FINAL REPORT

Ada Technology Support for NASA-GSFC

Contract No. NAS5-29173

May 9, 1986

AdaSoft, Inc.
9300 Annapolis Road
Lanham, Maryland 20706

TABLE OF CONTENTS

1.0 INTRODUCTION

In May of 1985, AdaSoft, Inc. was awarded a competitive contract to provide support to the Mission and Data Operations Directorate in the utilization of the Ada programming language and environments to perform directorate functions. In particular, our support was to include the review of Ada related project development plans, technical recommendations on the use of Ada, expert assistance in response to queries about the use of Ada, conducting of short seminars and problem solving sessions and evaluation of Ada software development at NASA/GSFC.

The period of performance on this contract was one year with a level of effort of approximately one-half man-year. The contract technical officer was Mr. Robert Nelson of Code 522 at NASA/GSFC.

2.0 SUMMARY OF WORK PERFORMED

During this contract, AdaSoft provided support to several different projects and individuals in a number of different Ada-related areas. This section contains a summary of support provided by AdaSoft on this contract over the past year.

2.1 MNET NCP Conversion

The Mission and Data Operations Directorate Network (MNET) conversion effort was chosen as the first task for evaluation and assistance by AdaSoft. The MNET project required the re-writing of the existing Network Control Program (NCP) in the Ada Programming Language. The DEC Ada compiler running on the VAX under VMS was used for the initial development effort. However, the major reason for selecting this project was the desire to produce Ada code that was transportable to other Ada compilers running on other computers under other operating systems. A primary goal of the AdaSoft participation in this project was to insure that Ada design and programming practices that facilitate transportability were being used.

After preliminary evaluation, AdaSoft identified several areas of investigation with regard to the NCP project. Some of these areas were also applicable to Ada usage in general. They included a review of the current NCP generic design, a review of NCP code and the investigation of problem areas encountered during software development.

During August, we continued assisting the NCP project. Joel Cohen of CSC called to discuss potential problems with reading network blocks into an Ada program such that the contents of the blocks could be easily referenced. Briefly stated, the network blocks have several different formats, so the most natural way of storing them is in a variant record. However, the type of block is defined by one or more bits within the block. Thus, a technique had to be devised to determine the block type, initialize

the discriminant to the proper value and store the block into the variant record. We discussed various approaches for placing the network block into a variant record with the appropriate discriminant value using as little system-dependent code as possible. We developed, compiled and tested several programs to illustrate various approaches to handling network blocks as Ada variant records. These test programs were provided to Joel Cohen of CSC.

In October, AdaSoft attended the Critical Design Review (CDR) for the Generic MNET Ada/NCP. It was our opinion that the level of detail presented was insufficient for a CDR and was more appropriate for a Preliminary Design Review (PDR). We believe the reasons for the lack of detail in the CDR were twofold. First, members of the NCP project staff were incapacitated during a portion of the design phase. Second, and more significant, is the fact that none of the personnel assigned to this project were experienced in the use of the Ada language. Therefore, much of the design time was undoubtedly used learning Ada. This is acceptable in a pilot project where the goal is to introduce the Ada language or ascertain its effectiveness in an application area and where no production software is to be generated under a specified schedule and at a specified cost. The risks of undertaking an Ada project without at least a core of analysts and programmers that are experienced in the use of Ada include cost overruns and/or the delivery of poorly designed and implemented software. This can lead to the Ada language gaining an undeserved reputation for being inefficient and unproductive. Therefore, it was our recommendation that GSFC require one or more key personnel who have significant experience in the design and development of software systems in the Ada language be assigned to all projects in which the Ada language will be used to produce production software. The number of Ada literate personnel would depend upon the size of the project.

In addition, we had some reservations about the design of the NCP as prsented at the CDR. These concerns involved the use of a single task, referred to as Transmission_Control_Process (TCP) to essentially control the flow of information through the system. It appeared that the use of a single task here could cause unnecessary delays in processing messages between different users and the network. For example, while the TCP is processing an input packet for user 1, an outgoing message from user 2 would be delayed waiting to be serviced by the TCP. These appeared to be independent operations so the processing of one should not delay the processing of the other. Unfortunately, the reality of the situation is that current Ada run-time implementations execute all tasks within a single process so that the processing performed by one task will affect the processing of other tasks. Our feeling is that while this is the reality of current implementations, a multi-tasking Ada system should be designed as if each task were assigned to a separate processor and could be executed in parallel except at rendezvous points. Our recommendation was to analyze the TCP to determine where it could be separated into several tasks that might provide a smoother flow of information through the system.

AdaSoft met with the NCP development team to discuss our reservations concerning their design. The intent of this meeting was to clarify our position and gain a better understanding of the requirements that led to the current design. Following the meeting, we generated an action item for the review board so that the NCP team could make a formal response to our comments.

This project was cancelled in April of 1986. It is our understanding that the decision had nothing to do with the use of the Ada language but that an off-the-shelf package was found that could provide the required functionality.

2.2 Ada Compiler Stress Testing

AdaSoft began work in June 1985 performing "stress" tests on the newly delivered version of the DEC Ada compiler. The purpose of these tests was to determine the performance limits of the DEC Ada compiler. AdaSoft obtained copies of a set of Whetstone benchmark programs developed at GTE and used to compare the performance of various Ada compilers. Comparable versions of these benchmark programs existed in Ada, Fortran and Pascal. Using these programs, the performance limits of the DEC Ada compiler could be determined and could be compared to those of the DEC Fortran and Pascal compilers. By running these programs, we hoped to obtain a comparison of the efficiency of the code generated by each of the compilers.

The benchmark programs written in Ada, Fortran and Pascal performed simple arithmetic operations contained within loops. We inserted output statements in each program and compared the results to insure that the programs were all producing the same results. After it was determined that the results were consistent, preliminary runs of the benchmark programs were made in a non-standalone environment. The results of the runs indicated that the Ada programs executed 30 to 40 percent faster than the Fortran or Pascal programs. Since these results were unexpected, assembly language code generated by each of the compilers was obtained for further study. Preliminary study of the assembly language code indicated that the Ada optimizer generated considerably different machine language code from that of the Fortran or Pascal optimizer and that, under certain conditions, the code generated by the optimizer in the Ada compiler was more efficient than that generated by other compilers.

It was determined that results from running the programs being used in the comparative analysis of Ada, Fortran and Pascal, had been published under a contract sponsored by WIS. Our results verified those that were published under the WIS contract.

In August 1985, the GRO Dynamic Simulator project requested that AdaSoft perform a different type of "stress" test on the DEC Ada compiler. These tests involved peforming various numbers of concurrent Ada compilations to determine the operational impact

on the system and its users. We created and tested the DCL command procedures required for performing the Ada compiler stress tests as outlined by the GRO project. Standalone computer time was obtained in early September to begin performing these tests.

In September, AdaSoft completed stress tests of the DEC Ada compiler. The results were presented to the GRO project management team led by Mr. Frank McGarry. The group included representatives from NASA management, the University of Maryland and the GRO project working group. After reviewing the results, the group agreed that it would be both interesting and informative to see the same type of stress tests applied to a comparable set of Fortran programs so that a comparison between the two development environments could be made. Mr. McGarry agreed to provide the Fortran programs and AdaSoft was directed to perform the necessary computer runs and prepare a summary of results similar to that presented for Ada.

AdaSoft made several sets of runs in an attempt to evaluate the performance of the DEC Ada compiler and subsequently to compare its performance to that of the Dec Fortran IV-PLUS compiler. Some anomalies were observed while analyzing the Ada compilation results. Initial runs were made on the DEC VAX-11/780 with a Floating Point Accelerator (FPA). After these runs were made, the 780 was upgraded to a 785, but with no FPA. The same set of runs was made on this new configuration in early October. As expected, these runs yielded better CPU times (approximately 10 percent less than the 780) but the elapsed time to complete the runs ranged from approximately the same as that obtained on the 780 for a single Ada compilation to 100 percent more for five concurrent Ada compilations. The results of these runs were discussed with the systems programmers and some modifications were made to certain VMS parameters. Then another set of the same runs was made in late October that yielded CPU times that were approximately 30 percent less than those of the original 780 runs but elapsed times that were still as much as 100 percent more than the original 780 runs. These results were reported to the systems programmers.

Another anomaly that occurred during these tests was the inability to duplicate elapsed times for the same runs, even when these runs were made back-to-back. These tests were conducted in a standalone environment so it was expected that the results of the same runs performed in the same evening would be almost exactly the same. It was observed, however, that the results of a run consisting of several concurrent Ada compilations differed by as much as 10 percent from those obtained making the same run a few minutes later. Neither AdaSoft nor the systems programmers had an explanation for this but it might have been caused by disk I/O contention since all user data sets accessed during a run were on the same disk.

In December, we re-ran the Ada compiler stress tests. The reason that these tests were re-run is that the VAX-11/785 had a Floating Point Accelerator (FPA) installed.

In all, three sets of Ada compiler stress test runs were made on the DSTL VAX-11/78x. The first set was made on September 4, 1985. The system configuration at that time consisted of a VAX-11/780 with FPA and no VAX cluster. The second set of runs was made on October 10, 1985. The 780 had been upgraded to a 785 but no FPA was installed. Also, the 785 was running as part of a VAX cluster with the 750 and 8600. The results of these runs were not reported previously because it was difficult to obtain a consistent set of times. Because of the length of time these runs took, only a maximum of five concurrent compilations were run.

When comparing the three sets of runs, two observations can be made. First, running the DEC Ada compiler on a VAX-11/78x without a Floating Point Accelerator is not recommended. Second, it appears that the VAX cluster has some impact on the compilation wall clock times when several (four or more) compilations are being run concurrently.

At the suggestion of the GRO management team, we ran "stress" tests on the DEC Fortran compiler similar to those run previously on the DEC Ada compiler. The Fortran programs used in the tests were selected by Frank McGarry of GSFC. They consisted of three Fortran programs, one of which was a main program and the other two were subprograms. Collectively, they contained approximately 225 non-comment statements. The environment in which they were run consisted of a VAX-11/785 with Floating Point Accelerator running as part of a VAX cluster. No other users were on the system when the tests were run. Comparing the results of these tests with earlier Ada "stress" tests shows that the impact of multiple concurrent Fortran compilations on the system is approximately one-third that of the Ada compilations.

2.3 FDAS Project Support

In July, AdaSoft presented a tutorial to FDAS project members on access types and dynamic memory management in Ada. We also began attending some of the FDAS weekly project meetings. We studied the FDAS requirements and preliminary design documents. After studying these documents, we presented certain suggestions and comments to the FDAS project members concerning the design of the system.

We continued to attend FDAS weekly meetings for about two months and provided comments on the ongoing systems design. The final set of comments was provided in the form of a written report delivered to Bob Nelson and the FDAS project management.

2.4 NASA/GSFC Ada Programming Standards

In September, AdaSoft assisted in presenting the concept of a NASA Ada Programming Standard to the GSFC Ada User's Group. With the assistance of Mr. Dan Roy of Century Computing and Mr. Ed Seidowitz of GSFC, a presentation was made to the user's group that addressed various aspects of Ada style. A questionnaire was developed and distributed to the user's group meeting in an attempt to sample the feeling of the group toward programming standards in general and Ada Programming Standards in particular.

In December, AdaSoft assisted Bob Nelson in organizing a working group to investigate the feasibility of producing Ada programming standards that might be used by new Ada projects at GSFC and perhaps used NASA-wide. The working group is comprised of 10 individuals that represent both GSFC and contractors that are working in Ada.

Paul Maresca attended the first meeting of this working group which was held on December 17. The primary issue discussed at the meeting was exactly what the objectives of the working group should be. Since Ada programming standards could conceivably address the entire software life cycle, some limited and hopefully achievable short-term goals had to be established for the group.

We were of the opinion that the group should address itself first to Ada coding and internal documentation (prologue and comments) standards. It was our feeling that GSFC and all of NASA should establish rigid standards in the area of coding and internal documentation for all delivered Ada code. This has two obvious benefits. First, programmers and programming teams do not waste time determining how their code should be formatted and documented and perhaps take time developing their own project standards. Second, all delivered code would look essentially the same, allowing programmers performing maintenance of Ada code which they did not write to feel "comfortable" with the code. Finally, every attempt should be made to make the prologues machine-readable. This would permit tools to be written to access these prologues to facilitate the reuseability of existing Ada code.

To support the rigid standards for delivered Ada code, we suggested that a software tool be developed to verify delivered source code. This could be supplied as GFE to all contractors developing Ada code for NASA. At the least, this tool could check prologues for valid content and format and reformat where possible; format the source code to the standard where necessary; and check for and report the use of non-transportable or implementation dependent features. Any additional tools constructed to reference the prologues could also be furnished as GFE where appropriate.

At the meeting on December 17, the group decided to collect all relevant existing standards and bring them to the next meeting. It was intended that these existing standards be used as a basis for the work of the Ada programming standards group.

During January, Paul Maresca attended the sessions of the Ada Programming Standards Working Group. At the first meeting of this group in January, he provided the members with copies of an article from the January, February 1986 Ada LETTERS that addressed Ada coding standards. Several additional sources of Ada programming standards and styles were presented at this meeting. As in previous meetings, the question of exactly what this group should produce was discussed.

We stated the opinion that the first product of the working group should address coding and internal documentation in a clear, concise and rigid manner. Each point in this document must be well thought out and clearly described. It must be accompanied by examples and motivation (i.e., an explanation of why this particular style element was adopted, what others were considered and why they were rejected). Collectively, the motivation for choosing a particular style could be treated as a separate rationale document. We believed that without a rationale for specifying a particular style, a list of rules and/or guidelines will be dismissed as another attempt to curtail programmer creativity. In fact, to "sell" this type of standard to programmers, it would be necessary to convince them that trying to evolve their own individual style for coding and internal documentation is a waste of time, effort and creativity that could be better applied to program design, implementation and testing. In general, programmers or even projects evolving their own style leads to loss of productivity during the development phase and the resultant inconsistent styles ultimately lead to lost productivity during the maintenance phase.

Also, at the first January meeting, Ed Seidewitz of GSFC presented an outline of a programming standards guide. The second meeting in January involved reviewing this guide. We reviewed this guide and voiced our opinion that it covered issues of design and language usage that were beyond what should be initially produced by the working group. However, this guide was a reasonable place to begin the process of generating an Ada programming style guide. As indicated previously, we felt that any style guide must have examples and motivation for each specified element of style. In addition, much of the wording in the original guide was either vague or too "high-tech." Each style element must be atomic and must be defined in a manner that any programmer can understand.

Paul Maresca continued attending the Ada Programming Standards Working Group meetings. A draft of the programming standards guide prepared by Ed Seidewitz was entered into the VAX and we obtained printed copies of this draft. We reviewed this draft and noted modifications where we felt they were necessary.

Although our contract has ended, the work of the Ada Programming Standards Group is by no means finished. We plan to continue attending their meetings and assisting in the development of a NASA/GSFC Ada programming standard.

2.5 Acquisition of NOSC Tools

In early January we responded to an announcement in the Ada Information Clearinghouse Newsletter indicating the availability of the Ada NOSC tools on magnetic tape from White Sands Missile Range. We obtained two magnetic tapes from Bob Nelson and sent these tapes to White Sands Missile Range to obtain copies of the Ada NOSC tool set that was developed under Government contract to NOSC. There are many different types of tools included in this set, some of which may be of use to GSFC. We intended to load these tools onto the 520 VAX and study them with the goal of determining what tools might be useful for work being done at GSFC.

We requested that the software be written on the tapes in a format readable on a DEC VAX. We were initially assured that the software would be sent to us quickly. However, as of the end of February, we still had not secured the software. We made several inquiries during the month to WSMR concerning the whereabouts of the tapes. After the second inquiry in early February, we were told that the tapes were being mailed and would arrive within the week. When they did not arrive, we made further inquiries and finally determined that WSMR was having problems writing the software on tape in a format readable by the DEC VAX under the VMS operating system. They had no projected date for the resolution of this problem. We could have obtained the software in a format readable under the UNIX operating system, but we knew of no such system at GSFC that would be available to us.

Although AdaSoft continued attempting to obtain a copy of the NOSC tools from White Sands Missile Range, we had not received them prior to the end of the contract. However, we learned that the University of Maryland had obtained the NOSC tools and had them available on a VAX running under the UNIX operating system. They indicated that they had the capability of writing tapes in a format that could be read on a VAX running the VMS operating system. We obtained a directory listing of the files containing the NOSC tools that were resident on the Maryland computer. We contacted Beth Katz of the University of Maryland and visited her at the university to obtain prologue listings for certain tools that we felt would be of interest to GSFC. The prologues give a brief description of what each of the tools do.

Prior to the end of the contract, AdaSoft had not actually obtained any of the NOSC tools in VAX VMS readable format from the University of Maryland. This is an effort that could be pursued by Government personnel as time permits since there does appear to be some worthwhile software available in the tool set.

2.6 Alsys Ada Compiler Evaluation

In late March of 1986, AdaSoft purchased the new Alsys Ada compiler for the IBM PC AT. We obtained a pre-validated version of the compiler and all of the information contained in this report is based on the use of that compiler. The compiler was validated in late April, but we had not yet obtained a copy of the validated version. We, at AdaSoft, believe that the availability of a validated Ada compiler on a personal computer is a major milestone in the maturation of the language since it makes the language accessible to a much greater number of programmers.

In conjunction with Bob Nelson, it was determined that it would be useful and informative to perform an evaluation of the new Alsys Ada compiler. Dan Roy of Century Computing had just completed work on an Ada compiler evaluation suite. He had run the suite on both the DEC Ada Compilation System (ACS), and the DG/ROLM Ada Development Environment (ADE) and had prepared a report comparing the two systems. With the approval of Bob Nelson, Dan Roy and Paul Maresca selected a subset of the evaluation suite to run on the Alsys compiler. The programs in the subset were transmitted from the DEC VAX to an IBM PC AT and compiled using the Alsys Ada compiler. Some minor problems were detected during compilation and were easily corrected. Most of these problems were caused by a difference in the size of the predefined type INTEGER between the Alsys compiler and the other compilers tested. Objects of type INTEGER use 16 bits on the Alsys compiler whereas they use 32 bits on the DEC ACS and DG ADE.

On April 15, Paul Maresca and Dan Roy presented the results of the Alsys compiler evaluation to the GSFC Ada Users Group. We also assisted Alsys personnel in demonstrating the Alsys compiler at that meeting. AdaSoft has prepared a separate report concerning the evaluation that contains not only results from the evaluation suite but a discussion of the facilities provided within the Alsys Ada environment.

2.7 Satellite Servicing Project Support

During April and May of 1986, AdaSoft provided assistance to the Satellite Servicing Project (SSP) at GSFC. This group is procuring the Alsys Ada compiler. AdaSoft had been shipped an extra copy of the compiler and was asked by Alsys to deliver that copy to GSFC until their copy was delivered. Bob Nelson provided the extra copy of the compiler to Barbara Scott of the SSP and Paul Maresca of AdaSoft installed the compiler on her IBM PC at GSFC.

The reason that the SSP is procuring the Alsys Ada compiler is that they are planning to obtain the Operations And Science Instrument Support Workstation (OASIS) system from the University of Colorado where it was developed and rehost it to the IBM PC AT using the Alsys Ada compiler. The development of the OASIS

system was funded by NASA's Office of Space Science and Applications so it should be available to GSFC at no cost. OASIS is written in Ada and is being developed on DEC VAX computers using the DEC Ada compiler. It can be used to operate and test a simulated or actual scientific instrument.

Since AdaSoft had just completed rehosting approximately 25,000 lines of Ada code from the DEC Ada compiler to the Alsys compiler, we met with Barbara Scott to discuss the rehosting of OASIS. She indicated that they had a meeting scheduled with representatives from the University of Colorado to discuss the rehosting. She asked us to prepare a list of questions for the meeting which we did. On April 23, we attended the meeting, discussed the rehosting and asked the questions that we had prepared. Barbara asked us to prepare a memo containing those questions and any additional questions we had so she could forward the memo to the University of Colorado to obtain complete answers. We prepared the memo as requested and submitted it to Barbara for forwarding.

During the meeting with the developers of OASIS, it became evident that some of the potential rehosting problems might arise from limitations imposed by the Alsys environment or from certain Ada language facilities that were not yet implemented in the Alsys Ada compiler. We offered to attempt to obtain answers to the questions concerning the Alsys implementation and report our findings to Barbara. She accepted our offer and we have discussed the potential problem areas with Alsys and have presented her with our findings.

2.8 Miscellaneous Support

During June of 1985, AdaSoft followed-up on STL System Performance Reports submitted by users concerning problems encountered using the beta test site version of the DEC Ada compiler. When version 1.0-7 of the DEC compiler was installed at GSFC, AdaSoft attempted to recreate the situations that caused the problems to determine whether they still existed in the new version.

At the October meeting of the GSFC Ada Users Group, we reported the results obtained from running Whetstone benchmark programs written in Ada and Fortran. These programs were compiled using the DEC Ada compiler and the DEC Fortran compiler respectively and were run on the DEC VAX-11/780.

During October, Bob Nelson was asked by Joel Wakeland of Bay St. Louis to provide some benchmark programs to be used in the procurement of an Ada environment for the development of the Space Station DMS Payload Simulator. Because the Ada Whetstone benchmark programs that we had used tested only Fortran-like computational capabilities, AdaSoft modified some code that we had developed so that it could be used as a benchmark test program. It contained additional Ada facilities such as dynamic

memory allocation, the use of unconstrained types, multi-tasking with the use of dynamic task creation and removal and exception handling.

On November 8, AdaSoft attended the GRO project PDR. Our opinion of the PDR was that it was well done. All of the technical areas were covered at the level expected in a PDR and some were presented in considerably more detail than might be expected.

Also in November, Paul Maresca participated in a meeting with Tom Rennie of GSFC concerning Ada Training for Goddard. Also present at this meeting were Bob Nelson, Bob Murphy and a colleague of Mr. Rennie. Several approaches to obtaining training for GSFC personnel and contractors were discussed. Also, the need for making GSFC management aware of what is happening in Ada, what will be happening in the future, and what actions should be taken to ensure that GSFC is prepared were discussed.

On March 11, Paul Maresca of AdaSoft attended the Critical Design Review for the Ada version of the GRO Dynamic Simulator. In our opinion, this was a very good CDR. It presented the operational aspects of the simulator in sufficient detail so that potential users could understand how various inputs cause the system to perform its designated functions and generate the desired results. Also, the system design was described in enough depth that it was clear that the designers understood the problem. One thing that was evident during the presentation was that the lack of knowledge of Ada and the Object Oriented Design methodology by most of the attendees caused some difficulties in understanding the design as presented. Hopefully, this problem will be alleviated as more projects are implemented in the Ada language using design methodologies that are suited to the features of the language.

3.0 PROJECT EVALUATION

As illustrated in the preceding section, AdaSoft provided support in many different areas during the period of performance of this project. It was gratifying to be able to assist some of these projects in their use of Ada. However, it was disappointing not to see more projects adopting the Ada language. Perhaps with the selection of Ada for the Space Station, more managers will feel comfortable with the use of Ada. It should be noted that the acceptance of Ada at GSFC mirrors its slower-than-expected acceptance by other organizations. There seems to be three major reasons for the slowness with which Ada is being accepted: the lack of production quality Ada compilers; the scarcity of well-trained Ada programmers; and the idea that Ada is "just another language."

The first problem - the lack of production quality compilers - is being resolved to some extent with the introduction of the DEC, Verdix and Alsys compilers as well as several others. These

compilers are generating code whose run-time efficiency is comparable to that of other higher-order languages. However, the compilation speed of the current Ada compilers is well below that of compilers for other languages which reduces programmer productivity. Also, we know of no validated Ada compiler that currently compiles the entire language.

We currently see no imminent solution to the scarcity of well-trained Ada programmers. There are many efforts underway to address this problem with perhaps as many proposed solutions as there are efforts. There appears to be no organized and well-directed effort either within GSFC or outside of it to attack this problem. There are short courses; there are long courses; there are hands-on courses; there are lecture-only courses; and there are Computer Assisted Instruction Courses (CAI). We feel that a mixture of these approaches applied in the correct order is what is required to train Ada programmers. We believe that Ada training should begin with a CAI course that covers all features of the language to at least an introductory level. The reason for using a CAI course first is that it is self-paced. The student can use the course whenever time permits; can move through it at a comfortable pace and can review concepts whenever necessary. Once the CAI course has been completed, a three to five day lecture course should follow that would provide an overview of Ada by a knowledgeable instructor. The student would be in a much better position to ask intelligent and informed questions after having completed the CAI course. Following the short lecture course, the student should take a six to eight week course that would include both lectures and hands-on use of the Ada language. The lectures could be held two or three times a week with at least one major project to be implemented in Ada during the course. At this point, the student would be prepared to serve as a non-lead member of an Ada programming team.

The final problem, and perhaps the most difficult to overcome, is the attitude taken by some programmers and many managers that Ada is just another programming language that can be "picked-up" in a week or so whenever needed. This attitude has caused many Government contractors to postpone Ada training until it is required to either bid or actually perform on a Government contract. This causes Government management to see a lack of experienced Ada programmers with which to staff their projects and makes them reluctant to choose Ada as their implementation language.

Certainly Ada is a programming language, but it is a complex and powerful language that requires both training and experience to use effectively. Perhaps more important than the language itself are the concepts upon which it is based. The concepts of incapsulation, abstraction, packaging, modularity, reuseability and maintainability. These concepts are forming the basis for much of the ongoing research in software engineering. Also, important work is being done in design methodologies and Program Design Languages that support and facilitate the use of Ada. It

is important for programmers not only to be knowledgeable in the Ada language, but to be familiar with the efforts that center around Ada to produce higher quality software.