IDA MEMORANDUM REPORT M-387

COMPILING AND PORTING THE NOSC TOOLS FOR USE BY THE DEFENSE LOGISTICS AGENCY

David Carney

May 1988

Prepared for
Defense Logistics Agency

INSTITUTE FOR DEFENSE ANALYSES
1801 N. Beauregard Street, Alexandria, Virginia 22311

DEFINITIONS

IDA publishes the following documents to report the results of its work.

Reports

Reports are the most authoritative and most carefully considered products IDA publishes. They normally embody results of major projects which (a) have a direct bearing on decisions affecting major programs, or (b) address issues of significant concern to the Executive Branch, the Congress and/or the public, or (c) address issues that have significant economic implications. IDA Reports are reviewed by outside panels of experts to ensure their high quality and relevance to the problems studied, and they are released by the President of IDA.

Papers

Papers normally address relatively restricted technical or policy issues. They communicate the results of special analyses, interim reports or phases of a task, ad hoc or quick reaction work. Papers are reviewed to ensure that they meet standards similar to those expected of refereed papers in professional journals.

Memorandum Reports

IDA Memorandum Reports are used for the convenience of the sponsors or the analysts to record substantive work done in quick reaction studies and major interactive technical support activities; to make available preliminary and tentative results of analyses or of working group and panel activities; to forward information that is essentially unanalyzed and unevaluated; or to make a record of conferences, meetings, or briefings, or of data developed in the course of an investigation. Review of Memorandum Reports is suited to their content and intended use.

The results of IDA work are also conveyed by briefings and informal memoranda to sponsors and others designated by the sponsors, when appropriate.

The work reported in this document was conducted under contract MDA 903 84 C 0031 for the Department of Defense. The publication of this IDA document does not indicate endorsement by the Department of Defense, nor should the contents be construed as reflecting the official position of that agency.

This Memorandum Report is published in order to make available the material it contains for the use and convenience of interested parties. The material has not necessarily been completely evaluated and analyzed, nor subjected to IDA review.

Approved for public release; distribution unlimited.

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Public release/unlimited distribution.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) IDA Memorandum Report M-387	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION: Institute for Defense Analyses 6b OFFICE SYMBOL: IDA	7a NAME OF MONITORING ORGANIZATION OUSDA, DIMO
6c ADDRESS (City, State, and Zip Code) 1801 N. Beauregard St. Alexandria, VA 22311	7b ADDRESS (City, State, and Zip Code) 1801 N. Beauregard St. Alexandria, VA 22311
8a NAME OF FUNDING/SPONSORING ORGANIZATION: Defense Logistics Agency 8b OFFICE SYMBOL (If applicable): DLA	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER MDA 903 84 C 0031
8c ADDRESS (City, State, and Zip Code) DLA-ZWS, 3A636 Cameron Station, Alexandria, VA 22304-6100	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. T-T5-423 / WORK UNIT ACCESSION NO.
11 TITLE (Include Security Classification) Compiling and Porting the NOSC Tools for Use by the Defense Logistics Agency (U)	
12 PERSONAL AUTHOR(S) David Carney	
13a TYPE OF REPORT: Final 13b TIME COVERED: FROM _____ TO _____	14 DATE OF REPORT (Year, Month, Day): 1988 May 15 PAGE COUNT: 32
16 SUPPLEMENTARY NOTATION	
17 COSATI CODES: FIELD / GROUP / SUB-GROUP	18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number) Ada programming language; software tools and techniques; portability; compilation; WWMCCS; WIS; configuration; NOSC tools.

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

The purpose of this IDA Memorandum Report is to document the issues encountered during an effort to supply a set of software tools to the Defense Logistics Agency (DLA). This effort entailed selecting the tools, compiling them at the Institute for Defense Analyses (IDA), and porting them to computers at DLA. All the tools were written in the Ada programming language and were placed in the public domain by the World Wide Command and Control Systems (WWMCCS) Information System, Joint Program Management Office (WIS JPMO). The effort to provide these software tools was part of a larger effort on the part of IDA to assist the DLA in using and evaluating the Ada language. The value of this study is to illustrate typical design questions that arise in the course of a large-scale Ada project and to indicate appropriate solutions, especially as regards the issue of portable software.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL Audrey A. Hook	22b TELEPHONE (Include area code): (703) 824-5501 22c OFFICE SYMBOL: IDA/CSED

IDA MEMORANDUM REPORT M-387

COMPILING AND PORTING THE NOSC TOOLS FOR USE BY THE DEFENSE LOGISTICS AGENCY

David Carney

May 1988

INSTITUTE FOR DEFENSE ANALYSES

Contract MDA 903 84 C 0031
Task T-T5-423

CONTENTS

LIST OF TABLES

PREFACE

The purpose of IDA Memorandum Report M-387, *Compiling and Porting the NOSC Tools for Use by the Defense Logistics Agency*, is to provide the DLA Prototype Project members with an analysis of the technical issues that can arise in large-scale software projects written in Ada. This report documents an effort to compile and port a large body of public domain software. Its conclusions concern the issue of software portability. This work partially fulfills requirements of IDA Task T-T5-423, Defense Logistics Information System. Reviewers of this document included: Bill Brykczynski, Audrey Hook, Joseph Linn, Katydean Price, Robert Winner, and James Wolfe.

1. INTRODUCTION

1.1 PURPOSE

The purpose of this study is to document the issues encountered during an effort to supply a set of software tools to the Defense Logistics Agency (DLA). This effort entailed selecting the tools, compiling them at the Institute for Defense Analyses (IDA), and porting them to computers at DLA. The effort documented in this study lasted five months, from January through May, 1987.

All of these tools were written in the Ada programming language and were placed in the public domain by the WWMCCS (World Wide Military Command and Control System) Information System, Joint Program Management Office (WIS JPMO).

The effort to provide these software tools was part of a larger effort on the part of the Computer and Software Engineering Division (CSED), IDA, to assist the DLA in using and evaluating the Ada language. The value of this report for DLA is to illustrate typical design questions that arise in the course of a large-scale Ada project, and to indicate appropriate solutions, especially as regards the issue of portable Ada software.

1.2 SCOPE

This paper is primarily concerned with configuration and compilation issues. The tools described in this paper were not subjected to a testing suite, nor is any rigorous assessment of the tools' quality made, except as regards the effort needed to compile and port them.

1.3 BACKGROUND

The Defense Logistics Agency is engaged in a long-term program to modernize its Automated Information Systems. This program is the Logistics Systems Modernization

Program (LSMP). One immediate task of the LSMP is to determine the feasibility of using the Ada language in DLA applications.

To that end, IDA and DLA are engaged in an Ada Prototype Project. Initially, a group of DLA programmers were trained in the use of Ada. They are currently engaged in writing large-scale demonstrations of Ada's capabilities. The nature of these demonstrations will be similar to the COBOL software that is currently in use by DLA.

The DLA Ada project will take place in a three-tier network environment, comprising an IBM 3090, one or more Gould PowerNode 9000s, and several Zenith Z248-PCs. The Zenith 248 is an IBM PC/AT compatible computer. The Ada compiler for both the IBM and the Gould was developed by TeleSoft, Inc. † and the compiler for the Zenith was developed by Alsys, Inc. The three-tier environment is modelled on the one proposed for the entire DLA LSMP.

A principal requirement for the Prototype Project is an Ada Programming Support Environment (APSE). The APSE is a "workbench" of Ada tools that can later be expanded and modified as the modernization project itself is expanded and modified. The tools described in this paper were chosen as prospective members of this workbench.

A large body of Ada software, which includes many of the needed APSE tools, had previously been developed by various contractors for the Naval Oceans System Center (NOSC). These contractors included Ford Aerospace, GTE, and Intermetrics, Inc. The principal purpose of the NOSC effort was to encourage the use of Ada by a large number of

† TeleSoft is a registered trademark of TeleSoft.

programmers throughout the industry. Since they were products of the first large-scale attempt to establish a body of reusable Ada software, the NOSC tools do not necessarily reflect the current state of Ada practice. All of the NOSC tools are available from the public domain repository on the SIMTEL20 node of the Defense Data Network.

2. TECHNICAL APPROACH

There were four subtasks involved in the attempt to create an APSE:

- Locating and selecting tools
- Compiling the tools at IDA
- Porting the tools to computers at DLA
- Evaluating the tools at DLA

Each of these subtasks is described in this section.

2.1 LOCATING AND SELECTING TOOLS

DLA initially proposed a list of requirements for the intended APSE. The eventual makeup of the APSE was intended to reflect these requirements as closely as possible.

In IDA Memorandum Report M-294, *Ada Prototype Project*, it was recommended that any tools for the APSE be acquired by examining public-domain software before resorting to commercially available software. Since the NOSC tools were in the public domain, they were ideal candidates for potential inclusion in the DLA APSE.

A natural division of the NOSC tools was size. Some were developed on and were clearly usable by PC-sized machines, while others were large, mainframe-sized tools. This partition would potentially be continued in the different tiers of the DLA architecture. Table 1 lists the mainframe tools and Table 2 lists the PC tools selected.

2.2 COMPILING THE TOOLS AT IDA

Before trying to develop the APSE at DLA, each individual tool was compiled using IDA computers. Particularly since the larger tools were developed with a VAX,† this permitted us to

distinguish between compiling a tool from source and porting that tool to another machine. All of the mainframe tools were compiled using the DEC Ada compiler on IDA's VAX 8600. As can be seen, the majority of them were written by Intermetrics, Inc. A potential benefit from this choice was that, since these tools were intended to have a common interface and used a common set of base packages, a high degree of integration could be factored into the APSE from the beginning. Another potential benefit was that tools could be informally assessed at IDA before making the effort to port them to DLA. The remaining tools were compiled on IDA's IBM PC/AT, with the hope that they could subsequently be ported to any of the three tiers at DLA. The compiler used at IDA was the Alsys compiler, version 1.3.

2.3 PORTING THE TOOLS TO DLA

There was some question as to where the mainframe tools would reside within the DLA tiers. Both the IBM and the Gould compilers are based on the TeleSoft compiler, and either tier should be a reasonable environment. Because the Gould was immediately available, it was chosen as the test locale. The tools that were compiled on the IBM PC/AT at IDA were ported to the Zenith tier at DLA. Once these had compiled on the Zenith, they could subsequently be ported, if need be, to either the Gould or the IBM.

† VAX is a registered trademark of Digital Equipment Corporation.

TABLE 1. Mainframe Tools

MAINFRAME TOOLS	
Name	*Developer*
Video	AdaSoft
Compilation Order	Intermetrics
Data Dictionary	Intermetrics
Document Manager	Intermetrics
Formatter	Intermetrics
Global Cross Referencer	Intermetrics
McCabe-Halstead Complexity Analyzer	Intermetrics
Requirements Tracer	Intermetrics
Standards Checker	Intermetrics
Statement Profiler	Intermetrics
Test & Analysis Tool (consisting of five subtools:) Automatic Path Analyzer Path Analyzer Performance Analyzer Source Instrumenter Self-Metric Analysis Tool	Intermetrics

TABLE 2. PC-Based Tools

PC-BASED TOOLS	
Name	*Developer*
Combine and Break	WIS JPMO
Dynamic Strings	WIS JPMO
Generic Message Handling Facility	Veda
List Processor	A & E Inc.
Manpower	GTE
MathLib	USAF
Menu Manager	Ford AeroSpace
Tracker	GTE
Transmission Control Protocol	E-Systems
UnitRep Protoptype	SAIC

2.4 TESTING THE TOOLS AT DLA

Any rigorous assessment of a tool's quality depended on the success of porting the tool. Any tool that could not be brought to execution at DLA was of no use, however excellent it might seem at IDA. Full evaluation was therefore delayed until a tool had been ported. Nontheless, merely bringing a tool to the point of execution at IDA implied an informal observation of a tool's behavior. If a tool was seen to be of limited or nil value at IDA, there seemed little value in porting it purely for the exercise.

3. FINDINGS

The following findings focus on the tasks of compiling the tools at IDA and porting them to DLA. They are first summarized, followed by a detailed description of each stage of the effort.

1. The large tools that were investigated all contain instances of machine-dependent code. These dependencies are of varying degree, ranging from trivial to severe.
2. The source code found in the SIMTEL20 repository represents different and contradictory versions of several of the NOSC tools.
3. The existing documentation is misleading concerning the program library structure needed to create a set of the Intermetrics tools.
4. The Gould compiler has a severe limitation in its limited capacity to compile large aggregates.
5. The Gould compiler's behavior with failing compilations is distinctly unfriendly.
6. Some source code that executed properly when compiled by the DEC compiler failed when compiled by the Gould compiler.
7. A major deficiency of many of the NOSC tools, both large and small, is the presence of code written on non-validated compilers.

3.1 MAINFRAME TOOLS

The majority of the mainframe tools were written by Intermetrics, and findings about them compose the major portion of this section. In terms of time spent, the work on these also constituted the major expenditure of time and effort.

3.1.1 Intermetrics Tools

There were substantial problems in compiling the Intermetrics tools on the VAX, and none of them could be brought to successful execution on the Gould. There were three major obstacles encountered:

- Obtaining a coherent set of sources
- Removing system-dependent code from sources
- Limitations of the Gould compiler

The first two obstacles were overcome, but the limitations of the Gould compiler prevented any tool from executing successfully.

3.1.1.1 Obtaining a Coherent Set of Sources

Each of these tools relies on a common set of low-level packages called the "Abstractions". All of the Abstractions were collected into a large (50,000) line file called "abs.src", a copy of which was included in the source directory for each tool. The abstractions include various utilities to handle list and string processing, interfaces, etc. Conceptually the notion is a sound one, but in practice there were significant problems.

Principally, the Abstractions underwent changes as the tools were being released to NOSC. Unfortunately, the sources available to IDA reflect several views of the Abstractions' changes. There were differing versions of some packages, including at least one where a procedure had been replaced by a package of the same name; both sources were present. Further, though there was an obvious intention for a common interface among all of the tools, there were at least three conflicting versions of interface packages. Finally, some of the later versions of the tools had dependencies on packages that were not found in any of the

"abs.src" files.

There were two alternatives possible. First, each tool could have its own abstractions library. This would simplify, and in some cases solve, the problem of variant sources, since it appeared that the version of an individual tool would be consistent within its own set of sources. The problem with this was the enormous size of the abstractions library; to reduplicate it for a dozen tools was simply untenable. In addition, one of the elements that made the tools attractive for the DLA APSE was their commonality, a feature that this strategy negated.

The second alternative, which was followed, was to assemble an acceptable abstractions library by weeding out the variants. This entailed a good deal of comparing sources to determine compatibilities. This alternative also depended on locating all of the missing abstractions packages. Fortunately, there was another version of many of these tools available at IDA. This provided all of the needed sources, and a coherent Abstractions library could be assembled.

One final, and highly time-consuming difficulty was caused by a deficiency in the documentation of each tool, where the instructions read as follows (paraphrased):

> Compile all of the abstractions [found in the file "abs.src"] into a program library...
>
> Compile everything [the sources for the particular tool] into the program library containing the abstractions or a sublibrary whose parent library contains the abstractions...

These instructions omit the pertinent fact that, if one intends to compile more than one tool, then the latter option:"... or a sublibrary whose parent library ..." is mandatory. In other words, each separate tool has source code that overwrites one or more of the abstractions packages.

Each must use a separate sublibrary (derived from the parent Abstractions library) so that it can autonomously overwrite what packages it needs to. This fact was not obvious, and only became apparent after long debugging of executables.

Once the Abstractions library was stabilized, all of the tools compiled and linked successfully. Although no rigorous testing was done, minimal observation indicated that the tools were highly useful. In one case (Compile Order) the tool was put to immediate use by other users of the IDA VAX.

3.1.1.2 Removing System-dependent Code

To port these packages to DLA's Gould, the system-dependent code had to be removed. Many of the tools had a large amount of such code, and all had at least some. It was present in three packages that depend on DEC Ada: STARLET, HOST_LIB, and VMS_LIB. STARLET is supplied by DEC as a system interface, and HOST_LIB is an Intermetrics-written package that depends on STARLET. The source for VMS_LIB could not be found. Because of the subprogram calls that were made to this package, by inference it is either a different version of STARLET or is dependent upon it.

Some of the DEC-dependent code was trivial, merely involving a system call to read a single command line argument. Other code performed more particular actions, such as stripping the "[." and "]" from a directory name, or making repeated system calls to expand wildcard names. This was especially true of package File_Manager, which is needed by the Data Dictionary, Compile Order, and Requirement Trace tools.

In the case of three other tools, Formatter, Standards Checker, and Statement Profiler, the dependency was of the trivial sort. In each of these cases, there was only one line of code that

involved package VMS_LIB, and in each case it was a system call to read the command-line argument. This was removed, thus removing the dependency on package VMS_LIB. The simplest expedient was to replace it by two calls to Text IO, one to prompt for input and the second to read the input using Get_Line. This meant that each tool now had a different user interface. In the original VAX version, the typical use of a tool was:

```
standards_check (source => abc.ada, output=> xyz.txt);
```

In the new version, the tool would first be invoked with no argument:

```
standards_check
```

At this point, the executable prompts for input, and the user now supplies:

```
source => abc.ada, output=>xyz.txt
```

This method succeeded with the Formatter, Standards Checker, and Statement Profiler. Each of these had the single line of code altered, was recompiled, and executed on the VAX. There was no attempt to establish this as a final solution, but only to find an expedient for removing VAX dependencies. The eventual form of command-line interface would be determined later by the DLA team.

For those other tools that depended on package File_Manager, this was not a workable strategy. Especially in the case of the Compile Order tool, where wildcard expansion of filenames is needed, the required changes needed to be more fundamental. No attempt was made to make those changes until the three "simple" tools described above had been successfully ported to the Gould.

3.1.1.3 Overcoming Limitations of the Gould Compiler

The Gould compiler had severe difficulties in compiling these tools. Only one tool has compiled entirely, and none have successfully executed. The difficulties lay in the following areas:

- Problems in compiling large source files
- Compiler behavior with failing compilations
- Inability to compile large aggregates

3.1.1.3.1 Problems in Compiling Large Source Files

The Intermetrics tools are all large pieces of software, some as large as 40,000 lines of code. The original form of each tool's source collects all of the packages in a single large file, which could (on the VAX) either be compiled as a whole or unpaged into separate files, one file per compilation unit.

It was impossible to compile any tool from a single file on the Gould. The reason was that if one of the units failed in compiling, the entire compilation failed. Since every file contained at least one failing unit (see below), every file was doomed to fail as a whole. The solution was to unpage the large file into separate files, one per compilation unit.

After splitting the large file into its component units, generic specifications and bodies had to be reunited into a single file since the Gould compiler demands that these be compiled as a single compilation. This is in accordance with the Ada Language Reference Manual (ANSI/MIL-STD-1815A, paragraph 10.3.9), but is was one further factor that added to the time spent preparing the sources for compilation.

3.1.1.3.2 Compiler Behavior with Failing Compilations

When the original large source file was submitted to the compiler, failing compilations were "invisible": a substantial period of time would pass during which the compiler was seen to be at work. But when the compilation would eventually end, the library did not reflect the compiled units. Nor were there error messages, and the user had no idea of what had occurred. It was only when the original large file was split into its component parts that specific errors could be associated with particular packages.

3.1.1.3.3 Compiling Large Aggregates

The cause of failure in each case was due to a significant limitation of the Gould compiler, which is its inability to compile large aggregates. The failing unit was the body of package ParseTables, a separate version of which is needed for each tool. This package, averaging about 3,500 lines in length, consists of aggregate assignments to arrays of integers. The arrays are very long, varying between 500 and 8,000 elements. In the case of every tool, this body failed to compile. The failure was "invisible" - there was no error message, but the compilation would fail. The failure was only indicated by the fact the unit was not in the library.

The eventual strategy was to further split the package up into subunits. But the act of breaking this package (ParseTables) proved to be both difficult and tedious, since it is automatically generated code. One tool, the Standards Checker, was selected as a test case. The original package body, 3,500 lines of code, was split into nine subunits, one for each aggregate. Eight of these compiled successfully, but the ninth still failed. This was further subdivided into nested subunits, with the single aggregate assignment replaced by slice assignments. When this was done the tool finally compiled.

3.1.1.3.4 Failure in Execution

When the executable for the Standards Checker was run, a Constraint Error was raised during the elaboration of package ParseTables. It was deemed fruitless to make a comparable effort with any other tool, and the attempt to use these tools as the foundation of the APSE was abandoned.

Note that the same code executed properly on the VAX. The only changes made to the code were the two calls to Text_IO.Put_Line and Text_IO.Get_Line (replacing the VAX/STARLET system call), and the partitioning into subunits. While it is conceivable that the partitioning may have been flawed, it is unlikely that this would cause an error of this type at runtime; it is much more probable that incorrect partitioning would cause compile-time errors. And without any other indication of where to begin debugging, the only recourse would be to redo the partitioning. Such a course of action, at that point, was not a reasonable strategy.

3.1.2 Video

This program, written by AdaSoft, is a screen-oriented menu manager. It was developed using a non-validated compiler from TeleSoft. Compiling it was hampered by the large number of illegalities in the code, principally in illegal use of OUT parameters. In general, the code quality was not high. In addition, there was a dependency on a low-level interface package whose source was not available. It was possible to rewrite the code enough to bring the tool to compilation.

Once the illegal code was removed and the dependencies adjusted, all of the packages compiled and linked successfully. Minimal observation indicated that the tool was of little potential value to DLA. Video was abandoned as a candidate for the APSE at that point.

3.2 PC TOOLS

The remaining items in the candidate toolset were those whose size permitted them to be

compiled on and ported to PCs. Several of these tools, like Video discussed above, had been developed using non-validated compilers, and could not be brought to successful compilation. These tools included:

- Generic Message Handling Facility
- List Processor
- Tracker
- Transmission Control Protocol
- UnitRep Prototype

All of these tools failed in compilation due to illegalities in the code, and all of them had been developed using non-validated compilers. The remaining tools were all successfully compiled on IDA's IBM PC/AT, and have all been compiled on the Zenith PC at DLA. These tools include:

- Combine and Break
- Dynamic Strings
- Manpower
- MathLib
- Menu Manager

4. RECOMMENDATIONS

1. An attempt should be made to compile and execute one of the large NOSC tools on the DLA IBM mainframe computer. If this step is successful then the plan to use the NOSC tools can continue.

2. The APSE for the DLA Ada Prototype Project should investigate commercial sources for candidate tools. This should be done regardless of the success or failure of recommendation 1.

3. Certain elements of an APSE should be originally available with a compiler. In particular, the Gould corporation should be notified that the Ada Debugger and Source Compilation Lister, released by TeleSoft with the compiler, must be made available to DLA.

Distribution List for IDA Memorandum Report M-387

NAME AND ADDRESS	NUMBER OF COPIES
Sponsor	
Ms. Sally Barnes DLA-ZWS, 3A636 Cameron Station Alexandria, VA 22304-6100	2 copies
Other	
Defense Technical Information Center Cameron Station Alexandria, VA 22314	2 copies
Mr. Ken Bowles TeleSoft 5959 Cornerstone Court West San Diego, CA 92121-9891	1 copy
Mr. Tim Brass Defense Logistics Agency 3990 East Broad Street Columbus, OH 43216-5002	1 copy
Ms. Tricia Oberndorf Naval Ocean Systems Center Code 425 San Diego, CA 92152-5000	1 copy
CSED Review Panel	
Dr. Dan Alpert, Director Center for Advanced Study University of Illinois 912 W. Illinois Street Urbana, Illinois 61801	1 copy
Dr. Barry W. Boehm TRW Defense Systems Group MS 2-2304 One Space Park Redondo Beach, CA 90278	1 copy
Dr. Ruth Davis The Pymatuning Group, Inc. 2000 N. 15th Street, Suite 707 Arlington, VA 22201	1 copy

NAME AND ADDRESS	NUMBER OF COPIES
Dr. Larry E. Druffel Software Engineering Institute Carnegie-Mellon University Pittsburgh, PA 15231	1 copy
Dr. C.E. Hutchinson, Dean Thayer School of Engineering Dartmouth College Hanover, NH 03755	1 copy
Mr. A.J. Jordano Manager, Systems & Software Engineering Headquarters Federal Systems Division 6600 Rockledge Dr. Bethesda, MD 20817	1 copy
Mr. Robert K. Lehto Mainstay 302 Mill St. Occoquan, VA 22125	1 copy
Mr. Oliver Selfridge 45 Percy Road Lexington, MA 02173	1 copy
IDA	
General W.Y. Smith, HQ	1 copy
Mr. Philip Major, HQ	1 copy
Dr. Jack Kramer, CSED	1 copy
Dr. Robert I. Winner, CSED	1 copy
Dr. John Salasin, CSED	1 copy
Ms. Anne Douville, CSED	1 copy
Mr. Terry Mayfield, CSED	1 copy
Dr. David Carney, CSED	2 copies
Ms. Audrey A. Hook, CSED	1 copy
Mr. Richard Waychoff, CSED	1 copy
Ms. Katydean Price, CSED	2 copies
IDA Control & Distribution Vault	3 copies