United States General Accounting Office

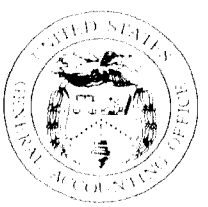
GAO

Report to the Chairman, Subcommittee on Defense, Committee on Appropriations, House of Representatives

March 1989

PROGRAMMING LANGUAGE

Status, Costs, and Issues Associated With Defense's Implementation of Ada



GAO/IMTEC-89-9

GAO United States
General Accounting Office
Washington, D.C. 20548

**Information Management and
Technology Division**

B-231334

March 24, 1989

The Honorable John P. Murtha
Chairman, Subcommittee on Defense
Committee on Appropriations
House of Representatives

Dear Mr. Chairman:

In response to your predecessor's request, we examined the Department of Defense's use of the Ada computer programming language. Defense has selected Ada as the language of choice for all computer applications. We found that Defense has not maintained complete records on projects using Ada, and has not demonstrated whether using Ada can help control software development and maintenance costs. We also found there are technical issues that need to be resolved before Ada can be effectively used as the computer programming language for all Defense computer applications.

We are sending copies of this report to the Secretary of Defense and to other interested parties. This report was prepared under the direction of Howard G. Rhile, Jr., Associate Director. Other major contributors are listed in appendix IX.

Sincerely yours,

Ralph V. Carlone

Ralph V. Carlone
Assistant Comptroller General

Executive Summary

Purpose

The Department of Defense has selected the Ada programming language as the single, common computer language for use in both its automated weapons and information systems. By using the Ada language and new software development methods that Ada supports, Defense expects to reduce software life cycle costs for its automated systems through (1) software sharing, (2) increased portability of software among systems, and (3) reduced software maintenance costs.

Citing Ada's lack of an extensive performance history and the problems that some Defense programs have apparently experienced in using Ada, the former Chairman, Subcommittee on Defense, House Committee on Appropriations, asked GAO to examine Defense's use of Ada. GAO's objectives were to identify the current and planned uses of Ada within Defense, costs associated with implementing Ada, and technical issues associated with its use. In reviewing the costs associated with implementing Ada, GAO also examined available evidence of cost savings accruing from its use.

Background

In 1974, Defense's future software development costs were estimated at more than $3 billion annually. At that time, over 300 programming languages were being used on Defense systems, making it difficult to move application programs among computer systems and expensive to maintain these programs. In response, Defense initiated work in 1975 that led to the development of the Ada programming language in 1979. Ada was established as a military standard in 1980, and was approved by the American National Standards Institute in 1983 and by the International Standards Organization in 1987. Also in 1987, Defense established a policy calling for the use of Ada for all computer applications, except where the use of another language could be demonstrated to be more cost-effective. (See ch. 1.)

To examine Defense's use of Ada, GAO planned to obtain information on all Defense projects that were using or planning to use the Ada programming language. In October 1987, Defense identified 123 Ada projects. However, this list of Defense projects was incomplete. By March 1988, an additional 75 projects had been identified. As agreed with the Subcommittee, since the total number of Defense projects using Ada could not be identified, we limited our work to information obtained on 100 Ada projects.

Results in Brief

Information on 100 Ada projects shows that Ada is being used in many different types of computer applications. However, while some information is known about Defense's use of Ada, this information is not complete. Neither Defense overall nor the individual military services maintain complete records on projects using Ada.

Defense has not yet demonstrated whether the use of Ada can help control software development and maintenance costs. A Defense official believes that although insufficient documentation currently exists, such documentation will become available as Defense implements programs using Ada. The total cost specifically associated with implementing Ada cannot be determined.

Five technical issues have affected the ability of Defense program managers to use Ada. Experts are divided as to whether these problems are inherent in the Ada language or whether they can be solved as the language matures.

Principal Findings

Status of Ada Usage Within the Department of Defense

Both Defense and the individual military services' inventories on Ada projects are incomplete. In GAO's view, such records are necessary to enable program officials to search out, discuss, and benefit from lessons learned on other projects. Such records are also necessary to facilitate the identification and sharing of computer programs—one of the expected benefits of using Ada that is intended to reduce the costs of developing and maintaining computer programs. Without complete information on Defense projects using Ada, it will be difficult for Defense to assess whether the use of Ada is achieving its intended objectives.

GAO obtained information on 100 Defense projects using Ada and found that the majority of projects were either being planned or were in development (87 percent). Most of these projects were being done by the Departments of the Air Force and the Army. The projects covered a wide range of activities. (See ch. 2.)

Costs

Defense has not designed projects to assess the long-term cost savings and other benefits from the use of Ada. This raises questions as to

whether the use of Ada can help control Defense's software development and maintenance costs, and fuels uncertainty as to whether Ada can achieve its intended objectives. Such uncertainty could be diminished by emphasizing projects demonstrating feasibility and cost savings from using Ada, and disseminating the results of these projects to Defense program managers.

Pinpointing costs associated with Defense's implementation of Ada is difficult. Projects that use Ada do not segregate the specific costs of developing application programs coded in Ada as distinct from other project costs. Therefore, the only identifiable costs are those associated with three Ada and software engineering-related organizations (about $201 million through fiscal year 1988), and projects undertaken by each military service to support Ada implementation (about $190 million). (See ch. 3.)

Technical Issues

Five specific technical issues have been raised as areas needing attention before Ada can be effectively used as the computer programming language for all Defense computer applications. Two involve the availability of software development tools and the performance quality of compilers for use in Defense projects, two involve the usability of Ada in real-time systems that require rapid data processing and distributed systems in which several computers process data simultaneously, and one involves the use of Ada application programs with data base management systems.

A compiler translates code from a high-order language in which programs are written into machine language processed by computers. The availability of Ada compilers and other software development tools was a problem in the early years of Defense's mandate to use Ada. This problem has since abated as new compilers and other tools have been developed. The focus today is on the quality of Ada compilers and the use of Ada features to support those real-time applications where precise timing control, processing speed, and computer program size are critical elements. Research is being done to develop ways to use these new Ada language features effectively.

Building real-time distributed systems is difficult regardless of the language used. The use of Ada features to build these systems holds promise once the real-time issues are resolved. Research on building real-time distributed systems in Ada is underway.

Although Ada can work with data base management systems, currently there is no standard method that permits Ada application programs to work with the Structured Query Language endorsed by the American National Standards Institute to access data in such systems. Four methods have been proposed to achieve this capability, but each poses technical issues that must be resolved. (See ch. 4.)

Recommendations

GAO recommends that the Secretary of Defense take steps to

- develop performance data that demonstrate whether Defense's use of Ada is achieving its goals,
- develop a Defense-wide information repository on computer application programs and modules written in Ada and make them available for reuse,
- gather and disseminate complete lists of all Defense projects using Ada, and
- obtain from an independent body of Ada experts an assessment of projects demonstrating intended cost savings associated with Ada's use, research efforts to overcome technical problems with specific applications, and progress in developing inventories of Ada software, along with any recommendations related to the appropriate use of Ada. (See ch. 5.)

Agency and Contractor Comments

The Department of Defense and the Software Engineering Institute commented on a draft of this report. These organizations generally agreed with GAO's findings and recommendations. Both organizations commented, however, that the report's discussion of Ada's technical issues unduly criticized the Ada language itself. Both organizations, however, acknowledged that there were technical issues associated with its implementation that need to be resolved. The report contains an evaluation of these comments in chapter 5.

Contents

Abbreviations

DOD	Department of Defense
GAO	General Accounting Office
IMTEC	Information Management and Technology Division
SQL	Structured Query Language

Introduction

The Department of Defense (DOD) has selected the Ada programming language as the single, common computer language for use in both its automated weapons and information systems. Using the Ada language and new methods of software development that Ada supports is expected to reduce software life cycle costs for DOD systems. These cost reductions are expected to result through (1) software sharing, (2) increased portability of software among computer systems, and (3) reduced software maintenance costs. This report discusses the current and planned uses of Ada within DOD, the costs and benefits associated with implementing Ada, and the status of technical issues affecting its use in DOD program applications.

Background

In 1974 DOD's future software development costs were estimated at more than $3 billion annually. At that time, over 300 programming languages or versions of these languages were being used on DOD systems. This made it difficult and expensive to move application programs among computer systems because different tools and expertise were required for each language.

Faced with these facts, in 1975 DOD initiated a project to define a single high-order language[1] to meet the programming needs of DOD embedded computer systems.[2] The Institute for Defense Analyses drafted requirements that were reviewed by experts in the military, industrial, and academic communities; they were refined on the basis of comments received. The revised draft requirements were evaluated against the capabilities of existing computer programming languages. Although no language was found that could satisfy all of the requirements, DOD concluded that it was feasible to construct a single high-order computer programming language that could meet its needs.

In 1977 DOD contracted with four vendors to competitively produce preliminary designs for this single high-order computer programming language. The preliminary designs from the four vendors were widely distributed. On the basis of comments received, the requirements for this language were made final and two of the vendors were chosen to continue design work to meet these final requirements. The designs

[1]High-order languages are computer programming languages that are several steps removed from basic machine instructions; that is, one instruction written in a high-order language will usually translate into more than one machine instruction. High-order languages can be used to write programs for use on different makes of computer hardware. As a result, the programs, with certain modifications, may be transferred among computers built by different manufacturers.

[2]An embedded computer is a computer built into a larger system, such as a weapon system.

developed by these two vendors were distributed for public comment in 1979. After considering the comments received, DOD selected the language designed by a team led by Jean D. Ichbiah at Cii-Honeywell Bull. This language was named Ada.

DOD approved Ada as a military standard programming language in 1980. The Ada language was subsequently approved by the American National Standards Institute in 1983 and by the International Standards Organization in 1987.

Although the requirements of embedded computer systems provided the motivation for the design of Ada, DOD has expanded its use beyond embedded computers to realize the expected benefits of Ada on other application programs. DOD has declared Ada as the required language for developing all military computer application programs, except where the use of another language can be demonstrated to be more cost-effective.

Ada Programming Language

Ada is a general-purpose high-order computer programming language that incorporates new features along with many of the special features of other commonly used programming languages. A September 1987 report of the Defense Science Board Task Force on Military Software stated:

"Software engineering methods and techniques have dramatically advanced over the last decade, yet these techniques are not generally practiced in DOD. Ada is not merely a programming language; it is a vehicle for new software practices and methods for specification, program structuring, development and maintenance."

[Text omitted]

"Ada supports the evolution and maintenance of reusable software, portable software, and real-time software."

A report prepared by the Software Engineering Institute[3] addressed the advantages and risks inherent in adopting Ada by stating that

"The Ada language effort focuses programming development methods and tools on a single language that supports modern software engineering techniques. Ada's role

[3]John Foreman and John Goodenough, Ada Adoption Handbook: A Program Manager's Guide, Software Engineering Institute, Carnegie-Mellon University, Technical Report CMU/SEI-87-TR-9, ESD-TR-87-110 (Pittsburgh, Pennsylvania: May 1987), pp. 19-20.

as the single, common, high-order programming language for computers integral to weapon systems is a major step forward in addressing DOD software development problems."

According to this report, the Ada language offers potential solutions to software development problems by

- reducing the costs of modifying and maintaining software;
- providing early identification of computer programming errors to reduce software development costs and to increase system reliability;
- moving source code among different computers with minimum change;
- serving as a focal point for developing a common set of high-quality software development tools and methodologies; and
- providing greater mobility of software personnel among projects at lower training costs.

Although potential advantages accrue in establishing the Ada programming language as a standard, the mere presence of Ada and supporting technology does not guarantee successful software development. It is possible to write bad computer application programs in Ada as well as in other languages. Using Ada effectively requires the continued use of sound software development practices, as well as a knowledge of the language and use of new software engineering methods that the language supports.

Three Defense Organizations Are Responsible for Encouraging the Use of Ada or Advancing Software Engineering Methodology

Three organizations have been established by DOD to ensure the smooth introduction, implementation, and life-cycle support for the use of Ada or to advance the use of new software engineering methods supported by the Ada programming language.

In 1980 DOD established the Ada Joint Program Office to manage the introduction of Ada. This office is managed within the Office of the Deputy Director for Defense Research and Engineering (Research and Advanced Technology). The primary responsibilities of this office are to

- ensure that Ada is implemented and maintained as a consistent, unambiguous standard recognized by DOD and the widest possible community;
- ensure that Ada is used by DOD managers in satisfying their computer programming needs; and
- support the development of Ada tools to improve productivity.

In 1984 DOD established the Software Technology for Adaptable, Reliable Systems Joint Program Office to advance software engineering technology. The goals of this office are to (1) improve quality and reliability in computer application programs, (2) promote the development and reuse of software modules, and (3) reduce the time and cost of developing software for DOD programs. Ada is the language of choice for all activities performed by this office. According to DOD, no other language has as many technical features supporting software engineering techniques or has the degree of standardization across so many computers as that which has been achieved with the Ada language. This office is supported by each of the military services and defense agencies, and is managed by the Defense Advanced Research Projects Agency.

Also in 1984, DOD established the Software Engineering Institute, a federally funded research and development center at Carnegie-Mellon University. The Institute was established to accelerate the transition and use of modern software engineering techniques and methods in DOD programs. While many of the Institute's activities are focused on general software engineering objectives, Ada is the primary language used by the Institute in pursuit of these objectives. The Electronic Systems Division, Air Force Systems Command is the administrative agent for the Institute. General policy and program guidance is provided by a joint advisory committee consisting of joint logistics commanders, and representatives from the Office of the Secretary of Defense and other defense agencies. The Institute is managed by the Defense Advanced Research Projects Agency.

Use of Ada Required by All Military Departments and Agencies

In June 1983 DOD issued a proposed revision to DOD Instruction 5000.31, "Interim List of DOD Approved High Order Programming Languages." One of the objectives of this proposal was to minimize the number of programming languages used within DOD. In a June 1983 memorandum to the military services and defense agencies, the Under Secretary of Defense for Research and Engineering directed that the proposed revision be implemented immediately, pending its issuance as an official policy. The Under Secretary stated in this memorandum that

"The Ada programming language shall become the single, common computer programming language for Defense mission-critical applications. Effective 1 January 1984 for programs entering Advanced Development and 1 July 1984 for programs entering Full-Scale Engineering Development, Ada shall be the programming language."

This proposed revision was never made final, however, and the instruction was replaced in April 1987 by DOD Directive 3405.1, "Computer Programming Language Policy." This directive establishes Ada as the single, common computer programming language for Defense computer resources used (1) in intelligence systems, (2) for the command and control of military forces, or (3) as an integral part of a weapon system. The directive further provides that Ada shall be used for all other computer applications, except when the use of another approved high-order language can be demonstrated to be more cost-effective over the application's life cycle.

DOD Directive 3405.2, "Use of Ada in Weapon Systems," was issued in March 1987. This directive established a policy that Ada immediately become the single common computer programming language throughout DOD for computers integral to weapons systems. This directive also prescribed procedures for using Ada in such systems.

Implementation of Ada by the Military Services

In January 1984 the Departments of the Air Force and the Army established procedures requiring the use of Ada in major programs. Procedures were also established for granting waivers from the requirement to use Ada when justified by life cycle cost and technical practicality. In the Air Force, requests for waivers from the requirement to use Ada in major programs required approval by the Air Force Technology and Security Division.[4] In the Army, such approval authority was delegated to the Assistant Secretary of the Army for Research, Development, and Acquisition, and later redelegated to the Assistant Chief of Staff for Information Management.[5] Between 1984 and 1987, Air Force and Army waiver officials received five and ten requests, respectively, for waivers from the requirement to use Ada in major programs. The Air Force approved all five requests, and the Army approved four of the ten waivers requested.

The Department of the Navy uses custom built computers for its aircraft and shipboard systems. Prior to November 1988, software development tools necessary to use Ada with these computers were not available and

[4]In November 1988, the responsibility for approving waivers for major intelligence, command and control, and weapons systems was delegated to the Principal Assistant to the Assistant Secretary of the Air Force (Acquisition). For major automated information systems, this responsibility was delegated to the Air Force Directorate of Architecture and Technology, Assistant Chief of Staff, Systems for Command, Control, Communications, and Computers.

[5]In September 1987, the responsibility for approving waivers was again redelegated to the Director of Information Systems for Command, Control, Communications, and Computers.

were being developed by the Navy (see ch. 2). Accordingly, the Navy did not require program managers to request waivers from the requirement to use Ada in developing application programs for these computers.

In 1985 the Navy issued a policy requiring the use of Ada in all computer programs to be used on commercially-available computer systems, unless a waiver was approved by the Commander, Space and Naval Warfare Systems Command. Between 1985 and 1987, the Navy received 43 requests for waivers from the requirement to use Ada in commercial computer applications. As of July 15, 1988, 23 waivers had been approved, two had been denied, one had been returned for more information, and 17 requests were pending.

In November 1988, the Department of the Navy issued an instruction formally implementing DOD Directives 3405.1 and 3405.2. At that time, Ada compilers for two of the Navy standard computers had been developed and were being tested for conformance with the Ada language standard.

Objectives, Scope, and Methodology

On June 4, 1987, the former Chairman, Subcommittee on Defense, House Committee on Appropriations, asked us to review DOD's use of the Ada programming language (see app. I). Our objectives were to identify (1) current and planned uses of Ada in DOD, (2) costs associated with implementing Ada, and (3) technical issues associated with its use. In reviewing the costs associated with implementing Ada, we also examined evidence of cost savings accruing from its use. This review did not include an examination of personnel costs or other personnel issues associated with the Ada programming language.

To accomplish our objectives, we analyzed records, studies, and other documentation related to the Subcommittee's areas of interest. We interviewed DOD officials in the (1) Ada Joint Program Office in Arlington, Virginia; (2) Software Technology for Adaptable, Reliable Systems Joint Program Office in Arlington, Virginia; (3) Department of the Air Force in Arlington, Virginia and Andrews Air Force Base, Maryland; (4) Department of the Army in Arlington, Virginia; and (5) Department of the Navy in Arlington, Virginia. We also interviewed software engineers at the Software Engineering Institute, a federally funded research and development center, in Pittsburgh, Pennsylvania.

By consulting with officials and reviewing available records at the Ada Joint Program Office and at each of the military services (Air Force,

Army, and Navy), 123 DOD projects were identifed that were using or planning to use the Ada programming language. As of March 1988, we had interviewed program officials and obtained program documentation on 100 of these projects at 22 organizations (see app. II) to determine the characteristics of each program and to identify experiences with the use of Ada.

We initially planned to obtain information on all 123 projects identified as using or planning to use Ada. In March 1988 we advised the Subcommittee that the original list of projects was substantially incomplete. Specifically, an additional 75 projects had been identified that were reportedly using or planning to use Ada. As agreed with the Subcommittee, we excluded from the scope of our work the remaining 23 projects originally identified, as well as any newly identified projects. Information on all these projects will be referred to the Ada Joint Program Office for their follow-up and use in gathering and disseminating information on the use of Ada.

We obtained information on the costs associated with implementing Ada by analyzing expenditures and budgets for the (1) Ada Joint Program Office; (2) Software Technology for Adaptable, Reliable Systems Joint Program Office; and (3) Software Engineering Institute. DOD project documentation was also analyzed to identify costs that were specifically associated with implementing Ada.

To identify and evaluate the technical issues associated with using Ada, we reviewed technical publications and gathered and analyzed information from the Ada Joint Program Office; the Software Technology for Adaptable, Reliable Systems Joint Program Office; and the Software Engineering Institute. To determine the availability of Ada software development tools, in February and March 1988 we gathered and analyzed information from 16 companies responsible for 75 percent of the validated Ada compilers being marketed. Further, we discussed our assessment of the technical issues affecting DOD's use of Ada with eight recognized Ada and/or software engineering experts in industry and academia (see app. III).

We conducted our review from July 1987 through September 1988 in accordance with generally accepted government auditing standards. Both DOD and the Software Engineering Institute provided written comments on a draft of this report. These organizations suggested some technical changes to this report which have been made where appropriate. Our evaluation of these comments is presented in chapter 5. DOD's

comments are included in appendix VII and the Institute's comments are included in appendix VIII.

Current and Planned Uses of Ada Within DOD

DOD and the military services do not maintain complete lists of projects using the Ada programming language. Such records would facilitate the identification and sharing of computer programs and lessons learned among Ada projects. While they are not required to create or maintain such lists, DOD and service officials agree that such information would be beneficial to other projects.

Our analysis of 100 DOD projects using or planning to use Ada showed that Ada is being used for a wide variety of purposes. The projects were designed to (1) assess the feasibility of using Ada for specific applications, (2) develop software tools necessary to use Ada, or (3) develop computer application programs. At the time of our analysis, 8 projects were planned, 79 were under development, and 13 had been completed.

Information on DOD Projects Using Ada Is Incomplete

DOD has established "language control agents" to support the use of each DOD-approved high-order language. Although responsibilities of these control agents include gathering and disseminating "appropriate" information regarding the use of their assigned high-order language and its associated tools, responsibilities do not specifically include keeping lists of projects using their respective languages.

The Ada Joint Program Office is the DOD control agent for Ada. This office has established a contractor-operated Ada Information Clearinghouse to gather and disseminate information on Ada projects, as well as information on Ada tools, conferences, seminars, and training activities. The clearinghouse solicits data on Ada projects from DOD, industry, and academia. The information submitted is verified with program sponsors and then published for use within the Ada language community. The purpose of this activity is to enable the entire Ada user community to benefit from individual and organizational experiences in the use of the language. Participation in this activity is voluntary and the director of the Ada Joint Program Office acknowledged that its list of DOD projects using or planning to use Ada is incomplete.

The reasons identified by the Ada Joint Program Office for the lack of a complete inventory of Ada projects include the voluntary nature of the listing and an associated lack of incentive for program managers to provide project information to the clearinghouse.

Between September 1987 and March 1988, each military service designated an "Ada Executive" to monitor programs relative to the use of Ada. Although in the past the services have not maintained complete

lists identifying all projects using Ada, each of these Ada Executives agrees that complete lists of programs using Ada would be useful. These officials believe that such lists would facilitate (1) sharing experiences with Ada projects, (2) identifying different kinds of programs written in Ada, and (3) serving as a catalog of programs that have developed software modules that might be reusable in other programs. The Navy Ada Executive said that his office is currently preparing a list of Navy projects using Ada which, once prepared, will be kept current.

Our analysis of records initially obtained from the Ada Information Clearinghouse and each of the military services in October 1987 identified 123 DOD projects that were using or planning to use Ada. Subsequently, the Air Force provided us with records showing 34 additional projects. While gathering information on projects at five locations, we identified 41 more projects that were not included in the records provided by DOD and the military services. Thus is it likely that many more DOD projects are using Ada than has been reported.

Projects Using or Planning to Use Ada

Information obtained on 100 DOD projects shows that Ada is being used for a variety of activities, ranging from studies and demonstration projects to developing application programs. Appendix IV is a list of these projects.

Status and Types of Ada Projects

At the time of our review, the majority of Ada projects were either being planned or were in development (87 percent). The majority of these Ada projects (86 percent) were sponsored by the Departments of the Air Force and the Army.

Table 2.1: Status of Ada Projects

Project status	Air Force	Army	Navy	Total
Planned	1	7		**8**
In development	30	37	12	**79**
Completed	9	2	2	**13**
Total	**40**	**46**	**14**	**100**

The projects cover studies and demonstrations to assess the suitability of Ada for specific applications, development of tools necessary to use Ada, and development of computer application programs. Application programs are being developed for use in command and control systems, avionics systems, trainers, simulators, test equipment, and many other

types of systems. The types of Ada projects planned or in development are shown in table 2.2.

Table 2.2: Types of Ada Projects Planned or in Development

Type of project	Air Force	Army	Navy	Total
Command, control, and communications	11	10	4	**25**
Studies, demonstrations, and software tools	12	2	1	**15**
Surveillance and reconnaissance		8		**8**
Trainers and simulators	4	2		**6**
Weapons systems	1	4	2	**7**
Avionics systems		3	1	**4**
Others	3	15	4	**22**
Total	**31**	**44**	**12**	**87**

As shown in table 2.3, the completed projects generally involved developing the tools necessary to use Ada, or studies to assess the feasibility of using Ada for specific applications.

Table 2.3: Types of Completed Ada Projects

Type of project	Air Force	Army	Navy	Total
Studies	3			**3**
Demonstration	1			**1**
Software development tools	3	1		**4**
Hardware development tool			1	**1**
Trainers	1		1	**2**
Simulator	1			**1**
Manufacturing system		1		**1**
Total	**9**	**2**	**2**	**13**

Software Development Tool Projects

Software development tools are computer programs used by a programmer to design, develop, and implement application programs. When the Ada language was developed, new tools had to be built to work with this language. During 1979 and the early 1980s, when few Ada tools were available from commercial sources, the military services initiated projects to develop tools needed to write Ada application programs.

Ada Language System

The Army initiated the Ada Language System project in 1980 to support software development, improve the productivity of programmers, and

improve management control over the software life cycle. In 1983 the Army released early versions of these tools to industry to stimulate industry interest in developing Ada tools. The Army planned to use these tools with new battlefield computer systems being developed under its Military Computer Family project. In 1984, however, the Army cancelled plans to develop standard battlefield system computers. The Ada Language System project was terminated in 1986. This project cost about $32.6 million and included over 500,000 lines of code and more than 70 distinct tools to support Ada programmer activities. To encourage industry support for the maturing Ada language and to maximize the benefits from its investment, the Army released the Ada Language System to the public domain.

Ada Language System/ Navy

The Navy initiated the Ada Language System/Navy project to limit the proliferation of service-unique Ada language support systems and to reduce overall DOD and Navy implementation costs. Formal system specifications were developed in fiscal years 1982 and 1983 for this ongoing project to develop Ada software development tools for use with newer generations of three standard Navy computers.

The Navy used the Army's Ada Language System as the development baseline for its Ada implementation effort. In this project, the Navy is adapting the Ada Language System to support its standard computers. The Navy is also developing additional tools to write application programs for these computers. Ada compilers and other software for two of the Navy standard computers have been developed and are currently being tested for conformance with the Ada language standard. A compiler is a program that translates code from a high-order language that is convenient for a programmer to use to machine language processed by computers. The Navy plans to mandate the use of these compilers and software for new starts and major software upgrades. Improved versions of the compilers, incorporating additional capabilities, are expected to be available by September 1990. This project is expected to cost the Navy about $79.7 million, which includes about $15 million in support of improvements for the standard computers.

According to the Navy Ada Implementation Plan, dated March 1987, reduced fiscal year 1987 funding caused a 12-month delay in the projected completion of this project—from September 1989 to September 1990.

Ada Integrated Environment

The Air Force initiated a project in 1979—the Ada Integrated Environment—to develop an Ada compiler and a fully integrated Ada programming support environment. In 1985, following cost growth and schedule delays, the scope of this project was narrowed to include only the compiler development portion. It was completed in 1987, at a cost of about $11.8 million.

Other Programming Languages Are Being Used With Ada in Most Computer Application Programs

Our analysis showed that Ada is generally used in conjunction with other languages. Computer programs in 59 of the 100 projects we analyzed involved both Ada and at least one other computer programming language. In 44 of these 59 projects, assembly language[1] was used. While we did not determine the specific reasons for the use of assembly language in these Ada projects, application programs written in other high-order languages generally use assembly language to write the portions of application programs that require rapid processing (see ch. 4).

[1] A computer programming language that corresponds directly with the instructions that the hardware will execute; that is, one instruction written in assembly language will translate into one machine instruction. Assembly languages vary among manufacturers, which could limit the portability of Ada programs that include portions written in assembly language.

The Total Cost and Benefits of Implementing Ada Have Not Been Determined

Just as the total number of DOD projects using Ada is unknown, so are the costs and benefits of its implementation. We were unable to determine the costs specifically associated with using Ada versus another language in developing computer application programs for use in operational systems such as command and control, avionics, and weapons. In fiscal years 1982 to 1988, about $201 million has been provided to three DOD organizations whose primary focus is on implementing Ada or new software engineering techniques. In addition, 23 of the 100 projects we analyzed focused on studies and demonstrations of using Ada in certain applications and developing Ada-related software development tools. Costs of these projects total about $190 million.

We were unable to identify any DOD projects designed to assess the long-term benefits and cost savings of using Ada. The director of the Ada Joint Program Office agreed that insufficient documentation currently exists but believes that such documentation will become available as DOD implements programs that use the language.

Organizational Costs

As discussed in chapter 1, DOD established three organizations whose primary focus is on Ada or on implementing new software engineering methods in DOD programs: (1) the Ada Joint Program Office, to control the Ada language standard and facilitate its use throughout DOD; (2) the Software Technology for Adaptable, Reliable Systems Joint Program Office, to advance the state of the art of software engineering technology; and (3) the Software Engineering Institute, to smooth the transition of new software engineering technology into use. Funding for these three organizations, since their inception through fiscal year 1988, totalled about $201 million.

Table 3.1: Costs by Organization

Dollars in millions

	Year							
	1982	**1983**	**1984**	**1985**	**1986**	**1987**	**1988**	**Total**
Ada Joint Program Office	$7.0[a]	$7.6[a]	$7.8[a]	$8.0[a]	$6.9[a]	$6.7[a]	$15.7[b]	**$59.7**
Software Technology for Adaptable, Reliable Systems Joint Program Office			4.9[a]	12.6[a]	27.0[a]	24.4[b]	24.6[b]	**93.5**
Software Engineering Institute				5.0[a]	8.9[a]	15.2[a]	18.9[b]	**48.0**
Total	**$7.0**	**$7.6**	**$12.7**	**$25.6**	**$42.8**	**$46.3**	**$59.2**	**$201.2**

[a]Actual expenditures.

[b]Estimated funding available.

As shown in table 3.1, costs for these organizations, in total, have been rising since fiscal year 1982.

Project Costs

The military services have financed projects designed to study or demonstrate the feasibility of using Ada in specific computer applications or to develop the tools needed to use Ada effectively. Of the 100 DOD projects we analyzed, 23 focused on objectives specifically related to evaluating the use of Ada in computer applications. The costs of these projects, as shown in table 3.2, total about $190 million.[1]

Table 3.2: DOD Projects Focusing on Ada Studies, Demonstrations, and Software Development Tools

Dollars in thousands

Type of project	Department	Number of projects	Actual/ estimated cost
Studies	Air Force	4	$680
	Army	1	400
Subtotal		**5**	**1,080**
Demonstrations	Air Force	11	49,390
Subtotal		**11**	**49,390**
Software development tools	Air Force	4	23,730
	Army	2	35,910
	Navy	1	79,700
Subtotal		**7**	**139,340**
Total		**23**	**$189,810**

[1]DOD officials stated that the Ada Joint Program Office and the Software Technology for Adaptable, Reliable Systems Joint Program Office often provide funding to the services for these types of projects. However, DOD could not identify how much of the $190 million was funded by these joint program offices.

Ada-Specific Costs for Developing Application Programs Are Unknown; Empirical Evidence of Cost Savings Is Limited

Our analysis of 100 DOD projects showed that 77 involved development of computer application programs for operational systems. While the total estimated cost of these projects exceeded $74 billion, we were unable to determine the Ada-specific costs associated with development of application programs for these projects. DOD's policy on the use of computer programming languages only requires the cost of using Ada to be determined when justifying the use of another language.

Although Ada-specific costs are not known, two studies have shown that the distribution of effort in a software development project involving Ada is different from approaches using other computer programming languages. For example, a study presented at the 1987 international conference of the Association for Computing Machinery Special Interest Group on Ada showed that a greater amount of effort is devoted to the requirements and design phase of Ada software development projects when compared with traditional software development projects. This increased initial investment is offset by a corresponding decrease in later phases, such as when writing and testing the code. Moreover, this study showed that designing reusable software and managing its reuse (that is, setting up and keeping libraries of reusable components current, making what's available known, and handling distribution) is expected to cost more than developing software that will be used one time. However, the study indicates, by reducing the amount of software developed for one-time use, total project cost savings from reuse will more than offset the increased costs associated with developing reusable software.

The director of the Ada Joint Program Office also believes that whatever additional costs might be incurred will be more than offset through life-cycle savings expected from the design of software modules that are reusable in other application programs, portable to other computers, and easily maintained.

Empirical Evidence of Ada Cost Savings Is Limited

We were unable to identify any DOD projects designed to assess the long-term benefits and cost savings expected from using Ada. Neither the director of the Ada Joint Program Office nor the Ada executives for the Air Force and the Navy are aware of any such projects. According to the Army Ada Executive, there are no such projects in the Army.

The Ada Board[2] believes that early actions taken by DOD to implement the use of Ada were a driving force on industry, leading to the development and maturation of Ada software development tools. Regarding the availability of empirical evidence supporting Ada's benefits, the director of the Ada Joint Program Office agreed that insufficient documentation currently exists for DOD programs. However, the director also believes that such documentation will become available as DOD programs experience the benefits accrued through the use of Ada. In addition, the director stated that there are several more global advantages to using Ada beyond the language itself, including the benefit to be realized through the use of a single language and the utilization of sound software engineering principles and practices throughout DOD.

In a December 1987 keynote address at an Ada exposition in Boston, Massachusetts, the Under Secretary of the Army expressed disappointment over a lack of documentation spelling out Ada's performance and cost savings over other languages. He further stated that he had yet to see convincing evidence of the language's ability to reduce the mounting software development and maintenance costs within DOD.

One of the projects we obtained information on is specifically designed to demonstrate an expected benefit of using Ada. The Common Ada Missile Package project, initiated by the Air Force in 1984, involves demonstrating the feasibility and value of developing reusable Ada software for missile applications. Existing missile flight software was examined to determine commonality, and about 450 stand-alone software packages, subprograms, or tasks were identified. Preliminary results of reusing this software in a simulated missile development project indicate a significant productivity increase in developing the software. This project, scheduled to be completed in 1990, is currently focusing on developing training materials and methods to facilitate the reuse of software in other projects.

[2]A federal advisory committee, composed of compiler developers, language designers, embedded system users, and government personnel chartered to advise the director of the Ada Joint Program Office.

Technical Issues Associated With the Use of Ada in DOD Program Applications

Five issues have affected the ability of DOD to effectively use the new Ada programming language. Two issues focus on the availability of software tools used to develop Ada application programs and on the performance quality of Ada compilers for use in projects. Two additional issues focus on the use of Ada with real-time systems that require rapid processing of data, and on real-time distributed systems in which several computers process data simultaneously. The remaining issue concerns the need for a standard approach when using Ada application programs with data base management systems.

The issue associated with availability of software development tools has abated in recent years as newer tools have been developed. The focus today is on improving the use of Ada features and on the quality of Ada compilers and other software to support those real-time and distributed computer applications where precise timing control, processing speed, and the size of computer programs are critical elements. Research is being done to develop ways to use these features effectively.

Although Ada can be used with data base management systems, there is currently no standard approach for an Ada application program to use the Structured Query Language approved by the American National Standards Institute. Several approaches are currently available, but a decision on a uniform approach is needed in order to achieve the full benefits of Ada's portability.

We discussed our assessment of all of these issues with eight experts in Ada and/or in modern software engineering practices (see app. III). Not all of the experts commented on each of the issues because their knowledge of certain technologies was not current or because their area of expertise did not cover certain of our technical issues. Those experts commenting on the individual issues, however, are generally in agreement with the above assessments of the issues.

Availability of Ada Software Development Tools Is Improving

Many software development tools are used to develop an application program. When the Ada language was developed, new tools had to be built to work with this language. There were few Ada programming tools available when DOD first endorsed the use of Ada for major programs in 1984. Since that time, however, a large variety of Ada tools have been developed by industry.

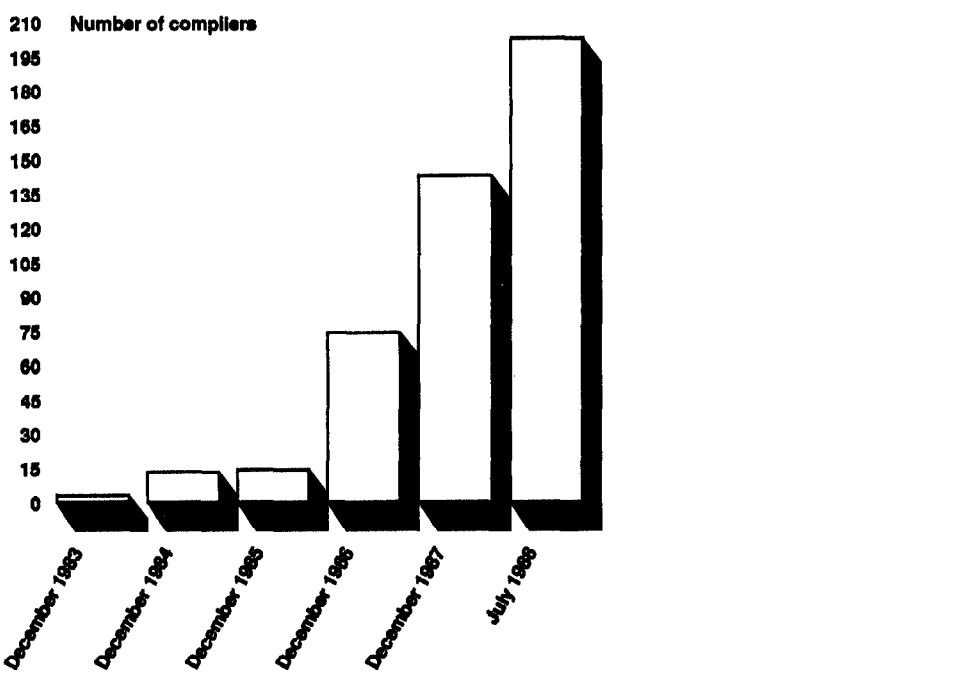
Ada Compilers

A compiler is an essential tool for writing an application program in a high-order language. Its purpose is to translate the high-order language code into machine language that can be executed by the computer.

DOD regulations require that an Ada compiler used to develop military computer programs be validated by the Ada Joint Program Office to ensure that the compiler's translation of Ada statements is in conformance with the language standard. The validation process currently consists of over 3,000 tests, which are updated every 18 months. Compiler validation certificates remain in effect for one year after termination of the test set used in the validation. Once a validation certificate expires, the compiler must be revalidated. However, once a validated compiler has been selected for use on a DOD project, it may be used throughout the life of the project and only needs to be revalidated if it is modified.

As shown in figure 4.1, the availability of validated Ada compilers grew slowly from 1983 to 1985, but has risen sharply in more recent years.

Figure 4.1: Growth of Validated Ada Compilers



The variety of Ada compilers now available permits Ada to be used on many different types of computers. There is, however, still a lack of validated compilers for some computers. For example, the Navy uses custom-built computers for its aircraft and shipboard systems that are not sold commercially. Because of this, private industry has not developed Ada compilers for these computers. As discussed in chapter 2, the Navy is now developing its own Ada compilers.

Five of the experts we consulted during this review commented on the availability of Ada compilers. Four of these experts believe that, except for the Navy's custom-built computers, the availability of Ada compilers is no longer a major problem for most DOD applications. However, one of the experts believes that the compiler availability problem is more widespread, particularly for embedded computers.

Other Software Development Tools

In addition to compilers, many other tools are used in developing an application program. The minimum tools needed include

- editors, to support a programmer in creating or modifying a computer program and its associated documentation;
- debuggers, to assist in detecting coding errors; and
- configuration managers, to help control changes to the program and its documentation.

Other tools may be required, depending on the particular software development project. For example, if the application program is being developed on one computer for use on a different one, necessary tools may include (1) a target simulator, a tool that simulates the target computer on the computer being used to develop computer application programs; and (2) a downloader, a tool that loads the application program on the target computer. In addition, many other tools, although not essential, are useful, such as code analyzers and documentation generators.

As with validated compilers, the availability of other software development tools initially grew slowly. A report by the Defense Science Board Task Force on Military Software showed that, on the basis of its 1985 work, a shortage of tools for developing Ada software existed. However, according to the Ada Board's response to the task force report, the situation has significantly improved and many vendors are now offering

tools for designing, controlling, documenting, testing, and maintaining Ada software.

Our review also showed that many tools are now available from commercial companies. In February and March 1988, we surveyed 16 companies that marketed 75 percent of the validated Ada compilers to determine the availability of tools that work with their compilers. We were told that, for the most part, all of the basic tools and many additional tools were available. Further, our analysis of the 100 DOD projects showed that, according to project officials, the availability of basic tools is no longer a major problem.

Seven of the experts consulted during our review commented on the availability of software development tools. Four of the seven experts believe that the availability of tools is no longer a major problem. Three of the seven experts, however, believe that tool availability is still a problem because many tools are not yet mature.

Better Tests Are Being Developed to Determine Compiler Performance

Although the compiler validation process assures that Ada compilers translate Ada statements in conformance with the language standard, it does not measure the compiler's "production quality;" that is, its ability to meet the performance criteria of a specific application program environment.

According to the Ada Adoption Handbook published by the Software Engineering Institute,[1] the production quality of an Ada compiler is usually measured in terms of

- compile time efficiency: the time it takes a compiler to translate Ada code (source code) into machine language (object code);
- object code efficiency: the size and speed of object code generated by a compiler, including the portion called run-time software, that provides supporting functions required to run a program on a target computer;
- compiler services: messages provided by a compiler to assist a programmer in writing a computer program. (Examples of such messages include listings showing the structure and control flow of source code, the memory location of each machine instruction, the machine code generated for each Ada statement, and explanations of programming errors found by the compiler.); and

[1]Foreman and Goodenough, pp. 28-31.

- support for embedded system requirements: special functions such as placing data in specific memory locations within computers, accessing specialized machine language instructions, and accomplishing required actions within precise time constraints.

For the most part, compilers, including Ada compilers, are not designed to optimize every production quality attribute. In selecting a compiler, tradeoffs are made between performance attributes and the project development requirements. For example, in the early stages of software development, compile time efficiency may be more important than object code efficiency because the program may be compiled many times before it is ready to run on the target computer. In later stages of the software development process, when the application program is ready for operational use, the object code efficiency may be more important than the time required to compile the program.

Several groups of tests have been developed by industry and professional associations that can be used to evaluate Ada compiler performance. According to software engineers at the Software Engineering Institute, these tests vary in size and quality and can be used to provide general evaluations. However, the collection of tests is not necessarily complete and, therefore, may not necessarily represent an adequate test of compiler performance. Currently, these engineers recommend that project managers use the best information available from one or more of these tests, or develop their own specific tests to assess the capabilities of competing compilers.

To alleviate this problem, the Air Force under the sponsorship of the Ada Joint Program Office is developing a comprehensive group of tests that will enable DOD program managers to compare the capabilities of different Ada compilers. The first version of these tests was released in 1988, and a second version incorporating comments from users is planned for 1989. Once successfully completed, DOD program managers will be able to use the test results to compare the performance of competing compilers and select the compiler that best satisfies their needs.

Promised Benefits of Ada Not Yet Achieved for Some Critical Real-Time Applications

Real-time processing pertains to the processing of data as it occurs and producing results quickly enough to affect the environment that produced the data. Examples of real-time systems include process control, target acquisition and tracking, and computer-aided navigation systems. High-order languages such as FORTRAN, JOVIAL, and CMS-2 have been used in the past to implement real-time applications, but have been less

effective for operations that require very fast or tightly controlled computer processing. As a result, these operations were written in assembly language. Ada was designed for use in real-time computer applications and is being used successfully in some applications. However, Ada has not worked well in critical real-time applications—applications that have severe time and memory constraints and/or require precise timing control. According to DOD, these problems involve both the Ada language and the compilers that implement the Ada language.

To deal with these difficulties the Institute's software engineers recommend the use of Ada whenever practical, but recognize that assembly language may be needed to implement the time-critical portions of applications, as is done with other high-order languages. Private industry and professional organizations have proposed other solutions, including changing the Ada language. Four of our experts believe that Ada's utility in critical real-time systems will increase as Ada compilers improve and mature. However, these experts also believe that some assembly code will always be required in severely constrained real-time applications.

Real-Time Computer Systems

Real-time computer systems require the capability to obtain data from an activity or process, perform computations, and return a response rapidly enough to affect the outcome of that activity or process. Depending on the application, the computer may be required to respond in seconds or in milliseconds. For example, in the World Wide Military Command and Control Information System, computers are required to respond to a user's request for information within a few seconds. On the other hand, in the Army's Cameo Bluejay project, computers will control attack and scout helicopters that search, detect, track, and jam enemy weapons. These computers are required to respond to a particular situation in less than four milliseconds.

Timing Control and Efficiency Problems With Using Ada for Critical Real-Time Applications

Ada was designed to provide the functionality needed in the development of real-time and embedded applications. Ada's tasking feature and the run-time executive provide many of these functions.

Ada's tasking feature provides the ability to specify multiple program units, called tasks, that can be executed concurrently. This feature was also designed so that specialized tasks could be developed to respond to interrupts in a structured manner. An interrupt is a mechanism that

forces a processor to take note of a special event; for example, the availability of high priority data that needs to be processed immediately. When an application receives an interrupt command during processing, it must suspend its current operations and respond to the special event. Upon completion of interrupt processing, normal processing continues. The ability to handle interrupts is an essential feature of real-time embedded computer systems.

The run-time executive is software provided by compiler vendors to perform important functions such as memory management and exception handling.[2] For general purpose computers, these functions are usually performed by an operating system.[3] Since many critical real-time applications are processed on embedded systems operating on computers that do not have this type of operating system support, the Ada language was designed to allow an Ada run-time executive as an integral part of an application program.

In some critical real-time applications, however, neither Ada's tasking feature nor the use of an Ada run-time executive have worked well. According to the Institute's handbook, the problems include both timing control and efficiency.

Timing control problems exist because the Ada language does not provide adequate mechanisms for controlling the timing of task scheduling. Difficulties have been reported in synchronizing the completion of tasks under certain conditions, and in executing tasks in order of priority. Also, an Ada run-time executive cannot effectively control the repeated execution of tasks in a sequential manner according to a fixed schedule. In general, critical real-time applications need some or all of these capabilities to perform their mission effectively. Appendix V contains a detailed discussion of the timing control problems associated with Ada's tasking feature and the use of an Ada run-time executive. This appendix also discusses an additional feature that may be needed in the Ada language to properly control tasking.

In addition to timing control problems, efficiency must also be considered when using Ada in critical real-time applications. For some critical

[2]Memory management is a process of allocating portions of memory to programs, and of keeping the programs separate from each other in memory. Exception handling is a process for handling events that cause suspension of normal program execution.

[3]An operating system controls the execution of programs and typically provides services such as resource allocation, program scheduling, input/output control, and memory management.

real-time applications, the run-time executive code, and the object code generated by the compiler currently are too large and do not run fast enough to meet performance requirements. The Institute's handbook reports that, for some critical real-time applications, the run-time executive takes too long to switch from running one task to another. The Institute's handbook also reports that these problems are not a reflection of defects in the Ada language, but are caused, instead, by the relative immaturity of currently available compilers and run-time executives. Further, there is evidence that the efficiency of code generated by compilers is improving.

Efforts to Improve Ada's Ability to Operate in Critical Real-Time Applications

Several approaches are being developed and evaluated by software engineers at the Institute, in private industry, and in professional organizations to address problems with implementing Ada in critical real-time systems. Some software engineers at the Institute agree that changes to the Ada language may be needed as a long-term solution to the timing control problems with the tasking feature and efficiency problems with the run-time executive. These changes would also include the establishment of instructions on how and when to use Ada's tasking feature in real-time applications. However, incorporation of such changes in the Ada language and the related instructions could take several years. In the near-term, these software engineers recommend that, if timing control or efficiency goals are not satisfied, the use of Ada should be avoided for certain small critical portions of computer applications, and that assembly language should be used instead. The Institute's handbook reports that, as compilers mature, the efficiency problems will diminish. These software engineers believe that most of Ada's benefits can still be achieved if the use of assembly language is limited and encapsulated in a few software modules.

With regard to improving the efficiency and reducing the size of Ada programs, the following actions have been suggested by either the Institute's handbook or by a working group of the Association for Computing Machinery: (1) Ada's run-time executive should be modified so that run-time system support is included for only those features actually needed in a specific application, and (2) compiler vendors should provide several versions of the run-time executive to meet different application requirements. According to the Institute, some compiler vendors have implemented the first suggestion.

Industry and private-sector institutions are also trying to solve these real-time computer application problems. Several companies are developing compilers and special-purpose hardware to resolve some of these problems, and institutions are performing research on real-time issues to better understand both the problems and the solutions.

Expert Views of the Real-Time Problem and Its Relationship to Ada

Of the seven experts who commented on this issue, six agreed that problems exist in using Ada in critical real-time systems. Four of the six experts believe that as better compilers are built, faster and more efficient computer programs will be produced, including the code necessary to implement the Ada tasking feature. Two of the six experts believe, however, that Ada's tasking feature will never be efficient enough to be used in critical real-time systems. A seventh expert did not believe that using Ada in critical real-time applications is a major problem because Ada compiler technology is sufficiently mature and the size and speed of the object code are sufficient for most applications.

Ada Features Hold Promise for Use in Real-Time Distributed Systems

Real-time distributed computer systems are decentralized: several interconnected (networked) computers process data simultaneously to jointly accomplish a mission. The computers may be dissimilar (e.g., different manufacturers and operating characteristics), and may be either widely dispersed geographically (as in the World Wide Military Command and Control Information System) or housed in one facility (as in a fire control system aboard a ship).

Five of the experts we consulted believe that the characteristics of real-time distributed systems are not yet fully understood, and that building such systems is difficult regardless of the language used. One expert said, for example, that

"The area of distributed systems is quite complicated, with many variations based on whether there is shared memory, the kind of communication, the handling of fault tolerance,[4] as well as other issues. No one language contains the features needed to support this wide variety of distributed environments."

There are no features in the Ada language, or any other high-order language, specifically designed for the development of real-time distributed systems. However, once problems with the tasking feature and run-time

[4]The capability of a computer system to continue to process an application even when the system is experiencing some operational problems.

executive are resolved, Ada could be useful in developing real-time distributed systems. For example, Ada's tasking feature could be used to partition a program so that it can be distributed among various computers. As stated by one expert, Ada's tasking feature is appropriate for some real-time distributed applications, principally those in which computers share memory with each other. In addition, an aspect of Ada's tasking feature called rendezvous could, if improved, handle some of the communications among distributed computers. Appendix VI contains a detailed description of the problem with the rendezvous feature for this type of communication, as well as additional Ada language features that could be added to assist in building distributed systems.

Using Ada to build distributed systems is currently the subject of independent research and development. These efforts include investigating how to (1) structure a distributed system using Ada (i.e., how to distribute functions across processors), (2) communicate among processors (i.e., using alternatives to Ada such as another language entirely), and (3) ensure that processing continues in the face of partial hardware failure (e.g., when one or more of the processors ceases to operate).

Full Ada Benefits Cannot Be Achieved Without a Uniform Approach to Using Ada With Data Base Management Systems

A data base is an organized collection of data that can be used by a variety of applications. It is controlled by a data base management system—a computer program that organizes, catalogs, stores, and retrieves information in the data base. An application program interacts with the data base management system to gain access to and retrieve information. Since Ada is now the language of choice by DOD, many data base application programs are being written in Ada. Examples of such programs include the Army's Standard Finance System Redesign—an installation-level accounting system, and the Navy's Submarine Satellite Information Exchange Subsystem—a message-processing system used to communicate with submarines.

Application programs written in Ada can interface with data base management systems written in other languages. However, no standard method has yet been established for such an interface. Several approaches have been devised, but a consensus is lacking on a standard approach both in the data processing community at large and among the experts we consulted. Consequently, problems exist in achieving some of the benefits of Ada in applications that interface with data base management systems. The Software Engineering Institute began work in March 1988 to develop a standard interface that would be acceptable to both the data base and Ada communities.

Data Base Management Systems

A data base management system controls the storage and retrieval of information in a computer system much as a librarian controls the documents in a library. The system manages the physical storage of and access to information, and provides the user with a convenient means to access that information. For example, the user of an airline reservation data base management system might ask "How many confirmed reservations have been made on a flight?" and not be concerned about how flight reservation files are organized in the system, how the information is formatted, what physical devices hold the information, or how to read the information on these devices.

The English-like query language used with data base management systems provides users with a simple, yet powerful, means to access and manipulate data. While some data base management systems offer vendor-unique query languages, the American National Standards Institute has endorsed one language, the Structured Query Language (SQL), which can be used for communicating with many data base management systems. The rationale for using a standard query language is analogous to that for using a standard programming language such as Ada: increased portability of software from computer to computer, increased ability to share software among users, decreased user training difficulty and expense, and decreased software maintenance costs. DOD, therefore, has focused on formulating approaches for interfacing Ada and SQL so that a user in one program may, for example, conveniently code algorithmic-type functions in Ada and data base management functions in SQL.

Methods for Implementing the Ada and SQL Interface

There are four primary methods for an Ada and SQL interface that have been proposed by either the American National Standards Institute, software engineers at the Software Engineering Institute, or the World Wide Military Command and Control Information System modernization program. One of these proposed methods may be acceptable to the Ada community with a few technical changes; the second violates the Ada language standard; the third is more difficult to use and maintain; and the fourth violates the SQL standard. All of these proposed methods include unresolved technical issues.

Ada and SQL Module Language Interface Method Proposed by the American National Standards Institute

In December 1987, the American National Standards Institute proposed extending the SQL standard to include an interface to Ada modules. Under this approach, a programmer would write programs that consist of both Ada and SQL modules, with the Ada modules and the SQL modules being separately compiled prior to execution. The advantages of this

approach include (1) use of a standard Ada and standard SQL, (2) portability, since both Ada and SQL run on many different computers, and (3) complete separation of SQL and Ada code, so that modules can be written and maintained separately. One disadvantage to this method is that there are currently very few SQL module compilers to work with this approach.

In May 1988, a special data base committee formed by the Software Engineering Institute proposed certain technical changes to the American National Standards Institute's proposal to better support the Ada programming language. The suggested changes deal primarily with the way data types are specified in Ada, and the way certain external procedures are referenced in Ada. If these changes are adopted, this special committee believes this proposal would be acceptable to the Ada community as a first step toward developing an Ada and SQL interface.

Ada and SQL Embedded Ada Interface Method Proposed by the American National Standards Institute

In December 1987, the American National Standards Institute also proposed a second method of interfacing SQL with high-order languages, including Ada. Under this approach, SQL statements are incorporated into the text of Ada application programs. Prior to compilation, a preprocessor removes the SQL statements, places them in a separate SQL module, and inserts the required Ada statements in the Ada program to link the Ada and SQL modules.

This approach has the same advantages as the first approach discussed above; however, the source code used to maintain the system usually consists of both SQL and Ada statements. Because of this mixture, this approach has generated considerable opposition from members of the Ada community, including the Ada Board. This approach is considered by the Ada Joint Program Office and the Ada Board to be a violation of the Ada standard because the application program would contain statements that are not defined in the Ada language. Additionally, the director of the Ada Joint Program Office has advised the American National Standards Institute that the use of a preprocessor jeopardizes the achievement of Ada's intended benefits of being readable and easily maintained.

Ada and SQL Interface Method Proposed by the Institute's Software Engineers

Software engineers at the Institute suggest a third method to interface Ada and SQL to comply with both the Ada standard and the proposed SQL standard. Under this method, called the pseudo-module approach, computer program modules containing embedded SQL are written in another

programming language, such as "C," and these modules are accessed as needed from an Ada program. This approach has the same advantages as the first approach discussed above. In addition, neither the Ada nor the SQL standard would be violated. The potential problem with this approach is that three languages would be needed instead of two, making the code more difficult to write, maintain, and transport among computers.

Ada and SQL Interface Method Proposed by the World Wide Military Command and Control Information System Modernization Program

A new query language, called Ada/SQL, has been developed by the World Wide Military Command and Control Information System modernization program. Statements in Ada/SQL have been designed to appear as similar to SQL as possible. All data elements in a data base are defined using Ada's data-definition rules. An advantage of this approach is that it allows the full use of Ada's functionality and data base query capabilities in a single language.

This Ada/SQL approach has several drawbacks. Despite its SQL-like appearance, Ada/SQL does not adhere to the SQL standard. Some of the SQL key words have been changed because they conflict with Ada's reserved words. For example, the word "select" in SQL means retrieve a data element from a data base. In Ada, "select" is used to specify which of several operations will be performed in a task. Therefore, the Ada/SQL approach uses the word "selec" to retrieve a data element from a data base, which is not standard SQL. According to software engineers at the Software Engineering Institute, one reason for SQL's popularity is that a variety of very powerful tools have been developed for use with this language. However, none of these tools will work with this new Ada/SQL approach. Since this is a new query language, vendors would have to develop and bring to maturity a completely different set of tools for Ada/SQL. As a result, until this new query language receives broad vendor support, portability across data base management systems is questionable.

Other Technical Conflicts Between Ada and SQL

A December 1987 technical report on interfacing Ada and SQL by the Software Engineering Institute also pointed out that there are a number of additional conflicts between the two languages. These conflicts, which will not be resolved simply by choosing an existing Ada and SQL interface, include the following:

- differences in the variety of data types and in the use of subtypes;

- differences in the definitions, assigned values, and operations performed on the various data types;
- differences in error-handling mechanisms;
- language design differences that affect the frequency with which programs must be recompiled after changes are made; and
- implementation concerns that affect program portability and reliability.

Without a solution to these technical conflicts, the benefits of both Ada and SQL may not be fully realized.

Research on the Ada and SQL Interface

The Software Engineering Institute began work in March 1988 to develop an acceptable methodology for implementing an Ada and SQL interface. This effort was requested and funded by the Ada Joint Program Office. The Institute is modifying and extending the pseudo-module approach previously discussed, developing standard interface packages, and developing solutions to the technical conflicts discussed above. In October 1988 the Institute published an interim report on this project to solicit comments on proposed guidelines for implementing a pseudo-module approach for interfacing Ada and SQL. Within a year, the Institute plans to test its methodology on an ongoing Army project.

Expert Views of the Data Base Problem and Its Relationship to Ada

The eight experts with whom we discussed this issue agreed that a problem exists in interfacing Ada with SQL when using data base management systems. Four believe it is an Ada language problem, two believe that the problem rests with the conflicting standards, one believes it is a lack of experience in working with the two languages, and one expressed no opinion. All of the experts believe that the problem can be resolved, but disagree as to the best solution. Three experts believe that either the Ada or the SQL language standard should be changed, two believe that more research is needed before a solution is developed, and three did not provide an opinion.

Conclusions and Recommendations

Conclusions

DOD's goal of using a standard programming language to develop software that is reusable in different applications, portable among a variety of computers, and easily maintainable is commendable. Such an approach offers the promise of reducing the enormous and increasing cost of developing and maintaining the software that is vital to accomplishing DOD's mission and support needs. This approach has been supported by the Software Engineering Institute and the Defense Science Board.

Reliance by DOD on Ada or any new language as a standard to support all computer systems—weapons systems, mission-critical systems, and information management systems—carries with it risks. Programming languages and their associated software development tools are complicated. It takes time and considerable experience using them in a variety of applications before they mature, and most experts agree that Ada has not yet matured. The risk of using Ada in DOD is heightened by the diversity of applications, computers, and software development tools necessitated by Defense programs.

DOD's experience in using Ada is limited. Whether the use of Ada will result in the production of software that is reusable, portable, and more easily maintained is not yet known. This uncertainty is further complicated by the lack of a complete inventory of projects written in Ada that would facilitate the sharing of experiences with the language.

It is also uncertain how effective Ada will be in certain applications. DOD's experience to date has shown that Ada currently has limitations in real-time applications that require precise timing control, very fast processing speed, and compact computer programs. Such limitations also inhibit its usefulness in developing distributed systems. Finally, although Ada can be used in data base management systems, no standard currently exists for Ada to interface with SQL, a query language used in many of these systems. Such a standard is needed to achieve the full benefits of Ada's portability. While research is currently underway to solve these problems, experts are divided as to whether these problems are inherent in the Ada language or whether they can be solved as the language matures. This lack of definitive information reduces management's ability to make informed decisions on the evolution of the language and its use as a standard within DOD.

Because of DOD's reliance on Ada to support its missions and the uncertainties associated with the use of the language, the Secretary of Defense needs sound advice on courses of action to deal with Ada-

related issues. The stakes are high, and decisions must be as well-researched and informed as possible. We believe that such advice could best be provided by a high-level body of independent experts on the Ada language and software engineering technology.

Recommendations to the Secretary of Defense

We support DOD's continued research focusing on resolving technical issues associated with the use of Ada in real-time, distributed, and data base applications. To more fully develop information that will allow more informed judgments on the use of Ada, we recommend that the Secretary of Defense direct the Ada Joint Program Office to

- develop performance data that demonstrate whether DOD's use of Ada is achieving its goals;
- develop a DOD-wide repository of computer applications and modules written in Ada, and make them available for reuse; and
- gather and disseminate complete lists of all DOD projects using Ada.

We further recommend that the Secretary of Defense establish a committee of independent experts on Ada and software engineering technology to monitor and periodically report to the Secretary on DOD's actions to implement Ada. Specifically, we recommend that this committee

- assess existing projects and propose additional projects if necessary to demonstrate the intended cost savings associated with using Ada;
- assess existing research efforts and identify where there is a need for further research to overcome the technical problems in using Ada in real-time, distributed, and data base applications;
- assess the progress and results of the Ada Joint Program Office in developing a repository of software written in Ada; and
- recommend appropriate courses of action in employing Ada.

Agency and Contractor Comments and Our Evaluation

In commenting on our draft report, DOD either fully or partially concurred with all of our recommendations and findings (see app. VII). Specifically, DOD concurred with our recommendations to (1) develop performance data that demonstrates whether DOD's use of Ada is achieving its goals and (2) gather and disseminate complete lists of all DOD projects using Ada.

DOD partially concurred with our recommendation to develop a DOD-wide repository of computer applications and modules written in Ada, and to make them available for reuse. DOD stated that repository technology

must be developed prior to establishing such a repository. DOD noted that experiences with existing software repositories demonstrate that there are numerous procedural, contractual, and technological issues that must be resolved before establishing an effective repository. While we did not review existing repository operations, we agree that appropriate technology should be developed to achieve a usable DOD-wide repository. Such a repository can facilitate sharing of computer programs and modules written in Ada, and can help DOD achieve the cost savings anticipated when it adopted Ada as its language of choice for use throughout the Department.

DOD partially concurred with our recommendations to establish a committee of independent experts on Ada and software engineering technology to assess (1) projects demonstrating intended cost savings associated with Ada's use, (2) research efforts to overcome technical problems with specific applications, and (3) progress in developing inventories of Ada software; and to recommend appropriate courses of action in employing Ada. DOD agrees that these actions would be beneficial; however, it does not agree with the need to establish an independent committee of experts. DOD believes that such monitoring and oversight could best be accomplished by the Ada executives in each of the military services, the Ada Board, and/or by the three DOD organizations discussed in this report.

We disagree with DOD's position that the responsibility for accomplishing these recommendations should be delegated to the services or to existing DOD organizations. Each of the actions recommended have global significance, impacting on all DOD components. Because of this wide-spread impact and because DOD has placed considerable reliance on Ada to accomplish its missions, we believe that issues impacting DOD's ability to fully implement Ada warrant monitoring by a high-level committee of experts that is independent of the services and other DOD organizations. We further believe that recommendations to the Secretary on appropriate courses of action in employing Ada would best be based on high-level independent assessments. DOD stated that the Ada Board is chartered to provide such recommendations; however, as noted in chapter 3, this board is only chartered to advise the director of the Ada Joint Program Office.

In our draft report, we recommended that the independent committee assess existing projects and propose additional projects to demonstrate the intended cost savings associated with using Ada. DOD partially concurred with this recommendation. DOD stated that a sufficient number of

DOD programs using Ada currently exist that can be assessed regarding their individual cost savings. We believe DOD's response to our draft report is inconsistent on this matter. As we noted in chapter 3, our review did not identify any DOD projects designed to assess the long-term benefits and cost savings expected from using Ada. In this regard, neither the director of the Ada Joint Program Office nor the Ada executives in each of the military services were aware of any such projects. Moreover, in commenting on chapter 3, DOD concurred with this finding. Since DOD has not identified any specific projects suitable for this assessment, we cannot validate DOD's position on this recommendation. Nevertheless, to more fully address this issue, we have modified this recommendation. We now recommend that the independent committee initially assess existing projects and propose additional projects, if necessary, to demonstrate the intended cost savings associated with use of the Ada programming language and modern software development methods.

The Software Engineering Institute also commented on a draft of this report in November 1988 (see app. VIII). The Institute generally agreed with our recommendations and findings.

Despite DOD's and the Institute's general concurrence with this report, there were a number of technical corrections suggested by these organizations. On the basis of comments received from both DOD and the Institute, changes have been made in the report where appropriate. There are, however, some areas where we do not agree with DOD and/or the Institute that our report is incorrect or misleading.

Both DOD and the Institute believe that the inclusion of all of the Institute's funding in this report is misleading. In their comments, they stated that it is incorrect to include all of the Institute's funding because only a small portion of the Institute's efforts are directly related to Ada. We believe that the Institute's total funding should be presented. While many of the Institute's activities are focused on general software engineering objectives, Ada is the primary language used by the Institute in pursuit of these objectives. DOD anticipates reduced software life cycle costs to result from using both (1) the Ada language and (2) the new methods of software development that Ada supports. Our report addresses the funding provided to DOD organizations for both of these activities. Moreover, the funding data is discussed in this report in terms of organizational costs, not as direct Ada costs.

The Institute also commented that our report suggests that it has a direct responsibility for Ada. We disagree with the Institute's assessment. Our report (see pp. 10, 11, and 21) acknowledges the differences in the roles of the three DOD organizations discussed in this report. Specifically, we acknowledge that the activities of the Institute, as well as the Software Technology for Adaptable, Reliable Systems Joint Program Office, are fundamentally related to improving software engineering technology.

DOD and the Institute also commented that problems with using Ada for critical real-time processing primarily result from inefficient tools. We agree that some of the problems may be related to inefficient tools, and we acknowledge in chapter 4 that the efficiency of such tools is improving. We are not convinced, however, that maturity of software development tools, by itself, will resolve problems associated with the use of Ada for all critical real-time processing applications. Chapter 4 and appendixes V and VI of this report discuss changes in the Ada language that experts believe may be needed to resolve some of these problems. While the Institute believes that there are no shortfalls in the Ada language that prohibit its use in critical real-time applications, DOD acknowledges that there are Ada language issues that need to be resolved. Such issues are now being addressed through the normal Ada language standard revision process.

Both DOD and the Software Engineering Institute also commented that our report's discussion of Ada's technical issues unduly criticized the language itself. Specifically, DOD commented that our interpretation of the facts concerning the technical issues associated with using Ada was unduly negative, in some instances. In this regard, DOD pointed out that the technical difficulties we identified with using Ada are primarily associated with specific implementations of the language, rather than with the language itself. As discussed in our report, we recognize that it takes time and considerable experience using a new programming language in a variety of applications before it matures, and most experts agree that it has not yet matured. DOD acknowledged that there were technical issues associated with the use of Ada that need to be resolved.

The Software Engineering Institute also commented that our report seems somewhat prejudiced by the tone of a question we addressed; that is, the Chairman's request to identify the issues associated with Ada's use. We were asked to identify the technical issues associated with the use of Ada because of the Chairman's concerns over Ada's lack of an

extensive performance history and the problems some Defense programs have apparently experienced in using Ada. We believe that our report accurately presents these issues, and the Institute agreed that our report correctly portrays the technical issues in using Ada.

Request Letter

Congress of the United States
House of Representatives
Committee on Appropriations
Washington, DC 20515

June 4, 1987

Honorable Charles A. Bowsher
Comptroller General of the United States
General Accounting Office
Washington, D.C. 20548

Dear Mr. Bowsher:

I am requesting that the General Accounting Office (GAO) review the Department of Defense's (DOD) use of the Ada programming language for major automated information system acquisition programs. DOD has selected Ada as the standard programming language for its most critical automated information systems. Ada, however, does not have an extensive performance history, and some major DoD programs that have used Ada have apparently experienced significant problems.

DoD's use of Ada is intended to ensure the use of modern software engineering principles that improve software quality and reduce automated information system life cycle costs. These objectives are laudable. The Subcommittee, however, is concerned about the level of risk in DoD's increasing reliance on the as yet unproven Ada programming language for mission critical system acquisitions. I therefore request that the GAO (1) review the current and planned uses of the Ada programming language in the DOD, and (2) identify problem areas associated with the use of Ada for such systems.

I would appreciate it if your staff would keep my Subcommittee periodically informed of the progress of this review. Questions for hearings on the fiscal year 1989 budget also should be prepared based on your review, and delivered to the Subcommittee in January of 1988. At that time a date can be set for your formal report. Mr. Bruce Meredith of the Subcommittee staff will be your contact on this assignment.

Sincerely,

Bill Chappell, Jr.
Chairman
Subcommittee on Defense

Locations of DOD Projects Included in This Report

Information on the 100 DOD projects using or planning to use Ada included in this report was obtained at the following locations:

Air Force Air Defense Weapons Center, Tyndall Air Force Base, Florida.

Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio.

Air Force Logistics Command, Air Force Logistics Center, Tinker Air Force Base, Oklahoma.

Air Force Systems Command, Aeronautical Systems Division, Wright-Patterson Air Force Base, Ohio.

Air Force Systems Command, Armament Division, Eglin Air Force Base, Florida.

Air Force Systems Command, Armament Laboratory, Eglin Air Force Base, Florida.

Air Force Systems Command, Electronic Systems Division, Hanscom Air Force Base, Massachusetts.

Air Force Systems Command, Foreign Technology Division, Wright-Patterson Air Force Base, Ohio.

Air Force Systems Command, Human Systems Division, Brooks Air Force Base, Texas.

Air Force Systems Command, Rome Air Development Center, Griffiss Air Force Base, New York.

Army Armament, Munitions and Chemical Command, Aberdeen Proving Ground, Maryland and Picatinny Arsenal, New Jersey.

Army Aviation Systems Command, Fort Monmouth, New Jersey and St. Louis, Missouri.

Army Communications and Electronics Command, Fort Belvoir, Virginia and Fort Monmouth, New Jersey.

Army Information Systems Command, Fort Belvoir, Virginia.

Army Missile Command, Redstone Arsenal, Alabama.

Army Tank and Automotive Command, Warren, Michigan.

Army Training and Doctrine Command, Fort Hood, Texas.

Naval Air Development Center, Warminster, Pennsylvania.

Naval Air Systems Command, Washington, D.C.

Naval Sea Systems Command, Washington, D.C.

Naval Training Systems Center, Orlando, Florida.

Space and Naval Warfare Systems Command, Washington, D.C.

Ada And/Or Software Engineering Experts Interviewed

Dr. Victor Basili, Professor, Department of Computer Sciences, University of Maryland.

Dr. Barry Boehm, Chief Scientist, TRW Incorporated, Redondo Beach, California.

Mr. Grady Booch, Director of Software Engineering Programs, Rational, Lakewood, Colorado.

Dr. Edsger W. Dijkstra, Professor, Department of Computer Sciences, University of Texas at Austin.

Dr. Jean D. Ichbiah, Chief Designer of the Ada Language, President, Alsys Incorporated, Waltham, Massachusetts.

Mr. Philip Kiviat, Vice President, Sage Federal Systems, Incorporated, Rockville, Maryland.

Dr. David L. Parnas, Professor, Department of Computing and Information Science, Queens University, Kingston, Ontario, Canada.

Ms. Jean E. Sammet, Programming Language Historian, Bethesda, Maryland.

Summary of DOD's Ada Projects Included in This Report

Dollars in millions

Project name and description	Project type
Air Force Projects	
Advanced Alignment Concepts To develop and demonstrate an integrated wing flexure mode/transfer alignment/ calibration filter for aircraft.	Demonstration
Advanced Processor Technology for Air-to-Air Missiles To develop and demonstrate an advanced missile and data processing system capable of meeting requirements of future multi-mode missile seekers.	Demonstration
Air Force Armament Laboratory Ada Compiler To develop an Ada compiler.	Tool
Autonomous Synthetic Aperture Radar Guidance To develop a new all-weather radar guidance seeker.	Demonstration
Common Ada Missile Packages To demonstrate reusable Ada software modules for missiles, and to develop a system for identifying available modules for specific applications.	Demonstration
Estimation and Guidance Study In-House Effort Exploratory development for research of target state estimators (position, velocity, and acceleration).	Study
Guidance Instruction Set Architecture To design, develop, and demonstrate a 32-bit instruction set architecture for tactical systems, that is optimized for guidance and control of weapons.	Demonstration
Have Slick To develop and demonstrate new technologies for flight control software that will perform guidance, navigation, and control functions.	Demonstration
Optimal Guidance Law Implementation To develop guidance algorithms and Kalman filters for tactical missile guidance systems.	Demonstration
Tactical Ada Guidance To investigate Ada's applicability for operation in a real-time tactical embedded environment.	Study
Ada Target Sensor Subsystem To investigate requirements for target sensor subsystems and the need for these requirements to be included in criteria for an Ada compiler. This study was performed by a contractor at no cost to the government and no code was written.	Study
Interactive Ada Workstation To develop prototype software and documentation to enhance the productivity of Ada programmers.	Demonstration
Ada Radar Model To study the feasibility of using Ada as a programming language for engineering-level simulation.	Study
Production Quality Ada Compiler System To develop a production quality Ada compiler system.	Tool

Cost		Lines of code[a]		Other languages used	Project status	Actual or estimated completion date	Types of processing environments		
Fiscal year 1989	Total project	Total	Ada				Real-time	Data base	Distributed
$0.00	$0.37	1,000	1,000		In development	1988	Yes	No	No
0.00	5.00	8,500	8,075	Assembly	In development	1989	Yes	No	No
0.00	0.88	80,000	0	PASCAL	Completed	1985	Yes	No	No
1.64	15.00	[b]	12,000	Assembly, "C"	In development	1990	Yes	No	No
0.40	4.23	43,685	43,675	Assembly	In development	1990	Yes	No	No
0.00	0.03	5,000	5,000		In development	1989	Yes	No	No
0.50	1.05	[b]	[b]		In development	1989	Yes	No	No
2.53	8.00	5,000	[b]	Assembly	In development	1989	Yes	No	No
0.00	0.96	1,750	1,750		In development	1988	Yes	No	No
0.00	0.35	3,034	3,034		Completed	1985	Yes	No	No
0.00	0.00	0	0		Completed	1987	Yes	No	Yes
0.00	1.91	1,000,000	0	LISP	In development	1988	No	No	No
0.00	0.30	3,000	3,000		Completed	1987	No	No	No
1.53	12.55	420,970	414,604	Assembly, FORTRAN, VAX DCL	In development	1990+	No	Yes	No

(continued)

Project name and description	Project type
Ada Compiler Evaluation Capability System To develop a series of tests which will enable users to compare the capabilities of different Ada compilers.	Test system
Ada Based Integrated Control System To integrate several aircraft functions, such as flight control, weapon delivery, and navigation into a total vehicle management system.	Demonstration
High Order Language Electronic Warfare Software Analysis To study and demonstrate Ada use for electronic warfare software applications.	Study and demonstration
Advanced Tactical Fighter To develop a new fighter aircraft.	Weapon
Integrated Electronic Warfare System A joint Air Force and Navy program to define, design, and demonstrate new technology for the next generation of electronic warfare for combat aircraft.	Demonstration
Ada Surface-to-Air Missile SA8 To develop a surface-to-air missile simulation capability coded in Ada.	Simulator
Air Force Institute of Technology Research Concept for Ada Development To develop a prototype Ada programming support environment for student use. This project was completed by students at the institute at no cost.	Tool
Microwave Landing System To develop equipment for use on civilian and military aircraft that will use ground signals to display the aircraft's position during landing.	Aircraft landing system
Granite Sentry To modernize supporting computers in the Air Defense Operational, Resource, and Weather centers of the North American Air Defense command.	Command and control
North Atlantic Defense System-Iceland Air Defense System To develop new radars and communications systems for air defense centers. This is a joint project with the North Atlantic Treaty Organization. The United States will fund $32.7 million of the total cost.	Command, control and communications
Sentinel Aspen To modernize intelligence training systems.	Training system
Joint Surveillance Target Attack Radar System Joint Air Force/Army program to develop a new airborne radar and command, control and communications systems. Ada will be used for the self defense suite subsystem.	Command, control and communications
Survivable Communications Integration System To develop survivable communications, message processing, and support systems for communications between sensor sites, the North American Air Defense command, and forward commands.	Communications
Military Airlift Command Information Processing System To develop an upgraded command and control system.	Command and control

Cost		Lines of code[a]		Other languages used	Project status	Actual or estimated completion date	Types of processing environments		
Fiscal year 1989	Total project	Total	Ada				Real-time	Data base	Distributed
$0.20	$0.96	36,614	33,788	MEDIAN	In development	1989	No	No	No
0.00	11.99	[b]	[b]	Assembly, JOVIAL, PASCAL	In development	1988	Yes	No	No
0.00	0.88	16,900	16,900		Completed	1988	Yes	No	No
702.00	12,952.00	[b]	[b]		In development	1996	Yes	Yes	Yes
75.90	[b]	73,000	73,000		In development	1988	Yes	No	Yes
0.00	0.14	40,000	40,000		Completed	1986	No	Yes	No
0.00	0.00	6,000	most	"C"	Completed	1986	No	Yes	No
25.10	132.00	[b]	[b]	PLM–86	In development (Ada use is in planning phase)	1998	Yes	No	No
31.20	200.60	335,000	335,000		In development	1993	Yes	Yes	Yes
0.00	380.00	300,000	300,000		Planned	1993	Yes	No	No
4.11	21.70	505,000	450,000	Assembly, FORTRAN	In development	1988	Yes	Yes	Yes
238.30	574.20	[b]	[b]		In development (Ada use is in planning phase)	1993	Yes	No	Yes
30.20	122.25	109,000	82,000	"C"	In development	1992	Yes	No	Yes
28.40	231.70	131,000	131,000		In development (Ada use is in planning phase)	1993	No	Yes	Yes

(continued)

Project name and description	Project type
Command Center Processing and Display System - Replacement To develop a new ballistic missile tactical warning command and control system for the North American Air Defense command.	Command and control
Space Defense Initiative Ballistic Missile Command and Control - Experimental Prototype To develop a simulation capability to evaluate alternative ballistic missile command and control systems.	Simulator
Transmit Processor To develop new transmitters and receivers for the Minimum Essential Emergency Communication system.	Communications
Miniature Receive Terminal To develop miniature receivers for use on strategic bombers for the Minimum Essential Emergency Communication system.	Communications
World Wide Military Command and Control System Information System To modernize data collection and processing systems.	Command and control
Advanced Training System To develop a technical training support system for course development, instruction, and resource management.	Trainer
Instructional Support System Integration, Transition, and Technology Support To develop a computer based training system. Two versions are being developed and will later be merged. The first version is operational and the second version is in the prototype stage.	Trainer
E-4B Message Processing System To develop a system to combine, display, and print information from communication systems on aircraft.	Communications
F-15 Maintenance Trainer To develop a system to train F-15 maintenance personnel.	Trainer
Tactical Air Command Weapon System Evaluation Program This is a continuing program to evaluate air-to-air missiles. Ada was used in two projects to develop systems to monitor the status of missiles and to display data from a data base of past shots.	Test
Digital Airborne Intercom Switching System To develop a communication switching system.	Communications
Ada Compilation System To develop an Ada compiler and run-time system.	Tool
Army Projects	
Regency Net To develop a high frequency radio communications system.	Communications
Advanced Field Artillery Tactical Data System To develop an integrated battlefield management and decision support system for deep battle and light infantry divisions.	Command and control
Army Test Program Set Support Environment To develop software tools that integrate management and engineering support into one comprehensive interactive framework.	Tool
Maneuver Control System To develop a tactical command and control system.	Command and control

Cost		Lines of code[a]		Other languages used	Project status	Actual or estimated completion date	Types of processing environments		
Fiscal year 1989	Total project	Total	Ada				Real-time	Data base	Distributed
$65.70	$264.10	428,700	428,700		In development	1992	Yes	Yes	Yes
1.70	86.50	365,000	345,000	FORTRAN, "C"	In development	1991	Yes	Yes	Yes
4.90	26.00	10,000	9,000	Assembly	In development	1988	Yes	No	No
1.50	129.60	36,700	26,200	Assembly	In development	1991	Yes	No	No
66.36	713.76	4,000,000	4,000,000		In development	[b]	Yes	Yes	Yes
3.43	45.05	300,000	300,000		In development	1989	Yes	No	Yes
0.40	3.40	350,000	345,000	Assembly	In development	1989	No	Yes	No
0.00	20.60	98,500	98,500	Assembly	In development	1991	Yes	No	No
0.00	0.17	3,500	3,150	Assembly	Completed	1986	Yes	No	No
8.00	[c]	45,000	[b]	COBOL, FORTRAN	In development	[c]	Yes	Yes	No
19.80	200.00	100,000	70,000	Assembly	In development	1988	Yes	No	No
0.00	11.82	400,000	386,000	Assembly, PASCAL	Completed	1987	No	No	No
14.76	843.86	150,000	[b]	Assembly, PASCAL	Planned	[d]	Yes	No	No
93.40	1,030.00	1,462,540	1,460,040	Assembly	In development	[b]	Yes	No	Yes
0.20	3.31	286,000	250,000	FORTRAN	In development	[b]	No	Yes	No
21.68	[b]	35,000	35,000		In development	[b]	No	Yes	Yes

(continued)

Project name and description	Project type
Intermediate Forward Test Equipment To develop adaptable automatic test equipment.	Test equipment
Single Subscriber Terminal To develop a communications terminal.	Communications
Net Control Station - Joint Tactical Information Distribution System To develop a centralized network management system.	Command, control and communications
Battlefield Electronics Communications-Electronics Operations Instruction System To develop a frequency management system.	Communications
Forward-Looking Infrared Mission Payload Subsystem To develop a system to perform target destination, acquisition, reconnaissance, and artillery adjustment functions.	Surveillance
Elevated Target Acquisition System To develop a surveillance and target acquisition system for maneuver brigades.	Surveillance
Multisensor Target Acquisition System To develop a millimeter wave radar target surveillance and acquisition system.	Surveillance
AN/UPD-7 To develop a surveillance system to provide target intelligence information to Corps commanders.	Surveillance
Cameo Bluejay To develop a survivability system for Army attack and scout helicopters.	Aircraft survivability system
AN/APR-39A (XE-2) To develop a radar warning receiver.	Aircraft survivability system
Advanced Quicklook Electronic Intelligence-Time Differential of Arrival System To develop an electronic intelligence system to detect, identify, and locate non-communications emitters.	Intelligence
Non-Cooperative Target Recognition System To develop a system for the forward area defense to identify hostile aircraft beyond visual range.	Aircraft identification system
JOINT STARS Downsized Ground Station Module To develop a Corps and Division level battlefield surveillance system.	Surveillance
M-60A3 To develop a modified fire control system for the M60 tank.	Fire control
Special Operations Aircraft To develop modifications for two Army aircraft to support special operations.	Avionics

Cost		Lines of code[a]		Other languages used	Project status	Actual or estimated completion date	Types of processing environments		
Fiscal year 1989	Total project	Total	Ada				Real-time	Data base	Distributed
$38.50	$963.00	600,000	200,000	PASCAL, "C"	In development	[b]	No	Yes	No
0.00	70.00	[b]	162,000	Assembly	In development	1993	Yes	Yes	Yes
9.00	[b]	105,000	4,000	Assembly, FORTRAN	In development	[b]	Yes	Yes	Yes
26.90	149.80	27,000	[b]	PASCAL	In development (Ada use is in planning phase)	[b]	No	Yes	Yes
17.54	388.95	16,600	11,060	Assembly	In development	1994	Yes	No	No
4.00	31.00	76,224	27,721	Assembly, PASCAL, PM/M, "C"	In development	1988	Yes	No	No
5.75	567.80	[b]	[b]	[b]	In development	[b]	Yes	No	No
0.00	30.00	[b]	[b]	Assembly, PASCAL, "C"	In development (Ada use is in planning phase)	1993	Yes	No	No
7.68	129.00	33,310	28,600	Assembly, "C"	In development	[b]	Yes	Yes	No
4.13	56.04	20,000	20,000		In development	[d]	Yes	No	No
3.95	7.70	45,000	36,000	Assembly	In development	[d]	Yes	Yes	No
61.80	841.40	[b]	[b]	Assembly, PASCAL	In development (Ada use is in planning phase)	1993	Yes	No	No
0.00	206.50	123,000	95,000	CMS–2M	In development	1998	Yes	No	No
0.00	1.85	3,700	3,100	Assembly	In development	[b]	Yes	No	No
235.80	765.40	150,000	12,500	Assembly, JOVIAL	In development	1992	Yes	No	No

(continued)

Project name and description	Project type
Light Helicopter Experimental Program To develop a new helicopter to meet Army aviation requirements of the future.	Avionics
T-800 Engine Monitor To develop an engine monitoring, fault isolation, and diagnostics system.	System monitor
Command Instrumentation System Trainer To develop a helicopter maintenance training system.	Trainer
Expert Subsystem Status Monitor To develop a helicopter subsystem monitor.	System monitor
Integrated Inertial Navigation System Data Processing Set To study converting existing software to Ada.	Study
Airborne Target Handover System/Avionics Integration To upgrade the AH64-A Apache helicopter control system.	Avionics
Miniature Global Positioning System To develop an airborne global positioning system receiver.	Receiver
Cockpit Emergency Procedures Trainer To develop a helicopter trainer.	Trainer
Combat Service Support Control System To develop a battlefield command and control system.	Command and control
Reserve Component Automation System To develop a resource management information system.	Management information system
Army World Wide Military Command and Control System Information System To develop an information system to provide command and control capability for use in decision making and reporting.	Command and control
US Army Europe Tactical Command and Control System To develop an automated, secret command and control capability.	Command and control
Single Channel Objective Tactical Terminal To develop a mobile communications terminal for the ground mobile and non-strategic nuclear forces.	Communications
Robotized Wire Harness Assembly System To develop a flexible manufacturing system using Ada application software.	Manufacturing technology
Mobile Automated Field Instrumentation System To develop a test performance data collection and analysis system.	Testing system
Tactical Jammer - A To develop a mobile communications jammer.	Communications jammer
Multiple Launch Rocket System Fire Direction System To develop a terminally guided submunition warhead system.	Weapon

Cost		Lines of code[a]		Other languages used	Project status	Actual or estimated completion date	Types of processing environments		
Fiscal year 1989	Total project	Total	Ada				Real-time	Data base	Distributed
$125.00	$34,768.00	3,042,000	(most)	Assembly	Planned	2008	Yes	No	No
56.00	555.00	23,500	20,000	Assembly	In development	2008	Yes	No	No
0.00	2.20	[b]	[b]		Planned	1989	Yes	No	No
0.00	0.40	[b]	[b]	[b]	Planned	[b]	Yes	No	No
0.00	0.40	14,000	13,300	Assembly	In development	1988	Yes	No	No
0.00	210.50	[b]	[b]	[b]	Planned	1992	Yes	Yes	No
1.65	[b]	30,000	30,000		Planned	[b]	Yes	No	No
0.00	1.13	[b]	[b]		In development	1988	Yes	No	No
5.50	[b]	[b]	[b]	[b]	In development (Ada use is in planning phase)	1996	Yes	Yes	Yes
13.12	426.00	[b]	[b]	[b]	Planned	1991	No	No	Yes
13.20	133.40	2,000,000	2,000,000		In development	1993	No	Yes	Yes
7.40	44.50	[e]	[e]		In development	[b]	No	Yes	Yes
93.10	939.70	124,000	96,720	Assembly	In development	[d]	Yes	No	No
0.00	1.80	27,000	3,000	AML, "C", AR-BASIC, PLM–86, FORTRAN	Completed	1985	No	No	No
5.20	112.50	65,650	65,000	Assembly, "C"	In development	[b]	Yes	No	Yes
[b]	166.80	44,500	34,500	Assembly	In development	[d]	Yes	No	No
53.40	422.20	[b]	[b]	[b]	In development (Ada use is in planning phase)	[b]	Yes	No	No

(continued)

Project name and description	Project type
Bradley Fighting Vehicle System To convert vehicle components to digital systems.	Combat vehicle
Howitzer Improvement Program To develop artillery weapon system improvements to provide greater reliability, availability, and effects.	Weapon
Nuclear Biological Chemical Reconnaissance System To develop a data collection, formatting, and reporting system.	Reconnaissance
Chemical and Biological Mini-Detector To develop a system to detect chemical and biological material.	Reconnaissance
Chemical Biological Mass Spectrometer To develop a system to detect chemical and biological threat agents.	Reconnaissance
Sense and Destroy Armor To develop an enhanced counterbattery capability for multiple launch rocket systems. Ada is being used only as a program design language. Other languages will be used for the application program.	Weapon
Improved HELLFIRE To develop an improved guided missile for the Apache helicopter.	Weapon
Radio Frequency Interferometer To develop an electromagnetic radiation or emitter sensing system.	Sensor
Standard Financial System Redesign To develop a new financial system for use at Army installations. Only the general accounting module will be coded in Ada.	Financial system
Ada Language System To develop an integrated set of Ada tools.	Tool
Navy Projects	
Design Evaluation Tool To develop a software program that simulates the AN/UYK-43 (V) computer to investigate the effects of proposed changes to the hardware.	Hardware development tool
Ada Language System/Navy To develop an Ada language capability for the Navy's standard computers.	Tool
AN/BSY 2 Submarine Combat System To develop a submarine combat system for the SSN-21 Seawolf attack submarine.	Submarine combat system
AN/SQQ-89 Improved To develop enhancements to the anti-submarine warfare combat system.	Anti-submarine combat system
P3 Update IV Program To develop an avionics suite for land based anti-submarine warfare aircraft to counter the emerging threat of quieter Soviet submarines.	Avionics

Cost		Lines of code[a]		Other languages used	Project status	Actual or estimated completion date	Types of processing environments		
Fiscal year 1989	Total project	Total	Ada				Real-time	Data base	Distributed
$1.07	$18.68	17,560	15,911	Assembly	In development	[b]	Yes	No	No
50.00	1,655.20	175,000	157,500	Assembly	In development	[b]	Yes	No	No
0.00	38.00	10,000	9,200	Assembly	In development	[b]	Yes	No	No
2.30	10.70	[b]	[b]	[b]	In development (Ada use is in planning phase)	[b]	Yes	No	No
3.20	11.90	12,000	4,800	Assembly	In development	1993	Yes	Yes	No
127.90	670.60	3,000	0	[b]	In development	[b]	Yes	No	No
5.50	26.50	17,050	13,950	Assembly	In development	[b]	Yes	No	No
49.60	128.10	13,000	7,800	Assembly	In development	1993	Yes	No	No
4.29	[b]	1,234,000	1,234,000		In development	[b]	No	Yes	No
0.00	32.60	500,000	500,000		Completed	1986	No	No	No
0.00	0.79	100,000	100,000		Completed	1987	No	No	No
6.40	79.70	750,000	720,000	Assembly, BLISS	In development	1990	Yes	No	No
304.00	1,601.30	2,916,000	1,932,000	Assembly, ECOS, "C", CMS–2	In development	[b]	Yes	Yes	Yes
48.70	952.70	[b]	[b]	ECOS	In development (Ada use is in planning phase)	[d]	Yes	No	No
133.00	2,066.00	770,910	681,250	CMS–2, "C"	In development	[d]	Yes	No	No

(continued)

Project name and description	Project type
Anti-Submarine Warfare Operations Center Command, Control and Communications Upgrade To modernize message and data processing capabilities.	Command, control and communications
Submarine Satellite Information Exchange Subsystem To develop improvements for the message processing system used to communicate with submarines.	Communications
Sea Lance To develop a long-range, submarine launched anti-submarine warfare weapon.	Weapon
Central Atmosphere Monitoring System MKII To develop an atmospheric monitoring system for nuclear submarines.	Monitoring system
High Frequency Anti-Jam To develop a communication system resistent to enemy countermeasures.	Communications
Enhanced Modular Signal Processor To develop general purpose programmable signal processors with a software development environment for a broad range of anti-submarine weapon applications.	Weapon
MK 74 MOD 15 Missile Fire Control System To develop a multi-purpose system providing target tracking, missile communications, terminal homing and surveillance capabilities.	Fire control
Training Device 2F88 S/N2 for the F4-J Aircraft To develop a trainer simulating the F4-J aircraft control, operating, and shutdown procedures.	Trainer
TACAMO Message Processing System To develop an automated message processing system for communications between submarines and other Navy units.	Communications

Cost		Lines of code[a]		Other languages used	Project status	Actual or estimated completion date	Types of processing environments		
Fiscal year 1989	Total project	Total	Ada				Real-time	Data base	Distributed
$9.98	$271.30	740,200	117,600	FORTRAN, PASCAL, VAX−11	In development	1996	No	Yes	No
4.10	23.88	83,509	79,300	MICRO II, "C"	In development	1990	Yes	Yes	No
128.10	2,736.10	33,300	21,300	Assembly, ZILOG, PLM	In development	[d]	Yes	No	No
5.94	54.41	4,000	4,000		In development (Ada use is in planning phase)	1995	Yes	No	No
0.00	110.20	99,005	89,705	Assembly	In development (Ada use is in planning phase)	[d]	Yes	No	No
29.40	4,000.00	1,000,000	0	CMS-2, ECOS	In development	[b]	No	No	No
6.50	27.30	205,000	125,000	Assembly	In development	1992	Yes	No	No
0.00	0.09	135,000	135,000		Completed	1985	Yes	No	No
3.03	6.64	45,000	30,000	Assembly, "C"	In development	1993	Yes	No	No

[a]There are different methods of counting the number of lines of software code; e.g., counting the physical lines or counting from one semi-colon to the next. Also, the number of lines of code for a processing function can vary depending on the computer programming language used. Because of these differences, the number of lines of code written for the projects reviewed are not comparable, but are useful only in providing general indications of the projects' size.

[b]Information was not available at the time of our review.

[c]This is a continuing project.

[d]Information is classified.

[e]This project is using Ada software developed for the Maneuver Control System.

Timing Control Problems With Using Ada for Real-Time Programming

Several timing control problems in using Ada features to build real-time systems have been identified in published literature on this subject and in discussions with software engineering experts. The problems discussed below relate to Ada's tasking feature and run-time executive, discussed in chapter 4.

Timing Control Problems With The ADA Tasking Feature

Ada's tasking feature provides the ability to specify multiple program units, called tasks, which can be executed concurrently. It was also designed so that specialized tasks could be developed to respond to interrupts in a structured manner. When an application receives an interrupt command during processing, it must suspend its current operation, and respond to the special event. Ada's tasking feature currently has a number of timing control problems in the following areas.

Delay Timing Mechanism

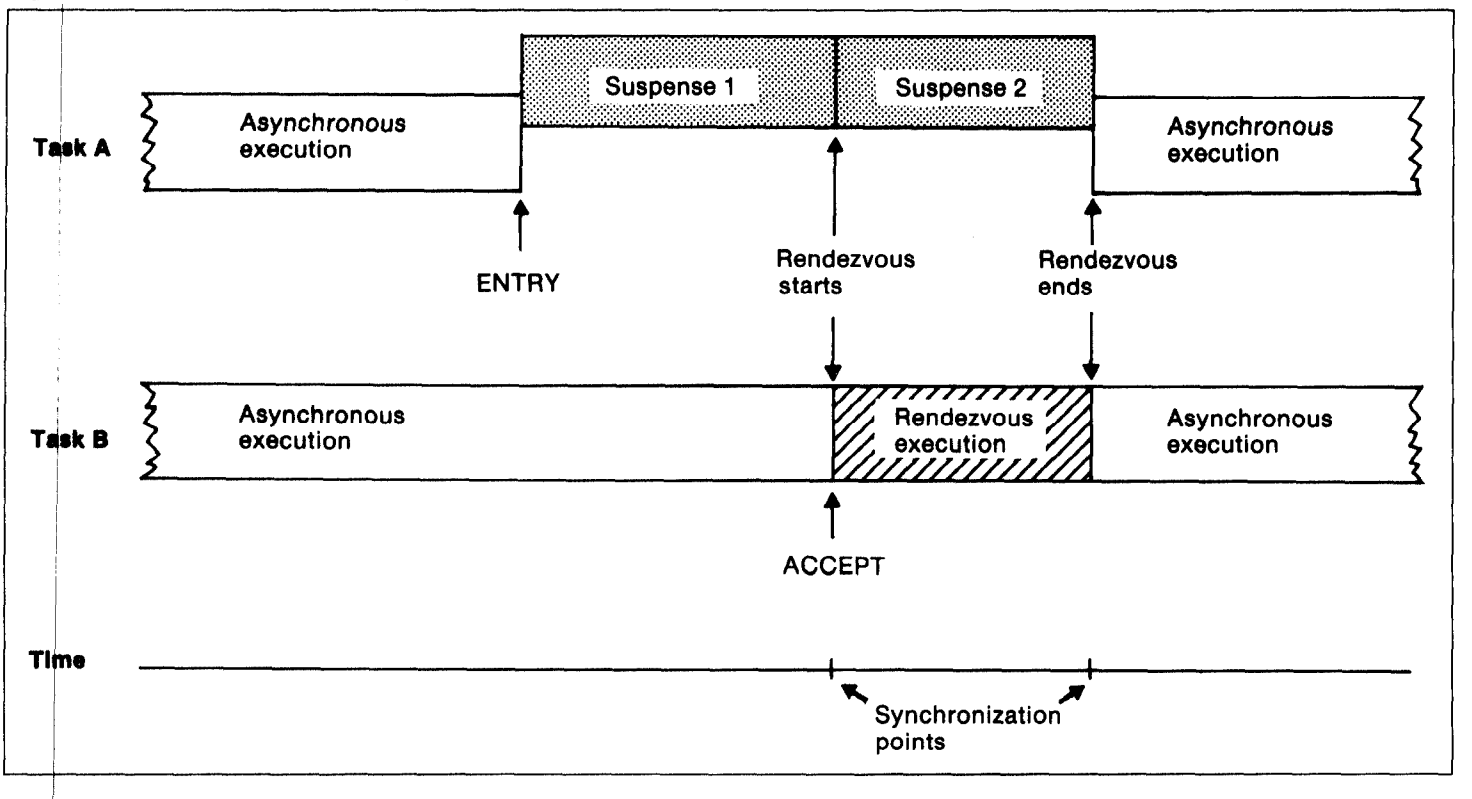
The timing behavior of the delay function within Ada's task scheduling feature is not predictable. The delay statement is Ada's mechanism for specifying when an event can be scheduled to occur. This mechanism only determines the earliest time at which an event could occur. Since only the earliest time an event can occur can be set by the delay statement, the event can occur at any time after the time specified in the delay. According to published literature, if a task has a critical deadline, a language other than Ada may have to be used to schedule its execution; that language is usually assembly.

Rendezvous Timing

Ada's mechanism for communication between two concurrent tasks is called rendezvous. (See fig. V.1.) When tasks are executing concurrently, yet independently, they are said to be executing asynchronously. In the figure, when task A reaches the ENTRY call statement, it requests a rendezvous with task B. If task B is not ready for a rendezvous (as depicted in the figure), task A is suspended. When task B reaches the ACCEPT statement (ready for the rendezvous), rendezvous commences and task B receives whatever information task A has to send. Task B then executes the requested operation (while task A remains suspended). When the operation is completed, task A is reinitiated, information is returned to task A, and both tasks revert to independent processing. This type of communication is referred to as a round-trip communication, since task A sends a message to task B and then waits for a message to come back.

This feature is like a telephone call to an airline reservation system. When a caller (task A) calls the reservation system, the caller is, in

Figure V.1: Ada Rendezvous Concept



effect, requesting a rendezvous with a reservations clerk (task B). If the clerk is busy, the caller is put on hold (Suspense 1 state). When the clerk is free and accepts the call (rendezvous starts), the caller asks for a reservation on a flight. The caller may wait again (Suspense 2 state) until the reservation clerk completes the transaction and informs the caller that the reservation has been made (rendezvous ends). The communication is now complete.

The problem with the rendezvous feature is that the "calling task" (task A) does not have sufficient control over the amount of time it must wait for a rendezvous to be completed. Whereas task A can request that the rendezvous be cancelled if it is not initiated within a specified time period (e.g., task A can limit the time spent in the Suspense 1 state), it cannot place a limit on how long it takes for task B to be completed (the time spent in the Suspense 2 state). Once rendezvous starts, task A will remain suspended until task B completes the rendezvous. If task B does not complete the rendezvous, task A has no way to withdraw from the communication, and will remain suspended.

If an application program must control the time taken from a request for rendezvous to rendezvous completion, the published literature states that a language other than Ada (usually assembly) must be used for the communication.

Task Prioritization

The Ada language provides a means of specifying the relative importance of tasks, and the order in which they are to be initiated. However, task execution does not always follow the desired prioritization.

Rendezvous Priority Inversion

The way the Ada rendezvous function was designed, priority inversion can sometimes occur. For example, consider a program with three tasks: task A has a high priority, task B has a medium priority, and task C has a low priority. If task A requests a rendezvous with task C, and task C is not ready, task A is suspended until task C can make the rendezvous. Task C will not get to the rendezvous point until all other tasks with priority higher than C (for example, task B), are serviced. Therefore, although task A had higher priority than task B, task B will be executed before task A is completed. This phenomenon is referred to as priority inversion. For the example above, the suggested solution, according to the published literature, is to change Ada to allow for priority inheritance; that is, when task A initiates a rendezvous with task C, task C would immediately inherit task A's priority.

Queuing Priority Inversion

Priority inversion can also occur if several tasks, with different priorities, are in the same queue. In Ada, all queues are strictly first-in, first-out, like customers waiting in a customer service line. If a low-priority task is queued before a high-priority task, the low-priority task will be serviced first. The suggested solution to this problem, according to the published literature, is to change the implementation of queues so that tasks in a queue are serviced according to their priority.

Interrupt Handler Priority Inversion

The Ada language provides the programmer with interrupt handling capability directly, through the tasking feature, rather than by using the traditional method of handling it in operating system software. In Ada, an interrupt handler is a task that responds to external devices such as printers, keyboards, alarms, and sensors on airplane control surfaces. Interrupts processed by Ada interrupt handlers may experience priority inversion, that is, higher priority interrupts may be blocked awaiting

completion of a lower priority interrupt. This can occur under the following two circumstances. First, in certain cases, a high-priority interrupt handler can be interrupted and prevented from returning to the "ready" state to respond to the next interrupt. This process reduces both the speed and the predictability with which a handler can respond to a burst of interrupts. Second, while a low-priority interrupt is processing, all other interrupts are blocked until the low-priority interrupt rendezvous has been completed. (See problems with rendezvous discussed above.) Some interrupt handlers must be in the "ready" state when an interrupt occurs; if not, information will be lost. Recommended solutions to these problems, according to the published literature, involve either issuing Ada language clarifications or making changes to the Ada language itself.

Missing Language Feature

Ada's tasking feature could be enhanced by the addition of a mechanism for interrupting a task asynchronously and "immediately" initiating some other operation. An earlier version of the Ada language contained this feature, which it called the "FAILURE exception," but it was deleted from the final version of the language. The "FAILURE exception" might be useful, for example, to allow a program to quickly switch from one activity to another without terminating the program. A recommended solution to this problem, according to the published literature, is to provide this capability with assembly language.

Timing Control Problems With the Ada Run-Time Executive

Ada's run-time executive is an integral part of an application program. It supplies important operating system functions such as memory management and exception handling. The timing control problems associated with the run-time executive primarily concern the need for more precise and predictable timing control. Ada's run-time system does not support the repeated execution of tasks in a sequential manner according to a fixed schedule. In some real-time applications, this precise execution is needed.

The traditional approach has been to design a cyclic executive,[1] usually in assembly language, which initiates all tasks in a specified sequence and executes that sequence repeatedly. However, Ada's run-time system cannot provide periodic task execution. Although it has been suggested

[1] A method of organizing real-time software so actions are executed periodically (cyclically) at a fixed rate predetermined by the programmer. The software is usually hand tailored using assembly language, and must be rewritten each time a change in the fixed rate is needed.

by the Software Engineering Institute in its Ada Adoption Handbook that a cyclic executive can be coded in Ada, it has been reported that Ada does not have some of the timing control functions needed to develop a cyclic executive. Moreover, Ada's timing mechanism, the delay statement, is not sufficiently deterministic for many situations. In a 1986 industry report submitted to the Army, it was also suggested that it may not be possible to predict the behavior of a scheduler (Ada's run-time executive) running on top of another scheduler (a cyclic executive coded in Ada). Because of these problems, Ada's tasking feature may have to be abandoned in some cases and a hand-tailored, assembly language, cyclic executive used. According to the Institute's handbook, if Ada's tasking feature is abandoned, a compilation system should be used that eliminates support for the tasking feature from Ada's run-time executive.

Problems With Using Ada in Real-Time Distributed Processing

Several special problems using Ada to build real-time distributed systems have been identified in published literature on this subject and in our discussions with software engineers at the Software Engineering Institute. The first relates to Ada's rendezvous mechanism (as discussed in ch. 4). The second relates to suggested new language features needed for distributed processing.

Ada's Rendezvous Mechanism

There are different methods of providing communication between two concurrently executing tasks. In Ada, this capability is provided by its rendezvous feature. (For an explanation of Ada's rendezvous feature, see app. V.) According to software engineers at the Software Engineering Institute, there are circumstances when a round-trip communication, as required in a rendezvous, is too slow. An example of an application with this type of requirement is a missile tracking system. Messages regarding the missile's position are being sent very rapidly. It is not critical that all messages be received. What is important is that the most up-to-date position information be received. To accomplish this type of communication, a "fire and forget" protocol is the traditional approach: messages are "fired" at fixed intervals, but the sender does not wait for a reply (it is assumed that the message is received). If one of the messages is not received (lost), it is more important to receive the next, more current, position information than to recover the old data. According to these engineers, much research is presently being conducted on an appropriate Ada implementation of this protocol.

Suggestions for New Ada Language Features

Two new features for the Ada language—explicit distribution and fault tolerance—were suggested in the distributed systems session at the International Workshop on Real-Time Ada Issues in May 1987. However, the workshop did not reach consensus on the merit of these features. It should be noted that other high-order programming languages, such as FORTRAN and "C," do not provide these features.

Explicit Distribution

Distributed systems involve several processors executing multiple tasks. When using Ada, the allocation of tasks to processors is under the control of the run-time executive, and the application programmer cannot influence the allocation. According to the published literature, arbitrary distribution of an Ada program (one under the control of the run-time executive) may cause unacceptable performance overhead. Under certain circumstances, it may significantly simplify the design or improve

the performance of an application if the designer can assign certain tasks to specific processors.

Fault Tolerance

Some distributed systems must continue operation despite the failure of one or more processors. Such systems need to react to processor failures by, for example, moving tasks to other processors. Several different methods to increase programmer control in the face of processor failure have been discussed in the published literature. Two of the methods that have been suggested to improve Ada's ability to support fault tolerance computing are explicit distribution, discussed above, and the asynchronous FAILURE exception (see app. V).

Comments From the Department of Defense

OFFICE OF THE DIRECTOR OF
DEFENSE RESEARCH AND ENGINEERING

WASHINGTON, DC 20301

(R&AT)

4 JAN 1989

Mr. Ralph V. Carlone
Assistant Comptroller General
Information Management and
Technology Division
U.S. General Accounting Office
Washington, D.C. 20548

Dear Mr. Carlone:

This is the Department of Defense (DoD) response to the General Accounting Office (GAO) Draft Report, "COMPUTER LANGUAGE STANDARDIZATION: Status, Costs, and Issues Associated with Defense's Implementation of Ada," dated November 4, 1988 (GAO Code 510230), OSD Case 7832.

The DoD generally concurs with the facts reported by the GAO. In some instances, however, the DoD disagrees with the GAO interpretation of these facts as being unduly negative.

As documented in the report, Ada is not merely a programming language, but a vehicle for new software engineering practices and methods for program specification, structuring, development and maintenance. The few technical difficulties that have been identified with the use of Ada are primarily associated with specific implementations of the language, rather than with the language itself. Ada has been successfully used on numerous programs and the DoD remains firmly committed to its use in the development of Defense systems.

The detailed DoD comments, setting forth certain clarifications, are provided in the enclosure. The opportunity to comment on the draft report is appreciated.

Sincerely,

George P. Millburn
Deputy Director
Defense Research and Engineering
(Research and Advanced Technology)

Enclosure

cc: Distribution

**GAO DRAFT REPORT - DATED NOVEMBER 4, 1988
(GAO CODE 510230) OSD CASE 7832**

"COMPUTER LANGUAGE STANDARDIZATION: STATUS, COSTS, AND ISSUES ASSOCIATED WITH DEFENSE'S IMPLEMENTATION OF ADA"

DEPARTMENT OF DEFENSE COMMENTS

* * * * * *

FINDINGS

FINDING A: Ada Programming Language: Background. The GAO reported that, in 1974, the DoD estimated its future software development costs would be more than $3 billion annually. The GAO stated that, at the time, over 300 programming languages were being used on Defense systems, making it difficult to move application programs among computer systems and expensive to maintain these programs. According to the GAO, the DoD-initiated work led to development of the Ada programming language in 1979. The GAO stated that Ada was established as a military standard in 1980, and approved by the American National Standards Institute in 1983, and by the International Standards Organization in 1987. The GAO further explained that, in 1987, the Department established a policy calling for the use of Ada for all computer applications, except where the use of another language could be demonstrated to be more cost-effective. The GAO reported that the original purpose of Ada was for use in embedded computer applications (those in which the computer is an integral part of a larger system). The GAO stated that Ada now incorporates many features of other languages, adds some new features, and supports the use of new methods for designing, developing, and maintaining software. (p. 2-3, pp. 14-18/GAO Draft Report)

Now on p.2 and pp. 8-10.

DOD RESPONSE: Concur. The GAO cites the acceptance of Ada as a military standard, an American National Standards Institute (ANSI) standard, and an International Standards Organization (ISO) standard. Although not specifically mentioned in the GAO report, Ada was also accepted as a Federal Information Processing Standard (FIPS) in October 1985. It should also be emphasized that (1) the standard accepted by all of these organizations remains the single standard maintained by the DoD; (2) the acceptance by all of these organizations of the same document for that standard has set a precedent for high order programming language standards; and (3) the commercial and international acceptance and use of this single standard offers additional benefits to the DoD in its use of Ada. The applicable DoD Directive 3405.1, "Computer Programming Language Policy," dated April 1987, states, "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." It also states, "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." The GAO statement (page 14) that the DoD has selected the Ada programming language as the single, common computer language for use in both its automated weapons and information systems tends to place the emphasis on more narrow terms and may confuse the focus of the directive.

FINDING B: Three Defense Organizations Are Responsible For Encouraging The Use Of Ada. The GAO stated that three organizations have been established by the DoD to ensure the smooth introduction, implementation, and life-cycle support for the use of Ada or to advance the use of new software engineering methods supported by the Ada programming language. According to the GAO, in 1980 the Department established the Ada Joint Program Office in the Office of the Deputy Under Secretary of Defense (Acquisition Policy) to manage the introduction of Ada. The GAO described the primary responsibilities of this office as ensuring the following:

- that Ada is implemented and maintained as a consistent, unambiguous standard recognized by the DoD and the widest possible community;
- that Ada is used by DoD managers in satisfying their computer programming needs; and
- that life-cycle support is provided for Ada through the development of a complete Ada programming support environment to improve productivity.

The GAO further explained that the DoD established the Software Technology for Adaptable, Reliable Systems Joint Program Office in the Office of the Secretary of Defense to advance software engineering technology. The GAO noted that the goals of this office are to (1) improve quality and reliability in computer application programs, (2) promote the development and reuse of software modules by DoD program managers, and (3) reduce the time and cost of developing software for DoD programs. The GAO observed that, inasmuch as the goals of this office are fundamentally the same as the goals of the Ada language, Ada is the language of choice for all its activities.

The GAO identified the Software Engineering Institute, a Federal research and development center managed by Carnegie-Mellon University, as the third organization having responsibility for Ada. According to the GAO, the institute was established in 1984 to accelerate the transition and use of modern software engineering techniques and methods in DoD programs. The GAO reported that Ada is the primary language used by the institute

Now on pp. 10-11.

in pursuit of these objectives. (p. 3, pp. 19-20/GAO Draft Report).

DoD RESPONSE: Partially concur. Although the GAO report attempts to describe how and why these three organizations were established, the facts presented in the report are not entirely accurate. The GAO report implies that the Ada Joint Program Office (AJPO) is responsible for the development of an Ada programming support environment. While, as a result of recent Congressional action for fiscal year 1989, the management for the development of the Ada Language System/Navy (ALS/N) was transferred from the Navy to the AJPO, the AJPO was not chartered to develop such an environment. The AJPO has supported the development of standard interfaces for Ada programming support environments. The AJPO is currently managed within the Office of the Deputy Director for Defense Research and Engineering (Research and Advanced Technology). The goals of the Software Technology for Adaptable, Reliable Systems (STARS) program are to improve productivity, improve quality and reliability, promote the development and application of reuse, and reduce the time and cost of developing defense software. Ada is the language of choice for STARS, but not because the goals of STARS are the same as Ada. Rather, no other language has as many technical features supporting software engineering techniques or has the degree of standardization across so many computers as that which has been achieved with the Ada language. Therefore, no other language available today or in the foreseeable future is as suitable for developing the technology that STARS is chartered to develop. The Software Engineering Institute (SEI) is a Federally Funded Research and Development Center (FFRDC). Both the STARS program and the SEI are now managed by the Defense Advanced Research Projects Agency (DARPA).

FINDING C: Use of Ada Required By All Military Departments And Agencies. The GAO found that DoD Directive 3405.1, "Computer Programming Language Policy," dated April 1987, established Ada as the single, common computer programming language for intelligence systems, command and control systems for military forces, and automated weapons systems. The GAO reported the directive further provides that Ada shall be used for all other computer applications, except when the use of another approved high-order language can be demonstrated to be more cost-effective over the application's life cycle. In addition, the GAO referred to DoD Directive 3405.2, "Use of Ada in Weapon Systems," dated March 1987, which (1) established a DoD policy that Ada immediately become the single common computer programming language throughout the Department of Defense and (2) prescribed procedures for using Ada in computers integral to weapons systems. (p. 3, pp. 21/GAO Draft Report)

Now on pp. 11-12.

DoD RESPONSE: Concur. However, the correct text found in DoD Directive 3405.1 was provided previously in the DoD Response to FINDING A. The cited directive does not reference the term

"automated weapons systems," as stated in the GAO report. DoD Directive 3405.2 states that "Ada shall be the single, common, high-order programming language, effective immediately," but its scope is limited specifically to all computers that are integral to weapon systems.

FINDING D: Implementation Of Ada By The Military Services. The GAO found that, in 1984, the Departments of the Air Force and the Army established procedures requiring the use of Ada in major programs. The GAO also identified procedures that were established for granting waivers from the requirement to use Ada when justified by life cycle cost and technical practicality. The GAO described Air Force procedures that require requests for waivers from the use of Ada to be approved by the Directorate of Architecture and Technology, Assistant Chief of Staff, Systems for Command, Control, Communications, and Computers. The GAO described the Army approval authority for waivers as the Director of Information Systems for Command, Control, Communications, and Computers. The GAO found that, between 1984 and 1987, Air Force and Army waiver officials received five and ten requests, respectively, for waivers from the requirement to use Ada in major programs. According to the GAO, the Air Force approved all five requests, and the Army approved four of the ten waivers requested. The GAO explained that the Navy uses custom built computers for its aircraft and shipboard systems and that software development tools necessary to use Ada with these computers are not available. Accordingly, the GAO found that the Navy does not require program managers to request waivers from the requirement to use Ada in developing application programs for these computers. The GAO observed that, in 1985, the Navy issued a policy requiring the use of Ada in all computer programs to be used on commercially-available computer systems, unless a waiver is approved by the Commander, Space and Naval Warfare Systems Command. The GAO found that, between 1985 and 1987, the Navy received 43 requests for waivers. According to the GAO, as of July 15, 1988, 23 waivers had been approved, two had been denied, one had been returned for more information, and 17 requests were still pending. (p. 3, pp. 22-23/GAO Draft Report)

Now on pp. 12-13.

DoD RESPONSE: Concur. It should be noted, however, that the Directorate of Architecture and Technology, Assistant Chief of Staff, Systems for Command, Control, Communications, and Computers (C4) (HQ USAF/SC) approves waivers for C4 systems only. The GAO report implies they are the approval authority for all systems. For systems identified in DoD Directive 3405.2, HQ USAF/SC provides a recommendation for waiver approval/disapproval to the Principal Assistant to the Assistant Secretary of the Air Force (Acquisition), SAF/AQ, the waiver approval authority for these systems. The Department of the Navy formally implemented DoD Directive 3405.1 and DoD Directive 3405.2 by SECNAVINST 5234.2, dated November 3, 1988. Ada compilers for the Navy standard AN/UYK-43 and AN/UYK-44 computers have been developed and are currently undergoing the

validation process.

FINDING E: Information On DoD Projects Using Ada Is Incomplete. The GAO found that the Office of the Secretary of Defense (OSD) and the Military Services do not maintain complete lists of projects using the Ada programming language. According to the GAO, the DoD has established "language control agents" to support the use of each DoD-approved high-order language. The GAO identified the Ada Joint Program Office as the DoD control agent for Ada. According to the GAO, this office has established a contractor-operated Ada Information Clearinghouse to gather and disseminate information on Ada tools, conferences, seminars, and training activities. The GAO cited reasons identified by the Ada Joint Program Office for the lack of complete inventory of Ada projects as (1) the voluntary nature of the listing and (2) an associated lack of incentive for program managers to provide project information to the clearinghouse. During its limited review, the GAO identified 74 additional computer projects that were using or planning to use Ada. The GAO, therefore, deduced that it is likely many more DoD projects are using Ada than have been reported. The GAO concluded that accurate record-keeping on projects using Ada would facilitate the identification and sharing of computer programs and lessons learned among Ada projects. (p. 4, pp. 27-29/GAO Draft Report)

Now on p. 2 and pp. 16-17.

DoD RESPONSE: Concur. The GAO states that complete lists of programs using Ada would be useful and would facilitate (1) sharing experiences with Ada projects, (2) identifying different kinds of programs written in Ada, and (3) serving as a catalogue of programs that have developed software modules that might be reusable in other programs. While the existence of a catalogue might facilitate communication among program managers that use Ada for similar applications, it needs to be understood the mere existence of a catalogue of programs that have developed Ada software modules does not mean the software modules will be reusable. Unless modules are designed for reusability and sufficient software documentation for those modules is available, it is unlikely that the modules will be reused. In addition, the costs associated with establishing and maintaining the catalogue might exceed the benefits to be accrued from occasional reuse.

FINDING F: Projects Using Or Planning To Use Ada. Based on its review of 100 DoD projects, the GAO found that Ada is being used for a variety of activities, ranging from studies and demonstration projects to developing application programs. The GAO observed that the majority of the projects reviewed were either being planned or were in development (87 percent). In addition, the GAO noted that the majority of the Ada projects it reviewed (86 percent) were being carried out by the Army and the Air Force. According to the GAO, these projects cover studies and demonstrations to assess the suitability of Ada for specific applications, development of tools necessary to use Ada, and

Now on p. 3 and pp. 17-18.

development of computer application programs. (p. 4, pp. 30-31/GAO Draft Report)

DoD RESPONSE: Concur. As stated, the GAO reviewed only 100 DoD projects, although there exist significantly more projects using Ada. Therefore, the percentages cited in the GAO report should not be construed as an accurate reflection of the corresponding percentages attributed across the spectrum of projects currently using Ada. Appendix IV of the GAO report provides a summary of the DoD Ada projects reviewed. However, by choosing not to include in its review examples of sizable Federal programs planned outside the DoD (such as those by National Aeronautics and Space Administration (NASA) and the Federal Aviation Administration) the report does not accurately portray the overall potential benefits to the Federal Government. The GAO also has not addressed multilateral Ada projects which the DoD has with its foreign allies, or individual Ada projects sponsored by those allies. In addition, not addressed are Ada projects in U.S. and foreign commercial applications, which demonstrate the credibility that Ada has gained in other sectors of the world economy. Since the GAO was tasked to report on Ada within the DoD, it would appear that the resultant impact on the DoD of Ada's commercial and international acceptance should also have been included.

FINDING G: Software Development Tool Projects. The GAO defined software development tools as computer programs used by a programmer to design, develop, and implement application programs. According to the GAO, during 1979 and the early 1980s, the Military Services initiated projects to develop tools needed to write Ada application programs. The GAO reported that, in 1983, the Army released early versions of the Ada tools to industry to stimulate industry interest in developing Ada tools. The GAO reported that the Army planned to use these tools with the new battlefield computer system. The GAO further reported that, in 1984, the Army canceled plans to develop standard battlefield computer systems after expending about $32.6 million. The GAO observed that the Army released the Ada Language System to the public domain to encourage industry support for the maturing Ada language and to maximize the benefits from its investment in Ada.

The GAO found that the Navy initiated the Ada Language System/Navy project to limit the proliferation of Service-unique Ada language support systems and to reduce overall DoD and Navy implementation costs. According to the GAO, the Navy used the Army Ada Language System as the development baseline for its Ada implementation effort. The GAO found that the Navy project is (1) adapting the Ada Language System to support its standard computers and (2) developing additional tools to write application programs for these computers. The GAO estimated that this project will cost the Navy about $79.7 million. The GAO also referenced a Navy Ada Implementation Plan dated March 1987, which identified a reduction in the FY 1987 funding that

caused a 12-month delay in the projected completion of this project--i.e., from September 1989 to September 1990.

The GAO reported that the Air Force project is to develop an Ada compiler and a fully integrated Ada programming support environment. The GAO found that cost growth and schedule delays resulted in a 1985 narrowing of the project scope to include only the compiler development portion, which was completed in 1987, at a cost of about $11.8 million. In addition, the GAO found that Ada is generally used in conjunction with other languages. The GAO concluded that computer programs in 59 of the 100 projects reviewed involved both Ada and at least one other computer programming language. The GAO observed that Ada was intended for use in applications that require fast computer processing; however, it does not, as yet, work well in applications that have severe time and memory constraints and/or require precise timing and control. (p. 5, pp. 31-34/GAO Draft Report)

Now on pp. 18-20.

DoD RESPONSE: Partially concur. Although the GAO has stated that the ALS/N will cost $79.7 million and implies that the ALS/N will not be available until 1990, Navy records show that the ALS/N has cost $46.5 million to date and has delivered compilers and run-time environments for the single/dual CPU AN/UYK-43 and single CPU AN/UYK-44 standard computers. Beginning in December 1988, the Navy expects to mandate the ALS/N for new starts and for major software upgrades. The September 1990 delivery adds multi-processing, multi-programming, and distributed Ada capabilities to the existing products at a projected cost of $18.3 million, for a total of $64.8 million. The balance of the ALS/N funding (approximately $15 million) is budgeted for FY91 through FY94 in support of pre-planned product improvements for the standard computers.

There appears to be an implied negative connotation to the GAO conclusion regarding the use of other computer programming languages with the use of Ada. The GAO did not, however, examine other high order language projects to determine the degree to which other programming languages may have been used in those projects. The degree to which other languages are used with Ada may be significantly less than the degree to which they are used with other high order language projects. Without a factual frame of reference, such information is misleading. In addition, Appendix IV of the GAO report, which provides a summary of the DoD Ada projects reviewed, lists the lines of code associated with a project in a manner which is also misleading. Because an Ada line of code will generally translate into many lines of assembler code, there is no direct correlation or factual basis for comparing the number of Ada lines of code to the number of assembler lines of code (or other language lines of code) as a means for determining the portion of the project that is not written in Ada. The GAO should, therefore, modify Appendix IV accordingly.

The GAO report also states that Ada does not work well in

applications that have severe time and memory constraints and/or require precise timing and control. The DoD acknowledges that there remain issues to be resolved regarding the Ada language and compilers that implement the Ada language. Some of these issues are related to the way in which compilers and their associated run-time environments are implemented. Other issues are currently being addressed as part of the Ada standard revision process. There are, however, projects (such as the Army Hellfire Missile), currently in full scale engineering development, which demonstrate that Ada can work well in a severe time and memory constrained application requiring precise timing and control.

FINDING H: Organizational Costs. The GAO found that, just as the total number of DoD projects using Ada is unknown, so are the costs and benefits of its implementation. The GAO reported that it was unable to determine the costs specifically associated with using Ada versus another language in developing computer application programs for use in operational systems, such as command and control, avionics, and weapons. The GAO observed that the funding for the three DoD established organizations (whose primary focus is on Ada or on implementing new software engineering methods in DoD programs) totaled about $201 million since their inception through FY 1988. In addition, the GAO observed that the Military Services have financed projects designed to study or demonstrate the feasibility of using Ada in specific computer applications or to develop the tools needed to use Ada effectively. The GAO found that 23 of the 100 projects it reviewed focused on objectives specifically related to evaluating the use of Ada in computer applications. The GAO calculated the cost of these projects at about $190 million. (p. 6-7, pp. 35-37/GAO Draft Report)

Now on pp. 3-4 and pp. 21-22.

DoD RESPONSE: Partially concur. The GAO has assumed that the Military Service projects, which total about $190 million, were funded without support from the three DoD established organizations. In fact, the Ada Joint Program Office and the STARS Joint Program Office often provide funding to the Services for such projects. The GAO report should specifically state which portion of the $190 million attributed to projects designed to study or demonstrate the feasibility of using Ada or to develop Ada tools are part of the $201 million funding for the three DoD established organizations. As stated in the report, the costs appear to be cumulative when, in fact, some may be counted twice. In addition, the GAO has assumed that all of the SEI funding is directly attributable to Ada. According to Air Force records, Ada accounts for approximately 10% of the SEI budget.

FINDING I: Ada-Specific Costs For Developing Application Programs Are Unknown: Empirical Evidence Of Cost Savings Is Limited. The GAO found that 77 of the 100 DoD projects it reviewed involved development of computer application programs

for operational systems. The GAO calculated that the total costs of these projects exceeded $74 billion. The GAO reported it could not, however, determine the Ada-specific costs associated with the development of application programs for these projects. The GAO cited the DoD policy on the use of computer programming languages, which only requires the cost of using Ada to be determined when justifying the use of another language. The GAO referenced two studies showing that the distribution of effort in a software development project involving Ada is different from approaches using other computer programming languages. According to the GAO, these studies concluded that, although a greater amount of effort is devoted to the requirements and design phase of Ada software development projects, the increased initial investment is offset by a corresponding decrease in later phases (such as when writing and testing the code). The GAO noted the study showed that designing reusable software and managing its reuse is expected to cost more than developing software for one-time use. The GAO further stated, however, the study concluded that by reducing the amount of software developed for one-time use, total project costs savings from reuse will more than offset the increased costs.

The GAO was also unable to identify any Department projects designed to assess the long-term benefits and cost savings expected from using Ada. The GAO referenced the Ada Board (a Federal Advisory Committee) statement that early actions taken by the DoD to implement the use of Ada were a driving force on industry, leading to the development and maturation of Ada software development tools. The GAO also referenced statements by DoD officials that documentation supporting Ada benefits will become available as DoD programs experience the benefits accrued through the use of Ada. The GAO explained, for example, that the Common Ada Missile Package Project was specifically designed to demonstrate an expected benefit of using Ada. According to the GAO, this project identified about 450 stand-alone software packages, subprograms, or tasks that could be reused. The GAO stated that preliminary results of reusing this software in a simulated missile development project indicates a significant productivity increase in developing the software. (pp. 6-7, pp. 38-41/GAO Draft Report)

Now on pp. 3-4 and pp. 23-24.

DoD RESPONSE: Concur. The GAO report has not, however, acknowledged that the inability to determine Ada-specific costs associated with the development of application programs is not unique to Ada. Such costs have not been tracked for any other high order programming language used. In addition, the GAO report has already indicated that Ada is not merely a programming language, but a vehicle for new software practices and methods for specification, program structuring, development and maintenance. As such, Ada provides the vehicle for a structured software engineering approach to systems engineering, and it is difficult to separate software engineering aspects from systems engineering aspects. The GAO report should recognize how difficult it is to collect such costs, even if

policy were to dictate such action.

The GAO addresses the dissatisfaction of the previous Under Secretary of the Army regarding the lack of convincing evidence of Ada's ability to reduce the mounting software development and maintenance costs within DoD. The fact that such evidence may never appear because the use of software in Defense systems is continuing to grow at such a rapid rate should also be addressed.

FINDING J: Availability of Ada Software Development Tools Is Improving. The GAO observed that many software development tools are used to develop an application program. The GAO reported that, when the Ada language was developed, new tools had to be built to work with this language. According to the GAO, there were few Ada programming tools available in 1984, when the DoD first endorsed the use of Ada for major programs. The GAO found, however, that since that time, a large variety of Ada tools have been developed by industry. The GAO pointed to DoD regulations requiring that an Ada compiler used to develop computer programs for military use be validated by the Ada Joint Program Office to ensure that the compiler's translation of Ada statements is in conformance with the language standard. According to the GAO, the compiler validation process currently consists of over 3,000 tests, which are updated annually. The GAO explained, however, that once a validated compiler has been selected for use on a DoD project, it may be used throughout the life of the project and need be revalidated only if it is modified. The GAO found that the availability of Ada compilers grew slowly from 1983 to 1985, but has risen sharply in more recent years. The GAO observed that the variety of Ada compilers now available permits Ada to be used on many different types of computers. The GAO observed, however, that there is still a lack of validated compilers for some computers. The GAO obtained the comments of five experts in Ada and/or in modern software engineering practices on the availability of Ada compilers. According to the GAO, four of the experts believed that, except for the Navy custom-built computers, the availability of Ada compilers is no longer a major problem for most of the DoD applications.

The GAO further noted that many other tools are used in developing an application program. According to the GAO, the minimum tools needed to develop any application program includes (1) editors, to support a programmer in creating or modifying a computer program and its associated documentation (2) debuggers, to assist in detecting coding errors and (3) configuration managers, to help control changes to the program and its documentation. The GAO also identified other tools that may be required, depending on the particular software development project, such as a target simulator (a tool that simulates the target computer on the computer being used to develop computer application programs) and a downloader (a tool that loads the application program on the target computer). The GAO found that

Now on p. 4 and pp. 25-28.

many tools are now available from commercial companies. In addition, the GAO reported that, again, most of the experts in Ada and/or in modern software engineering practices, it consulted believed that the availability of tools is no longer a major problem. (p. 7-8, pp. 43-48/GAO Draft Report)

DoD RESPONSE: Concur. It should be noted, however, that since the preparation of the GAO report, the update to the Ada Compiler Validation Capability (ACVC) test suite has been changed from an annual basis to every 18 months. Revalidation is no longer required on an annual basis; but rather, according to an extended schedule where the life of a validation certificate remains in effect for one year after the termination of the test suite used in the validation. If a compiler is validated early in the validation cycle, the life of a validation certificate could be as long as two and one half years.

The emphasis by the GAO on the "availability" of tools does not address the issue of robust software engineering environments that are designed to support large software engineering development efforts, distributed over several development sites, and that support software products and tools over a 20-30 year life cycle. In addition to soliciting the opinions of the experts cited in the report, it would have been beneficial if the GAO also had solicited the opinions of DoD program managers regarding such programming support environments.

FINDING K: Better Tests Are Being Developed To Determine Compiler Performance. The GAO found that the compiler validation process assures that Ada compilers translate Ada statements in conformance with the language; however, the test does not measure the compiler's "production quality"-- that is, its ability to meet the performance criteria of a specific application program environment. The GAO noted that the Ada Adoption Handbook, published by the Software Engineering Institute, states that production quality of an Ada compiler is usually measured in terms of (1) compile time efficiency, (2) object code efficiency, (3) compiler services, and (4) support for embedded system requirements. The GAO observed that, for the most part, compilers (including Ada compilers) are not designed to optimize every production quality attribute and tradeoffs are made between performance attributes and project development requirements. The GAO pointed to the comprehensive group of tests that are being developed to enable DoD program managers to compare the capabilities of different Ada compilers. According to the GAO, the first version of these tests is scheduled to be released in 1988, and the second version (incorporating comments from users) is planned for 1989. The GAO concluded that, once successfully completed, the DoD program managers will be able to run the tests comparing the performance of competing compilers and to select the compiler that best satisfies their needs. (p. 7-8, pp. 48-50/GAO Draft Report)

Now on p. 4 and pp. 28-29.

DoD RESPONSE: Concur. The GAO states that the Air Force is developing the test suite, which is known as the Ada Compiler Evaluation Capability (ACEC). Although not specifically stated in the report, the Ada Joint Program Office originated the effort in 1983 and has provided the entire funding for this effort. The first version of the ACEC is now available for use. The GAO has also stated that DoD program managers will be able to run the tests to compare performance of competing compilers. Such a scenario might easily result in a duplication of effort by many DoD program managers. The Ada Joint Program Office has already initiated action for the development of plans and procedures to have the authorized Ada Validation Facilities run the ACEC against those compilers submitted by vendors so that the results of those tests can later be submitted in the response for proposals requested by the various DoD program managers. The DoD program managers will then be responsible for soliciting this information in the request for proposals and for determining the degree to which such information will impact the overall technical evaluation of the resultant proposals.

FINDING L: Promised Benefits of Ada Not Yet Achieved For Some Critical Real-Time Applications. The GAO defined real-time processing as pertaining to the processing of data as it occurs and producing results quickly enough to affect the environment that produced the data. According to the GAO, the high-order languages such as FORTRAN, JOVIAL, and CMS-2 have been used in the past to implement real-time applications, but have been less effective for operations that require very fast or tightly controlled computer processing. The GAO noted that, as a result, these operations were written in assembly language. The GAO found that Ada was designed for use in real-time computer applications and is being used successfully in some applications. The GAO point out, however, that in critical real-time applications were severe time and memory constraints and/or requiring precise timing control, Ada has not worked well. The GAO observed that the Software Engineering Institute software engineers have recommended using Ada whenever possible, but recognize that assembly language may be needed to implement the time-critical portions of applications. The GAO reported that four of the experts in Ada language contacted stated that Ada's utility in critical real-time systems will increase as Ada compilers improve and mature, but some assembly code will always be required in severely constrained real-time applications. (p.8, pp. 51-57/GAO Draft Report)

Now on p. 4 and pp. 29-33.

DoD RESPONSE: Partially concur. The GAO statement that Ada has not worked well implies that the fault lies with the Ada language itself, rather than with the actual implementations of the language. Navy experience has shown that some forms of "priority inversion" are caused by the actual design/implementation, not by problems with the Ada language, and that time-critical processing can be achieved by utilizing the existing time interrupts from actual hardware via the Ada interrupt entry mechanism, as opposed to a cyclic executive as

discussed in Appendix V of the GAO report. In addition, the GAO has not included discussion of the current DoD efforts that are addressing real-time issues, such as the Ada Joint Program Office sponsored project at Fort Monmouth, or the activities of the Joint Integrated Avionics Working Group (JIAWG). With regard to the inevitability of some assembly code being included in real-time applications, as espoused by the experts, it would have been useful if the experts had offered criteria on what percentage of assembly code would be considered acceptable in Ada applications programs.

FINDING M: Ada Features Hold Promise For Use In Real-Time Distributed Systems. The GAO described real-time distributed computer systems as several interconnected computers that process data simultaneously to jointly accomplish a mission. The GAO stated that the computers may be dissimilar, and may be either widely dispersed geographically or housed in one facility. The GAO noted that one of the Ada experts it consulted believed the characteristics of real-time distributed systems are not yet fully understood, and building such systems is difficult regardless of the language used. The GAO observed that there are no features in the Ada language, or any other high-order language, specifically designed for the development of real-time distributed systems. The GAO concluded, however, that once problems with the tasking feature and run-time executive are resolved, Ada could be useful in developing real-time distributed systems. The GAO reported that one expert it consulted stated that the Ada tasking feature is appropriate for some real-time distributed applications, principally those in which computers share memory with each other. According to the GAO, an aspect of the Ada tasking feature called "rendevous" could, if improved, handle some of the communications among distributed computers. The GAO pointed out that using Ada to build distributed systems is currently the subject of independent research and development. The GAO identified these efforts as including how to (1) structure a distributed system using Ada, (2) communicate among processors, and (3) ensure that processing continues in the face of partial hardware failure. (p. 8, pp. 57-59/GAO Draft Report)

Now on p. 4 and pp. 33-34.

DoD RESPONSE: Concur. The GAO has not, however, addressed the issue of hardware for real-time distributed systems. It needs to be recognized that often the limitations of existing hardware, rather than the language used for the implementation, can be the critical determinant in the development of military systems.

FINDING N: Full Ada Benefits Cannot be Achieved Without a Uniform Approach To Using Ada With Data Base Management Systems. The GAO defined a data base as an organized collection of data that can be used by a variety of applications. The GAO described the data base management system controlling a data base as a computer program that organizes, catalogs, stores, and

retrieves information in the data base. The program that interacts with the data base management system to gain access and retrieve information is identified as an application program by the GAO. According to the GAO, many data base application programs are being written in Ada. The GAO explained that application programs written in Ada can interface with data base management systems written in other languages. The GAO noted, however, that no standard method has yet been established for the required interface. The GAO observed that several approaches have been devised, but a consensus is lacking on a standard approach, both in the data processing community at large and among the experts the GAO consulted. The GAO concluded that, consequently, problems exist in achieving some of the benefits of Ada in application interfacing with data base management systems.

The GAO explained that a data base management system controls the storage and retrieval of information in a computer system much as a librarian controls the documents in a library. The GAO described the English-like query language used with data base management systems, which provide users with a simple, yet powerful, means to access and manipulate data. While some data base management systems offer vendor-unique query languages, the GAO observed that the American National Standards Institute has endorsed one language, the Structured Query Language (SQL), which can be used for communicating with many data base management systems. The GAO noted the benefits of using a standard query language is analogous to that for using a standard program language such as Ada. The GAO found that the DoD has, therefore, focused on formulating approaches for interfacing Ada and SQL so that a user in one program may conveniently code algorithmic-type functions in Ada and data base management functions in SQL. The GAO pointed out that four primary methods for an Ada and SQL interface have been proposed, but all of the proposed methods include currently unresolved technical issues. The GAO concluded that, without a solution to these technical conflicts, the benefits of both Ada and SQL may not be fully realized. (p. 8, pp. 59-67/GAO Draft Report)

Now on p. 5 and pp. 34-38.

DoD RESPONSE: Partially concur. The GAO references the work of the Software Engineering Institute (SEI) regarding the development of a standard Ada/SQL interface that would be acceptable to both the data base and Ada communities. It should be recognized that the SEI embarked on this effort at the request of the Ada Joint Program Office (AJPO) and with AJPO funding. The AJPO has supported and funded the activities addressed within the report in order to overcome the specific technical difficulties that had been identified. The most recent SEI report (October 1988), provided as a result of these activities, does not identify any unresolved technical issues.

The issue of interfacing Ada with data base management systems is broader than determining an Ada/SQL interface standard. Use of a query language is just one method of data base access, and the relational model is just one of several models used for data

base access. The GAO has placed too much emphasis on the Ada/SQL interface issues. To conclude the section on the Ada/SQL discussion with a question to the experts on the broader issue of whether there is a problem in interfacing Ada with data base management systems is misleading. Additionally, the GAO has not indicated whether data base expertise was one of the criteria for the selection of the experts consulted for the preparation of this report.

RECOMMENDATIONS

Now on p. 5 and p. 40.

RECOMMENDATION 1: The GAO recommended the Secretary of Defense direct the Ada Joint Program Office to take steps to develop performance data that demonstrate whether Defense use of Ada is achieving its goals. (p. 8, pp. 70/GAO Draft Report)

DoD RESPONSE: Concur. Steps have already been initiated by both the Ada Joint Program Office and the STARS Joint Program Office to collect data resulting from DoD efforts where such data gathering procedures have been implemented. However, it must be recognized that the benefits of Ada are to be realized throughout the entire life-cycle of DoD systems, including the post deployment support of systems designed with modern software engineering techniques and implemented in Ada. With a typical life-cycle of 20-30 years for large DoD systems, the expectation for near term development of performance data correlated to such systems is unrealistic.

Now on p. 5 and p. 40.

RECOMMENDATION 2: The GAO recommended that the Secretary of Defense direct the Ada Joint Program Office to take steps (1) to develop a DoD-wide repository of computer applications and modules written in Ada; and (2) make them available for reuse. (p. 9, pp. 70/GAO Draft Report)

DoD RESPONSE: Partially concur. There currently exists an Ada Software Repository (ASR), established in 1984, that is a collection of general information, Ada programs, tools and educational material. The ASR, which is available on the Defense Data Network, currently contains over 1500 files. The ASR receives sponsorship and funding from the U.S. Army Information Systems Command and the STARS Joint Program Office. All of the information in the ASR is considered to be in the public domain and is accessible to users via several mechanisms.

Experience with the ASR, however, as well as with other software repositories, including the Computer Software Management Information Center (COSMIC), which is operated by the University of Georgia for NASA, and with the Common Ada Missile Packages referenced in this GAO report, indicates that a repository of Ada modules is not sufficient. There are numerous issues that

must also be addressed in order to establish such a repository. Such issues include: (1) ensuring adequate repository management, including configuration management, quality control, catalog generation, public relations, dissemination of components and documentation, and coordination with sponsors and developers; (2) defining quality control standards, including standards for coding, documentation and testing; (3) assisting users in finding the components or programs they need by providing a well-indexed, comprehensive catalog; (4) making the repository software as easy to obtain as possible; (5) publicizing the repository as widely as possible to software developers, potential users and government software acquisition organizations; and (6) contacting other repository developers to obtain specific information on software licensing agreements used, and agreements with government sponsors and contractors regarding the reuse of software.

In addition to the issues directly associated with the development of a Defense-wide information repository, there are contractual issues associated with: (1) the development of incentives for reusing Ada modules; (2) the specification of appropriate government contract language to accommodate the reuse of existing Ada software; and (3) the assessment of liability for the reuse of Ada software.

Therefore, repository technology must be developed prior to the development of the repository recommended by the GAO report. The STARS program includes the largest DoD funded initiatives to develop repository technology. Two contractors are currently funded under the STARS prime contracts to maintain operational repositories for the STARS program. These are operational today and will be continually improved. Repository technology is an essential adjunct to effective reusability and is currently being undertaken by the STARS program in fulfillment of its charter.

RECOMMENDATION 3: The GAO recommended that the Secretary of Defense direct the Ada Joint Program Office to take steps to gather and disseminate complete lists of all DoD projects using Ada. (p. 9, pp. 70/GAO Draft Report)

Now on p. 5 and p. 40.

DoD RESPONSE: Concur. A list of programs does little, however, to assist DoD program managers unless the list includes information such as the type of program, the application area, the point of contact, the size of the program, and the current status of the program. Although maintaining and constantly updating such a list is desirable, the costs associated with such a dynamic effort will be high and could exceed the utility of the effort. As has been noted by the GAO, the Ada Joint Program Office is currently gathering and disseminating this information. For reasons which were explained in the GAO report, the current list is not all encompassing. No program name is included on the list unless the information has first

been verified; otherwise, the credibility of the entire list would be jeopardized. The information included on this list is largely dependent upon such information being provided by the respective DoD Components. With the issuance of DoD Directive 3405.2 and the resultant identification of Ada Executive Officials for the DoD Components, such information has become more readily available.

RECOMMENDATION 4: The GAO recommended that the Secretary of Defense establish a committee of independent experts on Ada and software engineering technology to monitor and periodically report to the Secretary on DoD actions to implement Ada.

DoD RESPONSE: Partially concur. The DoD agrees that it would be beneficial to monitor the DoD actions to implement Ada. However, there already currently exists within the DoD a group of individuals who are responsible for the development of Ada implementation plans and for the monitoring of Ada programs. DoD Directive 3405.2 requires each DoD Component to designate an Ada Executive Official, who shall monitor programs relative to the use of Ada, support the Ada program activities in the DoD Component, and serve as a focal point in the DoD Component for all Ada program activities.

RECOMMENDATION 5: The GAO recommended that the Secretary of Defense direct the committee to assess existing projects and propose additional projects to demonstrate the intended cost saving associated with using Ada. (p. 9, pp. 70-71/GAO Draft Report)

Now on p.5 and p. 40.

DoD RESPONSE: Partially concur. The DoD agrees that it would be beneficial to determine cost savings associated with the use of Ada. It is the DoD position, however, that there currently exists a sufficient number of DoD programs using Ada that can be assessed regarding their individual cost savings associated with the use of Ada. To propose new projects for this purpose would be an inappropriate use of Defense funds. In addition, the assessment of these programs can best be accomplished by an internal DoD study or by an internal DoD committee, such as the Ada Executive Officials, who are more familiar with the issues associated with these programs.

RECOMMENDATION 6: The GAO recommended that the Secretary of Defense direct the committee to assess existing research efforts and identify where there is a need for further research to overcome the technical problems in using Ada in real-time, distributed, and data base applications. (p. 9, pp. 70-71/GAO Draft Report)

Now on p. 5 and p. 40.

DoD RESPONSE: Partially concur. The DoD agrees that the technical problems in using Ada in real-time, distributed and data base applications should be addressed and overcome. However, there is no need for an independent committee, as the three organizations cited in the GAO report have the responsibility for addressing the technical problems associated with the use of Ada. The Ada Technology Insertion Program (ATIP), sponsored by the Ada Joint Program Office, was established for the purpose of identifying and funding the work necessary to overcome specific technical barriers associated with the use of Ada in military systems. The successful results of previous ATIP efforts have been documented and published via the Ada Information Clearinghouse.

RECOMMENDATION 7: The GAO recommended that the Secretary of Defense direct the committee to assess the progress and results of the Ada Joint Program Office in developing a repository of software written in Ada. (p. 9, pp. 70-71/GAO Draft Report)

Now on p. 5 and p. 40.

DoD RESPONSE: Partially concur. The DoD agrees that it would be beneficial to assess the progress and results of the repository. However, this recommendation is based upon the premise that the Ada Joint Program Office will be directed to develop such a repository. As has been previously discussed in the DoD Response to Recommendation 2, the STARS program is currently responsible for repository technology. In addition, a DoD internal committee, such as the Ada Executive Officials, would be a more appropriate vehicle for assessing the progress and results of this repository technology.

RECOMMENDATION 8: The GAO recommended that the Secretary of Defense direct the committee to recommend appropriate courses of action in employing Ada. (p. 9, pp. 70-71/GAO Draft Report)

Now on p. 5 and p. 40.

DoD RESPONSE: Partially concur. The DoD agrees that it is beneficial to receive recommendations from an independent group of Ada experts regarding appropriate courses of action in employing Ada. However, there is no need to establish a new committee. The Federal advisory committee, the Ada Board, is the organization that is currently chartered to perform such activities. The Ada Board has already assisted the DoD through its reports: (1) "Ada Board Response To The Report Of The Defense Science Board Task Force on Military Software,", dated February 1988; and (2) "Ada Board's Recommended Ada 9X Strategy," dated September 1988.

Comments From the Software Engineering Institute

Software Engineering Institute

November 15, 1988

Mr. Ralph V. Carlone
Director
General Accounting Office
Washington, DC 20548

Dear Mr. Carlone:

Thank you for the opportunity to review the draft report on Computer Language Standardization: Status, Costs, and Issues Associated with Defense's Implementation of Ada. Unfortunately, the report arrived while I was away on a trip and several of my engineers were likewise traveling. Consequently, I have only been able to scan the report and will provide my own comments. I have not yet been able to coordinate with the other engineers in time to meet your November 15 deadline. I will, however, ask them to call Mr. Rhile with additional comments.

Although I had seen an earlier draft of those sections dealing with specific technical issues, this is the first view of the full report. I believe we have been able to assist Mr. Heyl in understanding the technical issues that he chose to address. I believe these are now essentially correct, although my engineers may call with specific comments.

Overall, the report seems somewhat prejudiced by the tone of the question that was being addressed. It seems biased toward the notion that there are significant problems with the language and with the program. There are technical issues associated with the adoption of Ada as there are with the adoption of almost any technology. The consensus of most who have investigated the question (including two previous GAO studies, the National Academy of Sciences, the Defense Science Board, and the Air Force Scientific Advisory Board), has been very complimentary of the Ada Program and the importance of adopting Ada for the DoD. In focusing on the difficulties experienced during this period of maturation, I am concerned that the report would be interpreted as indicative of a major problem, which I do not believe to be the case. Having said that, let me now address some specifics of the report.

Now on p. 3.

- on page 4 - the report says "Defense has not yet demonstrated whether the use of Ada can help control software development and maintenance costs." While this is true, it might be useful to point out that the length of the acquisition cycle makes such an early analysis impossible at this time. It is not the case that the DoD is not interested in providing such demonstration, or is not involved in such attempts, but that the time it takes to implement a software system for a major weapons system certainly exceeds five years. The benefits are expected to be derived over the entire software life cycle which is the life cycle of the system. That may be as much as 20 years or more. Consequently, all we can hope for is indications.

Now on p. 30.

- on page 34 - at the end of the paragraph there is a phrase that says "it does not as yet work well in applications". This phrase "it does not as yet work well" appears several times in the report. I think this is poor phrasing in that the statement indicates that Ada, the language, does not work well. The report needs to be more specific about what doesn't work anywhere. In this instance, and in most instances where the phrase is used in the report, a more correct statement would be that

Carnegie Mellon University
Pittsburgh, Pennsylvania 15213-3890
(412) 268-7700

current implementations are not sufficiently efficient for specific applications. In summary, it simply is not a correct statement to say "it does not yet work", because indeed functionally it works. It's simply that the code generators are not optimized to support specific timing needs.

Now on p. 21.

- on page 35 - the analysis of costs that includes STARS and SEI in the cost of Ada is a misrepresentation. Let me specifically address the question of the SEI. Note that the entire SEI budget was included as a cost of Ada with the comment that the SEI has a responsibility for Ada. The SEI has no specific responsibility for the Ada language other than tasking by the AJPO. The SEI has five programs.
  - The Process Program is aimed at improving the software engineering process which includes management, project planning, and the use of technology. Ada is not a factor in that program. It is neither used nor addressed because most of the issues are language independent.
  - The Education Program has the goal of producing a Master of Software Engineering degree which includes courses in software verification and validation, formal methods, project management, and the like. None of these courses are specific to the Ada language. When they work on courseware that requires the use of a language, Ada is generally used as the mechanism, but that is a small part of their effort.
  - The Methods Program is concerned with the methodology and supporting environments for software engineers. None of this work is language specific.
  - The Software Systems Program is involved in developing analytic techniques that support building real-time distributed systems. In the case of this program, while software is actually developed and many of the projects use the language, they are not specifically motivated toward Ada. Let me give an example: the real-time scheduling project aims at transitioning a scheduling protocol to assist people building real-time schedulers. The protocol is priority based and we are experimenting with the priority scheme in Ada to illustrate that this scheduling protocol is effective in real-time applications. The work is motivated by other technical considerations and Ada is simply the language of choice for that project.
  - The fifth program, Technology Transition, involves significant interface with DoD organizations in assisting them to adopt various technologies. While Ada is often a subject addressed, it is a very small part of the activity.

  We do have a collection of projects specifically motivated to assist the Ada and STARS Programs with technical support, and most of those projects are funded by the Ada and STARS Program. Even though they are motivated by Ada, to include the SEI budget is to engage in double accounting.
- Perhaps the only two projects being funded by the SEI that could honestly be presented as Ada projects would be: the Ada Adoption Handbook, written in 1986-87, and the Distributed Ada Real-time Kernel, which is a project to build a runtime kernel that would be callable by Ada programs. To add the total SEI budget to the cost of Ada is simply a misrepresentation and there is potentially some double accounting involved as well. Furthermore, it is incorrect to say that the SEI has specific responsibility for Ada, other than the specific tasking they receive from the Ada Joint Program Office. The SEI uses Ada because it is the best technical choice and often promotes its use as part of the SEI technology transition responsibility.

- While I do not have quite the same detailed perspective of STARS, much of the STARS Program is more generic and I believe Ada is a mechanism or a technology base on which the STARS Program expects to build. I would question the validity of the statement that the STARS Program has "responsibility" for the introduction of Ada. I do believe both the SEI and STARS will positively effect the introduction of the language.

Now on p. 25.

- on page 42 - after the first paragraph I would suggest that you admit to the investment that industry has made in compilers and tools. One of the real advantages of the Ada Program is that the DoD, although representing a diminishing percentage of software costs in the United States, is still a significant customer for the software industry and has used that to provide leadership in the adoption of an important technology direction. While the DoD has made an investment in support for the language, a more significant investment has been made by commercial industry. This point does not come through. A good place to point that out might be on page 45. All of the compilers, save one, have been developed with venture capital and private funds, not with government funding. The one exception being the Navy effort, which you point out.

Now on p. 27.

Now on p. 29.

- on page 50 - in the first full paragraph where you discuss the evaluation of compiler performance and say "according to software engineers at the Software Engineering Institute these tests vary in size and quality and can be used to provide general evaluations of compiler performance. However, none of these tests go into great depth." I think it would be more correct to say "the collection of tests is not necessarily complete and, therefore, do not necessarily represent an adequate test of performance". Some of the tests do go into depth. It is just that they may not expose weaknesses for a specific application.

Now on p. 30.

- on page 51 - The phrase "Ada has not worked well" is used again and I believe it is not the best way to express the problem.

Now on p. 31.

- on the bottom of page 53 and the beginning of page 54 - a similar phrase which says "the use of an Ada run-time executive has not worked well" is incorrect. There are run-time executives that are working quite well today and tasking facilities that are working quite well. It may very well be that a specific application's use of those may not have been appropriate to the need, but the more global statement is just not correct.

Now on p. 32.

- on page 55 - there is a comment that several approaches are being developed and evaluated by software engineers at the Institute, in private industry, and professional organizations, which address Ada's deficiencies in critical real-time systems. Again, I believe this is a prejudicial statement. The use of the term "Ada's deficiencies" would indicate that Ada does not provide some feature that is needed. I don't believe this is the circumstance. I believe that there is a need for various optimization techniques for approaches to designing with Ada and implementing with Ada that would enable its effective use in time critical systems. I am unaware of any specific deficiency in the Ada language. Rather, it is the lack of appropriate optimizations in specific implementations. This is a complex issue that can not be simply swept away with the phrase "Ada deficiency".

Now on p. 32.

- on page 56 - item #1 in the second paragraph, "Ada's runtime executive should be

modified so that runtime system support is included for only those features actually needed in a specific application". I think it would be useful to point out here that some vendors do this. This is a recommendation that was made several years ago and has been implemented by at least two vendors that I know of. While not all vendors yet provide that facility it should be pointed out that the trend is in that direction.

Now on p. 39.

- on page 68 - the first paragraph indicates that the approach that focuses on reusability is supported by the Software Engineering Institute and Defense Science Board. While it is a true statement, I believe it would be useful to add a phrase indicating that the institute is pursuing the technical and economic feasibility, as are many other organizations. It is premature to "endorse" widespread reuse at this point. In other words, we support the notion, think it is a powerful one, and are investigating its use. This does not mean that we have demonstrated its value.

The recommendations being made seem appropriate and consistent with the report. However, it's useful to recognize that the Ada Program has a significant number of responsibilities and limited resources. Some trade-off needs to be done to assess whether the recommendations are important enough to supplant other efforts, or whether the AJPO resources need to be increased. The AJPO is pursuing the introduction of the language into specific programs, is responsible for managing the evolution of the standard, and managing collateral standards such as interfaces to SQL and the like. They are also supporting evaluation tests that address some of the GAO earlier points. While the items recommended may indeed be appropriate, it would be a shame for Congress to direct the AJPO to undertake those things at the cost of other more important activities. It might be useful to acknowledge that those other activities are underway. At least make it clear that the recommendations are made in that context.

Again, thank you for the opportunity to provide these comments. My engineers will be studying the report more carefully and will call Mr. Rhile with additional comments, if appropriate.

Sincerely,

Larry E. Druffel
Director

LED:jlm

Attachment - 1

Major Contributors to This Report

Information Management and Technology Division, Washington, D.C.	Howard G. Rhile, Jr., Associate Director, (202) 275-9675 Norman F. Heyl, Senior Technical Evaluator Karlin I. Richardson, Technical Adviser Michael P. Fruitman, Supervisory Reports Analyst
Boston Regional Office	Frederick R. Cross, Jr., Regional Management Representative Gerald L. Laudermilk, Evaluator-in-Charge Martin F. Lobo, Supervisor
New York Regional Office	Gerda M. Lloyd, Supervisor

Requests for copies of GAO reports should be sent to:

U.S. General Accounting Office
Post Office Box 6015
Gaithersburg, Maryland 20877

Telephone 202-275-6241

The first five copies of each report are free. Additional copies are $2.00 each.

There is a 25% discount on orders for 100 or more copies mailed to a single address.

Orders must be prepaid by cash or by check or money order made out to the Superintendent of Documents.

United States
General Accounting Office
Washington, D.C. 20548

Official Business
Penalty for Private Use $300

First-Class Mail
Postage & Fees Paid
GAO
Permit No. G100