler 1992).

The final database interface discussed here is the construction of an interface between Ada and the dbVista III package now in use fro PC-targeted, THAMA-sponsored software. WES has developed such an interface, and it is currently being tested. It is not anticipated that dbVista will be widely used with Ada, particularly for developing new software, but the interface may be valuable for converting some existing programs to Ada when such software is being substantially modified. The Ada interface modules and a user guide are available from WES.

Analysis and Design Aids

Rational Rose, from Rational, is a graphical CASE tool which supports object-oriented analysis and design. It is available on IBM RS/6000 and Sun SPARC workstations as a stand alone product; X terminals attached to those systems may also access Rose. It does not require the Rational Environment. It allows developers to create class diagrams which serve as blueprints of high level system abstractions, specify class attributes which provide detailed design information, and illustrate the design through object diagrams of system mechanisms. Rose uses the Booch notation (Booch 1991); it enforces the notation's syntax and verifies its semantics by prohibiting occurrences of multiple inheritance, as well as by checking for class visibility and multiple class relationships. All of the underlying information is represented in ASCII, thus it allows developers to use their current source code control system to manage versions of the design just as they manage versions of the code. Rose is customizable and extensible; users may modify menus and integrate the product with other CASE tools. There are several possibilities for output: printed PostScript, encapsulated PostScript file, or FrameMaker-compatible file. A floating license which allows a single copy of Rose to be shared among multiple users is available for $3,995.

Like Rose, Teamwork/Ada is also based on the X Window System. Developed by Cadre Technologies, it is available on RISC workstations, including those from DEC, HP, IBM, and Sun, and also on X terminals connected to those hosts. The centerpiece of this product is the Ada Structure Graph (ASG) editor, which allows developers to create, view, and change Ada design components in a graphical manner using the Buhr notation (Buhr 1984). Using the ASG, it is possible to display a diagram in several windows, thus allowing detailed editing in a close-up view and simultaneous visualization of the global impact of a modification in a high-level view. Teamwork/Ada automates the

transition from design to implementation through integration with Cadre's Ada Source Builder (ASB) and Design Sensitive Editor (DSE). When used with the ASG, these two tools enforce consistency between the design and the coded implementation, prohibiting changes that would make the design obsolete. The ASB analyzes the output of the ASG and produces Ada code which corresponds to the design and contains "code frames." Implementation details may then be inserted into the generated code using the DSE, but only within the code frames. The DSE automatically detects Ada syntax errors, and may be configured to emulate other editors and to enforce site-specific coding standards. Existing Ada code may be processed using the ASG Builder which creates ASGs from source code in order to facilitate reuse, reengineering, and maintenance. Information produced during this design process is saved in a project database and may then be accessed by the Document Production Interface, Teamwork/DPI, to produce documentation compliant with DoD-STD-2167A. Teamwork/Ada is part of Cadre's Teamwork environment which includes facilities for simulation of software systems, construction of real-time systems, modeling of information, and communication with widely-used DBMSs (e.g., Ingres, Oracle, and Sybase). If the Corps decides to use the Rational Environment, Teamwork's interface to that system would make it a powerful addition to the coding environment. If the Corps decides against Rational, Teamwork, along with an appropriate compilation system, is surely one of the top contenders for an integrated development environment.

Metrics Collectors

An important aspect of the development process is quality control. Unless data is obtained to measure program quality, this will be difficult, if not impossible. One such tool for gathering these data is AdaMAT, from Dynamics Research Corporation. This is a static source code analyzer which uses DIANA-based metrics to measure management concerns such as reliability, portability, and maintainability, as well as software engineering concerns such as code simplicity, modularity, self-descriptiveness, clarity, and independence. These scores are calculated by calculating the ratio between the number of adherences to a particular guideline and the total number of adherences and nonadherences. AdaMAT is useful in many activities throughout a project, including design assessment, code walk throughs, error prevention and discovery, system installation and checkout, and maintenance.

How to Proceed

The use of Ada offers many advantages over other languages. Many of its features were specifically intended to support good software engineering practice; examples include its library consistency requirement its extremely thorough compile time checking. Its technical advantages are clearly seen and other languages, e.g., Turbo Pascal, Modula-2, and C++, have attempted to incorporate at least some of its features. Since Ada is more advanced than many other

language systems, it requires a higher level of sophistication for optimum use. It must be systematically introduced or chaos can result.

A well conceived plan should be followed which will promote a smooth transition to Ada without false starts and wasted effort. The following paragraphs propose an outline of this plan. Throughout the transition, the key to success will lie in the education of the personnel involved. Management must recognize that several years will elapse before development staff acquire the expertise and fully adopt the techniques that will produce the dramatic cost savings advertised by various Ada proponents. Development staff members must realize that formal training is only the first step in their personal transition to Ada, and that they must understand and then put into practice not just new programming language syntax, but a new development philosophy as well. Furthermore, project sponsors must understand that software engineering is engineering. If they were sponsoring a civil works project, such as the construction of a suspension bridge, they would not show up at the site a month after project initiation and ask what percent of the bridge has been completed or ask for a prototype bridge to drive over in order to determine its "look and feel." They must understand that the new development process to be instituted as part of this transition will involve more initial planning and less time spent coding and that the result will be a product which is more reliable and easier to maintain. Finally, everyone should realize that some aspects of the transition will be perpetual. Specifically, procedures and techniques must periodically evaluated and if they have proven ineffective, or if new technology has rendered them obsolete, then they must be changed.

It is imperative that several steps be immediately taken to begin the transition to use of Ada. If they are not, the first several projects implemented in Ada will be delayed while developers acquire expertise in the Ada language, object-oriented design, and software engineering methods and tools. These near-term actions, to be completed during the first year of the transition, include:

- Acquisition of Ada compilers. Alsys FirstAda seems to be a good first choice for PC platforms, although if an interface to Microsoft Windows is necessary, the Meridian product should be considered, as well. The Verdix compiler should be viewed as a strong contender for RISC-based workstations.

- Acquisition of an APSE for PC platforms. The selection of this environment will depend on the hardware/OS/compiler combination under consideration. For 80X86/DOS/Alsys, the Adam interface bundled with the Alsys compiler is a reasonable, cost-effective choice. For RISC-based workstations running UNIX, the situation is similar in that a high-quality APSE is available from the compiler vendor; this is particularly true of the Verdix compiler and its associated VADSpro programming environment.

- Initiation of an Ada training sequence. Many training vendors, e.g., EVB, Fastrak Training, and Texel, have courses that cover the topics listed earlier. One should be selected and an initial group should begin a formal training sequence.

- Selection of potential Ada experts. These individuals should immediately be assigned the tasks noted earlier, specifically those associated with reviewing current Ada technology and with assisting other developers in making the transition.

- Acquisition of Ada reference materials. These include the serials noted previously as well as items from the bibliography.

- Adoption of software engineering methodology as standard practice. This includes, among other things, development of formal design specifications, application of object-oriented design techniques, adoption of and adherence to an Ada style guide, code walkthroughs, thorough testing, and collection of software and productivity metrics for individuals as well as for the group as a whole.

- Decide whether or not to obtain the Rational Environment. This is an important initial decision on which many future decisions will depend, including selection of hardware, compilers, and CASE tools. Furthermore, if the decision is positive, then time must be allowed to train personnel in the use of the system.

Other tasks are equally important but need not be accomplished right away; work on them should begin during the first year of the transition and be completed by the end of the second. Note, however, that some of them are ongoing activities.

- Continuous investigation of available Ada technology. All of the areas discussed in this report require constant monitoring. Knowledge of advances in software engineering methodologies, CASE tools, and object-oriented technology, progress at Government laboratories, and status of other large-scale Ada projects will all be useful in current development efforts.

- Adoption of a reuse policy. The Ada experts noted above must investigate existing Government software repositories to determine what components will be useful to current projects. A local reuse library must be established.

- Acquire one or more RISC-based UNIX workstations. These platforms support better CASE tools and provide a development environment with more functionality. Appropriate tools should be purchased simultaneously, e.g., Cadre's Teamwork, HP's SoftBench, IDE's Software Through Pictures, and Rational Rose. These should be tested on a small project in a prototype fashion and the most promising product employed on subsequent projects.

- Encourage employees to stay current in the field. As part of their job assignment, employees should periodically review books, journal articles, case studies, or products and prepare a review; this could then be shared with the rest of the staff in a short one-hour presentation. If this were scheduled once a month, perhaps as a brown-bag luncheon/seminar, then it would not be burdensome to the presenter or the staff.

- Review progress in making the transition. This involves evaluation of all of the short-term steps noted above and making any necessary mid-course corrections. Specifically, attention should be paid to modification of estimating models based on experience with Ada to date, evaluation and possible revision of the training program, review and enhancement of software engineering techniques and tools, and updating library holdings on Ada and software engineering,

- Make plans for future efforts. Plans should be made to address the following issues: use of code generators, applications of artificial intelligence in the project's functional domain, and the transition to Ada 9X.

- Plan to participate in an SEI Software Process Assessment. This accreditation procedure evaluates an organization's state of software development practice based on the SEI Capability Maturity Model. This scale of this model's evaluation ranges from 1, which describes an ad hoc, chaotic situation to 5, which describes a software development approach that is repeatable, defined, managed, and optimizing. This is possibly the most important of all the recommendations.

The mandate to use Ada should be viewed as an opportunity for the Corps to assume a leadership role within the Army in the area of software development. The Corps of *Engineers* has a long history of *engineering* many things well, and there is no reason why the Corps should not *engineer* software well, too. Hopefully this report has provided initial guidance which will help make that goal a reality.

References

Ada 9X Project Office. (February 1991). *Ada 9X revisions relating to information systems applications*, Fact Sheet.

Booch, G. (1991). *Object-oriented design with applications*. Benjamin/Cummings, Menlo Park, California.

Buhr, R. J. (1984). *System design with Ada*. Prentice-Hall, Englewood Cliffs, New Jersey.

Cohen, N. H. (1986). *Ada as a second language*. McGraw Hill, New York.

Curnow, H. J. and Wichman, B. A. (February 1976). "A synthetic benchmark," *the Computer Journal* 19(1), 43-49.

D. Appleton Company. (1992). *Corporate information management process improvement methodology for DoD functional managers*. D. Appleton Company, Fairfax, Virginia.

Feiler, P., Dart, S., and Downey, G. (1988). An evaluation of the Rational Environment (CMU/SEI-88-TR-15), Software Engineering Institute, Pittsburgh, Pennsylvania.

Frakes, W. B., Fox, C. J., and Nejmeh, B. A. (1991). *Software engineering in the UNIX/C environment*. Prentice-Hall, Englewood Cliffs, New Jersey.

Goos, G., Wulf, W. A., Evans, A. Jr., and Butler, K. J. (1983). *DIANA: an intermediate language for Ada*. Lecture Notes in Computer Science, 161, Springer-Verlag, Berlin.

Institute for Electrical and Electronic Engineers. (1983). *IEEE standard glossary of software engineering terminology*. IEEE Std 729-1983, Institute for Electrical and Electronic Engineers, New York.

_____. (1984). *IEEE standard for software quality assurance plans*. IEEE Std 730-1984, Institute for Electrical and Electronic Engineers, New

York.

_____. (1986). *IEEE standard for software quality assurance planning*. IEEE Std 983-1986, Institute for Electrical and Electronic Engineers, New York.

Johansen, R. (1988). *Groupware: computer support for business teams*. The Free Press, New York.

National Institute of Standards and Technology. (November 8 1985). *Ada*. Federal Information Processing Standards Publication (FIPSPUB) 119, U.S. Department of Commerce, National Institute of Standards and Technology, Computer Systems Laboratory, Gaithersburg, Maryland.

_____. (February 2 1990). *Database language SQL*. Federal Information Processing Standards Publication (FIPSPUB) 127-1, U.S. Department of Commerce, National Institute of Standards and Technology, Computer Systems Laboratory, Gaithersburg, Maryland.

_____. (May 29 1990). *The user interface component of the applications portability profile*. Federal Information Processing Standards Publication (FIPSPUB) 158, U.S. Department of Commerce, National Institute of Standards and Technology, Computer Systems Laboratory, Gaithersburg, Maryland.

_____. (October 1991). *Federal Information Processing Standards Publications (FIPSPUBS) index*. NIST Publications List 58, U.S. Department of Commerce, National Institute of Standards and Technology, Computer Systems Laboratory, Gaithersburg, Maryland.

Nyberg, K. A. (1991). *Ada: sources & resources*. Grebyn Corporation, Vienna, Virginia.

Systems Research and Applications Corporation (SRA),. (October 9 1992). discussion between SRA and Corps personnel regarding the Rational Environment and alternatives, Fairfax, Virginia.

P. P. Texel & Co.. (1992). Ada coding (Version 3.1), P. P. Texel & Co., Eatontown, New Jersey.

_____. (1992). Object-oriented design with Ada (Version 3.1), P. P. Texel & Co., Eatontown, New Jersey.

_____. (1992). Object-oriented requirements analysis (Version 3.1), P. P. Texel & Co., Eatontown, New Jersey.

U.S. Congress. (November 5 1990). Public Law 101-511, U.S. Congress, Washington.

U.S. Department of Defense. (April 2 1987). Computer programming language policy (DoD Directive 3405.1), U.S. Department of Defense, Washington.

U.S. Department of the Air Force. (July 9 1991). Ada and C++: a business case analysis, U.S. Department of the Air Force, Washington.

U.S. Department of the Army. (July 16 1990). Implementation of the Ada programming language (HQDA LTR 25-90-1), U.S. Department of the Army, Washington.

U.S. General Accounting Office. (November 9 1979). Contracting for computer software development--serious problems require management attention to avoid wasting additional millions (FGMSD-80-4), U.S. General Accounting Office, Washington.

Wheeler, O. E. (September 30 1992). Ada transition study (prepared for the Waterways Experiment Station under Contract Number DACA39-92-K-0018), Memphis State University, Memphis, Tennessee.

Bibliography[1]

Andrews, E., ed. *Concurrent programming with Ada*. Benjamin-Cummings, Menlo Park, California. (ISBN: 0-8053-0088-0; $19.16/paper text)

Atkinson, C. (1991). *Object-oriented reuse, concurrency and distribution: an Ada-based approach*. Addison-Wesley, Reading, Massachusetts. 270 pp. (ISBN: 0-201-56527-7; $37.75)

The book includes an introduction to DRAGOON, the object-oriented language that combines the power of object-oriented languages with the software engineering features of Ada; examples of DRAGOON's use of multiple inheritance to handle aspects of concurrency and distribution; illustrations of how DRAGOON's features my be implemented in Ada.

Atkinson C., et al. (1988). *Ada for distributed systems*. Ada Companion Series, Cambridge University Press, Cambridge. 147 pp. (ISBN: 0-521-36154-0; $39.50)

Describes the final report of the DIstributed Ada DEMonstrated (DIADEM) project, which studied the problems and developed solutions for using Ada to program real-time, distributed control systems. Demonstrates new techniques for controlling such systems from a distributed Ada program.

Ausnit, C. N., et al. (1985). *Ada in practice*. Professional Computing Series, Springer-Verlag, Berlin. 195 pp. (ISBN: 0-521-36154-0; $32.50)

Identifies and resolves issues related to Ada usage and promotes effective use of Ada in general programming, design practice, and in embedded computer systems. Contains 15 case studies that cover five general areas of the Ada language: naming conventions, types, coding paradigms, exceptions and program structure.

Baker, L. (1989). *Artificial intelligence with Ada*. McGraw-Hill, New York. 361 pp. (ISBN: 0-07-003350-1; $39.95)

Presents approximately 8,000 lines of full coding in Ada along with functions which include backward-chaining expert systems shells forward chaining expert systems shells and an ATN natural language parser. Discusses the code for implementing each program and illustrates each by one or more examples.

Barnes, J. G. P. (1989). *Programming in Ada*. Addison-Wesley, Reading, Massachusetts. 494 pp. (ISBN: 0-201-17566-5; $32.25)

Discusses Ada using a tutorial style with numerous examples and exercises. Assumes readers have some knowledge of the principles of programming. Covers the following: Ada concepts, lexical style, scalar types, control structures, composite types subprograms, overall structures, private types, exceptions advanced types, numerics types, generics tasking, external interfaces.

Baum, J. (1986). *The calculating passion of Ada Byron*. Archon Books, Hamden, Connecticut. 133 pp. (ISBN: 0-208-02119-1; $23.50)

Details the life of Ada Byron, her training in mathematics, her tumultuous relationship with her mother and her contribution to the study of science.

Berzins, V. and Berzins, L. (1991). *Software engineering with abstractions*. Addison-Wesley, Reading, Massachusetts. (ISBN: 0-201-08004-4)

Biggerstaff, T. J. and Perlis, A. J., eds. (1989). *Software reusability concepts and models: Volume 1: concepts and models*. ACM Press, New York. (ISBN: 0-201-08017-6)

_____. (1989). *Software reusability concepts and models: Volume 2: applications and experience*. ACM Press, New York. (ISBN: 0-201-50018-3)

Bjoerner, D. and Oest, O. N. (1980). *Towards a formal description of Ada*. Lecture Notes in Computer Science, Springer-Verlag, Berlin. 630 pp. (ISBN: 0-387-102833; $31.00/trade)

Describes the Ada programming language, discusses compiler development and provides a formal definition of Ada.

Booch, G. (1991). *Object oriented design with applications*. Benjamin-Cummings, Menlo Park, California. (ISBN: 0-8053-0091-0)

Offers guidance for constructing object-oriented systems and provides a description of object-oriented design methods. Includes examples drawn from the author's experience in developing software systems and five application projects.

_____. (1988). *Software engineering with Ada*. Benjamin-Cummings,

Menlo Park, California. 580 pp. (ISBN: 0-8053-0604-8; $31.95)

Introduces Ada from a software engineering vantage. Addresses the issues of building complex systems. Includes new features in this second version: a more thorough introduction to Ada syntax and semantics, an updated section on object-oriented techniques to reflect the current state of knowledge and improved examples that illustrate good Ada style for production systems development.

Booch, G. (1987). *Software components With Ada: structures, tools, and subsystems*. Benjamin-Cummings, Menlo Park, California. 635 pp. (ISBN: 0-8053-0609-9; $35.50/paper)

Catalogs reusable software components and provides examples of Ada programming style. Presents a study of data structures and algorithms using Ada.

Bott, F., ed. (1991). *Ada yearbook 1991*. Van Nostrand Reinhold, New York. (ISBN: 0-442-30836-1; $54.95/trade)

Bover, D. (1991). *Introduction to Ada*. Addison-Wesley, Reading, Massachusetts. (ISBN: 0-201-50992-X; $30.25/trade)

Bryan, D. L. and Mendal, G. (1990). *Exploring Ada: Volume 1*. Prentice-Hall, Englewood Cliffs, New Jersey. (ISBN: 0-13-295684-5; $34.00/text) (See Volume 2 under Mendal)

Describes Ada's type model, statements, packages and subprograms. Includes programming features such as information hiding, facilities to model parallel tasks, data abstraction, and software reuse.

Bryant, R. and Vaget, B. W., eds. (1984). *Simulation in strongly typed languages: Ada, Pascal, Simula*. SCS Simulation Series, 13, Society for Computer Simulation, San Diego, California. (ISBN: 0-317-05019-2; $36.00/trade)

Buhr, R. J. (1990). *Practical visual techniques in system design with applications to Ada*. Prentice-Hall, Englewood Cliffs, New Jersey. 533 pp. (ISBN:0-13-880808-2; $43.20/casebound)

Offers a personal statement on how to use visual techniques to organize one's thinking during the design process.

_____. (1984). *System design with Ada*. Prentice-Hall, Englewood Cliffs, New Jersey. 256 pp. (ISBN: 0-13-880808-2; $48.00 paperback) (ISBN:0-13-881623-9; $55.00/text)

Stresses aspects of Ada important for design. Aims numerous examples of notations at teaching, learning, CAD, and uses in industrial practice. Contains three divisions: 1) provides a top down overview of the design features of Ada; 2) develops the design notation and provides a tutorial on the design process using simple examples; 3) treats advanced issues such as

implementing the X.25 packet switching protocol.

Burns, A. (1992). *Towards Ada9X*. IOS Press. (ISBN: 90-5199-075-8)

This book is a collection of edited papers on the general theme of Ada 9X. Two papers directly address the likely language changes. The first of these is written by one of the Ada9X distinguished reviewers. The second is by one of the team members that is actually implementing the language changes. A further paper describes how the new language features will directly support the programming of hard real-time systems. The book includes a paper written by the chairman of the ARTEWG group, that describes the new release of the catalog of interface features and options for an Ada run-time system (CIFO). Other areas covered include interface bindings, such as to SQL or POSIX, to Ada.

_____. (1985). *Concurrent programming in Ada*. Ada Companion Series, Cambridge University Press, Cambridge. 241 pp. (ISBN: 0-521-30033-9; $34.50/trade)

Reports on Ada tasking offering a detailed description and an assessment of the Ada language concerned with concurrent programming.

Burns, A. and Wellings, A. (1990). *Real-time systems and their programming languages*. Addison-Wesley, Reading, Massachusetts. 575 pp. (ISBN: 0-201-17529-0)

Provides a study of real-time systems engineering, and describes and evaluates the programming languages used in this domain. Considers three programming languages in detail: Ada, Modula-2, and Occam2.

Burns, A. (1987). *A review of Ada tasking*. Lecture Notes in Computer Science, 262, Springer-Verlag, Berlin. (ISBN: 0-387-18008-7; $15.50)

Byrne, W. E. (1991). *Software design techniques for large Ada systems*. Digital Press, Burlington, Massachusetts. 314 pp. (ISBN: 1-55558-053-X; $45)

Introduces design strategies for controlling complexities inherent in large computer programs and in software systems as groups of large computer programs executing concurrently. Focuses primarily on issues associated with the design of software systems as a whole rather than on localized design and coding issues.

Caverly, P. and Goldstein, P. (1986). *Introduction to Ada: a top down approach for programmers*. Brooks-Cole, Monterey, California. 237 pp. (ISBN:0-534-05820-5; $18.50/paper)

Organizes and emphasizes those features that distinguish Ada from other programming languages. Uses a cyclical approach to the treatment of many topics. Gives a brief history of the development of the Ada language. Introduces the I/O capabilities, procedures, character and numeric data types and subtypes, and the concept of an Ada program library. Discusses enumeration, array, record, and derived types, and demonstrates how the package can be

used to encapsulate data types. Explains access types and applications and the encapsulation of data objects in packages. Illustrates how finite-state machines can be represented by packages. Describes the essentials of tasking and deals with blocks and exceptions. Introduces the reader to private types types with discriminates, and generic units.

Cherry, G. W. (1984). *Parallel programming in ANSI standard Ada*. Prentice-Hall, Englewood Cliffs, New Jersey. 231 pp. (ISBN: 08359-5434-X; $48.00/text)

Explores parallel sorting, searching, root finding, process pipelining object (data) flow graphs, exception handling, etc., using Ada.

Chirlian, P. M. (1985). *Introduction to Ada*. Weber Systems. 291 pp. (ISBN: 0-916460-42-8; $19.95)

Provides a basic course in the Ada programming language. (Ada courses and/or self-study)

Clark, R. G. (1985). *Programming in Ada: a first course*. Cambridge University Press, Cambridge. 215 pp. (ISBN: 0-521-25728-X; $47.50/trade) (ISBN: 0-521-27675-6; $21.95/paper)

Introduces the Ada programming language. Targets persons without previous experience in programming. Details how to design solutions on a computer. Concentrates on solving simple problems in the early sections: the later sections explore how packages can be used in constructing large reliable programs. Emphasizes central features such as data types, subprograms, packages, separate compilation, exceptions and files. ANSI/MIL-STD-1815A-1983 is referenced throughout the book.

Cohen, N. C. (1986). *Ada as a second language*. McGraw-Hill, New York. 838 pp. (ISBN: 0-07-011589-3; $36.04/paper)

Explains Ada to those who wish to acquire a reading and writing knowledge of the Ada language. Also a programming reference source.

Conn, R., ed. (1990). *Ada software repository (asr)*. Zoetrope. 35 pp. (ISBN: 0-918432-78-2; $16.95/paper)

Describes how to use the Ada Software Repository, which contains Ada programs, software components, and educational materials, and resides on the host computer of the Defense Data Network (DDN).

Culwin, *Ada: a developmental approach*. Prentice-Hall, Englewood Cliffs, New Jersey.

Intended for use on courses which teach Ada as the first programming language. The book is designed to take the reader from the basic principles of programming to advanced techniques. This book provides a complete introduction to software development using the programming language, Ada. It is not only concerned with the production of Ada programs, but it is also an

introduction to the process of implementation and testing. Features include: a carefully structured tutorial which includes software development, design, testing, and production.

Dawes, J., et al., ed. (1990). *Selecting an Ada compilation system*. Ada Companion Series, Cambridge University Press, Cambridge. 173 pp. (ISBN:0-521-40498-3; $42.95)

Presents the findings of the Ada-Europe specialist group for compiler assessment.

Dawes, J. (1988). *The professional programmers guide to Ada*. Pittman Publishing, Marshfield, Massachusetts. (ISBN: 0-273-02821-9; $100.00x)

Dorchak, S. F. and Rice, P. B. (1989). *Writing readable Ada: a case study approach*. Heath, Lexington, Massachusetts. 244 pp. (ISBN: 0-669-12616-0; $17.00)

Contains a style guide, which gives suggestions for enhancing code readability; devotes a chapter to the discussion of concurrency, an advanced feature of modern programming languages; a fully coded Ada program, along with a sample run; a bibliography, which lists books and articles about Ada and software engineering principles; two indexes, one devoted exclusively to references of case study modules and the other listing important topics and concepts.

Elbert, T. F. (1989). *Embedded programming in Ada*. Van Nostrand Reinhold, New York. 523 pp. (ISBN: 0-442-22350-1; $55.00/trade)

Clarifies Ada for the practicing programmer and for the advanced engineering or computer science student. Assumes the reader has acquired a certain level of sophistication, general concepts normally found in introductory programming texts are not covered. Also, presumes the reader is familiar with operating systems and has a basic knowledge of some block-structured language such as PL/1 and Pascal.

Feldman, M. B. and Koffman, E.B. (1991). *Ada problem solving & program design*. Addison-Wesley, Reading, Massachusetts. (ISBN:0-201-5006-3/diskette) (ISBN:0-201-55560/trade)

Presents Ada to the beginning programmer with emphasis on packages. Contains no dynamic data structures, pointers, or tasking.

Feldman, M. B. (1985). *Data structures with Ada*. Prentice-Hall, Englewood Cliffs, New Jersey. 314 pp.

Highlights the use of Ada as a general purpose programming language. Includes the following: linked lists, queues and stacks, graphs, trees hash methods, sorting, etc. Does not include generics; it was written before compilers could handle generics. Free software available from the author.

Feuer, A. R. and Gehani, N. (1984). *Comparing & assessing programming languages: Ada, C & Pascal.* Prentice-Hall, Englewood Cliffs, New Jersey. (ISBN: 0-13-154840-9; $32.00/paper text)

Fisher, D. A., ed. (1986). *Ada language reference manual.* Gensoft Corp.. (ISBN: 0-9618252-0-0; $12.95/paper text)

Fisher, G., ed. (1989). *Approved Ada language commentaries.* Ada Letters Series, 9, ACM Press, New York. (ISBN: 0-89791-311-6; $30.00/paper text)

Ford, B., et al. (1987). *Scientific Ada.* Ada Companion Series, Cambridge University Press, Cambridge. 386 pp. (ISBN: 0-521-33258-3; $44.50/trade)

Explores aspects of the Ada programming language that are relevant to the scientific (i.e., numeric) community at large. Concentrates on the numeric models of Ada and a number of Ada-specific features (e.g., generics). Reviews guidelines for the design of large scientific libraries in Ada.

Gautier, R. J. and Wallis, P. J. (1990). *Software reuse with Ada.* Peregrinus Ltd., Stevenage, Hertfordshire, England. 205 pp. (ISBN: 0-86341-173-8)

Contains three sections: 1) general reuse issues, comprises a collection of papers on various aspects of Ada software reuse; 2) case studies of Ada reuse in practice; and 3) Ada Reuse Guidelines which appear in their final form in this section.

Gehani, N. (1989). *Ada: an advanced introduction.* Prentice-Hall, Englewood Cliffs, New Jersey. 280 pp. (ISBN: 0-13-004334-6 $32.40/paper)

Introduces advanced problem-solving in Ada. Emphasizes modular programming as good programming practice.

_____. (1984). *Ada: concurrent programming.* Prentice-Hall, Englewood Cliffs, New Jersey. 261 pp. (ISBN: 0-13-004011-8; out of print)

Offers a large collection of concurrent algorithms, expressed in terms of the constructs provided by Ada, as the support for concurrent computation. Explains the concurrent programming facilities in Ada and shows how to use them effectively in writing concurrent programs. Surveys concurrent programming in other languages, and discusses issues specific to concurrent programming facilities in Ada.

_____. (1987). *Unix Ada programming.* Prentice-Hall, Englewood Cliffs, New Jersey. 310 pp. (ISBN: 0-13-938325-5; $34.00/paper)

Focuses on the novel aspects of the Ada language and explains them by many examples written out in full. Examines the interesting differences between the Ada language and other programming languages. Also, notes the similarities between Ada, Pascal, C, PL/I, and Fortran.

Gilpin, G. (1987). *Ada: a guided tour and tutorial.* Prentice-Hall, Englewood

Cliffs, New Jersey. 410 pp. (ISBN: 0-13-73599-0; $21.95/paper)

Reports on the developments in control structures, scalar data types multi-tasking, program structure and access types.

Goldsack, S. J. (1986). *Ada for specification: possibility and limitations*. Ada Companion Series, Cambridge University Press, Cambridge. 265 pp. (ISBN: 0-521-30853-4; $7.50/trade)

Examines the use, role, features and purpose of specification languages, particularly Ada, in a large-scale software project.

Gonzalez, D. W. (1991). *Ada programmer's handbook*. Benjamin-Cummings, Menlo Park, California. (ISBN: 0-8053-2529-8; $13.95/paper)

_____. (1991). *Ada programmer's handbook and language reference manual*. Benjamin-Cummings, Menlo Park, California. 200 pp. (ISBN: 0-8053-2528-X; 19.95/paper)

Presents information intended for those professionals transitioning to Ada. includes a glossary.

Goos, G., et al. (1987). *Diana: an intermediate language for Ada*. Lecture Notes in Computer Science, Springer-Verlag, Berlin. 201 pp. (ISBN: 0-387-12695-3; $20.00/paper)

Describes DIANA, a Descriptive Intermediate attributed Notation for Ada, which resulted from a merger of the properties of two earlier similar intermediate forms: TCOL and AIDA.

Habermann, A. and Perry, D. E. (1983). *Ada for experienced programmers*. Computer Science Series, Addison-Wesley, Reading, Massachusetts. 480 pp. (ISBN: 0-201-11481-X; $29.25/paper)

Offers a comparative review of Ada and Pascal, using dual program examples to illustrate software engineering techniques.

Habermann, A. N., ed. (1987.). *System development & Ada*. Lecture Notes in Computer Science, 275, Springer-Verlag, Berlin. (ISBN: 0-387-18341-8; $25.70/paper)

Heilbrunner, S. (1988). *Ada in industry: Proceedings of the Ada-Europe International Conference, Munich, June 7-9, 1988*. Ada Companion Series, Cambridge University Press, Cambridge. 262 pp. (ISBN:0-521-36347-0; $42.50/trade)

Provides state of the art reports on the Ada programming language.

Hibbard, P., et al. (1983). *Studies in Ada style*. Springer-Verlag, Berlin. 101 pp. (ISBN: 0-387-90816-1; $21.50/paper)

Presents concepts of the abstractions embodied in Ada with five examples: a queue, a graph structure, a console driver, a table handler and a solution to

Laplace's equation using multiple tasks.

Ichbiah, J., et al. (1991). *Rationale for the design of the Ada programming language*. Cambridge University Press, Cambridge. (ISBN: 0-521-39267-5; $54.95)

Presents the rationale behind the design and development of the Ada programming language.

Johnson, P. I. (1985). *The Ada primer*. McGrawi-Hill, New York. (out of print)

_____. (1990). *Ada applications and administration*. McGraw-Hill, New York. 209 pp. (ISBN: 0-07-032627-4ISBN; $39.95/Text edition)

Explains how to ensure the reliable, error-free, cost-effective operation of large computer systems with Ada. Updates and revises earlier edition (first entitled The Ada Primer).

Jones, D. (1989). *Ada in action with practical programming examples*. John Wiley & Sons, New York. 487 pp. (ISBN: 0-471-50747-4; $57.95/text) (ISBN: 0-471-60708-8; $34.95/paper text)

Helps Ada programmers avoid common pitfalls and provides them with many reusable Ada routines. Discusses a variety of numeric considerations user interfaces, utility routines, and software engineering and testing. Provides examples of Ada code.

Katzan, H. Jr. (1984). *Invitation to Ada (condensed edition)*. Petrocelli, Princeton, New Jersey. 173 pp. (out of print)

Introduces Ada in terms of three broad classes of applications: numerical, system programming, and real-time programming.

_____. (1984). *Invitation to Ada*. Petrocelli, Princeton, New Jersey. (ISBN: 0-89433-239-2; $14.95/paper text)

_____. (1982). *Invitation to Ada & the Ada reference manual*. Petrocelli. 429 pp. (ISBN: 0-89433-132-9; $34.95/text)

Calls for the scientific computing community to adopt the Ada programming language. Part II is the Ada Reference Manual 1980 version.

Keeffe, D., et al. (1985). *Pulse: an Ada-based distributed operating system*. APIC Studies in Data Processing, 26, Academic Press, New York. (ISBN: 0-12-402979-1; $39.95/paper)

Keller, J. (1988). *The Ada challenge 1988: strategies risk & payoffs*. Pasha Publications. (ISBN: 0-935453-22-9; $174.00/paper)

Krell, B. (1992). *Developing with Ada: life-cycle methods*. Bantam Books, New York. (ISBN 0-553-0909-3; $54.95/hard cover)

Dr. Krell offers his opinion on the key to using Ada to its fullest potential: a

tested development methodology for implementing real-time Ada systems quickly and efficiently, from requirements and code generation through design and test. By applying the steps outlined in Dr. Krell's book, "software engineers can create real-time systems that are flexible, integrate easily, perform well, and satisfy user needs," according to the publisher.

Krieg-Brueckner, B., et al., ed. (1987). *Anna: a language for annotating Ada programs*. Lecture Notes in Computer Science, 260, Springer-Verlag, Berlin. (ISBN: 0-387-17980-1; $15.50/paper)

Ledgard, H. (1983). *Ada: a first introduction*. Springer-Verlag, Berlin. 130 pp. (ISBN: 0-540-90814-5)

Assumes that the reader has experience with some other higher order programming language. Outlines several key features of Ada; a treatment of the facilities -- concept of data types, the basic statements in the language, subprograms, packages, and general program structure.

———————. (1987). *Ada: an introduction*. Springer-Verlag, Berlin. (ISBN: 0-387-90814-5; $22.00/paper text)

Lewi, P. and Paredaens, J. *Data structures of Pascal, Algol Sixty-Eight, PL-1 & Ada*. (ISBN: 0-387-15121-4; $49.00/paper)

Lomuto, N. (1987). *Problem solving methods with examples in Ada*. Prentice-Hall, Englewood Cliffs, New Jersey. 176 pp. (ISBN: 0-13-721325-5)

Contains a collection of hints on solving programming problems. Includes examples along with sections on the art of thinking, analyzing the problem, systematic development, looking back, ideas for ideas, and case studies.

Luckham, D. C., et al. (1990). *Programming with specifications: an introduction to ANNA, a Language for Specifying Ada Programs*. Texts and Monographs in Computer Science, Springer-Verlag, Berlin. 416 pp. (ISBN: 0-387-97254-4)

Offers an in-depth look at ANNA, a form of the Ada language in which specially marked comments act as formal annotations about the program to which they are attached.

Lynch, B., ed. (1990). *Ada: experiences & prospects: Proceedings of the Ada-Europe International Conference, Dublin, 1990*. Ada Companion Series, Cambridge University Press, Cambridge. (ISBN: 0-521-39522-4; $9.50/trade)

Lyons, T. G. (1986). *Selecting an Ada environment*. Ada Companion Series, Cambridge University Press, Cambridge. 239 pp. (ISBN: 0-521-32594-3 (British); $29.95/trade)

Provides an overview of the Ada Programming Support Environment (APSE). Covers six main issues in selecting an environment. Contains summaries of current approaches to likely problems, indications of deficiencies in existing

knowledge, and checklists of questions to ask when considering a particular environment.

McDermid, J. A. and Ripken, K. (1984). *Life cycle support in the Ada environment*. Ada Companion Series, Cambridge University Press, Cambridge. (out of print)

McGettrick, A. D. (1982). *Program verification using Ada*. Cambridge University Press, Cambridge. 345 pp. (ISBN: 0-521-24215-0; $57.50/Trade) (ISBN: 0-521-28531-3; $29.95/paper)

Discusses such topics as correctness of nonbranching programs invariants and termination proofs via well formed sets, elementary types, arrays, records, access types, packages as well as an encapsulation mechanism for abstract data types, and parallelism.

Miller, N. E. and Peterson, C. (1989). *File structures with Ada*. Benjamin-Cummings, Menlo Park, California. (ISBN-8053-0440-1; $39.95/text)

Mendal, G. and Bryan, D. L. (1992). *Exploring Ada: Volume 2*. Prentice-Hall, Englewood Cliffs, New Jersey. (ISBN: 0-13-297227-1) (See Volume 1 under Bryan)

A method of presentation based on the Socratic method, provides coverage and the semantics of Ada. Discusses focused problems individually. The second volume expands on the larger issues dealing with Ada's more advanced features.

Mohnkern, G. L. and Mohnkern, B. (1986). *Applied Ada*. TAB Books, Blue Ridge Summit, Pennsylvania. 326 pp. (ISBN: 0-8306-2736-7)

Introduces the Ada language on a practical level. Targets persons who understand the basic terminology and concepts of programming. (A particular language is not a prerequisite.) Instructs through examples of programs written in Ada.

Musser, D. R. and Stepanov, A. A. (1989). *The Ada generic library linear list processing packages*. Springer-Verlag, Berlin. 265 pp. (ISBN: 0-387-97133-5: $39.00/trade)

Discloses the purpose of the Ada Generic Library as an attempt to provide Ada Programmers with an extensive, well-documented library of generic packages whose use can substantially increase productivity and reliability. Contains eight Ada packages, with over 170 subprograms for various linear data structures based on linked lists.

Naiditch, D. (1989). *Rendezvous with Ada: a programmer's introduction*. John Wiley & Sons, New York. 477 pp. (ISBN: 0-471-61654-0; $39,95/paper)

Explains Ada to the beginning programmer (knowledge of at least one high level programming language is advised). Concludes each chapter with exercises.

Nielsen, K. (1992). *Object-oriented design with Ada/maximizing reusability for real-time systems*. Bantam Books, New York. (ISBN: 0-553-08955-2)

Shows Ada programmers how to design, implement, and maintain reusable real-time software systems using the object-oriented methods.

Nielsen, K. and Shumate, K. (1988). *Designing large realtime systems with Ada*. McGraw-Hill, New York. 464 pp. (ISBN: 0-07-046536-3; $58.95/text)

Presents a comprehensive methodology for the design and implementation of large realtime systems in Ada. (The reader is expected to have a basic understanding of the Ada programming language.)

Nielsen, K. (1990). *Ada in distributed realtime systems*. McGraw-Hill, New York. 371 pp. (ISBN: 0-07-046544-4; $58.95/text)

Emphasizes design paradigms and heuristics for the practicing software engineer. Provides important background material for the builder of operating systems and runtime support environments for distributed systems. Contains data on distributed systems, process abstraction and Ada for distributed realtime systems, design paradigms for distributed systems, inter-processor communication, virtual and physical nodes, and fault tolerance.

Nissen, J. and Wallis, P. (1984). *Portability and style in Ada*. Ada Companion Series, Cambridge University Press, Cambridge. 202 pp. (out of print)

Examines style and portability guidelines for Ada programmers. Results of work by the Ada-Europe Portability Working Group.

Nyberg, K. A. (1991). *Ada: sources & resources*. Grebyn Corporation, Vienna, Virginia. P. O. Box 497 Vienna, VA. Telephone: 703/281-2194

Nyberg, K. A., ed. (1991). *Annotated Ada reference manual*. Grebyn Corporation, Vienna, Virginia. P. O. Box 497 Vienna, VA. Telephone: 703/281-2194

Contains the full text of ANSI/MIL-STD-1815A with inline annotations derived from the Ada Rapporteur Group of the International Organization for Standards responsible for maintaining the Ada language.

Olsen, E. W. and Whitehill, S.B. (1983.). *Ada for programmers*. Prentice-Hall, Englewood Cliffs, New Jersey. 310 pp. (ISBN: 0-8359-0149-1; $38.00)

Includes many of the subtleties of Ada in a self-paced tutorial format. Contains the following: conceptual overview; predefined types; expressions; basic Ada statements; subprograms; packages; etc.

Orme, A., et al. (1992). *Reusable Ada components sourcebook*. Cambridge University Press, Cambridge. 286 pp. (ISBN: 0 521 40351 0; $49.95)

The authors consider how the Ada software components that may be found in this book could be used. According to the publishers both the novice and the expert software engineer will find useful information in this book.

Pokrass, D. and Bray, G. (1985). *Understanding Ada: a software engineering approach*. John Wiley & Sons, New York. (ISBN: 0-471-87833-2; $32.95/paper)

Price, D. (1984). *Introduction to Ada*. Prentice-Hall, Englewood Cliffs, New Jersey. 150 pp. (ISBN:0-13-477646-1; $26.95/trade)

Presents examples, programs, and program fragments showing Ada's power as a general purpose language, plus step-by-step explanations demonstrating how Ada simplifies the production of large programs. Requires little technical or mathematical sophistication.

Pyle, I. C. (1985). *The Ada programming language*. Prentice-Hall, Englewood Cliffs, New Jersey. 345 pp. (ISBN 0-13-003906-3)

Describes the basic features of the Ada programming language. Covers the issues of program structure, and discusses machine specific issues. Assumes prior knowledge of programming. Introduces topics in the context of embedded systems. Covers the following areas: the basic features of Ada; the particular programming concepts in Ada that will probably be new to most programmers; the issues of program structure; the machine-specific issues that can be expressed in a machine-independent language and advanced treatment.

_____. (1991). *Developing safety critical systems with Ada*. Prentice-Hall, Englewood Cliffs, New Jersey. (ISBN: 0-13-204298-3; $39.00/paper)

A presentation of concepts for practicing engineers or programmers involved with the development of safety-related computer-based systems. Considers the different roles involved in accepting safety related systems and the corresponding human activities. Illustrates how Ada provides a framework in which the design rules for safety can be applied and confirmed. The author explains relationships, with major published guidelines for development of safety related software. Interprets guidelines specifically for Ada. The material presented is for three contemporary viewpoints: analyzer, synthesizer, checker. A senior-level course in Ada programming and software engineering.

Rogers, M. W. (1984). *Ada: language, compilers and bibliography*. Cambridge University Press, Cambridge. 332 pp. (ISBN: 0-521-26464-2; $24.95/trade)

Offers a photo reprint of the Ada standard, a guide listing the characteristics of an implementation that should be taken into account in the specification or selection of an Ada compiler and a bibliography.

Saib, S. H. and Fritz, R. E. (1985). *Introduction to programming in Ada*. Holt, Reinhart and Winston, New York. (ISBN: 0-03-059487-1; $28.95/text)

_____. (1983). *Tutorial: the Ada programming language*. IEEE Computer Society. 538 pp. (ISBN: 0-8053-7070-6; $25.56/paper)

Covers such topics as the history and current status of Ada; basic language;

schedule for industry and DoD; preventing error; readable maintainable, modular systems; real-time features, portability; and environments for Ada.

Savitch, W. J., et al. (1992). *Ada: an introduction to the art and science of programming*. Benjamin-Cummings, Menlo Park, California. (ISBN: 0-8053-7070-6; $33.95/paper text)

Written specifically for the first programming course. It starts with variable declarations, simple arithmetic expressions, simplified input-output, and builds upward toward subprograms and packages. A chapter-by- chapter instructor's guide is also available, as is a program disk with more than 140 completed programs from the text.

Saxon, J. A. and Fritz, R. E. (1983). *Beginning programming with Ada*. Prentice-Hall, Englewood Cliffs, New Jersey. (out of print)

Shimer, R. (1989). *Ada*. Amigo Projects. (ISBN: 0-685-30433-7; $12.00/paper text)

Shumate, K. C. (1987). *Understanding concurrency in Ada*. McGraw-Hill, New York. 595 pp. (ISBN: 0-07-057299-2ISBN; $58.95/text)

Presents a detailed exposition of concurrency in Ada. Looks at five case studies and gives an advanced introduction to real-time programming.

_____. *Understanding Ada*. John Wiley & Sons, New York. (ISBN; 0-471-605-204; $51.00/text)

_____. (1989). *Understanding Ada: with abstract data types*. John Wiley & Sons, New York. (ISBN: 0-471-60347-3; $21.50/text)

Skansholm, J. (1988). *Ada from the beginning*. Addison-Wesley, Reading, Massachusetts. 617 pp. (ISBN: 0-201-17522-3; $29.25)

Describes the principles and concepts of programming in a logical and easy-to-understand sequence and discusses the important features of Ada (except parallel programming). Teaches the basics of writing computer programs. Emphasizes the fundamentals of good programming. Provides a grounding in the programming language Ada. Covers the following: programming designs, the basics of Ada, control statements, types subprograms, data structures, packages, input/output, exceptions dynamic data structures, files, and generic units.

Smedema, C. H., et al. (1983). *The programming languages Pascal, Modula, Chill, Ada*. Prentice-Hall, Englewood Cliffs, New Jersey. 154 pp. (ISBN: 0-685-08596-1; $16.95/trade)

Provides an informal introduction to the most important characteristics of Pascal, Modula, CHILL, and Ada. Discusses languages in historical order. Includes the history, application area, standardization aspects and future prospects of each.

Sodhi, J. (1990). *Computer systems techniques: devclopment, implementation, and software maintenance*. TPR, Scarsdale, New York. (ISBN: 0-8306-3376-6) Phone: 800-822-8138

_____. (1990). *Managing Ada projects Using software engineering*. TPR, Scarsdale, New York. 246 pp. (ISBN: 0-8306-0290-9; $34.95/trade) Phone: 800-822-8138

Describes some of the practical aspects of developing a flawless project in Ada.

_____. (1991). *Software engineering: methods, management, and CASE tools*. TPR (a division of McGraw-Hill), New York. (ISBN: 0-8306-3442-8) hone: 800/822-8138

Software Productivity Consortium. (1989). *Ada quality and style: guidelines for professional programmers*. Van Nostrand Reinhold, New York. 230 pp. (ISBN: 0-442-23805-3; $26.95/paper)

Provides a set of specific guidelines for using the powerful features of Ada in a disciplined manner. Consists of concise statements of the principles that should be followed, and the rationale for each guideline.

Sommerville, I. and Morrison, R. (1987). *Developing large software systems with Ada*. International Computer Science Series, Addison-Wesley, Reading, Massachusetts. (ISBN: 0-201-14227-9; $26.95/paper text)

Stein, D. (1985). *Ada: a life and legacy*. MIT Press, Cambridge, Massachusetts. 321 pp. (ISBN: 0-262-19242-X; $30.00/Trade) (ISBN: 0-262-69116-7; $10.95)

Presents the view that Ada Byron's mathematical and scientific achievements have been exaggerated.

Stratford-Collins, M. J. (1982). *Ada: a programmer's conversion course*. Ellis Horwood Series in Computers & Their Applications, John Wiley & Sons, New York. (ISBN: 0-470-27332-1; $56.95/trade)

Tafvelin, S., ed. (1987). *Ada components: libraries and tools*. Ada Companion Series, Cambridge University Press, Cambridge. 288 pp. (ISBN: 0-521-34636-3; $44.50/trade)

Comprises the proceedings of the international conference organized by Ada Europe with the support of the Commission of the European Communities and the collaboration of SIGAda.

Tedd, M., et al. (1984). *Ada for multi-microprocessors*. Ada Companion Series, Cambridge University Press, Cambridge. 208 pp. (ISBN: 0-521-301033; $4450/trade)

Addresses those problems of distributed systems that arise out of the nature of Ada and support environments. Discusses the issues of how to construct

and run an Ada program for a valuable target configuration of several microcomputers, interconnected through shared memories multi-access busses, local area networks, and end-to-end lines.

Texel, P. (1986). *Introduction to Ada: packages for programmers*. Wadsworth Publishing, Belmont, California. 441 pp. (ISBN: 0-534-06348-9; out of print)

Provides a guide to Ada that contains complete packages, I/O facilities and sample programs, emphasizing top-down design throughout.

Toole, B. A. *Ada, the enchantress of numbers: a selection from the letters of Lord Byron's daughter and her description of the first computer*. Strawberry Press, New York. (ISBN: 0-912647-09-4; 29.95)

The author states that she selected the letters in such a way to enable the reader to follow a loose story line of Lady Ada Lovelace's life. In her letters, Ada describes her thoughts of the first computer, and Ms. Toole relates these descriptions to the modern software language, Ada.

Tremblay, J., et al. (1990). *Programming in Ada*. McGraw-Hill, New York. 489 pp. (ISBN: 0-07-065180-9; $24.60/paper text)

Explains computer science concepts in an algorithmic framework, with a strong emphasis on problem solving and solution development.

Uhl, J. (1982). *An attribute grammar for the semantic analysis of Ada*. Lecture Notes in Computer Science Series, 139, Springer-Verlag, Berlin. (out of print)

Unger, B. (1984). *Simulation software & Ada*, SCS Simulation Series. (ISBN: 0-911801-03-0; $16.00/paper)

U.S. Department of Defense. (1983). *The programming language Ada: a reference manual: proposed standard document*. Lecture Notes in Computer Sciences, 106, Springer-Verlag, Berlin. (ISBN: 0-387-10693-6; $19.00/paper)

Vasilescu, E. N. (1987). *Ada programming with applications*. Allyn and Bacon, Newton, California. 539 pp. (out of print)

Offers a bottom-up approach to commercial and business uses of Ada emphasizing the features of Ada that are most like those of traditional languages.

Vasilescu, E. M. (1986). *Ada programming*. Allyn and Bacon, Newton, California. (out of print)

Volper, D. and Katz, M. D. (1990). *Introduction to programming Using Ada*. Prentice-Hall, Englewood Cliffs, New Jersey. 650 pp. (ISBN: 0-13-493529-2; $30.00)

Uses the spiral approach as the presentation methodology in this introductory course in Ada programming.

Wallace, R. H. (1986). *Practitioner's guide to Ada*. McGraw-Hill, New York. 373 pp. (ISBN: 0-07-067922-3; $39.95)

Discusses the issues to be considered when making the transition to Ada on selecting toolsets, and on using the language effectively. Covers the following: Ada as a language; Ada Oriented Development Environments; Ada oriented design methodologies; Ada policies and standards; Ada products and vendors; sources of Ada-related information; making the transition to Ada and other uses of Ada.

Wallach, Y. (1990). *Parallel processing & Ada*. Prentice-Hall, Englewood Cliffs, New Jersey. (ISBN: 0-13-650789-1; $54.00/casebound)

Wallis, P. J. (1986). *Ada: managing the transition*. Ada Companion Series, Cambridge University Press, Cambridge. (ISBN: 0-521-33091-2; $44.50/trade)

Wallis, P. J., ed. *Ada software tools interfaces*, Lecture Notes in Computer Science Series, 180. (ISBN: 0-387-13878-1; $16.00/paper)

Watt, D. A., et al. (1987). *Ada language and methodology*. Prentice-Hall, Englewood Cliffs, New Jersey. 515 pp. (ISBN: 0-13-004078-9; $37.00/paper)

Covers the Ada language in detail and introduces program methodologies appropriate for use with Ada. Discusses the following topics: 1) covers a subset of Ada broadly comparable with most other programming languages; 2) introduces the features of Ada that make it suitable for the construction of large programs; 3) completes the treatment of the data types of Ada; 4) concludes the treatment of program structures.

Wegner, P. (1980). *Programming with Ada: an introduction by means of graduated examples*. Prentice-Hall, Englewood Cliffs, New Jersey. (out of print)

Wiener, R. S. and Sincovec, R. F. (1983). *Programming in Ada*. John Wiley & Sons, New York. 345 pp. (out of print

Describes the major features of the Ada programming language covering basic control and data structures associated with Ada, and powerful advanced features that differentiate it from previous programming languages.

_____. (1984). *Software engineering with Modula-2 & Ada*. John Wiley & Sons, New York. (ISBN: 0-471-89014-6; $51.95/text)

Winters, J. (1987). *Power programming with Ada for the IBM PC*. TAB Books, Blue Ridge Summit, Pennsylvania. 207 pp. (ISBN: 0-8306-2902-5; $16.95/paper) (ISBN 0-8306-7902-2; $24.95/trade)

Analyses programs in Ada for personal computers. Written for the beginning Ada programmer in a style very easy to read and follow.

Young, S. J. (1983). *An introduction to Ada.* John Wiley and Sons, New York. 400 pp. (out of print)

Introduces the programming language and explains the underlying program design methodology, illustrated with examples.

Appendix A
DoD Directive 3405.1

This appendix contains the text of DoD Directive 3405.1 issued on April 2, 1987 which specifies computer programming language policy for the DoD.

Department of Defense
DIRECTIVE

April 2, 1987
NUMBER 3405.1

ASD(C)

SUBJECT: Computer Programming Language Policy

References: (a) DoD Instruction 5000.31, "Interim List of DoD Approved Higher Order Programming Languages (HOL)," November 24,1976 (hereby canceled)
(b) DoD Directive 7740.1, "DoD Information Resources Management Program," June 20, 1983
(c) DoD Directive 5000.1, "Major System Acquisitions," March 12, 1986
(d) DoD Directive 5000.29, "Management of Computer Resources in Major Defense Systems," April 26, 1976
(e) through (j), see enclosure 1

A. PURPOSE

This Directive supersedes reference (a) and supports references (b) and (c) by establishing policy for computer programming languages used for the development and support of all DoD software.

B. APPLICABILITY AND SCOPE

This Directive:

1. Applies to the Office of the Secretary of Defense (OSD), the Military Departments (including the National Guard and Reserve), the Organization of the Joint Chiefs of Staff (OJCS), the Unified and Specified Commands, the Inspector General of the Department of Defense (IG, DoD), the Defense Agencies, and nonappropriated fund activities (hereafter referred to collectively as "DoD Components").

2. Covers all computer resources managed under reference (d) or DoD Directive 7920.1 (reference (e)).

3. Need not be applied retroactively to systems that have entered full-scale development or have passed Milestone II of references (c) and (e), and for which a documented language commitment was made in compliance with previous policy.

C. DEFINITIONS

Special terms used in this Directive are explained in enclosure 2; otherwise, refer to the "American National Dictionary for Information Processing Systems" (reference (f)).

D. POLICY

It is DoD policy to:

1. Satisfy functional requirements, enhance mission performance, and provide operational support through the use of modern software concepts,

advanced software technology, software life-cycle support tools, and standard programming languages.

2. Achieve improvements in DoD software management through the implementation of processes for control of the use of higher order languages, including specification of standards and waiver procedures.

3. Limit the number of programming languages used within the Department of Defense to facilitate achievement of the goal of transition to the use of Ada* (reference (g)) for DoD software development.

a. The Ada programming language shall be the single, common, computer programming language for Defense computer resources used in intelligence systems, for the command and control of military forces, or as an integral part of a weapon system. Programming languages other than Ada that were authorized and being used in full-scale development may continue to be used through deployment and for software maintenance, but not for major software upgrades.

b. Ada shall be used for all other applications, except when the use of another approved higher order language is more cost-effective over the application's life-cycle, in keeping with the long-range goal of establishing Ada as the primary DoD higher order language (HOL).

c. When Ada is not used, only the other standard higher order programming languages shown in enclosure 3 shall be used to meet custom-developed procedural language programming requirements. The use of specific HOL's shall be based on capabilities of the language to meet system requirements. Guidance in selecting the appropriate HOL to use is provided in NBS Special Publication 500-117 (reference (h)).

4. Prefer, based on an analysis of the life-cycle costs and impact, use of:

a. Off-the-shelf application packages and advanced software technology.

b. Ada-based software and tools.

c. Approved standard HOL's.

5. Consider the potential impact on competition for future software and/or hardware enhancements or replacement when selecting Defense, public domain, or commercially available software packages, or advanced software technology.

6. Use life-cycle management practices, as required by DoD Directive 7920.1 (reference (e)) and DoD Directive 5000.29 (reference (d)), for the development, support, and use of software, whether custom-developed or commercially acquired.

7. Reduce software obsolescence and the cost of software maintenance through use of approved programming languages and appropriate advanced software technology during all phases of the software life-cycle.

E. RESPONSIBILITIES

1. The Assistant Secretary of Defense (Comptroller) (ASD(C)) and the Under Secretary of Defense (Acquisition) (USD(A)) shall jointly:

a. Ensure that the policy and procedures in this Directive are implemented.

b. Assign responsibility to a specific DoD Component to act as the DoD language-control agent for each DoD-approved standard HOL.

c. Process nominations for changes to the list of approved HOL's.

2. The Assistant Secretary of Defense (Comptroller) shall:

a. For automated information systems, establish programs, as appropriate, for the enhancement of the software engineering process and the transition of such technology from the marketplace and research programs to application within general purpose automated data processing systems.

b. Define research and development requirements for automated information systems after consultation with DoD Components and provide such requirements to USD(A) for inclusion in their research and development program.

3. The Under Secretary of Defense (Acquisition) shall:

a. Establish and support a software and information technology research and development program that is responsive to the identified needs.

b. Manage the DoD Ada program and maintain and Ada Joint Program Office (AJPO) to oversee the maintenance of the Ada language and the insertion of Ada-related technology into the Department of Defense.

c. Establish research programs, as appropriate, for the enhancement of software engineering technology and transferring such technology to use in intelligence systems and systems for the command and control of military forces, and to computer resources that are an integral part of a weapon system.

4. The Head of Each DoD Component shall:

a. Implement and execute internal procedures consistent with the policy and procedures in this Directive.

b. Designate a language-control agent for each approved HOL for which the DoD Component is assigned responsibility and ensure compliance with the procedures in enclosure 4.

c. Institute a process for granting waivers to the use of approved HOL's in accordance with section F., below.

d. Specifically address in the Component's overall computer resources planning process:

(1) The use of appropriate advanced software technology for developing new applications and technological upgrades of existing systems.

(2) The current use of assembly languages, nonstandard HOL's, vendor extensions, and enhancements of standard HOL's, and actions taken to ensure that such use is minimized.

e. Establish a program for evaluating, prototyping, and inserting advanced software technology into the development, modification, and maintenance process, and hold operational software managers accountable for

investment in and migration to advanced software technology for their particular environment.

f. Establish and maintain training, education, and career development programs that will ensure that DoD personnel are fully able to use new advanced software technologies.

F. WAIVER PROCEDURES

1. Waivers to the policy in subsection D.3., above, shall be strictly controlled and closely reviewed. Authority for issuing waivers is delegated to each DoD Component.

a. Each proposed waiver shall contain full justification and shall, at a minimum, include a life-cycle cost analysis and a risk analysis that addresses technical performance and schedule impact. Each waiver granted by the DoD Component shall apply to only one system or subsystem.

b. Justification for granted waivers shall be provided to USD(A) or ASD(C) within the scope of their individual responsibilities, as periodically requested for review.

2. A waiver NEED NOT be obtained for use of:

a. Commercially available off-the-shelf applications software that is not modified or maintained by the Department of Defense.

b. Commercially available off-the-shelf advanced software technology that is not modified or maintained by the Department of Defense.

c. Computer programming languages required to implement vendor-provided updates to commercially supplied off-the-shelf software. Use of such languages shall be restricted to implementing the vendor updates.

3. A waiver IS REQUIRED for use of unmodified Defense or public domain software that does not conform to the language requirements of subsection D.3., above. Maintenance of the software shall be specifically addressed in the waiver request to include life-cycle maintenance costs and the availability of source codes and necessary software tools.

G. EFFECTIVE DATE AND IMPLEMENTATION

This Directive is effective immediately. Forward one copy of implementing documents to the Assistant Secretary of Defense (Comptroller) and one copy to the Under Secretary of Defense (Acquisition) within 120 days.

William H. Taft, IV
Deputy Secretary of Defense

Enclosures - 4
1. References
2. Special Terms and Definitions
3. DoD-Approved Higher Order Programming Languages
4. Procedures for Controlling Higher Order Languages (HOL)

*Ada is a Registered Trademark of the U.S. Government (Ada Joint Program Office).

(e) DoD Directive 7920.1, "Life Cycle Management of Automated Information Systems (AIS)," October 17, 1978
(f) National Bureau of Standards (NBS) FIPS Publication 11-2, "American National Dictionary for Information Processing Systems," May 9, 1983
(g) ANSI/MIL-STD-1815A-1983, "Ada Programming Language," February 1983
(h) National Bureau of Standards Special Publication 500-117, "Selection and Use of General Purpose Programming Languages," October 1984
(i) DoD 4120.3-M, "Defense Standardization and Specification Program Policies Procedures and Instructions," August 1978, authorized by DoD Directive 4120.3, February 10, 1979
(j) DoD Directive 5010.19, "Configuration Management," May 1, 1979

SPECIAL TERMS AND DEFINITIONS

1. **Advanced Software Technology.** Software tools, life-cycle support environments (including program support environments), nonprocedural languages, modern data base management systems, and other technologies that provide improvements in productivity, usability, maintainability, portability, etc., over those capabilities commonly in use.

2. **Automated Information Systems.** A collection of functional user and automatic data processing personnel, procedures, and equipment (including automatic data processing equipment(ADPE)) that is designed, built, operated, and maintained to collect, record, process, store, retrieve, and display information.

3. **Major Software Upgrade.** Redesign or addition of more than one-third of the software.

DoD-Approved Higher Order Programming Languages

Language	Standard Number	DoD Control Agent	Industry Control Agent
Ada	ANSI/MIL-STD-1815A-1983 (FIPS 119)	Ada Joint Program Office	ANSI
C/ATLAS	IEEE STD 716-1985	Navy	IEEE
COBOL	ANSI X3.23-1985 (FIPS 21-2)	Air Force	ANSI
CMS-2M	NAVSEA 0967LP-598-2210-1982	Navy	N/A
CMS-2Y	NAVSEA Manual M-5049, M-5045 M-5044-1981	Navy	N/A
FORTRAN	ANSI X3.9-1978 (FIPS 69-1)	Air Force	ANSI
JOVIAL (J73)	MIL-STD-1589C (USAF)	Air Force	N/A
Minimal BASIC	ANSI X3.60-1978 (FIPS 68-1)	Air Force	ANSI
PASCAL	ANSI/IEEE 770X3.97-1983 (FIPS 109)	Air Force	ANSI
SPL/1	SPL/1 Language Reference Manual, Intermetrics Report No. 172-1	Navy	N/A

Note. See NBS Special Publication 500-117 (reference (h)).

PROCEDURES FOR CONTROLLING HIGHER ORDER LANGUAGES (HOL)

1. All Ada compilers that are used for creation of software to be delivered to or maintained by the Government shall be formally validated in accordance with procedures and guidelines set by the AJPO.

2. Each DoD-approved HOL shall be assigned to a DoD language-control agent, as shown in enclosure 3, who shall:

a. Have the authority and responsibility for proper support of all language-control activities needed to provide for necessary modification and improvement of the assigned HOL. The agent shall operate in accordance with DoD 4120.3-M (reference (i)).

b. Provide configuration control for DoD HOL's in accordance with DoD Directive 5010.19 (reference (j)). For HOL's controlled under industry (e.g., Institute for Electrical and Electronic Engineers or American National Standards Institute) procedures, the agent shall represent the Department of Defense to the controlling body.

c. Maintain a single standard definition of the assigned HOL and make this definition document available as a Federal, DoD, military, or adopted industry standard. The agent shall also gather and disseminate appropriate information regarding use of the HOL, its compilers, interpreters, and associated tools.

3. A DoD Component may nominate a language for removal from the list of approved languages by submitting a justification document, which presents the rationale for the proposed deletion and an impact analysis, to the ASD(C), who will coordinate it with USD(A).

4. A DoD Component may also nominate a language for inclusion on the list of approved languages by submitting a justification document to the ASD(C), who will coordinate it with USD(A). The justification document shall include the following:

a. A detailed rationale for using the language, including how the candidate language meets specific DoD requirements that are not satisfied by the approved languages.

b. A description of the language and the environment and a detailed unambiguous specification of the language.

c. An economic analysis of the impact of the language over its expected life-cycle.

d. A detailed plan for implementing and supporting the language, including identification of the DoD Component that will accept designation as control agent for the language.

Appendix B
HQDA LTR 25-90-1

This appendix contains the text of HQDA LTR 25-90-1 issued on 1 July 1990 which provides guidance on implementing use of the Ada programming language with the Department of the Army.

DEPARTMENT OF THE ARMY
WASHINGTON, D.C. 20310

*HQDA LTR 25-90-1
*This letter supersedes HQDA Letter 25-88-5, 21 June 1988.

SAIS-ADO (14 May 1990) 16 July 1990

Expires 16 July 1992

SUBJECT: Implementation of the Ada Programming Language

SEE DISTRIBUTION

1. Purpose. This letter amplifies Army policy and guidelines for implementing the Ad programming language as required by DOD Directives 3402.1 and 3405.2

2. References. Related publications are listed below.

a. DOD Directive 3405.1, 2 April 1987, Computer Programming Language Policy.

b. DOD Directive 3405.2, 30 March 1987, Use of Ada in Weapon Systems.

c. ANSI/MIL-STD-1815A-1983, Ada Programming Language, 22 January 1983.

d. HQDA message, 031309Z August 1987, SQL Relational Data Base Language Standard.

3. Explanation of abbreviations. Abbreviations used in this letter are explained in the glossary.

4. Applicability and scope. This letter applies to all computer resources used to develop, modify, maintain, or support Army software. These resources include but are not limited to automated information systems, intelligence systems, tactical systems, and weapon systems that have information resources such as computers as part of, or embedded in, the host system. They also include but are not limited to systems developed by or for major commands, program executive officers/program managers, central design activities, combat development facilities, and laboratories. Except in instances noted in paragraph 6a, this policy needs not be applied retroactively to systems that have entered full-scale development or deployment phases of the life cycle, or for which a waiver has been approved by Headquarters, Department of the Army (HQDA).

5. Responsibilities

a. The Director of Information Systems for Command, Control,

Communications, and Computers (DISC4) will--

(1) Act as the Army Ada Executive Official and serve as the Army focal point for all Ada program activities. The DISC4 will review and approve all requests for exceptions or waivers

(2) Develop and execute the policy and plans necessary to ensure successful Army-wide transition to, and implementation of, the Ada language and its associated technology, including processes for software engineering and software engineering project management.

b. Major Army command (MACOM) commanders and/or program executive officers (PEOs) will--

(1) Develop appropriate policy to support an Ada implementation plan.

(2) Develop, submit, maintain, and execute an Ada implementation plan in the format shown in Appendix A. The implementation plan will be in two parts (systems and organizational) and will be submitted/updated on an annual basis. The systems portion will address all major systems (such as mission critical computer resources (MCCR), Standard Army Management Information System (STAMIS), command standard, command unique, and multi-user/location) and those systems that provide input to or receive output from these systems. This portion will also include a schedule for transition to Ada, as appropriate. The organizational portion will address planning, training, and support available for migrating to Ada technology.

(3) Maintain, as a PEO, an Ada implementation plan for those systems under their purview and ensure that assigned program/project/product managers (PMs) implement Ada in accordance with Army policy.

(4) Ensure that software designers/developers are fully trained in the use of the Ada language, technology, and software engineering processes, with particular emphasis on developing components that are tested, validated, and documented for inclusion in reuse libraries.

(5) Ensure that military and government civilian personnel in all software skill groups are aware of new advanced software technologies for possible implementation.

(6) Develop procedures and guidelines that address good software engineering principles that, as a minimum, address software reuse, portability, and management controls.

c. Heads of HQDA activities will develop, maintain, and submit implementation plans to the Department of the Army Information Manager (DAIM) for execution. The HQDA staff

agencies and their field operating agencies (FOAs) are considered collectively as a MACOM for execution of this letter.

6. Policy.

a. The Ada programming language as defined in ANSI/MIL-STD-1815A-1983 is the single, common, high order computer programming language for all computer resources used in the Army unless another language is mandated by a higher level directive. Approvals to use another approved standard high order language (HOL), as defined in DOD Directive 3405.1, will only be granted when the use of the other language is estimated/calculated to be more cost effective or more operationally effective over the applications' life cycle. Programming languages other than Ada that were authorized and are being used in full-scale development of these systems may continue to be used through deployment and for software maintenance. In those specific instances where Army systems must interface with non-Department of Defense (DOD) agencies, such as the Central Intelligence Agency (CIA) and Federal Bureau of Investigation (FBI), Ada is preferred but not required. Existing software need not be rewritten in Ada solely for the purpose of converting to Ada. All systems, however, will transition to Ada when the next hardware/software upgrade requires modification of more than one-third of the existing code over the system life cycle, unless a waiver is obtained.

b. All requests for exceptions to use another approved HOL will have fully documented rationale. The requests will address technical feasibility and life-cycle cost analysis or cite the applicable higher level directive.

c. When software components for Army systems are being acquired and/or developed, good software engineering principles will be exercised to facilitate the use of Ada. The approach to acquiring and/or developing software components will be based on an analysis of life-cycle costs and operational efficiency. Major considerations should be:

(1) The use or modification of existing Ada software.

(2) The use of off-the-shelf software and advanced software technology, implemented in other than Ada, for which no modification or Government maintenance is required. Advanced software technology includes software tools, life-cycle support environments, nonprocedural languages, and modern database management systems (DBMSs) that provide improvements in productivity, usability, maintainability, and portability. A waiver is not required for non-developmental item (NDI) software application packages and advanced software technology that are not modified by or for DOD. In those instances where existing software requires modification to ensure the total system meets requirements, a waiver is required if more than one-third of the source code is being changed and the changes are not written in

Ada.

(a) Regarding the use of fourth generation languages (4GLs), the following apply:

1 The approved ad hoc query and Database Management System interface language for Army systems is the Structured Query Language (SQL), Federal Information Processing Standard (FIPS) 127. In accordance with HQDA message, 031309Z August 1987, SQL will be used for relational databases as the interface between programs and the supporting DBMS. A waiver is not required for any system using an SQL-compliant DBMS in conjunction with Ada.

2 Non-SQL-compliant 4GLs may be used without the requirement for a waiver to develop prototypes during requirements definition and in short term/ad hoc applications (less than 3 years' useful life). In no case will a non-Ada prototype be fielded during system implementation nor will an ad hoc application exceed the time limitation without an approved Ada waiver.

(b) With the exception noted in subparagraph 2, above, 4GLs will not replace the requirement for, or the use of , Ada.

(3) The development of new Ada software. If source code generators are used in the development of Ada software, they must produce an Ada source code that complies with ANSI/MIL-STD-1815A-1983.

(4) The impact on certain critical processes that currently cannot be performed efficiently in an HOL due to timing and/or sizing constraints. These functions, requiring very fast or tightly controlled computer processing, are more appropriately written at the machine level (for example, micro-code/assembly language). In such instances, a waiver is not required if the ration of non-Ada source code to Ada source code (terminal semicolon count) doer not exceed 15 percent. An Ada waiver is required if the total machine leval code exceeds 10,000 lines.

(5) The requirement that projects use a validated Ada compiler, as defined by the Ada Joint Program Office (AJPO) Ada Compiler Validation Procedures, at the start of formal testing. Providing no changes are made to the compiler, it may be used for the balance of the project's life cycle even though its associated validation certificate may have expired. If the compiler is altered, then a validation is necessary.

d. If system requirements cannot be satisfied by paragraph 6c, then a waiver approval is required from HQDA, DISC4 (SAIS-ADO) to:

(1) Develop Army software in another computer language.

(2) Acquire off-the-shelf software, implemented in other than Ada, which requires Government maintenance or modification of more than one-third of the total system software.

(3) Implement a system/subsystem in a 4GL.

(4) Develop an Army system, with severe time and/or size constraints, in which the machine level to Ada source code ratio exceeds 15 percent or the total machine language code exceeds 10,000 lines.

e. In all instances, however, anyone requesting a waiver must demonstrate that the software strategy is more cost-effective or more operationally effective over the system life and must include a statement of maintainability from the responsible software maintainer.

7. Waivers

a. with the exception noted in paragraph 6c, a waiver must be obtained to develop any non-Ada software.

b. Justification must address the following issues:

(1) The waiver request will provide adequate technical description to address limitations, documentation, portability, maintainability, and usability of the proposed software language or package.

(2) The waiver request will provide complete life-cycle economic rationale for both Ada and the requested language. For tactical systems, intelligence systems, and embedded weapon systems, the waiver request must also include a risk analysis that addresses technical performance and schedule impact.

c. A waiver request for all new initiatives must be approved prior to Milestone I approval. Prototypes may use advanced software technology (such as 4GLs) in accordance with paragraph 6c(2). However, sunk costs for a non-Ada prototype will not be considered justification for a waiver.

d. An existing system undergoing modification, as defined in paragraph 6a, must have received a waiver prior to system redevelopment regardless of the cost.

e. The long-term costs of supporting programmers, environments, and software code for diverse languages will be closely scrutinized when waivers are considered.

f. Waivers will apply only to the specified system or subsystem identified.

g. Waiver processing procedures are as follows:

(1) When there is a PEO/PM structure in place the waiver request will be submitted from the PM through the PEO (and MACOM if appropriate) to the DISC4 (SAIS-ADO). All requests will have a statement of maintainability from the applicable Life Cycle Software Engineering Center (LCSEC), Software Development Center (SDC), or Government software engineering organization that will be responsible for maintenance of the system.

(2) When there is no PEO/PM structure in place, the waiver request will be submitted through the cognizant software support center through the MACOM to the DISC4.

(3) Waivers may be denied at any level but can only be approved by the DISC4.

8. Effective date and implementation. This directive is effective immediately. MACOM Deputy Chiefs of Staff for Information Management (DCSIMs) and PEOs will forward a copy of their consolidated initial/updated Ada implementation plans (appendix A) to HQDA (SAIS-ADO) Washington, DC 20310-0107 by 1 October 1990. Updates to the Ada Implementation Plan are due on 1 October annually.

Appendix A

Ada Systems Implementation Plan

1. MACOM/Installation
2. POC Name/Telephone
3. System Name and Acronym
4. Current Life Cycle Management Phase
5. System Fielding Date
6. ADP Hardware Used
7. Computer Operating System
8. Software Languages Used by Subsystem(s) Including Support Software
   a. Name
   b. Lines of Code
   c. Percent of System
9. Database Management System (DBMS) Used
10. DBMS Interface Technique
11. Program Design Language/Implementation Language
12. Project Approval Documentation (Computer Resources Management Plan (CRMP), Acquisition Plans, and so on, with status)
13. Date Waiver Approved (if applicable)
14. Ada Transition DAtes (start and finish)
15. Planned Upgrade Date(s) (for either hardware or software)
16. Maintenance Responsibility
17. System Documentation Standard Used
18. Transition to Ada (narrative explanation)

Ada Organizational Implementation Plan

1. Human Resources
   a Education and Training of Incumbent Management and Technical Personnel
   b. Accession/Recruitment of Qualified Ada Personnel
   c. Ada Support Contractor
2. Resources
   a. Financial
   b. Technical Status
      (1) Ada Support Environment (including interface DOD Standard 1838)

      (2) Interface to Operational Environment
         (a) DBMS
         (b) Operating System
         (c) Graphics Support

Glossary

Abbreviations

ADP ---------- automatic data processing

AJPO ---------- Ada Joint Program Office

ANSI ---------- American National Standards Institute

CIA ---------- Central Intelligence Agency

CRMP ---------- Computer Resources Management Plan

DAIM ---------- Department of the Army Information Manager

DBMS ---------- Database Management System

DCSIM ---------- Deputy Chief of Staff for Information Management

DISC4 ---------- Director of Information Systems for Command Control, Communications, and Computers

DOD ---------- Department of Defense

FBI ---------- Federal Bureau of Investigation

4GL ---------- fourth-generation language

FIPS ---------- Federal Information Processing Standards

FOA ---------- field operating agency

HOL ---------- high order language

HQDA ---------- Headquarters, Department of the Army

LCSEC ---------- Life Cycle Software Engineering Center

MACOM ---------- major Army command

MCCR ---------- mission critical computer resources

NDI ---------- non-developmental item

PEO ---------- program executive officer

PM ---------- program/project/product manager

POC ---------- point of contact

SDC ---------- Software Development Center

SQL ---------- structured query language

STAMIS ---------- Standard Army Management Information System

BY ORDER OF THE SECRETARY OF THE ARMY:

MILTON H. HAMILTON
Administrative Assistant to the
Secretary of the Army

DISTRIBUTION:

HQDA (DACS-ZA)
HQDA (SAFM)
HQDA (SARD)
HQDA (SAAA)
HQDA (SAIS-ZA)
HQDA (SAIG-ZA)
HQDA (DAMI-ZA)
HQDA (DALO-ZA)
HQDA (DAMO-ZA)
HQDA (DAPE-ZA)
HQDA (DAEN-ZA)
HQDA (DASG-ZA)
HQDA (NGB-ZA)
HQDA (DAAR-ZA)
HQDA (DAJA-ZA)
HQDA (DACH-ZA)

COMMANDER-IN-CHIEF
  U.S. ARMY, EUROPE AND SEVENTH ARMY
COMMANDERS
  EIGHTH U.S. ARMY
  FORCES COMMAND
  U.S. ARMY MATERIEL COMMAND
  U.S. ARMY TRAINING AND DOCTRINE COMMAND
  U.S. ARMY INFORMATION SYSTEMS COMMAND
  U.S. ARMY JAPAN
  U.S. ARMY WESTERN COMMAND
  MILITARY TRAFFIC MANAGEMENT COMMAND
  U.S. ARMY CRIMINAL INVESTIGATION COMMAND
  U.S. ARMY HEALTH SERVICES COMMAND
  U.S. ARMY SOUTH
SUPERINTENDENT, U.S. MILITARY ACADEMY

CF:
HQDA (SASA)
HQDA (SAUS)
HQDA (SACW)
HQDA (SAIL)
HQDA (SAMR)
HQDA (SAGC)
HQDA (SAAG-ZA)
HQDA (SALL)
HQDA (SAPA)
HQDA (SADBU)
COMMANDERS
  U.S. ARMY MILITARY DISTRICT OF WASHINGTON
  U.S. ARMY RECRUITING COMMAND

DIRECTOR, DEFENSE LOGISTICS AGENCY
PROGRAM EXECUTIVE OFFICERS
  COMMUNICATIONS
  STANDARD ARMY MANAGEMENT INFORMATION SYSTEM
  COMMAND AND CONTROL SYSTEMS
  STRATEGIC INFORMATION SYSTEMS
  ARMAMENTS
  CHEMICAL/NUCLEAR
  ARMORED SYSTEMS MODERNIZATION
  AVIATION
  COMBAT SUPPORT
  FIRE SUPPORT
  AIR DEFENSE
  INTELLIGENCE AND ELECTRONIC WARFARE
  STRATEGIC DEFENSE

Appendix C
The Congressional Ada Mandate[1]

This mandate was first included in the fiscal year 1991 appropriations bill (H.R. 5803) for the Department of Defense. That bill was signed by the President on November 5, 1990, and became Public Law 101-511. In the FY 1991 act, the section number and wording were:

> Sec. 8092. Notwithstanding any other provisions of law, after June 1, 1991, where cost effective, all Department of Defense software shall be written in the programming language Ada, in the absence of special exemption by an official designated by the Secretary of Defense.

Similar wording was also included in the FY 1992 appropriations bill, Public Law 102-172, enacted November 26, 1991. There, the mandate can be found in Section 8073. The FY 1991 appropriations bill originated in the House as H.R. 5803. In the version reported out of the House and sent to the Senate, this section (originally numbered Sec. 8084) had not contained the proviso "where cost effective"; as amended by the Senate, the mandate was deleted entirely; when House and Senate conferees met to reconcile differences in the two versions, they restored the mandate with the "where cost effective" proviso. With this proviso, the mandate was then a part of the final version of the appropriations bill passed by both houses and signed by the President. The following appeared in House Report 101-822, which accompanied the original House-passed version of H.R. 5803.

Ada Programming Language. — The Department of Defense developed Ada to reduce the cost of development and support of software systems written in the hundreds of languages used by the DOD through the early 1980s. Beside the training economies of scale arising from a common language, Ada enables software cost reduction in several other ways: (1) its constructs have been chosen to be building blocks for disciplined software engineering; (2) its internal checking inhibits errors in large systems lying beyond the feasibility of manual checking; and (3) its separation of software module interfaces from their implementations facilitates and encourages reuse of already-built and tested program parts. While each of these advantages is important, Ada's encouragement of software engineering is fundamental. Software practitioners increasingly believe the application of engineering disciplines is the only currently-feasible avenue toward controlling unbridled software cost escalation in ever-larger and more complex systems. In March, 1987, the Deputy Secretary of Defense mandated use of Ada in DOD weapons systems and strongly recommended it for other DOD applications. This mandate has stimulated the development of commercially-available Ada compilers and support tools that are fully responsive to almost all DOD requirements. However, there are still too many other languages being used in the DOD, and thus the cost benefits of Ada are being substantially delayed. Therefore, the Committee has included a new general provision, Section 8084, that enforces the DOD policy to make use of Ada mandatory. It will remove any doubt of full DOD transition to Ada, particularly in other than weapons systems applications. It will stimulate DOD to move forward quickly with Ada-based software engineering education and cataloguing/reuse systems. In addition, U.S. and commercial users have already expanded tremendously the use of Ada and Ada-related technology. The DOD, by extending its Ada mandate, can leverage off these commercial advances. Navy Ada is considered to be the same as Ada for the purposes of this legislation, and the term Ada is otherwise defined by ANSI/MIL-STD-1815. The Committee envisions that the Office of the Secretary of Defense will administer the general provision in a manner that prevents disruption to weapon systems that are well into development. The Committee directs that applications using or currently planning to use the Enhanced Modular Signal Processor (EMSP) be exempted from mandatory use of Ada as a matter of policy.

Appendix D
Ada vs. C++

On July 9, 1991, the Air Force released to the public a report of a business case they conducted to determine under what circumstances a waiver to the DoD Ada requirement might be warranted for use of C++, particularly in DoD's Corporate Information Management (CIM) program. The report is titled, "Ada and C++: A Business Case Analysis."[1]

Since its release the report has received a great deal of publicity in various newspapers and journals. The information that follows was excerpted from remarks made by Mr. Lloyd K. Mosemann, II Deputy Assistant Secretary of the Air Force (Communications, Computers, and Logistics), at a press conference held July 9, 1991.

I. Introduction

There has never been any intention to question DoD's commitment to Ada, but only to identify when waivers for C++ might be warranted. This business case will support the development of DoD programming language policy for information systems and C3 systems.

I might say at the outset that language comparison is not merely a scientific issue: it evokes strong emotions as well, in that to a certain extent people adopt "favorite" languages for reasons other than purely dispassionate analysis, much as one might not be able to explain why he/she roots for the Chicago Cubs or drinks Coke rather than Pepsi. The task is also rendered difficult because there are yet no well-established and standard methods for conducting such

[1] Those who have an account with the Defense Technical Information Center (DTIC) may purchase "Ada and C++: A Business Case Analysis" from DTIC, Cameron Station, Alexandria, Virginia 2314, 703/274-7633; Order No. AD A253 087; Cost $20.82. All others may purchase it from the National Technical Information Service (NTIS), U.S. Department of Commerce, 5825 Port Royal Road, Springfield, Virginia 22161, 703/487-4600; Order No. AD A253 087; Cost $43.00.

comparisons. For these reasons, we endeavored to make our study as quantitative as possible, asking the best experts we could find to use a variety of methods that have historically been used for business analysis in such tasks. We felt that by using a variety of methods and comparing their results, we would avoid the skewing that might result from the sole use of a single method.

In our business case, therefore, several different approaches were undertaken to identify, from a business perspective, when the life cycle cost effectiveness of C++ might be greater than that of Ada.

- The first, conducted by the Institute for Defense Analyses (IDA), examined quantitatively the availability of tools and training for the two languages.
- The second, conducted by the Software Engineering Institute, applied to this problem a quantitative language selection methodology developed by IBM for the Federal Aviation Administration (FAA).
- The third, conducted by CTA, Inc., analyzed cost and cost analysis.
- And the fourth, conducted by the TRW Corporation, applied a standard cost model in depth to both languages for a typical information systems/C3 project (micro analysis).
- In addition, the Naval Postgraduate School (NPS) was asked to address the overall policy issue of Ada, particularly in the context of emerging fourth-generation language (4GL) software technology.

Each of the substudies reached the same conclusion: there are no compelling reasons to waive the Ada requirement to use C++.

The business case analysis was directed at information systems and C3 systems. However, there is no reason to believe the results would differ for computer programs embedded in weapons systems.

Let me now summarize for you the salient quantitative results of each study, and I think you will understand more fully how we arrived at our conclusion.

II. Substudy Results.

A. Tools, Environments, and Training: IDA Substudy.

The Institute for Defense Analyses (IDA) collected and analyzed information on the market availability of commercial- off-the-shelf products available from U.S. sources for Ada and C++ compilers, tools, education, and training. The study provided a large quantity of demographic data. For example, there are 28 companies located in the U.S. that have Ada compilers with currently validated status; 18 vendors offer C++ compilers. The Ada compiler vendors are more likely to have been in business five years or more. Ada "validation" is more rigorous than that of other high order languages: only Ada is monitored and approved for conformity to a standard, without supersets or subsets, by a

government-controlled process. By contrast, no validation or even a standard of any kind exists for C++, although a standard by 1994 is expected.

Both languages are supported on PCs and workstations. Ada is also supported on mainframes. Ada, but not C++, has cross compilation systems.

Ada is supported with program engineering tools. Compiler vendors provide a rich set. Code generators exist for Ada but none so far for C++. There is considerable variability among C++ products in language features supported and libraries provided.

Ada is taught in 43 states at 223 universities and 13 DoD installations. C++ is taught in four states at four universities and no DoD installations. There are more Ada than C++ courses available. The cost of training is about equal, but Ada course variety is wider.

B. Faceted IBM Language Selection Methodology: SEI Substudy.

The Federal Aviation Administration (FAA) contracted with IBM in the mid-1980s to evaluate high order languages for use on its Advanced Automation System (AAS) Program. In response, IBM developed a formal, quantitative faceted methodology comparing 48 language features (criteria) in six categories. This IBM study concluded that use of Ada was "in the ultimate best interest of the AAS program and its goals, and that warrants coping with the temporary risks/problems that loom large in the near term in order to reap the significant benefits/payoffs over the long term."

Using this same methodology for each of the 48 criteria, the Software Engineering Institute (SEI) evaluated Ada and C++ for application in information systems/C3 systems. The original FAA weighted scores for the six criteria categories were as shown in this matrix:

Category	Max.	Ada	C	Pascal	JOVIAL	FORTRAN
Capability	16.7	6.1	9.6	10.4	7.6	3.9
Efficiency	16.4	8.0	11.8	10.8	11.0	11.1
Availability/Reliability	22.6	21.5	11.6	14.5	15.6	10.3
Maintainability/Extensibility	17.4	14.0	10.2	12.2	6.8	8.3
Lifecycle cost	11.3	8.2	7.4	7.8	4.9	5.2
Risk	15.6	8.8	8.9	7.6	9.6	8.2
Total	100.0	76.6	59.5	63.3	55.5	47.0

The 1991 weighted scores for the six criteria categories were:

Category	Max.	Ada	C++
Capability	16.7	15.3	11.3
Efficiency	16.4	10.7	10.9
Availability/Reliability	22.6	19.1	12.6
Maintainability/Extensibility	17.4	13.6	11.4
Lifecycle cost	11.3	8.4	8.0
Risk	15.6	11.7	9.8
Total	100.0	78.8	63.9

In 1985 Ada was considered considerably more capable than C. Today, the SEI study found there is still a significant difference between Ada and C++, C's successor. The relative efficiency of Ada has improved markedly; Ada still scores significantly higher in availability/reliability; the Ada advantage in maintainability/extensibility persists; and from a position of parity in 1985, Ada has attained in 1991 a significant advantage over C++ in lowered risk.

An attachment lists numerous major Ada information systems/C3 systems. It is not widely appreciated that such extensive use is now being made of Ada: in fact, the Ada 9X Project Office reports that the U.S. Ada market, excluding training, services, and government research/development, now exceeds $1 billion.

C. Macro Cost Analysis: CTA Substudy.

CTA compiled and compared available productivity and cost data of Ada and C++. Much of their data comes from Reifer Consultants' extensive database, one of the best, largest, and most current programming language cost databases now available.

Average productivity across the four domains for which data exists (environment/tools, telecommunications, test (with simulators) and other) for both Ada and C++ projects is shown in this matrix. Note the productivity advantage for Ada:

	Productivity (SLOC/MM)	Number of Data Points
Norm (all languages)	183	543
Average (Ada)	210	153
Average (C++)	187	23
First project (Ada)	152	38
First project (C++)	161	7

The C++ project data reflected information on 23 projects taken from seven firms who had been using C++, Unix, and object-oriented techniques for over 2 years. All projects were new developments. Application size ranged from 25 to 500 KSLOCs (thousand source lines of code). Average size was about 100 KSLOC.

The average costs across the four domains for both Ada and C++ projects are shown in this matrix.

	Cost ($/SLOC)	Number of Data Points
Cost (all languages)	$70	543
Average (Ada)	65	153
Average (C++)	55	23

Typically, the Ada developments were performed in accordance with military standards and incorporated formal reviews, additional documentation, and additional engineering support activities such as formal quality assurance (QA) and configuration management (CM). Most C++ projects are commercial and do not extensively incorporate such activities. Additionally, on such projects developers are typically intimately involved with users, resulting in considerably less requirements engineering effort. Consequently, applications on which C++ is used are inherently less costly, so that the reported productivity rates are favorably skewed toward C++.

The average error rates across the four domains for both Ada and C++ projects were:

	Integration Error Rates (Errors/KSLOC)	FQT Error Rates (Errors/KSLOC)	Number of Data Points
Norm (all languages)	33	3	543
Average (Ada)	24	1	153
Average (C++)	31	3	23

The integration error rates include all errors caught in test from start of integration testing until completion of software Formal Qualification Test (FQT). The FQT error rate includes only those errors found during the FQT process.

A so-called "transition state analysis" performed by Reifer's group indicates that 26 of the 38 firms within the Ada database had successfully made the changeover to effective use of Ada, while none of the 7 firms in the C++ database had made the transition. Also, none of the 7 firms were fully using C++'s inheritance and other advanced features.

The standardization maturity of Ada was found by the CTA to be particularly important. While Ada has a firm and well policed standard, allowing neither supersets nor subsets, it will be years before a stable C++ language specification is established. New features are being considered for the latest standard C++ release. Vendors are likely to offer their own enhanced versions of C++ compilers and CASE tools, complicating portability and reuse.

Finally, the original arguments for establishing a single programming language for military applications were found to remain. Common training, tools, understanding, and standards simplify acquisition, support, and maintenance. The study concluded that after maturing for a decade, Ada's benefits have been proven for all application classes. Ada projects have reported 15% higher productivity with increased quality and double the average size. Normalizing these data to comparable size projects would result in an expected Ada productivity advantage of about 35%. Ada should be the near term language of choice. C++, the study felt, still needs significant maturing before it is a low risk solution for a large DoD application.

D. Micro Cost Analysis: TRW Substudy.

TRW performed a tradeoff analysis that generalized recent corporate cost analyses on a typical real-world information systems/C3 systems project. Their study defined a set of maximally independent criteria, judged each language with respect to those criteria, and then translated those judgments into cost impacts to emphasize the importance of each criterion from a lifecycle cost perspective. Results were translated into perturbations of Boehm's Ada COCOMO cost model.

Rankings of the two languages based on this analysis are shown in this matrix (0 = no support; 5 = excellent support), followed by a total score, a weighted sum of the rankings based on weights determined by an expert panel:

Category	Ada	C++
Reliable S/W Engineering	4.5	3.2
Maintainable S/W Engineering	4.4	3.2
Reusable S/W Engineering	4.1	3.8
Realtime S/W Engineering	4.1	2.8
Portable S/W Engineering	3.6	2.9
Runtime Performance	3.0	3.6
Compile-Time Performance	2.3	3.1
Multilingual Support	3.1	2.4
OOD/Abstraction Support	3.9	4.6
Program Support Environment	4.1	2.1
Readability	4.4	2.9
Writeability	3.4	3.5
Large Scale S/W Engineering	4.9	3.3
COTS S/W Integration	2.8	3.6
Precedent Experience	3.6	1.5
Popularity	2.8	4.0
Existing Skill Base	3.0	1.8
Acceptance	2.5	3.3
Total Score for Mgt Info Systems (Ada score is 23% higher)	1631	1324
Total Score for C3 Systems (Ada score is 24% higher)	1738	1401

The study concluded that both Ada and C++ represent improved vehicles for software engineering of higher quality products. Currently, C++ was estimated to be approximately 3 years behind Ada in its maturity and tool support. The case study used in this report (the Command Center Processing and Display System-- Replacement) demonstrated development cost advantages for Ada on the order of 35% and maintenance cost advantages for Ada on the order of 70% under today's technologies. In the far term (1994+), the study felt, this Ada advantage might erode to approximately a 10% advantage in development costs and 30% in maintenance costs for a typical development intensive system.

The study listed the primary strengths of Ada as its support for realtime domains and large scale program development. Its primary weaknesses are its compile-time and runtime efficiency. The primary strengths of C++ listed were its support for better object oriented design, support for COTS integration, and its compile-time and runtime efficiency. Its main weaknesses were identified as its support for reliability and large scale program development. In general, the study felt Ada's weaknesses to be solved by ever-increasing hardware performance and compiler technology advancement. C++ weaknesses, on the other hand, remain to be solved by advances in its support environment.

E. Ada Policy Issues: NPS Study.

Concurrently with the preparation of this Ada and C++ Business Case Analysis, the Naval Postgraduate School (NPS) reported on policy issues on the use of Ada for Management Information Systems. Their report, an analysis of the need to see Ada in a total and evolving context, is an important vision statement leading from Ada as the primary third-generation language (3GL) to its conception as the basis for evolving to higher levels of productivity in so-called 3 1/2 GL and 4GL environments.

Rather than concentrating on programming language selection, the NPS report focuses on and argues for needed advances in software development technology. In particular, the Report contends, while traditional factors such as programming language selection, better training, and computer-assisted software engineering (CASE) tools can enhance productivity modestly, a fundamental change in the software development paradigm will be necessary to achieve an order of magnitude gain. Such a gain is possible through use of 4GLs, languages that will ultimately enable the developer to define the complete design of an application entirely in the 4GL's own high-level specification language. The specification is then translated automatically by the 4GL into an executable program. When accompanied by a productive development environment, an evolutionary implementation methodology, and well trained development teams, the report asserts, 4GLs can provide a tenfold gain in productivity.

An intermediate step cited by the report in the movement to 4GLs is 3 1/2 GL programming, a term referring to the extensive use of CASE tools coupled with a high level of code reuse. The 3 1/2 GL approach requires a strong commitment to codifying and accrediting code modules, to the point where it becomes easier and more desirable to reuse code than to rewrite it.

Although experience with 4GLs has not yet been extensive (with existing experience limited largely to specific functional domains such as financial management and transaction processing), 4GLs are attractive for several reasons. One is their robustness under change: changes made to the application, for whatever reason, are made at the specification level and then re- translated automatically into executable code. Another is the facility with which they can be integrated into tightly knit and full-featured development environments. For these reasons, the report strongly recommends that the DoD discourage use of traditional 3GL programming and take bold steps to incorporate the 4GL paradigm.

Finally, the report recommends that, given the importance of Ada to DoD software, greater effort and funding should be provided for the key Ada initiatives: the Ada Technology Improvement Program, Ada 9X, and Ada education initiatives.

Two issues on 3 1/2 GLs and 4GLs related to this business case were outside the scope of the NPS report. The first of these is that, for the foreseeable future, state-of-the-art limitations will probably keep 4GLs from generating more than half the total code required by most applications. In such cases, where a substantial amount of 3GL programming will be required to complete application

development, use of a 3 1/2 GL approach, rather than a 4GL approach, is preferable.

Another issue outside the scope of the NPS Report was the evaluation of the relative merits of Ada and C++ as target (output) languages for 4GL application generators. However, as section V.C of the NPS report points out, a "standard, stable target language portable to a variety of hardware platforms" with good software reuse and interface definition capabilities is appealing. Although more study of the characteristics desired in 4GL target languages is warranted, the SEI and TRW substudies suggest no particular advantage of C++ over Ada in software reuse and interface definition, so there appears no reason to waive DoD's Ada requirement in favor of C++ as a target language for 4GLs.

III. Conclusions.

All four substudies which specifically compared Ada and C++ (IDA, SEI, CTA, TRW) report a significant near term Ada advantage over C++ for all categories of systems. This advantage could be eroded as C++ and its supporting environments mature over the next few years. On the other hand, as aggressive overseas Ada initiatives stimulate even wider domestic Ada interest, as Ada tools/environments further mature, and when the Ada update (Ada 9X) is complete, the balance could tip further in Ada's favor.

Adding to the case for Ada is the fact that the Ada scoring so well in the business case was Ada's 1983 version, MIL-STD-1815A. Just as C++ incorporates into C certain software engineering concepts already in Ada (e.g., modularity, strong typing, specification of interfaces), so an Ada update now underway will bring into Ada selected features now included in C++. This update, known as the Ada 9X Project, is targeted for completion in 1993. The product of extensive community involvement (including the C3 and MIS communities), Ada 9X will bring to Ada such improvements as decimal arithmetic, international character sets, improved input/output, support for calls between Ada and other languages, further representation specifications, and inheritance/polymorphism (popular features of C++). The Ada 9X Project Office lists one of the goals of Ada 9X as "to provide all the flexibility of C++ with the safety, reliability, and understandability of Ada 83."

At the same time, Ada 9X has been designed so that neither existing Ada benefits nor performance will be lost. For example, Ada 9X inheritance will be controlled so as not to reduce lifecycle supportability. Some have criticized OOP features such as inheritance as potentially dangerous to DOD software mission goals (such as safety, reliability, and dependability).

Bjarne Stroustrup himself, the originator of C++, has been quoted as follows: "C makes it easy for you to shoot yourself in the foot. C++ makes that harder, but when you do, it blows away your whole leg."

In summary, it is not possible to make a credible case for the existence of classes of "more cost effective" C++ systems compared to Ada. Business cost

effectiveness data collected for this study are typified by the TRW study's conclusion that Ada provides development cost advantages on the order of 35% and maintenance cost advantages on the order of 70%. In terms of full lifecycle costs, it will be some time before data exists which could justify a cost savings for C++. Today, there is limited lifecycle data available for Ada and almost none for C++.

For the foreseeable future, then, this business case shows that there are more than enough reasons for the DoD to stick firmly with Ada, both for all high order language (3GL and 3 1/2 GL) development and for exclusive use as a target language of 4GL application generators in the large class of applications for which 3GL code must supplement generated code.

Appendix E
Selected Ada Vendors

Company:	AdaSoft
Products:	AdaManager/AdaQuest, AdaMentor Computer Managed Instruction System, Graphical Modeling System (SL-GMS), AdaSoft Graphical User Interface (GUI), AdaSoft Textual User Interface (TUI).
Address:	AdaSoft, Inc., 8750-9 Cherry Lane, Laurel, MD 20707; Voice: (301) 725-7014; FAX: (301) 725-0980.
Company:	AETECH
Products:	Ada Workstation Environment (AWE), XAda APSE.
Address:	AETECH, 5841 Edison Place, Suite 110, Carlsbad, CA 92008; Voice: (619) 431-7714; FAX: (619) 431-0860.
Company:	Alsys
Products:	Ada compilers for a variety of platforms.
Address:	Alsys, Inc., 67 S. Bedord St., Burlington, MA 01803-5152; Voice: (617) 270-0030; FAX: (617) 270-6882.
Company:	Cadre Technologies
Products:	Teamwork (software engineering tool set).
Address:	Cadre Technologies, 222 Richmond Street, Providence, RI 02903; Voice: (401) 351-CASE; FAX: (401) 351-7380.
Company:	Caine, Farber & Gordon
Products:	PDL/81, a program design language which aids in the design and documentation of software. It is available for DOS, UNIX, and VAX platforms.
Address:	Caine, Farber & Gordon, Inc., 1010 East Union Street, Pasadena, CA 91106r; Voice: (800) 424-3070; Voice: (818) 449-3070; FAX: (818) 440-1742.

Company: Dynamics Research Corporation
Products: AdaMat.
Address: Dynamics Research Corporation, 60 Frontage Road, Andover, MA 01810; Voice: (800) 522-7321.

Company: EVB Software Engineering
Products: Complexity Measurement Tool (CMT), GRAMMI (interface builder).
Address: EVB Software Engineering, 5303 Spectrum Drive, Frederick, MD 21701; Voice: (301) 695-6960; FAX: (301) 695-7734.

Company: Fastrak Training
Products: Ada training and consulting.
Address: Quarry Park Place, Suite 300, 9175 Guilford Road, Columbia, MD 21046-1802; Voice: (301) 924-0050; FAX: (301) 924-3049.

Company: Idaho National Engineering Laboratory
Products: AdaSAGE.
Address: Idaho National Engineering Laboratory, EG&G Idaho, Inc., Special Applications Unit, P. O. Box 1625, Idaho Falls, ID 83415-1609; Voice: (208) 526-0656.

Company: Interactive Development Environments
Products: Software Through Pictures (Ada development environment).
Address: Interactive Development Environments 595 Market Street, 10th Floor, San Francisco, CA 94105; Voice: (800) 888-IDE1; Voice: (415) 543-0900; FAX: (415) 543-0145.

Company: Irvine Compiler Corporation
Products: ICC Ada, an Ada compiler for the HP 9000 Models 300, 400, 700, and 800, as well as a number of cross compilers.
Address: Irvine Compiler Corporation, 34 Executive Park, Suite 270, Irvine, CA 92714; Voice: (714) 250-1366; FAX: (714) 250-0676; E-mail: jkohli@irvine.com.

Company: Meridian Software Systems
Products: The AdaVantage compiler for PCs and some UNIX workstations.
Address: Meridian Software Systems, 10 Pasteur St., Irvine, CA 92718; Voice: (800) 221-2522; Voice: (714) 727-0700; FAX: (714) 727-3583.

Company: R. & R. Software
Products: Ada compilers for PCs.
Address: R. & R. Software, Inc., P. O. Box 1512, Madison, WI 53701; Voice: (800) 722-3248; Voice: (608) 244-6436.

Company: Rational
Products: The Rational Environment, Rational Rose.

Address: Rational, 2800 San Tomas Expressway, Santa Clara, CA 95051-0951; Voice: (408) 496-3600; FAX: (408) 496-3636.

Company: P. P. Texel & Company
Products: Ada training and consulting.
Address: Victoria Plaza, Building 4, Suite 9, 615 Hope Road, Eatontown, NJ 07724; Voice: (908) 922-6323.

Company: Verdix Corporation
Products: self-hosted compilers (VADSSelf), cross compilers (VADSCross), and Ada Programming Support Environment (VADS APSE).
Address: Verdix Corporation, 14130-A Sullyfield Circle, Chantilly, VA 22021; Voice: (703) 318-5800.

Appendix F
Ada-Related Organizations

Organization: ACM Special Interest Group on Ada (SIGAda)
Synopsis: ACM, founded in 1947, is the oldest of association for computing professiona'.s. It sponsors a number of special interest groups (SIGs) of which SIGAda is one. SIGAda promotes interest in and study of the Ada programming language. Its monthly publication is *Ada Letters* (\$20/year for ACM members, \$42/year others).
POC Mark Gerhardt, ESL, Inc., 495 Java Drive, MS M507, Sunnyvale, CA 94088-3510; VOICE: (408) 752-2459; E-mail: gerhardt@ajpo.sei.cmu.edu.

Organization: Ada Joint Program Office (AJPO)
Synopsis: The Ada Joint Program Office (AJPO) was established in December 1980. Its activities include validating Ada compilers, supporting the development and distribution of software tools for Ada, and ensuring that Ada-related efforts of the various services complement, not duplicate, one another. In this capacity, AJPO is responsible for managing efforts to provide training and life-cycle support for Ada.
POC: John Solomond, The Ada Joint Program Office, The Pentagon, Room 3E114, Washington, DC 20301-3080; Voice: (703) 614-0208; E-mail: solomond@ajpo.sei.cmu.edu.

Organization: Software Engineering Institute (SEI)
Synopsis: Realizing that the defense of the U.S. is increasingly dependent on computer-based systems, the DoD established the Software Engineering Institute (SEI) at Carnegie Mellon University in December 1984. The SEI staff, drawn from academia, government, and ind ıstry, consists of approximately 250 technical and support personnel. They are engaged in various research and development activities designed to improve the quality of software systems and of the software engineering process. The ultimate goal of the SEI is to change this process from a labor-

intensive craft to a well-managed mass-production oriented activity.

POC: Customer Relations, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213-3890; Voice: (412) 268-5800.

Organization: Software Technology Support Center (STSC)

Synopsis: The STSC, located at the Ogden Air Logistics Center (AFMC), Hill Air Force Base, Utah, was established by HQ USAF to promote development and management of tools, methods, and environments for software production throughout the services

POC: Software Technology Support Center, Attn: Customer Service, OO-ALC/TISE, Hill AFB, UT 84056; Voice: (801) 777-8045, FAX: (801) 777-8069

Appendix G
Ada Events Calendar[1]

The Ada Events Calendar includes information on upcoming Ada conferences, etc. It lists only those programs with fixed dates, and does not include programs, such as classes, that are scheduled on a continuing basis. Note, however, that many, if not most, of the conferences listed below are conducted on an annula basis

Date	October 5-7, 1992
Event	6th SEI Conference on Software Engineering Education
Location	Hyatt Islandia Hotel, San Diego, California
Sponsor	Software Engineering Institute
Synopsis	The SEI Conference on Software Engineering Education is an annual conference that brings together educators from universities, industry, and government to discuss issues related to the content, structure, and delivery of software engineering education. The conference format includes referred papers, panel discussions, reports, tutorials, and workshop sessions.
POC	Mary Ellen Rizzo, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213; Tel: 412/268-3007; FAX: 412/268-5758; Internet: mer@sei.cmu.edu.

Date:	October 14-15, 1992
Event	ASIS Technical Meeting
Location:	Institute for Defense Analyses (IDA), Alexandria, VA
Sponsor:	IDA
POC	Clyde Robey, IDA, Tel: 703/845-6666.

Date: October 14-15, 1992
Event: Ada UK International Conference
Location: Britannia International Hotel, London
Sponsor: Ada Language UK, Ltd.
POC: Helen Byard, Administrator, Ada UK, P.O. 322, York YO1 3GY, England; Tel: (UK) 0904 412740; Fax: (UK) 0904 426702.

Date: November 9-13, 1992
Event: Third Eurospace Ada in Aerospace Symposium
Location: Vienna, Austria
POC: Ms. Rosy Plet, Eurospace, 16 bis, Avenue Bousquet, F-75007 Paris, France; Tel: +33-1-45 55 83 53; Fax: +33-1-45 51 99 23.

Date: November 16-20, 1992
Event: TRI-Ada '92
Location: Orange County Convention Center, Orlando, FL.
Synopsis: The annual TRI-Ada Conference and Exposition is the Ada community's largest and most prestigious event. TRI-Ada, as its name implies, reaches out and brings together three broad bases in the Ada community—government, industry, and academia—with a program that covers all issues of importance to Ada interests. The theme for this year's conference is, "The Business of Ada."
POC: TRI-Ada '92, Danieli & O'Keefe Associates, Inc., Conference Management, Chiswick Park, 490 Boston Post Road, Sudbury, MA 01776 USA

Date: December 2-3, 1992
Event: NASA/GSFC Software Engineering Laboratory (SEL) 7th Annual Software Engineering Workshop
Synopsis: The workshop is an annual forum where software practitioners exchange information on the measurement, utilization, and evaluation of software methods, models, and tools. There will be a presentation of approximately 15 papers. Papers are being solicited which include the following topics: experiments in software development or management; experiences with software tools, models, methodologies; software measures; software reuse; software process assessment and improvement.
POC: Mr. Mark Cashion, NASA/Goddard Space Flight Center, Code 552, Greenbelt, MD 20771, USA; Phone: (301) 286-6347; FAX: (301) 286-9183

Date: December 7-11, 1992
Event: Toulouse '92: Fifth International Conference on Software Engineering & Its Applications
Location: Toulouse, France
Synopsis: Methods, tools, standards, and organization are the major aspects of software engineering covered by the conference. The conference will include tutorials, exhibits, and an industrial forum.

POC: Jean-Cleaude Rault, EC2, 269 rue de la Garenne, F-92024 Nanterre Cedex, France; Tel: +33-1-47 80 70 00; Fax: +33-1-47 80 66 29

Date: December 8-10, 1992
Event: STARS '92
Location: Omni Shoreham Hotel, Washington, D.C.
Sponsors: STARS, Boeing, IBM, Paramax
Synopsis: Megaprogramming concepts, the firm foundation in products, relevant successes, and upcoming plans will be discussed at the program. Integration of process and reuse within the Software Technology for Adaptable Reliable Systems (STARS) environment will be demonstrated. There will be exhibits of megaprogramming work in progress by STARS as well as affiliates and subcontractors. Evening receptions will facilitate networking with government and industry leaders.
POC: The STARS '92 Conference Center, 3 Church Circle - Suite 194, Annapolis, MD 21401; Fax: (410) 267-0332; E-Mail: STARS92-Desk@STARS.Rossyln.Unisys.com

Date: December 10, 1992
Event: Ada's Birthday

Date: March 15-18, 1993
Event: 11th Annual National Conference on Ada Technology
Location: Williamsburg Hilton and National Conference Center, Williamsburg, Virginia.
Synopsis: The emphasis for this conference will be software engineering, while continuing to emphasize Ada as an important building block. Papers on applied aspects of software engineering and also experimentation and research are being accepted. Presenters must register for the Conference.
POC: ANCOAT 93, c/o Rosenberg & Risinger, 11287 W Washington Blvd., Culver City, CA 90230, (310) 397-6338

Date: March 21-23, 1993
Event: 5th Annual Oregon Workshop on Software Metrics
Location: Silver Falls State Conference Center, Portland Oregon
Synopsis: The Annual Oregon Workshop on Software was founded to allow the interchange of ideas and experiences between those using metrics and those performing research in the area. Call for participation is sought from both practitioners and researchers. Participation may consist of delivering a paper, organizing and leading a panel session, or leading a mini-tutorial on some aspect of software measurement.
POC: Warren Harrison, Center for Software Quality Research, Portland State University, Portland, OR 97207-0751; (503) 725-3108; warren@cs.pdx.edu

Date: March 24-26, 1993
Event: Second International Workshop on Software Reusability
Location: Lucca, Italy
Sponsors: IEEE Computer Society, ACM SIGSOFT
Synopsis: The themes for this year's workshop include methods, tools and environments, reuse library methods, generative approaches to reuse, constructive approaches to reuse, theoretical aspects of reuse, organizational and management techniques for implementing reuse, domain analysis methods and techniques. Attendance is limited to 100.

Date: April 18-23, 1993
Event: Fifth Annual Software Technology Conference
Location: Red Lion Hotel & Salt Palace Convention Center, Salt Lake City, Utah
Sponsors: U.S. Army, U.S. Navy, U.S. Air Force
Theme: Software - The Force Multiplier
Synopsis: The program will include tutorials, vendor demonstrations, presentations, and "birds-of-a-feather" discussion groups. The theme for this year's conference is, "Software - The Force Multiplier." General sessions will address management information systems, embedded computers, and command and control. Proposed topics for presentations include reuse, environments, Ada implementation, software inspections, change management, object oriented programming, automated software process enactment, metrics, re-engineering, software engineering, software maintenance, DoD software initiatives, configuration management, and software process improvement.
POC: Dana Dovenbarger, Conference Manager, Software Technology Support Center, OO-ALC/TISE, Hill AFB, UT 84056; Phone: (801) 777-7411; DSN 458-7411; FAX: (801) 777-8069

Date: May 17-21, 1993
Event: ICSE: 15th International Conference on Software Engineering
Location: Baltimore, MD.
Sponsor: IEEE Computer Society Technical Committee on Software Engineering and ACM Special Interest Group on Software Engineering.

Date: June 14-18, 1993
Event: Ada Europe
Location: Paris, France
POC: Ada Europe '93, AFCET, 156 Bd. Pereire, 75017 Paris, France

Date: September 18-23, 1993
Event: TRI-Ada '93
Location: Seattle, Washington
Synopsis: The theme for the 1993 conference is, "The Management and Engineering of Software."

Appendix H
Glossary of Acronyms

ACE	Ada Command Environment (STARS)
ACM	Association for Computing Machinery
ACVC	Ada Compiler Validation Capability (AJPO)
Ada	Not an acronym
ADA	American Dental Association
ADA	Americans with Disabilities Act
Ada-IC	Ada Information Clearinghouse
AdaJUG	Ada Joint User's Group
AIC	Ada Information Clearinghouse
ALRM	Ada Language Reference Manual
AJPO	Ada Joint Program Office
ANNA	ANNotated Ada (Stanford University)
APSE	Ada Programming Support Environment
ASAP	Ada Static source code Analyzer Program
ASEET	Ada Software Engineering Education and training Team
ASR	Ada Software Repository
ATIP	Ada Technology Insertion Program (AJPO)
CAIS	Common APSE Interface Set
CARDS	Central Archive for Reusable Defense Software
CASE	Computer Aided Software Engineering
CMU	Carnegie-Mellon University
COSMIC	COmputer Software Management and Information Center (NASA)
CREASE	Catalog of Resources for Education in Ada and Software Engineering
DACS	Data and Analysis Center for Software
DARPA	Defense Advanced Research Projects Agency
DIANA	Distributed Intermediate Attributed Notation for Ada
DID	Data Item Description
DTIC	Defense Technical Information Center
FIPS	Federal Information Processing Standard
GRACE	Generic Reusable Ada Components for Engineering (EVB)
GRAMMI	Generated Reusable Ada Man Machine Interface (ESL)
HOL	High Order Language

HOLWG	High Order Language Working Group
IDD	Interface Design Document
IEEE	Institute for Electrical and Electronic Engineers
IEEE CS	IEEE Computer Society
IPSE	Integrated Project Support Environment
KAPSE	Kernel APSE
KIT	KAPSE Interface Team
KLOC	Thousand LOC
LOC	Lines Of Code
LRM	Language Reference Manual
MAPSE	Minimal APSE
MIL-STD	Military Standard
NBS	National Bureau of Standards (obsolete, now NIST)
NISE	NASA Initiative on Software Engineering
NIST	National Institute for Standards and Technology (formerly NBS)
NUMWG	NUMerics Working Group (SIGAda)
OO	Object Oriented
OOA	Object Oriented Analysis
OOD	Object Oriented Design
OOP	Object Oriented Programming
OOSD	Object Oriented Structured Design
PCTE	Portable Common Tool Environment (ESPRIT)
PDL	Program Design Language
PIWG	Performance Issues Working Group (SIGAda)
RADC	Rome Air Development Center
RIACS	Research Institute for Advanced Computer Science (NASA)
SAME	SQL Ada Module Extension
SDD	Software Design Document
SE	Software Engineering
SEI	Software Engineering Institute (Carnegie Mellon University)
SGML	Standard Generalized Markup Language
SIGAda	Special Interest Group on Ada (ACM)
SIGPLAN	Special Interest Group on Programming LANguages
SIGSOFt	Special Interest Group on SOFTware
SQA	Software Quality Assurance
SJPO	STARS Joint Program Office
STARS	Software Technology for Adaptable Reliable Systems
STSC	Software Technology Support Center
VADS	Verdix Ada Development System (Verdix)
WADAS	Washington Ada Symposium

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1994	Final Report

4. TITLE AND SUBTITLE

Transition to Ada

5. FUNDING NUMBERS

6. AUTHOR(S)

William A. Ward, Jr.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Faculty Court West 20
School of Computer and Information Sciences
University of South Alabama
Mobile, AL 36688

8. PERFORMING ORGANIZATION REPORT NUMBER

Technical Report
USA/CIS-94-TR-01

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Information Technology Laboratory
U.S. Army Engineer Waterways Experiment Station
3909 Halls Ferry Road, Vicksburg, MS 39180-6199

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

Technical Report
ITL-94-8

11. SUPPLEMENTARY NOTES

Available from National Technical Information Service, 5285 Port Royal Road, Springfield, VA 22161.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report addresses issues relevant to the transition to the use of Ada from a Corps of Engineers perspective. The direct discussion of these issues is preceded by background material on Ada itself. First, the Department of Defense (DoD) software crisis that led to the development of Ada is described and the causes underlying it are discussed. Next, a brief history of Ada is presented to show how it fits into the Government's approach to meeting the crisis. This includes a discussion of the guidelines which apply to the use of Ada, specifically the Congressional mandate to use Ada and the pertinent DoD and Army regulations. The second major section of this report discusses the Corps transition to Ada. This transition will involve not only a change in the programming language used by the Corps, but also a change in development philosophy; software engineering principles must be incorporated into the development process for the transition to be successful. The various issues to be addressed by the Corps in order to accomplish this are then presented. The report concludes with recommendations concerning practical steps Corps development sites can take to ensure a successful transition to the use of Ada.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Ada, software engineering, Rational Environment	104
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED		